Eötvös Loránd University

Faculty of Informatics

Erasmus Mundus Joint Master
in Intelligent Field Robotic Systems

Robust LiDAR-Inertial Localization with Prior Maps in GNSS-Challenged Environments

Author:
Eliyas Kidanemaraim Abraha
Intelligent Field Robotics Systems, MSc

Internal supervisor:
Zoltan Istenes
Associate Professor at ELTE

External supervisor:
Mohammad Aldibaja
Associate Professor at SMART

Budapest, 2025

Abstract

This thesis presents a real-time, map-based localization system for autonomous robots operating in large-scale, GNSS-denied, and partially dynamic environments. The system integrates LiDAR-Inertial Odometry (FAST-LIO2) with scan-to-map registration using the multithreaded Normal Distributions Transform (NDT-OMP) algorithm. These components are fused through a sliding-window factor graph optimization framework, enabling both high-frequency motion estimation and global drift correction.

To support scalability and computational efficiency, the system incorporates a dynamic submap loading strategy that retrieves only the local region of the prior 3D map relevant to the robot's current pose. The use of NDT-OMP allows for parallel scan matching, contributing to real-time operation with a per-frame latency under 23 milliseconds. Additionally, a deep learning-based 3D object detection module (CenterPoint) is integrated to remove dynamic elements from the LiDAR input, improving registration stability in semi-dynamic scenes.

Experimental evaluations on both benchmark and custom datasets demonstrate that the system achieves centimeter- to decimeter-level localization accuracy while maintaining real-time performance. The system was further tested under simulated environmental degradation, including moderate fog noise, and maintained robust localization performance. It also proved effective in feature-sparse regions and unmapped transition zones, where scan-to-map alignment is typically unreliable. Finally, a modular architectural extension is proposed to support future integration of additional sensing modalities—such as vision or radar—enhancing adaptability in perceptually degraded environments. These contributions collectively advance the robustness, efficiency, and extensibility of real-time localization for autonomous systems.

Acknowledgements

I would like to express my sincere gratitude to my academic supervisor, **Dr. Zoltan Istenes**, and my industrial supervisor, **Dr. Mohammed Aldibaja**, for their continued guidance, encouragement, and generosity in sharing their knowledge throughout the course of my master's thesis. Their insights and support were instrumental in shaping the quality and direction of this work. I also extend my sincere appreciation to the **Smart Mechatronics group at Saxion University of Applied Sciences** for hosting my thesis and providing a supportive and engaging research environment.

I am deeply thankful to the professors at the **University of Girona** and **Eötvös Loránd University** for their dedicated instruction and support from the beginning of the **IFRoS program**. Their efforts have greatly contributed to my growth and understanding in the field of robotics. I also thank the **IFRoS program coordinators** for their exceptional organization and continuous assistance throughout the program.

I would like to thank all my **classmates and friends** in the IFRoS program for their companionship, teamwork, and encouragement during this enriching journey across countries and institutions.

On a personal note, I am deeply grateful to my **parents** and my two **wonderful sisters** for their unwavering love, constant motivation, and emotional support. Their belief in me has been a continuous source of strength and inspiration throughout this academic journey.

Contents

Abstract i

Acknowledgements

List of Figures iii

List of Algorithms v

List of Abbreviations vi

1 Introduction 1

1.1 Motivation . . . 1
1.2 Objective . . . 2
1.3 Research Questions . . . 3
1.4 Scope and limitations . . . 3
1.4.1 Scope . . . 3
1.4.2 Limitations . . . 4
1.5 Overview of the thesis structure . . . 4

2 Litreature Review 6

2.1 Overview of Relevant Literature . . . 6
2.1.1 LiDAR and LiDAR-Inertial Odometry Techniques . . . 6
2.1.2 SLAM and Loop Closure-Based Localization . . . 8
2.1.3 Localization Using Prior Maps . . . 8
2.1.4 Point Cloud Registration Techniques . . . 9
2.1.5 Dynamic Object Removal Techniques . . . 10
2.2 Theoretical Background . . . 11
2.2.1 Normal Distributions Transform (NDT) . . . 11
2.2.2 Sensor Fusion Approaches . . . 12
2.3 Research Gaps . . . 15

3 Methodology **17**
3.1 System Overview . . . 17
3.1.1 LiDAR Scan Pre-Processing . . . 19
3.1.2 Map Pre-Processing . . . 20
3.2 LIDAR Inertia Odometry . . . 21
3.3 Scan-to-Map Matching . . . 22
3.4 Real-Time Factor Graph Optimization . . . 23
3.4.1 Factor Graph Formulation . . . 24
3.4.2 Maximum A Posteriori (MAP) Estimation . . . 25
3.4.3 Fixed-Lag Smoothing and Marginalization . . . 25
3.4.4 Outlier Rejection via Mahalanobis Distance . . . 28
3.5 Dynamic Object Detection and Removal . . . 28
3.5.1 Dynamic Object Detection . . . 28
3.5.2 Detected Object removal . . . 29
3.6 Environmental Noise Simulation . . . 29

4 Experimental Setup and Result **31**
4.1 Experimental Environment . . . 31
4.1.1 Hardware and Software Setup . . . 31
4.1.2 Datasets Collection and Map Preparation . . . 32
4.2 Experimental Evaluation . . . 33
4.2.1 Evaluation Metrics and Methodology . . . 33
4.2.2 Localization Performance . . . 34
4.3 Summary of Results . . . 45

5 Discussion **46**
5.1 Interpretation of Results . . . 46
5.1.1 Accuracy and Robustness . . . 46
5.1.2 Real-Time Performance . . . 47
5.1.3 Environmental Adaptability . . . 47
5.2 Implications . . . 48
5.3 Limitations and Future Work . . . 48

6 Conclusion **50**
6.1 Summary of Findings . . . 50

A Trajectory Plot **51**

Bibliography **53**

List of Figures

2.1 Architecture of FAST-LIO2: A tightly coupled LiDAR-Inertial Odometry [13] 7
2.2 Overview of CenterPoint 3d object detection framework. [27] 11
2.3 Factor graph representation: green nodes are state variables (poses), blue nodes are motion model factors, and red nodes represent sensor measurements. 14

3.1 System architecture of the proposed map-based localization pipeline. The system fuses high-frequency LiDAR-inertial odometry with global map-based pose corrections using scan-to-map matching and factor graph optimization. 18
3.2 Overview of LiDAR scan pre-processing steps. The raw point cloud undergoes spatial cropping, statistical outlier removal, voxel-grid downsampling, and dynamic object removal. 20
3.3 Demonstration of fixed-lag smoothing with a lag size of $n = 4$ in a factor graph-based localization system. 27

4.1 Hardware setup used for real-world data collection and testing: (a) Hunter robotic platform; (b) onboard sensors. 32
4.2 Saxion Sequence 01 – Trajectory Alignment with Zoomed Comparison 35
4.3 Absolute Pose Error Comparison of (a) Saxion Sequence 2 and (b) Saxion Sequence 3 36
4.4 Computation characteristics of the proposed localization system: (a) Sliding-window optimization scalability and (b) real-time frame-wise timing breakdown. 37
4.5 Top-down view of the localization trajectory overlaid on the pre-built 3D map. The red line indicates the estimated trajectory, while the white rectangle structures represent dynamically loaded local submaps within a fixed radius during runtime. This demonstrates how local map tiling supports scalable scan-to-map registration. 37
4.6 Detecting and removing dynamic objects from 3d LIDAR point-cloud 38
4.7 Point-cloud Registration Performance with and without Dynamic Object Removal 39
4.9 Error compression in transition zone 41

4.10 Point-cloud snapshots under increasing composite geometric noise levels: (a) Original scan; (b) Mild noise; (c) Moderate noise; (d) Severe noise. 42
4.11 Error comparison in transition zone 43
4.12 KITTI Sequence 05 – Trajectory Alignment with Zoomed Comparison 45

5.1 Comparison of sensor fusion architectures. (a) The current design integrates FAST-LIO2 with NDT-based map matching in a tightly coupled manner. (b) The proposed approach restructures the system into a modular factor graph, supporting IMU pre-integration and extensibility to visual or radar factors for increased robustness in degraded environments. 49

A.1 Saxion Sequence 02– Trajectory Alignment with Zoomed Comparison 51
A.2 Saxion Sequence 03 – Trajectory Alignment with Zoomed Comparison 52

List of Algorithms

1 LiDAR-Inertial Localization with Prior Map 19

List of Abbreviations

APE Absolute Pose Error

DoF Degrees of Freedom

EKF Extended Kalman Filter

FAST-LIO2 Fast LiDAR-Inertial Odometry 2

FOV Field of View

GNSS Global Navigation Satellite System

ICP Iterative Closest Point

IMU Inertial Measurement Unit

iSAM2 Incremental Smoothing and Mapping 2

LIDAR Light Detection and Ranging

LIO LiDAR-Inertial Odometry

MAP Maximum A Posteriori

MCL Monte Carlo Localization

NDT Normal Distributions Transform

NDT-OMP NDT with OpenMP (multithreaded) implementation

OMP Open Multi-Processing

PCL Point Cloud Library

RMSE Root Mean Square Error

ROS Robot Operating System

RTK Real-Time Kinematic

SLAM Simultaneous Localization and Mapping

STD Standard Deviation

TF Transform (coordinate transform in ROS)

Chapter 1

Introduction

1.1 Motivation

High-precision and reliable localization is crucial for autonomous vehicles and robots, as it directly affects tasks like path planning, obstacle avoidance, and other repetitive missions [1]. Typically, positioning relies on Global Navigation Satellite Systems (GNSS) and Inertial Navigation Systems (INS), which complement each other. GNSS can provide absolute position without drift especially when using differential signals while INS delivers frequent updates to track movement[2]. However, in urban areas with tall buildings or indoor environments, GNSS signals can be blocked or distorted, and INS alone may accumulate significant drift over time[3]. Moreover, these systems do not inherently provide information about the surrounding environment.

To address such limitations, researchers have adopted onboard sensors like LiDAR, monocular cameras, and stereo cameras. One widely used method is Simultaneous Localization and Mapping (SLAM), which enables a robot to estimate its position in a completely new environment without needing a prior map. While SLAM is flexible and powerful, it can drift if loop closures are infrequent or the robot cannot revisit known areas to correct its estimates.

An alternative that improves accuracy and stability is map-based localization, where a pre-built map (created beforehand by SLAM, GNSS-based surveys, or other methods) is used as a global reference [4]. In this approach, the robots's current sensor data such as LiDAR scans are matched against the detailed information in the map to determine its position. This matching process helps minimize drift and provides robust localization, even in environments where GNSS signals are unavailable or unreliable. Once the map is constructed, indoor and outdoor autonomous systems can repeatedly use it to perform their tasks more reliably, whether that involves delivering goods, patrolling a facility, or

navigating urban streets. As a result, map-based localization is considered an indispensable component of future autonomous navigation, especially when consistent high accuracy and global reference information are required.

Despite its advantages, map-based localization poses several challenges.First,real-time performance requires efficient management of high-resolution 3D maps and the use of fast, precise scan-to-map matching algorithm for accurate pose estimation. Second, integrating local motion estimation (e.g., via LiDAR-Inertial Odometry) with global map alignment must be handled carefully to ensure both consistency and computational feasibility. Finally, the presence of dynamic objects and adverse environmental conditions further complicates the localization pipeline.

