POZNAN UNIVERSITY OF TECHNOLOGY

Efficient problem representations and computational models for 3D laser-based simultaneous localization and mapping

by

Krzysztof Ćwian

in the
Institute of Robotics and Machine Intelligence
Faculty of Control, Robotics and Electrical Engineering

Supervisor: Prof. Piotr Skrzypczyński, Ph.D., D.Sc.
Auxilary supervisor: Michał Nowicki, Ph.D., D.Sc.

May 6, 2025

Abstract

Autonomous robots depend on precise localization and an accurate environmental model for efficient operation. This is why Simultaneous Localization and Mapping (SLAM) is a crucial area of research in robotics, enabling autonomous systems to navigate and understand their environments without prior knowledge. This dissertation focuses on LiDAR-based SLAM, emphasizing the role of different map representations and their impact on localization accuracy.

It introduces two distinct approaches, resulting in a scalable mapping solution that also enhances the robustness of SLAM in diverse environments. The first one is based on a feature-based structure with planar and linear features, while the second utilizes surfels that are optimized through bundle adjustment along with the poses. The outcomes of the research demonstrate how these representations influence the SLAM performance by improving data association, reducing redundancy, and enhancing long-term map consistency. It also investigates the role of map structure in SLAM efficiency, analyzing how feature selection impacts computational requirements.

Moreover, the dissertation presents a method for incorporating data from the Global Navigation Satellite System (GNSS) into the SLAM framework using a factor graph optimization approach. Although LiDAR-based SLAM achieves high local accuracy, integrating GNSS data improves overall reliability and reduce drift, especially in large-scale environments where long trajectories are involved. The proposed GNSS-augmented SLAM system incorporates raw pseudorange and Doppler shift measurements, effectively reducing accumulated errors and enhancing global positioning accuracy.

Beyond theoretical contributions, this research validates its findings through real-world tests. Experimental evaluations demonstrate that the proposed map representations contribute to more accurate and reliable localization, particularly in structure-rich environments. In addition, optimized map structure significantly improves its quality in terms of both qualitative and quantitative metrics. The scalability of the developed systems is validated on publicly available, large-scale datasets, highlighting its effectiveness in processing long sequences. Furthermore, conducted evaluation confirms that the fusion of GNSS and LiDAR data improves trajectory estimation, providing robust localization in various environments where LiDAR-based methods alone may face difficulties due to feature sparsity.

Moreover, two real-world experiments were conducted in urban transportation and agricultural applications. The deployment of a GNSS-based localization system for electric buses demonstrates its effectiveness in the metropolitan area, highlighting the feasibility of integrating advanced localization into public transportation. Agricultural field tests show that the combination of GNSS with visual odometry enhances localization reliability, particularly in environments with inconsistent availability of the satellite signal. These practical implementations reinforce the importance of multi-sensor fusion in addressing real-world challenges.

In conclusion, the results presented in this dissertation underscore the importance of LiDAR-based SLAM, the impact of map representations on localization accuracy, and the benefits of integrating GNSS data for large-scale mapping. By advancing SLAM methodologies through multi-sensor fusion and optimization techniques, this research establishes the basis for more reliable and efficient autonomous systems across a wide range of applications. The dissertation also explores potential future advances in SLAM methodologies that can be implemented to further refine its performance.

Streszczenie

Autonomiczne roboty wymagają precyzyjnej lokalizacji oraz dokładnego modelu środowiska aby działać efektywnie. Z tego względu systemy Simultaneous Localization and Mapping (SLAM) stanowią kluczowy obszar badań w robotyce, umożliwiając autonomiczną nawigację oraz zrozumienie otoczenia bez wcześniejszej wiedzy na jego temat. Niniejsza rozprawa koncentruje się na systemach SLAM wykorzystujących dane z sensora LiDAR, podkreślając rolę różnych reprezentacji map oraz ich wpływ na dokładność dostarczanej lokalizacji.

W pracy zaprezentowano dwa różne podejścia do budowania mapy, prowadzące do opracowania skalowalnego rozwiązania zwiększającego dokładność systemu SLAM w różnorodnych środowiskach. Pierwsze z nich bazuje na strukturze cech z wykorzystaniem elementów płaszczyznowych i liniowych, natomiast drugie oparte jest na reprezentacjach surfelowych, optymalizowanych razem z trajektorią za pomocą metody bundle adjustment. Przeprowadzone badania pokazują jak wykorzystane reprezentacje wpływają na działanie systemu SLAM poprzez poprawę dopasowania danych, redukcję liczby punktów oraz zwiększenie spójności mapy. Rozpatrzony został również wpływ doboru cech w strukturze mapy na wydajność obliczeniową i efektywność systemu.

Dodatkowo rozprawa prezentuje metodę integracji danych z globalnego systemu nawigacji satelitarnej (GNSS) z systemem SLAM przy użyciu optymalizacji grafu ograniczeń. Mimo że systemy LiDAR SLAM zapewniają wysoką dokładność w lokalnym otoczeniu, integracja danych GNSS pomaga zwiększyć niezawodność oraz ograniczyć dryf w przypadku długich trajektorii. Zaproponowane rozwiązanie wykorzystuje surowe pomiary pseudoodległości i przesunięcia Dopplera, skutecznie poprawiając globalną dokładność lokalizacji.

Skuteczność opracowanych metod została zweryfikowana w rzeczywistych eksperymentach, które pokazują, że zaproponowane reprezentacje map przyczyniają się do dokładniejszej i bardziej niezawodnej lokalizacji. Ponadto optymalizacja struktury mapy znacząco poprawia jej jakość zarówno pod względem wizualnym, jak i numerycznej wartości błędu. Skalowalność opracowanych systemów została zweryfikowana na publicznie dostępnych zbiorach danych. Dodatkowo, przeprowadzona ewaluacja potwierdza, że fuzja danych GNSS oraz LiDAR zapewnia poprawę lokalizacji w środowiskach, gdzie metody korzystające wyłącznie z sensora LiDAR mają trudności.

W pracy przedstawiono również wyniki eksperymentów dotyczących systemów lokalizacji dla elektrycznych autobusów w transporcie miejskim oraz maszyn w rolnictwie, przeprowadzonych w rzeczywistych warunkach oraz w kontekście ich praktycznego zastosowania. Implementacja metody lokalizacji autobusu wykazała jego skuteczność w zurbanizowanym środowisku, podkreślając tym samym możliwość jego integracji z infrastrukturą transportu publicznego. Dodatkowo, testy lokalizacji maszyny rolniczej w warunkach polowych pokazały, że połączenie danych GNSS z odometrią wizyjną poprawia niezawodność lokalizacji, szczególnie w środowiskach w których występują zakłócenia sygnału satelitarnego. Testy te podkreślają znaczenie fuzji danych z wielu sensorów w przypadku systemów, których zadaniem jest praca w rzeczywistych warunkach.

Podsumowując, wyniki przedstawione w niniejszej rozprawie podkreślają znaczenie systemów SLAM, wpływ reprezentacji mapy na dokładność lokalizacji oraz korzyści płynące z integracji danych GNSS, które poprawiają dokładność estymowanej trajektorii. Opracowane rozwiązania wykorzystują fuzję wielosensoryczną realizowaną poprzez optymalizację grafu ograniczeń oraz stanowią podstawę do dalszego rozwoju bardziej efektywnych metod. W rozprawie zawarto również potencjalne przyszłe prace, mające na celu dalszą poprawę skuteczności systemów SLAM.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor, Prof. Piotr Skrzypczyński, for his support and scientific guidance. His insightful suggestions and constant availability were invaluable for my research. I also want to thank my co-supervisor, Prof. Michał Nowicki, for his expertise and helpful feedback that contributed greatly to this thesis.

Moreover, I would like to extend my appreciation to Prof. Giorgio Grisetti for the possibility to complete a three-month internship at Sapienza University of Rome. This experience provided me with the opportunity to collaborate with leading experts in robotics and enhance my technical and professional skills. I am also very grateful to all my colleagues at Sapienza University for the inspiring discussions, collaborative spirit, and shared commitment during my visit.

Last but not least, I would like to extend my appreciation to my family. Their support and encouragement have been a foundational strength for me during this journey.

Abbreviations

ADAS Advanced Driver Assistance System

ATE Absolute Trajectory Error

BA Bundle Adjustment

CAN Controller Area Network

DCC Dajeon Convention Center

DOF Degrees of Freedom

ECEF Earth-centered Earth-fixed

EKF Extended Kalman Filter

GALS GNSS-Augmented LiDAR SLAM

GNSS Global Navigation Satellite Systems

GPGPU General Purpose Graphic Processing Unit

GPS Global Positioning System

GUI Graphical User Interface

HMI Human-Machine Interface

ICP Iterative Closest Point

ILS Iterative Least Squares

IMU Inertial Measurement Unit

INS Inertial Navigation System

LCD Liquid Crystal Display

LiDAR Light Detection and Ranging

LM Levenberg-Marquardt

LOAM Lidar Odometry and Mapping

LTE Long Term Evolution

NC Newer College

NDT Normal Distributions Transform

NLOS Non-Line of Sight

NLS Nonlinear Least Squares

NTRIP Networked Transport of RTCM via Internet Protocol

ORB Oriented FAST and Rotated BRIEF

PCA Principal Component Analysis

PGO Pose Graph Optimization

PPP Precise Point Positioning

RAIM Receiver Autonomous Integrity Monitoring

RINEX Receiver Independent Exchange System

RMS Root Mean Square

ROS Robot Operating System

RPE Relative Trajectory Error

RTCM Radio Technical Commission for Maritime

RTK Real Time Kinematic

SDF Signed Distance Function

SfM Structure from Motion

SLAM Simultaneous Localization and Mapping

TST Tsim Sha Tsui

UKF Unscented Kalman Filter

URA User Range Accuracy

UTM Universal Transverse Mercator

VBR Vision Benchmark in Rome

VO Visual Odometry

WGS-84 World Geodetic System '84

Notation

$\mathbb{R}$	the set of real numbers
$\mathcal{K}$	the set of sensor poses
$\mathcal{F}_\pi$	the set of planar features
$\mathcal{F}_\lambda$	the set of linear features
$\mathcal{C}_i$	the i-th input point cloud
$\mathcal{T}_i$	the i-th kd-tree
$\mathcal{S}$	the set of surfels
$\mathcal{S}_i$	the set of surfels associated with measurements from i-th pose
$\mathcal{S}^*$	the set of optimal surfels
$\mathcal{A}_s$	the set of leaves associatied with the surfel s
$\mathcal{Q}$	the reconstructed map
$\mathcal{R}$	the reference map
$\mathcal{M}$	the set of graph nodes associated with the LiDAR SLAM constraints
$\mathcal{G}$	the set of graph nodes associated with the GNSS constraints
$\mathcal{L}$	the set of graph nodes associated with loop closure constraints
$\mathcal{O}$	the set of graph nodes associated with Visual Odometry constraints
$\mathcal{N}$	the set of all observed satellites
E	the total negative log-likelihood function
e	the residual error function
f	the point-to-plane error function
g	the point-to-line error function
h	the pose-to-pose error function
π	the planar feature
λ	the linear feature
l	the leaf of a kd-tree
s	the surfel
δ_T	the pose update
$\mathbf{T}_\mathbf{i}^*$	the optimal i-th pose estimate
$\mathbf{T}_i$	the i-th pose estimate

$\mathbf{T}_s$	the transformation from the beginning to the end of the scan
$\mathbf{T}_i^S$	the i-th pose estimated using LiDAR SLAM
$\mathbf{T}_j^L$	the j-th pose estimated using loop closure
$\mathbf{T}_i^G$	the i-th pose estimated using GNSS
$\mathbf{T}_i^O$	the i-th pose estimated using Visual Odometry
$\mathbf{R}$	the rotation matrix
$\mathbf{R}_i$	the rotation matrix of the i-th pose
$\mathbf{X}$	the state vectors for all nodes in the graph
$\mathbf{X}_i$	the state vector for i-th node in the graph
$\mathbf{\Sigma}_S$	the covariance matrix for LiDAR SLAM constraints
$\mathbf{\Sigma}_L$	the covariance matrix for loop closure constraints
$\mathbf{\Omega}$	the information matrix
$\mathbf{\Omega}_D$	the information matrix for Doppler shift constraints
$\mathbf{\Omega}_S$	the information matrix for LiDAR SLAM constraints
$\mathbf{\Omega}_L$	the information matrix for loop closure constraints
$\mathbf{\Omega}_G$	the information matrix for GNSS positioning constraints
$\mathbf{\Omega}_O$	the information matrix for Visual Odometry constraints
$\mathbf{\Omega}_P$	the information matrix for the pseudorange constraints
$\mathbf{W}$	the eigenvector matrix
$\mathbf{t}_i$	the translation vector of the i-th pose
$\mathbf{r}_i^r$	the i-th position of a receiver
$\mathbf{v}_i^r$	the i-th velocity of a receiver
$\mathbf{r}_n^s$	the position of the n-th satellite
$\mathbf{p}$	the point in 3D space
$\mathbf{p}_i$	the i-th point in 3D space
$\mathbf{p}_s$	the center point of a surfel
$\mathbf{p}_l$	the center point of a leaf
$\mathbf{p}_q$	the point in a reconstructed map
$\mathbf{p}_r$	the point in a reference map
$\mathbf{n}_\pi$	the normal vector of a planar feature
$\mathbf{n_l}$	the normal vector of a leaf
$\mathbf{n_s}$	the normal vector of a surfel
$\mathbf{l}_d$	the direction of the line in Plücker coordinates
$\mathbf{l}_m$	the moment of a line in Plücker coordinates
$\mathbf{w}$	the eigenvector
$\boldsymbol{\omega}$	the coordinates in the tangent plane of the 2-sphere S^2
$\mathbf{e}_{i,i+1}^D$	the error for Doppler shift constraints between graph nodes
$\mathbf{e}_{i,i+1}^S$	the error for LiDAR SLAM constraints between graph nodes

$\mathbf{e}_{i,j}^{L}$	the error for loop closure constraints between graph nodes
$\mathbf{e}_{i,i+1}^{G}$	the error for GNSS constraints between graph nodes
$\mathbf{e}_{i,i+1}^{O}$	the error for Visual Odometry constraints between graph nodes
e_i^p	the error for pseudorange constraints for i-th graph node
ε	the mean residual error
ϕ_i	the horizontal angle of the i-th LiDAR measurement
ϕ_s	the horizontal angle of the LiDAR measurement at the beginning of a scan
ϕ_e	the horizontal angle of the LiDAR measurement at the end of a scan
i	the index of the measurement
l_i	the index of i-th LiDAR scanning ring
d_π	the distance from the planar feature to the origin
d_e	the Euclidean distance between two points
d_n	the point-to-plane or point-to-line distance
p_f	planarity or linearity coefficients of a feature
α	the angle between two vectors
θ	the angle of the normal vector on the 2-sphere S^2
m_l	the magnitude of the moment vector $\mathbf{l}_m$
r_s	the radius of a surfel
q_s	the displacement of surfel along its normal
r_i	the range of the i-th simulated LiDAR measurement
N_l	the number of simulated LiDAR measurements
N_f	the number of points in a given feature
N_q	the number of points in the reconstructed map
N_r	the number of points in the reference map.
$b_{\max}$	the maximum size of the leaf
$b_{\min}$	the minimum flatness of a leaf
ρ_{Huber}	the Huber loss function
ρ_{ker}	the threshold for Huber robust estimator
$d_{\text{C-L1}}$	the Chamfer-L1 distance
t_i	the time of i-th measurement
$t_{i,i+1}$	the time interval between the i-th and $(i+1)$-th pose
t_s	the duration of a scan
$p_{i,n}$	the pseudorange measurement between the i-th pose and n-th satellite
$\rho_{i,n}$	the geometric range between the i-th pose and the n-th satellite
δ_n^s	the clock bias of n-th satellite
δ_i^r	the receiver clock bias for the i-th state vector
$\delta_{i,1\ldots4}^r$	the receiver clock bias for the i-th state vector and GNSS constellation
$d_{i,n}^{ion}$	the ionospheric delay for the i-th pose and n-th satellite

$d_{i,n}^{trop}$	the tropospheric delay for the i-th pose and n-th satellite
$\epsilon_{i,n}^{p}$	the pseudorange measurement error for the i-th pose and n-th satellite
f_n	the carrier wave frequency for the n-th satellite
Δf_n	the Doppler shift value for the n-th satellite
θ_{el}	the elevation angle of the satellite
θ_{az}	the azimuth angle of the satellite
$r_{x,i}^{r}, r_{y,i}^{r}, r_{z,i}^{r}$	the x, y, z components of the i-th position of the receiver
$r_{x,n}^{s}, r_{y,n}^{s}, r_{z,n}^{s}$	the x, y, z components of the n-th satellite position
v_n	the radial velocity for the n-th satellite
v_i^r	the i-th velocity of the receiver
$v_{x,i}^{r}, v_{y,i}^{r}, v_{z,i}^{r}$	the x, y, z components of the i-th velocity of the receiver
$v_i^{r_G}$	the i-th velocity of the receiver calculated using pseudorange measurements
$v_i^{r_D}$	the i-th velocity of the receiver calculated using Doppler shift measurements
$v_i^{r_S}$	the i-th velocity of the receiver calculated using successive SLAM poses
σ_{ATE}	the standard deviation of the Absolute Trajectory Error
σ_l	the standard deviation of the LiDAR measurement
σ_p^2	the total variance of the pseudorange error
σ_m^2	the variance of pseudorange measurement
σ_e^2	the variance of ephemeris measurement
σ_c^2	the variance of code bias measurement
σ_{ion}^2	the variance of ionospheric delay measurement
σ_{trop}^2	the variance of tropospheric delay measurement
σ_x^2, σ_y^2, σ_z^2	the variance of the pose along the x, y, z axes
σ_ϕ^2, σ_ψ^2, σ_θ^2	the variance of the pose along the roll, pitch, yaw axes
c	the speed of light
ω_e	the angular velocity of the Earth
f_{L1}	the $L1$ carrier wave frequency
$l_0, \ldots, l_6$	the GNSS measurement error factors

Contents

Chapter 1

Introduction

1.1 Motivation

Autonomous robots require accurate localization and a reliable environment model to operate efficiently. These factors are essential for effective task and motion planning, understanding scene context, and facilitating human-robot collaboration. In unfamiliar environments, a robot must address the problem of Simultaneous Localization and Mapping (SLAM), a fundamental challenge in robotics that enables it to construct a map while simultaneously determining its pose within this map.

In recent years, extensive research has been conducted on SLAM and related topics, covering various approaches, including Visual Odometry (VO) and sensor fusion techniques [1, 2]. While simpler SLAM problems, such as those in structured 2D environments, are considered to be largely solved, more complex and realistic scenarios remain challenging. Issues such as real-time mapping in large-scale 3D environments, adaptation to dynamic surroundings, and continuous map updates over an extended period of time are still active areas of study [3]. In addition, improving robustness against sensor noise, computational efficiency, and scalability are key challenges in modern SLAM research.

Rapid adoption of autonomous systems across diverse sectors, such as autonomous vehicles, robotic exploration, drone delivery, augmented reality, logistics, agriculture, and disaster response, has created an urgent need for robust and scalable localization and mapping solutions [4, 5]. These applications rely heavily on precise positioning to perform their tasks safely and efficiently. For example, self-driving cars require real-time, high-accuracy localization to navigate dynamic urban environments, delivery drones depend on reliable positioning to reach their destinations, and agricultural machinery requires it to achieve high precision in agro-technical treatments. Similarly, in industrial settings, mobile robots used for warehouse automation need accurate information about their pose to optimize inventory management [6]. As these applications continue to expand, there is an increasing need for advanced localization solutions capable of working in complex scenarios.

However, many of these applications operate in challenging unstructured environments where traditional SLAM approaches often struggle. Autonomous vehicles, for instance, must navigate dynamic urban settings filled with moving objects such as pedestrians and vehicles, while robotic systems deployed in disaster zones or planetary exploration face featureless or highly variable terrains. These environments pose significant challenges for SLAM systems, requiring them to be not only accurate, but also robust. Furthermore, urban environments present unique additional difficulties in localization, especially in areas with tall buildings, tunnels, or underground parking garages, where the Global Navigation Satellite Systems (GNSS) signals are unreliable or entirely unavailable. Therefore, SLAM solutions often provide precise 3D trajectory estimation using multiple sensors available on-board. Innovations in multi-sensor integration have significantly enhanced SLAM capabilities, improving the robustness of navigation in complex and dynamic environments [4].

Recent advances in sensing technologies, computational power, and machine learning have opened new possibilities for mobile robotics and localization algorithms. For example, the development of high-resolution Light Detection and Ranging (LiDAR) sensors has significantly improved the accuracy and resilience of localization systems [7]. In addition, innovations in laser technology, such as solid-state and flash LiDARs, have enhanced performance while reducing cost and size. These improvements make it more accessible for various robotic applications. Unlike camera-based SLAM, which relies on visual features that can be obstructed or lost in poor lighting conditions, modern LiDARs provide high-precision 3D depth information by measuring the time taken for laser pulses to reflect off objects. This ensures reliable localization even in challenging scenarios such as low-light, foggy, or featureless environments where solution based on cameras often struggle. Thus, LiDAR-based SLAM systems have emerged as a promising solution because of their ability to provide accurate 3D environmental data. These technological advances motivate researchers to explore new approaches to localization, driving innovation in mobile robotics. At the same time, breakthroughs in algorithms, such as probabilistic filtering and graph-based optimization, have also enabled more efficient and scalable solutions [8].

Despite their potential, many systems still face limitations in terms of scalability, computational efficiency, and robustness in real world scenarios [9]. Addressing these challenges is a key motivation for this research, as it can improve the performance of LiDAR-based SLAM, allowing autonomous systems to navigate in complex real-world environments with greater reliability. Furthermore, overcoming these limitations would enhance the capabilities of mobile robots, enabling them to perform a diverse range of tasks autonomously.

1.2 Problem statement

Although LiDAR-based SLAM has demonstrated strong performance in diverse applications due to its reliability in various environmental conditions, it still faces significant challenges in terms of accuracy, robustness, and scalability. Traditional methods, particularly those based on Iterative Closest Point (ICP), are susceptible to local minima and cumulative drift, leading to degraded performance in long-term navigation and large-scale environments. These limitations become

especially pronounced in feature-sparse or repetitive settings where incorrect data associations introduce localization errors.

Furthermore, the accuracy of SLAM is highly dependent on the representation of the map, the choice of features, and the strategies used for feature matching, as different approaches influence how well the system can establish reliable correspondences between point clouds. Most commonly, these approaches include using raw points, geometric primitives (planes, edges, and corners), intensity-based features, or learning-based descriptors. In addition, the selection of the map representation directly impacts computational efficiency and storage requirements. Many systems rely on storing dense point cloud data, which significantly increases memory usage and computational demands, particularly in large-scale scenarios [10]. Therefore, to enable long-term operation, it is essential to adopt efficient mapping strategies that store only the most relevant features for localization.

Moreover, many SLAM systems do not integrate additional sensory data, such as GNSS, which could provide absolute positioning constraints to improve accuracy and mitigate drift. However, naive fusion of GNSS with LiDAR data can introduce inconsistencies in the estimated trajectory due to degradation of the GNSS signal in urban canyons, tunnels, or areas with dense vegetation. Thus, a robust optimization framework is needed to effectively integrate GNSS measurements with LiDAR-based motion and minimize the uncertainty of pose estimation. The tight integration of GNSS and SLAM can address these challenges by enabling error correction, reducing drift, and improving global consistency. The structure of the optimization problem itself, including how constraints are defined and weighted, also has a significant impact on achieving accurate and reliable localization. Recent advances in graph-based optimization and factor graph SLAM offer promising solutions to this problem, but require further exploration in the context of LiDAR-based systems.

Another challenge is the practical implementation of SLAM for modern LiDAR sensors, which provide precise, high-resolution point cloud data at significantly faster rates than previous generations. Existing SLAM software frameworks may not be optimized to handle the full potential of these sensors, which requires new data processing pipelines and software architectures to maximize their effectiveness.

Taking into account these challenges related to LiDAR-based SLAM, the scientific thesis presented in this dissertation is formulated as follows:

It is possible to formulate the problem of estimating the motion of a LiDAR sensor as an optimization problem over the feature-based map representation, including constraints stemming from GNSS measurements, which leads to elimination of the local minima problems commonly observed for the ICP-based SLAM algorithms. This new formulation combined with an efficient map structure that stores only the data that are suitable for localization allows an implementation of SLAM that is scalable to long trajectories.

1.3 Content of the thesis

The dissertation is structured into seven chapters, each addressing a different aspect of the research. The remainder of the manuscript is structured as follows.

Chapter 2 provides an overview of the state of the art in SLAM, discussing various architectures, scan registration methods, and loop closure techniques. It also explores challenges related to robot localization and introduces factor graphs, highlighting their benefits, such as possibility to integrate data from multiple sources. It also examines methods to improve GNSS-based localization in the context of urban transport applications.

Chapter 3 presents two approaches to map representation. The first approach utilizes a feature-based structure with large planar and linear elements. It details the process of feature management, optimization strategies, and different factor graph structures. The second approach employs surfels combined with a bundle adjustment method to jointly optimize the map and trajectory. It also describes the process of generating surfels from raw scans, the data association technique, the uncertainty model for LiDAR measurements, and the approach to optimizing surfels and poses.

Chapter 4 focuses on the integration of GNSS measurements with SLAM to improve long-term accuracy. The first part discusses the fusion of raw GNSS measurements with LiDAR-based SLAM, including strategies for filtering invalid GNSS observations. The second part presents an approach for correcting the GNSS trajectory by integrating it with simple VO, effectively mitigating erroneous positioning errors.

Chapter 5 contains an experimental evaluation of the systems developed in Chapters 3 and 4. The evaluation includes tests with different LiDAR sensors and datasets, including self-recorded data, as well as comparisons with existing SLAM systems. It also analyzes the computational efficiency of the first proposed method. In addition, it includes an assessment of the refined maps by comparing it to a reference model. The chapter further demonstrates the effectiveness of integrating GNSS measurements to enhance trajectory estimation, comparing the results obtained with different SLAM systems and GNSS receivers.

Chapter 6 presents two practical applications of localization systems employing GNSS and their evaluation. The first case study involves the localization of a bus in urban environment. It shows a practical case-study of using GNSS alone for localizing over a large and diversified area, demonstrating both strengths and limitations of such an approach. The second case focuses on localization of an agricultural robot operating in field conditions, where the GNSS localization is improved using a VO system. This demonstrates that the problem formulation and software developed for the GNSS-Augmented LiDAR SLAM (GALS) system can be easily adopted to different sensory and scenario variants, addressing practical use-cases.

Chapter 7 concludes the dissertation by summarizing the key contributions of the research and outlining potential directions for future work.

1.4 Projects and publications

Author of the thesis participated as researcher in the following projects:

- "Advanced driver assistance system for precision maneuvers with single-body and articulated urban buses", funded by the National Centre for Research and Development under the agreement POIR.04.01.02-00- 0081/17-00, 10.2018–06.2021,
- "Robotization of the process of increasing the quality of hemp seed material", funded by the National Centre for Research and Development under the agreement POIR.01.01.01-00-2271/20, 04.2022–06.2023.

The author also received the following scholarship:

- NAWA STER Mobility – 3-month Ph.D. internship at Sapienza Univerity of Rome under supervision of Prof. Giorgio Grisetti, funded by the Polish National Agency for Academic Exchange under the STER programme "Towards Internationalization of Poznan University of Technology Doctoral School (2022-2024)", 02.2024–05.2024.

Results presented in this dissertation have already been partially published in the following journal articles:

1. K. Ćwian, M. R. Nowicki, J. Wietrzykowski, and P. Skrzypczyński, "Large-scale lidar slam with factor graph optimization on high-level geometric features," *Sensors*, vol. 21, no. 10, p. 3445, 2021. IF: 3.847.
2. I. Esfandiyar, K. Ćwian, M. R. Nowicki, and P. Skrzypczyński, "GNSS-Based driver assistance for charging electric city buses: Implementation and lessons learned from field testing," *Remote Sensing*, vol. 15, no. 11, p. 2938, 2023. IF: 4.2.
3. M. Nijak, P. Skrzypczyński, K. Ćwian, M. Zawada, S. Szymczyk, and J. Wojciechowski, "On the importance of precise positioning in robotised agriculture," *Remote Sensing*, vol. 16, no. 6, p. 985, 2024. IF: 4.2.
4. K. Ćwian, L. Di Giammarino, S. Ferrari, T. Ciarfuglia, G. Grisetti, and P. Skrzypczyński, "MAD-BA: 3D LiDAR bundle adjustment – from uncertainty modelling to structure optimization,", *IEEE Robotics and Automation Letters*, accepted on May 05, 2025.

The thesis also includes results presented during the following conference:

1. K. Ćwian, M. R. Nowicki, and P. Skrzypczyński, "GNSS-Augmented LiDAR SLAM for accurate vehicle localization in large scale urban environments," in *17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, (Singapore), pp. 701–708, 2022.

Chapter 2

Related Work

2.1 State of the art in LiDAR SLAM

2.1.1 Architectures of SLAM

Most of SLAM research over the past two decades has focused on passive visual perception [3], however, active 3D laser sensors, commonly known as LiDAR, offer significant practical advantages over passive cameras. Unlike cameras, which struggle in low-light conditions and with sudden changes in illumination, LiDAR operates effectively regardless of lighting. Although passive cameras remain the most cost-effective and easily integrated sensing option, LiDAR technology is increasingly recognized as a key component in the self-driving vehicle industry [11]. Beyond autonomous transportation, LiDAR is also essential for various robotic applications, such as underground inspections, where reliable perception in dark or unstructured environments is crucial [12].

In advanced visual SLAM, efficient handling of historical data in the environment map is essential for accuracy [13], as optimization is favored over filtering methods such as the Kalman filter or particle filter. By jointly optimizing the trajectory of the sensor (robot) and the map, the system can update the linearization points and mitigate errors caused by approximations and incorrect measurements through the use of robust cost functions [14]. Most LiDAR-based SLAM systems rely on raw point clouds for map representation. This approach is often practical, as it does not depend on specific environmental features and, when combined with loop closure detection and global pose-based optimization, can produce highly accurate maps [15]. However, real-time performance in such methods is typically achievable with massively parallel processing using a General Purpose Graphic Processing Unit (GPGPU) [16], or by employing techniques such as voxelization to increase computational efficiency. Additionally, raw point-based representations struggle to differentiate salient features, such as objects of particular classes or distinct geometric structures. This limitation makes it difficult to track these features across multiple scans in the way visual tracking does. Moreover, the lack of semantic information can lead to incorrect associations between scans due to the closest match strategy, ultimately reducing the number of

valid constraints available for map optimization. Several previous studies have explored the use of volumetric and surfel-based map representations for LiDAR data. In [17], a voxel grid representation with subsampled point clouds was employed, while in [18], the Normal Distributions Transform (NDT) was used for compact 3D map representation and scan-to-model registration. One limitation of grid-based map representations is their reliance on global nearest-neighbor searches to establish correspondences between scan points and the model, making these methods computationally inefficient.

Another approach is to employ surfels, which are small patches used to represent surfaces in 3D mapping. This representation gained popularity in SLAM with the success of the RGB-D-based ElasticFusion method [19], which directly influenced research on LiDAR-based systems [20]. Droeschel and Behnke [21] introduced a hybrid approach that combined a surfel representation with a multi-resolution, robot-centric local grid map. By maintaining higher resolution in areas closer to the LiDAR sensor, their method efficiently aggregated scanned points into surfels and incorporated local maps into a multi-level graph-based SLAM framework. Additionally, Behley and Stachniss [22] leveraged surfel-based mapping to establish projective data associations between LiDAR scans and rendered views of the map, allowing large-scale real-time operation. Similarly, recent methods, such as MAD-ICP [23], have shown that the use of surfels can improve the accuracy and the speed of registration by employing robust data association through kd-tree search. A recent work [24] introduces a triangle mesh-based representation for LiDAR mapping, which, unlike point clouds or surfels, offers a more structured and memory-efficient model of the environment. This method, however, is not employed for SLAM but rather for incremental mapping with known poses, enabling efficient updates and compact storage while preserving geometric accuracy.

Volumetric representations have also been utilized to address LiDAR-based SLAM problems using deep learning methods. However, standard LiDAR data representations, such as point clouds, are not differentiable [25]. To address this issue, state-of-the-art neural models for processing range data either ensure invariance to point-order permutation, such as PointNet [26], or adopt a voxel-based representation [27]. While PointNet and its extensions [28] enable point cloud processing through convolutional networks, they are generally considered too computationally intensive for real-time SLAM applications. Among the few attempts to use neural networks for LiDAR-based localization in unknown environments, Velas et al. [29] reformulated the pose estimation problem as a classification task. The L^3-Net system [30] integrates a lightweight variant of PointNet with another network that aggregates matching costs from all features within a volumetric model to produce pose estimates. However, formulating pose estimation as either a classification or cost-accumulation problem reduces accuracy since the results are constrained by the discrete structure of the data representation. The DeepLO method [31] has demonstrated pose estimation accuracy comparable to model-based systems by employing an unsupervised loss function derived from the iterative closest point formulation. It also leverages the normal vector consistency to generate a confidence map that assigns weights to the factors of the loss function. Despite the progress in learnable LiDAR SLAM through unsupervised learning, DeepLO had to be trained separately for each of the three datasets it was tested on. This highlights a fundamental challenge in fully learnable LiDAR-based SLAM, which is its potential difficulty in generalizing to previously unseen environments.

In most modern LiDAR-based odometry and SLAM systems, the map is either not optimized at all or optimized solely as a graph of past sensor poses. Raw point-based or volumetric representations do not support global optimization with historical state marginalization, which is essential for keeping the map size manageable and efficiently handling previous data. This is a key factor in achieving high accuracy in visual SLAM, where Bundle Adjustment (BA) optimization plays a crucial role [32]. In visual SLAM, the map is typically structured as a collection of images (direct methods) or a graph of sparse feature points [13]. Although there are 3D salient point feature detectors that can be applied to point clouds [33], their high computational cost makes them impractical for large-scale outdoor SLAM applications.

Few LiDAR-based SLAM systems attempt to structure the map beyond simple point representations. Lidar Odometry and Mapping (LOAM) [34], which is one of the most influential architectures in LiDAR-based localization, segments scans into semantically distinct planar patches and lines, applying different distance metrics to these categories during the ICP registration process. However, LOAM does not incorporate geometric features into the map itself. Instead, it represents the environment as a large point cloud organized on a regular grid to enhance the efficiency of data association. A key innovation of LOAM lies in its software architecture, which separates real-time scan-to-scan pose tracking (incremental odometry) from a slower but more accurate scan-to-map localization process. By selectively choosing points that represent stable and well-defined parts of the environment and refining incremental odometry through model-based registration, LOAM remains one of the most accurate real-time LiDAR-based localization systems. This success has inspired numerous researchers to refine aspects of LOAM while preserving its parallel odometry and mapping structure. The PlaneLOAM system [35], introduced in Section 3.2, follows this approach. Similarly, LeGO-LOAM [36] enhances LOAM by explicitly detecting the ground plane, distinguishing between ground and non-ground points, and using this information to determine sensor attitude (pitch and roll). Then it estimates the remaining components of the 6-Degrees of Freedom (DOF) sensor pose using a simplified model. LLOAM [37] extends LOAM by incorporating a loop closing module based on learned descriptors for point cloud segments [38] and constructing a pose graph using sensor poses and relative transformations from LOAM-like odometry. Furthermore, the system in [39] combines LOAM-style LiDAR odometry and mapping with SegMatch-based loop detection [38]. It introduces ground plane constraints to improve relative pose estimation, but continues to use a basic pose-graph structure for handling detected loop closures.

A natural approach to handling LiDAR data in partially structured urban environments is to utilize high-level geometric features, such as planar and linear segments. Weingarten and Siegwart [40] first incorporated planar features into Extended Kalman Filter (EKF)-based SLAM with 3D LiDAR data. Later, [41] demonstrated the use of planar segments within a pose-graph SLAM framework, utilizing a fast registration method for noisy planar features with unknown correspondences [42]. A method for fast and accurate extraction of planar surfaces from LiDAR data was proposed in [43] and applied in a filtering-based SLAM system. Grant et al. [44] further explored high-level features from LiDAR data in SLAM by leveraging a factor graph framework. Their SLAM method represents extracted planar features as landmarks in a factor graph optimization framework, while also incorporating LiDAR odometry factors between sensor poses to enhance performance in less structured environments. Loop closure is achieved by comparing

planar features stored in a global map, establishing constraints (factors) between similar features, and optimizing the graph.

2.1.2 Scan registration methods

The most common methods for estimating the relative transformation between two scans are variations of the ICP algorithm [45]. Although ICP is widely used and has undergone numerous refinements [46], it remains computationally expensive and vulnerable to data association errors. This is because each time the sensor pose is updated, the algorithm must re-establish point correspondences through a search process. One way to mitigate this issue is to carefully select points for matching. LOAM addresses this by applying point-to-plane and point-to-line distance metrics, associating points with planar and linear features, respectively [34]. Similarly, IMLS-SLAM [47] utilizes implicit moving least squares surfaces and selects points based on a criterion that leverages normal vectors to constrain all 6-DOF of the sensor. The study in [47] also highlights that retaining too many points that provide weak constraints, such as those from non-stationary objects or points with significant measurement noise, can degrade transformation accuracy.

Another challenge in registering consecutive LiDAR scans is compensating for motion-induced distortions in range measurements, similar to rolling shutter effects in cameras. These distortions occur when scans are captured while the sensor is in motion, which is a common scenario in autonomous vehicle applications. The registration method must be adapted to address this issue [48], or the scans must be undistorted using a motion prior. Some LiDAR SLAM systems mitigate these distortions by integrating tightly coupled Inertial Measurement Unit (IMU) data [49]. However, if external motion predictions are unavailable, corrections must be made while simultaneously estimating the sensor's movement. One approach to this problem, proposed in [21], models the sensor trajectory as a continuous B-spline in $SE(3)$ and interpolates between poses. In contrast, LOAM and LeGO-LOAM use a simpler strategy by assuming a constant sensor velocity, which simplifies interpolation [34, 36]. This assumption is particularly effective for ground vehicles with mostly planar motion and provides a practical solution for motion estimation between consecutive poses in LiDAR odometry.

2.1.3 Closing loops in SLAM

Despite the extensive research on LiDAR-based localization and mapping, relatively few published systems provide a complete SLAM solution, as many lack loop closure capabilities. This means that they cannot recognize previously visited areas and incorporate them into the map. Many existing approaches focus on LiDAR odometry with scan-to-model registration [34, 36, 47, 49]. While these methods optimize the sensor trajectory and incrementally build the map, they do not correct for trajectory drift by optimizing the map itself. This limitation is particularly problematic in structured urban environments, where the absence of map corrections can result in inconsistencies in object representation.

Although ICP registration can be used for loop closure by aligning new scans with the existing map [50], it becomes inefficient for large loops due to the high computational cost of nearest neighbor search. A more efficient alternative is appearance-based loop closure detection, provided that an effective data representation for LiDAR scans is implemented. Magnusson et al. [51] were among the first to explore this approach, using NDT and location histograms for loop detection. Droeschel and Behnke [21] leveraged local surfel-based maps to identify loop closures, while [22] improved this method by generating virtual views of the global map. This enabled projective associations that enhanced the robustness of detecting previously visited areas, even in cases of only partial overlap.

Recently, neural network-based models have proven effective in addressing the loop closure problem for LiDAR data. The SegMap system [52] represents the environment as point cloud segments, each with learned descriptors. It detects previously visited locations by matching a local segment map, built from recent LiDAR scans, to a global segment map. These segments are incrementally formed by accumulating LiDAR points in a dynamic voxel grid, while a deep neural network generates their descriptors [38]. The learned representation in SegMap is highly distinctive, enabling reliable matching of local and global maps even in cases of significant trajectory drift. This approach has already been integrated into LOAM-inspired SLAM systems [37, 39], where loop closure constraints are incorporated using pose-graph optimization. However, previous implementations have not fully leveraged SegMap's potential within a comprehensive, BA-like factor graph optimization framework. The work on PlaneLOAM [53] demonstrated that by combining robust loop detection from SegMap with precise feature-to-feature data registration, a LiDAR SLAM system can achieve improved trajectory and map accuracy, surpassing the limitations of pose-graph optimization alone.

2.2 Factor graphs in SLAM

2.2.1 The evolution of factor graphs

Over time, SLAM has evolved from simple probability-based methods to more advanced systems that use modern optimization techniques. Among these, factor graphs have become a powerful and flexible tool because they effectively capture the complex relationships between different variables in SLAM. Fast processing and the ability to handle large-scale environments are critical for real-world applications, such as self-driving cars, robotics, and augmented reality. Factor graphs play a key role in meeting these demands. This section explores how factor graphs are used in SLAM, highlighting their progress, benefits, and integration with other algorithms.

Factor graphs were introduced to SLAM as an extension of the Bayesian network and Markov Random Field paradigms. The work introduced by Kaess et al. [54] efficiently employed factor graphs for real-time SLAM by allowing incremental updates of the graph structure. This approach initiated research on representing the SLAM problem as a sparse optimization problem, significantly improving computational efficiency over the methods based on the EKF and particle filter. The development of iSAM2 [55] further refined the framework by introducing

Bayes tree-based relinearization and variable ordering, allowing efficient handling of large-scale SLAM problems. This work demonstrated the scalability and accuracy of factor graph-based approaches compared to traditional methods.

In factor graphs, the SLAM problem is modeled as graph where variables represent robot poses and map features, while factors represent measurements or constraints. Each factor encodes a probabilistic relationship derived from sensor observations or motion models. This representation enables a decomposition of the SLAM problem into smaller subproblems, which can be solved efficiently using Nonlinear Least Squares (NLS) optimization techniques, such as the Levenberg-Marquardt or Gauss-Newton algorithms. The modularity of factor graphs has made them especially useful for integrating various sensor modalities, including LiDAR, cameras, and inertial sensors. An example can be the GTSAM [56] and g2o [57] libraries, which have become a widely adopted tool for implementing factor graph-based SLAM systems, with applications in both research and industry.

Pose Graph Optimization (PGO), which is a type of factor graph, where only poses are optimized, is also widely used to refine trajectories in LiDAR SLAM [22, 58]. It represents the trajectory as a graph, where poses are linked by relative transformations. However, while PGO maintains computational efficiency, it can gradually accumulate drift and cause inconsistencies due to its dependency on fixed pairwise constraints [59].

Several comparative studies have highlighted the advantages of factor graphs over traditional SLAM methods. Unlike EKF-based SLAM, which suffers from quadratic complexity in map size and linearization errors, factor graphs leverage sparse matrix structures and iterative optimization techniques, resulting in better scalability and robustness [60]. Moreover, factor graphs allow for full optimization, providing globally consistent trajectory estimates compared to the local consistency offered by filtering-based approaches. Research has also shown that factor graphs outperform filter-based methods, particularly in scenarios involving large state spaces [61]. These advantages have made factor graphs a preferred choice for modern SLAM systems, particularly in applications requiring high precision and large-scale mapping.

2.2.2 Advancements in factor graph-based SLAM

Many advancements in factor graph-based SLAM have been introduced to address challenges in real-world applications. A significant extension is Dynamic-SLAM [62], which enhances traditional SLAM frameworks, typically designed for static environments, by integrating semantic information into factor graphs. This allows the system to model and compensate for moving objects, improving localization and mapping accuracy in dynamic conditions. Another important development is the work of Cunningham et al. [63], which is particularly beneficial for multi-robot systems. By leveraging factor graphs to share and integrate data across multiple agents, it enhances scalability and collaborative mapping capabilities, demonstrating efficient and robust multi-agent SLAM using factor graph optimization. In addition, a key technique used in factor graphs to manage computational complexity is marginalization, which allows for the removal of certain variables while preserving their influence on the remaining graph [64]. This is particularly important in long-term SLAM applications, where the graph can grow indefinitely as

the robot explores its environment. Furthermore, methods such as non-linear graph sparsification [65] help maintain computational efficiency without sacrificing accuracy, making factor graph-based SLAM more adaptable to real-world conditions. Combined with advanced optimization techniques, these approaches enable SLAM systems to handle large-scale 3D environments with millions of landmarks [22].

Approaches to make SLAM more robust have also gained attention as researchers seek to mitigate the impact of outliers and sensor noise. For this purpose, robust loss functions and switchable constraints have been incorporated into factor graph formulations, ensuring more reliable performance in complex environments. To improve robustness against outliers and noise, Agarwal et al. [66] proposed dynamic covariance scaling, a method designed to mitigate the influence of erroneous measurements during map optimization. This technique dynamically adjusts measurement covariances, preventing outliers from distorting the mapping process and ensuring more reliable performance in challenging environments.

Recent advancements have also explored the integration of factor graphs with machine learning techniques. Neural networks have been used to predict factors or improve data association, improving the robustness and adaptability of SLAM systems in unstructured environments. These hybrid methods, such as NeRF-LOAM [67], demonstrate the potential to combine traditional optimization with modern learning methods.

In addition, factor graphs offer a robust framework for combining data from different types of sensors. Their structure supports efficient optimization and uncertainty handling, leading to enhanced SLAM performance. Techniques, such as preintegrating IMU measurements, proposed by Forster et al. [68], have significantly improved visual-inertial SLAM. LiDAR-based systems, such as LIO-SAM [69], also integrate an IMU with LiDAR odometry using a factor graph-based optimization framework. By fusing LiDAR data with IMU pre-integration constraints, LIO-SAM achieves improved state estimation accuracy, particularly in challenging environments with sparse geometric features. In addition, it employs loop closure detection and IMU-based initialization to enhance robustness in long-term scenarios. This combination of LiDAR and IMU within a graph-based optimization framework makes it a precise and reliable system, offering real-time performance over long trajectories.