This thesis addresses these challenges by proposing a hybrid system that combines real-time LiDAR-Inertial Odometry (FAST-LIO2), efficient map-matching via multithreaded NDT (NDT-OMP), and a sliding-window factor graph fusion approach. The system dynamically loads submaps around the robot based on a configurable radius, allowing scalable scan-matching while maintaining global consistency.In addition, a learning-based object detection module identifies and removes dynamic objects from the LiDAR scans, further enhancing robustness. Together, these components form a real-time, accurate, and robust localization framework tailored for large-scale, semi-dynamic environments.

1.2 Objective

The primary objective of this thesis is to develop a real-time, LiDAR-based localization framework that achieves both high-frequency motion estimation and drift-free global consistency in large-scale environments. This is accomplished by integrating LiDAR-Inertial Odometry with map-based scan matching, and fusing them through a computationally efficient sliding-window factor graph.

- To integrate a robust LiDAR-Inertial Odometry(LIO) module using FAST-LIO2, capable of providing real-time pose estimates by tightly coupling LiDAR and IMU data.
- To develop a scan-to-map matching module by integrating a multithreaded Normal Distributions Transform (NDT-OMP) algorithm, enabling real-time alignment of incoming LiDAR scans with a pre-built 3D point cloud map.
- To fuse LiDAR-Inertial Odometry(LIO) and map matching results using a sliding-window factor graph optimization approach, enabling efficient, bounded optimization that retains local consistency while discarding outdated states.

- To integrate a learning-based dynamic object detection module into the LiDAR processing pipeline.
- To design and implement a grid-based submap management system that loads only the relevant local tiles within a configurable radius of the robot's position and incrementally replaces out-of-range tiles without reloading the full map.
- To evaluate the proposed system on benchmark datasets and custom scenarios, measuring performance in terms of localization accuracy, computation time, memory usage, and robustness under varying environmental conditions.

1.3 Research Questions

The primary questions guiding this research are:

1. How can LiDAR-Inertial odometry(FAST-LIO2) and NDT-based map matching be effectively fused to enhance real-time localization accuracy and robustness?
2. How does integrating a prior map improve the accuracy and robustness of LiDAR-based localization in GNSS-challenged environments?
3. How does the proposed system maintain real-time performance and scalability in large-scale environments?
4. Does the proposed localization method maintain accurate localization in feature sparse and noisy environmental condition?

1.4 Scope and limitations

1.4.1 Scope

The proposed localization system is designed for real-time operation in diverse environments, including indoor spaces, outdoor areas, and urban environments. It assumes the availability of a high-resolution 3D LIDAR map of the operation environment.Furthermore, the system is capable of maintaining localization even when the robot temporarily traverses boundary or transition zones where the pre-built map is sparse or unavailable. The system provides accurate pose estimation by fusing LIDAR-Inertial Odometry with scan-to-map matching, and its particularly suited scenarios where GNSS is unreliable or unavailable.

1.4.2 Limitations

- The system does not perform global localization (i.e., it cannot determine an unknown initial pose within the map). It assumes the robot starts with a known or approximated initial position close to its true location.

1.4 Contributions of the Thesis

This thesis contributes to the advancement of real-time, map-based localization in autonomous systems by addressing key challenges related to accuracy, robustness, and scalability in GNSS-denied and semi-dynamic environments. The main contributions of this work are summarized as follows:

1. **Hybrid Localization Framework:** Developed a real-time localization pipeline that combines FAST-LIO2 odometry with NDT-based scan-to-map matching within a sliding-window factor graph, achieving drift-resilient 6-DoF pose estimation.

2. **Dynamic Submap Loading:** Implemented an adaptive, tile-based map management strategy that loads only relevant submaps based on the robot's estimated position, ensuring scalability and memory efficiency in large-scale environments.

3. **Learning-Based Dynamic Object Removal:** Integrated a deep-learning-based CenterPoint detector into the LiDAR preprocessing stage to identify and remove dynamic objects, improving the robustness of scan matching in semi-dynamic scenes.

4. **Real-Time Performance:** Achieved an average per-frame processing latency below 23 milliseconds through multithreaded NDT registration, voxel filtering, and efficient scan-to-map alignment.

5. **Robustness in Degraded Conditions:** Demonstrated reliable localization performance in feature-sparse areas and under moderate visibility degradation (e.g., fog), with fallback to high-rate odometry when scan matching becomes unreliable.

1.5 Overview of the thesis structure

This thesis is organized as follows. **Chapter 2** reviews relevant literature on LiDAR-based localization, covering LiDAR–Inertial Odometry, SLAM, map-based methods, scan matching, sensor fusion, and learning-based dynamic object removal. **Chapter 3** details the proposed system architecture, outlining the design and integration of each module.

Chapter 4 presents the experimental setup and results, including performance metrics and comparisons with baseline methods. **Chapter 5** discusses key findings, addresses system limitations, and outlines potential directions for improvement and future work. Finally, **Chapter 6** concludes the thesis by summarizing the main contributions.

Chapter 2

Litreature Review

2.1 Overview of Relevant Literature

This section provides an overview of the foundational literature relevant to the design and development of the proposed localization system. It reviews key concepts and techniques in LiDAR and LiDAR-Inertial Odometry, SLAM, map-based localization, point cloud registration, and dynamic object detection using 3D LiDAR data.

2.1.1 LiDAR and LiDAR-Inertial Odometry Techniques

Localization is a crucial capability for mobile robots, referring to the process of determining the robot's position and orientation in a given environment. Early attempts at localization depended heavily on wheel encoders, commonly known as wheel odometry, but this method suffered from significant drift whenever the wheels slipped or the ground was uneven [5]. As robotics advanced, researchers explored range sensors and visual sensors to improve motion estimation [6]. At the same time, algorithms like Iterative Closest Point (ICP) emerged, enabling the alignment of consecutive scans and sparking a separate stream of research in range sensor-based odometry [7].

Building on these ideas, LiDAR-only odometry gained popularity due to LiDAR's immunity to lighting changes and ability to capture dense 3D data. One key approach uses straightforward scan matching via ICP [7], but such methods can struggle in environments with few unique geometric features. To address this, feature-based systems extract distinctive cues from each LiDAR scan, as demonstrated in LOAM [8], where sharp edges and planar surfaces guide the matching process. While LiDAR-only odometry can produce accurate short-term estimates, it may drift over time if the robot moves rapidly or the environment is mostly repetitive.

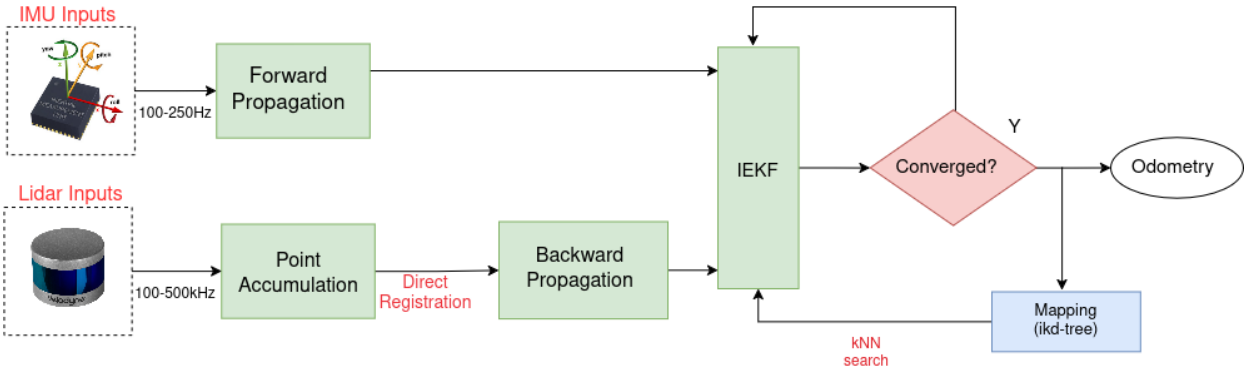


Figure 2.1: Architecture of FAST-LIO2: A tightly coupled LiDAR-Inertial Odometry [13]

To reduce long-term drift, many researchers incorporate IMU (Inertial Measurement Unit) data, resulting in what is called LiDAR–Inertial Odometry (LIO). In a loosely coupled setup, LiDAR and IMU each compute their own pose, and then a higher-level filter (like an Extended Kalman Filter) merges these estimates [9]. This approach is easier to design because the LiDAR odometry module and the IMU integration module work somewhat independently. However, the separate modules might not fully exploit each other's measurements.

By contrast, tightly coupled designs feed raw IMU data directly into the LiDAR optimization. For instance, LIO-SAM employs a factor graph where both LiDAR and IMU preintegration factors are updated in the same framework, allowing more constraints to be used in each calculation [10]. As graph-based LiDAR–inertial odometry solutions grew more complex and demanding, researchers began exploring filter-based methods built around the Kalman filter. One notable example is LINS [11], which uses an iterated Error State Kalman filter (iESKF) to speed up pose estimation; however, it still faces performance bottlenecks when dealing with large numbers of LiDAR points, since computing the Kalman gain for every point can be costly. In response, FAST-LIO [12] introduced a more efficient Kalman gain calculation that significantly lowers this overhead. Later, As shown in Figure 2.1 FAST-LIO2 [13] expanded on that approach by completely removing the feature extraction step and matching raw LiDAR points directly to the map. To accelerate nearest-neighbor queries, it employs a specialized data structure called an ikd-Tree, allowing the system to achieve both high accuracy and even faster run.

Due to its high computational efficiency, accuracy, and real-time capability, FAST-LIO2 is adopted in this thesis as the LiDAR-Inertial Odometry (LIO) front-end. Its design aligns with the system's objective to maintain high-frequency odometry while minimizing drift, providing a solid foundation for downstream scan-to-map correction and graph-based fusion.

2.1.2 SLAM and Loop Closure-Based Localization

Moving beyond pure odometry, SLAM (Simultaneous Localization and Mapping) takes an extra step by adding loop closure mechanisms to maintain global consistency [14]. While LiDAR–Inertial Odometry can reduce local drift, it remains prone to cumulative error over long distances if no global corrections are made. By detecting revisits to previously mapped areas, a full SLAM pipeline like Google Cartographer (for 2D LiDAR) [15] or other graph-based back ends [16] refines the entire trajectory, resulting in notably higher accuracy.Recent SLAM systems like MOLA slam [17] and KISS-ICP[18] have further advanced LiDAR-based localization by improving loop closure detection and trajectory optimization for large-scale environments. However, these systems carry greater computational cost and design complexity, since maintaining a global graph or submap structure calls for advanced data management.

SLAM inherently relies on loop-closure events or prior constraints to limit its global drift; if these loop closures are sparse or absent over a long trajectory, the system can still accumulate significant positioning errors [16, 14]. That limitation underscores the motivation for map-based localization, in which an existing map is used to provide absolute constraints and reduce drift, even in lengthy or repetitive environments where loop closures are not guaranteed.