Most LiDAR SLAM approaches focus mainly on local registration and do not incorporate joint pose and structure optimization. However, global refinement methods, such as BA, are widely used in visual SLAM and Structure from Motion (SfM) frameworks. In these applications factor graphs are crucial, as they represent the relationships between sensor poses, 3D landmarks, and feature observations, allowing for a globally consistent reconstruction of the environment. In contrast to images that have dense and structured grids, LiDAR data consist of sparse and irregular point clouds, especially at greater ranges, which makes BA computationally demanding. As a result, many LiDAR SLAM frameworks prioritize local registration methods, such as ICP and its variants, which rely on subsampled point clouds [70, 71], distinctive geometric features [34], or NDT [18]. However, while subsampling can help reduce noise, it often comes at the cost of losing critical geometric details necessary for precise alignment. To address the computational complexity of BA, hierarchical methods such as HBA [72] have been introduced, but their performance is heavily dependent on an accurate initial trajectory, making them less reliable in

complex environments. For global optimization, Liu and Zhang [73] proposed a trajectory refinement method based on LOAM-style features, while [74] applied photometric techniques from visual SLAM to high-resolution LiDAR data. However, these methods struggle with robustness and scalability, particularly in low-resolution or noisy conditions. Other techniques, such as Signed Distance Function (SDF) [75] and implicit mapping [67], aim to improve noise filtering and provide compact scene representations. Despite their advantages, they do not integrate pose refinement and structural optimization within a single framework.

In general, factor graphs offer a flexible and efficient framework for modeling and solving complex mapping and localization problems. Furthermore, they enable the integration of data from heterogeneous sensors, thereby enhancing the accuracy of the estimation process. As a result, factor graphs have become a fundamental tool in robotics research, contributing to the reliability and scalability of SLAM systems, which has made them particularly valuable in applications such as autonomous navigation and mobile robotics.

2.3 SLAM and GNSS

2.3.1 Methods of improving GNSS positioning accuracy

GNSS is the most widely used technology for determining the position of self-propelled mobile platforms [76], offering a globally accessible solution for navigation. However, the accuracy and reliability of GNSS-based positioning depend on factors such as the number of visible satellites, atmospheric conditions, and the availability of differential correction signals [77, 78]. Although GNSS provides sufficient precision for many robotic applications, relying solely on satellite-based navigation without additional safeguards or algorithms to enhance reliability can lead to positioning errors. To address these challenges, various methods have been developed to validate and refine GNSS, incorporating filtering algorithms and on-board sensors to improve positioning stability and accuracy [78–80].

GNSS-based navigation systems used in many robotic applications must ensure high accuracy and operational safety. A critical aspect of this is achieving sufficient integrity within the navigation system, which involves the ability to detect and report hardware malfunctions or unreliable position estimates. Integrity, in this context, refers to the trustworthiness of the provided data, particularly when the system must alert the user or the control system about potential failures at a specific moment and with a known probability [81, 82]. One commonly employed method for detecting navigation faults is Receiver Autonomous Integrity Monitoring (RAIM). It was developed in the early 1990s, before satellite navigation was common for civilians, and works by analyzing redundant satellite measurements to identify errors and inconsistencies in positioning data. Modern RAIM techniques utilize various approaches, including least-squares estimation and Kalman filters. As research shows, implementing RAIM significantly improves the stability and reliability of positioning in many robotic systems [83, 84].

Cost-effective methods to enhance positioning accuracy, commonly discussed in the literature, often involve integrating inertial sensors. One such approach, presented in [85], combines data

from two low-cost GNSS receivers and an IMU. In this setup, the GNSS receivers utilize Real Time Kinematic (RTK) corrections from an external service, and by applying appropriate software and filtering the data using the EKF, authors achieved improved positioning accuracy, even when one receiver stopped receiving these corrections. However, many GNSS receivers are not equipped with a built-in IMU, and the integration of an external one presents challenges, such as the need for isolation from mechanical vibrations, which can be particularly problematic in many mobile robotic applications.

An alternative approach to enhancing positioning accuracy involves the use of SLAM algorithms with passive cameras or LiDARs. LiDAR SLAM, when combined with GNSS, can improve the accuracy of position estimation [86, 87] and enables seamless localization in both indoor and outdoor environments by dynamically switching between these methods depending on signal availability. Such methods, leveraging data fusion, filtering, or optimization algorithms, can achieve high positioning precision even in environments where satellite signals are weak or unavailable [80, 88]. A similar approach can be applied to visual SLAM, which offers a major cost advantage over LiDAR SLAM by substituting expensive LiDAR sensors with standard RGB cameras [89]. However, visual SLAM relies on the detection of distinguishable objects within the camera field of view, which makes this method particularly effective in structured environments or near buildings, where GNSS signals are often degraded due to obstructions [90]. Despite these limitations, ongoing research continues to explore hardware and software advances aimed at improving the effectiveness of visual SLAM in open spaces [79, 91], allowing for their integration with GNSS systems.

2.3.2 Approaches to SLAM and GNSS integration

Early research on enhancing satellite-based localization of ground vehicles focused on combining GNSS and SLAM pose estimates. This approach relied on 2D scanners and Kalman filtering as the fusion method [92]. More recent advances have applied filter-based techniques to integrate GNSS with 3D LiDAR and IMU data, using the EKF [93] or the Unscented Kalman Filter (UKF) [94].

However, as previously discussed in Section 2.2, factor graph optimization achieves superior performance compared to EKF or UKF when it comes to the fusion of SLAM and IMU data. This also applies to the integration of GNSS, as suggested by the experimental findings of [95]. The EKF struggles with outliers and linearization errors, particularly when vehicle poses rely on noisy GNSS data. A multi-state constraint Kalman filter [96] can help mitigate these issues, as demonstrated in [97], but does not fully exploit historical data. In contrast, formulating the problem as a factor graph allows for greater use of independent constraints and enables re-linearization with past states during optimization [98].

Wen et al. [95] highlighted the advantages of a tightly coupled GNSS integration approach. This method directly incorporates raw measurements, such as pseudoranges and Doppler frequency shift. Recently, tight coupling has been applied in the Kalman filter framework to integrate GNSS with LiDAR [97] and in graph optimization to fuse IMU [99], visual SLAM [100], and LiDAR SLAM [101] with GNSS raw measurements. However, the LiDAR SLAM algorithm in [101] is

not suitable for large-scale urban scenarios and relies on a custom hardware configuration for pre-integration of inertial data.

In contrast, loosely coupled methods rely on the position of the vehicle provided by the GNSS receiver. Optimization-based GNSS/LiDAR localization systems, such as [102, 103], use a loosely coupled approach to integrate LiDAR SLAM components based on LOAM. However, both approaches depend on precise external corrections, as [102] employs RTK GNSS, whereas [103] relies on Precise Point Positioning (PPP). Similarly, the LIO-SAM system [69] can loosely integrate GNSS positions to enhance global consistency.

Although factor graph formulation efficiently represents large-scale NLS optimization problems, it still assumes Gaussian sensor noise, which is not valid for Non-Line of Sight (NLOS) GNSS measurements. To address this, [104] introduced a technique for estimating GNSS measurement covariance by using a Gaussian Mixture Model to detect non-Gaussian outliers.

2.3.3 GNSS applications in public transportation

GNSS is widely used for vehicle monitoring in urban traffic [105, 106], including bus tracking [107] and analyzing driver behavior to detect unsafe actions such as speeding or sudden braking [108]. One of the primary advantages of GNSS is the affordability of the receivers, particularly when compared to alternative sensors such as LiDAR or cameras. Additionally, GNSS systems provide latitude and longitude coordinates within a global reference frame centered on the Earth, such as the World Geodetic System '84 (WGS-84). This ensures that the GNSS coordinates remain independent of the receiver position during startup and initialization.

In recent years, research on public transportation has increasingly focused on sustainable solutions, particularly electric buses. A comprehensive review of advancements in this field can be found in [109]. Although efforts are being made to develop autonomous buses [110], Advanced Driver Assistance System (ADAS) [111, 112] play a crucial role in enhancing safety and efficiency, especially in environments where full automation is not yet feasible.

In particular, urban environments pose challenges for accurate GNSS localization due to signal reflections and limited satellite visibility [113]. To improve positioning accuracy, researchers investigate GNSS-based enhancements [106]. Although aeronautical GNSS integrity methods are being adapted for urban environments [114], they require high data redundancy, which is often unavailable due to multipath effects and NLOS signals. Consequently, RAIM algorithms relying solely on GNSS pseudoranges are ineffective in such conditions.

As previously discussed, an alternative to relying solely on satellite-based localization is to integrate GNSS with pose estimates from exteroceptive sensors using fusion techniques [115]. This approach can involve leveraging omnidirectional vision [116] to improve satellite visibility, as well as utilizing multiple GNSS constellations (e.g., GPS, GLONASS, Galileo, and Beidou) and frequency bands. Additionally, modern GNSS receivers can extract raw measurements, such as pseudorange, Doppler shift, and carrier phase, to refine position estimates by modeling specific errors [117].

Although multi-sensor localization systems can address GNSS degradation caused by poor satellite visibility, multipath effects, or signal attenuation, they often come with significant drawbacks, including high sensor costs and increased computational complexity. In certain scenarios, a more practical solution may be the use of differential techniques, such as RTK GNSS [118, 119], which significantly improves accuracy compared to standalone GNSS methods and provides reliable localization in urban environments.

Chapter 3

Efficient map representations

3.1 Introduction

Accurate and consistent 3D reconstruction plays a crucial role in robotics, augmented and virtual reality, infrastructure inspection, and environmental monitoring. Modern 3D LiDAR sensors capture millions of points per second, making it possible to map large-scale environments with high detail [10]. However, ensuring global consistency in these maps requires precise scan alignment into a unified model that remains accurate, scalable, and robust against noise. BA is the gold standard for 3D reconstruction, as it jointly optimizes sensor poses and structure to achieve globally consistent results [120]. This technique is particularly effective in visual SLAM, where dense, structured image data provide rich visual cues for feature matching and correspondence estimation [121, 122]. However, LiDAR-based systems present unique challenges, including sparse and irregular data distributions and the absence of visual cues, making feature extraction and efficient map representation more difficult.

Moreover, in LiDAR-based SLAM, the choice of map representation is crucial for balancing accuracy, computational efficiency, and memory usage. Different representations are suitable for different applications, depending on factors such as the complexity of the environment, the required level of detail, and the available computational resources. Traditional approaches often rely on dense point clouds, which, while highly detailed, lead to significant storage requirements and computational overhead, particularly in large-scale or long-term mapping scenarios. At the same time, extracting salient point features from point clouds is computationally demanding and susceptible to inaccuracies due to range measurement uncertainties [33].

To enhance efficiency, various structured and feature-based representations have been developed, enabling more compact storage while preserving essential geometric information for localization and mapping. This chapter presents two approaches to processing LiDAR point clouds: one focuses on extracting higher-level geometric features, such as line segments and planar patches, while the other utilizes surfels. In both cases, these spatially extended features aggregate numerous points, making them more resilient to range measurement errors, as points that do not fit any feature can be discarded early in the processing pipeline. Moreover, SLAM system utilizing

higher-level features is expected to have improved accuracy because they offer additional geometric constraints compared to point-to-point correspondences typically used in methods processing raw point clouds.

In LiDAR SLAM, nonlinear optimization is commonly utilized for iterative registration of successive scans or aligning new scans with an existing map. However, the volume of data produced by modern LiDARs makes it impractical to use raw points as nodes in a factor graph. In contrast, geometric representations can be leveraged in optimization frameworks to simultaneously refine both the trajectory of a robot and the map. Therefore, the appropriate map structure can make methods such as BA feasible for LiDAR SLAM, similar to its application in visual systems [120].

3.2 LiDAR SLAM using high-level planar features

To address these challenges, the PlaneLOAM system was developed, as described in this section. It is a LiDAR-based SLAM system that leverages higher-level geometric features by grouping raw point cloud data into basic geometric forms. Instead of relying solely on individual points, this system represents the environment using planar patches and line segments, which are linked to the robot's trajectory through observations. This structured representation enables the construction of a factor graph that incorporates constraints derived from sensor motion and feature observations.

To improve long-term mapping accuracy, a loop detection mechanism based on SegMap [52] was integrated, which learns robust descriptors from segmented point clouds. These descriptors allow the system to recognize previously visited locations, adding new constraints to the factor graph. By establishing connections between previously unrelated features, this approach enhances consistency in large-scale mapping. In contrast, the method in [44] relies on geometric tests to determine planar feature similarity, which is more susceptible to errors caused by trajectory drift, viewpoint changes, and occlusions compared to the learned descriptor-based matching.

While PlaneLOAM introduces novel features, a new map representation, and an optimized processing pipeline, its core architecture remains similar to the LOAM system [34]. Like LOAM, this method combines real-time scan-to-scan pose estimation (odometry) with a slower but more precise scan-to-map localization approach. Previous research [35] demonstrated the effectiveness of this new mapping framework and evaluated its improvements in trajectory estimation accuracy compared to LOAM. Building upon that framework, a new version was developed that incorporates loop closure detection and map optimization using high-level features.

As mentioned above, the PlaneLOAM system builds on the steps used in LOAM by splitting the calculation process into the *odometry* and *mapping* modules and enhancing the functionality of LOAM with the integration of a loop closure module. The block diagram showing the PlaneLOAM architecture is presented in Figure 3.1, while the following subsections describe in detail each module of the system.

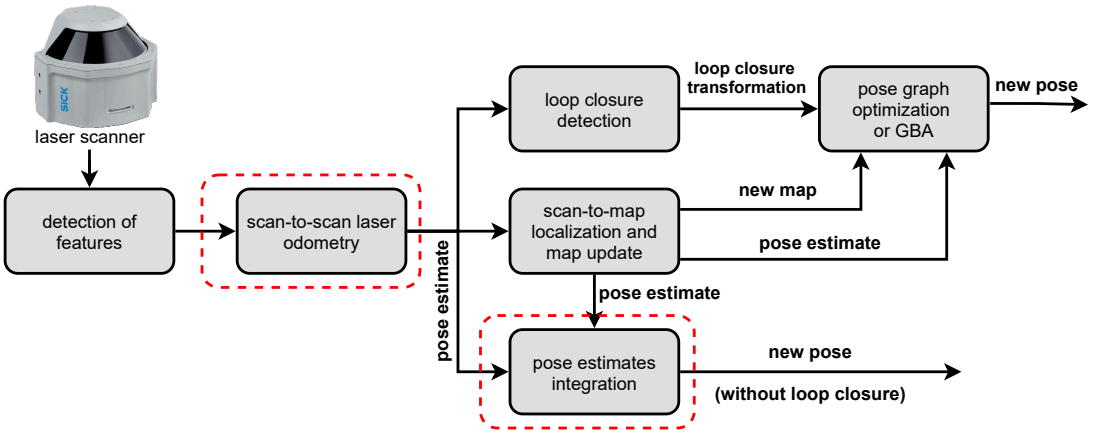

FIGURE 3.1: Block diagram showing the PlaneLOAM architecture. Blocks with red dotted borders show no substantial changes when compared to LOAM.

3.2.1 Odometry

In the odometry thread, PlaneLOAM operates similarly to the LOAM algorithm, with minor adjustments to support various 3D LiDARs. Firstly, the newly acquired point cloud is analyzed to extract sets of points that are classified as either planar or linear. Subsequently, these points are paired with the nearest points of the equivalent type (planar or linear) from the previously acquired point cloud. The matching process is conducted using the Euclidean distance between points of the same category, incorporating a straightforward sensor motion prediction model. For real-time performance, efficient kd-trees are utilized to accelerate the matching process. Constraints for the optimization problem, whose goal is to determine the pose estimate, are established by using the matches between points from successive point clouds. Although matching is done on a point-to-point basis, the optimization constraints rely on point-to-line and point-to-plane distances. Similarly to LOAM, at each step, equations for lines and planes are computed dynamically from the set of the five closest points in the previous point cloud. The steps for matching points and pose estimation are carried out repeatedly until the optimization reaches convergence. Consequently, the optimization problem in the odometry step can be expressed as:

$$\mathbf{T}_i{}^* = \underset{\mathbf{T}_i}{\operatorname{argmin}} \left(\sum_j f\left(\mathbf{p}_j, t_j, \mathbf{T}_i\right) + \sum_k g\left(\mathbf{p}_k, t_k, \mathbf{T}_i\right) \right), \tag{3.1}$$

where $f\left(\mathbf{p}_j, t_j, \mathbf{T}_i\right)$ denotes the point-to-line distance for a point $\mathbf{p}_j$ at its corresponding acquisition time t_j, counted from the start of the scan, $g\left(\mathbf{p}_k, t_k, \mathbf{T}_i\right)$ represents the point-to-plane distance for point $\mathbf{p}_k$ at its corresponding acquisition time t_k, also counted from the start of the scan, and $\mathbf{T}_i{}^*$ represents the pose estimate.

In contrast to global shutter cameras, LiDAR systems perform measurements point by point or in a small group of points. Therefore, when the sensor moves, each measurement in a complete scan is collected from a distinct pose. In order to obtain accurate estimates, it is essential to transform all measurements to a single reference frame by accounting for the LiDAR motion. In this work, the end of the scan was chosen as the reference frame for the LiDAR and the points were transformed accordingly. Assuming the velocity remained constant throughout the

scan, the transformation from a point $\mathbf{p}^m$ defined in the LiDAR reference frame at the time of measurement to a point $\mathbf{p}^e$ in the reference frame at the end of the scan can be calculated as follows:

$$\mathbf{p}^e = \exp\left\{\frac{t_s - t_i}{t_s}\log \mathbf{T}_s\right\}\mathbf{p}^m, \tag{3.2}$$

where t_i denotes the time of the i-th measurement with respect to the start of the scan, t_s represents the total duration of the scan, and $\mathbf{T}_s$ specifies the transformation that maps the point from the beginning to the end of the scan. The operators exp and log map between rigid-body transformations in the Lie group $SE(3)$ and their corresponding twists in the Lie algebra $\mathfrak{se}(3)$. The exponential map (*exp*) converts an infinitesimal transformation in $\mathfrak{se}(3)$ into a finite transformation in $SE(3)$, while the logarithmic map (*log*) performs the inverse operation, extracting a twist from a transformation matrix in $SE(3)$.

The transformation is determined by using the LOAM odometry, where t_s is set to 0.1 s for all LiDARs referenced in this work, while t_i varies depending on the sensor type:

- Velodyne HDL-64E:
$$t_i = \frac{\phi_i - \phi_s}{\phi_e - \phi_s}t_s, \tag{3.3}$$
where ϕ_i denotes the horizontal angle for the i-th measurement, ϕ_s represents the horizontal angle at the beginning of the scan, and ϕ_e is the horizontal angle at the end of the scan

- Sick MRS 6124:
$$t_i = \left(\frac{\lfloor l_i/6 \rfloor}{4} + \frac{\phi_i - \phi_s}{2\pi}\right)t_s, \tag{3.4}$$
where l_i represents the index of a scanning ring, counted from the top.

- Ouster OS1-64:
$$t_i = \frac{\lfloor i/64 \rfloor}{1024}t_s, \tag{3.5}$$
where i is an index of the measurement.

3.2.2 Mapping

The LOAM system classifies registered scan points as linear or planar points, which are utilized for optimization in both *odometry* and *mapping* steps. However, these points do not create higher-level features, as they are maintained as unordered point clouds for both categories. When needed, the five nearest points of a particular type from these combined point clouds are utilized to create a constraint.

The proposed PlaneLOAM implements an alternative map representation that aggregates points into high-level features. This method does not restrict the number of points, allowing a single feature to represent large objects, like a wall of a building or a road surface. Figure 3.2 illustrates a conceptual comparison of a wall representation in PlaneLOAM with the one in the original open-source LOAM implementation.

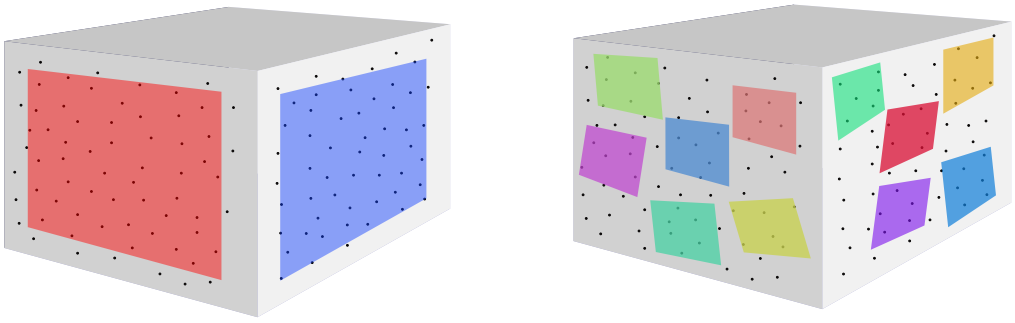

FIGURE 3.2: A conceptual comparison between the high-level planar feature in the PlaneLOAM system (left) and the cloud of planar points in the LOAM system (right).

Map representation

The map generated by PlaneLOAM consists of both linear and planar features, however, the presented approach affects mostly planes, since they tend to create large structures, regardless of the surrounding environment. Figure 3.3 illustrates a diagram of the structure of the stored map.

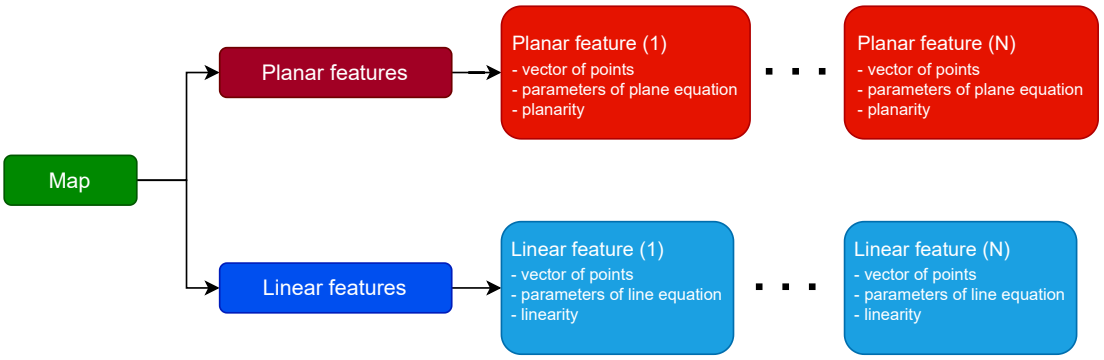

FIGURE 3.3: A diagram illustrating the organization of a stored environment map, which includes lines and planes that form high-level features.

Initially, each planar feature π is generated using five nearby points collected from one LiDAR scan. Its equation $\pi = (\mathbf{n}_\pi, d_\pi)$ is four-dimensional, with $\mathbf{n}_p$ representing the unit length normal vector of the plane and n_d denoting the distance from the plane to the origin. For simplicity, this equation is often referred to as the plane equation and meets the condition:

$$\mathbf{p} \cdot \mathbf{n}_\pi + d_\pi = 0 \tag{3.6}$$

for every point $\mathbf{p}$ on that plane. The normal vector $\mathbf{n}_\pi$ is determined through Principal Component Analysis (PCA) of a given feature's covariance matrix and is defined as the eigenvector associated with the smallest eigenvalue. The covariance matrix is determined relative to the centroid of a specified feature and quantifies the dispersion of all points associated with that feature. To represent the line segments $\lambda = (\mathbf{l}_d, \mathbf{l}_m)$, the six-dimensional Plücker coordinates are used, where $\mathbf{l}_d$ denotes the direction of the line and $\mathbf{l}_m$ denotes the moment of a line, both expressed in the global frame. The direction of the line $\mathbf{l}_d$ is initially determined by two points,

$\mathbf{p}_1$ and $\mathbf{p}_2$, as given by the equation:

$$\mathbf{l}_d = \frac{\mathbf{p}_2 - \mathbf{p}_1}{\|\mathbf{p}_2 - \mathbf{p}_1\|}. \tag{3.7}$$

The moment of the line $\mathbf{l}_m$ is calculated as:

$$\mathbf{l}_m = \frac{\mathbf{p}_1 \times \mathbf{p}_2}{\|\mathbf{p}_2 - \mathbf{p}_1\|}. \tag{3.8}$$

Depending on the size of the feature, $\mathbf{p}_1$ and $\mathbf{p}_2$ are either two actual points or two synthetic points selected to be on the line in the direction of $\mathbf{l}_d$, which is the eigenvector associated with the highest eigenvalue obtained from PCA.

The parameters of both the plane and the line are adjusted each time new points are incorporated into the feature, but this occurs only as long as the number of points remains below 30. Beyond this limit, the equation parameters that describe the feature stay fixed. Every feature retains additional information, including its centroid and the covariance matrix of points, which were also determined during the equation computation. Additionally, each feature is associated with coefficients that indicate either *planarity* or *linearity* (depending on the type of feature), which is used as an indicator of its quality. *Planarity* and *linearity*, represented by p_f, indicate the percentage of points that lie within a certain proximity to the feature, specifically less than 0.2 m in the proposed system. This percentage is determined using the following equation:

$$p_f = \frac{N_{f,0.2}}{N_f} \cdot 100\%, \tag{3.9}$$

where $N_{f,0.2}$ denotes the number of points located within a 0.2 m radius from the line's or plane's center, and N_f represents the total number of points in the chosen feature. That coefficient is used to determine the validity of a particular feature and to assess the dispersion of the points. The value of the parameter p_f, and consequently the quality of the feature, can potentially be enhanced by removing points that are further from the plane or line than a specified threshold. All generated features are processed through a series of steps that include creation, updating, deletion, and merging. The block diagram shown in Figure 3.4 illustrates the life-cycle of the high-level features.

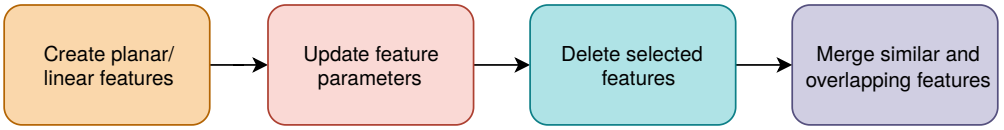

FIGURE 3.4: The processing pipeline of the developed system involves the creation, updating, deletion, and merging of features.

Creating features

The initial stage of the processing pipeline involves generating planar and linear features from the points registered in each scan. To form a plane, five nearby points are identified, and the plane equation is derived using their coordinates. While it is feasible to establish a plane equation with just three points, using more points enhances accuracy and decreases the likelihood of

generating invalid features. As the planar equation has predetermined parameters, every newly incorporated point must conform to the model dictated by these parameters, within a specified tolerance. The algorithm for adding points to the features is presented in Figure 3.5.

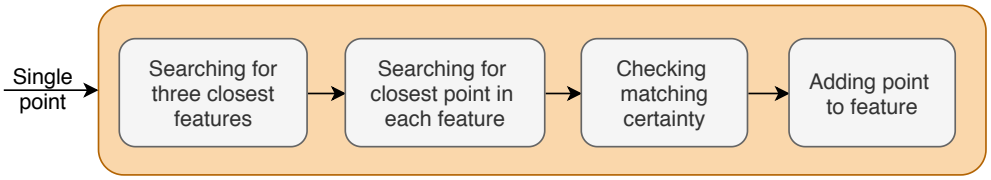

FIGURE 3.5: Steps taken to match a single point with an existing feature.

At first, for each new scanning point, the point-to-plane or point-to-line distance to the three closest features is determined, and the system attempts to match that point to one of these features. The point-to-plane distance is determined by following equation:

$$d_n = f\left(\mathbf{p}, \mathbf{T}_i\right) = \| \left(\mathbf{T}_i \mathbf{p}\right) \cdot \mathbf{n}_\pi + d_\pi \|, \tag{3.10}$$

where $\mathbf{p}$ represents a vector for the considered point, and $\mathbf{T}_i$ represents i-th pose (transformation from the local to the global coordinate system). For linear features, the distance from a point to a line is determined as

$$d_n = g\left(\mathbf{p}, \mathbf{T}_i\right) = \|(\mathbf{T}_i \mathbf{p}) \times \mathbf{l}_d - \mathbf{l}_m\|. \tag{3.11}$$

The algorithm initially discards any features beyond a distance of 0.6 m and then selects the three nearest features. The next step involves identifying the shortest point-to-point distance between the given point and the points associated with these features. The distance should be less than 0.7 m, preventing the inclusion of points that coincidentally meet the feature's equation. Finally, the algorithm verifies whether there is more than one feature that satisfies these criteria. In such a scenario, it computes the ratio of point-to-point distances to determine if any of them is notably smaller, according to the equation:

$$\frac{d_{e1}}{d_{e2}} < 0.7, \tag{3.12}$$

where: d_{e1} and d_{e2} represent the distances from the specified point to the nearest points in the first and second closest features, as illustrated in Figure 3.6. If the aforementioned equation is not met, the considered point will not be added to any of these features to avoid reducing the accuracy of the localization estimate due to potential incorrect correspondences. All these conditions need to be fulfilled in order for a new point to be added to the map. The initial condition guarantees that each point associated with a particular feature comply with its equation. The second condition prevents the creation of features that have an inconsistent density of points, while the last condition ensures that the matching process was carried out properly.

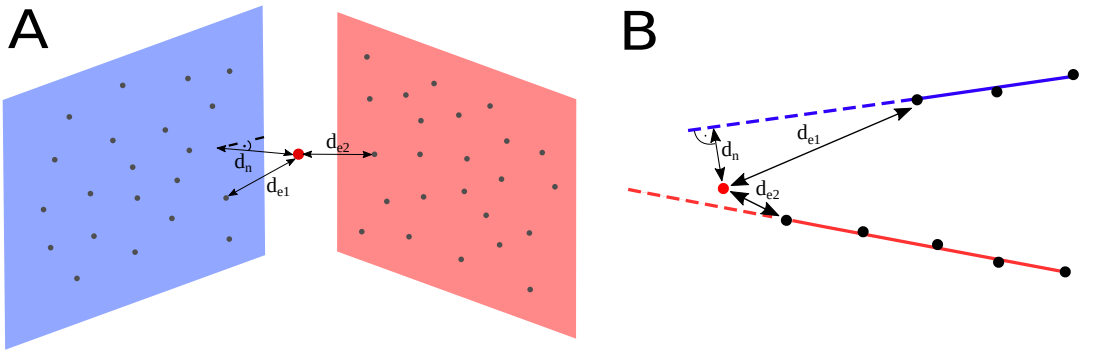

FIGURE 3.6: Distance to the closest feature along its normal (d_n), distance to the nearest point in the closest feature (d_{e1}) and the distance to the nearest point in a second closest feature (d_{e2}) taken into account when adding a point to the current planar (A) and linear (B) feature.

If the conditions previously stated are not met, a given point can still be associated with the feature that contains fewer than 5 points. In such a case, the considered point only needs to be within 1 m of any of these points. If that condition is not satisfied as well, a new plane feature is generated, to which additional points may be added as the rest of the scan is processed. Thanks to this approach, new features are generated only when a particular point does not align with any currently existing one, resulting in a map that contains larger features. It is important to mention that the specified parameters were established through several simulation tests, selecting values that minimize localization error.

Updating and deleting features

As modern LiDARs can generate tens of thousands of points in just one scan, it is essential to decrease the number of features stored in the map. To achieve this, features that are coplanar and colinear are combined, and features that are too small are removed after each scan is processed. To further enhance the system's performance, a voxel grid filter is applied to reduce the number of points in each feature. The process for updating all features is shown in Figure 3.7.

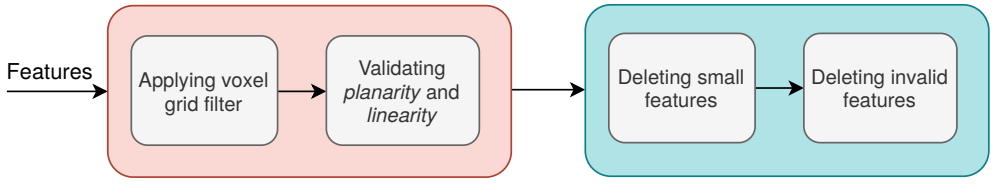

FIGURE 3.7: Block diagram showing steps for updating existing features and removing those that are too small or invalid.

Initially, the algorithm iterates through all the features to identify planes containing fewer than 5 points and lines with fewer than 3 points. These features are scheduled to be removed because they contain only a few points and their equations have not been calculated yet. The subsequent step involves applying a voxel grid filter to each feature. Afterwards, the algorithm verifies the validity of all remaining features. To accomplish this, it employs pre-determined *planarity* and *linearity* parameters based on Equation 3.9. The threshold for determining the validity of a

feature is set to 80%, and any feature that does not meet this criterion will be removed. The removal of chosen features is performed once after the completion of the update step.

Every time a new scan is obtained, the plane and line features are extracted and added to a map. This means that the number of current features keeps growing with each scan, despite the deleting and merging steps. This leads to a progressively increasing computation time required for assigning new points to existing features and finding correspondences between points and features during pose optimization. Thus, it became crucial to develop a solution enabling PlaneLOAM to operate on long sequences in real-time. This was achieved by choosing features that were not used during the optimization process and setting them as *inactive*. These features are not considered when assigning new points and during subsequent optimization, although they are stored in the map structure. A similar approach to this solution is employed in LOAM, which segments the map into cubes and eliminates points that fall outside the scanner's range during the search for point matches. In the case of PlaneLOAM, the *inactive* features are stored to enable the loop closure detection, as detailed in Section 3.2.3.

Merging features

The final step in the map processing pipeline employed by PlaneLOAM involves merging features that are coplanar and colinear. To merge two planes or lines, it is essential to confirm that they overlap and to verify if the resulting feature also will be valid. Figure 3.8 illustrates the block diagram presenting the procedure for merging two features.

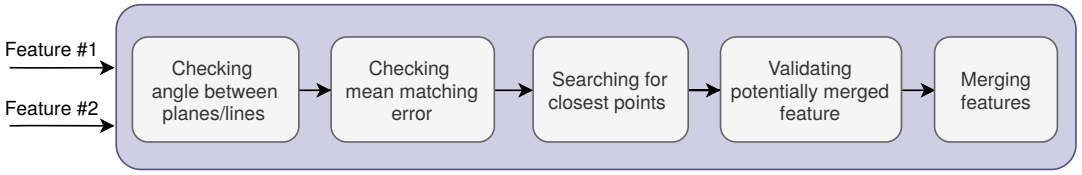

FIGURE 3.8: The process of merging two features based on angle, matching error, and distance between them.

The initial step in merging planar features involves computing the angle α using the following equation:

$$\alpha = \arccos\left(\frac{\mathbf{n}_{\pi 1} \cdot \mathbf{n}_{\pi 2}}{\|\mathbf{n}_{\pi 1}\| \cdot \|\mathbf{n}_{\pi 2}\|}\right), \tag{3.13}$$

where $\mathbf{n}_{\pi,i}$ represents a normal vector of the i-th plane. In the case of linear features the equation is expressed as follows:

$$\alpha = \arccos\left(\mathbf{l}_{d1} \cdot \mathbf{l}_{d2}\right), \tag{3.14}$$

where $\mathbf{l}_{di}$ represents the normalized line direction. The upper limit for the angle between them is set to 10°. If this condition is satisfied, point-to-plane (or point-to-line) distances d_{ni} between points from the first feature and the plane (or line) of the second feature are used to compute the mean residual error. A similar error is determined by using the distances measured between the points of the second feature and the equation of the first feature. These errors are calculated

with the following formulas:

$$\varepsilon_1 = \frac{1}{N_{f1}} \sum d_{ni},\ i = 1...N_{f1} \tag{3.15}$$

$$\varepsilon_2 = \frac{1}{N_{f2}} \sum d_{ni},\ i = 1...N_{f2} \tag{3.16}$$

To continue the merging process, the values of these errors must not exceed 0.1 m. Figure 3.9A provides a visualization of the angle α considered during merging together with the distance d_{ni} used to compute the matching error.

The subsequent step is to find the minimum point-to-point distance d_e between two planes (see Fig. 3.9B). To proceed with the merging, it must be less than 1 m. This ensures that two features overlap or are in proximity to one another. For lines, all specified distances are illustrated in Figure 3.9C.

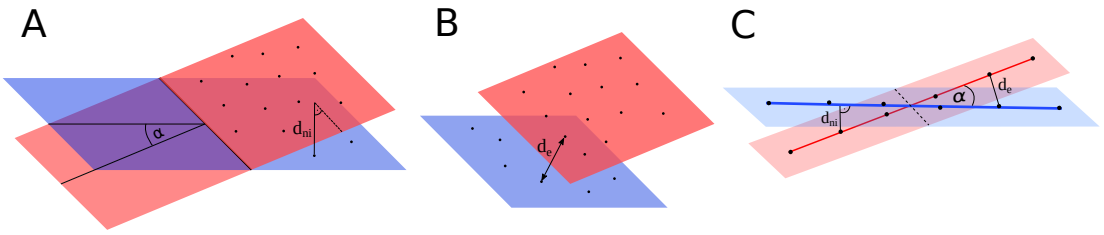

FIGURE 3.9: The angle (α) between two planes and example point-to-plane distance (d_{ni}) (A), distance d_e between two points on the planes considered for merging (B), angle α between two lines, an exemplary point-to-line distance d_{ni} and the distance d_e between any two points on the lines being matched (C).

If all of the previously mentioned criteria are met, the final step is to check if the newly created feature is valid. This is achieved by determining the *planarity* or *linearity* of that feature. If the value exceeds 80%, it indicates that merging can be executed. Otherwise, the process of merging is discontinued, as it would result in an invalid feature that would subsequently be removed in the next iteration.

Pose estimation

In both LOAM and the developed PlaneLOAM, the estimation of the pose within the *mapping* process relies on the point-to-line and point-to-plane distances. However, the distinction lies in the fact that PlaneLOAM matches the chosen linear and planar points from the latest point cloud to the globally consistent map of the high-level features. The optimization problem in this case is presented in Figure 3.10. Each high-level feature includes information about its equation as well as the list of points associated with it.

A key difficulty with employing a map consisting of high-level features lies in quickly matching new points to the increasing number of features stored in the map to enable real-time performance. Thus, when matching a point to map features, the system initially identifies the nearest point using efficient kd-trees and subsequently obtains the ID of the high-level feature associated with this point. Consequently, the PlaneLOAM concurrently handles two types of environment

map: one using points, similar to LOAM, and another relying on high-level features. Identifying point-to-feature correspondences through point-to-point matching is crucial for developing a real-time operational system.

In the PlaneLOAM system, the verification of correspondence between a point and a map feature follows a methodology similar to that described for the creation of features in Section 3.2.2. The main difference are the values of the chosen parameters. The distance to an existing plane or line (d_n) should be less than 0.2 m or 0.4 m accordingly. Additionally, the distance to the nearest point within the feature (d_{e1}) must be below 0.2 m, and the distance to the nearest point in the next nearest feature (d_{e2}) must exceed $0.7d_{e1}$.

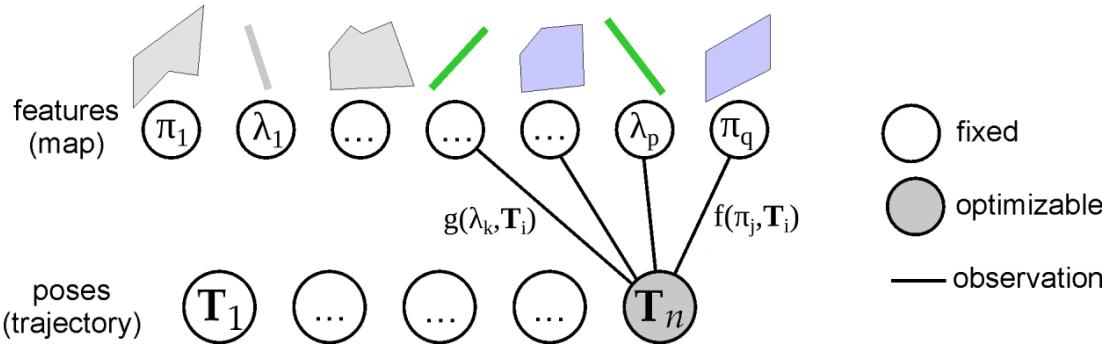

FIGURE 3.10: In PlaneLOAM, the pose $\mathbf{T}_i$ is determined by solving the optimization problem through the use of the point-to-plane ($f(\pi_j, \mathbf{T}_i)$) and point-to-line ($g(\lambda_k, \mathbf{T}_i)$) constraints. The parameters of the observed features are constant throughout the optimization process. In comparison to the LOAM method, the parameters for each feature are explicitly stored within the map.

In the *mapping* step, the optimization problem can be expressed as follows:

$$\mathbf{T}_i{}^* = \underset{\mathbf{T}_i}{\operatorname{argmin}} \left(\sum_j f\left(\pi_j, \mathbf{T}_i\right) + \sum_k g\left(\lambda_k, \mathbf{T}_i\right) \right), \tag{3.17}$$

where $f\left(\pi_j, \mathbf{T}_i\right)$ is the point-to-plane distance while $g\left(\lambda_k, \mathbf{T}_i\right)$ represents the point-to-line distance. In contrast to the *odometry* stage, all the points have an identical timestamp, as they are transformed to the end of the scan. This eliminates the need to consider the acquisition time of each individual point for computation purposes. Additionally, in PlaneLOAM, the parameters for the plane and line equations that are used to calculate the cost functions ($f\left(\pi_j, \mathbf{T}_i\right)$ or $g\left(\lambda_k, \mathbf{T}_i\right)$), are saved in the high-level features.

3.2.3 Loop closure

The loop closure module is divided into two main parts: loop closure detection and update of the information stored in the PlaneLOAM system. The update can be executed using either the pose graph representation or optimization on high-level features. The subsequent sections provide a more detailed explanation of the proposed approach.

Loop closing detection

Detection of loop closure is accomplished through the Segmap [52] system. This system processes LiDAR scans that have been motion compensated using PlaneLOAM to construct a map of segments, where each segment has an associated descriptor. Subsequently, these segments are aligned between the existing local map and the global target map to identify locations that have been visited before. Once segment correspondences are found, Segmap calculates a transformation between the sensor's position in the local map and its position during the previous visit. This calculated transformation is later used as a constraint in the factor graph. Figure 3.11 illustrates an example map generated by Segmap along with the segment matches.

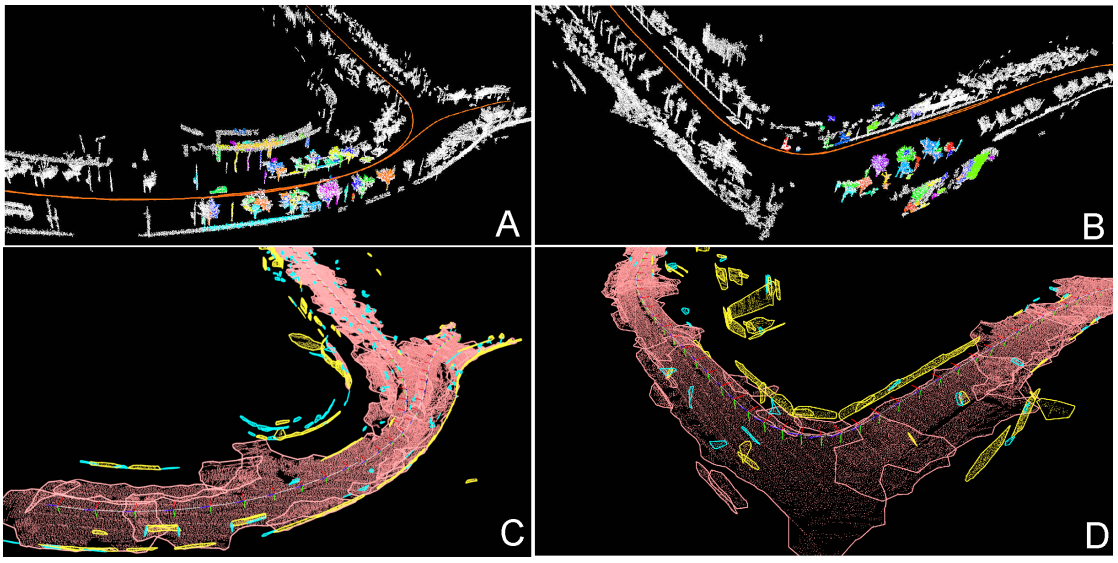

FIGURE 3.11: Utilizing the SegMap method for detecting loop closures within sequences from the MulRan dataset (A, B). Local maps of semantically segmented point clouds, represented in various colors, are matched against a global map of point cloud segments displayed in white. Identical regions are depicted as maps of high-level features (C, D), with orange indicating planar patches for the ground plane and approximately vertical features represented in yellow for larger walls and cyan for smaller planar elements.

In order to achieve accurate transformations,the geometric consistency grouping distance threshold is set to 0.5 m and a minimum cluster size to 8 (refer to [52] for more details). In order to prevent incorrect loop closures, the system applies the distance and cluster size criterion once again and improves the transformation using a point-to-plane ICP technique. The calculation of the global mapping trajectory uses only ICP factors, excluding odometry factors. It is important to note that the deep neural network used to generate the descriptors for the segments was trained on a different dataset (KITTI), which uses a different kind of LiDAR (Velodyne HDL-64E vs. Ouster OS1-64) and was recorded in a different environment.

Pose graph optimization

The outcome of the *odometry* and *mapping* processes provides a pose estimate for the latest sensor scan, which extends the length of the entire trajectory. In LOAM, after the *mapping* thread completes the scan processing, the estimated pose remains unchanged. In PlaneLOAM,

whenever the SegMap module detects a loop closure, all prior sensor poses are updated. In the proposed approach, the standard factor graph representation is used, where each sensor pose is depicted as a node. At the same time, the relative transformations between consecutive poses are modeled as edges linking the corresponding nodes, as presented in Figure 3.12. When SegMap identifies a new loop closure, it adds a new relative transformation connecting the relevant nodes in the graph. In the optimization process, the objective is to obtain new pose estimates that minimize the following error:

$$\underset{\mathbf{T}_1,\ldots,\mathbf{T}_n}{\operatorname{argmin}} \left(\sum_i h(\mathbf{T}_i, \mathbf{T}_{i+1}) \Omega_{i,i+1} h(\mathbf{T}_i, \mathbf{T}_{i+1}) + \sum_j h(\mathbf{T}_{l(j,1)}, \mathbf{T}_{l(j,2)}) \Omega_{l(j)} h(\mathbf{T}_{l(j,1)}, \mathbf{T}_{l(j,2)}) \right), \quad (3.18)$$

where $\mathbf{T}_1, ..., \mathbf{T}_n$ are the optimized poses, $h(\mathbf{T}_i, \mathbf{T}_{i+1})$ denotes the error between the i-th and $(i+1)$-th poses obtained from the *mapping* step, and $h(\mathbf{T}_{l(j,1)}, \mathbf{T}_{l(j,2)})$ is the error for the j-th loop closure, specified as the error between the $l(j,1)$-th and $l(j,2)$-th poses associated with that loop closure observation. The $\Omega_{i,i+1}$ and $\Omega_{l(j)}$ represent the 6×6 diagonal information matrices for factors connecting subsequent poses and those arising from loop closures. Their values were tuned experimentally to minimize the trajectory estimation error. In pose graph optimization, observations of features are not directly used, but they make it possible to determine the pose-to-pose constraints in the graph.

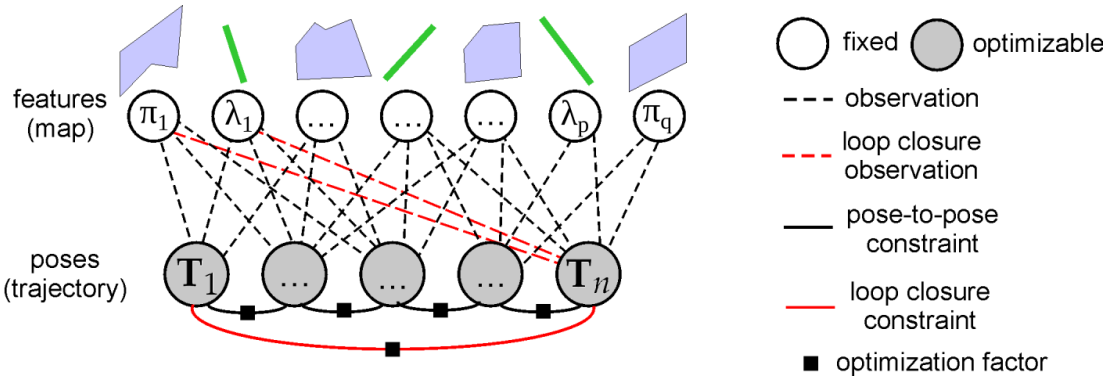

FIGURE 3.12: In PlaneLOAM, pose graph optimization is conducted with pose-to-pose constraints that come from the mapping process and link successive poses. Loop closure constraints connect non-consecutive poses and are determined via segment-based loop closure detection. The features in the map are excluded from optimization, as their observations are integrated into pose-to-pose constraints.

Loop closure based on map features

The approaches used in visual SLAM indicate that optimal accuracy is achieved through joint optimization of sensor poses and features through BA [120]. In PlaneLOAM, a system to execute a global BA was developed, enabling the optimization of all recorded poses and feature parameters to obtain the most precise trajectory estimate (see Fig. 3.13). In the proposed system, global BA is conducted exclusively after the detection of loop closure.

The system receives a relative transformation between two sensor poses from the SegMap module, which serves as initial data for loop closure factors. However, the objective is to leverage the feature-based structure of the map to enhance the accuracy of this initial transformation. Thus,

when evaluating two poses A and B matched via SegMap, the system initially identifies the plane and line features visible from these poses, denoted by $\mathcal{F}_A$ and $\mathcal{F}_B$. Subsequently, using the SegMap relative transformation, the points in the features $\mathcal{F}_A$ are matched by associating them with the features $\mathcal{F}_B$. This matching procedure is performed in a manner similar to the *mapping* step. Consequently, an optimized transformation linking the poses A and B is obtained, which is then used in the loop closure pipeline.

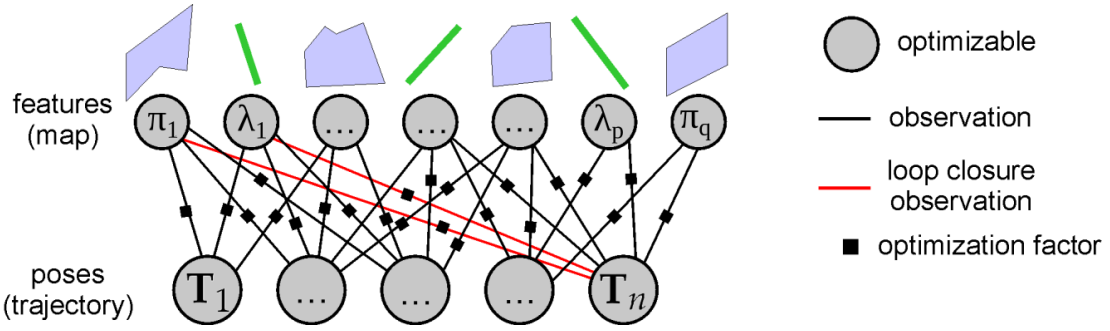

FIGURE 3.13: Global optimization leveraging pose-to-feature constraints to refine both the pose estimates ($\mathbf{T}_i$) and features (π_j, λ_k) stored in the map. There are no edges between poses, as the constraints coming from *mapping* and loop closure module are incorporated directly through the merging of corresponding features from re-visited locations.

It is mainly used to merge high-level map features that represent the same physical plane or line but have been assigned different IDs as a result of a system drift. After the features are merged based on the previously specified rules, the BA optimization is performed to minimize the following cost function:

$$\underset{\mathcal{K},\mathcal{F}_\pi,\mathcal{F}_\lambda}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \sum_{j \in \mathcal{F}_\pi} f\left(\pi_i, \mathbf{T_j}\right) \mathbf{\Omega}_{i,j} f\left(\pi_i, \mathbf{T_j}\right) + \sum_{i=1}^{n} \sum_{j \in \mathcal{F}_\lambda} g\left(\lambda_i, \mathbf{T_j}\right) \mathbf{\Omega}_{i,j} g\left(\lambda_i, \mathbf{T_j}\right) \right), \tag{3.19}$$

where $\mathcal{K}$, $\mathcal{F}_\pi$, $\mathcal{F}_\lambda$ represent the optimized sensor poses, planes, and lines, respectively, $\mathcal{F}_{\pi,i}$ and $\mathcal{F}_{\lambda,i}$ are the sets of indices for all planes and lines that are visible from the i-th pose of the sensor, $f\left(\pi_i, \mathbf{T_j}\right)$ and $g\left(\lambda_i, \mathbf{T_j}\right)$ are the error function for measurement between the i-th plane or line and the j-th pose, and $\Omega_{i,j}$ represents the information matrix for pose-to-feature factors. $\Omega_{i,j}$ is defined as a 3×3 diagonal matrix for pose-to-line constraints and a scalar for pose-to-plane constraints, with the values determined experimentally. Throughout the optimization process, the parameter sets for sensor poses, plane equations, and line parameters are jointly refined to minimize the overall error of the system.

Plane representation for optimization

In order to incorporate the parameters of the planes and lines directly into the optimization process, the features must be expressed using their minimal representation, and the Jacobians for the parameters need to be well defined. Thus, during the optimization process, the parameters associated with planes and lines are converted to minimal representations, and once optimization is finished, the results are reverted to their initial form to allow for the update of the map. The following paragraphs provide a more detailed explanation of the minimal representations utilized for both planes and lines.

In PlaneLOAM, the planes are characterized by a four-dimensional vector expressed as $\begin{bmatrix}\mathbf{n}_\pi & d_\pi\end{bmatrix}$, where $\mathbf{n}_\pi$ represents the normal and d_π denotes the distance from the origin of the coordinate frame. The minimal representation for the plane is three-dimensional due to the normalization of its normal vector. To create a minimal representation, the concepts described in [123] are employed, where exponential and logarithmic functions project elements from (n+1)-dimensional spheres onto n-dimensional tangent hyperplanes. Based on this concept, the plane normal $\left(\mathbf{n}_{\pi 3\times 1} = \begin{bmatrix}n_1 & n_2 & n_3\end{bmatrix}^T\right)$ is considered as an element of the S^2 unit sphere and can be mapped to a point on the tangent hyperplane $\left(\begin{bmatrix}\omega_x & \omega_y\end{bmatrix}^T\right)$:

$$\theta = \operatorname{acos}(n_3), \tag{3.20}$$

$$\omega_x = -n_2\frac{\theta}{\sin(\theta)}, \quad \omega_y = n_1\frac{\theta}{\sin(\theta)}, \tag{3.21}$$

where $\begin{bmatrix}\omega_x & \omega_y\end{bmatrix}^T$ represents the plane normal $\mathbf{n}_\pi$. Consequently, from the initial four-dimensional representation $(\mathbf{n}_\pi, d_\pi)$, the minimal three-dimensional representation $(\omega_x, \omega_y, d_\pi)$ is obtained. Using these equations allows for the calculation of the analytical Jacobian corresponding to each element of the proposed representation.

This spherical transformation has two particular cases that must be addressed. The first of them is when $\theta \to 0$, which requires a series expansion of $\frac{\theta}{\sin(\theta)}$. The other case occurs when $\theta \to \pi$, which can be avoided by replacing the specified plane with the equivalent one where $\mathbf{n}_\pi = -\mathbf{n}_\pi$ and $d'_\pi = -d_\pi$.

After optimization, the original 4D representation $\left(\mathbf{n}_\pi = \begin{bmatrix}n_1 & n_2 & n_3\end{bmatrix}^T, d_\pi\right)$ can be obtained using the exponential map that transforms it from the tangent hyperplane back to the S^2 group:

$$\theta = \sqrt{\omega_x^2 + \omega_y^2}, \tag{3.22}$$

$$n_1 = \omega_x\frac{\sin(\theta)}{\theta}, \quad n_2 = \omega_y\frac{\sin(\theta)}{\theta}, \quad n_3 = \cos(\theta), \tag{3.23}$$

Minimal line representation

In PlaneLOAM, lines are expressed using 6-dimensional Plücker coordinates $\begin{bmatrix}\mathbf{l}_d & \mathbf{l}_m\end{bmatrix}$, where $\mathbf{l}_d$ denotes the line's direction and $\mathbf{l}_m$ represents its moment. These three-dimensional components are orthogonal to each other. To ensure consistency, the direction of the line $\mathbf{l}_d$ is maintained in a normalized form with its first component being non-negative, that is, $\mathbf{l}_d(0) \geq 0$. To create a minimal representation, the concepts of [124] are employed, which make use of the properties of the $SO(3)$ group. This method encodes the line's parameters as a rotation matrix:

$$\mathbf{R} = \left[\mathbf{l}_d, \frac{\mathbf{l}_m}{\|\mathbf{l}_m\|}, \frac{\mathbf{l}_d \times \mathbf{l}_m}{\|\mathbf{l}_m\|}\right] \tag{3.24}$$

and subsequently calculates the representation using Lie algebra:

$$\boldsymbol{\omega} = \log\left(\mathbf{R}\right), \tag{3.25}$$

where $\log(\cdot)$ is the Lie algebra ($\mathfrak{so}(3)$) representation of $\mathbf{R}$. As a result, $\boldsymbol{\omega}$ encodes both the normalized direction and the normalized moment of the line. In order to recover the initial line parametrization, the line moment length is kept as the fourth parameter ($m_l = \|\mathbf{l}_m\|$). In summary, the initial representation $(\mathbf{l_d}, \mathbf{l_m})_{6\times1}$ is transformed into its minimal form $(\boldsymbol{\omega}, m_l)_{4\times1}$, taking into account three special cases:

- $\|\boldsymbol{\omega}\| \to 0$ - in such a case, a series expansion of $\frac{\theta}{\sin(\theta)}$ is used,
- $\|\boldsymbol{\omega}\| \to \pi$ - this problem is circumvented by conducting the computation with an equivalent Plücker representation ($\mathbf{l_d}' = -\mathbf{l_d}$ and $\mathbf{l_m}' = -\mathbf{l_m}$),
- $\|\mathbf{l_m}\| = 0$ - m_l is set to 0 and then the conversion is performed with the new unit vector $\mathbf{l}_m$, which is orthogonal to $\mathbf{l}_d$. Since $m_l = 0$, the arbitrary choice of $\mathbf{l}_m$ has no effect on the Plücker coordinates derived from the minimal representation.

After optimization, the original six-dimensional representation can be obtained with:

$$\mathbf{l}_d = \exp(\boldsymbol{\omega}) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{l}_m = m_l \exp(\boldsymbol{\omega}) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \tag{3.26}$$

where $\exp(\cdot)$ calculates the SO(3) group representation using the $\mathfrak{so}(3)$ Lie algebra element.

3.3 Bundle adjustment of surfel-based map and poses

This section presents the proposed approach to 3D LiDAR BA, called MAD-BA, which simultaneously optimizes both the map and the poses to maintain geometric consistency and accuracy. The method operates directly on 3D point cloud data using surfels, which are compact, disk-like elements that effectively represent surface geometry and provide an alternative to the high-level geometric features. The block diagram illustrating all the main processing steps of the developed system is presented in Figure 3.14.

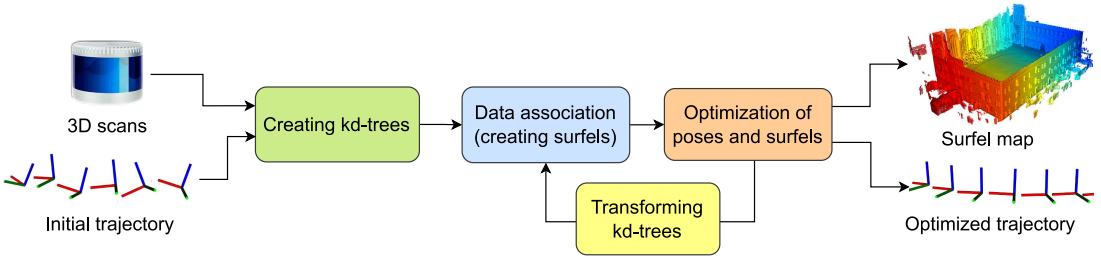

FIGURE 3.14: Block diagram of the proposed system. The input consists of 3D scans and an initial trajectory, while the output includes the surfel map and the refined trajectory. The key steps are: kd-tree construction, data association, optimization, and updating the kd-trees.

The following subsections present the methodology, covering surfel-based scene representation, efficient data association using kd-trees, and the optimization process to refine both poses and surfels. The optimization employs the Levenberg-Marquardt (LM) algorithm and the cost function designed to minimize geometric errors through a LiDAR uncertainty model.

3.3.1 Creating kd-trees

The initial stage of the processing pipeline involves converting each input point cloud $\mathcal{C}_i$ into a kd-tree $\mathcal{T}_i$. Each kd-tree $\mathcal{T}_i$ includes multiple leaves $l = (\mathbf{p}_l, \mathbf{n}_l)$ that contain a small subset of the point cloud. The leaf of a kd-tree is defined by the mean point $\mathbf{p}_l \in \mathbb{R}^3$ and the surface normal $\mathbf{n}_l \in \mathbb{R}^3$.

The process of creating kd-trees is recursive and starts by computing the mean and the covariance matrix of the given point cloud, followed by extracting the eigenvectors $\mathbf{W} = \begin{bmatrix}\mathbf{w}_0 & \mathbf{w}_1 & \mathbf{w}_2\end{bmatrix}$ from the covariance matrix using PCA. The eigenvector with the highest eigenvalue $\mathbf{w}_2$ is assigned as the direction of the greatest variation, while the eigenvector corresponding to the smallest eigenvalue $\mathbf{w}_0$ is identified as the surface normal.

A subsequent step involves computing an oriented bounding box for a given point cloud, which determines the minimum and maximum dimensions along each axis. If the largest dimension of the bounding box is smaller than a threshold $b_{\max} = 0.2\,\mathrm{m}$, indicating a compact cluster of nearby points, the recursion is terminated. If the bounding box extends beyond this threshold, the point cloud is divided into two subsets using the previously calculated mean value and direction corresponding to the eigenvector $\mathbf{w}_2$, as presented in Figure 3.15.

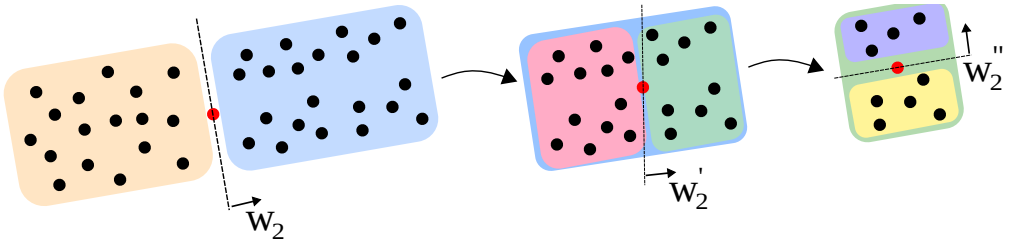

FIGURE 3.15: The strategy of building kd-trees, which relies on identifying the axis of maximum variance using PCA to divide the point cloud into smaller nodes [23]. The splitting stops when the size of the bounding box around the points in a node (marked with different colors) is below the threshold $b_{\max}$. In that case a node becomes a leaf of the kd-tree.

This splitting process is applied recursively to each subset, creating the left and right child nodes of the kd-tree. In parts of point cloud with many points, this method provides reliable surface normal estimates. However, in sparse regions where leaf nodes may contain only a few points, normals can become less accurate. To address this problem, when the shortest dimension of the bounding box corresponding to a given kd-tree node is less than a threshold $b_{\min}$, its normal is propagated to its children. This approach improves normal estimation for leafs with a small number of points without increasing the complexity of the kd-tree construction.

The final step involves transforming the kd-trees to the global coordinate frame using the initial poses provided as the input of the system. Importantly, this process does not modify neither the structure of the kd-tree nor the relative position of the leaves, ensuring that it can still be used for nearest neighbor search. As a result, kd-trees created from the input scans can be effectively used for matching purposes using global coordinates of leaves, which is necessary to perform data association step.

3.3.2 Data Association

The problem is represented using dense surfels and poses derived from a LiDAR odometry or SLAM pipeline. Each pose is represented as a homogeneous 3D transformation matrix $\mathbf{T}_i \in SE(3)$, while a surfel $s = \langle \mathbf{n}_s, \mathbf{p}_s, r_s \rangle$, is an oriented disk defined by its center point $\mathbf{p}_s \in \mathbb{R}^3$, surface normal vector $\mathbf{n}_s \in \mathbb{R}^3$, and radius $r_s \in \mathbb{R}$. Surfels are particularly effective for scene representation because they allow efficient fusion and updates during BA. Moreover, they can adapt well to structural changes caused, for example, by loop closures. Compared to voxel-based methods, surfels capture thin surfaces and complex geometry more effectively. In addition, surfels avoid the computationally intensive process of updating topology during optimization unlike mesh-based approaches.

Surfels are generated before optimization and preserved only during a single iteration. This approach ensures reliable data association, even if the initial poses are not accurate and far from optimum. They are formed by associating multiple leaves from different scans. In this way, they aggregate measurements of the same map region acquired from various poses. To associate estimated surfels with the leaves, the previously described kd-trees are utilized [23]. For each new LiDAR scan $\mathcal{C}_i$, a kd-tree $\mathcal{T}_i$ is constructed as a preprocessing step. This process generates a data structure that encodes the plane segmentation of the point cloud and enables efficient nearest-neighbor queries. The next step is to match all kd-tree leaves to the existing surfel set $\mathcal{S}$. A new surfel is generated if a leaf cannot be associated to any surfel. For a leaf l to be matched with a surel s, it must satisfy the following conditions:

- it must be the nearest neighbor to one of the leaves already associated with the surfel s,
- the Euclidean distance d_e between the surfel center $\mathbf{p}_s$ and the leaf mean $\mathbf{p}_l$ should be below 0.5 m,
- the distance d_n between the surfel center $\mathbf{p}_s$ and the leaf mean $\mathbf{p}_l$, computed along the normal of the surfel $\mathbf{n}_s$ should be lower than 1.0 m,
- the angle α between normals of a surfel $\mathbf{n}_s$ and a leaf $\mathbf{n}_l$ should be lower than 5°.

The visual illustration of the geometric criteria required to associate a leaf with a surfel is shown in Figure 3.16.

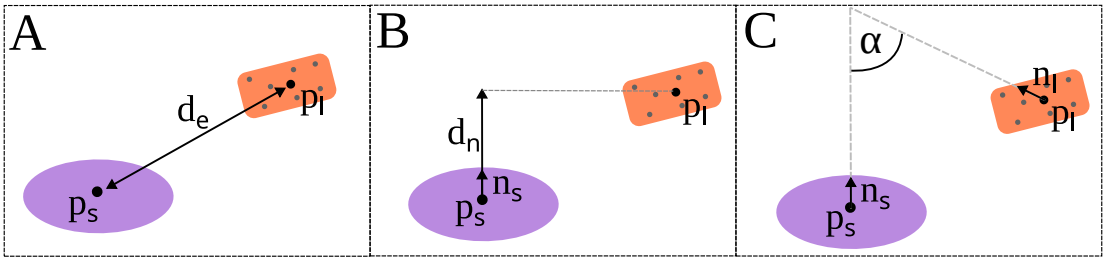

FIGURE 3.16: Geometric conditions required to match a leaf l with a surfel s. It includes the Euclidean distance between the surfel center and the leaf (A), the Euclidean distance along the surfel normal (B), and the angle between their normal vectors (C).

In the proposed method, a surfel must include at least two leaves from two different poses to be considered during optimization. However, it is beneficial to gather as many measurements as possible from various poses to better constrain the problem. An example visualization of the measurements that were obtained from different poses and associated within a surfels is presented in Figure 3.17.

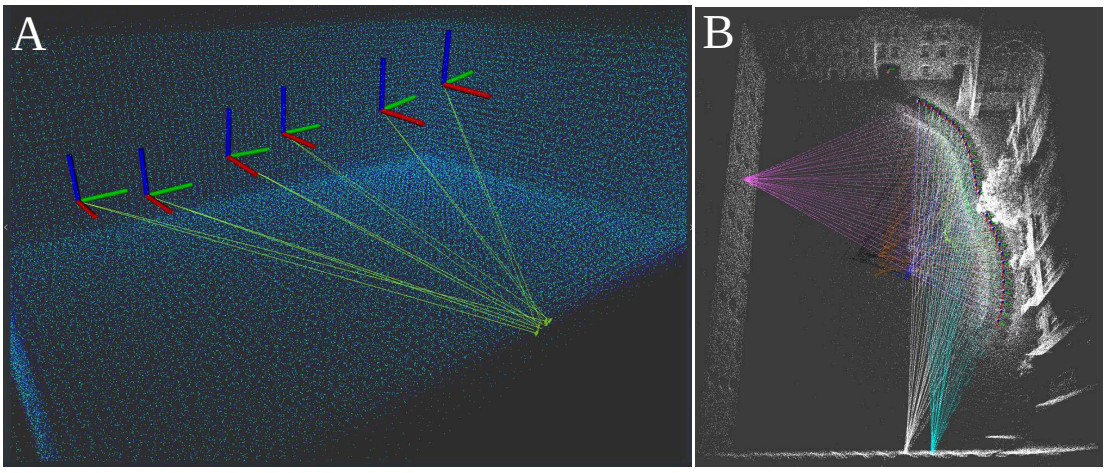

FIGURE 3.17: Visualization of the measurements coming from different poses (scans) that were associated to the same surfels. Figures (A) and (B) show examples for synthetic and real scans, respectively. Each line connects the pose with the center of the observed leaf.

When a new measurement leaf is associated with a surfel s, its mean, normal, and radius are updated. This is done by averaging the surfel centers $\mathbf{p}_s$ with the means of the corresponding leaf points $\mathbf{p}_l$ and averaging the surfel normals $\mathbf{n}_s$ with the normals of the leaf points $\mathbf{n}_l$. The radius r_s is set to the largest radius among the associated leaves to fully capture the variability of the points. Given the sparsity of LiDAR data, this approach ensures that the surfel accurately represents the local geometry, preventing gaps or inadequate coverage that could result in mapping inconsistencies. The result of this step is the surfel map, whose examples and comparison with the point cloud is presented in Figure 3.18.

As noted previously, surfels are maintained for only one iteration, which means that the association process is repeated multiple times during optimization. However, the efficiency of the kd-trees ensures that re-association can be performed quickly and reliably even in scenarios with large-scale data. It ensures that nearest-neighbor queries are accurate and guarantee the best surfel-leaf matches. Moreover, the PCA-based tree-building process minimizes the depth of the kd-tree [125], making it suitable for such an application. The steps performed during data associations are also presented in Algorithm 1, which complements this section.

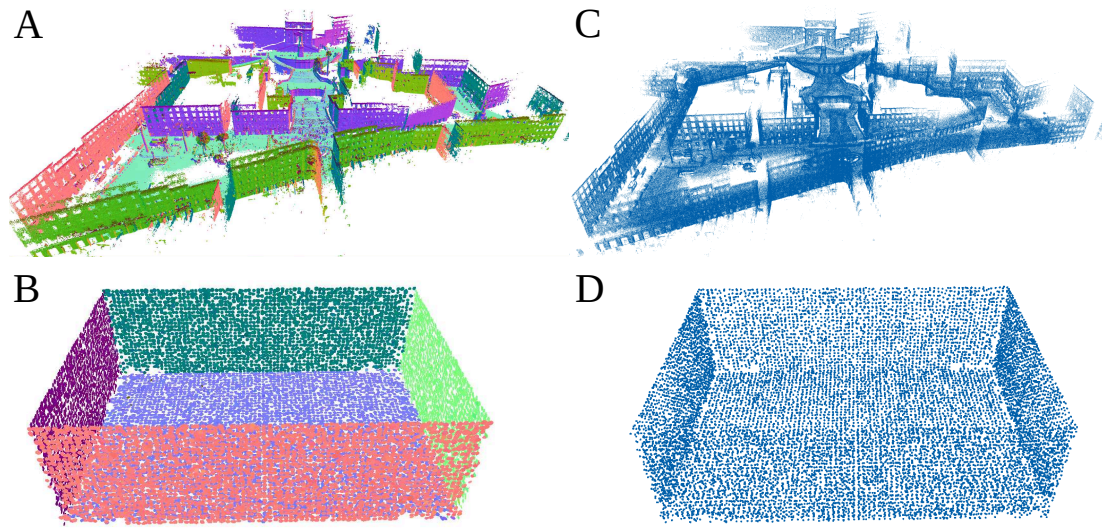

FIGURE 3.18: The comparison of the surfel map (A, B) with the point cloud (C, D). Surfels provide an alternative representation of the 3D surfaces that is more robust and versatile compared to standard points. Each surfel contain information about its normal vector (marked with different colors), which is particularly useful for matching purposes and rendering.

Algorithm 1 Data Association

input: kd-trees $\{\mathcal{T}\}$
output: surfel set $\mathcal{S}$
local: list of leaves associations $\mathcal{L}$

```
for T_i ∈ {T} do
    for T_j ∈ {T} such that T_i ≠ T_j do
        for l_i ∈ T_i do
            A_i ← {}                                   ▷ empty set of leaves associations
            if ∃A ∈ L such that l_i ∈ A then
                A_i ← A
            else
                A_i.add(l_i)                           ▷ leaf does not match any surfel
                L.append(A_i)

            l_j ← T_j.nearestNeighbor(l_i)
            if A_i.checkMatch(l_j) then
                A_i.add(l_j)                           ▷ leaf-surfel match

S ← createSurfels(L)                                   ▷ surfel creation/surfel normal update
return S
```

3.3.3 Optimization

The goal of BA is to jointly optimize surfels and poses. To minimize the cost function, the proposed system uses the second-order LM method. The optimization process runs for a specified number of iterations or until convergence. During each iteration, both surfels and poses are optimized together using an efficient Iterative Least Squares (ILS) solver [126]. Algorithm 2 provides a general description of the executed steps. In the case of poses, all 6-DOF are updated, however, surfels are restricted to move only along their normal direction. This constraint preserves the local surface geometry, reduces the optimization complexity by limiting the degrees of freedom, and aligns with LiDAR measurement uncertainty, which is most significant along

the surfel normal. In addition, it minimizes ambiguities in the data association step and ensures stable map refinements.

Algorithm 2 MAD-BA

input: clouds $\{\mathcal{C}_i\}$, initial poses $\{\mathbf{T}_i\}$
output: surfel set $\mathcal{S}^*$, refined poses $\{\mathbf{T}_i\}^*$

```
{T_i} ← buildKdTrees({C_i})                          ▷ construct kd-trees
for n ∈ # iterations do
    {T_i} ← transform({T_i}, {T_i})                  ▷ transform kd-trees
    S ← createMatches({T_i})                         ▷ Algorithm 1
    S*, {T_i}* ← optimize(S, {T_i})                  ▷ optimize poses and surfels
```

The general idea of the optimization step is illustrated in Figure 3.19. The pipeline incorporates information about LiDAR measurement uncertainty along with the raw scans and input trajectory provided by a LiDAR SLAM or odometry system. Using these inputs, the method jointly optimizes both the poses and the map, leading to their refinement with each iteration. By integrating LiDAR uncertainty directly into the optimization, the framework not only improves the robustness but also enhances the overall consistency of the resulting trajectory and map.

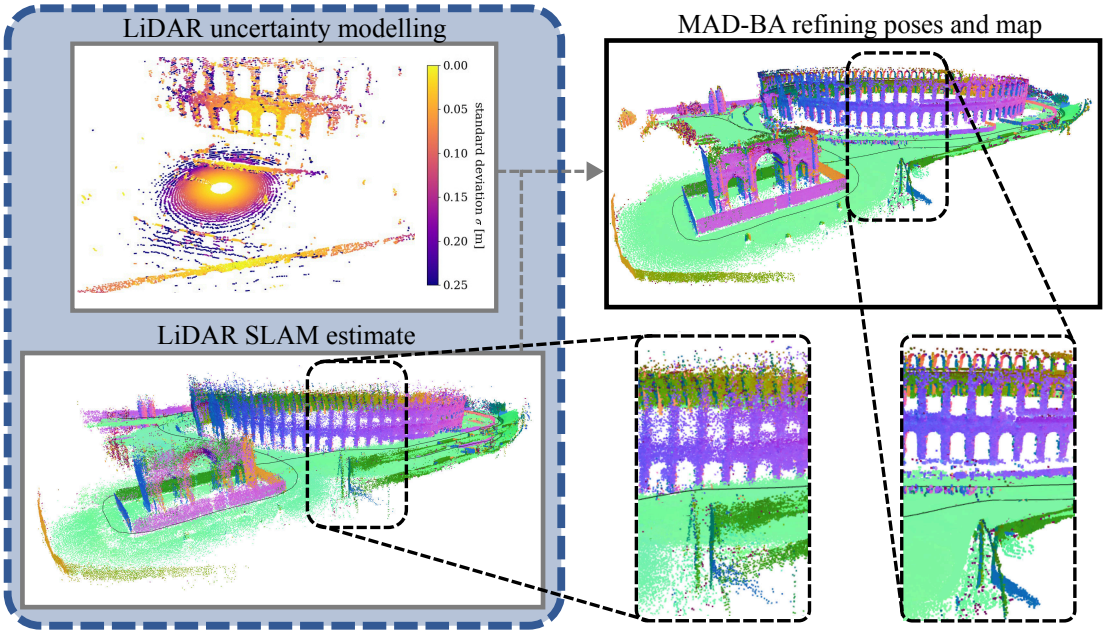

FIGURE 3.19: Pipeline of the optimization process. Starting with a trajectory estimated by a LiDAR SLAM system, the developed method jointly refines the poses and map. It utilizes the proposed uncertainty model to weight the optimization process. The two insets in the bottom right corner qualitatively showcase the improvements in map quality achieved through this approach.

LiDAR uncertainty model

LiDAR measurements are inherently noisy because of the varying environmental conditions. Accurate modeling of range uncertainty enables the system to prioritize reliable measurements, preventing optimization degradation, and ensuring meaningful contributions to BA [126]. However, existing uncertainty models developed for 2D LiDAR scanners [127] or airborne LiDAR systems [128] do not generalize well across different LiDAR sensors and application scenarios.

The shape of a LiDAR waveform is affected by sensor parameters, such as beam divergence, and measurement conditions such as range and angle of incidence. As the range or angle of incidence increases, the area intersected by the beam expands, resulting in a broader waveform. Unfortunately, the methods used by LiDAR firmware to process waveforms and calculate ranges are proprietary and generally undisclosed by manufacturers. Although it is intuitive that the waveform peak is critical for determining the range, the way broader waveforms are interpreted remains unclear. Additionally, commercial LiDAR systems typically output only processed point clouds, without providing access to raw waveform data. This lack of waveform details restricts the ability to analyze or model sources of measurement errors.

Beam divergence is a significant factor in various laser applications. In airborne LiDAR, for example, long-range measurements can result in a laser footprint that covers multiple objects within the vertical profile [128, 129]. Simulating LiDAR with idealized assumptions limits the ability of the training models to generalize to real data. To generate realistic scans from novel viewpoints, a physically accurate sensing model is introduced in [130]. In [131], an optimal beam shaping method is proposed to correct the intensity data, while [132] presents an analytical formula that links the waveform and the features of the measured target. However, these studies do not provide a general framework for estimating the uncertainty of measurement.

This work addresses this gap by contributing a method for estimating uncertainty and integrating it into a weighting optimization framework. The proposed approach is broadly applicable and relies only on the beam divergence, which is typically specified in the LiDAR manufacturer's documentation. The technique incorporates optical simulation, where a beam (modeled as a cone) is cast from the sensor's origin toward the mean position of each leaf. The waveform corresponds to the intensity of the beam echo measured over time. In practice, simulation discretizes this function by casting multiple lines within the beam cone, as illustrated in Figure 3.20. This process can be performed for each leaf immediately after its creation during the construction of the kd-tree. Although the waveform exists in time, the simulation samples a set of N_l ranges r_i. The standard deviation σ_l is then calculated using the following formula:

$$\sigma_l = \sqrt{\frac{1}{N_l} \sum_{i=1}^{N_l} (r_i - \|\mathbf{p}_l\|)^2} \tag{3.27}$$

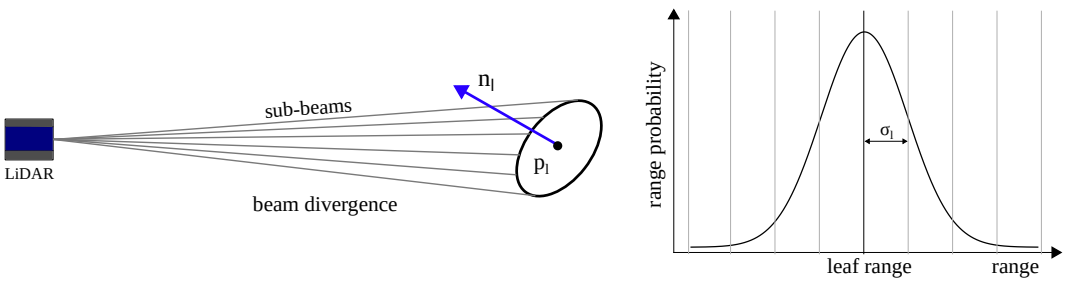

FIGURE 3.20: The model of LiDAR uncertainty, showing a single LiDAR beam simulated by casting a set of sub-beams towards a leaf l (left). The measurement uncertainty is represented by a Gaussian distribution derived from the sampled ranges (right).

Cost Function

The objective of the described method is to jointly optimize surfels and poses to achieve the maximum geometric consistency, where each pose corresponds to a 3D LiDAR point cloud. Let $\mathcal{K}$ represent the set of all poses, and $\mathcal{S}_i$ denote the set of surfels with associated measurements in pose $\mathbf{T}_i \in \mathcal{K}$. To robustly manage outliers, the Huber robust loss function ρ_{Huber}, parameterized by ρ_{ker}, is applied to weight the residual e. As a result, the total negative log-likelihood can be expressed as:

$$E(\mathcal{K}, \mathcal{S}) = \sum_{\mathbf{T}_i \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} e\left(\mathbf{T}_i, s\right). \tag{3.28}$$

The error term $e(\mathbf{T}_i, s)$ represents a sum of the point-to-plane distances between the surfel center $\mathbf{p}_s$ and the leaf mean $\mathbf{p}_l$, measured along the surfel's normal $\mathbf{n}_s$ and transformed to the local reference frame $\mathbf{T}_i$:

$$e(\mathbf{T}_i, s) = \sum_{j \in \mathcal{A}_s} \rho_{\text{Huber}}\left(\sigma_l^{-1} (\mathbf{T}_i^{-1} \mathbf{n}_s)^T \left(\mathbf{T}_i^{-1} \mathbf{p}_s - \mathbf{T}_i^{-1} \mathbf{p}_{l_j}\right)\right), \tag{3.29}$$

where σ_l is the standard deviation of the LiDAR measurements calucated using Equation 3.27, while $\mathcal{A}_s$ is the set of all the leaf matches for the surfel s, determined as outlined in Section 3.3.2.

Surfel optimization

The normal vector of a surfel is not directly modified during the non-linear BA optimization. However, the surfels are regenerated after each iteration of the described method. Firstly, this process involves transforming the kd-trees according to the updated pose estimates. Subsequently, the data association step is repeated, creating new correspondences between leaves from different scans. This incremental approach enables continuous refinement of the map and mitigates issues related to finding invalid correspondences that can occur when the input pose estimates are inaccurate. As described in Section 3.3.2, the normal vector of each surfel is calculated as the average of the normals of its leaves, which, in turn, are estimated during the creation of kd-trees that are built only once during the first stage of BA.

After updating the normals $\mathbf{n}_s$ through matching of kd-trees, poses and surfels are jointly optimized using the LM algorithm. During this optimization, the movement of the surfels is restricted to their normal directions $\mathbf{n}_s$, allowing the updated position of the surfel to be parameterized as $q_s \mathbf{n}_s + \mathbf{p}_s$. This joint refinement strategy establishes fewer constraints and avoids discontinuities that may arise when optimizing poses and structure separately. Despite the fact that independent surfel optimization is quicker and allows parallel processing [133], the joint approach provides better results by effectively capturing the relationships between poses and structure, resulting in more accurate and consistent improvements.

Pose optimization

The updates of the poses δ_T are parameterized as local transformations in the Lie algebra $\mathfrak{se}(3)$. Consequently, the transformation $\mathbf{T}_i$, which represents the pose in the global reference

frame, is incrementally refined using the operation $\mathbf{T}_i \exp(\delta_T)$. This approach of parameterizing updates in the Lie algebra ensures that updates to rotations remain well defined and free from singularities [122]. This methodology is systematically applied to all poses represented within the factor graph.

It is worth noting that the optimization of the poses not only refines the estimated trajectory but also significantly improves the structure of the map, especially if the initial error of the poses is substantial. This is caused by the fact that after each optimization, the kd-trees are transformed according to new pose estimates, which increases the overlap of the leaves and underlying scan points. The visualization of this effect is presented in Figure 3.21.

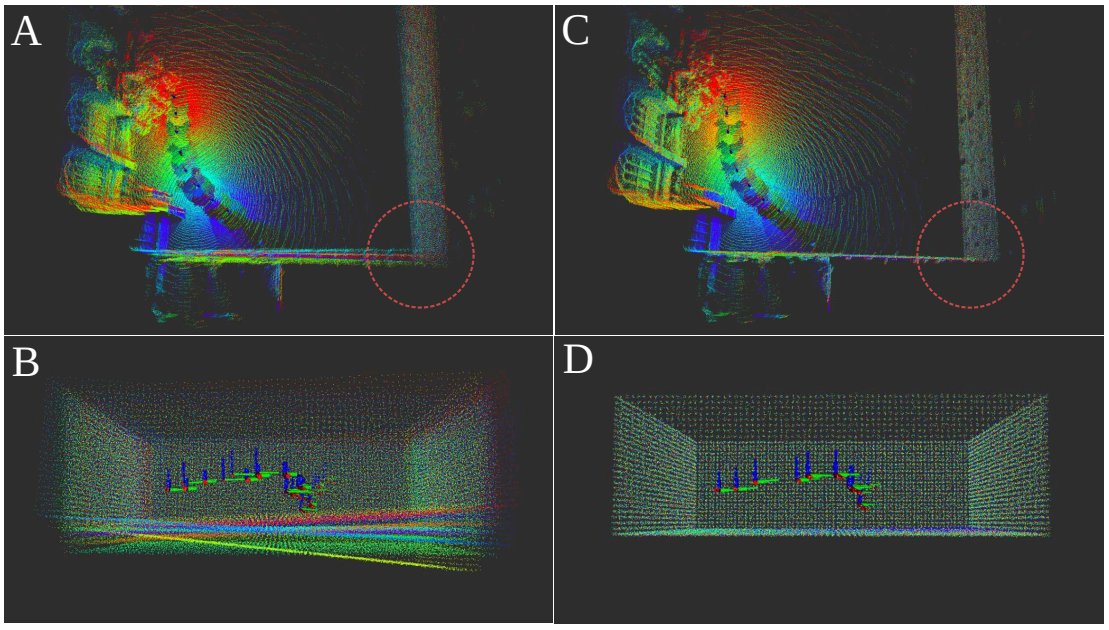

FIGURE 3.21: The impact of pose optimization on aligning registered scans that are used to construct kd-trees, visualized for both real (A, C) and synthetic data (B, D). Each color represents points from a different scan. Figures (A, B) display the scans before optimization, while Figures (C, D) show the improved alignment after optimization. The scan points are displayed solely for visualization and are not utilized in the system pipeline after kd-trees are created.

Although the primary enhancement achieved through this process lies in the refinement of the pose estimates, surfels play a crucial role in ensuring accurate alignment. Moreover, as the evaluation results show (see Sec. 5.2), this effect would not be possible if only poses were optimized, as the joint optimization of surfels and trajectory yields the best results. In general, the proposed approach incorporates multiple constraints between all poses, interconnecting them through surfels, which significantly reduces drift and ensures global consistency. Furthermore, the outlier rejection method that uses the robust loss function successfully handles spurious measurements and helps achieve the best performance. An example showing how the surfel map is improved through this LiDAR BA is presented in Figure 3.22.

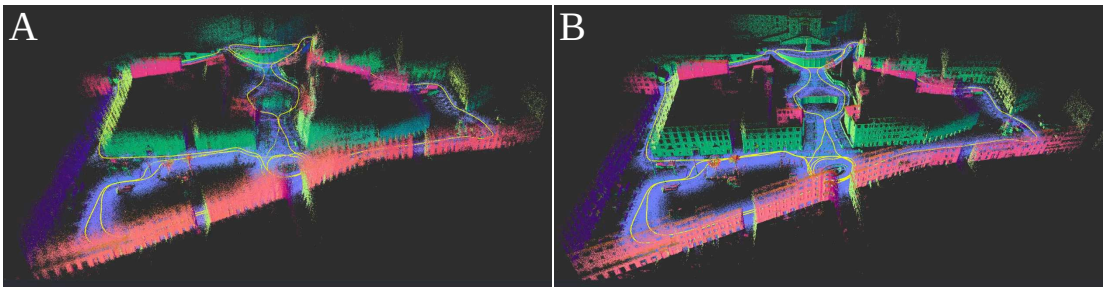

FIGURE 3.22: Effect of trajectory and surfel map optimization. Figure (A) presents the surfel map generated with the initial poses, and (B) shows the map after applying BA method.

3.3.4 Converting existing point-based maps to surfels

The described method for creating surfel maps can be applied to data from various sensors, including LiDARs and RGB-D cameras. Moreover, it is also possible to convert existing point cloud maps into surfels, extending the utility of previously captured data. An example of a surfel map, converted from a point-based map created using the DJI Matrice drone and capturing the campus of the Poznan University of Technology, is presented in Figure 3.23.

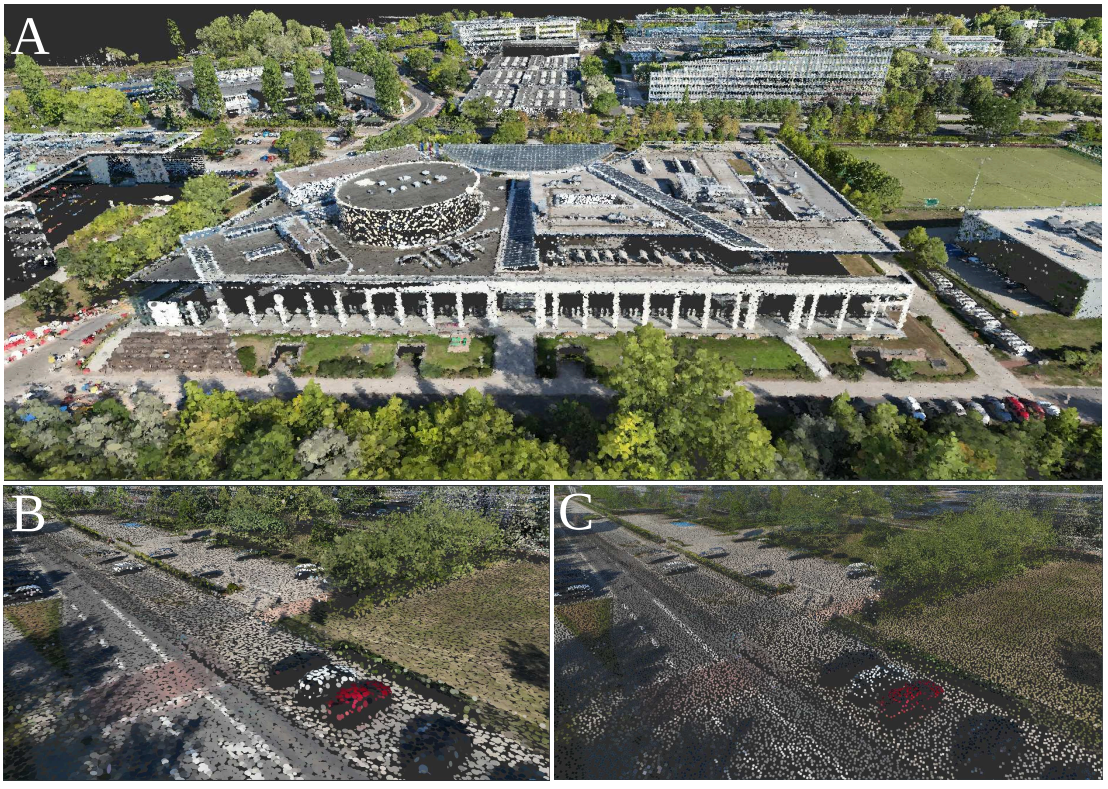

FIGURE 3.23: An example of a surfel map, generated by converting a point-based map of the Poznan University of Technology campus (A, B), and the fragment of the original point cloud map for comparison (C). The surfels include information about surface normals, enabling a more accurate representation of flat areas such as road surfaces.