2.1.3 Localization Using Prior Maps

Estimation of the position of a sensor on a map is essential for navigation systems. While several kinds of map representations are used, depending on the use scenario 3D point cloud maps are among the most popular representations owing to their simplicity and expressiveness.[19]Because constructing a point cloud map is relatively easy with recent precise range sensors and mapping algorithms, point cloud maps are used for a wide range of applications, from indoor service robots to outdoor driving of autonomous vehicles.

Monte Carlo Localization (MCL) is a well-established probabilistic technique for 2D map-based localization [20]. It excels in handling non-linear and non-Gaussian systems, offering robustness to ambiguous observations and multi-modal distributions. However, its application in 3D localization is rare due to the exponential increase in the number of particles required to represent the full 6-DoF state space. This makes real-time use in point cloud-based 3D environments computationally prohibitive

Multiple recent works highlight the potential of map-aided localization for achieving robust and precise vehicle pose estimation.[21] propose building a 2D road-surface reflectivity map from lidar data, then using real-time lidar scan correlation against this map to localize

within a few decimeters in busy urban areas.Similarly,[3] adopt a detailed segmentation approach: each incoming lidar frame is broken down into ground, curb, edge, and surface features,which are then aligned with a prior feature map; the refined estimates are further fused with a MEMS-IMU for centimeter-level accuracy at high frequency.

By contrast,[22] utilize stereo-based "visual point cloud" maps, avoiding LiDAR hardware yet involving dense stereo matching for both the offline (map) and online (frame) steps. Their system then registers these point clouds via normal distribution transform (NDT). While it lowers sensor costs, it adds a heavy stereo workload and remains susceptible to lighting or texture issues. The authors also report that deviations can occur at sharp turns, albeit these are later corrected. Finally, [23] integrate LOAM odometry with a global segment-based prior map (via SegMap) for re-localization when drift accumulates. The system occasionally triggers place recognition in the global map, confirms geometry with RANSAC, then refines the pose using fine-grained ICP.

By contrast, [22] utilize stereo-based "visual point cloud" maps, avoiding LiDAR hardware yet involving dense stereo matching for both the offline (map) and online (frame) steps. Their system then registers these point clouds via normal distribution transform (NDT). While it lowers sensor costs, it adds a heavy stereo workload and remains susceptible to lighting or texture issues.The authors also report that deviations can occur at sharp turns, albeit these are later corrected. Similarly, [1] propose a stereo vision localization method within 3D LiDAR maps by extracting visual features in the Bird's-Eye View (BEV) domain and matching them to the LiDAR map. Finally, [23] integrate LOAM odometry with a global segment-based prior map (via SegMap) for re-localization when drift accumulates. The system occasionally triggers place recognition in the global map, confirms geometry with RANSAC, then refines the pose using fine-grained ICP.

Together, these works illustrate the diversity of map-based localization strategies, but many are constrained by computational inefficiency, dependence on handcrafted or road-specific map features, and limited scalability or adaptability to dynamic and degraded environments. This underscores the need for a more efficient, generalizable, and real-time-capable localization framework.

2.1.4 Point Cloud Registration Techniques

Point cloud registration is a key technique in LiDAR-based localization, used to align a current LiDAR scan with a prior map to estimate the sensor's pose. The classic Iterative Closest Point (ICP) algorithm performs this by minimizing distances between matched point pairs across the two clouds. While ICP is conceptually simple, it suffers from sensitivity to

noise, poor initial alignment, and local minima, making it unreliable for large-scale or noisy environments [7].

To address these limitations, the Normal Distributions Transform (NDT) was proposed by [24] and further elaborated in subsequent research[25]. NDT offers a compact representation of surfaces by converting point clouds into a set of local probability density functions (PDFs), each characterizing a surface section.Instead of relying on discrete point matches, NDT represents the map as a grid of voxels, each containing a Gaussian distribution that models the local geometry. Scan points are registered by maximizing their likelihood under these distributions. This makes NDT more robust to outliers, partial overlaps, and noisy measurements.

Modern implementations such as NDT-OMP further improve runtime performance through multi-threading, enabling real-time scan-to-map matching in large-scale environments [26]. Due to its robustness and efficiency, NDT is widely used in autonomous driving and is adopted in this thesis for scan-matching against a pre-built point cloud map.

2.1.5 Dynamic Object Removal Techniques

In LiDAR-based localization, scan-matching techniques are widely employed to align real-time sensor data with a pre-built static map. However, dynamic objects such as moving vehicles, pedestrians, and cyclists introduce discrepancies between the current point cloud and the static reference. These inconsistencies can significantly degrade localization performance, leading to pose drift, poor convergence, or failure in highly dynamic environments.

To enhance the reliability of scan-matching, dynamic object removal has been introduced as a crucial preprocessing step. Early methods, including Euclidean clustering, geometric filtering, and background subtraction, identified dynamic elements based on heuristics such as size, shape, or temporal consistency [3, 26] . Although computationally efficient, these approaches often struggle in cluttered or semi-structured environments, resulting in high false positive rates and poor generalization. Recent advancements leverage deep learning-based 3D object detection models such as PointPillars[28], SECOND [29], and CenterPoint [27] to semantically identify dynamic agents. By predicting 3D bounding boxes for known classes, these models enable precise removal of dynamic objects, preserving static features critical for accurate localization.

Among these, CenterPoint[27]has emerged as a state-of-the-art solution. It employs a center-based detection paradigm over voxelized LiDAR input and achieves high accuracy on large-scale datasets such as nuScenes[30], making it particularly suitable for real-time dynamic object filtering.

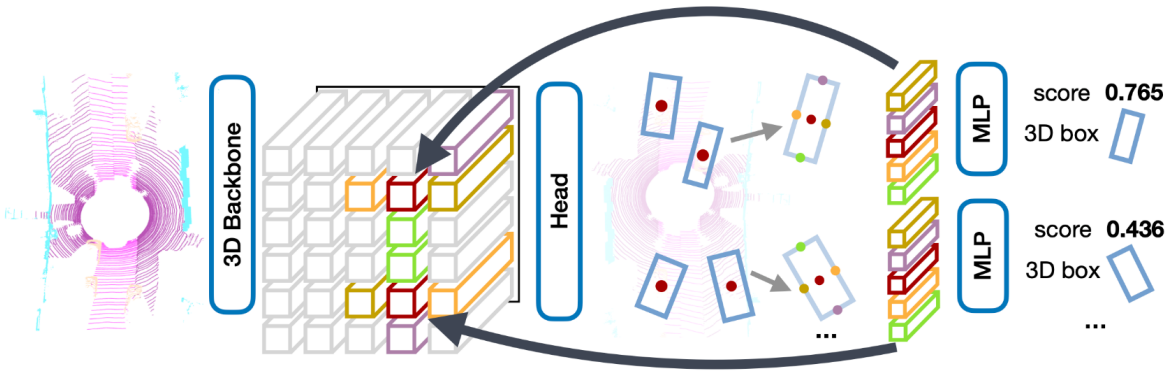


Figure 2.2: Overview of CenterPoint 3d object detection framework. [27]

CenterPoint follows a two-stage architecture[27]. First, the raw LiDAR point cloud is voxelized into a sparse 3D tensor, allowing efficient processing by sparse convolutional neural networks. A backbone network typically based on PointPillars [28] or SECOND[29] extracts spatial features while maintaining the geometric structure of the environment. These features are projected onto a Bird's-Eye View (BEV) representation, preserving spatial relationships and enabling faster 2D convolutional operations. In the second stage, a keypoint-based detection head predicts object centers by generating a center heatmap over the BEV feature map. Following center localization, the model regresses the full set of object attributes, including 3D bounding box dimensions, orientation (yaw angle), velocity, and semantic class label. This decoupled center-to-attribute approach simplifies the detection task and enhances robustness against small, distant, or partially occluded objects, as illustrated in Figure 2.2.Due to its high detection accuracy and real-time inference capability, CenterPoint is adopted in this work as the dynamic object detection module.

2.2 Theoretical Background

2.2.1 Normal Distributions Transform (NDT)

The NDT algorithm begins by subdividing the space occupied by a scan into discrete grid cells—squares in 2D or cubes in 3D. A PDF is calculated for each cell based on the distribution of points within that cell. Each PDF describes how points within the cell are likely generated, assuming the points result from a multivariate normal random process. The probability density function for a cell is defined as:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right) \tag{2.1}$$

where μ and Σ denote the mean vector and covariance matrix of the reference of the

scan surface points within the cell where x lies. The mean and the covariance are computed as

$$\mu = \frac{1}{m}\sum_{k=1}^{m}\mathbf{y}_k \tag{2.2}$$

$$\mathbf{\Sigma} = \frac{1}{m-1}\sum_{k=1}^{m}(\mathbf{y}_k - \mu)(\mathbf{y}_k - \mu)^T \tag{2.3}$$

where $y_k = 1, ..., m \ldots$ are the positions of the reference scan points contained in the cell.

This probabilistic representation gives a piecewise smooth surface approximation with continuous derivatives, capturing the local surface position, orientation, and smoothness effectively.

When applying NDT for scan registration, the objective is to determine the pose of the current scan that maximizes the likelihood that scan points align with the reference map surface.The parameters to be optimised; that is, the rotation and translation of the pose estimate of the current scan, can be encoded in a vector $\vec{p}$. The current scan is represented as a point cloud $X = \{\vec{x}_1, \ldots, \vec{x}_n\}$. Assume that there is a spatial transformation function $T(\vec{p}, \vec{x})$ that moves a point $\vec{x}$ in space by the pose $\vec{p}$.Given some PDF $\vec{p}$ for scan points, the optimal pose maximizes the likelihood function:

$$\Psi = \prod_{k=1}^{n} p(T(\mathbf{p}, \mathbf{x}_k)) \tag{2.4}$$

or equivalently minimizes the negative log-likelihood of Ψ

$$-\log \Psi = -\sum_{k=1}^{n} \log p(T(\mathbf{p}, \mathbf{x}_k)) \tag{2.5}$$

The primary distinction between 2D and 3D NDT algorithms resides in the spatial transformation function and its partial derivatives. In 2D, rotations are represented by a single angle, while in 3D, rotations require more complex representations(rotation matrices or quaternion representations).

2.2.2 Sensor Fusion Approaches

Filtered Based Approaches

Filtering-based sensor fusion recursively estimates a system's state using two main phases: a prediction step (where the state is propagated using a motion model) and an update step (where incoming sensor measurements refine that prediction) [20].This real-time,

sequential estimation framework is widely employed in robotics to handle noisy sensors while maintaining tractable computational load. Nonetheless, large global corrections or loop closures can be challenging to incorporate without more advanced smoothing or factor-graph techniques[14].

1. **Extended Kalman Filter(EKF)** extends the linear Kalman Filter [31] to handle nonlinear systems by linearizing the motion and measurement models around the current estimate. It is widely used due to its computational simplicity, although linearization can degrade accuracy in highly nonlinear settings [20].

2. **Error State Kalman Filter(ESKF)** focuses on estimating the deviation between a nominal trajectory and the true state [32]. A separate routine generates a continuous nominal estimate, while the ESKF maintains small error states for position, orientation, and sensor biases:
$$\delta \mathbf{x}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k^{\text{nominal}}. \tag{2.6}$$
By filtering only $\delta \mathbf{x}_k$, it improves numerical stability in rotation and bias terms, and is particularly effective for high-rate IMU fusion (e.g., in FAST-LIO [12] or LINS [11]). Major loop closures or global corrections remain difficult in a strictly forward-update framework [14].