Overall, surfels provide an alternative to traditional point cloud representations, offering several advantages that make them particularly useful for mapping and localization tasks in robotics and computer vision. Unlike point clouds, which consist solely of discrete points in 3D space,

surfels incorporate additional information about the orientation of the groups of points. As a result, they can better represent surfaces, allowing for more accurate 3D modeling, and enhance the fidelity of the map. Moreover, the inclusion of surface normals also enables faster and more efficient processing. For example, surfels allow the use of point-to-plane variants of the ICP algorithm, which align surfaces more effectively compared to the traditional point-to-point version. This improves both the speed and accuracy of the scan alignment, which is crucial for 3D reconstruction and real-time applications such as SLAM.

Chapter 4

Efficient use of GNSS data

4.1 GNSS and LiDAR SLAM integration

4.1.1 Introduction

Vehicle localization is commonly achieved using GNSS, which provides satisfactory accuracy in open environments. However, in urban areas with dense high-rise buildings, the accuracy and availability of pose estimates significantly degrade [134]. Meanwhile, extensive research has been conducted on SLAM techniques for vehicles, utilizing either passive cameras or active LiDAR sensors [4]. Recent advancements in LiDAR technology have made these sensors increasingly popular in self-driving vehicles due to their independence from lighting conditions and robustness in various environmental settings [135]. Although LiDAR SLAM or LiDAR odometry can deliver precise vehicle trajectories in highly structured and static environments [53, 76], these systems are prone to drift in scenarios that lack sufficient features or exhibit significant dynamics. Integrating GNSS measurements can help mitigate this drift by providing absolute (global) pose references. On the other hand, LiDAR SLAM performs reliably in areas where GNSS localization often struggles due to urban canyons and obstructions that reduce the visibility of the satellites. This complementary nature of GNSS and LiDAR SLAM indicates the potential for their integration to enhance localization performance.

This section describes the method developed for augmenting existing LiDAR-based localization systems with GNSS measurements. The proposed approach employs a factor graph formulation [98], which models the problem as a graph of constraints on vehicle states. Optimization is performed using the computationally efficient g^{2}o library [57], enabling precise and robust localization in challenging environments.

Most LiDAR odometry and SLAM algorithms estimate vehicle ego-motion by either associating consecutive observations along the trajectory or matching local observations (features or points) to a global map. These matched observations create constraints on the states of the vehicle equipped with a LiDAR sensor. While these states and constraints can be represented as a factor graph [98], this section introduces a novel method that enhances the factor graph by

incorporating constraints derived from absolute and relative GNSS measurements, such as pseudoranges and Doppler shifts. The proposed approach, named GALS, adopts the concept of tight integration within the factor graph, a technique that has been widely considered as the state of the art in visual-inertial SLAM [100], but has not yet been fully explored for the integration of LiDAR SLAM with GNSS data. Tight integration offers a significant advantage by utilizing a large number of individual GNSS measurements, which more accurately reflect the constraints imposed by multiconstellation GNSS data compared to the vehicle poses provided by the GNSS receiver in a loose integration approach. The proposed method is largely independent of the specific algorithm used to derive LiDAR constraints, making GALS a versatile framework for improving the quality of global trajectories generated by various SLAM systems. Furthermore, it supports the inclusion of constraints that come from loop closure detection. To address the challenges of inaccurate GNSS measurements, the developed system introduces a novel filtering procedure that leverages vehicle speed constraints. Furthermore, the proposed approach was thoroughly evaluated on the challenging UrbanNav dataset [136], across multiple environments, using different LiDAR-based SLAM algorithms (such as LOAM [137] and LIO-SAM [69]) and various GNSS receivers (see Sec. 5.3).

4.1.2 GNSS-based localization

GNSS receivers in most of the applications are primarily used to provide positioning data as latitude, longitude, and altitude. Interestingly, some GNSS modules additionally offer access to raw GNSS measurements such as pseudorange, Doppler shift, and carrier phase [99]. Depending on the type of receiver, these measurements can support multiple satellite constellations (e.g. GPS, GLONASS, Galileo, Beidou) and frequencies, enhancing positioning accuracy.

The pseudorange represents the measured distance between the receiver antenna and the satellite, calculated based on the signal travel time. However, it does not reflect the true distance due to factors such as satellite and receiver clock biases, which affect timing accuracy. Signal propagation is additionally influenced by Earth's atmospheric conditions and phenomena such as multipath interference. Consequently, a i-th pseudorange measurement $p_{i,n}$ between receiver and n-th satellite can be formulated as [138]:

$$p_{i,n} = \rho_{i,n} - (\delta_n^s - \delta_i^r) \cdot c + d_{i,n}^{ion} + d_{i,n}^{trop} + \epsilon_{i,n}^p, \tag{4.1}$$

where $\rho_{i,n}$ is the geometric range, δ_n^s and δ_i^r represent satellite and receiver clock biases, c is the speed of light, $d_{i,n}^{ion}$ and $d_{i,n}^{trop}$ are delays caused by ionospheric and tropospheric effects, and $\epsilon_{i,n}^p$ accounts for errors resulting from multipath interference and receiver noise. Satellite clock biases δ_n^s can be roughly calculated using parameters sent in GNSS messages, while the receiver clock bias δ_i^r must be estimated along with the position of the receiver. To achieve this, for each estimated position, at least four pseudorange measurements from satellites within the same constellation are required.

Doppler shift measurements capture the change in frequency of a carrier wave caused by the relative motion between a satellite and a receiver. This frequency shift is used to calculate the

radial velocity v_n, which represents the rate at which the distance between the n-th satellite and the receiver changes over time. The radial velocity can be determined using the equation:

$$v_n = \frac{c}{f_n} \cdot \Delta f_n, \tag{4.2}$$

where f_n is the carrier wave frequency, and Δf_n is the Doppler shift value for the n-th satellite. Doppler shift values can be utilized to estimate the receiver's velocity components along three axes ($v^r_{x,i}$, $v^r_{y,i}$, $v^r_{z,i}$) by processing v_n values across multiple satellites and their constellations.

The carrier phase of a GNSS signal also provides information about the distance between a satellite and a receiver. However, this range is measured in cycles of the carrier frequency, enabling very precise distance measurements. However, the main problem related to these observations is that the total number of cycles along the signal path is initially unknown, which is referred to as integer cycle ambiguity. Resolving this ambiguity requires advanced methods, such as PPP or differential GNSS, however, these techniques require external corrections. Since the developed system does not have access to such external corrections, it relies solely on pseudorange and Doppler shift measurements.

4.1.3 Factor graph-based integration

To integrate raw GNSS measurements with constraints from a SLAM system, the proposed GALS framework uses the g^{2}o library [57] for factor graph optimization. The key advantage of GALS lies in its flexibility, allowing LiDAR-related constraints to be supplied by any SLAM or odometry system that is capable of estimating 3D poses. The optimization process focuses on enhancing the localization accuracy of the receiver (i.e., vehicle) while also addressing the local nature of the SLAM trajectory estimation. By incorporating pseudorange constraints, the entire trajectory is transformed into a global coordinate frame. The structure of the proposed graph, which features optimizable nodes for each vehicle state and measurement constraints represented as edges, is illustrated in Figure 4.1, while the following subsections provide a detailed explanation of the factor graph formulation used in GALS.

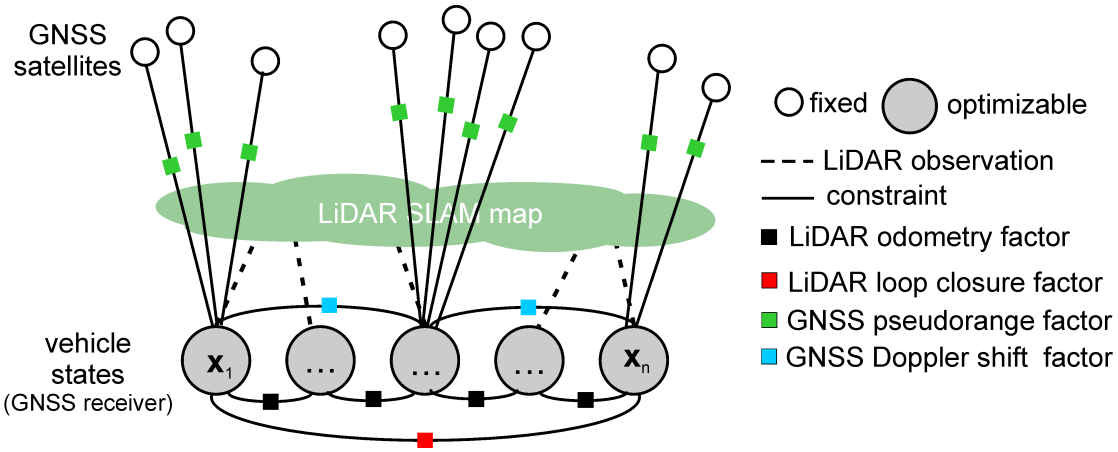

FIGURE 4.1: The overall structure of the factor graph in the GALS approach, combining LiDAR-based vehicle localization with GNSS measurements. The graph's nodes include state vector of a GNSS receiver and positions of satellites. It uses four types of factors: LiDAR odometry, LiDAR loop closure, GNSS pseudorange, and GNSS Doppler shift.

Factor graph nodes

Each optimizable node in the graph corresponds to the i-th state vector of a vehicle, denoted as $\mathbf{X}_i$, which includes a 6-DOF receiver pose $\mathbf{T}_i$ (consisting of a position and orientation) and individual receiver clock biases for each GNSS constellation:

$$\mathbf{X}_i = [\mathbf{T}_i, \delta^r_{i,1}, \delta^r_{i,2}, \delta^r_{i,3}, \delta^r_{i,4}], \tag{4.3}$$

where $\delta^r_{i,1\ldots4}$ represent the clock biases for GPS, Galileo, GLONASS, and Beidou, respectively. The clock bias drifts are not estimated, as experiments showed that including them did not improve trajectory accuracy but increased computational overhead.

The fixed (non-optimizable) nodes represent GNSS satellites, whose positions are known at the time of measurement. The receiver position $\mathbf{r}^r_i$ (a translational part of $\mathbf{T}_i$) is calculated in the Earth-centered Earth-fixed (ECEF) coordinate system (see Fig. 4.2), consistent with the n-th satellite position $\mathbf{r}^s_n$ and the receiver velocity $\mathbf{v}^r_i$ that are also expressed in the same frame.

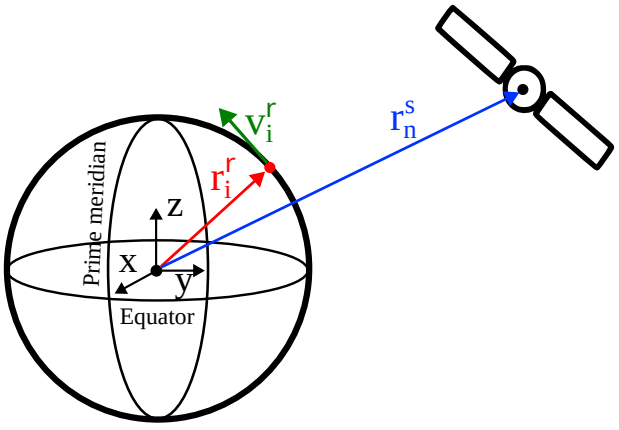

FIGURE 4.2: The receiver position $\mathbf{r}^r_i$, satellite position $\mathbf{r}^s_n$ and the receiver velocity $\mathbf{v}^r_i$ expressed in ECEF, which is a fixed reference frame with its origin at the Earth's center of mass.

It is assumed that the LiDAR and GNSS receiver mounted on the vehicle have been calibrated in advance and their extrinsic parameters are known. As a result, LiDAR scans are expressed in the receiver's local coordinate frame, simplifying the mathematical formulation.

Factor graph pseudorange edges

In urban environments, where GNSS measurements are frequently degraded, developed GALS employs a two-stage processing approach. Initially, the 3-DOF receiver position (latitude, longitude, and altitude) is estimated using only pseudorange measurements. These preliminary estimates are then utilized for GNSS data filtering. Subsequently, pseudorange constraints are formulated and applied in the factor graph only for the measurements that have been positively verified.

To estimate the receiver's position and clock biases, the scalar pseudorange measurement error is utilized, derived from Equation 4.1:

$$e_i^p = p_{i,n} - [\rho_{i,n} - (\delta_n^s - \delta_i^r) \cdot c + d_{i,n}^{ion} + d_{i,n}^{trop} + \epsilon_{i,n}^p], \tag{4.4}$$

where i represents the node (vehicle state) index, n corresponds to a specific satellite index (across all constellations), and $\rho_{i,n}$ denotes the geometric range to the receiver:

$$\rho_{i,n} = ||\mathbf{r}_i^r - \mathbf{r}_n^s|| + \omega_e(r_{x,n}^s \cdot r_{y,i}^r - r_{y,n}^s \cdot r_{x,i}^r)/c, \tag{4.5}$$

where ω_e denotes the angular velocity of the Earth. Satellite position is determined using broadcast navigation data, with calculations performed using the open-source RTKLIB library [138]. In the described approach, RTKLIB is employed to parse and process data from multiple GNSS systems, enabling the extraction of raw measurements while distinguishing between different constellation types. Furthermore, RTKLIB provides essential information related to each observation, including the n-th satellite coordinates $\mathbf{r}_n^s$, the satellite clock bias δ_n^s, the ionospheric delay $d_{i,n}^{ion}$, the tropospheric delay $d_{i,n}^{trop}$, and the i-th velocity $\mathbf{v}_i^r$ calculated from Doppler shift. Using this information, the 3-DOF position of the receiver can be estimated. The cost function minimized during the optimization is expressed as:

$$\underset{\mathbf{T}_i, \delta_{i,1..4}^r}{\operatorname{argmin}} \sum_i e_i^p(\mathbf{T}_i, \delta_{i,1...4}^r) \mathbf{\Omega}_p e_i^p(\mathbf{T}_i, \delta_{i,1...4}^r). \tag{4.6}$$

The single-element information matrix $\mathbf{\Omega}_p$ for the pseudorange constraint is derived from σ_p^2, the scalar variance of pseudorange error, which is calculated as:

$$\sigma_p^2 = \sigma_m^2 + \sigma_e^2 + \sigma_c^2 + \sigma_{ion}^2 + \sigma_{trop}^2, \tag{4.7}$$

where σ_m^2, σ_e^2, σ_c^2, σ_{ion}^2, and σ_{trop}^2 represent the variances of pseudorange measurement, ephemeris, code bias, ionospheric delay, and tropospheric delay, respectively [138]. The pseudorange measurement variance for a specific satellite is estimated as: $\sigma_m^2 = l_0^2 \cdot [l_1^2 \cdot (l_2^2 + l_3^2 / \sin(\theta_{el,i})]$, where θ_{el} represents the elevation angle of the satellite and (l_0, l_1, l_2, l_3) are measurement error factors. These factors are set to $(1.0, 100, 0.003, 0.003)$ for the GPS, Galileo, and Beidou constellations,

while $l_0 = 1.5$ is used for GLONASS [138]. The variance σ_e^2 is derived based on User Range Accuracy (URA) [139], a tabulated accuracy indicator transmitted by the satellite. The remaining components of Equation 4.7 are computed as follows: $\sigma_c^2 = l_4^2$, $\sigma_{ion}^2 = (d_{i,n}^{ion} \cdot l_5)^2 \cdot (f_{L1}/f_n)^2$, and $\sigma_{trop}^2 = (l_6/\sin(\theta_{az} + 0.1))^2$, where f_n is the frequency of the n-th satellite carrier wave, θ_{az} is the azimuth angle of the satellite, and f_{L1} is a constant ($1.57542{\cdot}10^9$ Hz). The error factor values are $l_4 = 0.3$, $l_5 = 0.5$, and $l_6 = 0.3$ [138].

Factor graph Doppler shift edges

Doppler shift constraints are incorporated in the form of 3-DOF translations, calculated based on the receiver velocity $\mathbf{v}_i^r$ and the time interval between consecutive GNSS observations. The receiver velocity is estimated beforehand using multiple Doppler measurements. The error function for distance constraints derived from Doppler shift is expressed as:

$$\mathbf{e}_{i,i+1}^D = \mathbf{r}_{i+1}^r - \mathbf{r}_i^r - \mathbf{v}_{i,i+1}^{r_D} \cdot t_{i,i+1}, \tag{4.8}$$

where $\mathbf{v}_{i,i+1}^{r_D}$ is the average velocity, and $t_{i,i+1}$ is the time interval between poses with index i and $i+1$. The 3×3 information matrix $\mathbf{\Omega}_D$, associated with these measurements is computed using RTKLIB procedures during velocity estimation, leveraging the least squares method [138].

Factor graph LiDAR SLAM edges

SLAM-based constraints take the form of 6-DOF transformations that link successive vehicle states. In this case, LiDAR SLAM odometry between consecutive vehicle poses is provided by the LOAM [34] or LIO-SAM [69] system. Since SLAM constraints represent homogeneous transformations, the error function is defined as:

$$\mathbf{e}_{i,i+1}^S = [(\mathbf{T}_i^S)^{-1} \cdot \mathbf{T}_{i+1}^S]^{-1} \cdot (\mathbf{T}_i)^{-1} \cdot \mathbf{T}_{i+1}, \tag{4.9}$$

where $\mathbf{T}_i^S$ and $\mathbf{T}_{i+1}^S$ represent the poses of successive vehicle states obtained from the SLAM trajectory. If a full SLAM method that can identify revisited places is used with GALS, the factor graph is extended with additional constraints from the loop closures. These constraints follow a similar formulation to Equation 4.9:

$$\mathbf{e}_{i,j}^L = [(\mathbf{T}_i^S)^{-1} \cdot \mathbf{T}_j^L]^{-1} \cdot (\mathbf{T}_i)^{-1} \cdot \mathbf{T}_j, \tag{4.10}$$

where $\mathbf{T}_i^S$ is again the pose of the i-th vehicle state from LiDAR SLAM, and $\mathbf{T}_j^L$ is the corresponding pose retrieved during the loop closure procedure.

The 6×6 information matrix $\mathbf{\Omega}_s$ for the LiDAR SLAM odometry constraints is derived by inverting the covariance matrix $\mathbf{\Sigma}_S$, which represents the uncertainty of vehicle pose. While this covariance matrix could be modeled as a function of surrounding feature points, as suggested in [140] for LOAM-like LiDAR localization methods, the proposed approach adopts a simplified

strategy, modeling the covariance matrix as:

$$\mathbf{\Sigma}_S = \text{diag}\,(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_\phi^2, \sigma_\psi^2, \sigma_\theta^2), \tag{4.11}$$

where σ_x^2, σ_y^2, σ_z^2, σ_ϕ^2, σ_ψ^2, σ_θ^2 represent the variance of the pose along the x, y, z, roll, pitch, and yaw axes, respectively. These diagonal covariance values are experimentally determined for the UrbanNav sequences to develop the model that minimizes the errors relative to the ground truth.

The computed covariance is scaled based on the distance traveled by the vehicle from the last node and is subsequently used to calculate $\mathbf{\Omega}_S$. Although this method does not account for variations in the number of points or features in the environment, it enables GALS to function with any LiDAR SLAM or odometry system that lacks the ability to estimate its own covariance.

The information matrix $\mathbf{\Omega}_L$ for the loop closure constraints is estimated in a similar way. However, in this case, the covariance matrix $\mathbf{\Sigma}_L$ is scaled by the number of graph nodes between the current pose and the loop-closed pose. This accounts for the inevitable drift in LiDAR SLAM over longer trajectories.

4.1.4 Optimization strategy with filtration mechanism

The values of the state vectors $\mathbf{X}_i$ are estimated using the g^{2}o library, which employs a graph-based representation to minimize the total squared error across all edges in the graph:

$$\begin{aligned} E(\mathbf{X}) = &\sum_{i\in\mathcal{M}} e(\mathbf{X}_i, \mathbf{X}_{i+1}, (\mathbf{e}_{i,i+1}^S)^T)\ \mathbf{\Omega}_S\ e(\mathbf{X}_i, \mathbf{X}_{i+1}, \mathbf{e}_{i,i+1}^S) \\ &+ \sum_{i,j\in\mathcal{L}} e(\mathbf{X}_i, \mathbf{X}_j, (\mathbf{e}_{i,j}^L)^T)\ \mathbf{\Omega}_L\ e(\mathbf{X}_i, \mathbf{X}_j, \mathbf{e}_{i,j}^L) \\ &+ \sum_{k\in\mathcal{G}} \sum_{n\in\mathcal{N}} e(\mathbf{X}_k, \mathbf{r}_n^s, e_k^p)\ \mathbf{\Omega}_P\ e(\mathbf{X}_k, \mathbf{r}_n^s, e_k^p) \\ &+ \sum_{k\in\mathcal{G}} e(\mathbf{X}_k, \mathbf{X}_{k+1}, (\mathbf{e}_{k,k+1}^D)^T)\ \mathbf{\Omega}_D\ e(\mathbf{X}_k, \mathbf{X}_j, \mathbf{e}_{k,k+1}^D), \end{aligned} \tag{4.12}$$

where $e(.)$ represents the error function, parameterized differently for each component. The sets $\mathcal{M}$, $\mathcal{L}$, and $\mathcal{G}$ correspond to the factor graph nodes associated with LiDAR SLAM odometry measurements, detected loop closures, and valid GNSS measurements, respectively. In addition, $\mathcal{N}$ denotes the set of all observed satellites across the utilized constellations. Since LiDAR SLAM odometry updates are more frequent than GNSS data acquisition, not every vehicle state includes pseudorange or Doppler shift constraints.

GALS utilizes an efficient gradient-based solver implemented in g^{2}o, which requires a good initial guess to converge to the desired global minimum. The proposed optimization strategy is structured into three main steps:

1. Initial state estimation: the initial positions of the state vector (3-DOF) are calculated using only pseudorange information stored in the relevant graph edges, providing a reasonable initial guess. Other edges are excluded from the optimization at this stage. This

step ensures proper solver convergence and allows for further filtering of states to eliminate erroneous GNSS measurements.

2. GNSS filtration: this step is critical due to the challenges posed by urban localization, where numerous NLOS and multipath signal receptions can introduce errors. Filtering them helps remove unreliable observations.

3. Final optimization: all states are optimized using both GNSS and SLAM system measurements to produce the final state estimates. GALS performs optimization continuously, considering a moving window of the 100 most recent poses. The final results are obtained through global optimization, which includes all nodes and edges added to the graph, ensuring that the estimated trajectory is optimal in terms of localization accuracy.

It should be noted that the first estimates of vehicle orientation within the global coordinate system can be imprecise. This is because the only orientation information comes from the SLAM system constraints that are expressed in the local frame. However, once the vehicle experiences a significant pose change, the information about the relative transforms allows the factor graph to accurately estimate the orientation. From that point forward, SLAM measurements can provide global pose estimates for the vehicle, even in parts of the sequence where GNSS signals are unavailable.

Filtration of GNSS measurements

GNSS measurement filtration is based on receiver speed values, which are calculated using three different methods. Given that GNSS systems typically provide less accurate altitude measurements compared to horizontal coordinates, filtration is applied separately to the horizontal and vertical axes using following scalar velocities:

- $v_i^{r_G}$ – velocity calculated for the i-th state using the distance and time difference between consecutive positions derived solely from pseudorange estimates,
- $v_i^{r_D}$ – velocity determined using Doppler shift measurements, computed through RTKLIB procedures,
- $v_i^{r_S}$ – velocity obtained via transformation between two successive poses from the SLAM trajectory.

Since $v_i^{r_G}$ is the most prone to errors and inaccuracies, it is compared to $v_i^{r_D}$ and $v_i^{r_S}$. If at least one of the following conditions is satisfied for any axis:

$$v_i^{r_G} > 2 \cdot v_i^{r_D}, \quad v_i^{r_G} > 2 \cdot v_i^{r_S}, \tag{4.13}$$

indicating that $v_i^{r_G}$ significantly exceeds the other velocity estimates, the GNSS measurements corresponding to that state are discarded and excluded from the factor graph integration. Exemplary results of this filtration process are shown in Figure 4.3, where the receiver positions

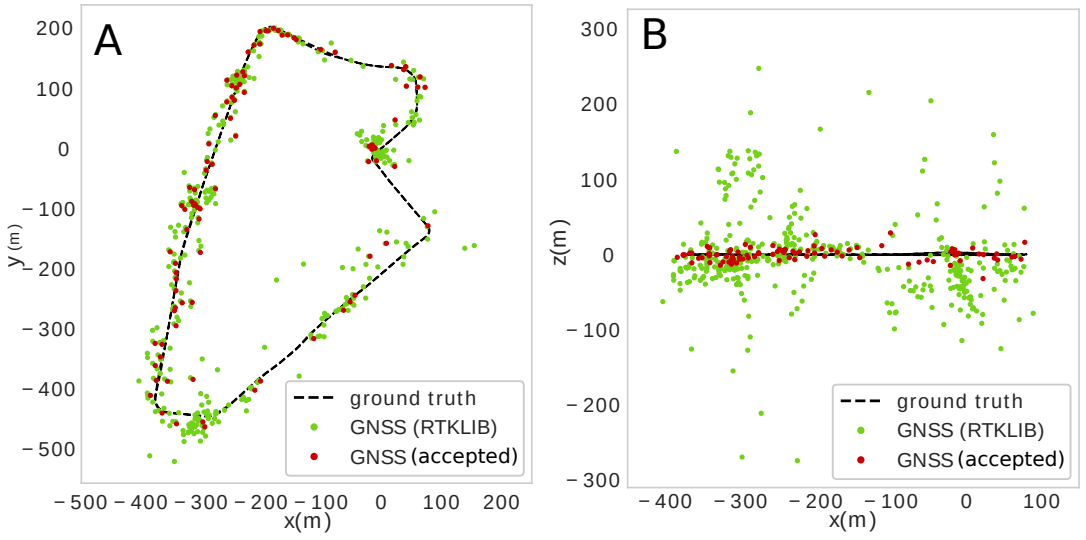

FIGURE 4.3: Visualization of the filtering process, showing the receiver positions obtained using the RTKLIB library (green points) and filtered positions incorporated into the factor graph (red points). Subplots (A) and (B) illustrate views in the XY and XZ planes, respectively.

computed using RTKLIB are compared to the filtered and accepted positions incorporated into the factor graph.

Once the filtering step is completed, the optimization process is carried out. As a result, it produces a trajectory in the global coordinate frame, which can be visualized on a map, as shown in Figure 4.4. In summary, this approach leverages factor graph representation to incorporate multiple raw GNSS constraints which ensure a consistent and accurate trajectory. By integrating these constraints, the developed method effectively mitigates the drift typically observed in SLAM systems. Therefore, it improves localization performance and provides a robust GNSS-assisted system for long-term navigation.

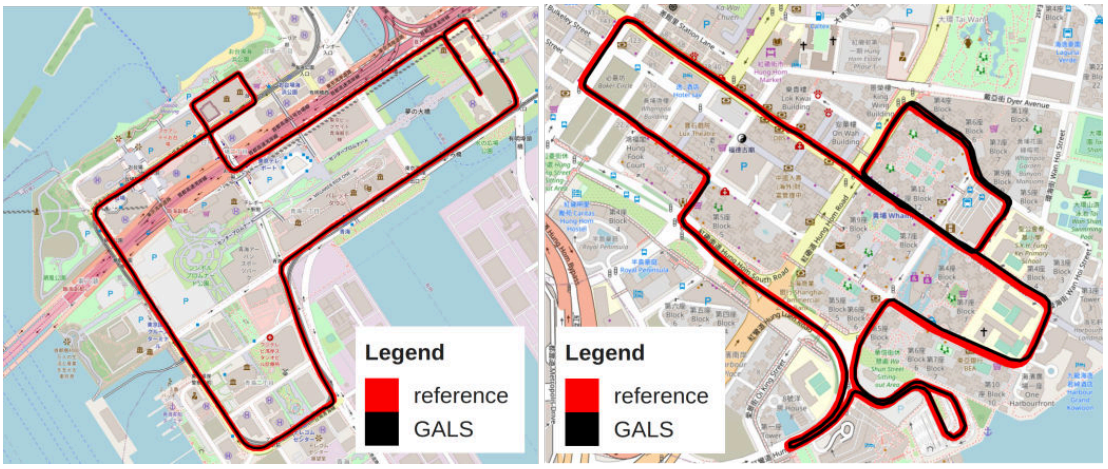

FIGURE 4.4: Trajectories for the Odaiba (A) and Whampoa (B) sequences from the UrbanNav dataset obtained using the proposed GALS approach. The estimated trajectories, along with the ground truth (reference), are overlaid on top of maps from OpenStreetMap.

4.2 GNSS and Visual Odometry integration

4.2.1 Introduction

While LiDAR-based SLAM methods have proven effective for localization in structured environments, other sensor modalities, such as cameras, can also provide valuable information for estimating the trajectory of a vehicle or robot. Visual SLAM or odometry algorithms offer an alternative approach by extracting and tracking visual features from consecutive camera frames to estimate motion and reconstruct the environment. However, like LiDAR SLAM, these systems suffer from drift over time, particularly in feature-scarce environments such as open fields or areas with repetitive visual patterns. Thus, this section described how high-precision positioning, achieved through the integration of GNSS data and VO, can be applied in agricultural settings, with a focus on field operations.

The proposed method integrates VO measurements with GNSS data to enhance positioning accuracy. By combining these two systems, it effectively mitigates short-term inaccuracies in GNSS readings, demonstrating the feasibility and effectiveness of such an approach in agricultural applications. This integration ensures reliable localization even in situations where GNSS signals degrade. In addition, by using widely available hardware, such as laptops with built-in cameras, this approach provides a cost-effective solution for improving localization accuracy in challenging agricultural environments.

The presented method was evaluated in agricultural environments (see Sec. 6.2), where integration of GNSS and VO offers significant benefits for field navigation. Its performance was assessed by comparing it with a commercial GNSS RTK system. The results demonstrate that factor graph optimization significantly improves localization accuracy and robustness in scenarios where GNSS localization degrades, providing a reliable solution for autonomous agricultural robots operating in large-scale outdoor environments.

4.2.2 GNSS localization in agriculture

The integration of GNSS technologies into agricultural practices has been a focal point of recent research, driven by the need for increased efficiency and precision in farming operations. Studies have demonstrated that the application of GNSS systems significantly enhances the accuracy of field tasks, ranging from soil sampling to yield mapping [141, 142]. These technologies have evolved to support high-precision tasks with minimal errors, essential for precision farming success [143, 144]. Research in precision agriculture has mainly focused on the development and assessment of various GNSS-based solutions. Gou et al. [145] and Upadhyaya et al. [146] have categorized positioning accuracy needs, emphasizing that different agricultural tasks require different levels of precision. For instance, while resource management can tolerate lower accuracy, tasks such as automated tractor guidance demand centimeter-level precision. Recent advancements have focused on combining GNSS with supplementary technologies such as VO to overcome limitations in accuracy and reliability. This hybrid approach has been shown to mitigate GNSS

inaccuracies in challenging environments, enhancing overall positioning precision [147, 148]. The combination of GNSS with visual data not only offers improved accuracy, but also provides a cost-effective alternative for small-scale farms that may not afford high-end GNSS solutions. Several studies have explored the environmental and economic impacts of GNSS applications in agriculture. Tayebi et al. [149] highlight the potential of GNSS to reduce the use of fertilizers and pesticides by enabling variable-rate applications. This targeted approach not only conserves resources, but also minimizes environmental pollution, aligning with the sustainability goals of the European Union [150].

A significant technological advancement in agriculture is the development of field robots. These robots are anticipated to further streamline field operations by minimizing the need for human operators and are designed to handle non-routine tasks that demand cognitive abilities such as perception, memory, and quick decision-making and action execution [151, 152]. Furthermore, the role of GNSS in enhancing operator efficiency has been extensively documented. The use of guidance systems significantly reduces overlap and gaps during field operations, leading to lower fuel consumption and input costs while increasing productivity [153]. Studies have reported cost savings, demonstrating the financial viability of adopting precision positioning systems [148, 149].

In general, the body of work underscores the innovative potential of GNSS-based technologies in modern agriculture. By improving field operation accuracy, reducing input usage and supporting sustainable farming practices, GNSS systems are essential to the future of precision agriculture. Ongoing research continues to refine these technologies, focusing on enhancing their accessibility and effectiveness for farms of all sizes.

4.2.3 Correcting GNSS trajectories with visual odometry

The literature review and experimental findings indicate that GNSS-based localization for agricultural robots encounters several challenges related to environmental factors and the GNSS equipment. Some issues observed during the experiments, such as incorrect robot poses caused by the temporary unavailability of RTCM corrections, can be mitigated through the implementation of an auxiliary localization system based on external measurements. As already discussed, the integration of visual [100] or LiDAR odometry [89, 101] with GNSS localization has become a prominent strategy in this field [88]. LiDAR-based systems offer high precision and robustness [87], but their high costs and maintenance requirements discourage the adoption of such technologiesin some applications. Furthermore, the lack of distinct features and the dynamic nature of vegetation-covered areas, which result in low repeatability in feature detection, reduce the effectiveness of LiDAR-based localization methods in agricultural scenarios [154].

Visual solutions that use passive cameras tend to be more cost-effective, as they leverage photometric features common in most environments [155]. Specifically, integrating monocular VO, which uses a single camera to track robot movement by observing visual features in the surroundings [156], presents an economical complement to GNSS. Direct VO, which estimates motion by minimizing the differences in pixel intensity between consecutive frames [157], offers a dense and continuous motion estimation. However, it can be sensitive to lighting changes and may struggle in low-texture or repetitive environments. In contrast, feature-based VO [158] provides

a sparser but potentially more robust solution, outperforming in scenarios where direct methods face difficulties. Feature-based approaches are often more adaptable to lighting variations and can perform well in environments with limited texture, making them suitable for a wide range of practical applications.

4.2.4 Monocular visual odometry as external localization method

Monocular VO utilizes data from a single camera to track a robot movement over time. However, it faces challenges such as scale ambiguity and struggles in dynamic environments [156]. To address these issues, this section presents the method for the integration of GNSS localization with VO, using the open-source ORB-SLAM3 library [13, 159]. ORB-SLAM3 employs Oriented FAST and Rotated BRIEF (ORB) features, which are well-suited for real-time applications, offering reliable and accurate tracking even under varying lighting conditions. Its ability to construct and optimize a map of the environment, along with keyframe-based optimization, makes it a popular choice for monocular VO applications [88, 159].

For large-area field localization, ORB-SLAM3 was configured based on preliminary experimental results, including reducing the input camera image size to exclude the sky, ensuring point features are detected only below the horizon line (see Fig. 4.5A). This adjustment prevents the tracking thread from focusing excessively on distant features, such as trees above the horizon, which often lack saliency and detection repeatability. This modification consequently improves the robustness of tracking in such scenarios (see Fig. 4.5B).

FIGURE 4.5: The visual representation of ORB point features (green) detected by ORB-SLAM3 in the image (A) and the tracked point features in the map (red/black) along with the camera trajectory (blue) (B).

In contrast to full SLAM operations, where ORB-SLAM3 detects and closes loops in revisited areas, the proposed approach does not utilize loop closure feature. This decision is based on two main factors. First, the sparse and uneven distribution of point features in vegetation-covered environments, where few salient objects are visible in the camera's field of view, makes the loop detection mechanism unreliable. Second, the localization constraints from ORB-SLAM3 are applied only locally, leveraging its robust visual tracking capabilities while avoiding issues related to drift and scale changes that can arise from unreliable loop closing and re-localization.

The developed solution builds upon previous work integrating LiDAR SLAM and GNSS [160], using a factor graph formulation to estimate the robot trajectory with pose constraints derived from two different sources. In this approach, the 6-DOF transformations generated by ORB-SLAM3 serve as constraints in the optimization process. In the experiments, the laptop computer

camera was used to provide visual data for the odometry pipeline. This makes the solution highly cost-effective, as it requires no additional hardware to be installed on the robot.

4.2.5 Integration using factor graph

Similarly to the previous cases, the trajectories derived from VO and GNSS data were combined using a factor graph representation, which effectively estimates the poses of the robot by incorporating multiple data sources [8]. The general structure of the constructed factor graph is illustrated in Figure 4.6.

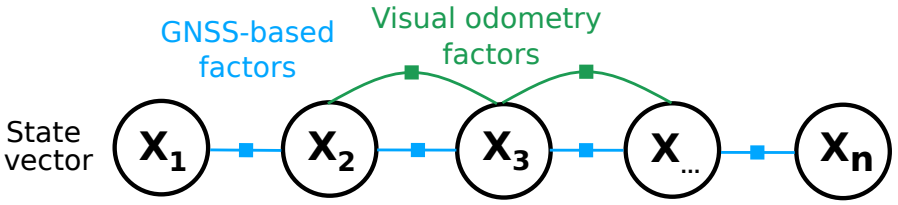

FIGURE 4.6: The general structure of the factor graph. It includes nodes representing the state vectors and edges that serve as constraints derived from both the GNSS and VO system.

The i-th estimated state vector, denoted as $\mathbf{X}_i$, contains the 6-DOF pose of a robot, comprising a rotation matrix $\mathbf{R}_i$ and a translation vector $\mathbf{t}_i$:

$$\mathbf{X}_i = [\mathbf{R}_i, \mathbf{t}_i]. \tag{4.14}$$

The graph is primarily built based on GNSS data, with new nodes and edges added whenever a new GNSS positioning message is received. The initial values for each state vector, used in the factor graph optimization process, also come from GNSS measurements. VO information is incorporated only when GNSS positioning accuracy degrades. This approach ensures that the trajectory is primarily determined by satellite navigation data, minimizing issues such as pose and scale drift.

There are several scenarios in which the accuracy of the GNSS system can degrade, making it temporarily insufficient for precision farming applications. To address this, the presented approach involves detecting these instances and integrating the information from the VO system accordingly. The decision to incorporate odometry measurements into the factor graph is based on the variance of the GNSS data. This variance, representing the accuracy of the robot pose, can be calculated from previous measurements or directly obtained from the u-blox GNSS receiver. In this case, the latter method was chiosen, as it is more accurate and eliminates the need for additional computations to generate the covariance matrix $\mathbf{\Omega}_G$ associated with GNSS positioning.

During the experiments, the variance values for the x, y, z axes reported by u-blox receivers in the RTK mode were consistent across all axes, reaching approximately $\sigma^2_{x,y,z} = 2 \cdot 10^{-4}\,\mathrm{m}^2$. Any significant deviations from these values indicate a degradation in accuracy, triggering the integration of VO measurements into the factor graph to enhance localization accuracy. Both GNSS and odometry measurements are incorporated into the factor graph similarly. The poses obtained from these systems are used to calculate the increments between consecutive graph

nodes, which are then added as edges. For GNSS data, the error function used in optimization is defined as:

$$\mathbf{e}_{i,i+1}^{G} = [(\mathbf{T}_{i}^{G})^{-1} \cdot \mathbf{T}_{i+1}^{G}]^{-1} \cdot (\mathbf{X}_{i})^{-1} \cdot \mathbf{X}_{i+1}, \tag{4.15}$$

where $\mathbf{T}_{i}^{G}$ and $\mathbf{T}_{i+1}^{G}$ are the poses from the GNSS trajectory.

The integration of ORB-SLAM3 data follows a similar process but involves additional steps. First, the measurements from VO are synchronized with GNSS data by matching their timestamps. Second, the scale of the VO trajectory must be determined, as the monocular SLAM system used in the experiments does not inherently provide scale information.

This step utilizes the Umeyama algorithm [161] to minimize the error between the previous GNSS and ORB-SLAM3 trajectories and calculate the scale value. It also aligns the two trajectories, enabling the estimation of the transformation between their respective coordinate frames. In addition, it provides the standard deviation of the VO poses relative to RTK GNSS data, which is used to compute the covariance matrix $\mathbf{\Omega}_{O}$ required for the optimization. Once these steps are completed, selected segments of the ORB-SLAM3 trajectory can be integrated into the factor graph when necessary. The error function for these constraints is defined as:

$$\mathbf{e}_{i,i+1}^{O} = [(\mathbf{T}_{i}^{O})^{-1} \cdot \mathbf{T}_{i+1}^{O}]^{-1} \cdot (\mathbf{X}_{i})^{-1} \cdot \mathbf{X}_{i+1}, \tag{4.16}$$

where $\mathbf{T}_{i}^{O}$ and $\mathbf{T}_{i+1}^{O}$ represent the poses from the VO system.

The outcome of factor graph optimization is the estimated state vector $\mathbf{X}$ for each node, which, in this case, includes the pose and orientation of the robot. The optimization process is performed using the g^{2}o [57] library, which minimizes the sum of squared errors across all edges in the graph:

$$\begin{aligned} E(\mathbf{X}) = &\sum_{i \in \mathcal{G}} e(\mathbf{X}_{i}, \mathbf{X}_{i+1}, (\mathbf{e}_{i,i+1}^{G})^{T})\ \mathbf{\Omega}_{G}\ e(\mathbf{X}_{i}, \mathbf{X}_{i+1}, \mathbf{e}_{i,i+1}^{G}) \\ &+ \sum_{i \in \mathcal{O}} e(\mathbf{X}_{i}, \mathbf{X}_{i+1}, (\mathbf{e}_{i,i+1}^{O})^{T})\ \mathbf{\Omega}_{O}\ e(\mathbf{X}_{i}, \mathbf{X}_{i+1}, \mathbf{e}_{i,i+1}^{O}), \end{aligned} \tag{4.17}$$

where $e()$ is an error function corresponding to each edge type, and $\mathcal{G}$ and $\mathcal{O}$ are the sets of factor graph nodes with GNSS-based and VO constraints, respectively.

Chapter 5

Experimental evaluation

5.1 Evaluation of LiDAR SLAM with high-level features

5.1.1 Introduction

This section presents the evaluation of the PlaneLOAM method described in Section 3.2, which uses high-level features as a map representation. The purpose of the experiments was to quantitatively assess the accuracy of the developed system. Specifically, they aimed to determine whether the proposed structure of the global map improves the precision of trajectory estimation. To achieve this, the results from PlaneLOAM were compared with those obtained with the publicly accessible open-source version of the LOAM[1] using the same sequences. The proposed system was tested on three separate datasets with various LiDAR sensors. Moreover, three distinct variants of PlaneLOAM were evaluated: one without loop closures but incorporating high-level features, another that implements loop closures using a basic pose graph, and a third that utilizes the factor graph approach.

Evaluation method

The evaluation methodology is based on the Absolute Trajectory Error (ATE) metrics, introduced in [162], which is widely used for the comparison of SLAM systems. The ATE metrics measure the discrepancy between the given pose of the sensor and its corresponding pose in the ground-truth trajectory by calculating the Euclidean distance between them. These correspondences are identified using the timestamps that were assigned during the scan acquisition process. To compute ATE, it is essential to first align both trajectories by transforming them into a common reference frame. To determine the rigid body transformation between the two sets of poses that represent the estimated and ground-truth paths, the Umeyama algorithm [161] is used. The alignment calculation involves all the estimated poses, as it decreases the overall

[1] This version was obtained from `https://github.com/laboshinl/loam_velodyne`

trajectory deviation. Although it is feasible to utilize only initial poses to determine the transformation, this approach leads to substantial pose discrepancies and erroneously increases the ATE over time.

For quantitative evaluation, the Root Mean Square (RMS) of the ATE metric is calulated for the complete trajectory and denoted by $\mathrm{ATE}_{\mathrm{RMS}}$. Furthermore, the highest local ATE values ($\mathrm{ATE}_{\mathrm{max}}$), along with the standard deviation across the trajectory (σ_{ATE}) are presented. Higher values of σ_{ATE} indicate that the system encounters difficulties with specific segments of the trajectory, although it performs more effectively in other areas.

Datasets

The experiments for evaluation were conducted on two publicly accessible datasets:

- KITTI [163]: Recorded with a Velodyne HDL-64E, mostly in suburban and rural parts of a Karlsruhe city. It is one of the most widely used benchmarks in robotics, computer vision, and autonomous driving for applications such as the evaluation of SLAM algorithms [22, 39, 47, 49]. It contains 22 sequences that are split into two groups: 11 sequences are designated for testing and include ground-truth trajectories, while the other 11 are intended for evaluation. However, the ground-truth data provided by the GNSS/IMU system is less accurate compared to modern high-precision systems, which can affect error evaluation for certain sequences. Although the KITTI benchmark typically includes corrections for motion distortion in LiDAR scans, in the presented experiments, the raw scans without this correction were used. Given that PlaneLOAM is designed for environments with many geometrical features, only three KITTI sequences were chosen for evaluation: 00, 05, and 07. These sequences were recorded in residential areas and therefore include numerous planar and linear structures. These sequences also include loops, enabling the assessment of the feature-based approach to loop closing.
- Mulran [164]: Designed primarily to evaluate place recognition, however, it also incorporates accurate ground-truth trajectories, which facilitates its use in the evaluation of SLAM. Mulran includes sequences collected from four distinct environments, but in this case, the Dajeon Convention Center (DCC) was chosen for the experiments, as it is known for its structural diversity. Moreover, the DCC sequence is characterized by narrow roads with tall buildings and multiple loops, which facilitates the evaluation of the presented loop closure methods. The LiDAR utilized in MulRan is an Ouster OS1-64, while the ground-truth trajectories were generated using SLAM that integrates inertial sensors, GNSS, and ICP scan matching. Although it incorporates multi-sensory data, the proposed approach uses only the raw LiDAR scans and the ground-truth trajectories.

5.1.2 Accuracy of trajectory estimation

Figure 5.1A and Figure 5.1B illustrate the trajectory accuracy for the KITTI 00 sequence, comparing PlaneLOAM with the open-source LOAM system. The plots follow the ATE format

introduced by Sturm *et al.* [162]. Since LOAM does not have the functionalities to detect loop closures, this feature was also disabled in PlaneLOAM for this evaluation. This decision enables a fair comparison of the accuracy between the two systems, as both utilize the same scan-to-scan and scan-to-map matching to estimate the final trajectories. The primary difference lies in the fact that PlaneLOAM uses high-level features, which lead to an improved trajectory and reduced ATE residuals (see Fig. 5.1C). It is also worth mentioning that the open-source version of LOAM achieved results comparable to the results documented in other works [29, 52].

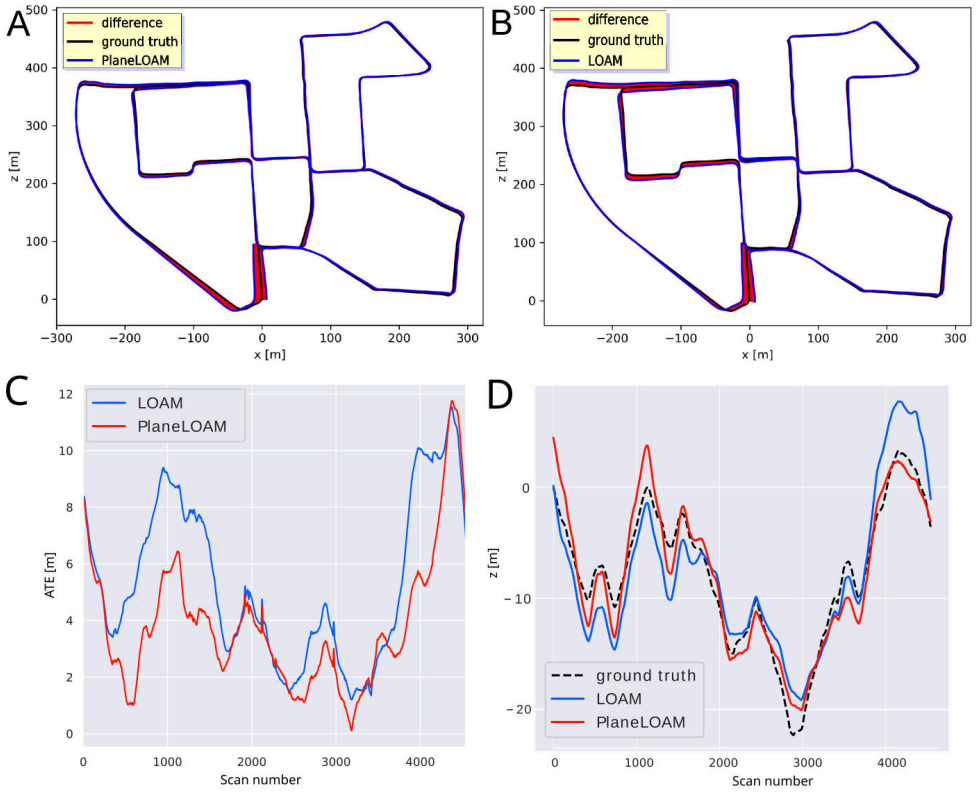

FIGURE 5.1: ATE for the KITTI 00 dataset obtained using a Velodyne HDL-64E LiDAR sensor. Trajectory (A) produced by PlaneLOAM shows smaller deviations from the ground truth compared to the trajectory derived from the open-source LOAM (B), as can be also observed in the plot depicting ATE relative to consecutive scan numbers (C). Reduced error in the z axis enhances the overall performance of PlaneLOAM (D)

A notable benefit of the improved ground plane representation in PlaneLOAM is that it allowed the proposed system to maintain more precise elevation estimates throughout the entire trajectory (see Fig. 5.1D). The numerical results for the KITTI 00, 05, and 07 sequences are presented in Table 5.1. Based on these results, it can be seen that the developed method outperforms open-source LOAM in terms of the ATE metric in all these sequences, recorded in a well-structured urban environment.

Figure 5.2 illustrates the chosen sections of the global maps of high-level features generated by PlaneLOAM. Figures 5.2A and 5.2B show maps for the KITTI 00 sequence, while Figure 5.2C illustrates a segment of the map derived from sequence 07. To enhance clarity, only planar features are presented. Horizontal planar features, such as the ground plane, are depicted in pink. Larger, approximately vertical elements such as walls, and fences are represented in yellow, whereas smaller vertical features containing fewer than 50 points are illustrated in cyan.

TABLE 5.1: Comparison of ATE calculated for chosen sequences (recorded in urban environments) from KITTI and Mulran datasets.

dataset	PlaneLOAM			LOAM (open-source)		
sequence	ATE_{RMS}	ATE_{max}	σ_{ATE}	ATE_{RMS}	ATE_{max}	σ_{ATE}
KITTI 00	4.52 m	11.76 m	2.36 m	6.05 m	11.55 m	2.83 m
KITTI 05	3.21 m	8.04 m	1.59 m	3.39 m	11.26 m	1.86 m
KITTI 07	0.50 m	0.82 m	0.20 m	0.68 m	1.28 m	0.28 m
MulRan DCC	11.02 m	18.17 m	4.46 m	14.81 m	27.95 m	7.24 m

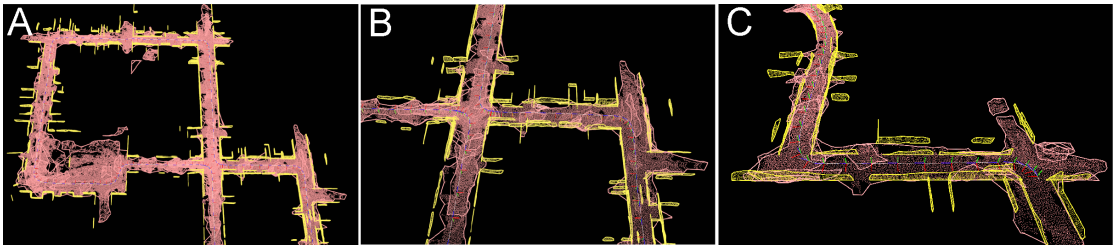

FIGURE 5.2: Segments of the global maps from KITTI 00 (A, B) and KITTI 07 (C) generated by PlaneLOAM, showing high-level geometrical features. Nearly vertical planar attributes are highlighted in yellow, whereas the ground plane is represented by pink patches.

An analogous experiment was conducted for the Mulran DCC sequence. Trajectories with estimated ATE values using LOAM and PlaneLOAM are shown in Figures 5.3A and 5.3B, respectively. In the case of this sequence, the obtained errors are larger than those observed in the KITTI tests for both SLAM systems. However, PlaneLOAM once again provided a more precise trajectory with lower ATE values (see Fig. 5.3C) and improved elevation accuracy (see Fig. 5.3D). The quantitative results of this experiment are also presented in Table 5.1.

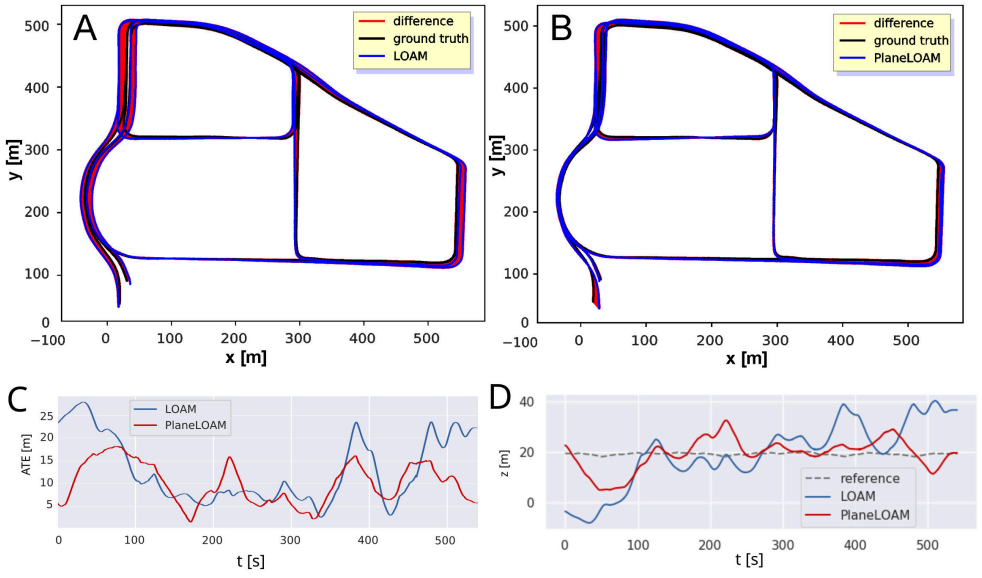

FIGURE 5.3: ATE for the MulRan DCC sequence recorded using an Ouster OS1-64 LiDAR. This sequence features multiple loops, however the PlaneLOAM (A) and LOAM (B) trajectory estimations results were achieved without any loop closure. The plot illustrating ATE values over time (C) demonstrates that PlaneLOAM maintains a smaller overall pose error compared to LOAM throughout the majority of the trajectory, notably providing more accurate estimates of elevation values (D).

5.1.3 SLAM with high-level features in different environments

An essential part of the conducted research is to demonstrate the effectiveness of the proposed approach to LiDAR SLAM. This involves using different 3D LiDARs in scenarios typical for vehicle localization in urban traffic, such as for city buses. Consequently, PlaneLOAM was also tested on sequences collected during independent experiments. One of these sequences was recorded using a car with a roof-mounted Sick MRS-6124 LiDAR, characterized by 24 scanning lines and 120° forward-looking field of view. It also includes data from RTK GNSS u-blox ZED-F9P module [165], which achieves a centimeter-level positioning accuracy at 10 Hz, provided that it observes a sufficient number of satellites. This configuration was used in a combined outdoor and indoor experimental scenario conducted in the large Posnania shopping center located in Poznan. The car started on a public road near Posnania, proceeded to the entrance of the underground parking lot, navigated through the underground area, and exited onto a different public road on the opposite side of the mall. Note that GNSS-based ground-truth data were recorded only for the outdoor sections of this experiment. Nevertheless, the goal was to show that PlaneLOAM can effectively handle such mixed scenarios and resume tracking the reference trajectory with acceptable errors after traversing the underground environment.

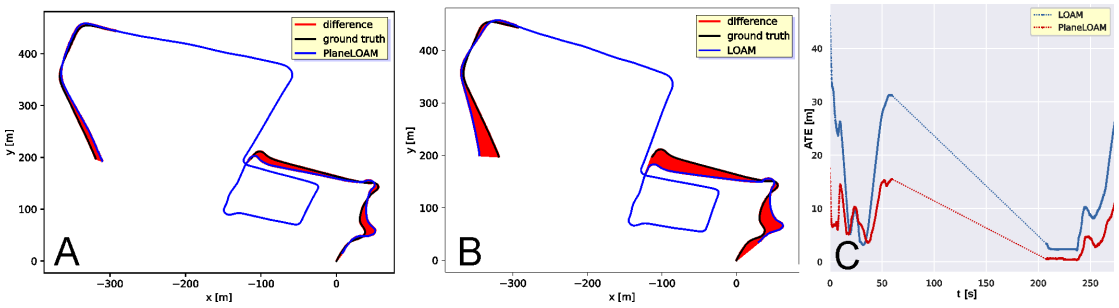

FIGURE 5.4: Results of an experiment in the mixed outdoor/indoor environment of the Posnania shopping mall. The trajectories estimated by PlaneLOAM (A) and LOAM (B) were based on scans from a Sick MRS-6124 LiDAR. Although GPS ground truth was available only for the outdoor portion, the ATE plot (C) demonstrates that PlaneLOAM maintained significantly lower drift while navigating the underground parking lot.

Figures 5.4A and 5.4B illustrate the positional errors for this sequence for PlaneLOAM and LOAM, respectively. The change in ATE values over time, shown in Figure 5.4C, reveals that LOAM generally exhibits a much higher inaccuracy for most of the route. The largest error at the beginning of the sequence comes from the trajectory alignment used to compute the ATE metrics and generate the plot. This implies that LOAM experienced a substantial orientation error at the beginning of the sequence, and aligning the trajectories using only the initial segment would lead to an even larger discrepancy in the rest of the plot. However, the primary focus of the evaluation is put on absolute pose errors relative to an external reference system. A trajectory with low ATE for consecutive sensor poses implicitly ensures that the corresponding global map is accurate. This consistency is not captured by relative error metrics, such as those used by Geiger *et al.* [163] or the Relative Trajectory Error (RPE) defined along with ATE in [162]. To highlight that PlaneLOAM generates globally consistent maps suitable for tasks like motion planning, Figure 5.5A illustrates the entire map estimated in the *Posnania* experiment. As shown in Figure 5.5B, the map is rich in geometric details, accurately representing minor

architectural elements, such as pillars, even with the limited scanning density and horizontal field of view of the Sick LiDAR.

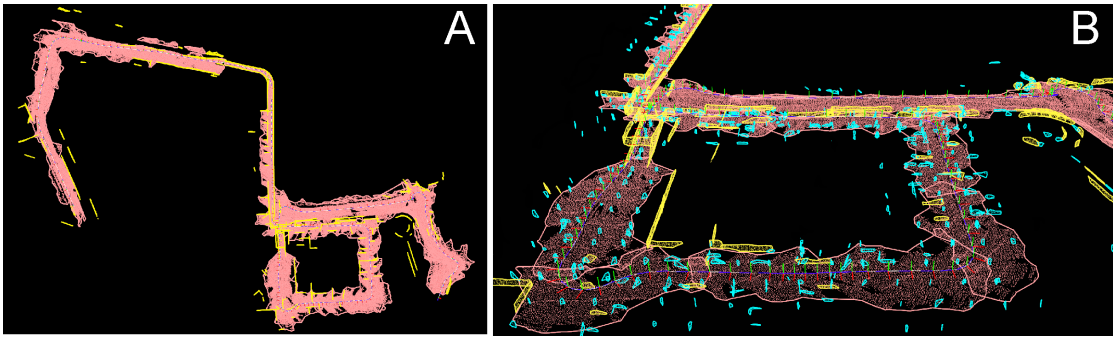

FIGURE 5.5: Visualization of the complete global map generated by PlaneLOAM during the *Posnania* experiment (A), along with a close-up of its central section (B), highlighting the detailed structure of the underground parking lot, including walls (yellow), smaller vertical features like pillars (cyan), and the ground plane (pink).

The second experiment involved a city bus equipped with the same Sick MRS-6124 LiDAR and a GNSS receiver (see Fig. 5.6A). Data were collected in a small town near Poznan, characterized by moderately structured environments with narrow roads bordered by residential buildings, as well as numerous trees and bushes (see Fig. 5.6B). The relatively low-rise surroundings allowed for a strong GNSS signal across the entire area. These experiments were carried out in February under cloudy conditions with occasional rain, which did not appear to impact the performance of the Sick LiDAR.

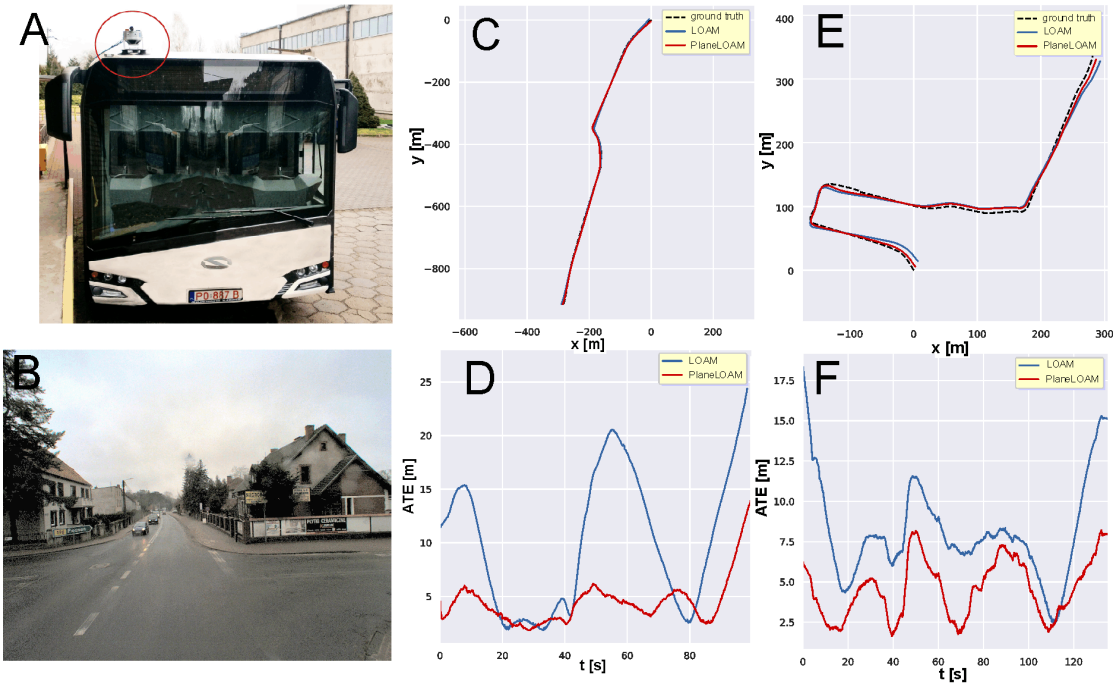

FIGURE 5.6: Results from localization experiments using a city bus equipped with a single Sick MRS-6124 LiDAR (A). The two sequences, *TownCentre* (C, D) and *TownSuburbs* (E, F), were recorded in a moderately structured environment of a small town (B). The trajectories estimated by PlaneLOAM align more closely with the ground truth compared to those of open-source LOAM (C, E), with PlaneLOAM showing significantly lower ATE values (D, F)

The estimated trajectories from PlaneLOAM and LOAM are compared to the ground truth in Figures 5.6C and 5.6E for the *TownCentre* and *TownSuburbs* experiments, respectively. To maintain clarity of the figure, the ATE values are shown separately in Figures 5.6D and 5.6F. PlaneLOAM outperformed LOAM in both sequences, despite the less structured environment and the additional sensor motion noise caused by the fact that LiDAR was mounted high on the roof of the bus. ATE statistics for all three scenarios are summarized in Table 5.2.

TABLE 5.2: Comparison of ATE metrics for sequences captured using a Sick MRS-6124 LiDAR

dataset	PlaneLOAM			LOAM (open-source)		
sequence	ATE_{RMS}	ATE_{max}	σ_{ATE}	ATE_{RMS}	ATE_{max}	σ_{ATE}
TownCentre	4.93 m	14.05 m	2.10 m	6.05 m	24.90 m	6.48 m
TownSuburbs	4.91 m	8.22 m	1.79 m	8.71 m	18.37 m	3.04 m
Posnania	7.40 m	17.50 m	4.58 m	15.90 m	46.13 m	9.99 m

5.1.4 Analysis of the computation time

The new map structure introduced in PlaneLOAM affects the computational complexity associated with processing a single scan. While these changes are negligible in the laser odometry thread, which largely follows the implementation of the open-source LOAM, they have a significant impact on the mapping thread responsible for producing more accurate pose estimates. Therefore, this section presents a detailed analysis of the computation time for individual processing steps in the mapping thread, comparing PlaneLOAM with the open-source LOAM. The side-by-side comparison, presented in Table 5.3, highlights that despite the increased complexity of the map structure, the use of high-level geometric features enables faster processing. This is mainly because fewer individual elements need to be searched and compared.

The analysis was conducted on an Intel Core i7-9700TE CPU, using the Mulran DCC sequence. This sequence is considered the most demanding among those used for evaluation due to the large point clouds produced by the Ouster OS1-64 LiDAR and the complex environment, which includes both planar and linear features. The times reported in Table 5.3 are averaged over 5542 scans. The observed standard deviation in computation time was negligible and, in PlaneLOAM, primarily influenced by the number of features generated per scan. Detailed descriptions of the PlaneLOAM processing steps in Table 5.3 are provided in Section 3.2.2, while their LOAM counterparts can be found in [34].

The reduced processing time in PlaneLOAM compared to LOAM is mainly caused by the much shorter time required to build the kd-trees. PlaneLOAM uses local kd-trees to identify points for computing point-to-feature constraints in optimization, but avoids using them during feature creation and updating. In PlaneLOAM, kd-trees contain only the points that make up the high-level features, reducing the number of points that are processed. In addition, features inactive at a given time are excluded from matching operations, saving time in computing point-to-feature constraints.

The computation time comparison excludes the loop-closing module, as it is not implemented in the LOAM. The SegMap method, which detects loop closures, has an average inference time of 155 milliseconds for the Mulran DCC sequence. In contrast, pose graph optimization using g^2o

TABLE 5.3: Comparison of the computation time required to process a single scan in the mapping thread between PlaneLOAM and LOAM

processing step	sub-step	computation time [ms]	
		PlaneLOAM	LOAM (open-source)
Point clouds preparation		8.18	7.70
Construction of kd-tree		2.49	75.80
Point-to-line constraints		14.05	18.77
	kd-tree search	8.45	10.87
	other computations	5.60	7.90
Point-to-plane constraints		13.11	18.30
	kd-tree search	8.22	9.70
	other computations	4.89	8.60
Optimization time		0.45	0.57
Point clouds post-processing		34.71	32.88
Processing planar features		35.77	not applicable
	adding points	22.76	
	updating features	2.22	
	deleting features	0.02	
	merging features	10.77	
Processing linear features		34.95	not applicable
	adding points	15.53	
	updating features	16.65	
	deleting features	0.05	
	merging features	2.72	
Total time per scan		143.71	154.02

requires between 60 and 240 seconds, depending on the number of poses in the graph. While pose graph-based loop closures cannot be performed in real-time, delays of a few minutes are acceptable for online operation, provided that they are handled by a separate background thread. Full factor graph optimization is even more time-intensive and takes between 280 and 500 seconds for the examined sequences. However, full optimization incorporates information about multiple loop closures and is performed offline, which allows PlaneLOAM to create highly accurate maps of the environment that are suitable, for example, for motion planning.

5.1.5 Approaches to loop closing in feature-based LiDAR SLAM

The third set of experiments examines the integration of the SegMatch/SegMap loop detection with PlaneLOAM. Most LiDAR SLAM systems that implement loop detection employ a pose graph approach and form edges between pose nodes using relative transformations between the corresponding locations. These constraints are often derived using the ICP [15] or directly from the SegMatch method [37]. In contrast, the proposed approach leverages high-level feature-based map representation to process detected loop candidates in a manner analogous to the loop closure strategies used in visual SLAM, by associating individual features and optimizing the factor graph in a BA-like framework.

Although there are several techniques to detect loop closures using LiDAR range data (see [16] for a comparative review), these methods are less effective than the applied SegMap approach or require specific map representations [21, 22], which cannot be implemented within PlaneLOAM.

Thus, this section focuses on comparing different methods for utilizing detected loop closures to improve trajectory and map consistency rather than comparing various loop detection methods.

The pose-based approach closely resembles the method outlined in [52], as it utilizes either PlaneLOAM or LOAM to estimate the LiDAR odometry between successive sensor poses (graph nodes) and to perform motion compensation in 3D scans. Loop closures are identified using the SegMap algorithm and incorporated as constraints within the pose graph for optimization. In contrast, the BA-like approach is applied exclusively to PlaneLOAM, as the original LOAM lacks a map structure that supports feature-level matching. The trajectories estimated without loop closing for LOAM and PlaneLOAM are shown in Figures 5.7A and 5.7D for KITTI 00 and MulRan DCC, respectively. The pose-based loop closing results are shown in Figures 5.7B and 5.7E, while results for the BA-like approach in PlaneLOAM, which uses high-level feature alignment, are illustrated in Figure 5.7C for KITTI 00 and Figure 5.7F for MulRan DCC. Based on these results, it can be noted that the pose graph approach significantly improves accuracy over the open-loop version for both LOAM and PlaneLOAM. Quantitative results for KITTI 00 show a 13% improvement in ATE_{RMS} for PlaneLOAM and 28% for LOAM (see Tab. 5.4). LOAM experiences a greater relative improvement due to its poorer baseline performance, although PlaneLOAM achieves better absolute accuracy in all scenarios. In the MulRan DCC case (see Tab. 5.5), the benefits of loop closure are even greater, with improvements of approximately 32% for PlaneLOAM and 34% for LOAM. The DCC sequence presents greater environmental diversity compared to KITTI 00, making it less suitable for high-level feature mapping, which explains the smaller performance gap between PlaneLOAM and LOAM. However, PlaneLOAM continues to demonstrate the lowest absolute error values in all configurations.

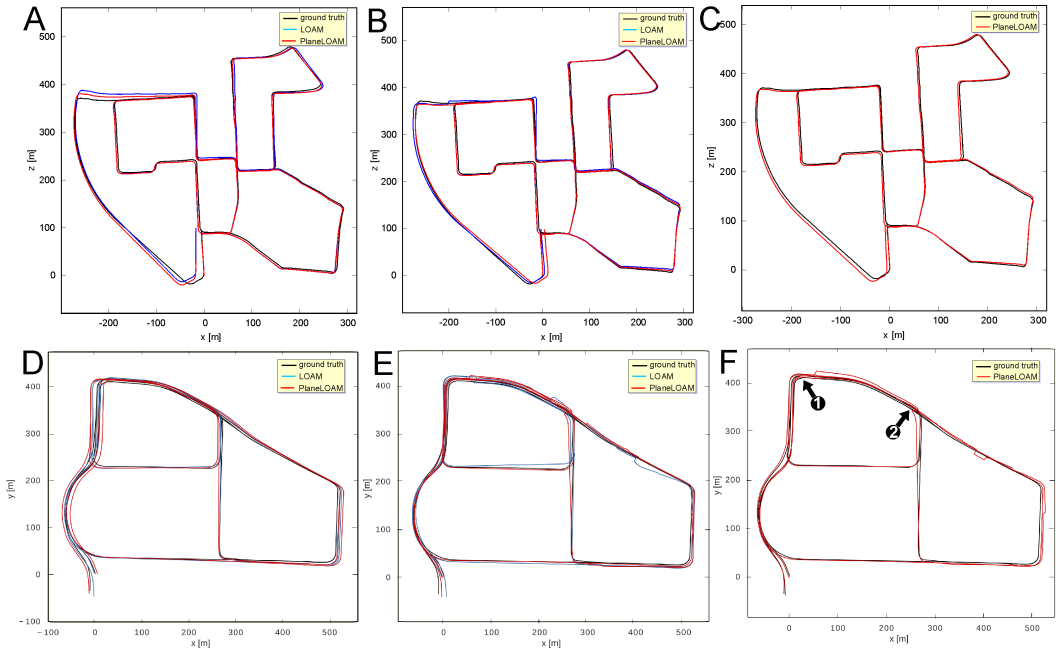

FIGURE 5.7: Estimated trajectories for the KITTI 00 (A, B, C) and MulRan DCC (D, E, F) sequences using PlaneLOAM and open-source LOAM systems. Scenarios include no loop detection (A, D), SegMap loop detection with pose graph (B, E), and high-level feature alignment with BA optimization (C, F). The arrows indicate the locations where the point clouds depicted in Figure 5.8 were obtained.

Table 5.4: Comparison of ATE for different approaches to loop closing for the KITTI 00 sequence

approach to	PlaneLOAM			LOAM (open-source)		
loop closing	ATE_{RMS}	ATE_{max}	σ_{ATE}	ATE_{RMS}	ATE_{max}	σ_{ATE}
none	4.52 m	11.76 m	2.36 m	6.05 m	11.55 m	2.83 m
pose graph	3.96 m	9.68 m	2.17 m	4.34 m	11.15 m	2.38 m
features alignment + BA	2.47 m	7.11 m	1.48 m	N/A	N/A	N/A

Table 5.5: Comparison of ATE obtained for various loop closing approaches for the MulRan DCC sequence

approach to	PlaneLOAM			LOAM (open-source)		
loop closing	ATE_{RMS}	ATE_{max}	σ_{ATE}	ATE_{RMS}	ATE_{max}	σ_{ATE}
none	11.02 m	18.17 m	4.46 m	14.81 m	27.95 m	7.24 m
pose graph	7.41 m	19.23 m	3.59 m	9.77 m	27.51 m	4.26 m
features alignment + BA	5.40 m	13.78 m	2.01 m	N/A	N/A	N/A

The accuracy of the trajectory estimation improves significantly when loop closing is achieved using feature matching capabilities of PlaneLOAM, combined with BA-like optimization. This method refines not only the sensor poses but also the locations of features. For the KITTI 00 sequence, this approach results in an improvement of 45% over the baseline in terms of ATE_{RMS}, while for the MulRan DCC sequence, the improvement reaches 51%. These results align with previous conclusions, as PlaneLOAM performs better on KITTI than on MulRan, potentially amplifying the impact of loop closures. Nonetheless, the use of high-level feature-based representation proved its effectiveness in performing loop closures even in the MulRan DCC scenario, where loop detection often relies on features such as trees and vegetation. Despite this, the geometric structures present in the scene still provide sufficient constraints to enhance trajectory estimation.

In addition, the loop closure method, which leverages BA-like optimization, is capable of closing multiple loops of varying sizes within a single optimization session. This is particularly useful when a test vehicle follows a closed path and revisits specific locations multiple times. Such a scenario occurred in the Mulran DCC sequence, where the vehicle returned to the same location (indicated by the arrow ① in Figure 5.7F) four times. This resulted in closing loops of different scales: a smaller loop along the inner, nearly circular path, and a much larger loop encompassing the entire DCC area along the outer path. Loop closures succeeded even though the vehicle approached the same location from different directions, as illustrated by the example point clouds from Ouster OS1-64 in Figures 5.8A and 5.8B. Furthermore, the loop closure module demonstrated resilience to moderate changes in the appearance of the environment between observations, as shown by another example from the Mulran DCC experiment (indicated by the arrow ② in Figure 5.7F) and depicted in Figures 5.8C and 5.8D. These results validate the effectiveness of combining location matching with learned SegMap descriptors and BA-like optimization using features, at it establishes a robust and adaptable loop closure mechanism suitable for autonomous driving and large-scale environment mapping.

Finally, the trajectory estimation results for the KITTI 00 sequence were evaluated by comparing different loop closure approaches using the evaluation method originally introduced with the KITTI dataset [163]. Unlike ATE metrics, this method calculates separate translation and

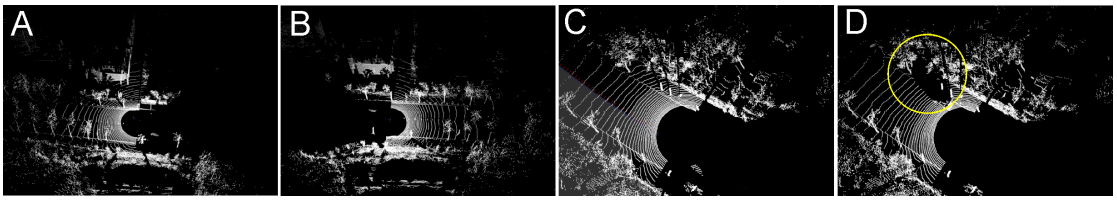

FIGURE 5.8: Visualization of example point clouds from the Mulran DCC sequence: a location visited four times while traversing a circular path with successful loop closures in spite of the reversed motion direction (A, B) and a location where the environment has changed due to a removed vehicle (encircled), but a loop closure was still detected (C, D)

rotation errors relative to the traveled distance and presents them as functions of the trajectory segment length. Although these relative accuracy metrics do not directly reflect errors in the overall trajectory shape, they align with the findings in [52], showing that both translation (see Fig. 5.9A) and rotation (see Fig. 5.9B) errors decrease over longer paths. This trend occurs because for shorter paths, the results depend more heavily on open-loop LiDAR odometry. Translation error plots exhibit a similar descending trend for longer routes when using both LOAM and PlaneLOAM with the pose-graph approach, though PlaneLOAM achieves lower error values due to superior odometry estimates. Leveraging high-level features for loop closure yields even better performance, but only for longer paths, as shorter segments rely on the same LiDAR odometry as the baseline method. It is also important to note that the detected loop closures are not evenly distributed along the trajectory. As the vehicle travels larger distances, the number of detected loop closure candidates increases significantly (see Fig. 5.9C) because the vehicle reenters previously mapped areas.

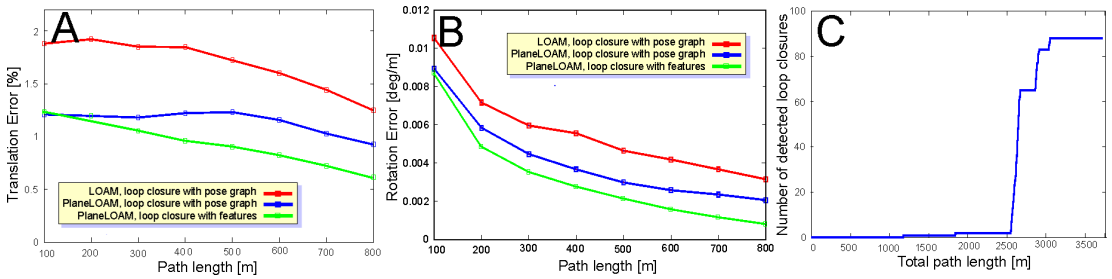

FIGURE 5.9: The impact of different loop closing methods on trajectory estimation accuracy for the KITTI 00 sequence, presented using the standard KITTI evaluation framework [163], which separates translation (A) and rotation (B) errors. Additionally, the number of detected loop closure candidates grows substantially as the vehicle travels along its route and revisits previously mapped regions (C).

5.2 Evaluation of surfel-based map representation

5.2.1 Introduction

This section presents the evaluation of the MAD-BA method described in Section 3.3, which utilizes surfels as a map representation. To quantitatively validate this approach, the ATE is used to evaluate the pose estimation and the Chamfer-L1 distance to assess the accuracy of map reconstruction. The experiments were conducted on a PC equipped with an Intel Core i9-9900KF

CPU @ 3.60GHz and 64GB of RAM. The presented methodology employs a LiDAR BA process to refine both poses and scene geometry, starting from an initial set of poses and point clouds derived from LiDAR odometry or SLAM, with a minimal set of parameters. The results indicate that the proposed LiDAR BA approach outperforms existing methods in terms of accuracy. The experimental results are validated in various scenarios, such as urban traffic, tight corridors, stairways, and in both static and dynamic conditions. The evaluation also includes comparisons with other LiDAR-based global refinement strategies such as BALM2 [73] and HBA [72].

Datasets

The comprehensive experimental campaign was conducted using the following datasets:

- Newer College (NC) [166]: Collected using a handheld Ouster OS0-128 LiDAR in environments featuring structured and vegetated areas. It is suitable for evaluating the accuracy of the trajectory and the map, as it includes ground-truth data for both, created using the Leica BLK360 scanner. According to the authors, the provided ground truth achieves centimeter-level accuracy over poses and points in the map.
- Vision Benchmark in Rome (VBR) [167]: Recorded mostly in Rome using OS1-64 for handheld and OS0-128 for car-mounted sequences. It adds diversity to the evaluation by incorporating complex urban scenarios, with many narrow streets and dynamic objects, such as vehicles and pedestrians. The provided ground-truth trajectory was obtained by fusing LiDAR, IMU, and RTK GNSS data, ensuring centimeter-level accuracy.
- KITTI [163]: One of the most widely used benchmarks in robotics, previously introduced in Section 5.1.1. For this evaluation, sequences 00–10 were utilized, excluding sequences 01 and 02 recorded on a highway.

Evaluation method

The goal of the developed system is to improve the consistency of both the structure and poses. Therefore, the quantitative evaluations were performed using the ATE_{RMS} for $SE(3)$ poses, and metrics such as accuracy, completion, Chamfer-L1 distance, and F-score for the map. The first three parameters for map evaluation assess geometric discrepancy by measuring the average closest-point distance between the reconstructed map and a reference model, while the F-score evaluates their alignment. For pose evaluation, the method previously introduced in Section 5.1.1 was applied: the output trajectory is initially aligned with the ground truth using timestamps to determine the necessary associations. Then, the RMS of error is calculated based on the translational differences between all the matched poses. Trajectory evaluation is performed on three datasets, while map evaluation is limited to the NC dataset, which is the only one offering a high-resolution ground-truth map. To evaluate the reconstructed map $\mathcal{Q}$ against the reference model $\mathcal{R}$, accuracy, completion, and Chamfer-L1 distance are calculated as follows:

$$d_{\text{C-L1}}(\mathcal{Q},\mathcal{R}) = \underbrace{\frac{1}{2N_q}\sum_{\mathbf{p}_q\in\mathcal{Q}}\min_{\mathbf{p}_r\in\mathcal{R}}\|\mathbf{p}_q-\mathbf{p}_r\|}_{\text{accuracy}} + \underbrace{\frac{1}{2N_r}\sum_{\mathbf{p}_r\in\mathcal{R}}\min_{\mathbf{p}_q\in\mathcal{Q}}\|\mathbf{p}_r-\mathbf{p}_q\|}_{\text{completion}} \tag{5.1}$$

where p_q, p_r denote the coordinates of points and N_q, N_r represent the number of points in $\mathcal{Q}$ and $\mathcal{R}$, respectively.

Methods chosen for comparison

Two distinct approaches were selected for comparison with the proposed method. The first is BALM2 [73], a global LiDAR refinement technique that minimizes the distance from the feature points to the corresponding edges or planes. Unlike visual SLAM, where features are jointly optimized with poses, BALM2 simplifies the process by analytically eliminating features from the optimization, thereby reducing computational complexity. However, this method focuses solely on pose refinement without optimizing scene geometry, limiting its ability to address structural inconsistencies in the map. Additionally, its dependence on an adaptive voxelization method for feature correspondence limits its scalability in highly dynamic or complex environments.

The second approach is HBA [72], designed to address the computational challenges of LiDAR BA on large-scale maps. HBA employs a hierarchical framework that combines bottom-up BA with top-down pose graph optimization. Although this design improves efficiency by reducing the optimization scale, the reliance on pose graph optimization decouples trajectory estimation from structure refinement, which can lead to inconsistencies in the final map, especially in the case of long trajectories or complex environments.

The second approach is HBA [72], which is designed to decrease the computational demands of LiDAR BA on large-scale maps. HBA utilizes a hierarchical framework that combines bottom-up BA with top-down pose graph optimization. This approach enhances efficiency by reducing the optimization scale, but its reliance on pose graph optimization decouples trajectory estimation from structure refinement. This decoupling can result in discrepancies in the final map, especially for long trajectories or in complex environments.

In contrast, the developed method integrates both pose and structure optimization within a unified BA framework, avoiding the decoupling issues found in both BALM2 and HBA.

Parameters

All evaluated systems require configuration of particular parameters that influence their performance. In the developed approach, the key parameters are the threshold values used to identify correspondences between surfels in the data association step:

- $d_e = 0.5\,\text{m}$ - the maximum allowable distance between surfel and a leaf considered for matching,
- $d_n = 1.0\,\text{m}$ - the maximum distance between the surfel and a leaf, computed along the surfel's normal,

- $\alpha = 5^\circ$ - the maximum angle between the normals of a surfel and a leaf,
- $\rho_{\text{ker}} = 0.1\,\text{m}$ - the threshold for the Huber robust estimator used during optimization of a factor graph,

and for constructing the kd-trees:

- $b_{\text{max}} = 0.2\,\text{m}$ - the maximum size of kd-tree leaves,
- $b_{\text{min}} = 0.1\,\text{m}$ - the minimum flatness of a leaf below which its normal is propagated to its children.

These default parameter values were used consistently during the evaluation to ensure reliable results across various sequences and datasets. For the HBA method [72], also the default parameter values were used, while for BALM2 [73], the initial `voxel_size` was changed from 2.0 m to 1.0 m to improve its performance.

5.2.2 Results of trajectory evaluation

Table 5.6 presents a comprehensive summary of the evaluation results for all presented methods and datasets. The first column shows the error associated with the input trajectory, which is necessary as the initial guess for all tested systems. For this purpose, again the open-source implementation of LOAM [34] was utilized, as it is a widely used framework for LiDAR-only odometry and mapping. However, the results for KITTI sequences 01 and 02 have been excluded from the table, as LOAM was unable to generate estimates for these specific trajectories.

		Initial trajectory		BALM2 [73]		HBA [72]		Pose-only		MAD-BA w/o uncert.		MAD-BA with uncert.	
	Sequence	RMS ↓	MAX ↓	RMS ↓	MAX ↓	RMS ↓	MAX ↓	RMS ↓	MAX ↓	RMS ↓	MAX ↓	RMS ↓	MAX ↓
NC [166]	quad-easy	0.098	0.218	0.093	0.249	0.131	0.801	0.086	0.214	0.086	0.183	**0.083**	0.177
	maths-easy	0.090	0.189	0.162	0.420	0.135	0.606	0.087	0.189	0.054	0.163	**0.043**	0.176
	cloister	0.119	0.297	0.319	1.823	0.466	2.745	**0.098**	0.364	0.112	0.356	0.106	0.318
	stairs	0.368	0.717	0.366	0.700	3.104	9.088	0.375	1.429	**0.067**	0.126	0.070	0.132
	underground-easy	0.127	0.358	0.120	0.342	0.434	1.019	0.080	0.352	0.073	0.330	**0.072**	0.331
VBR [167]	Colosseo	2.920	5.873	2.877	5.553	1.123	2.763	2.891	5.887	0.698	1.585	**0.565**	1.161
	Spagna	0.963	1.912	1.058	5.479	1.628	4.625	1.053	3.953	0.412	1.229	**0.155**	0.522
	DIAG	0.341	0.804	0.425	1.945	0.960	2.510	0.335	2.014	**0.101**	0.438	0.103	0.455
	Campus	1.777	4.928	2.121	6.587	1.055	3.944	1.741	5.214	**1.025**	2.972	1.109	2.973
	Ciampino	6.078	16.097	6.086	15.669	5.641	16.645	6.040	15.881	3.878	13.711	**3.123**	9.566
	Pincio	1.488	2.914	1.719	3.293	1.508	4.589	1.499	3.304	**0.914**	3.177	1.121	3.869
KITTI [168]	00	6.051	11.552	5.585	16.075	4.899	9.897	5.145	11.326	3.717	7.657	**3.544**	6.473
	03	1.005	1.873	1.144	2.522	4.166	11.279	1.176	2.817	1.002	1.609	**0.983**	1.669
	04	0.444	1.162	0.507	1.226	0.532	1.776	0.399	1.075	**0.411**	1.111	0.521	1.371
	05	3.392	11.263	3.001	9.624	**1.321**	6.083	3.006	10.290	1.335	2.497	1.426	2.682
	06	1.400	2.039	2.498	19.379	**0.668**	1.488	1.350	2.179	0.856	1.747	0.892	1.819
	07	0.684	1.281	0.748	1.197	**0.289**	0.602	0.576	0.791	0.392	0.765	0.395	0.704
	08	4.034	12.668	8.198	67.587	4.553	12.815	**4.027**	12.319	6.535	16.982	5.449	14.873
	09	2.505	7.304	8.858	57.515	3.410	6.084	2.487	7.096	**1.343**	2.778	1.424	2.886
	10	2.024	3.720	2.725	10.738	4.179	8.107	2.051	3.807	**1.854**	4.193	2.191	4.422
	avg	1.741	4.289	2.430	11.396	2.010	5.373	1.725	4.525	1.243	3.181	**1.169**	**2.829**

TABLE 5.6: Quantitative results of ATE for all tested methods across different publicly available datasets. Each error is reported in m, and the last row of the table shows the total average of the error in each column. The best results for each sequence (the lowest RMSE) are shown in bold.

It is worth noting that the quality of the initial trajectory plays a significant role in the performance of refinement systems. An inaccurate initial estimate can result in inaccurate scan

correspondences, leading the optimization process to converge to a local minimum. This limitation is particularly apparent in the results for BALM2 [73], as shown in the second column of the table. BALM2 frequently exhibits errors that are similar to or even worse than those of the initial trajectory. This suggests that its voxel-based discretization method faces difficulties when dealing with the suboptimal starting trajectory provided by LOAM.

HBA [72], whose outcomes are presented in the third column, demonstrate improved performance in certain sequences, such as KITTI 05, 06, and 07, achieving the lowest error in these cases. However, its inconsistent results across other datasets highlight its sensitivity to the initial trajectory as well. HBA method is based on multi-view ICP, but in addition, it also incorporates PGO to simplify optimization. Even though this approach is typically faster, it sacrifices some accuracy compared to BA. Although parameter tuning could enhance the performance of both BALM2 and HBA, a consistent configuration across all sequences was used to ensure a fair and unbiased evaluation, thereby emphasizing the inherent robustness of each system.

The last three columns in Table 5.6 show the results for various versions of the developed system. In first version the system is configured to optimize only the poses and to keep all surfels fixed. The factor graph structure remains the same as in the other versions, but in this case, no updates are made to the surfels during the optimization process. The two subsequent columns illustrate the results of the BA method, showing the effects of integrating uncertainty into the optimization. In the first case, the uncertainty value $\sigma_l = 1$ (see Eq. 3.29) is applied to all measurements, giving them equal weight during optimization. In the second case, σ_l is determined by the uncertainty estimated for each measurement during the kd-tree creation, as explained in Section 3.3.3. This approach assigns greater weight to measurements with higher confidence, leading to improvements in both pose accuracy and map consistency. Based on the obtained results, it can be stated that both BA configurations significantly reduce errors compared to the fixed surfel approach. Moreover, incorporating uncertainty into the optimization improves overall results by effectively using weighted measurements. In addition, Figure 5.10 shows that the joint optimization of surfels and poses reduces the number of iterations required for convergence, compared to the pose-only variant.

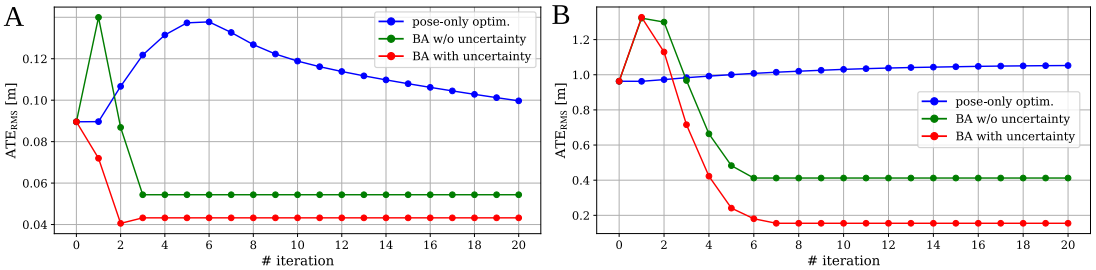

FIGURE 5.10: The ATE computed for each iteration of the developed BA algorithm. The plots compare three versions of the system for the *math-easy* (A) and *Spagna* (B) sequences. The results indicate that both variants of BA not only lower the error but also accelerate the algorithm convergence relative to pose-only optimization. The version with uncertainty produces best results, as measurements with higher confidence are given greater weights.