3. **Unscented Kalman Filter(UKF)** avoids explicit Jacobian linearization by employing a deterministic sampling approach (sigma points) [33]. A set of points $\{\boldsymbol{\chi}_i\}$ is chosen around the mean and propagated through the nonlinear functions $\mathbf{f}$ and $\mathbf{h}$. The resulting transformed set describes the posterior distribution more accurately than a simple first-order expansion. While the UKF handles significant nonlinearities better than the EKF, it can be more computationally demanding [20].

Factor Graph Based Sensor Fusion

Factor graphs are a powerful framework for multi-sensor fusion in robotics, enabling the simultaneous estimation of a robot's trajectory over time. Unlike filtering approaches which operate recursively, factor graphs formulate state estimation as a global optimization problem, making them well-suited for handling nonlinearity, drift correction, and loop closures efficiently [14, 34].

A factor graph is an undirected bipartite graph that separates variable nodes (robot poses) from factor nodes (motion constraints or sensor measurements) [35]. As shown in Figure 2.3, green nodes represent state variables, while red and blue nodes represent measurement and motion constraints, respectively. Each factor connects only to the variables it

influences, forming a sparse, structured representation that is computationally efficient for optimization.

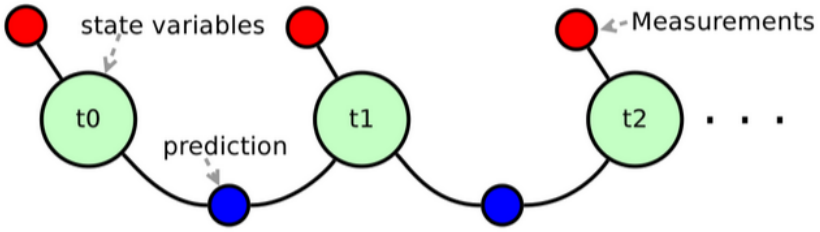


Figure 2.3: Factor graph representation: green nodes are state variables (poses), blue nodes are motion model factors, and red nodes represent sensor measurements.

Mathematically, a factor graph models the joint posterior distribution over all states X, given observations Z, as a product of smaller, local functions:

$$P(X \mid Z) \propto \prod_k \phi_k(X_{\alpha(k)}), \tag{2.7}$$

where ϕ_k is a factor derived from a sensor model or prior, and $X_{\alpha(k)}$ is the subset of state variables relevant to that factor.

In the context of localization, let $X = \{x_0, x_1, \ldots, x_n\}$ be the robot's pose trajectory and $Z = \{z_1, \ldots, z_M\}$ be the set of sensor measurements. The joint posterior can be factorized as:

$$P(X \mid Z) \propto P(x_0) \prod_{k=1}^{n} P(x_k \mid x_{k-1}) \prod_{m=1}^{M} P(z_m \mid X_{\alpha(m)}), \tag{2.8}$$

where:

- $P(x_0)$ is the prior on the initial state,
- $P(x_k \mid x_{k-1})$ is the motion model (e.g., from IMU preintegration),
- $P(z_m \mid X_{\alpha(m)})$ is the sensor measurement model (e.g., LiDAR scan matching).

Assuming Gaussian noise, the Maximum A Posteriori (MAP) estimation reduces to a nonlinear least-squares problem:

$$X^* = \arg\min_X \sum_k \|h_k(X_{\alpha(k)}) - z_k\|^2_{\Sigma_k}, \tag{2.9}$$

where $h_k(\cdot)$ is the measurement function, z_k is the observation, and Σ_k is the associated noise covariance matrix.

Incremental Solvers

For real-time or large-scale applications, solving the entire problem from scratch at each step may be intractable [20]. Incremental smoothing algorithms like iSAM2 [36] maintain a factorization of the problem in a data structure (often a Bayes tree) that can be efficiently updated with new measurements. Rather than re-solving globally each time, iSAM2 carries out local re-linearization and partial variable reordering, providing near-constant-time updates for many practical cases.FurtherMore for Bounded computational cost , a sliding-window strategy are used to optimize only a fixed set of recent states by discarding older variables while preserving accuracy [37]

2.3 Research Gaps

The literature review highlights several gaps and challenges in existing LiDAR-based localization systems, particularly in the context of real-time performance, scalability, and robustness. These gaps inform the design and objectives of the proposed system. Key identified gaps include:

- **Long-Term Drift and Global Consistency:** Existing LIO and SLAM often suffer from long-term drift and loss of global consistency, particularly in GNSS-denied environments. While SLAM approaches attempt to address this via loop closures, their effectiveness is limited in environments with sparse or delayed loop detection.

- **Computational Intractability of MCL:** Monte Carlo Localization (MCL) is robust for 2D localization but computationally impractical for 3D due to the exponential increase in required particles. This limits its application in dense 3D point cloud and 6-DoF pose estimation.

- **Real-Time Performance and Scalability:** map-based localization approaches struggle with real-time performance and scalability in large maps. Standard algorithms like ICP and NDT often fail to maintain real-time performance as map size increases, leading to delayed convergence or misalignment.

- **Generalization:** Existing map-based approaches rely on handcrafted features ,reflectivity profiles, or selected geometric structures from street-level maps limiting their generalization across diverse environments.These systems often fail to generalize across diverse environments and degrade in areas with low geometric richness, unmapped zones, or transitional boundary regions.

- **Dynamic Object Handling:** Existing systems assume static environments and do not systematically address the impact of dynamic objects on scan matching. Dynamic object removal is rarely incorporated in real-time.
- **Environmental Degradation:** Adverse conditions such as fog or sensor noise are often underrepresented in evaluation protocols, leaving a gap in understanding system resilience under realistic field conditions.

Chapter 3

Methodology

This chapter outlines the methodological foundation of the proposed localization system. It describes the main system components and how they are integrated to estimate the robot's pose by combining LiDAR and inertial measurements with a pre-built 3D map. The focus is on the overall process and key implementation choices that support accurate and real-time localization.

3.1 System Overview

The proposed localization system estimates the 6-DoF pose of a mobile robot by combining LiDAR and inertial sensor data with a prior 3D map of the environment. The system is designed to operate in real time and deliver accurate, drift-resilient localization in complex, large-scale, and partially dynamic environments.

The localization pipeline begins with preprocessing of the raw LiDAR point cloud. This includes filtering, downsampling, and transformation to a consistent reference frame. To improve robustness in semi-dynamic scenes, transient objects such as vehicles and pedestrians are detected using a deep learning-based 3D object detection model and removed from the scan prior to registration.

In parallel, inertial measurements from the IMU are fused with deskewed LiDAR data using the FAST-LIO2 framework [13], which provides high-frequency odometry estimates through tightly coupled LiDAR-inertial integration. This module ensures locally consistent motion tracking with minimal latency.

To correct drift and maintain alignment with a global reference, the system performs scan-to-map matching. A local submap is dynamically loaded from the global map based on the robot's current estimated position. Only relevant tiles within a specified radius are retrieved, optimizing memory and computation. The filtered LiDAR scan is then aligned

with this submap using a multithreaded Normal Distributions Transform (NDT-OMP) algorithm [26], which produces a globally referenced pose estimate.

Both odometry and scan-matching results are fused using a fixed-lag smoothing approach based on factor graph optimization. This module maintains a sliding window of recent trajectory states and performs real-time optimization to output a drift-resilient and globally consistent pose estimate.

The overall system architecture and step-by-step algorithm flow are illustrated in Figure 3.1 and Algorithm 1, respectively. The subsequent sections describe each component of the system in detail, including LiDAR preprocessing, local submap loading, odometry estimation, scan matching, graph-based fusion, and dynamic object removal.

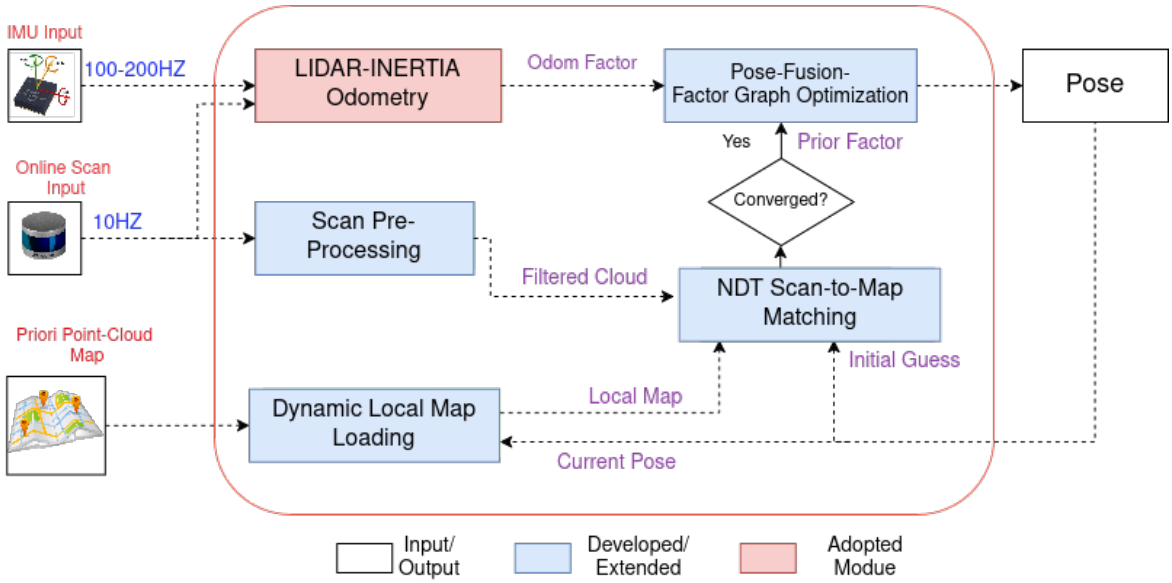


Figure 3.1: System architecture of the proposed map-based localization pipeline. The system fuses high-frequency LiDAR-inertial odometry with global map-based pose corrections using scan-to-map matching and factor graph optimization.

Algorithm 1 LiDAR-Inertial Localization with Prior Map

Input: $\mathcal{P}_t$ (LiDAR Point Cloud), $\mathcal{I}_t$ (IMU Data), $\mathcal{M}$ (Prior Map)
Output: $\mathbf{T}_t$ (Real-time 6-DoF Pose)

while System is running **do**
  Acquire LiDAR point cloud $\mathcal{P}_t$ and IMU data $\mathcal{I}_t$
  Perform scan pre-processing on $\mathcal{P}_t$ to obtain filtered point cloud $\mathcal{P}_t^{filtered}$
  Estimate LiDAR-Inertial Odometry using $\{\mathcal{P}_t^{filtered}, \mathcal{I}_t\}$ to obtain odometry pose $\mathbf{T}_t^{lio}$
  if $||\mathbf{T}_t - \mathbf{T}_{last}^{map}|| > d_{threshold}$ **then**
    Load dynamic local map $\mathcal{M}_t^{local}$ from $\mathcal{M}$ centered at $\mathbf{T}_t$
    Update $\mathbf{T}_{last}^{map} \leftarrow \mathbf{T}_t$
  end if
  Perform scan-to-map matching between $\mathcal{P}_t^{filtered}$ and $\mathcal{M}_t^{local}$ to obtain map-based pose $\mathbf{T}_t^{map}$
  if Matching score $> s_{threshold}$ **then**
    Add prior factor with $\mathbf{T}_t^{map}$ to factor graph
  end if
  Add odometry factor with $\mathbf{T}_t^{lio}$ to factor graph
  Optimize sliding window factor graph to obtain final pose $\mathbf{T}_t$
end while

3.1.1 LiDAR Scan Pre-Processing

Raw LiDAR scans typically produce large point-clouds that vary in density across the range of the sensor. While points near the sensor are densely sampled, those at longer distances become increasingly sparse due to fixed angular resolution. This non-uniform density, combined with measurement noise and dynamic objects, introduces challenges for efficient and robust localization. To address these issues, I apply a structured pre-processing pipeline that filters, reduces, and cleans the incoming data prior to further processing.

As illustrated in Figure 3.2, the following steps are applied to each incoming point cloud:

- **Crop-Box Filtering:** I use a spatial crop-box filter to remove points outside a user-defined region of interest centered around the sensor. This step eliminates distant or irrelevant points, reducing computational load and focusing on the region critical for accurate scan alignment.

- **Statistical Outlier Removal:** To mitigate the effect of random noise, I apply a statistical filter that removes isolated or spurious measurements. For each point, the average distance to its k nearest neighbors is computed. Points exceeding the global mean by a threshold multiple of the standard deviation are discarded as outliers.

- **Voxel-Grid Downsampling:** To reduce data size while preserving geometric structure, I downsample the filtered point cloud using a voxel-grid filter. The space is

partitioned into uniform voxels, and all points within each voxel are replaced by their centroid. This ensures that the point cloud remains computationally tractable without compromising registration fidelity.

- **Dynamic Object Removal:** To improve robustness in dynamic environments, points associated with moving objects are removed. I use a learning-based detector (CenterPoint) to identify dynamic agents and remove their corresponding points using 3D bounding boxes. Further details are provided in Section 3.5.

This cleaned and downsampled point-cloud is then used as input to the scan-to-map matching module, where it is aligned with a local submap to estimate the robot's global pose.

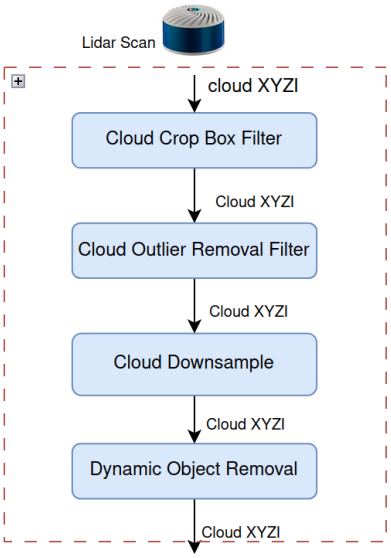


Figure 3.2: Overview of LiDAR scan pre-processing steps. The raw point cloud undergoes spatial cropping, statistical outlier removal, voxel-grid downsampling, and dynamic object removal.

3.1.2 Map Pre-Processing

To support real-time localization in large-scale environments, the global point-cloud map is pre-processed into a 2D grid of tiles each stored as an individual point-cloud file. During this offline division, every point's x–y coordinates are used to assign it to a grid cell of side length s, where s is chosen to balance tile granularity against file count.Each grid cell is saved as an individual point cloud file, and its index is stored as metadata for fast retrieval. During localization, these indexes enable loading only the relevant grids within the robot's local map radius, reducing memory usage and improving processing efficiency.

At runtime, I maintain a dynamic local sub-map around the robot by loading only those tiles whose centers lie within a radius

$$r_{\text{map}} = r_{\text{lidar}} + r_{\text{margin}}, \tag{3.1}$$

where r_{lidar} is the LiDAR's maximum range and r_{margin} is a small buffer to ensure sufficient overlap for robust scan-to-map matching. To avoid unnecessary I/O, I trigger a map update only when the robot's current position $\mathbf{T}_t$ moves farther than a threshold d_{thresh} from the last update location $\mathbf{T}_{\text{last}}$, that is,

$$\|\mathbf{T}_t - \mathbf{T}_{\text{last}}\|_2 > d_{\text{thresh}}. \tag{3.2}$$

When equation 3.2 is satisfied, the system identifies all grid cells whose centers fall within the circle of radius r_{map} about $\mathbf{T}_t$. Those tiles not yet in memory are loaded from the global map, while any previously loaded tiles that now lie outside this radius are evicted. Finally, $\mathbf{T}_{\text{last}}$ is updated to the new robot position. This selective, radius-based loading and unloading scheme keeps the local map size bounded, minimizes memory usage, and ensures that scan-to-map matching always operates on the most relevant portion of the prior map.

3.2 LIDAR Inertia Odometry

FAST-LIO2 serves as the high-frequency LiDAR–Inertial Odometry front-end in our localization pipeline. At each timestep t, the filter ingests IMU angular rate $\omega_m(t)$ and linear acceleration $a_m(t)$ (200 Hz) together with motion-compensated LiDAR point clouds (10 Hz), deskewed using the same IMU measurements. All sensor data are transformed into a common body frame via the known extrinsic calibration

$$T_{\text{imu}\to\text{lidar}} \in SE(3). \tag{3.3}$$

FAST-LIO2 implements a tightly-coupled iterated Kalman filter, directly associating each incoming LiDAR point with the local map using an incremental k-d tree (ikd-Tree), thereby eliminating explicit feature extraction and frame-to-frame scan matching. At each LiDAR update, the filter propagates the state estimate $\mathbf{x}(t)$ via the IMU pre-integration equations, then corrects it using the residuals between measured points and their expected positions in the map. The result is a 6-DOF pose estimate

$$\hat{\mathbf{T}}_{WB}(t) = \begin{bmatrix} \mathbf{R}_{WB}(t) & \mathbf{t}_{WB}(t) \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3), \tag{3.4}$$

which we publish as an odometry factor in our global factor graph. IMU noise densities and bias random-walk coefficients $(\sigma_\omega, \sigma_a, \sigma_{b_\omega}, \sigma_{b_a})$ are set according to the manufacturer's specifications. While FAST-LIO2 exhibits negligible drift over short intervals, residual error on long trajectories is corrected through map matching constraints in the subsequent optimization (Section 3.4).

3.3 Scan-to-Map Matching

The Scan-to-Map Matching module is a critical component of the proposed localization system. Its primary objective is to align the current LiDAR scan with a dynamically loaded local submap and estimate the robot's pose relative to the prior map coordinate frame.Several registration techniques were considered during system design. While Iterative Closest Point (ICP)[7] and the standard Normal Distributions Transform (NDT)[24] are commonly used, they were not selected due to their high computational cost and sensitivity to initial pose estimates.

To overcome these limitations, this work adopts a multithreaded implementation of the Normal Distributions Transform (NDT-OMP) [26], which leverages OpenMP-based parallelization to significantly accelerate computation. NDT-OMP maintains the robustness of NDT while enabling real-time performance through parallel evaluation of gradients and Hessians during scan registration. This makes it particularly well-suited for large-scale and time-constrained localization tasks.

The module begins by voxelizing the local map: each occupied cell i is represented by a Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$ at a resolution r that balances map fidelity against computational load. Given an initial pose estimate $\mathbf{p}_0 = [x_0, y_0, z_0, \phi_0, \theta_0, \psi_0]^T$ (from the previous fused pose), the algorithm refines this estimate by minimizing the negative log-likelihood of the transformed scan points under their enclosing Gaussians (more detail Section 2.2.1):

$$J(\mathbf{p}) = -\sum_{k=1}^{N} \log\big[\mathcal{N}\big(T(\mathbf{p}, \mathbf{x}_k) \mid \mu_{c(k)}, \Sigma_{c(k)}\big)\big].$$

To meet real-time constrain, the computation of gradient(first derivative) and Hessian(second derivative) of the cost function are parallelized over T threads at each Newton iteration (up to $N_{\max}$).The set of LIDAR scan points $\{\mathbf{x}_k\}_{k=1}^{N}$ partitioned into disjoint subsets $\{\mathcal{K}_t\}_{t=1}^{T}$. Each thread t independently accumulates local gradient and Hessian contributions:

$$\mathbf{g}^{(t)} = \sum_{k \in \mathcal{K}_t} \nabla_p \big[-\log p(T(\mathbf{p}, \mathbf{x}_k)) \big], \tag{3.5}$$

$$\mathbf{H}^{(t)} = \sum_{k \in \mathcal{K}_t} \nabla_p^2 \big[-\log p(T(\mathbf{p}, \mathbf{x}_k)) \big]. \tag{3.6}$$

After all threads complete, a final reduction merges these into the global quantities.This lock-free design uses per-thread buffers and OpenMP's guided scheduling to balance uneven voxel densities.

$$\mathbf{g} = \sum_{t=1}^{T} \mathbf{g}^{(t)}, \quad \mathbf{H} = \sum_{t=1}^{T} \mathbf{H}^{(t)}. \tag{3.7}$$

At each iteration the pose update uses a line-search–scaled Newton step:

$$\mathbf{p} \leftarrow \mathbf{p} - \alpha \, \mathbf{H}(\mathbf{p})^{-1} \, \mathbf{g}(\mathbf{p}), \quad \alpha \in [0, 1], \quad \|\Delta \mathbf{p}\| < \varepsilon. \tag{3.8}$$

The key parameters governing the NDT-OMP scan matching process are summarized in Table 3.1. These parameters control the resolution of the voxel grid, convergence criteria, optimization behavior, and parallelization strategy.

Table 3.1: Key parameters of the NDT-OMP Scan-to-Map Matching implementation

Parameter	**Symbol**	**Description**
Voxel resolution	r	Side length of each map voxel
Initial step size	α_0	Starting multiplier for Newton update
Convergence threshold	ε	Minimum norm of pose update for convergence
Maximum iterations	$N_{\max}$	Upper limit on Newton-method iterations
Thread count	T	Number of parallel threads used in optimization

Once scan matching converges, the resulting optimized pose is used as a global correction in the fusion stage, where it is integrated with LiDAR-inertial odometry using a factor graph-based optimization framework.

3.4 Real-Time Factor Graph Optimization

To fuse odometry from FAST-LIO2 with global corrections from scan-to-map matching, a factor graph with fixed-lag smoothing is employed. An Extended Kalman Filter (EKF) was initially considered, using FAST-LIO2 as the prediction model and map-based updates. However, the high-dimensional state of FAST-LIO2, which includes LiDAR–IMU calibration and bias terms, made EKF integration unstable and impractical.

The factor graph approach offers a more modular and scalable solution by modeling each constraint independently and optimizing them jointly. The use of a sliding window limits computation to recent states, ensuring real-time performance.

The following section details the formulation of the factor graph and the types of factors used.

3.4.1 Factor Graph Formulation

We denote the robot trajectory as a sequence of poses:

$$\mathcal{X} = \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k, \quad \mathbf{x}_i \in SE(3) \tag{3.9}$$

where each pose $\mathbf{x}i$ represents the 6-DOF position and orientation of the robot at discrete time step i , and SE(3) denotes the special Euclidean group representing 3D rigid body transformations.