5.2.3 Results of map evaluation

The map evaluation involved comparing the ground-truth model with the surfel maps generated using the initial trajectory and those produced by MAD-BA with and without uncertainty information. The numerical values of the computed metrics are shown in Table 5.7. For comparison, the table also includes results from the point-based map generated by the HBA system. The *underground-easy* sequence is excluded from this comparison because a ground-truth map is not available for this part of the dataset. To exclude non-overlapping sections of the refined and reference maps, a threshold was used for the maximum allowable distance between points in these maps [67]. This threshold was set to 1 m, however, similar results are obtained with different threshold values, as illustrated in Figure 5.11. Based on the presented results, it can be seen that the incorporation of uncertainty improves the performance for all calculated metrics. The only exception is the *stairs* sequence, which contains narrow corridors where individual measurements are already highly reliable. Moreover, the part of the reference map corresponding to *stairs* sequence has some visible artifacts, such as double walls, which could also cause increased error.

	initial map				HBA				MAD-BA w/o uncert.				MAD-BA with uncert.			
Sequence	acc.↓	comp.↓	C-L1↓	F-sc.↑	acc.↓	comp.↓	C-L1↓	F-sc.↑	acc.↓	comp.↓	C-L1↓	F-sc.↑	acc.↓	comp.↓	C-L1↓	F-sc.↑
quad_easy	16.39	22.52	19.45	68.35	14.59	42.93	28.76	41.41	12.72	6.97	9.85	92.04	12.70	6.93	**9.82**	**92.07**
math_easy	19.22	26.87	23.05	58.55	16.86	43.49	30.17	38.70	14.44	10.65	12.55	87.20	14.35	10.54	**12.44**	**87.26**
cloister	22.78	16.31	19.54	69.18	18.62	39.18	28.90	45.76	21.78	7.34	14.56	77.86	21.18	6.95	**14.06**	**78.96**
stairs	28.91	23.66	26.29	55.51	33.82	32.35	33.08	41.41	24.32	7.22	**15.77**	**73.50**	24.56	7.28	15.92	73.36

TABLE 5.7: The quantitative results of the map evaluation. It includes the following metrics: accuracy (acc.), completion (comp.), Chamfer-L1 (C-L1), and F-score (F-sc.). The first three metrics are reported in centimeters, whereas the F-score is given as a percentage. The column labeled *initial map* shows the results for the surfel map created using the initial trajectory obtained using LOAM. An additional threshold of 0.2 m was applied to calculate the F-score. The best results are highlighted in bold.

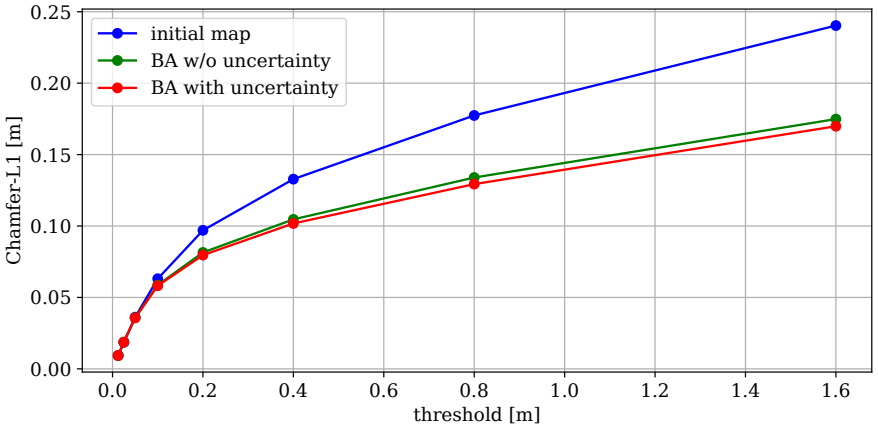

FIGURE 5.11: Chamfer-L1 distance calculated for the *cloister* sequence for different distance thresholds. The initial map was generated using the LOAM trajectory. Both versions of the proposed BA significantly improve the map quality, however, the incorporation of uncertainty information provides additional enhancements.

Furthermore, a comparison of the maps obtained from the NC *maths-easy* and *quad-easy* sequences was generated and presented in Figure 5.12. It shows the Euclidean distance calculated

from each point in the refined map to the nearest point in the reference model. These comparisons demonstrate a notable enhancement in map quality after applying the VA method, which effectively integrates measurements from different scans, leading to a more precise and reliable model of the environment.

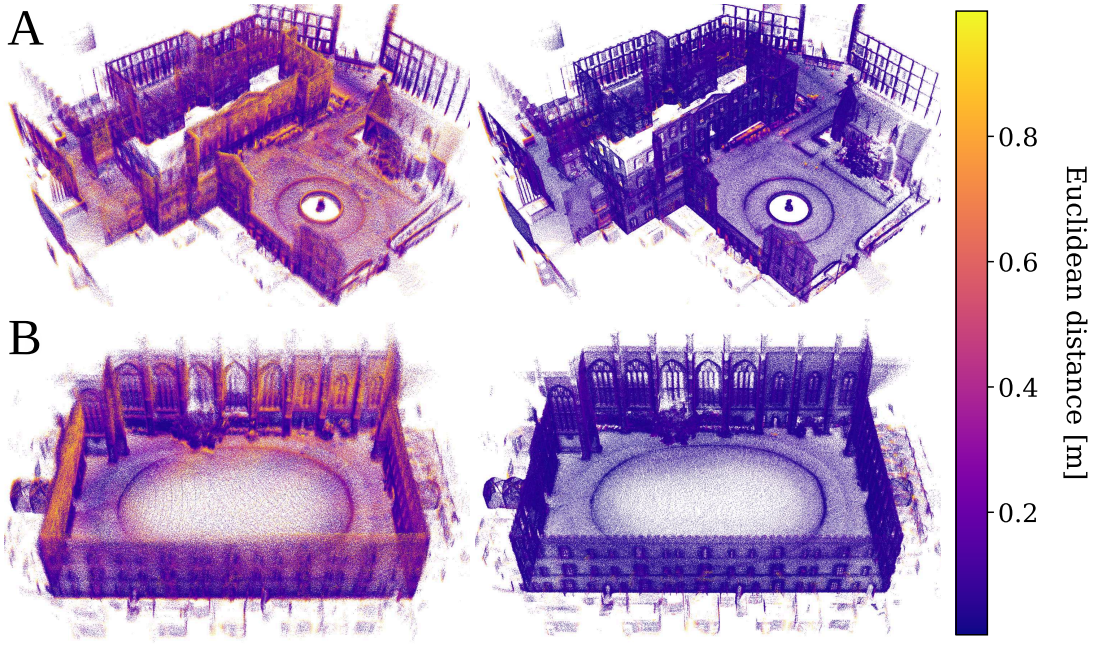

FIGURE 5.12: Qualitative results of maps generated from the NC *math-easy* (A) and NC *quad-easy* (B) sequences. The color of each surfel point indicates its Euclidean distance to the nearest point in the ground-truth map. The images on the left illustrate maps created using the initial trajectory (before optimization), while the right images show the results after applying BA with uncertainty method.

A further benefit of the proposed map representation is its ability to implicitly exclude points associated with dynamic objects, such as moving cars or pedestrians. Although these filtering characteristics are hard to measure, qualitative examples of this effect are provided using a part of the VBR *Spagna* and KITTI 07 sequences in Figure 5.13.

This filtering effect is related to the way in which surfels are created and how they associate leafs from different scans. When the given object is moving fast enough, there are no corresponding leaves in subsequent scans, which effectively excludes them from the surfel creation process. This leads to the inherent filtering properties of this approach, enabling it to automatically handle dynamic objects in the environment. The exclusion of these points offers substantial advantages for applications such as autonomous driving and robotic navigation. Since moving objects do not constitute permanent features of the environment, their removal enhances the accuracy of the map, also leading to improved localization performance. This is crucial for tasks that require precise navigation based on a consistent representation of the environment. Additionally, the number of surfels in a map is much lower than points in the original scans, which reduces its size, while still preserving the important structure of the surroundings. As a result, filtering out dynamic points not only improves the overall performance of the system, but also makes it more robust in environments with many dynamic objects.

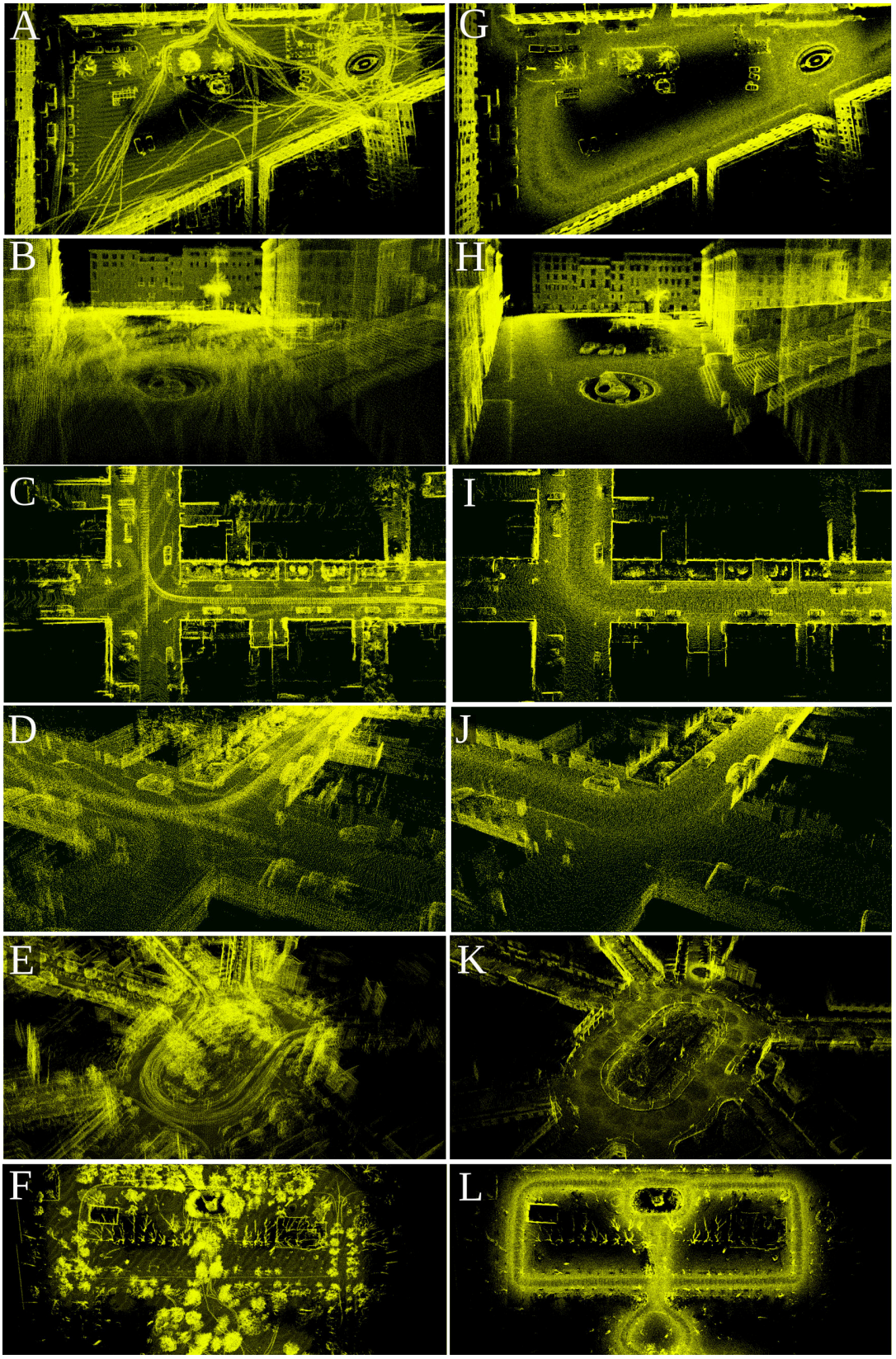

FIGURE 5.13: Comparison of the maps generated by LOAM (A–F) and MAD-BA method (G–L), with the surfel map converted to a point cloud. The visualizations show part of the *Spagna* (A, B, G, H), KITTI 07 (C, D, I, J), *Ciampino* (E, K), and *Pincio* (F, L) sequences. Proposed approach offers a clearer representation of the environment by removing noise and filtering out points from dynamic objects such as moving cars and pedestrians.

5.2.4 Comparison of high-level planar features with surfels

PlaneLOAM and MAD-BA systems represent two different strategies for structuring point cloud data, each with distinct advantages and trade-offs. Both methods aim to reduce raw point cloud complexity by grouping points into features, enabling more efficient data association and optimization. They share common elements, such as using point-to-plane or point-to-line error functions for data association and leveraging kd-trees for correspondence search. However, their approaches to feature representation and optimization differ considerably.

One of the primary differences between these two representations is the size of the features. The planar features in PlaneLOAM function similarly to surfels but are not limited in size, meaning that they can represent large flat areas, such as the walls of buildings or floors, more effectively. In contrast, surfels are smaller, which allows for capturing finer details of the environment, making them particularly effective in areas with high surface variation. Additionally, in the surfel-based approach, once the points are converted into surfels, the original point cloud data is no longer used in subsequent processing. This also slightly simplifies the correspondence search and feature matching, as it requires fewer parameters. The smaller size also makes them more flexible in optimization, as they can be modified more easily compared to large planar features. This makes surfel-based approaches particularly effective in high-accuracy reconstruction. However, the small size also means that surfels struggle to efficiently represent large flat surfaces, where big planar features might be more suitable.

Another important distinction lies in the computational efficiency of the approaches, however, this is more related to the purpose of the system rather than the map representation. PlaneLOAM was developed for real-time operation, enabling continuous localization and mapping without excessive computational overhead. In contrast, MAD-BA require matching all scans with each other, making it significantly more computationally expensive. For this reason it is intended for offline processing, where accuracy is prioritized over speed.

The comparison of the accuracy of both systems is presented in Table 5.8. It was conducted using three selected KITTI sequences, previously analyzed in the thesis, but now summarized in a single table for direct comparison. To ensure a fair evaluation, neither system incorporated loop closure information, and based on the accuracy metrics, MAD-BA demonstrated superior performance in all analyzed sequences. Moreover, based on the trajectory and error over time plots, illustrated in Figure 5.14, it can be noticed that MAD-BA exhibits lower error at both the beginning and end of the trajectories. This might be caused by the fact that matching surfels is more reliable than aligning large planar or linear features, particularly in the case of significant trajectory drift. Additionally, the point-to-plane threshold in the data association step differs between the two approaches, and it is set to 5.0 m for surfels and 1.0 m for planar features. Since scans from the start and end of the trajectory are more likely to be matched in the MAD-BA system, this results in reduced error, resembling the effect of loop closure without explicitly implementing it.

Figure 5.15 shows the comparison of a map created using the PlaneLOAM and MAD-BA systems. To enhance clarity, each planar feature in the PlaneLOAM map is assigned a distinct color, while surfels in the MAD-BA map are colored according to their normals.

TABLE 5.8: ATE comparison for KITTI sequences between PlaneLOAM, employing high-level features, and MAD-BA, utilizing surfels.

dataset	PlaneLOAM			MAD-BA		
sequence	ATE_{RMS}	ATE_{max}	σ_{ATE}	ATE_{RMS}	ATE_{max}	σ_{ATE}
KITTI 00	4.52 m	11.76 m	2.36 m	3.54 m	6.47 m	1.35 m
KITTI 05	3.21 m	8.04 m	1.59 m	1.43 m	2.68 m	0.64 m
KITTI 07	0.50 m	0.82 m	0.20 m	0.40 m	0.70 m	0.16 m

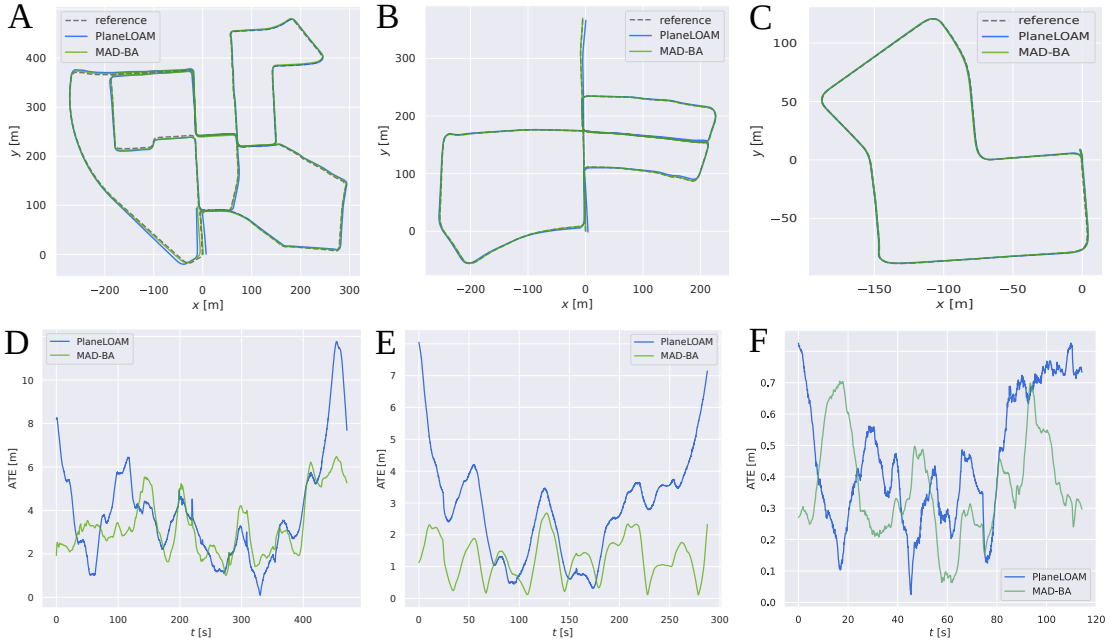

FIGURE 5.14: Trajectory plots for the analyzed KITTI sequences (A, B, C) and the corresponding error over time (D, E, F) for PlaneLOAM (blue line) and MAD-BA (green line).

The most significant difference lies in the way each system represents surfaces. Surfels divide walls into multiple small elements, which can be beneficial when dealing with uneven or textured surfaces. PlaneLOAM, on the other hand, represents an entire wall as a single planar feature, making it efficient for large, flat structures but potentially less adaptive to surface variations.

Another distinction between the two approaches is how features are managed. PlaneLOAM requires additional processing steps, such as merging and removing invalid features, to maintain map consistency. In contrast, surfels are neither merged nor deleted, making their management simpler while still allowing for dynamic refinement. On the other hand, the surfel-based map is constructed iteratively. After each optimization step, the system updates the poses and recreates the surfels. This iterative refinement process ensures continuous improvement in the map accuracy over time but is more demanding in terms of computations.

In general, both approaches provide viable alternatives to traditional point-based maps, each offering distinct advantages depending on the structural characteristics of the environment and the required level of detail.

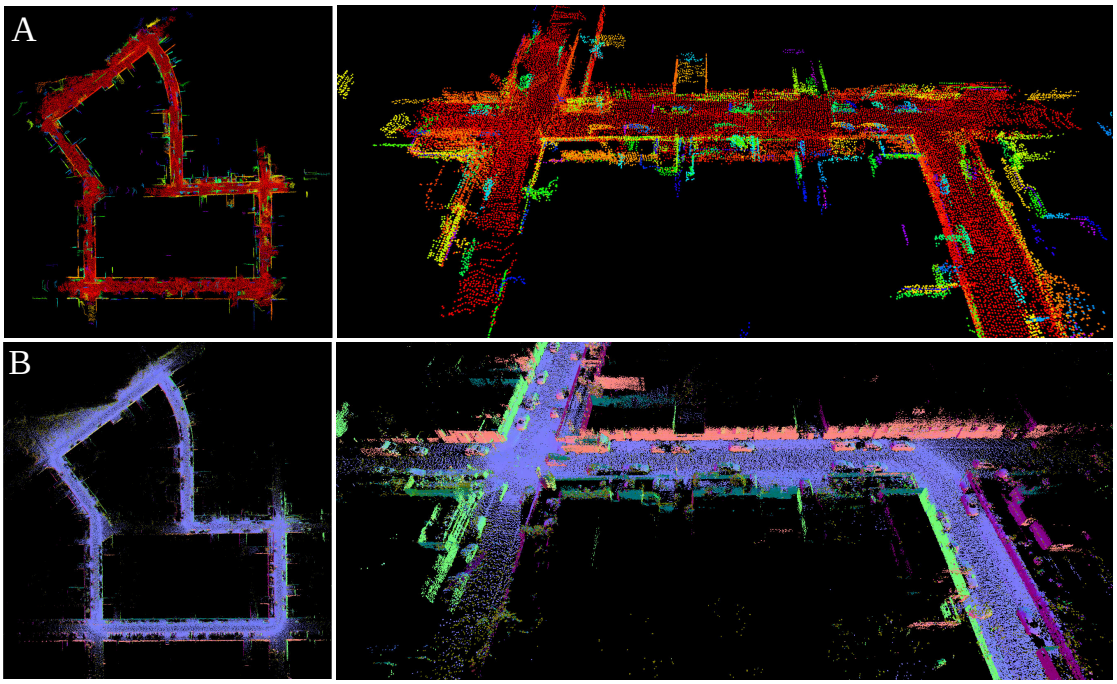

FIGURE 5.15: Comparison of the maps created using high-level planar features in PlaneLOAM (A) and surfel-based representation in MAD-BA (B). Each planar feature is assigned a unique color for better visibility, while surfels are coloured based on their normal vector.

5.3 Evaluation of LiDAR SLAM with GNSS integration

5.3.1 Introduction

This section presents the evaluation of the method for integration of SLAM with GNSS data, introduced in Section 4.1, and focuses on validating the performance and effectiveness of the developed approach. It also demonstrates how these systems can be combined to provide robust localization in urban environments and produce a globally consistent trajectory.

Urban canyons, characterized by tall buildings and complex infrastructures, often degrade GNSS signals, leading to inaccuracies and loss of signal. SLAM systems, on the other hand, offer reliable local localization by mapping the environment and tracking the motion of a vehicle. However, without a global reference, SLAM can accumulate drift over time. In this context, the synergy between these two systems leverages the strengths of both of them: GNSS provides global position references and SLAM ensures local accuracy and continuity.

Dataset

The experiments were carried out using the UrbanNav dataset [136], which is specifically designed to test navigation systems and provides data from complex urban environments that present significant challenges for GNSS-based positioning. It is publicly available and includes sequences recorded in densely populated areas. The presented experiments focus on two sequences from

Tokyo (Odaiba and Shinjuku) and three sequences from different parts of Hong Kong: Tsim Sha Tsui (TST), Whampoa, and Mongkok.

The dataset includes raw GNSS measurements in Receiver Independent Exchange System (RINEX) format, widely used for this type of application. RINEX data consists of two file types: observation and navigation. Observation files are recorded by a GNSS receiver mounted on a moving vehicle and contain the following values:

- time of measurement,
- pseudorange,
- carrier phase,
- Doppler shift.

Navigation files provide information about satellite trajectories to compute their positions at specific times and are not necessarily recorded by the receiver whose position is being calculated. However, they must include data about satellites present in the observation file to ensure completeness for processing.

The sequences recorded in Tokyo feature data from two GNSS receivers: Trimble NetR9 and u-blox ZED-F9P. For uniformity, only the data from u-blox ZED-F9P is processed, as it is also present in the Hong Kong sequences. In addition, Hong Kong sequences include raw GNSS measurements from four different smartphone models. For the presented evaluation, data from Xiaomi Mi8 was selected, as it supports multiple constellations and dual-frequency bands (L1/L5) for Global Positioning System (GPS), providing relatively high precision.

In terms of SLAM systems, Odaiba and Shinjuku sequences were processed using LOAM [137] with data provided by the Velodyne VLP-32C sensor. These sequences were not processed with LIO-SAM [69] due to the absence of a 9-axis IMU required by LIO-SAM. In contrast, trajectories from Hong Kong were generated using both LOAM and LIO-SAM using data from Velodyne HDL-32E. For clarity, Table 5.9 presents which sequences of the UrbanNav dataset and which GNSS receivers were used for the evaluation.

TABLE 5.9: The sequences of UrbanNav dataset that were processed with the use of different GNSS receivers for the purpose of evaluation. The ✓ sign means that given sequence was processed and ✗ means that it was not

dataset	LOAM		LIO-SAM	
sequence	u-blox ZED-F9P	Xiaomi Mi8	u-blox ZED-F9P	Xiaomi Mi8
Odaiba	✓	✗	✗	✗
Shinjuku	✓	✗	✗	✗
TST	✓	✓	✓	✓
Whampoa	✓	✓	✓	✓
Mongkok	✓	✓	✓	✓

By analyzing the above sequences, recorded in Tokyo and Hong Kong, the performance improvements achieved through the developed system were evaluated, comparing it with the standalone GNSS and SLAM approaches.

Evaluation method

The results in the following section are calculated by comparing the trajectories provided by the evaluated system with the ground truth, as already introduced in Section 5.1.1. The evaluation focuses on ATE metrics, because it highlights the ability of a system to maintain precision and consistency across diverse urban scenarios. The errors are reported as ATE_{RMS} and ATE_{max}, representing the RMS and maximum ATE values, respectively. Ground-truth data, provided for all sequences by an RTK GNSS/Inertial Navigation System (INS) receiver, is recorded in the WGS-84 standard (latitude, longitude, altitude) and was directly converted to the ECEF frame for evaluation purposes.

It is important to note that LIO-SAM includes functionality for loop closure detection. In the presented comparison of GALS with LOAM and LIO-SAM, the emphasis was placed on demonstrating the impact of IMU integration in LIO-SAM. Therefore, to ensure a fair comparison, the loop closure functionality was disabled. However, in a separate experiment, this feature was enabled to demonstrate that the GALS system is capable of managing loop closure constraints.

The comparison also includes results from the RTKLIB library, which, as described in Section 4.1.3, was used to process raw GNSS measurements and provides positioning information using GNSS-based data.

5.3.2 Accuracy of trajectory estimation

GNSS data source: u-blox ZED-F9P

Table 5.10 and Figure 5.16 provide a detailed comparison of the trajectory estimation outcomes across three localization methods: LOAM, RTKLIB, and GALS, tested on five different UrbanNav sequences. The raw GNSS data utilized in this analysis are obtained from a u-blox ZED-F9P receiver. The results demonstrate a notable reduction in trajectory estimation errors when using the GALS system compared to both LOAM and RTKLIB. This improvement is evident across all sequences, however, for the Whampoa, the reduction in error is less significant. This can be attributed to the lack of valid GNSS measurements over a substantial part of the trajectory, which limits the accuracy improvement typically provided by the GALS system.

TABLE 5.10: Comparison of ATE metrics calculated for trajectories obtained using three different localization approaches: LOAM, RTKLIB, and GALS across UrbanNav sequences. The RTKLIB utilize data from a u-blox ZED-F9P receiver, while the GALS approach integrates data from both LOAM and the u-blox ZED-F9P. The lowest ATE_{max} values are highlighted in bold.

dataset	LOAM		RTKLIB (u-blox ZED-F9P)		GALS (proposed)	
sequence	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}
Odaiba	64.24 m	135.85 m	8.02 m	41.24 m	**5.88** m	11.90 m
Shinjuku	53.46 m	104.92 m	11.67 m	74.30 m	**6.84** m	22.70 m
TST	12.51 m	31.159 m	14.76 m	57.47 m	**3.83** m	6.86 m
Whampoa	35.99 m	76.848 m	9.55 m	40.33 m	**9.26** m	26.66 m
Mongkok	7.39 m	25.366 m	27.01 m	90.48 m	**5.64** m	18.44 m

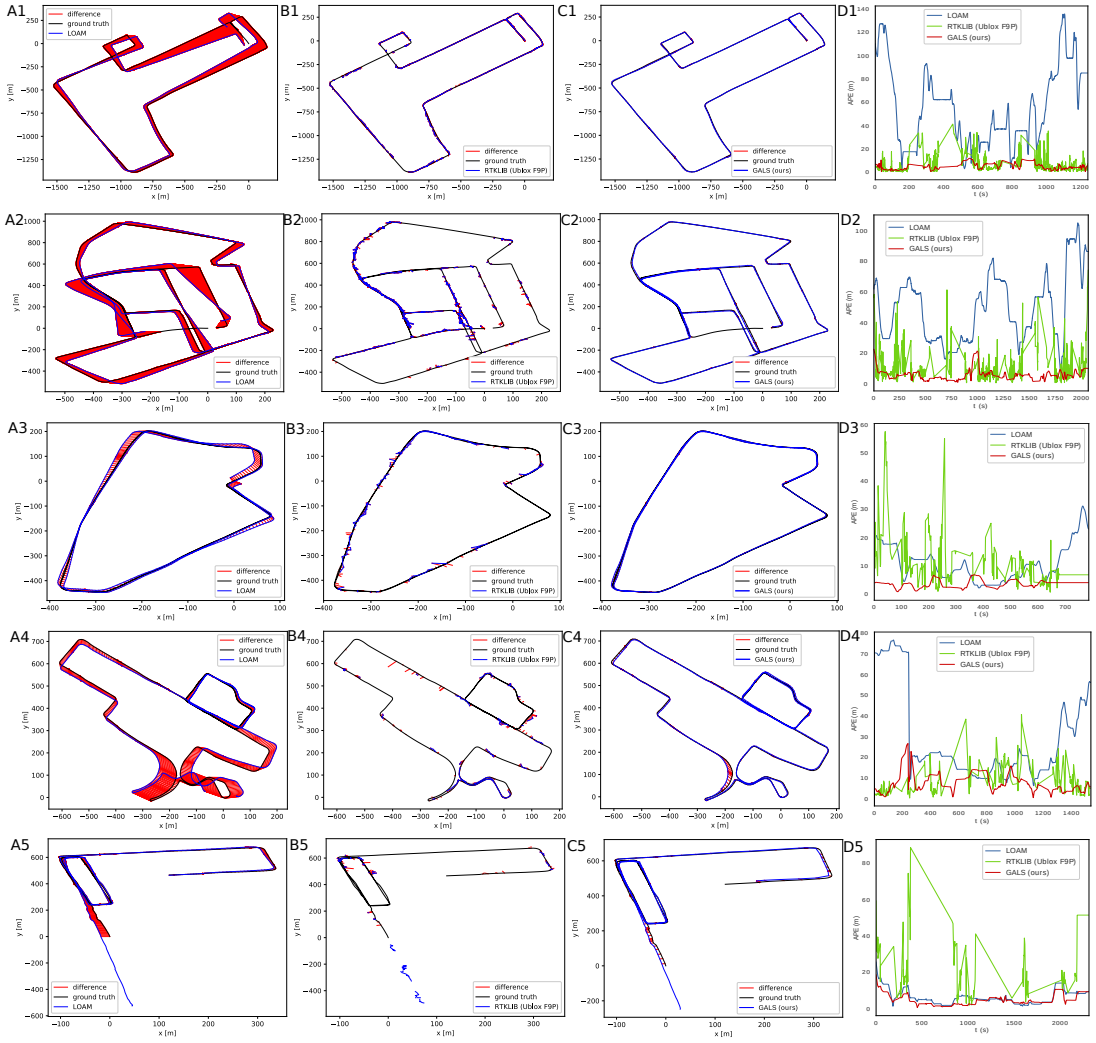

FIGURE 5.16: Comparison of ATE metrics calculated for the five UrbanNav sequences: Odaiba (1), Shinjuku (2), TST (3), Whampoa (4), and Mongkok (5). The plots illustrate the trajectory errors for each sequence using three different localization methods: LOAM (A), RTKLIB with data from the u-blox ZED-F9P receiver (B), and GALS, which combines data from LOAM and the u-blox ZED-F9P (C). Additionally, plot (D) provides a detailed view of ATE over time for each method. These plots complement the results presented in Table 5.10

Table 5.11 presents analogous results, this time focusing on the performance of LIO-SAM integrated with GALS, in contrast to the LOAM system used previously. The RTKLIB localization results remain unchanged from Table 5.10 and are included here again to facilitate direct comparison. It should be noted that LIO-SAM consistently outperforms LOAM, primarily due to the integration of IMU with LiDAR data. In this way LIO-SAM enhances robustness in dynamic environments and during fast motion, where LOAM may encounter difficulties. This reduces potential drift and ensures a more stable pose estimation, which also leads to enhanced precision in the GALS trajectory. An additional comparison of the obtained routes, overlaid on OpenStreeMap maps, is presented in Figure 5.17.

As noted earlier, the loop detection features of LIO-SAM were disabled to produce these results. However, the impact of incorporating loop closures from LIO-SAM into GALS was also evaluated and presented in Table 5.12 and Figure 5.18. The (ATE_{RMS}) for an example trajectory

TABLE 5.11: Comparison of ATE metrics for trajectories derived from three approaches: LIO-SAM, RTKLIB using data from a u-blox ZED-F9P receiver, and GALS, which integrates data from LIO-SAM and the u-blox ZED-F9P. This comparison highlights the improvements achieved when LIO-SAM is employed in combination with GALS for trajectory estimation.

dataset	LIO-SAM		RTKLIB (u-blox ZED-F9P)		GALS (proposed)	
sequence	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}
TST	11.07 m	21.59 m	14.76 m	57.47 m	**3.07** m	7.41 m
Whampoa	10.03 m	26.27 m	9.55 m	40.33 m	**4.54** m	8.83 m
Mongkok	4.88 m	19.44 m	27.01 m	90.48 m	**4.38** m	10.87 m

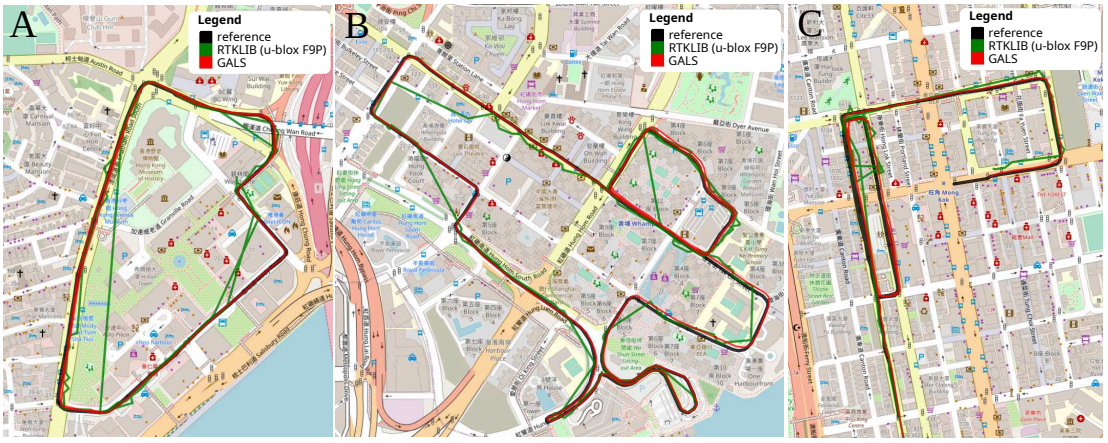

FIGURE 5.17: Trajectories for the TST (A) and Whampoa (B) and Mongkok (C) sequences from the UrbanNav dataset obtained using the proposed GALS approach with LIO-SAM (red line), RTKLIB (green line) and ground-truth reference (black line) overlaid on OpenStreetMap. The long green line segments in the RTKLIB trajectory indicate that multiple intermediate positions were filtered out.

estimated by LIO-SAM with loop closures is lower compared to the trajectory that relies solely on SLAM odometry (see Tab. 5.11). Although this enhancement is noticeable, it has a limited effect on the trajectory estimated by GALS, provided that filtered GNSS measurements are employed to maintain global consistency. The methodology for this filtering process is described in Section 4.1.4 and is necessary to achieve the best performance. As the results indicate, even with loop closure constraints, the factor graph cannot rectify errors introduced by inaccurate GNSS measurements, reflecting the influence of raw data and filtering mechanism on trajectory estimation, as demonstrated in Figure 5.18B.

TABLE 5.12: Absolute Trajectory Error calculated for the TST sequence, illustrating the impact of enabling or disabling specific components of our system: loop closures (LC) and the filtration of GNSS measurements. This comparison highlights how these features influence the accuracy of the resulting trajectories.

dataset	LIO-SAM w/ LC		GALS w/o filtering		GALS w/ filtering	
sequence	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}
TST	5.68 m	15.15 m	8.14 m	18.08 m	**2.17** m	6.65 m

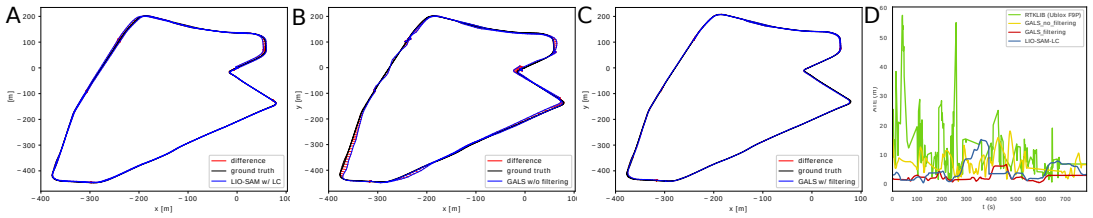

FIGURE 5.18: Comparison of the ATE for the TST sequence, calculated under different system configurations. Plot (A) shows the results using LIO-SAM with loop closure detection enabled, plot (B) presents ATE for GALS optimization without applying any filtration to the GNSS measurements, plot (C) illustrates the results with GALS optimization that includes GNSS filtration, and plot (D) shows how the error evolves over time for each configuration. These plots complement the results in Table 5.12.

GNSS data source: Xiaomi Mi8

The subsequent experiments are similar to those described previously, however, in this case, the raw GNSS data is sourced from a built-in GNSS receiver in the Xiaomi Mi8 smartphone instead of the u-blox ZED-F9P. Additionally, only sequences from Hong Kong are analyzed, as the corresponding data for the routes recorded in Tokyo are not available. Tables 5.13 and 5.14 present a quantitative evaluation of the trajectory error for these experiments that involve the use of LOAM and LIO-SAM, respectively. The related plots are shown in Figure 5.19. Despite the significantly larger GNSS positioning errors (see column B of Fig. 5.19), the GALS system still produces substantially more accurate trajectories compared to those generated by LOAM and LIO-SAM, which rely solely on LiDAR or a combination of LiDAR and IMU data. This enhanced performance is attributed to the GNSS filtration process, which effectively filters out invalid positions and helps mitigate the drift of SLAM. This is particularly beneficial in long-distance sequences, where even low-quality GNSS measurements can support more accurate trajectory estimation by reducing the accumulated errors over time.

TABLE 5.13: Comparison of ATE metrics for trajectories obtained with LOAM, RTKLIB using data from Xiaomi Mi8, and GALS combining data from LOAM and Xiaomi Mi8

dataset	LOAM		RTKLIB (Xiaomi Mi8)		GALS (proposed)	
sequence	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}
TST	12.51 m	31.159 m	84.38 m	683.60 m	**4.81** m	9.13 m
Whampoa	35.99 m	76.848 m	48.18 m	263.07 m	**16.57** m	43.35 m
Mongkok	7.39 m	25.366 m	93.84 m	665.07 m	**5.94** m	17.86 m

TABLE 5.14: Comparison of ATE metrics for trajectories derived from LIO-SAM, RTKLIB using data from a Xiaomi Mi8, and GALS, which integrates data from both LIO-SAM and the Xiaomi Mi8.

dataset	LIO-SAM		RTKLIB (Xiaomi Mi8)		GALS (proposed)	
sequence	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}
TST	11.07 m	21.59 m	84.38 m	683.60 m	**3.69** m	6.27 m
Whampoa	10.03 m	26.27 m	48.18 m	263.07 m	**7.19** m	16.44 m
Mongkok	4.88 m	19.44 m	93.84 m	665.07 m	**4.41** m	9.14 m

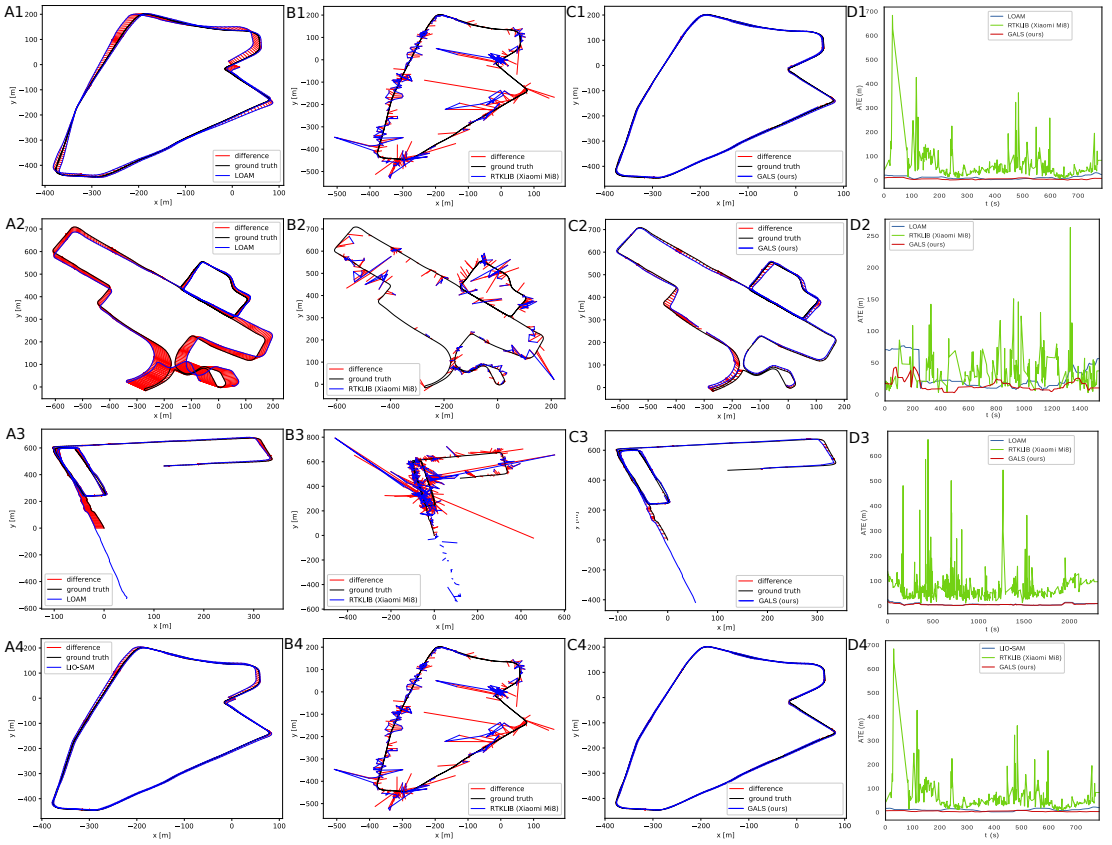

FIGURE 5.19: Comparison of ATE calculated for the TST (1), Whampoa (2), Mongkok (3), and TST (4) sequences. The plots display trajectory errors for LOAM (A1–A3) or LIO-SAM (A4), RTKLIB with Xiaomi Mi8 data (B), and GALS (C). Plot (D) illustrates how ATE varies over time, providing a temporal analysis of the localization accuracy for each method.

Accuracy comparison between RTKLIB and the GNSS receiver

The final results of this section focus on comparing the localization accuracy achieved directly from the GNSS receivers versus using RTKLIB with raw measurements. The errors, calculated based on the provided WGS-84 coordinates from both u-blox ZED-F9P and Xiaomi Mi8 receivers, are presented in Table 5.15.

TABLE 5.15: ATE for positions returned directly by u-blox ZED-F9P and Xiaomi Mi 8 receivers.

dataset	u-blox ZED-F9P		Xiaomi Mi 8	
sequence	ATE_{RMS}	ATE_{max}	ATE_{RMS}	ATE_{max}
TST	3.84 m	9.67 m	11.73 m	28.94 m
Whampoa	5.70 m	17.85 m	13.09 m	48.25 m
Mongkok	9.79 m	33.52 m	18.81 m	51.91 m

Based on these results, it can be seen that the localization results obtained using RTKLIB and the raw GNSS data are inferior to those provided directly by the GNSS receiver. This might be due to a number of significant factors. First, raw GNSS data includes various sources of error, such as multipath effects, atmospheric delays, and clock inaccuracies, which RTKLIB must address. GNSS receivers are equipped with tuned internal algorithms designed to handle these errors

effectively in real time, whereas RTKLIB may provide worse results without the proprietary enhancements that receivers use. Furthermore, RTKLIB processes raw GNSS data in a post-processing mode, which requires the completeness of collected RINEX files. If the necessary data, such as the trajectory of a given satellite, is inaccurate, outdated, or unavailable, the localization accuracy can degrade. In contrast, GNSS receivers provide real-time processed solutions that are optimized for immediate use, thus often yielding more reliable results. In addition, the hardware integration of the GNSS receivers further improves their performance, since the system is optimized for both hardware and software. Nonetheless, the proposed GALS system offers a versatile framework for GNSS-augmented LiDAR SLAM by tightly integrating raw GNSS measurements with existing LiDAR SLAM algorithms. The development of a factor graph formulation that enables the use of raw GNSS data from low-cost receivers, without requiring external corrections, substantially enhances both the local accuracy and global consistency of the estimated trajectories.

Chapter 6

Practical applications of GNSS

6.1 Localization system for an electric city bus

6.1.1 Introduction

This section presents the results of the development of a localization system for electric city buses, which is part of a project aimed at creating an ADAS. The primary goal of this system is to assist drivers during docking maneuvers at charging stations located at bus stops or depots. However, while providing an overview of the entire system, this section primarily focuses on the localization component, detailing the approaches and tests conducted to evaluate its performance. In addition, it also reports the results of tests of a GNSS-based localization system for buses operating in a real urban environment with regular daily traffic and passengers. For that reason, it provides a valuable perspective on the practical challenges of deploying such a system in real-world applications.