1. Prior Factor A prior factor is used to anchor the initial pose to a known global reference frame. It models the prior distribution over the initial state , expressed as:

$$\phi_{\text{prior}}(\mathbf{x}_0) = \exp\left(-\frac{1}{2}|\mathbf{x}_0 - \bar{\mathbf{x}}_0|^2 \Sigma_0\right) \tag{3.10}$$

In this formulation, $\mathbf{x}_0$ is the estimated initial pose of the robot. The term $\bar{\mathbf{x}}_0$ denotes the prior mean pose from the first LIO output or initialization, and $\mathbf{\Sigma_0}$ is the prior covariance matrix reflecting the uncertainty. This Gaussian prior serves as a fixed anchor to stabilize the trajectory estimation.

2. Odometry (Between) Factors Relative motion from LIO is encoded using between factors, which represent the relative transformation between consecutive poses:

$$\phi_{\text{odom}}(\mathbf{x}i-1, \mathbf{x}i) = \exp\left(-\frac{1}{2}\left|f\text{odom}(\mathbf{x}_{i-1}, \mathbf{x}i) - \bar{\mathbf{z}}i^{\text{odom}}\right|^2 \Sigma\text{odom}\right) \tag{3.11}$$

Here,$\bar{\mathbf{z}}i^{\text{odom}}$ is the relative pose measurement between time steps i and *i+1*, derived from the LIO module. The $f$odom(.) function calculates the predicted relative transformation between the estimated poses $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$, . The $\mathbf{\Sigma}$odom term is the associated covariance matrix, which quantifies the uncertainty in the odometry measurement. These between factors form the sequential backbone of the trajectory by chaining consecutive pose estimates together through relative motion constraints.

3. Map-Matching Priors Absolute pose estimates derived from scan-to-map matching using Normal Distributions Transform (NDT) are incorporated as prior constraints on select poses:

$$\phi_{\text{map}}(\mathbf{x}_j) = \exp\left(-\frac{1}{2}|\mathbf{x}j - \bar{\mathbf{x}}j^{\text{map}}|^2\Sigma\text{map}\right) \tag{3.12}$$

In this expression, $\bar{\mathbf{x}}j^{\text{map}}$ denotes the global pose obtained by aligning the current LiDAR scan with a pre-built prior map using scan matching techniques. The associated covariance matrix $\mathbf{\Sigma}$map expresses the uncertainty of the registration process, accounting for alignment quality and sensor noise. These priors provide absolute spatial constraints within the factor graph, which help to correct accumulated drift from relative odometry and enhance the global consistency of the trajectory estimation.

3.4.2 Maximum A Posteriori (MAP) Estimation

The objective of the sensor fusion system is to compute the most probable robot trajectory given all available measurements, including odometry and map-matching observations. This is formulated as a Maximum A Posteriori (MAP) estimation problem, expressed as the following nonlinear least-squares objective:

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \left(\|\mathbf{x}_0 - \bar{\mathbf{x}}_0\|^2_{\Sigma_0} + \sum_{i=1}^{k} \left\| f_{\text{odom}}(\mathbf{x}_{i-1}, \mathbf{x}_i) - \bar{\mathbf{z}}_i^{\text{odom}} \right\|^2_{\Sigma_{\text{odom}}} + \sum_{j \in \mathcal{M}} \left\| \mathbf{x}_j - \bar{\mathbf{x}}_j^{\text{map}} \right\|^2_{\Sigma_{\text{map}}} \right) \tag{3.13}$$

Each term in the cost function represents a Mahalanobis distance, penalizing deviations from the respective measurements weighted by their uncertainty. Solving this objective yields the optimal trajectory estimate $\mathbf{X}^*$, which fuses relative odometry and global pose observations in a probabilistically consistent manner.

3.4.3 Fixed-Lag Smoothing and Marginalization

In our proposed system, fixed-lag smoothing is used to maintain real-time performance while optimizing the recent trajectory of the robot. Instead of keeping the entire history of poses in the factor graph, we define a sliding time window of length τ, such that only poses within the interval $[t_k - \tau, t_k]$ are retained, where t_k is the current time. This ensures that the graph grows only within a fixed temporal horizon, regardless of how long the system has been running.

Let $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k$ represent the sequence of robot poses over time. At each time step, the latest pose $\mathbf{x}_k$ is added to the factor graph along with new factors derived from LIO

(odometry) and map-matching. Older poses such as $\mathbf{x}_i$ for which $t_i < t_k - \tau$ are marginalized out to limit the graph size.

The process of marginalization involves removing a variable (pose) from the graph while preserving its influence on the remaining variables. Suppose the state vector is partitioned as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_m \\ \mathbf{x}_r \end{bmatrix}, \tag{3.14}$$

where $\mathbf{x}_m$ are the variables to be marginalized (e.g., old poses outside the lag window), and $\mathbf{x}_r$ are the variables retained in the active window. The joint probability distribution is expressed as:

$$P(\mathbf{x}_m, \mathbf{x}_r) = P(\mathbf{x}_m|\mathbf{x}_r)P(\mathbf{x}_r). \tag{3.15}$$

To obtain a reduced representation, we marginalize out $\mathbf{x}_m$:

$$P(\mathbf{x}_r) = \int P(\mathbf{x}_m, \mathbf{x}_r)\, d\mathbf{x}_m. \tag{3.16}$$

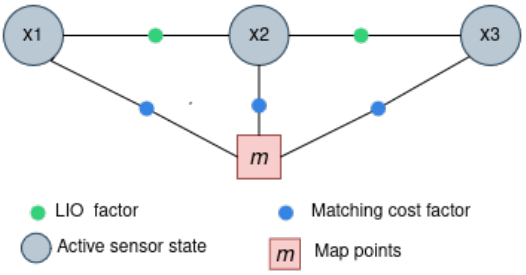
As a result, a new dense factor is introduced between variables in $\mathbf{x}_r$ that were connected to $\mathbf{x}_m$, preserving the influence of the eliminated variables.

The process of fixed-lag smoothing is illustrated in Figure 3.3 using a temporal window of length $n = 4$. Initially (Figure 3.3a), the system contains states $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$, connected by odometry and map-matching factors. As pose $\mathbf{x}_4$ is added at time t_4 (Figure 3.3b), it links to $\mathbf{x}_3$ through a new odometry factor and includes a map constraint. Since all poses $\mathbf{x}_1$ to $\mathbf{x}_4$ lie within the lag window $[t_1, t_4]$, the graph remains fully active and no marginalization occurs.

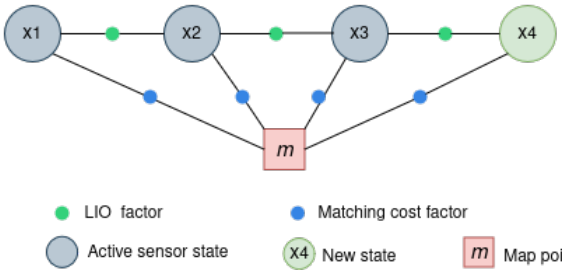
When the system reaches time step t_5, pose $\mathbf{x}_5$ is added, along with its corresponding odometry and map-matching constraints. At this point, pose $\mathbf{x}_1$ falls outside the fixed-lag window $[t_2, t_5]$. As a result, it is marginalized. This process eliminates $\mathbf{x}_1$ from the graph but replaces its influence with a dense prior that connects to the remaining relevant states, particularly $\mathbf{x}_2$ and beyond (Figure 3.3c). The marginalized node is no longer directly optimized, but its effect is preserved through conditional probability in the form of a Bayes net.

At the next step, time t_6, a new pose $\mathbf{x}_6$ is introduced. To maintain the lag window $[t_3, t_6]$, the system marginalizes pose $\mathbf{x}_2$, introducing another dense factor that connects to the active variables. The current active states now consist of $\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_5$, and $\mathbf{x}_6$, all within the sliding lag window (Figure 3.3d).

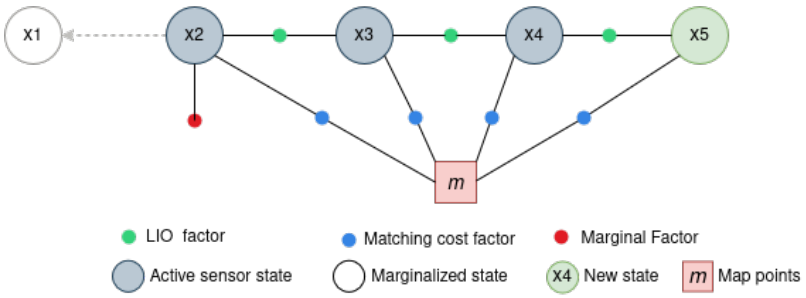
This incremental update mechanism, implemented using the GTSAM library, maintains a bounded graph size by marginalizing old variables into Bayes net conditionals while pre-



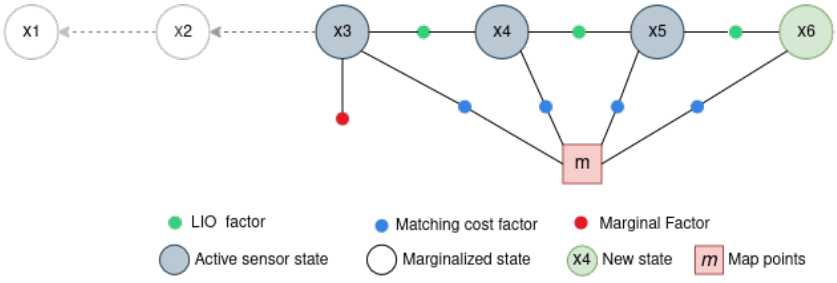
(a) The initial factor graph contains active states $\mathbf{x}_1$ to $\mathbf{x}_3$, linked through LiDAR-inertial odometry (LIO) and map-matching factors.



(b) A new state $\mathbf{x}_4$ is introduced, remaining within the lag window.



(c) As state $\mathbf{x}_5$ is added, the oldest state $\mathbf{x}_1$ is marginalized and its influence is preserved via a prior factor.



(d) At time t_6, $\mathbf{x}_6$ is introduced and $\mathbf{x}_2$ is marginalized, maintaining a fixed-lag window over the most recent four states.

Figure 3.3: Demonstration of fixed-lag smoothing with a lag size of $n = 4$ in a factor graph-based localization system.

serving historical information. Recent poses within the active window are kept in a factor graph and updated via iSAM2, striking a balance between computational efficiency and estimation accuracy for real-time localization.

3.4.4 Outlier Rejection via Mahalanobis Distance

To ensure robust fusion of odometry and scan–map correction constraints, we apply a probabilistic outlier test based on the Mahalanobis distance. Let the correction measurement be the 6-DOF pose $\mathbf{z} = [x, y, z, \phi, \theta, \psi]^\mathsf{T}$ provided by the NDT-based scan matching, with associated covariance $\mathbf{\Sigma}_z$. Given the current filter estimate $\hat{\mathbf{z}}$, we compute the innovation

$$\Delta\mathbf{z} \;=\; \mathbf{z} - \hat{\mathbf{z}}.$$

The squared Mahalanobis distance is then

$$D_M^2 \;=\; \Delta\mathbf{z}^\mathsf{T}\, \mathbf{\Sigma}_z^{-1}\, \Delta\mathbf{z}. \tag{3.17}$$

Under the Gaussian assumption, D_M^2 follows a χ^2 distribution with $d = 6$ degrees of freedom. We reject the correction if

$$D_M^2 \;>\; \chi^2_{d,\,0.95}, \tag{3.18}$$

where $\chi^2_{d,\,0.95}$ is the 95% quantile of the χ^2_d distribution. Otherwise, the measurement is accepted and incorporated into the factor graph. This criterion effectively discards implausible scan-matching corrections while preserving valid global pose updates.