The developed ADAS offers a solution to docking challenges by providing visual guidance to the driver that allows them to navigate the vehicle along an optimal path. Although the bus equipped with this system is not fully autonomous, the ADAS significantly reduces driver stress and compensates for lack of experience. Previous studies [112, 169] introduced the overall concept of the ADAS for both single-segment and articulated buses. These works covered the development of motion planning and control subsystems, as well as a method for using a single camera to localize the bus relative to the charging station. The prototype system was developed through a collaborative research project between Poznan University of Technology and Solaris Bus & Coach, a leading European electric bus manufacturer. For the experiments presented here, the bus was equipped with a simplified version of the ADAS, which incorporated significant modifications from the earlier prototype. These changes were driven by an economic analysis that assessed the profitability of adding additional hardware to the vehicle. As a result, external sensors such as cameras and LiDARs were removed, and the system now relies entirely on data from the GNSS and the internal sensors of the bus. In contrast to the version that employs a vision system [169], the current implementation can work effectively at night and under adverse weather conditions,

such as snow or heavy rain. These modifications have resulted in a cost-effective system that can be retrofitted to nearly any existing electric bus, providing reliable guidance to inexperienced drivers toward the charging station.

Although standard GNSS receivers generally provide adequate accuracy for most applications, scenarios such as the one presented here demand higher precision. This can be achieved by methods such as RTK GNSS [119]. The level of accuracy achieved using this technique varies depending on the type and manufacturer of the receiver. For example, the u-blox F9P module [165] claims a horizontal accuracy of 1.5 m in standard mode and 0.01 m in RTK mode under optimal conditions. Given these capabilities, this localization method was chosen, as it provides the necessary precision and reliability to determine the position of a bus at a reasonable cost. Furthermore, the RTK GNSS ensures highly consistent pose estimates [170], which is particularly beneficial for tasks related to precise maneuvers.

6.1.2 Architecture of the system

Hardware

The hardware used in this system comprises several key components: a single-board computer, a CAN-USB adapter, two RTK GNSS receivers and a Long Term Evolution (LTE) module. A schematic diagram of the system is presented in Figure 6.1.

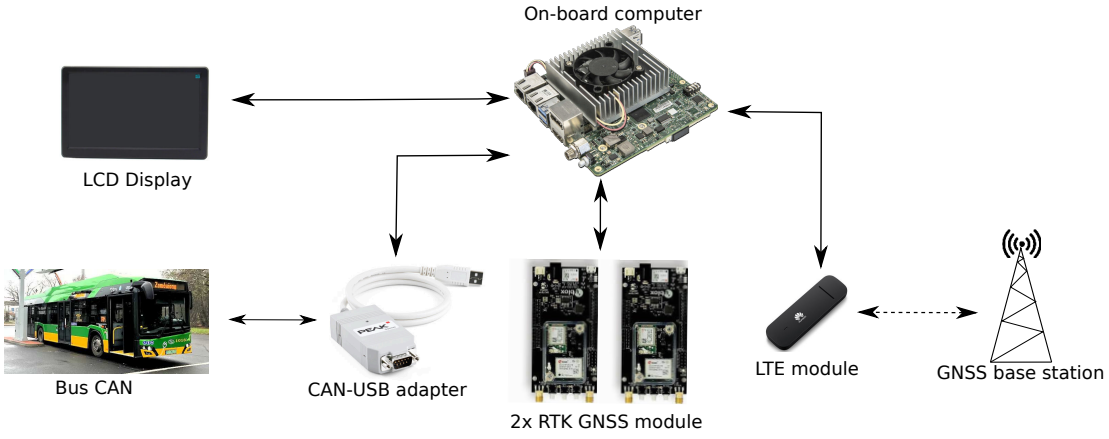

FIGURE 6.1: Hardware utilized in the ADAS system. The on-board computer handles computations, the GNSS modules provide bus position and heading, the CAN-USB adapter reads sensor data from the bus, the LTE module receives Radio Technical Commission for Maritime (RTCM) corrections from the base station via the network, and the LCD displays the system interface to the driver.

The selected on-board computer, the UP Xtreme, is responsible for running the software and managing data from connected peripherals. Its primary functions include processing Controller Area Network (CAN) messages, managing data from the GNSS receivers, and maintaining communication with the GNSS base station through the LTE module. During docking operations, it also performs motion planning and control. Details about the software used on this computer are provided in Section 6.1.2.

The UP Xtreme was chosen due to its Intel i5 CPU with x86 architecture, which offers greater versatility for software installation compared to ARM-based boards like the Raspberry Pi or Odroid. ARM boards often require specific kernels or Linux versions, while the UP Xtreme supports the official Ubuntu as its operating system. Additionally, its broad input voltage range (12–60 V DC) allows it to be powered directly from the bus, eliminating the need for extra power supplies or DC-DC converters. Alternatives like the UP Board and UP Squared were considered but discarded due to their lower performance, which resulted in longer initialization times for the motion planner and Graphical User Interface (GUI) before maneuvers.

The CAN-USB adapter facilitates a connection to the CAN bus of a vehicle, enabling the retrieval of its current state. Data obtained via the CAN bus include vehicle steering angle, speed, and pantograph status (folded or raised). The steering angle and speed are used for bus odometry calculations, which the motion planner relies on in the event of a GNSS signal loss. Meanwhile, the pantograph state serves as a signal that the driver intends to dock at the charging station. CAN communication was selected because all the required messages were already accessible. The PEAK-System CAN-USB adapter was chosen for its galvanic isolation, which ensured that any failure in our system would not interfere with the regular operation of the bus.

In terms of RTK GNSS receivers, two u-blox ZED-F9P modules are employed. The first, called the moving base, determines the position of the bus, while the second, called the rover, is used to calculate the yaw angle (heading). Information about the position and heading of a bus is crucial for this application, as it is used to plan the reference path that guides the driver throughout the maneuver. In addition, it also helps to identify the moment when the system should be activated. Since GNSS modules require external antennas for optimal performance, they were installed on the roof of the bus, as shown in Figure 6.2. This placement ensures an unobstructed view of the sky, allowing uninterrupted reception of satellite signals. The first antenna is placed 0.2 m in front of the rear axle and links to the moving base receiver for the acquisition of position data. Meanwhile, the second antenna is located 4.9 m in front of the first and is attached to the rover module. The alignment of GNSS antenna positions with the bus coordinate system and the synchronization of GNSS data with the internal sensors was achieved through offline calibration conducted prior to testing. The calibration process, which relies on the optimization techniques described in [171], ensures high accuracy and repeatability. Once calibrated, the synchronization between the sensors is managed using software timestamps.

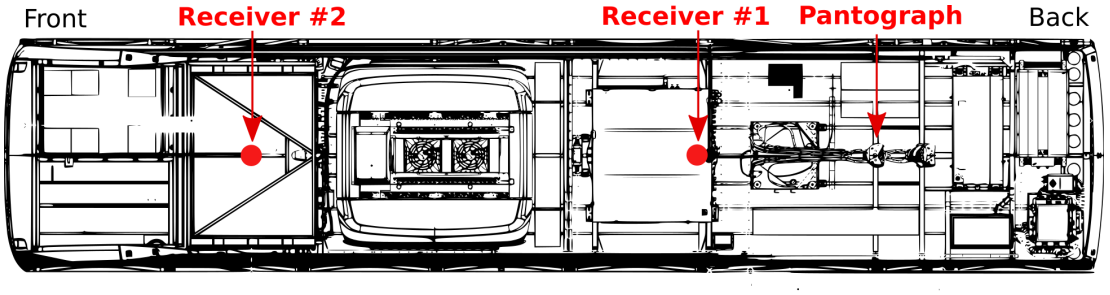

FIGURE 6.2: Placement of the GNSS antennas concerning the geometry of a bus roof. The first antenna is positioned 0.2 m in front of the rear axle to determine the vehicle's position. The second antenna is installed 4.9 m ahead of the first and provides the information about heading of a bus.

The u-blox ZED-F9P receivers were selected for several reasons. Most importantly, these modules support the moving base and rover configuration in RTK mode, which can simultaneously provide the position and heading of the vehicle. Additionally, their integration is straightforward with existing Linux-compatible software. The modules are also cost-effective and widely available, which was particularly advantageous during the pandemic when the availability of electronic components was limited. Although other GNSS receivers at a similar price point, such as the Swift Navigation Piksi Multi and SKYTRAQ PX1122R, are available, they lack built-in heading estimation. Using such alternatives would require manual calculations of the heading angle based on two antenna positions, resulting in reduced accuracy.

The next hardware component is an Liquid Crystal Display (LCD) that acts as a graphical interface for the driver. It shows key information, including the current position of the bus, the planned path, the desired steering angle, and the path error. This display is connected to the ADAS computer and automatically turns on when the bus approaches the charging position, showing the Human-Machine Interface (HMI) that provides the driver with a clear understanding of the maneuver.

The final hardware element is an LTE module, which facilitates remote access and provides internet connectivity to the on-board computer. A generic LTE module powered directly through the USB port of the UP Extreme computer was chosen, as this setup allows it to automatically start synchronously with the other components of the system. Additionally, the on-board computer uses this module to wirelessly connect to the stationary GNSS base station, enabling it to receive differential corrections.

Differential corrections aim to enhance positioning accuracy by minimizing errors associated with GNSS signal propagation and processing. Specifically, these corrections address delays caused by the Earth atmosphere, such as tropospheric and ionospheric delays. These delays impact the calculation of the pseudorange by affecting the signal travel time. In addition, GNSS corrections include information about satellite clock biases, which influence measurements since the signal travel time is determined using the difference between the receiver and satellite clocks. A GNSS base station continuously calculates these corrections based on signals from satellites and the known fixed position of its antenna [172]. These corrections, known as pseudorange corrections, can then be shared with other receivers to improve their calculations and overall positioning accuracy. This is possible because atmospheric effects on GNSS signals are similar for all receivers within a short distance from each other, typically several dozen kilometers. Corrections also provide precise satellite orbit parameters, which are essential for determining satellite positions at any given time. In addition, they refine carrier phase measurements, which are based on the phase shift of the signal carrier wave. These measurements are more computationally intensive than pseudorange calculations, and hence they require additional messages that are transmitted along with the differential corrections. In the presented implementation, these messages are transmitted in the Radio Technical Commission for Maritime (RTCM) standard to the on-board computer, which forwards them to the moving base receiver, ensuring the RTK localization mode [172]. The source of RTCM messages is the ASG-EUPOS network, which provides data from multiple reference stations in Poland through an internet-based service.

Unfortunately, certain errors cannot be mitigated by using RTCM data, such as those arising from signal multipathing and obstructions. These issues depend on the actual position of a receiver and its nearby environment. The primary causes of such errors are tall buildings and trees, which can reflect or block satellite signals. As a result, GNSS localization tends to be less accurate in highly urbanized areas and may fail entirely in tunnels or underground parkings.

Another critical consideration is LTE network coverage in the areas where the RTK GNSS system operates. Although this is typically not an issue in urban regions, a lack of internet connectivity prevents the reception of RTCM corrections, thereby reducing positioning accuracy. Short interruptions in internet access, which can occur even in cities, are generally not problematic, since RTCM corrections remain valid for several seconds. However, prolonged connectivity issues cause the receiver to exit RTK mode and revert to standard FIX mode, which offers a maximum static accuracy of 1.5 m. Although it is satisfactory for determining the rough position of a bus, this accuracy is insufficient for precise docking maneuvers. Thus, RTCM corrections are essential for the proper functioning of the system.

The GNSS receiver operates in three main modes: standard FIX, RTK FLOAT, and RTK FIXED. In RTK FLOAT mode, the receiver has acquired RTCM corrections but has not yet achieved the highest precision level. Accuracy in this mode is typically within tens of centimeters, which is better than in the case of standard FIX but still depends on the number of visible satellites and signal quality. RTK FIXED mode is the most precise one, and it is achieved when the receiver resolves carrier phase cycle ambiguities and determines the exact number of radio wavelengths between satellites and the receiver. This enables centimeter-level accuracy, which can be maintained as long as RTCM data are consistently transmitted and the antennas have a clear sky view. All three modes (standard FIX, RTK FLOAT, and RTK FIXED) apply to both the moving base and the rover modules. Regardless of the mode, the receivers utilize signals from multiple satellite constellations and frequency bands, including GPS (L1/L2), Galileo (E1/E5), and GLONASS (L1/L2). This multi-constellation, multi-frequency approach ensures that the system always has sufficient satellite visibility. Moreover, it also reduces the time required to initialize after power-up. Notably, the u-blox ZED-F9P receivers used in the system start calculating its position as soon as the bus ignition is turned on, and usually achieves standard FIX mode even before the operating system fully initializes. However, achieving RTK FIXED mode requires the LTE module to establish a network connection and begin RTCM data transmission, which typically takes 40–60 seconds.

Software

Concerning the software, the system uses Ubuntu and Robot Operating System (ROS), a popular set of open-source libraries and tools for robotic applications. ROS offers significant flexibility, as each system component has its own package that can be independently started or stopped as needed. The system architecture is illustrated in Figure. 6.3, with all software components implemented as ROS nodes, facilitating seamless communication between packages.

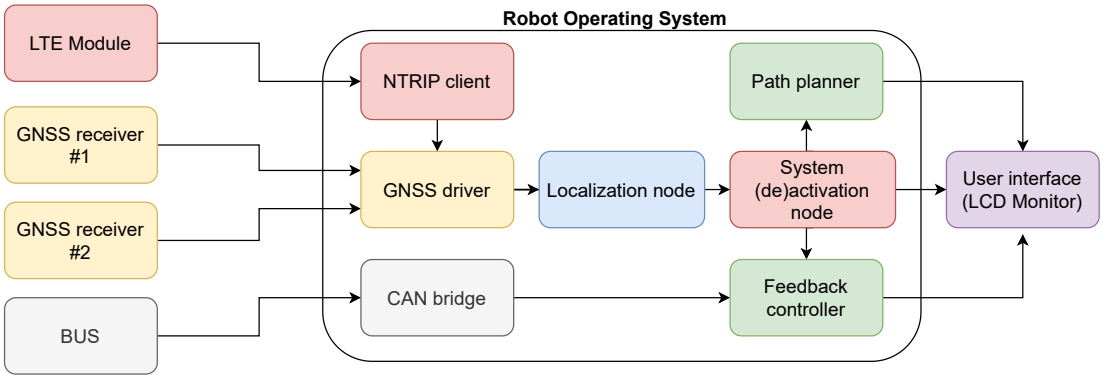

FIGURE 6.3: Overview of the system architecture. The software is implemented as ROS nodes, including the NTRIP client, GNSS driver, CAN bridge, localization node, system activation node, path planner, and feedback controller. The NTRIP client retrieves RTCM corrections via the LTE module. The GNSS driver processes data from u-blox ZED-F9P receivers. The CAN bridge interfaces with bus-mounted sensors. The localization node calculates the bus pose relative to the charger. The system activation node initiates motion planning as the bus arrives at the depot. The path planner and feedback controller generate the reference path and guide the driver during docking maneuvers.

The first software component is an Networked Transport of RTCM via Internet Protocol (NTRIP) client[1], which retrieves RTCM corrections using the LTE module connected to the on-board computer. As described in Section 6.1.2, these corrections are sourced from the ASG-EUPOS internet-based service, which aggregates data from multiple GNSS base stations across Poland. The NTRIP client's role is to establish a connection with this service, receive the correction data, and pass them on to the GNSS driver. Simultaneously, the GNSS driver transmits the RTCM messages to the moving base receiver, enabling it to operate in RTK mode. Additionally, the driver retrieves position and heading data from the moving base and rover modules. This functionality is implemented using two modified ROS packages: `nmea_navsat_driver`[2] and `ublox`[3]. The GNSS data are updated at a frequency of 10 Hz, which is satisfactory for localization in urban traffic.

The CAN bridge is responsible for reading the current state of the internal sensors and converting these data into ROS messages, enabling straightforward integration with the pose estimation and motion planning packages. The implementation leverages the `ros_canopen`[4] interface and utilizes proprietary CAN messages provided by Solaris Bus & Coach S.A. The data collected via the CAN interface is used to compute the bus odometry and monitor the state of the pantograph.

The localization node determines the position and orientation of the bus relative to the charger by using data from the GNSS driver and the predefined global coordinates of the charger. Since the GNSS receivers provide location data in the form of (latitude, longitude, altitude), this information is converted into the Universal Transverse Mercator (UTM) system with Cartesian coordinates. For this conversion, the `gps_common`[5] package is used in conjunction with the Eigen library, which facilitates the required transformations between local reference frames and

[1] `https://github.com/LORD-MicroStrain/ntrip_client`
[2] `https://github.com/ros-drivers/nmea_navsat_driver`
[3] `https://github.com/KumarRobotics/ublox`
[4] `https://github.com/ros-industrial/ros_canopen`
[5] `https://github.com/swri-robotics/gps_umd/tree/master/gps_common`

computes the bus pose relative to the charging station. This calculated pose is subsequently utilized by the system activation node to initiate the motion planning system shortly before the docking maneuver.

The next software component is the path planner, which generates the reference path. It utilizes a general path planning framework based on the non-linear optimization of a 7th-degree polynomial, ensuring smooth and kinematically feasible paths. The planner formulates path planning as an optimization problem constrained by initial and final configurations, kinematic limits (curvature and rate of curvature), and obstacle avoidance. Positional constraints are derived from bounding boxes generated using OpenStreetMap data, which define a safe space for vehicle movement. The optimization process, implemented using CasADi and the IPOPT library, ensures obstacle-free paths with minimal curvature changes. It balances computational efficiency with precision, generating reference paths that guide the bus during docking maneuvers.

The last module is the feedback controller that assists the driver in reaching the charging station. It ensures precise guidance by continuously adjusting the steering based on real-time deviations from the reference path. It calculates the error between the actual pose of the bus and the desired trajectory, considering position, orientation, and curvature. Using this error, the controller generates steering angle commands to minimize the deviation while maintaining smooth motion. By accounting for the kinematic constraints of the bus, it provides reliable and accurate feedback, enabling the driver to follow the reference path effectively, even in confined spaces or challenging docking scenarios.

A key requirement set by the bus fleet operator for our ADAS was that the system should operate autonomously without requiring any input or parameter adjustments from the driver. This specification was primarily driven by safety concerns. As a result, the system activates automatically when the driver starts the bus. Once the on-board computer boots up, the GNSS localization software starts to operate in the background. During this process, the monitor in driver's cabin is turned off to avoid potential distractions. When the bus approaches the designated location, the system triggers a package responsible for path planning and calculating the steering angle at a distance of 55 meters from the charger. This early activation ensures that the motion planner and the graphical interface have enough time to initialize. Concurrently, the driver's monitor turns on, and by the time the bus is 35 meters away, the system is fully operational. Since the bus position is calculated relative to the charger, all distance-based conditions can be easily tested and adjusted for various depots and charging stations as needed. Additional safeguards have been implemented to prevent unintentional system activation, such as when approaching the depot from the opposite direction. Once all components of the system have been activated successfully, the driver can follow the steering suggestions displayed on the monitor. The system will automatically deactivate when the bus halts beneath the charger or if it detects the driver performing an unrelated maneuver, such as driving across the depot without the intention to dock. In addition, the system will shut down if the bus moves more than 60 meters from the charger, regardless of driver actions. Although this condition is redundant, it ensures that the system is ready for the next approach.

It should be noted that the described system is directly powered by the bus, which can slightly increase its overall power demand. However, the flexibility of ROS packages allows them to

be deactivated once the maneuver is complete. As a result, the main software runs only when the bus is near the charger, minimizing power consumption for most of the time. The Intel i5-8365UE CPU used in our UP Xtreme setup has a maximum power consumption of 15 W, while the monitor consumes approximately 5.5 W. Unfortunately, other components, such as the LTE module and CAN-USB adapter, are powered via USB ports and cannot be automatically turned off. By reducing CPU activity and turning off the monitor, it is possible to save up to around 20 W of power. Although these savings may seem negligible compared to the total energy consumption of an electric bus, which typically ranges between 0.8 and 1.7 kWh per kilometer, it remains a good practice to minimize power usage whenever possible.

6.1.3 Performance evaluation of the GNSS-based localization system

The overall performance of the GNSS system was assessed on three distinct routes in Poznan during the regular operation of an electric bus in urban traffic. The average distances and travel times for the analyzed sequences are summarized in Table 6.1. This table also provides details on the number of RTK FIXED poses recorded for each sequence, reflecting the overall reliability of the system. As noted previously, RTK FIXED represents the high-precision localization mode with centimeter-level accuracy, which is critical for performing maneuvers such as docking at a charging station.

TABLE 6.1: The recorded city bus routes that were used to analyze the reliability and performance of GNSS localization. RTK FIXED positions represent a high-precision mode with centimeter-level accuracy. Lower precision modes, such as the standard FIX and RTK FLOAT modes, indicate the reduced accuracy of the system.

Sequence (bus stops)	Distance	Time	No. of precision RTK FIXED poses	No. of poses in low precision modes
Dworzec Zachodni–Kacza	7.483 km	34.77 min.	20558	284
Garbary PKM–Strzeszyn	11.616 km	42.42 min.	23749	1691
Garbary PKM–Os. Dębina	9.290 km	43.93 min.	25264	1090

The data indicates that only a small percentage of registered poses (1.36%, 6.65%, and 4.14% for each route, respectively) were non-RTK. These discrepancies are mainly due to the limited visibility of the sky caused by roadside structures and vegetation. However, it should be noted that the GNSS signals were consistently available throughout all sequences, with no interruptions or gaps in the logged trajectories.

To perform the statistical analysis of GNSS localization performance, the ATE metric was used, as introduced in Section 5.1.1. Since the results were collected during regular bus operations under typical day-to-day conditions, it was impossible to implement additional measurement systems to provide ground-truth data. For that reason, the ATE was calculated between the positions of the moving base and the rover receivers at the corresponding timestamps. The error was averaged over the route and is presented in Table 6.2. To compute the ATE, the rover coordinates were transformed to the moving base frame by translating them along the longitudinal axis of the bus (in the global coordinate system) by a length of 4.9 m, representing the exact distance between the two antennas mounted on the roof of the vehicle. It should be noted that this translation introduces an additional positional error for the rover, as it depends on the

accuracy of the heading estimation. Unfortunately, due to real-world testing conditions, it was not possible to measure the heading estimation error directly. Such an evaluation would require either higher-precision ground-truth data or an additional pair of GNSS receivers to compare their measurements. Despite this limitation, the results show that the smallest positional discrepancy between the moving base and rover was observed for the Dworzec Zachodni–Kacza sequence. This sequence also had a lower number of non-RTK poses, which corresponds with the improved localization performance.

TABLE 6.2: ATE computed between the poses from the moving base and rover receivers. To calculate the error, the rover poses were transformed to the moving base frame. The columns include the sequence name, and statistics of the error: RMS, max, mean and standard deviation, respectively.

Sequence (bus stops)	ATE_{RMS}	ATE_{max}	ATE_{mean}	ATE_{std}
Dworzec Zachodni–Kacza	0.276 m	3.113 m	0.175 m	0.213 m
Garbary PKM–Strzeszyn	0.602 m	4.377 m	0.428 m	0.423 m
Garbary PKM–Os. Dębina	0.619 m	4.340 m	0.440 m	0.436 m

An additional factor that could potentially degrade localization accuracy is mobile network coverage, as our system relies on continuous internet access to receive differential corrections for GNSS receivers. Without these corrections, the precision necessary for the system to function correctly would be insufficient. However, this was not an issue during our tests, as the LTE network coverage was consistently available. The recorded routes are shown in Figures 6.4 and visualize the sections of the trajectories where the accuracy of the GNSS localization decreased.

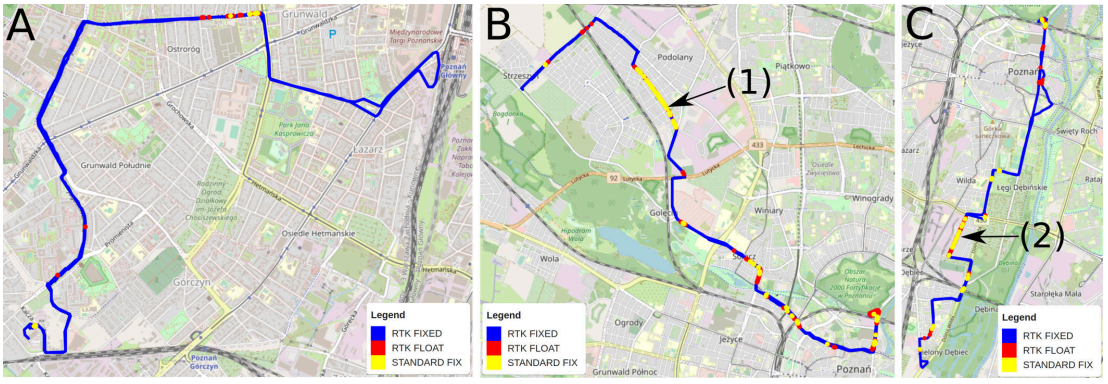

FIGURE 6.4: The recorded routes of the city bus shown on OpenStreetMap. RTK FIXED poses are depicted with blue lines, the less accurate RTK FLOAT with red lines, and the least accurate standard GNSS poses with yellow lines. Plot (A) corresponds to the Dworzec Zachodni–Kacza route, (B) to Garbary PKM–Strzeszyn, and (C) to Garbary PKM–Os. Dębina.

From these plots, it can be observed that there are certain segments of the trajectories, particularly in Figures 6.4B and 6.4C (indicated by numbers (1) and (2)), with only the standard FIX mode. These segments correspond to narrow streets with dense tree canopies and high-rise buildings (illustrated in Fig. 6.5), which obstruct GNSS signals and increase the localization error. Unfortunately, the surroundings have a considerable impact on the accuracy of the GNSS localization, and such situations are unavoidable [105, 108].

Figure 6.6 illustrates the error between the moving base and the rover, calculated similarly to the result in Table 6.2, but focused on a specific segment of the Garbary PKM–Os. Dębina

FIGURE 6.5: Examples of trajectory segments where the RTK FIXED mode of GNSS was unavailable due to limited sky visibility caused by buildings and trees. Image (1) corresponds to a section of the trajectory from Figure 6.4B, while image (2) illustrates a scene from Figure 6.4C.

trajectory. The statistical summary of this error is presented in Table 6.3. The results indicate that the accuracy of the GNSS-based localization decreases in these areas, as the error values for this segment are higher than those calculated for the entire sequence. In particular, the RMS and mean error values increased by approximately 0.16 m and 0.2 m, while the standard deviation of the error remained mostly unchanged. This suggests that the positions recorded in non-RTK mode may have a consistent offset, potentially caused by factors such as GNSS signal multipathing.

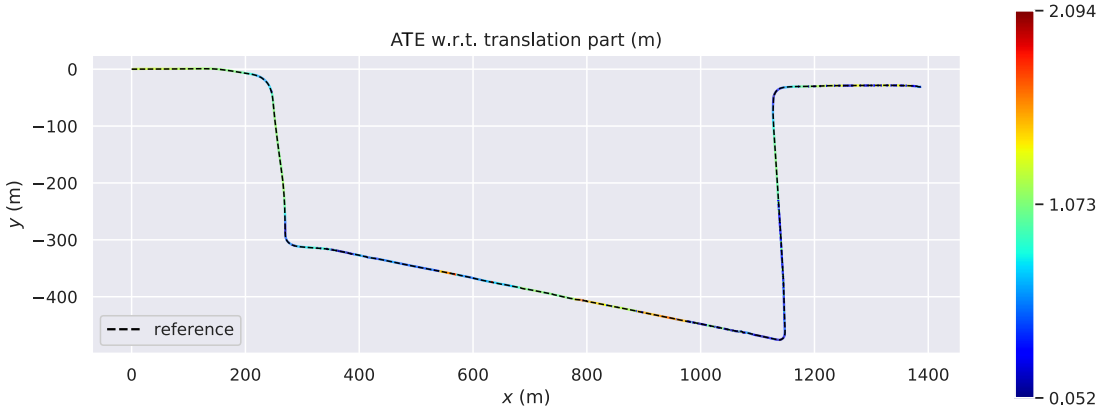

FIGURE 6.6: ATE calculated between the positions of the moving base (reference) and rover. The error corresponds to a section of the Garbary PKM–Os. Dębina route where the RTK FIXED mode was partially unavailable due to nearby high-rise buildings and trees.

TABLE 6.3: Errors between the *moving base* and *rover* positions for a segment of the Garbary PKM - Os. Dębina route where the RTK FIXED mode was partially unavailable. The columns represent the sequence name and the error statistics: RMS, max, mean, and standard deviation, respectively.

Sequence (bus stops)	ATE_{RMS}	ATE_{max}	ATE_{mean}	ATE_{std}
part of Garbary PKM–Os. Dębina	0.777 m	2.0943 m	0.640 m	0.439 m

The most critical part of the trajectory is the area near the charging station, where the system is designed to operate. The developed ADAS can guide the bus driver to any charging station of a specified type, provided that the geographical coordinates of a charger are known. For one of the experiments, which falls outside the scope of this thesis, the ADAS was configured to operate with an electric charger located at the bus depot near the Garbary PKM station. Consequently, the focus was put on assessing the accuracy of the GNSS localization specifically in this area. Figure 6.7A highlights the locations of all the charging stations at the depot, with the one targeted by the system marked in green. Figure 6.7B shows the GNSS localization modes in the vicinity of the depot. Notably, the RTK FIXED mode was available throughout the entire approach path, enabling precise bus positioning during docking maneuvers using only GNSS navigation alone, without the need for any external sensors. This robustness is caused by the open space around the chargers and the absence of nearby high-rise buildings that could obstruct satellite signals.

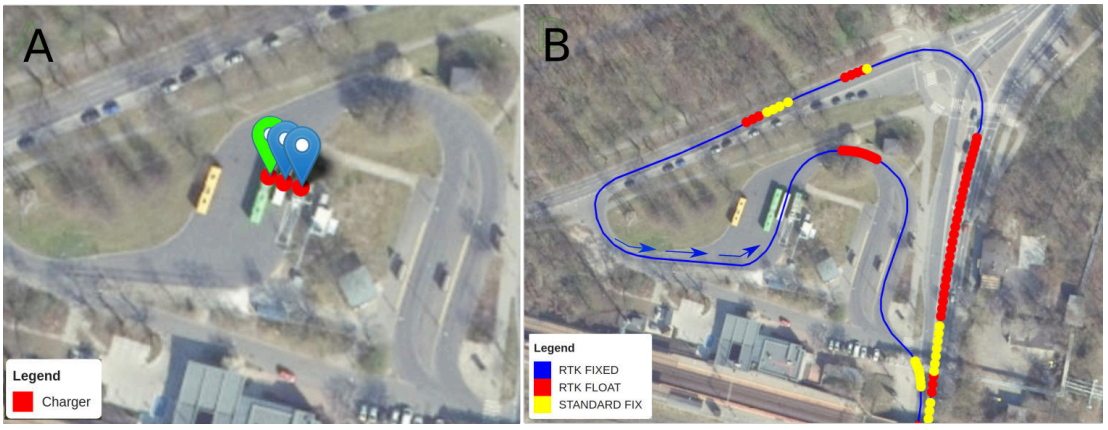

FIGURE 6.7: Location of charging stations at the Garbary PKM depot (A). The green marker shows the station that was used for system evaluation. GNSS localization status near the depot (B), including the approach path where the RTK FIXED mode was consistently available. Arrows indicate the driving direction.

6.2 Localization system for agricultural robot

6.2.1 Introduction

Another significant application of GNSS technology is in agriculture, where it plays a crucial role in improving precision farming practices. By enabling accurate positioning and navigation, GNSS systems support efficient crop management, automated machinery operation, and the development of agricultural robotics that contribute to more sustainable and productive farming methods. Moreover, the growing advantages of using GNSS receivers for crop management are driving the development of various hardware and software solutions to achieve high-precision positioning. Although many devices on the market perform effectively under field conditions, they are often designed specifically for agricultural applications and come with high costs. To make precision farming more accessible to smaller farms, it is essential to balance the trade-off between achieving adequate accuracy and the cost of components [78]. Furthermore, due to

environmental regulations imposed by the European Union [150], an increasing number of farms are required to adopt expensive solutions to minimize chemical usage, thereby promoting more ecological farming practices.

Taking into account these factors, this section explores the application of precision positioning techniques, such as the GNSS and VO, within the agricultural sector, specifically focusing on field work operations. The experimental part focuses on the evaluation of GNSS-based localization devices under real-world field conditions. It compares the positioning accuracy of the commercial and low-cost GNSS solution. It also examines the enhancements provided by integrating satellite navigation with VO, addressing short-term inaccuracies in GNSS readings, and demonstrates the practicality of such a solution in agricultural environments. This approach utilizes accessible hardware, such as laptops with built-in cameras, providing a cost-effective method to improve positioning accuracy in challenging field conditions.

6.2.2 Experimental setup

The experiment was conducted using an electric field robot presented in Figure 6.8A. It is a development platform designed for the research and testing of precision agriculture systems. During the tests, the vehicle was manually operated using the remote controller. The experiment was carried out in a small agricultural field with a size of approximately 30 by 200 m. The field area with one of the recorded sequences is shown in Figure 6.8B.

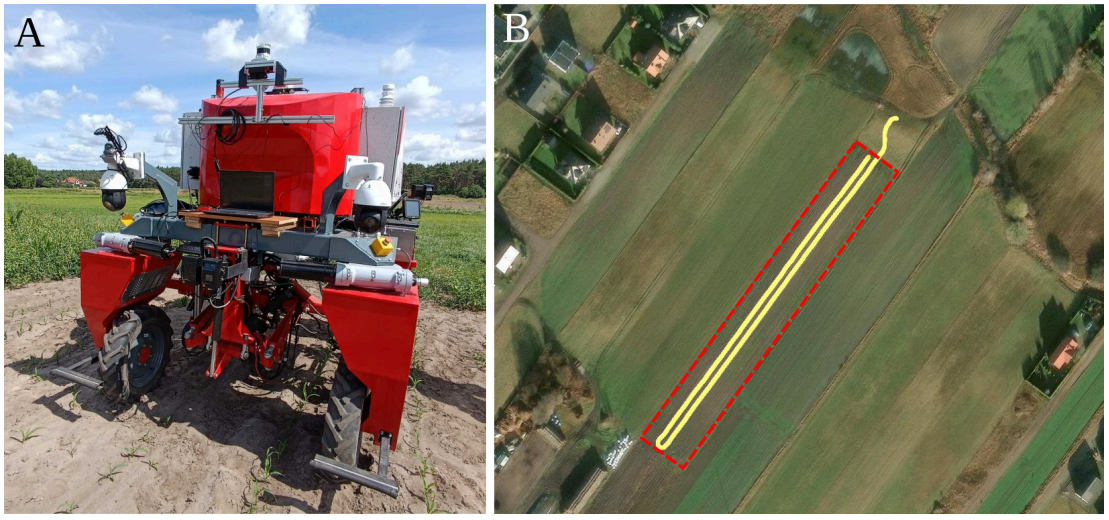

FIGURE 6.8: An electric robot used for the experiments (A) and the map presenting the field, where the tests were conducted (B). The red dashed rectangle indicates the field area, while yellow line shows one of the recorded trajectories.

The robot was equipped with additional sensors and hardware used for data collection. The setup consists of the following components:

- Topcon ASG-2 GNSS receiver
- two u-blox ZED-F9P GNSS receivers
- laptop with built-in camera

The first component used in the experiment is the Topcon ASG-2 GNSS receiver. It is designed primarily for agricultural applications and offers a positioning accuracy of up to 1 cm in the RTK mode, which makes it ideal for applications that require high precision, such as automated steering and precision planting. In addition, it includes an IMU with a 3-axis accelerometer, gyroscope, and magnetometer. This integration allows for better navigation stability and positioning, especially in challenging environments where GNSS signals may be obstructed. During the presented experiments, the AGS-2 receiver was used as a reference system and provided a ground-truth trajectory for comparison.

The next components were two u-blox ZED-F9P GNSS modules. They were set up in a moving base - rover configuration, analogous to the setup described in Section 6.1.2, which enables the estimation of both the position and heading of a robot. The ZED-F9P modules also offer high accuracy and are suitable for agricultural applications such as the precise control of agricultural machinery. At the same time, they are available at a relatively lower cost, making advanced positioning technologies accessible to smaller farms that may not be able to afford more expensive systems.

Both Topcon and u-blox receivers process signals from multiple satellite constellations, including GPS, Galileo, Beidou, and GLONASS, enhancing accuracy by using a larger number of satellites for positioning. Moreover, utilizing signals from multiple frequency bands significantly reduces the time necessary to get first position estimates. Furthermore, all GNSS receivers operated in RTK mode, benefiting from RTCM corrections provided by the ASG-Eupos system. These corrections were sent using NTRIP, and allowed further improvement of the location accuracy.

The final component was a laptop that served multiple functions. It was responsible for recording data from the GNSS receivers, acquiring RTCM corrections, and recording video from the built-in camera. For that reason, it was positioned at the front of the robot to ensure an optimal field of view.

6.2.3 Evalauation of the GNSS-based localization

The experiment involved conducting two different passes with the agricultural robot. The first one involved driving the robot back and forth to the end of the field, featuring a single turn and a straightforward path in both directions (see Fig. 6.8B). The second pass was relatively longer and included multiple turns, reflecting the routes that agricultural machinery often follows in real-world field operations. Both passes provided GNSS data that were subsequently used to evaluate the accuracy and consistency of the GNSS systems in a scenario that closely resembles actual agricultural practices. The evaluation was performed by computing the ATE between the reference trajectory from the commercial Topcon ASG-2 receiver and the one obtained from u-blox ZED-F9P. The trajectory plots along with the calculated error are shown in Figures 6.9 and 6.9 for the first and second passes, respectively. In addition, the numerical values of the calculated metrics are presented in Table 6.4.

The results indicate that the difference between the compared trajectories is within a few centimeters. However, some random disturbances can be observed in the ZED-F9P trajectories that

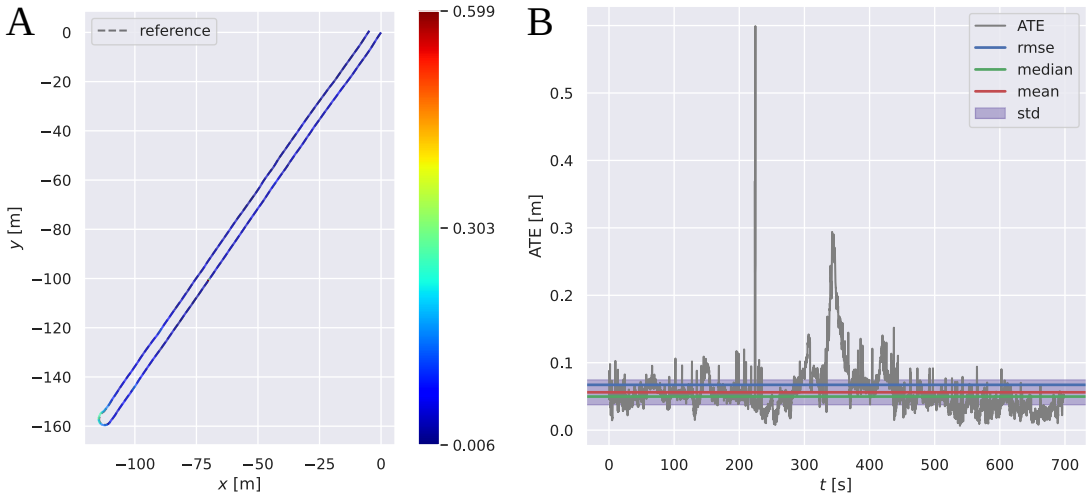

FIGURE 6.9: ATE calculated between the reference (Topcon ASG-2 receiver) and u-blox F9P trajectories (A). The plot presenting the error and its statistics over time (B). The results relate to the first pass of the robot.

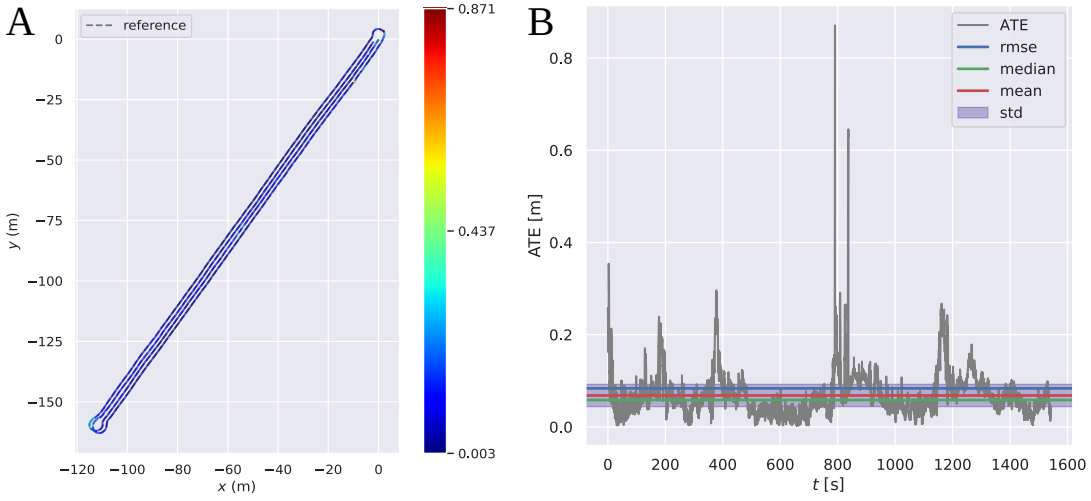

FIGURE 6.10: ATE calculated between the reference (Topcon ASG-2) and u-blox F9P trajectories (A). The plot presents the error and its statistics over time (B). The results relate to the second pass of the robot

TABLE 6.4: The Absolute Trajectory Error metrics calculated between the reference (Topcon ASG-2) and u-blox F9P trajectories for both passes carried out in the experiment. All values are expressed in meters, except for Sum of Squared Errors (SSE), which is in meters squared.

Run	RMS	mean	median	std	min	max	SSE
1	0.0671	0.0561	0.0497	0.0369	0.0064	0.5991	28.7672
2	0.0836	0.0682	0.0583	0.0483	0.0029	0.8706	99.3947

cause the error to increase randomly to a level of around 0.6–0.8 m. These deviations may be attributed to several potential factors. One possible cause is the temporal absence of the RTCM corrections, which are sent from the moving base to the rover module. These corrections are transmitted at a rate of 10 Hz and facilitate an accurate calculation of the heading along with the position. However, any interruptions or errors in the transmission of these messages could potentially have contributed to the inaccuracies visible on the trajectory.

Moreover, additional factors can affect signal quality and accuracy, such as multipath interference, where GNSS signals bounce off surfaces, such as flat terrain, before reaching the receiver, resulting in inaccurate position calculations. In addition, atmospheric conditions and the arrangement of GNSS satellites can also contribute to degraded signal performance. These combined factors may have played a role in the disturbances observed in the ZED-F9P trajectories, highlighting the complexity of achieving precise GNSS localization in dynamic field conditions.

6.2.4 Correcting GNSS trajectories with visual odometry

To validate the effectiveness of the GNSS trajectory correction approach described in Section 4.2, the trajectories recorded with the ZED-F9P receiver were integrated with VO measurements. As discussed previously, GNSS data can be subject to random positional disturbances, which can potentially be mitigated using the proposed optimization technique. An example segment of the trajectory where GNSS accuracy was reduced, yet ORB-SLAM3 provided reliable pose tracking, is shown in Figure 6.11A. Simultaneously, Figure 6.11B illustrates the variance in the coordinates obtained from GNSS receiver. These plots demonstrate that disturbances in satellite positioning can be detected by analyzing their variance. Consequently, such instances trigger the integration of the VO measurements, and after transforming and synchronizing the ORB-SLAM3 trajectory with the GNSS data, they can be added to the factor graph for optimization.

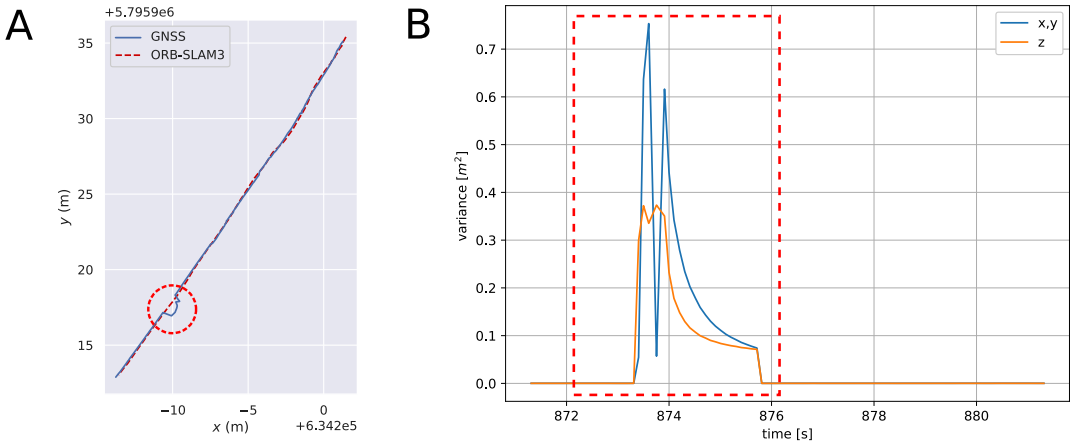

FIGURE 6.11: (A) Comparison between segments of the GNSS and ORB-SLAM3 trajectories. The red circle highlights where GNSS positioning accuracy degraded, while ORB-SLAM3 maintained reliable pose estimation. (B) The variance of x, y, z coordinates provided directly by the receiver. The segment of the plot highlighted by the red rectangle in (B) corresponds to the section of the trajectory indicated by the red circle in (A).