3.5 Dynamic Object Detection and Removal

3.5.1 Dynamic Object Detection

To identify and remove dynamic objects from the LiDAR point cloud, this work utilizes a deep learning-based 3D object detection pipeline based on CenterPoint[27]. The model is developed and trained using the MMDetection3D framework[38], and is deployed for real-time inference using the Autoware-compatible implementation[39].

The detection architecture adopts the two-stage design of CenterPoint, as shown in Figure 2.2. First, the raw LiDAR point cloud is voxelized and processed by a voxel feature encoder and a PointPillars-based sparse convolutional backbone[28], which extracts high-level spatial features. These features are projected into a Bird's-Eye View (BEV) map, enabling efficient object detection. A keypoint-based detection head then identifies object centers and regresses 3D bounding box attributes, including size, orientation, and velocity.

The final output includes class-labeled 3D detections, which are used to support dynamic object removal.

Instead of training from scratch, this work utilizes a pretrained CenterPoint model trained on the nuScenes dataset [30], comprising over 28k LiDAR frames. The model includes five object classes: CAR, TRUCK, BUS, BICYCLE, and PEDESTRIAN. These categories cover the primary dynamic agents present in urban driving scenarios. After training, the model is exported to the ONNX (Open Neural Network Exchange) format, which supports cross-platform deployment and enables hardware-accelerated inference using TensorRT.

3.5.2 Detected Object removal

Following object detection, dynamic elements are removed from the LiDAR point cloud using their predicted 3D bounding boxes. For each detected object, its position, dimensions, and orientation are used to define an oriented 3D cropping volume. These bounding boxes are expanded slightly with a fixed margin to ensure full exclusion of the object's points, accounting for detection uncertainty or sensor noise.

Each bounding box is then used to define a geometric filter in 3D space. The removal process involves identifying all LiDAR points that fall within the transformed bounding box and excluding them from the original scan. This is performed iteratively for all detected dynamic objects in each frame. The result is a filtered point cloud that retains only static environmental features, which are then used for downstream localization tasks.

3.6 Environmental Noise Simulation

To assess the robustness of the proposed localization pipeline under adverse conditions, we simulate the impact of atmospheric fog on LiDAR scans. Fog causes scattering and absorption of laser beams, leading to partial loss of valid returns, the appearance of false near-field echoes, and a reduction in return intensity. Our simulation adopts the physically grounded model proposed by Teufel et al. [40], applied only to live LiDAR scans. The degradation severity is controlled by the *meteorological visibility* V (in meters), defined as the maximum distance at which an object can be reliably seen under foggy conditions.

The fog model introduces three major effects, each governed by simple visibility-dependent equations: *(i)* deletion of real points, *(ii)* modification into near-sensor backscatter points, and *(iii)* intensity attenuation.

Point Deletion. Real LiDAR returns are probabilistically removed based on their distance d and visibility V, using:

$$p_{\text{delete}}(d) = \frac{a \cdot e^{bV}}{a \cdot e^{bV} + 1}$$

Here, $a = -0.70$ and $b = -0.024$ are empirically derived constants that define the effect of fog density on the likelihood of return loss. Deletion probability increases with distance and with denser fog (lower V).

Backscatter Modification. Points that are not deleted may be converted into *false returns*, simulating backscatter from fog droplets. The probability of modification increases with distance:

$$p_{\text{modify}}(d) = 1 - e^{-d \cdot \epsilon}, \quad \epsilon = 0.23 \cdot e^{-0.0082V}$$

The new range is drawn from an exponential distribution:

$$f(x) = \frac{1}{\lambda} e^{-x/\lambda}, \quad \lambda = -0.006 \cdot V + 2.31$$

This causes the point to be radially pulled closer to the sensor, mimicking typical fog backscatter effects.

Intensity Attenuation. Return intensity is reduced due to scattering loss along the beam path:

$$I(d) = I_0 \cdot e^{-\gamma d}, \quad \gamma = 0.02$$

Points with final intensity below a threshold (e.g., 1000 for 16-bit returns) are discarded. Simulated backscatter points are assigned random intensities in the range of 0–32% of the maximum value, consistent with real sensor behavior under fog.

This fog simulation enables a physically interpretable, parameterized degradation model suitable for evaluating localization performance under various visibility conditions.

Chapter 4

Experimental Setup and Result

This chapter presents the experimental framework and evaluation of the proposed LiDAR-inertial localization system. It includes the hardware and software configurations, dataset preparation, and a detailed performance analysis under both standard and challenging conditions. The evaluation focuses on accuracy, robustness, and real-time performance across different environments.

4.1 Experimental Environment

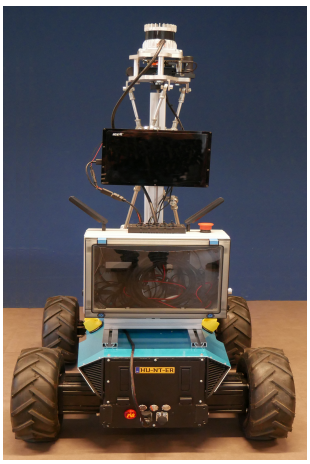
4.1.1 Hardware and Software Setup

All experiments were conducted on a development laptop equipped with an AMD Ryzen™ 7 processor, 32 GB of DDR5 RAM, and NVIDIA GPU for accelerated computation , as well as on a Hunter mobile robotic platform for on-campus data collection.The robot is equipped with an Ouster OS1-128 LiDAR sensor for 3D environmental perception, an onboard Inertial Measurement Unit (IMU) for motion estimation, and a camera for visual reference (Figure 4.1).

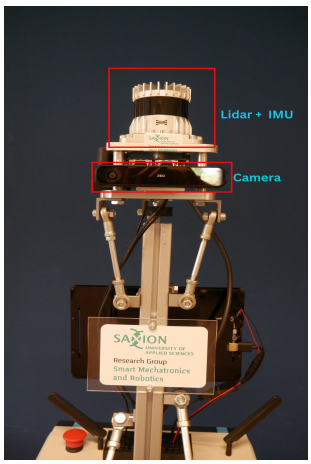
Our approach is implemented primarily in C++ on top of the ROS 2 Humble middleware. We rely on:

- **PCL (Point Cloud Library):** for point cloud data structures and filtering.
- **Eigen:** for linear algebra operations (transformations, matrix computations).

Additional scripts in Python were employed for plotting and minor data manipulations.



(a) Hunter mobile robot platform



(b) Sensor configuration (LiDAR, IMU, Camera)

Figure 4.1: Hardware setup used for real-world data collection and testing: (a) Hunter robotic platform; (b) onboard sensors.

4.1.2 Datasets Collection and Map Preparation

We evaluate the proposed localization system across a combination of public benchmark datasets and custom-recorded sequences. These datasets include suburban, campus, and indoor environments with varying levels of dynamic activity, structural repetition, and trajectory complexity. Table 4.1 summarizes the key characteristics of each sequence, including the environment type and total trajectory length.

Table 4.1: Dataset Overview and Environmental Characteristics

Dataset (Seq ID)	**Environment Description**	**Trajectory**
Saxion Seq-01	Outdoor campus area with moderate pedestrian activity and smooth motion.	∼0.9 km/12 min
Saxion Seq-02	Outdoor campus area with dynamic objects (pedestrians, parked cars), sharp turns, low loop closure opportunities and narrow passages.	∼1.15 km/16 min
Saxion Seq-03	Long sub-urban path with dynamic objects (pedestrians, parked cars), mixed building heights and repetitive environment (apartment blocks).	∼1.8 km/27 min
Saxion Seq-04	Indoor area characterized by long glass-walled corridors and structurally repetitive office layouts.	∼0.43 km/11 min
KITTI Seq-05	Residential environment with urban roads, sidewalks, parked vehicles, repeating patterns, tree, moderate traffic and fast motion.	∼2.2 km/5 min
MulRan KAIST-03	campus area with moderate traffic and multiple distinguishable structures	∼6.2 km/14 min

Benchmark Dataset We used sequences 05 from the KITTI odometry benchmark[41] and sequences KAiST-03 from the MulRan dataset[42].The official ground-truth poses were used to transform and aggregate each LiDAR frame into a global coordinate system for generate 3D priori map.

Custom Saxion Dataset As no direct ground truth was available for this dataset, we employed Fast-LIO2 for initial SLAM-based mapping, followed by offline global optimization using Hierarchical LiDAR Bundle Adjustment (HBA) [43] to ensure global consistency across repeated traversals.

In both cases, the final global point cloud is subdivided into manageable tiles for efficient local map loading during localization. Each tile is stored as a separate file or database entry keyed by tile coordinates, allowing the system to load only relevant tiles based on the vehicle's current estimated position.

4.2 Experimental Evaluation

4.2.1 Evaluation Metrics and Methodology

To assess the localization performance, we use a combination of translational and rotational error metrics, with a focus on Absolute Pose Error (APE). APE quantifies the Euclidean distance between the estimated trajectory and the reference trajectory after applying full SE(3) alignment using the Umeyama method. This alignment ensures fair comparison by removing global offsets in rotation, translation, and scale.

All trajectory evaluations were conducted using the EVO evaluation toolkit [44], which provides statistical metrics including root mean square error (RMSE), mean, median, and standard deviation. For each sequence, we align the estimated trajectory to the ground truth and report the APE over time, along with summary statistics.
In addition to APE, we assess the system's real-time performance by measuring:

- Per-frame execution time across key modules (preprocessing, scan matching, optimization),
- Registration convergence rate and iteration count,
- Localization success rate under environmental degradation (e.g., fog, sparse features).

The evaluation includes comparisons against two categories of : component baselines, which consist of the individual modules integrated into our system (Fast-LIO2[13] and scan-map-matching scan matching), and external map-based localization methods.

4.2.2 Localization Performance

Comparison with baseline methods

Quantitative results for the Saxion sequences are presented in Tables 4.2, 4.3, and 4.4 , summarizing translational and rotational error statistics (RMSE, mean, and maximum) after SE(3) alignment. Each sequence represents a different environmental condition and trajectory complexity. The results show that the proposed fusion-based system consistently outperforms both baseline methods across all sequences.

In Sequence 1(see Table 4.2), the proposed method achieves centimeter-level accuracy, with translation RMSE of 0.067 m, compared to 5.47 m in Fast-LIO2. The rotational RMSE is also reduced from 2.072° to 0.583°, indicating significant drift correction not only in position but also in orientation.