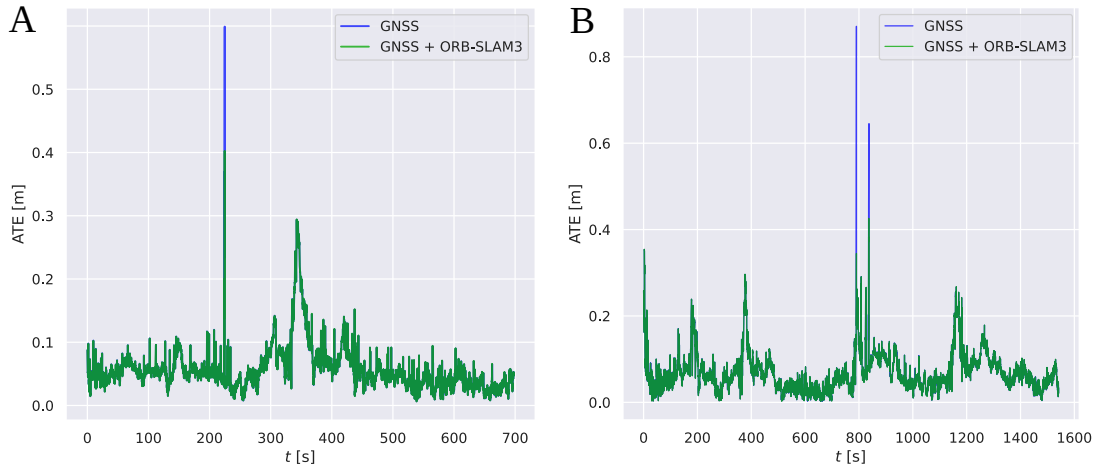

FIGURE 6.12: The comparison of ATE calculated for the GNSS path and the trajectory obtained using factor graph approach, which additionally integrates measurements from ORB-SLAM3. Plot (A) represents the results for the first pass, whereas (B) for the second.

The result of this trajectory correction is shown in Figure 6.13, which presents a section of the robot path before and after optimization. The disturbance highlighted in the red circle corresponds to the disturbance shown in Figure 6.11.

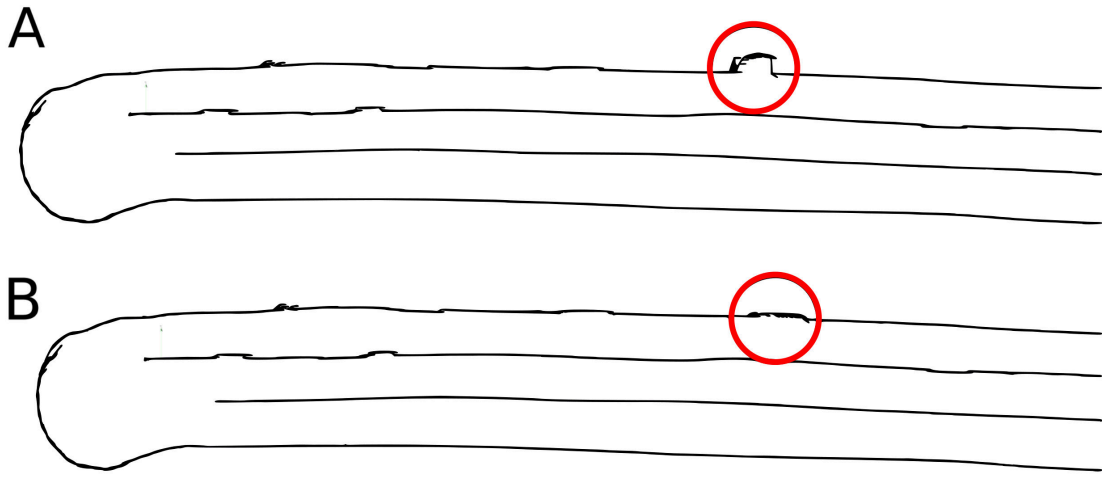

FIGURE 6.13: (A) The estimated trajectory before incorporating constraints derived from VO into the graph. (B) The trajectory after optimization, which enhanced the segment exhibiting noticeable positioning disturbances indicated by the red circle.

In addition, Table 6.5 presents the numerical values of ATE, calculated after applying trajectory corrections using the factor graph approach. The results correspond to the trajectories recorded during the first and second experimental runs, respectively. Compared to Table 6.4, which show analogous results before optimization, the most significant improvement is observed in the maximum error value. Improvements in the other columns are relatively minor. This is because only a short segment of both trajectories was affected by disturbances, resulting in minimal overall accuracy changes. However, in precision agriculture applications, even a few incorrect robot positions can have a notable impact on the results. In such cases, the presented approach

TABLE 6.5: ATE calculated for the trajectory estimated using ZED-F9P receiver during the first run. The trajectory was improved by integrating the visual odometry measurements into the factor graph. All values are expressed in m, except for Sum of Squared Errors (SSE), which is in m^2.

Run	**RMS**	**mean**	**median**	**std**	**min**	**max**	**SSE**
1	0.0660	0.0558	0.0497	0.0352	0.0062	0.4026	27.7628
2	0.0815	0.06787	0.0582	0.0451	0.0025	0.4261	94.4173

helps detect and mitigate GNSS disturbances by significantly reducing the unpredictable errors in the estimated trajectories.

In general, integrating GNSS and VO using factor graphs offers a suitable solution for precise and reliable localization, even in challenging environments. By combining satellite-based positioning with relative motion estimated using a visual system, factor graphs can effectively reduce positioning errors, particularly in areas with degraded GNSS signals. Experimental results underscore the robust performance of VO over short distances, even with low-cost cameras such as those built into laptops. As a result, such an approach proved to be particularly effective in compensating temporary GNSS degradations, such as interruptions in correction data.

Chapter 7

Conclusions

7.1 Summary

This dissertation explores various approaches to the SLAM problem, with a particular focus on different map representations and the integration of GNSS data using factor graph optimization. A key feature of the factor graph-based approach lies in its adaptability to integrate data from different sources, such as LiDAR or visual SLAM, loop closures, and raw GNSS measurements, which allows for robust and accurate state estimation. This flexibility makes it an ideal choice for a wide range of applications, including autonomous navigation and robotic exploration. The research presented in this dissertation contributes to the advancement of SLAM methodologies by demonstrating how map structures, optimization strategies, and multi-sensor fusion enhance localization accuracy and long-term robustness.

The feature-based PlaneLOAM system, introduced in Chapter 3, demonstrates significant potential to improve localization accuracy by minimizing errors during data association. This improvement is achieved by generating features from a larger set of points, which leads to more precise calculations of the plane equation. The experimental results confirm that PlaneLOAM consistently outperforms the baseline in terms of accuracy in all tested sequences. Additionally, the proposed approach optimizes pose estimation using a reduced number of points, as constraints are applied more selectively. This refined method improves the precision of the correspondences, ultimately leading to a better overall performance. Beyond evaluations on publicly available datasets, real-world experiments were conducted at the university campus and on public roads to assess the accuracy of the system in practical scenarios. These tests verified that the proposed feature-based approach, when combined with LiDAR, is suitable for localization and structure-rich mapping in large environments. It is important to note that the performance of the system is highly dependent on parameter selection, as these values significantly influence the size and number of generated features. Improper parameter tuning can lead to unreliable system behavior. Consequently, substantial effort was dedicated to determining a set of universal parameter values that ensure stable performance across various environments.

The second approach to map representation introduces a novel framework that simultaneously optimizes both poses and the 3D map, utilizing surfels. This method enables a continuous and structured representation of the environment while preserving rich geometric details. Unlike traditional point cloud-based mapping techniques, surfel-based models facilitate efficient data association, reduce redundancy, and improve map consistency. In addition, it incorporates a generalized LiDAR uncertainty model, which enables weighting of measurements during optimization. By accounting for sensor-specific uncertainties, this model enhances the reliability of pose and map estimation, particularly in challenging environments where measurement noise and inconsistencies can degrade accuracy. Experimental evaluations conducted on multiple publicly available datasets confirm that the proposed framework surpasses existing LiDAR-based mapping and localization methods in both robustness and precision. The system effectively estimates accurate poses and generates high-fidelity maps, while inherently suppressing dynamic artifacts and mitigating LiDAR skewing effects. This leads to the creation of clean, structured maps, which are crucial for applications demanding detailed and reliable 3D environmental representations, such as autonomous navigation, robotic exploration, and large-scale mapping. Furthermore, the ability of the framework to integrate surfel-based optimization with LiDAR uncertainty modeling presents new opportunities to improve long-term map consistency and computational efficiency.

An effective approach to improving the robustness of SLAM, particularly over long trajectories, is to combine it with GNSS data. Chapter 4 introduces GALS, a flexible framework for GNSS-augmented LiDAR SLAM that tightly integrates raw pseudorange and Doppler shift measurements with existing LiDAR SLAM algorithms. By leveraging raw GNSS observations, this method enhances SLAM accuracy and robustness, particularly in environments where LiDAR-based localization alone may struggle due to limited feature availability or accumulated drift. To validate the effectiveness of GALS, extensive experiments were conducted on five sequences from the challenging UrbanNav dataset. The results demonstrated consistent improvements across all tested sequences, even when using a smartphone-grade GNSS receiver, which highlights the adaptability of the framework to low-cost sensor configurations. In terms of ATE metric, GALS outperforms recently proposed optimization-based approaches evaluated on the same benchmark [103, 104]. These results underscore the ability of the framework to improve trajectory estimation accuracy and improve overall SLAM performance in complex urban settings. Beyond that, GALS shows significant potential for offline SLAM applications in large-scale urban mapping, where it can supplement existing SLAM systems with global GNSS measurements to ensure consistent map generation.

The first part of Chapter 6 presents the findings of a study on the ADAS designed to assist bus drivers in docking at charging stations located at bus stops or depots. The proposed system introduces a cost-effective solution that can be integrated into existing buses. The case-study emphasizes the practical aspects of system deployment, detailing how key GNSS functionalities were achieved with minimal hardware and installation costs. In addition, a comprehensive real-world evaluation was conducted in collaboration with a public transportation operator in Poznań, which further demonstrates the effectiveness of the system in an operational environment. Experimental results confirm that the localization solution, based on dual GNSS receivers with RTK corrections obtained through publicly available internet services, provides sufficient accuracy for the bus localization. These findings indicate that RTK-GNSS technology is mature

and reliable for supporting precise vehicle maneuvers, provided that operations avoid environments where GNSS signals may be degraded, such as areas with high-rise buildings or dense tree canopies. The results of using GNSS-only localization over a large urban area demonstrates also the limits of this localization technology, and suggest that integration with an affordable SLAM solution might be necessary in a production version. Although satellite signal interference remains a potential factor that affects localization accuracy, the study confirms that the RTCM corrections sent over the LTE network were consistently reliable, independent of environmental conditions or weather.

The second part of Chapter 6 explores the use of GNSS-based precision localization systems and GNSS-SLAM hybrid solutions in agricultural applications. The findings demonstrate that cost-effective differential GNSS can provide sufficient localization precision for practical real-world scenarios. Experimental results show that the ZED-F9P receiver system, which uses a dual module setup (one operating in moving base mode and the other in rover mode), achieved satisfactory positioning accuracy compared to a reference trajectory. However, to address temporary outages of the corrections, resulting in visible path artifacts, a hybrid approach was implemented, combining GNSS localization with low-cost VO. The study demonstrated that the factor-graph-based approach to integration proposed in the GALS system is also effective in this case. Here, the VO system, much more affordable than LiDAR-based SLAM or odometry, provided accurate and reliable localization over short distances, even when using low-cost hardware, such as a built-in laptop camera.

7.2 Thesis contribution

The ideas introduced in this thesis, along with their experimental validations, offer several advancements to the current state-of-the-art in robotics. The main contributions can be summarized as follows:

PlaneLOAM – a feature-based SLAM system that enhances the LOAM framework through a novel map representation using planar and linear features. While maintaining the core software architecture of the original system, the proposed approach improves data association, map management, and includes loop closure detection, leading to enhanced localization accuracy and robustness. Extensive evaluation on three diverse datasets using different LiDAR sensors demonstrates that PlaneLOAM significantly outperforms the baseline LOAM system in trajectory estimation. These results confirm the effectiveness of higher-level feature representations and advanced optimization techniques in improving SLAM performance for real-world applications. Additionally, the integration of loop closure and geometric constraints for global map optimization ensures greater long-term consistency.

MAD-BA – a unified global optimization framework that refines both the LiDAR poses and the structure of the map using a surfel-based representation. The system introduces a scalable and efficient BA method tailored specifically for LiDAR data. It leverages a generalized LiDAR uncertainty model to enhance the reliability of the optimization process, ensuring global consistency and robustness. The evaluation on public datasets confirms that the

developed system outperforms existing methods, producing more accurate trajectories and maps while inherently handling dynamic artifacts in LiDAR point clouds.

GALS – a framework for tightly coupled integration of LiDAR-based SLAM and GNSS measurements using a factor graph formulation. The developed system incorporates raw pseudorange and Doppler shift constraints directly into the factor graph, enabling robust localization with low-cost GNSS receivers. In addition, a filtering procedure designed to mitigate non-Gaussian noise in GNSS data ensures higher accuracy and global consistency in trajectory estimation. The evaluation carried out using the UrbanNav dataset demonstrates that the developed system improves localization using different LiDAR SLAM algorithms and GNSS receivers.

Real-world validation of GNSS-based localization for urban transport and agricultural applications. The work presented in the dissertation includes real-world experiments to assess the feasibility, reliability, and practical challenges of deploying GNSS-based systems in real-world scenarios. In the urban environment, the ADAS system for electric buses was developed and evaluated, highlighting both the technical feasibility and practical limitations of using GNSS-based localization system in public transportation. In the agricultural setting, the evaluation showed that the integration of VO with GNSS positioning can effectively compensate for short-term GNSS inaccuracies, providing a cost-effective method of improving localization.

When considering the hypothesis proposed at the beginning of the dissertation, it is possible to conclude that feature-based SLAM can be effectively formulated as an optimization problem, integrating constraints from GNSS measurements to overcome local minima issues of ICP-based methods. The experimental results confirm that the proposed systems achieve higher accuracy compared to traditional point-based approaches and validate the main hypothesis of this dissertation.

7.3 Future work

Although the developed systems have shown advancements over other existing approaches, there is still potential for further improvement. Future work can be focused on enhancing SLAM systems through a unified approach that leverages one of the proposed map representations while also performing multi-sensor fusion within a single framework. Moreover, to increase real-time processing capabilities, one key direction involves incorporating a keyframe-based optimization approach, as seen in visual SLAM systems such as ORB-SLAM, to reduce the number of poses requiring optimization. Furthermore, integrating marginalization techniques into global BA could further improve computational efficiency while maintaining accuracy.

Moreover, the crucial aspect of future SLAM advancements would be the integration of IMU alongside LiDAR and GNSS data. Integration of IMU would help minimize localization errors, especially in highly dynamic environments. This could be achieved by capturing rapid movements and sudden orientation changes, thereby enhancing the reliability of SLAM. Additionally, leveraging pre-integration methods will enable efficient use of IMU data without introducing unnecessary computational overhead.

Another key aspect of future SLAM research involves improving loop closure capabilities, as it is essential to reduce accumulated drift in SLAM systems, particularly in large-scale or long-term operations. A key challenge is ensuring robust data association between revisited locations under varying conditions, such as changes in lighting, weather, or dynamic objects. Future research can focus on integrating appearance-based and geometric-based loop closure detection methods or deep learning techniques to improve feature matching under different conditions. Another area of focus could be adaptive loop closure strategies that dynamically adjust the frequency of loop closure detection based on the system's confidence in the localization accuracy. For example, in GNSS-denied environments, the system can increase the loop closure frequency to compensate for the lack of absolute position updates, whereas in environments with strong GNSS signals, fewer loop closures may be required.

The last potential direction would be to address the challenges posed by dynamic environments by explicitly eliminating non-stationary objects such as pedestrians and vehicles. Although surfel representation naturally handles this to some extent, integrating deep learning-based object recognition methods into SLAM pipelines can further enhance this idea. The integration of such an approach into SLAM pipelines would improve robustness in urban and cluttered scenarios.

In conclusion, integrating LiDAR, GNSS and IMU data into a unified framework offers a promising approach to improving SLAM accuracy and robustness, resulting in more reliable and precise localization. Future research will prioritize optimizing this multi-sensor fusion to enhance scalability, adaptability, and real-time efficiency in diverse applications, including mobile robotics, urban navigation, and precision agriculture. By implementing these improvements, SLAM systems can achieve greater resilience and performance in complex and dynamic environments.

Bibliography

[1] A. Tourani, H. Bavle, J. L. Sanchez-Lopez, and H. Voos, "Visual SLAM: What are the current trends and what to expect?," *Sensors*, vol. 22, no. 23, 2022.

[2] X. Xu, L. Zhang, J. Yang, C. Cao, W. Wang, Y. Ran, Z. Tan, and M. Luo, "A review of multi-sensor fusion SLAM systems based on 3D LIDAR," *Remote Sensing*, vol. 14, no. 12, 2022.

[3] 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: Towards the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[4] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[5] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios, "Recent advances in indoor localization: A survey on theoretical approaches and applications," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1327–1346, 2017.

[6] A. Moura, J. Antunes, A. Dias, A. Martins, and J. Almeida, "Graph-SLAM approach for indoor UAV localization in warehouse logistics applications," in *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 4–11, 2021.

[7] L. Huang, "Review on LiDAR-based SLAM techniques," in *2021 International Conference on Signal Processing and Machine Learning (CONF-SPML)*, pp. 163–168, 2021.

[8] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[9] H. Yin, X. Xu, S. Lu, X. Chen, R. Xiong, S. Shen, C. Stachniss, and Y. Wang, "A survey on global LiDAR localization: Challenges, advances and open problems," *International Journal of Computer Vision*, vol. 132, no. 8, pp. 3139–3171, 2024.

[10] J. Bedkowski, *Large-Scale Simultaneous Localization and Mapping*. Cognitive Intelligence and Robotics, Springer, 2022.

[11] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," 2020.

[12] M. Li, H. Zhu, S. You, C. Tang, and Y. Li, "Efficient laser-based 3D SLAM in real time for coal mine rescue robots," in *IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems*, pp. 971–976, 2018.

[13] C. Campos, R. Elvira, J. J. Gómez Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," 2020.

[14] H. Strasdat, J. Montiel, and A. Davison, "Real-time mococular SLAM: why filter?," in *IEEE International Conference on Robotics and Automation*, (Anchorage), pp. 2657–2664, 2010.

[15] O. Wulf, A. Nüchter, J. Hertzberg, and B. Wagner, "Benchmarking urban six-degree-of-freedom simultaneous localization and mapping," *Journal Field Robotics*, vol. 25, no. 3, pp. 148–163, 2008.

[16] J. Będkowski, T. Röhling, F. Hoeller, D. Shulz, and F. Schneider, "Benchmark of 6D SLAM (6D Simultaneous Localization and Mapping) algorithms with robotic mobile mapping systems," *Foundations of Computing and Decision Sciences*, vol. 42, no. 3, pp. 275–295, 2017.

[17] F. Moosmann and C. Stiller, "Velodyne SLAM," in *IEEE Intelligent Vehicles Symposium*, pp. 393–398, 2011.

[18] T. Stoyanov, M. Magnusson, H. Andreasson, and A. J. Lilienthal, "Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1377–1393, 2012.

[19] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Robotics: Science and Systems*, (Rome), 2015.

[20] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, "Elastic LiDAR fusion: Dense map-centric continuous-time SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1206–1213, 2018.

[21] D. Droeschel and S. Behnke, "Efficient continuous-time SLAM for 3D lidar-based online mapping," in *IEEE International Conference on Robotics and Automation*, pp. 5000–5007, 2018.

[22] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in *Robotics: Science and Systems (RSS)*, 06 2018.

[23] S. Ferrari, L. D. Giammarino, L. Brizi, and G. Grisetti, "MAD-ICP: It is all about matching data – robust and informed LiDAR odometry," *IEEE Robotics and Automation Letters (RA-L)*, vol. 9, no. 11, pp. 9175–9182, 2024.

[24] J. Niedzwiedzki, P. Lipinski, and L. Podsedkowski, "IDTMM: incremental direct triangle mesh mapping," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 5416–5423, 2023.

[25] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang, "Deep learning on point clouds and its application: A survey," *Sensors*, vol. 19, no. 19, p. 4188, 2019.

[26] C. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017.

[27] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3D object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499, 2018.

[28] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: deep hierarchical feature learning on point sets in a metric space," in *31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 5105—-5114, Curran Associates Inc., 2017.

[29] M. Velas, M. Spanel, M. Hradis, and A. Herout, "CNN for IMU assisted odometry estimation using velodyne LiDAR," in *International Conference on Autnomous Robotic Systems and Computing*, pp. 71–77, 2018.

[30] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L^3-net: Towards learning based LiDAR localization for autonomous driving," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6382–6391, 2019.

[31] Y. Cho, G. Kim, and A. Kim, "Unsupervised geometry-aware deep LiDAR odometry," in *IEEE International Conference on Robotics and Automation*, pp. 2145–2152, 2020.

[32] K. Konolige, "Sparse sparse bundle adjustment," in *British Machine Vision Conference*, pp. 102.1–102.11, 2010.

[33] X.-F. Hana, J. Jin, J. Xie, M.-J. Wang, and W. A. Jiang, "Comprehensive review of 3D point cloud descriptors," 2018.

[34] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time.," in *Robotics: Science and Systems*, 2014.

[35] K. Ćwian, M. Nowicki, T. Nowak, and P. Skrzypczyński, "Planar features for accurate laser-based 3-D SLAM in urban environments," in *Advanced, Contemporary Control* (A. Bartoszewicz, J. Kabzinski, J. Kacprzyk, ed.), vol. 1196 of *AISC*, (Cham), pp. 941–953, 2020.

[36] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, 2018.

[37] X. Ji, L. Zuo, C. Zhang, and Y. Liu, "LLOAM: LiDAR odometry and mapping with loop-closure detection based correction," in *IEEE International Conference on Mechatronics and Automation*, pp. 2475–2480, 2019.

[38] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "SegMatch: Segment based place recognition in 3D point clouds," in *IEEE International Conference on Robotics and Automation*, pp. 5266–5272, 2017.

[39] X. Liu, L. Zhang, S. Qin, D. Tian, S. Ouyang, and C. Chen, "Optimized LOAM using ground plane constraints and SegMatch-based loop detection," *Sensors*, vol. 19, no. 24, p. 5419, 2019.

[40] J. Weingarten and R. Siegwart, "3D SLAM using planar segments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3062–3067, 2006.

[41] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3D mapping," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 424–441, 2010.

[42] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation," *Journal of Field Robotics*, vol. 27, no. 1, pp. 52–84, 2010.

[43] K. Lenac, A. Kitanov, R. Cupec, and I. Petrović, "Fast planar surface 3D SLAM using LIDAR," *Robotics and Autonomous Systems*, vol. 92, pp. 197–220, 2017.

[44] W. S. Grant, R. Voorhies, and L. Itti, "Efficient Velodyne SLAM with point and plane features," *Autonomous Robots*, vol. 43, no. 5, pp. 1207–1224, 2019.

[45] F. Pomerleau, T. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[46] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems*, (Seattle, USA), 2009.

[47] J. Deschaud, "IMLS-SLAM: Scan-to-model matching based on 3D data," in *IEEE International Conference on Robotics and Automation*, (Brisbane), pp. 2480–2485, 2018.

[48] J. Elseberg, D. Borrmann, K. Lingemann, and A. Nüchter, "Non-rigid registration and rectification of 3D laser scans," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1546–1552, 2010.

[49] F. Neuhaus, T. Koß, R. Kohnen, and D. Paulus, "MC2SLAM: Real-time inertial lidar odometry using two-scan motion compensation," in *Pattern Recognition* (T. Brox, A. Bruhn, and M. Fritz, eds.), (Cham), pp. 60–72, Springer International Publishing, 2019.

[50] E. Mendes, P. Koch, and S. Lacroix, "ICP-based pose-graph SLAM," in *IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 195–200, 2016.

[51] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, "Appearance-based loop detection from 3D laser data using the normal distributions transform," in *IEEE International Conference on Robotics and Automation*, pp. 23–28, 2009.

[52] R. Dubé, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: Segment-based mapping and localization using data-driven descriptors," *The International Journal of Robotics Research*, vol. 39, no. 2–3, pp. 339–355, 2020.

[53] K. Ćwian, M. R. Nowicki, J. Wietrzykowski, and P. Skrzypczyński, "Large-scale LiDAR SLAM with factor graph optimization on high-level geometric features," *Sensors*, vol. 21, no. 10, p. 3445, 2021.

[54] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[55] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3281–3288, 2011.

[56] F. Dellaert and G. Contributors, "borglab/gtsam." `https://github.com/borglab/gtsam`, May 2022.

[57] R. Kümerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^{2}o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation*, (Shanghai, China), pp. 3607–3613, 2011.

[58] L. D. Giammarino, L. Brizi, T. Guadagnino, C. Stachniss, and G. Grisetti, "MD-SLAM: Multi-cue direct SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11047–11054, IEEE, 2022.

[59] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 809–816, 2016.

[60] H. Strasdat, J. Montiel, and A. J. Davison, "Visual SLAM: Why filter?," *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[61] D. Rosen, K. Doherty, A. Espinoza, and J. Leonard, "Advances in inference and representation for simultaneous localization and mapping," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, 2021.

[62] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, "Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robotics and Autonomous Systems*, vol. 117, pp. 1–16, 2019.

[63] A. Cunningham, K. M. Wurm, W. Burgard, and F. Dellaert, "Fully distributed scalable smoothing and mapping with robust multi-robot data association," in *2012 IEEE International Conference on Robotics and Automation*, pp. 1093–1100, 2012.

[64] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice, "Generic node removal for factor-graph slam," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1371–1385, 2014.

[65] M. Mazuran, G. D. Tipaldi, L. Spinello, and W. Burgard, "Nonlinear graph sparsification for SLAM," in *Robotics: Science and Systems*, 2014.

[66] P. Agarwal, G. Grisetti, G. D. Tipaldi, L. Spinello, W. Burgard, and C. Stachniss, "Experimental analysis of dynamic covariance scaling for robust map optimization under bad initial estimates," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3626–3631, 2014.

[67] J. Deng, Q. Wu, X. Chen, S. Xia, Z. Sun, G. Liu, W. Yu, and L. Pei, "NeRF-LOAM: Neural implicit representation for large-scale incremental LiDAR odometry and mapping," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8184–8193, 2023.

[68] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[69] 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 *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5135–5142, 2020.

[70] M. Velas, M. Spanel, and A. Herout, "Collar line segments for fast odometry estimation from Velodyne point clouds," in *IEEE International Conference on Robotics and Automation*, pp. 4486–4495, 2016.

[71] 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 (RA-L)*, vol. 8, no. 2, pp. 1029–1036, 2023.

[72] X. Liu, Z. Liu, F. Kong, and F. Zhang, "Large-scale LiDAR consistent mapping using hierarchical LiDAR bundle adjustment," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 3, pp. 1523–1530, 2023.

[73] Z. Liu and F. Zhang, "Balm: Bundle adjustment for LiDAR mapping," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 3184–3191, 2021.

[74] L. Di Giammarino, E. Giacomini, L. Brizi, O. Salem, and G. Grisetti, "Photometric LiDAR and RGB-D bundle adjustment," *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[75] L. Wu, C. Le Gentil, and T. Vidal-Calleja, "VDB-GPDF: Online Gaussian process distance field with VDB structure," *IEEE Robotics and Automation Letters (RA-L)*, vol. 10, no. 1, pp. 374–381, 2025.

[76] J. Będkowski, H. Nowak, B. Kubiak, W. Studzinski, M. Janeczek, S. Karas, A. Kopaczewski, P. Makosiej, J. Koszuk, M. Pec, and K. Miksa, "A novel approach to global positioning system accuracy assessment, verified on LiDAR alignment of one million kilometers at a continent scale, as a foundation for autonomous driving safety analysis," *Sensors*, vol. 21, no. 17, p. 5691, 2021.

[77] S. J. Levoir, P. A. Farley, T. Sun, and C. Xu, "High-accuracy adaptive low-cost location sensing subsystems for autonomous rover in precision agriculture," *Industry Applications*, vol. vol. 1, pp. 74–94, 2020.

[78] N. V. Nguyen and W. Cho, "Performance evaluation of a typical low-cost multi-frequency multi-GNSS device for positioning and navigation in agriculture–part 2: Dynamic testing," *AgriEngineering*, vol. 5, pp. 127–140, 2023.

[79] G. Supper, N. Barta, A. Gronauer, and V. Motsch, "Localization accuracy of a robot platform using indoor positioning methods in a realistic outdoor setting," *Die Bodenkultur: Journal of Land Management, Food and Enviroment*, vol. 72(3), pp. 133–139, 2021.

[80] W. Zhang, L. Gong, S. Huang, S. Wu, and C. Liu, "Factor graph-based high-precision visual positioning for agricultural robots with fiducial markers," *Computers and Electronics in Agriculture*, vol. 201, p. 107295, 2022.

[81] Y. Yang and J. Xu, "GNSS reveiver autonomous integrity monitoring (RAIM) algorithm based on robust estimation," *Geodesy and Geodynamics*, vol. 7(2), pp. 117–123, 2016.

[82] J. Blanch, T. Walker, P. Enge, Y. Lee, B. Pervan, M. Rippl, A. Spletter, and V. Kropp, "Baseline advanced RAIM user algorithm and possible improvements," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51(1), pp. 713–732, 2015.

[83] S. Hewitson and J. Wang, "Extended receiver autonomous integrity monitoring (eRAIM) for GNSS/INS integration," *Journal of Surveying Engineering*, vol. 136, pp. 13–22, 2010.

[84] S. Bhattacharyya and D. Gebre-Egziaber, "Kalman filter–based RAIM for GNSS receivers," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51(3), pp. 2444–2459, 2015.

[85] D. Vieira, R. Orjuela, M. Spisser, and M. Basset, "Positioning and attitude determination for precision agriculture robots based on IMU and two RTK GPSs sensor fusion," *IFAC PapersOnLine*, vol. 55-32, pp. 60–65, 2022.

[86] G. Fan, J. Huang, D. Yang, and L. Rao, "Sampling visual SLAM with a wide-angle camera for legged mobile robots," *IET Cyber-Systems and Robotics*, vol. 4, no. 4, pp. 356–375, 2022.

[87] X. He, W. Gao, C. Sheng, Z. Zhang, S. Pan, L. Duan, H. Zhang, and X. Lu, "LiDAR-visual-inertial odometry based on optimized visual point-line features," *Remote Sensing*, vol. 14, no. 3, 2022.

[88] M. Cao, J. Zhang, and W. Chen, "Visual-inertial-laser SLAM based on ORB-SLAM3," *Unmanned Systems*, vol. 12, no. 1, pp. 1–10, 2023.

[89] P. Skrzypczyński and K. Ćwian, "Localization of agricultural robots: Challenges, solutions, and a new approach," in *Automation 2023: Key Challenges in Automation, Robotics and Measurement Techniques*, (Poland), 2023.

[90] Y. Ampatzidis, L. De Bellis, and A. Luvisi, "iPathology: Robotic applications and management of plants and plant diseases," *Sustainability*, vol. 9, p. 1010, 2017.

[91] S. Zaman, L. Comba, A. Biglia, D. Ricauda Aimonino, P. Barge, and P. Gay, "Cost-effective visual odometry system for vehicle motion control in agricultural environments," *Computers and Electronics in Agriculture*, vol. 162, pp. 82–94, 2019.

[92] M. Joerger and B. Pervan, "Autonomous ground vehicle navigation using integrated GPS and laser-scanner measurements," in *2006 IEEE/ION Position, Location, And Navigation Symposium*, pp. 988–997, 2006.

[93] G. Wan, X. Yang, R. Cai, H. Li, Y. Zhou, H. Wang, and S. Song, "Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, p. 4670–4677, IEEE, 2018.

[94] A. C. B. Chiella, H. N. Machado, B. O. S. Teixeira, and G. A. S. Pereira, "GNSS/LiDAR-based navigation of an aerial robot in sparse forests," *Sensors*, vol. 19, no. 19, 2019.

[95] 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*, vol. 68, no. 2, pp. 315–331, 2021.

[96] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *IEEE International Conference on Robotics and Automation*, pp. 3565–3572, 2007.

[97] X. Li, H. Wang, S. Li, S. Feng, X. Wang, and J. Liao, "GIL: a tightly coupled GNSS PPP/INS/LiDAR method for precise vehicle navigation," *Satellite Navigation*, vol. 2, no. 1, p. 26, 2021.

[98] F. Dellaert, "Factor graphs: Exploiting structure in robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. Volume 4, 2021, pp. 141–166, 2021.

[99] W. Wen and L. T. Hsu, "Towards robust GNSS positioning and real-time kinematic using factor graph optimization," in *IEEE International Conference on Robotics and Automation*, pp. 5884–5890, 2021.

[100] J. Liu, W. Gao, and Z. Hu, "Optimization-based visual-inertial SLAM tightly coupled with raw GNSS measurements," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11612–11618, 2021.

[101] L. Chang, X. Niu, T. Liu, J. Tang, and C. Qian, "GNSS/INS/LiDAR-SLAM integrated navigation system based on graph optimization," *Remote Sensing*, vol. 11, no. 9, 2019.

[102] G. He, X. Yuan, Y. Zhuang, and H. Hu, "An integrated GNSS/LiDAR-SLAM pose estimation framework for large-scale map building in partially GNSS-denied environments," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–9, 2021.

[103] T. Li, L. Pei, Y. Xiang, Q. Wu, S. Xia, L. Tao, X. Guan, and W. Yu, "P3-LOAM: PPP/LiDAR loosely coupled SLAM with accurate covariance estimation and robust RAIM in urban canyon environment," *IEEE Sensors Journal*, vol. 21, no. 5, pp. 6660–6671, 2021.

[104] W. Wen, X. Bai, L.-T. Hsu, and T. Pfeifer, "GNSS/LiDAR integration aided by self-adaptive Gaussian mixture models in urban scenarios: An approach robust to non-Gaussian noise," in *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pp. 647–654, 2020.

[105] Y. Gu, L.-T. Hsu, and S. Kamijo, "Towards lane-level traffic monitoring in urban environment using precise probe vehicle data derived from three-dimensional map aided differential GNSS," *IATSS Research*, vol. 42, no. 4, pp. 248–258, 2018.

[106] W. Lee, H. Cho, S. Hyeong, and W. Chung, "Practical modeling of GNSS for autonomous vehicles in urban environments," *Sensors*, vol. 19, no. 19, p. 4236, 2019.

[107] N. Viandier, J. Marais, E. De Verdalle, and A. Prestail, "Positioning urban buses: GNSS performances," in *8th International Conference on ITS Telecommunications*, pp. 51–55, 2008.

[108] Y. Yang, J. Yan, J. Guo, Y. Kuang, M. Yin, S. Wang, and C. Ma, "Driving behavior analysis of city buses based on real-time GNSS traces and road information," *Sensors*, vol. 21, no. 3, p. 687, 2021.

[109] R. Deng, Y. Liu, W. Chen, and H. Liang, "A survey on electric buses: Energy storage, power management, and charging scheduling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 1, pp. 9–22, 2021.

[110] C. Iclodean, N. Cordos, and B. O. Varga, "Autonomous shuttle bus for public transportation: A review," *Energies*, vol. 13, no. 11, p. 2917, 2020.

[111] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 6–22, 2014.

[112] M. M. Michalek, T. Gawron, M. Nowicki, and P. Skrzypczynski, "Precise docking at charging stations for large-capacity vehicles: An advanced driver-assistance system for drivers of electric urban buses," *IEEE Vehicular Technology Magazine*, vol. 16, no. 3, pp. 57–65, 2021.

[113] H. B. Swaminathan, A. Sommer, A. Becker, and M. Atzmueller, "Performance evaluation of GNSS position augmentation methods for autonomous vehicles in urban environments," *Sensors*, vol. 22, no. 21, p. 8419, 2022.

[114] N. Zhu, J. Marais, D. Bétaille, and M. Berbineau, "GNSS position integrity in urban environments: A review of literature," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2762–2778, 2018.

[115] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 829–846, 2018.

[116] J. Marais, C. Meurie, D. Attia, Y. Ruichek, and A. Flancquart, "Toward accurate localization in guided transport: Combining GNSS data and imaging information," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 188–197, 2014.

[117] N. Viandier, D. F. Nahimana, J. Marais, and E. Duflos, "GNSS performance enhancement in urban environment based on pseudo-range error model," in *Location and Navigation Symposium (PLANS)*, pp. 377–382, 2008.

[118] K. M. Ng, J. Johari, S. A. C. Abdullah, A. Ahmad, and B. N. Laja, "Performance evaluation of the RTK-GNSS navigating under different landscape," in *18th International Conference on Control, Automation and Systems (ICCAS)*, pp. 1424–1428, 2018.

[119] D. Janos, P. Kuras, and L. Ortyl, "Evaluation of low-cost RTK GNSS receiver in motion under demanding conditions," *Measurement*, vol. 201, p. 111647, 2022.

[120] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment — a modern synthesis," in *Vision Algorithms: Theory and Practice* (B. Triggs, A. Zisserman, and R. Szeliski, eds.), (Berlin, Heidelberg), pp. 298–372, Springer Berlin Heidelberg, 2000.

[121] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[122] C. Engels, H. Stewénius, and D. Nistér, "Bundle adjustment rules," *Photogrammetric computer vision*, vol. 2, no. 32, 2006.

[123] S. R. Buss and J. P. Fillmore, "Spherical averages and applications to spherical splines and interpolation," *ACM Transactions on Graphics*, vol. 20, no. 2, pp. 95–126, 2001.

[124] A. Bartoli, "On the non-linear optimization of projective motion using minimal parameters," in *European Conference on Computer Vision (ECCV)*, (Copenhagen), pp. 340–354, 2002.

[125] J. McNames, "A fast nearest-neighbor algorithm based on a principal axis search tree," *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, vol. 23, no. 9, pp. 964–976, 2001.

[126] G. Grisetti, T. Guadagnino, I. Aloise, M. Colosi, B. D. Corte, and D. Schlegel, "Least squares optimization: From theory to practice," *Robotics*, vol. 9, no. 3, p. 51, 2020.

[127] P. Skrzypczynski, "Spatial uncertainty management for simultaneous localization and mapping," in *IEEE International Conference on Robotics and Automation*, pp. 4050–4055, 2007.

[128] A. Wehr and U. Lohr, "Airborne laser scanning—an introduction and overview," *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, vol. 54, no. 2-3, pp. 68–82, 1999.

[129] E. P. Baltsavias, "Airborne laser scanning: basic relations and formulas," *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, vol. 54, no. 2-3, pp. 199–214, 1999.

[130] S. Huang, Z. Gojcic, Z. Wang, F. Williams, Y. Kasten, S. Fidler, K. Schindler, and O. Litany, "Neural LiDAR fields for novel view synthesis," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 18236–18246, 2023.

[131] F. Xu, Y. Wang, X. Yang, B. Zhang, and F. Li, "Correction of linear-array lidar intensity data using an optimal beam shaping approach," *Optics and Lasers in Engineering*, vol. 83, pp. 90–98, 2016.

[132] Y. Hu, A. Hou, Q. Ma, N. Zhao, S. Xu, and J. Fang, "Analytical formula to investigate the modulation of sloped targets using LiDAR waveform," *Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–12, 2021.

[133] T. Schops, T. Sattler, and M. Pollefeys, "Bad SLAM: Bundle adjusted direct RGBD-D SLAM," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 134–144, 2019.

[134] K.-W. Chiang, G.-J. Tsai, H.-J. Chu, and N. El-Sheimy, "Performance enhancement of INS/GNSS/refreshed-SLAM integration for acceptable lane-level navigation accuracy," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 2463–2476, 2020.

[135] R. Roriz, J. Cabral, and T. Gomes, "Automotive LiDAR technology: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6282–6297, 2022.

[136] L.-T. Hsu, N. Kubo, W. Wen, W. Chen, Z. Liu, T. Suzuki, and J. Meguro, "UrbanNav: An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas," in *34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, pp. 226–256, 2021.

[137] J. Zhang and S. Singh, "Low-drift and real-time LiDAR odometry and mapping," *Autonomous Robots*, vol. 41, no. 2, pp. 401–416, 2017.

[138] T. Takasu, "RTKLIB: Open source program package for RTK-GPS," in *FOSS4G 2009*, (Tokyo), 2009.

[139] S. Perea, M. Meurer, M. Rippl, B. Belabbas, and M. Joerger, "URA/SISA analysis for GPS and Galileo to support ARAIM," *Journal of the Institute of Navigation*, vol. 64, pp. 237–254, 2017.

[140] A. Shetty and G. X. Gao, "Covariance estimation for GPS-LiDAR sensor fusion for UAVs," in *30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017)*, 2017.

[141] M. S. A. Mahmud, M. S. Z. Abidin, A. A. Emmanuel, and H. S. Hasan, "Robotics and automation in agriculture: Present and future applications," *Applications of Modelling and Simulation*, vol. 4, pp. 130–140, 2020.

[142] D. Mulla and R. Khosla, "Historical evolution and recent advances in precision farming," in *Soil-Specific Farming* (R. Lal and B. A. Stewart, eds.), pp. 1–36, Publishing House: CRC Press, United States, 2015.

[143] D. Radocaj, I. Plascak, G. Heffer, and M. Jurisic, "A low-cost global navigation satellite system positioning accuracy assessment method for agricultural machinery," *Applied Sciences*, vol. 12, p. 693, 2022.

[144] J. Si, Y. Niu, J. Lu, and H. Zhang, "High-precision estimation of steering angle of agricultural tractors using GPS and low-accuracy MEMS," *IEEE Transactions on vehicular technology*, vol. vol. 68, no. 12, pp. 11738–11745, 2019.

[145] J. Guo, X. Li, Z. Li, L. Hu, G. Yang, C. Zhao, D. Fairbairn, D. Watson, and M. Ge, "Multi-GNSS precise point positioning for precision agriculture," *Precise Agric*, vol. 19, pp. 895–911, 2018.

[146] M. Perez-Ruiz and S. K. Upadhyaya, "GNSS in precision agricultural operations," in *New Approach of Indoor and Outdoor Localization Systems* (F. B. Elbahhar and A. Rivenq, eds.), ch. 1, IntechOpen, 2012.

[147] N. V. Nguyen, W. Cho, and K. Hayashi, "Performance evaluation of a typical low-cost multi-frequency multi-GNSS device for positioning and navigation in agriculture – part 1: Static testing," *Smart Agricultural Technology*, vol. 1, p. 100004, 2021.

[148] P. Catania, A. Comparetti, P. Febo, G. Morelli, S. Orlando, E. Roma, and M. Vallone, "Position accuracy comparision of GNSS receivers used for mapping and guidance of agricultural machines," *Agronomy*, vol. 10, p. 924, 2020.

[149] A. Tayebi, J. Gomez, M. Fernandez, F. de Adana, and O. Gutierrez, "Low-cost experimental application of real-time kinematic positioning for increasing the benefits in cereal crops," *International Journal of Agricultural and Biological Engineering*, vol. 14, pp. 175–181, 2021.

[150] G. Gargano, F. Licciardo, M. Verrascina, and B. Zanetti, "The agroecological approach as a model for multifunctional agriculture and farming towards the european green deal 2030—some evidence from the italian experience," *Sustainability*, vol. 13(4), p. 2215, 2021.

[151] V. Marinoudi, M. Lampridi, D. Kateris, S. Pearson, C. G. Sorensen, and D. Bochtis, "The future of agricultural jobs in view of robotization," *Sustainability*, vol. 13, p. 12109, 2021.

[152] A. Botta, P. Cavallone, L. Baglieri, G. Colucci, L. Tagliavini, and G. Quaglia, "A review of robots, perception, and tasks in precision agriculture," *Applied Mechanics*, vol. 3, no. 3, pp. 830–854, 2022.

[153] "Precision agriculture: an opportunity for EU farmers- potential support with the CAP 2014-2020. directorate-general for internal policies. policy department b. structural and cohesion polices.." European Parliament, 2014.

[154] P. Skrzypczyński, "Mobile robot localization: Where we are and what are the challenges?," in *Automation 2017* (R. Szewczyk, C. Zieliński, and M. Kaliczyńska, eds.), (Cham), pp. 249–267, Springer International Publishing, 2017.

[155] F. Shu, P. Lesur, Y. Xie, A. Pagani, and D. Stricker, "SLAM in the field: An evaluation of monocular mapping and localization on challenging dynamic agricultural environment," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1760–1770, 2021.

[156] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[157] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 834–849, Springer International Publishing, 2014.

[158] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[159] J. Cremona, R. Comelli, and T. Pire, "Experimental evaluation of visual-inertial odometry systems for arable farming," *Journal of Field Robotics*, vol. 39, no. 7, pp. 1121–1135, 2022.

[160] K. Ćwian, M. R. Nowicki, and P. Skrzypczyński, "GNSS-augmented LiDAR SLAM for accurate vehicle localization in large scale urban environments," in *17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, (Singapore), pp. 701–708, 2022.

[161] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 4, pp. 80–92, 1991.

[162] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2012.

[163] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition*, (Rhode Island), pp. 3354–3361, 2012.

[164] G. Kim, Y. S. Park, Y. Cho, J. Jeong, and A. Kim, "MulRan: Multimodal range dataset for urban place recognition," in *IEEE International Conference on Robotics and Automation*, pp. 6246–6253, 2020.

[165] u-blox, "ZED-F9P-04B high precision GNSS module." `https://content.u-blox.com/sites/default/files/ZED-F9P-04B_DataSheet_UBX-21044850.pdf`. Accessed: 2025-02-28.

[166] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4353–4360, 2020.

[167] L. Brizi, E. Giacomini, L. D. Giammarino, S. Ferrari, O. Salem, L. D. Rebotti, and G. Grisetti, "VBR: A vision benchmark in Rome," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15868–15874, 2024.

[168] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Intl. Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1231–1237, 2013.

[169] T. Nowak, M. R. Nowicki, and P. Skrzypczyński, "Vision-based positioning of electric buses for assisted docking to charging stations," *International Journal of Applied Mathematics and Computer Science*, vol. 32, no. 4, pp. 583–599, 2022.

[170] V. Ho, K. Rauf, I. Passchier, F. Rijks, and T. Witsenboer, "Accuracy assessment of RTK GNSS based positioning systems for automated driving," in *15th Workshop on Positioning, Navigation and Communications (WPNC)*, pp. 1–6, 2018.

[171] M. R. Nowicki, "A data-driven and application-aware approach to sensory system calibration in an autonomous vehicle," *Measurement*, vol. 194, p. 111002, 2022.

[172] A. Rietdorf, C. Daub, and P. Loef, "Precise positioning in real-time using navigation satellites and telecommunication," in *3rd Workshop on Positioning, Navigation and Communication (WPNC)*, pp. 209–218, 2006.