Table 4.2: Translation (APE) and Rotation Error statistics for Saxion **Sequence 1**

Method	Metric	Max	Mean	Median	Min	RMSE	Std Dev
Proposed (Fusion)	APE (m)	0.395	**0.052**	**0.041**	**0.003**	**0.067**	**0.042**
	Rot. (deg)	**4.715**	**0.321**	**0.169**	0.030	**0.583**	**0.486**
NDT Map Matching	APE (m)	**0.363**	0.074	0.067	0.012	0.084	**0.041**
	Rot. (deg)	5.832	0.503	0.299	**0.012**	0.813	0.639
Fast-LIO2	APE (m)	8.94	4.456	4.933	0.322	5.47	1.645
	Rot. (deg)	6.213	1.631	1.304	0.419	2.072	1.278

Note: Bold values indicate the best performance across each metric.

In more complex scenarios (Saxion Sequences 2 and 3), which include dynamic agents, longer sequence, and sharp turns, the proposed fusion method achieves decimeter-level accuracy(see Figure 4.3), significantly reducing translational RMSE compared to Fast-LIO2 (from 3.8 m to 0.11–0.13 m) and improving rotational RMSE by up to 1.6°. While the proposed method exhibits a slightly increase its error in some cases, it consistently achieves lower RMSE, indicating better average accuracy and more stable performance across the entire trajectory.

Across all sequences, the APE translation standard deviation remains within a narrow range of 0.042–0.093 meters and APE rotation 0.4 - 0.74 Degree, indicating that the proposed system achieves both low error and high temporal consistency.

Trajectory plots Figures 4.2, A.1, and A.2 illustrate the alignment of estimated poses with reference trajectory. Zoomed-in regions further emphasize the proposed method's ability to track the trajectory accurately, even in complex zones and long trajectory.

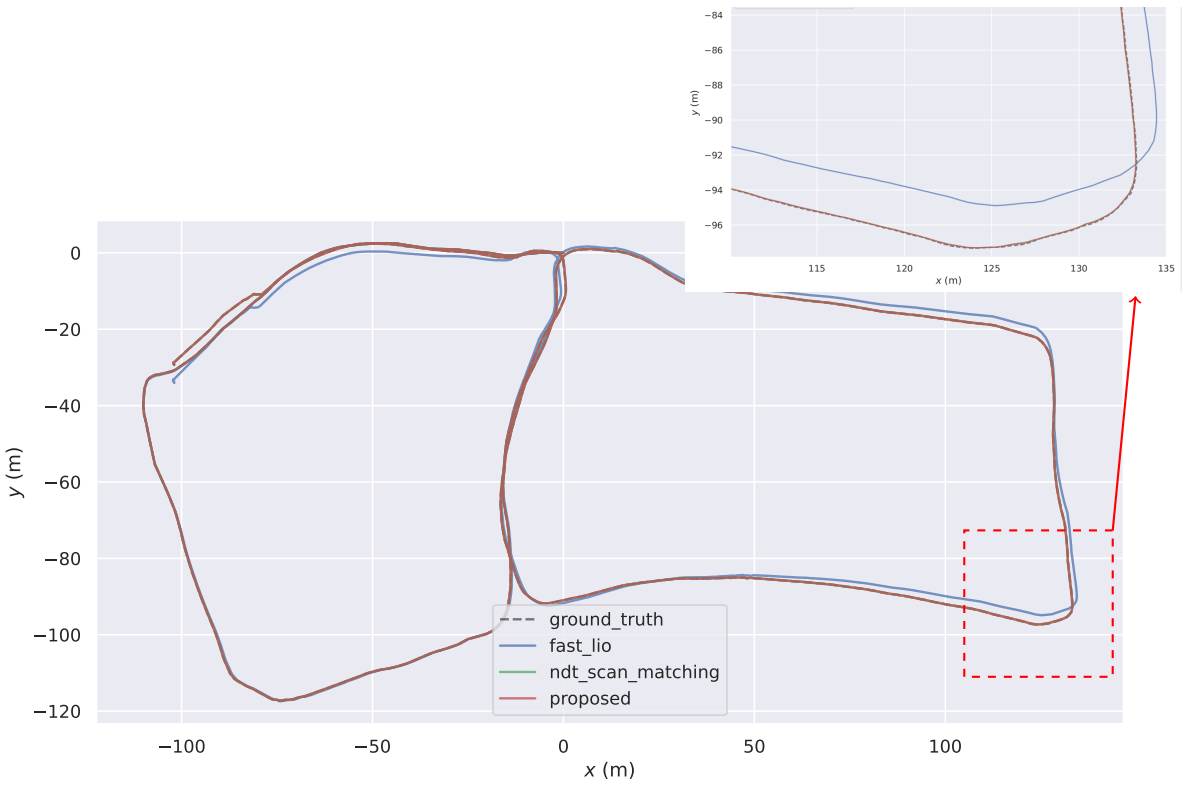


Figure 4.2: **Saxion Sequence 01 – Trajectory Alignment.** The figure shows the estimated trajectories from FAST-LIO2, low frequency NDT map matching , and the proposed method, overlaid against ground truth. The red dashed box indicates the zoomed region.

Table 4.3: Translation (APE) and Rotation Error statistics for Saxion **Sequence 2**

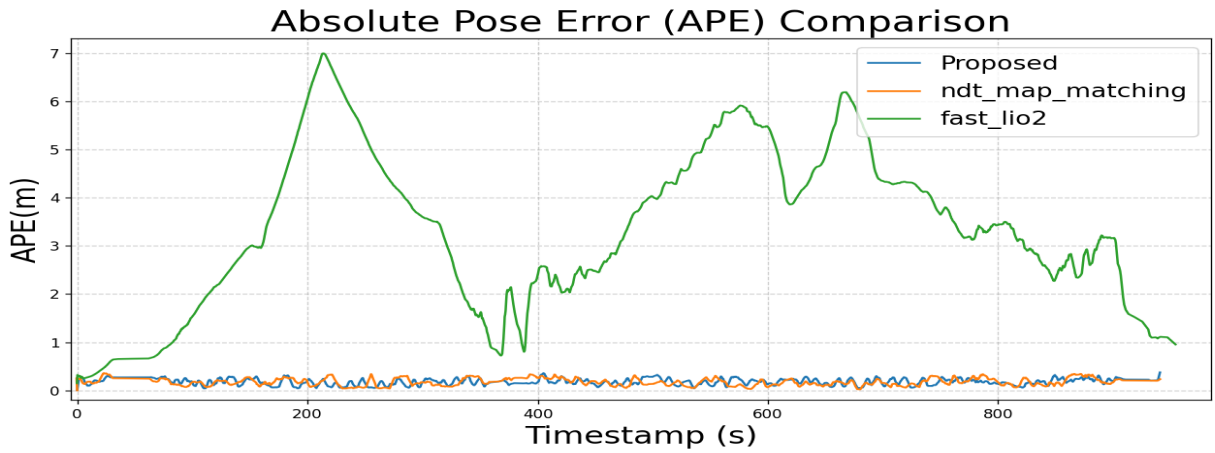
Method	Metric	Max	Mean	Median	Min	RMSE	Std Dev
Proposed (Fusion)	APE (m)	0.440	**0.090**	**0.075**	**0.005**	**0.107**	**0.058**
	Rot. (deg)	4.356	**0.782**	**0.539**	**0.051**	**1.046**	**0.695**
NDT Map Matching	APE (m)	**0.245**	0.102	0.094	0.009	0.118	0.059
	Rot. (deg)	**2.639**	1.195	1.127	0.103	1.393	0.715
Fast-LIO2	APE (m)	6.9941	3.5576	3.44	0.070	3.88	3.8813
	Rot. (deg)	5.494	2.250	1.568	0.202	2.680	1.456

Note: Bold values indicate the best performance across each metric.

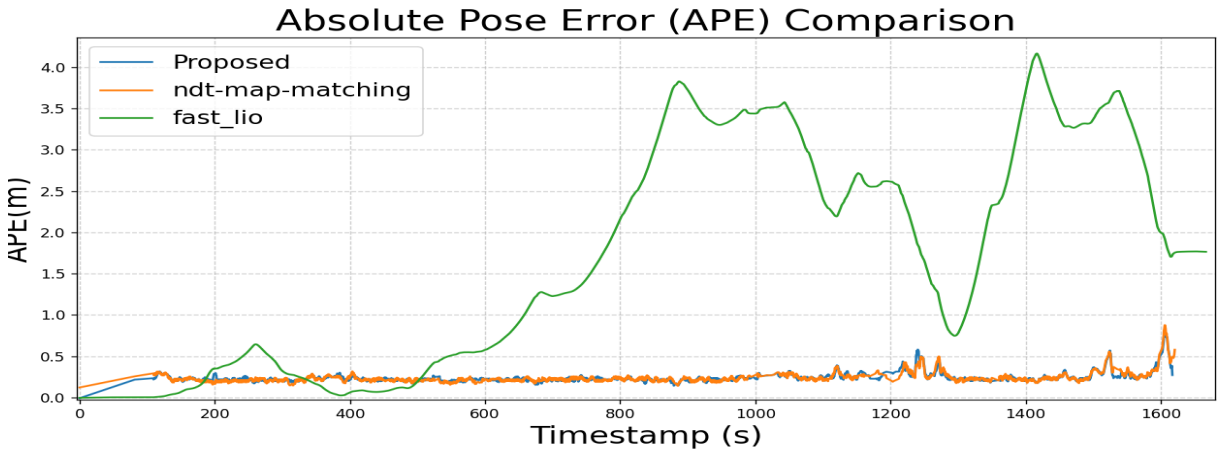
Table 4.4: Translation (APE) and Rotation Error statistics for Saxion **Sequence 3**

Method	Metric	Max	Mean	Median	Min	RMSE	Std Dev
Proposed (Fusion)	APE (m)	0.821	**0.097**	**0.072**	**0.004**	0.135	0.094
	Rot. (deg)	**2.016**	**0.725**	**0.472**	**0.076**	**1.039**	**0.743**
NDT Map Matching	APE (m)	**0.297**	0.113	0.100	0.025	**0.126**	**0.056**
	Rot. (deg)	3.236	1.373	1.286	0.131	1.652	0.919
Fast-LIO2	APE (m)	4.075	1.468	1.254	0.028	2.162	1.145
	Rot. (deg)	2.750	0.955	0.953	0.069	1.088	0.522

Note: Bold values indicate the best performance across each metric.



(a) APE Saxion Sequence 2



(b) APE Saxion Sequence 3

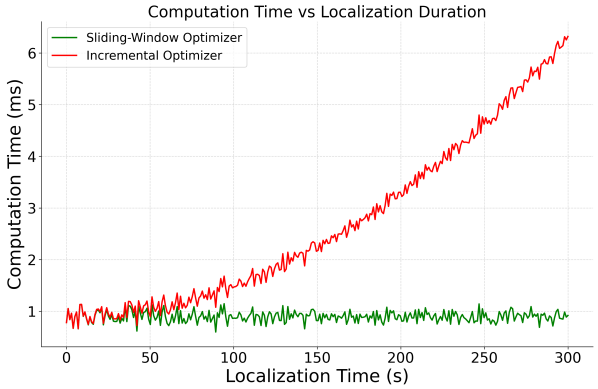
Figure 4.3: Absolute Pose Error Comparison of (a) Saxion Sequence 2 and (b) Saxion Sequence 3 .

Real-time Performance

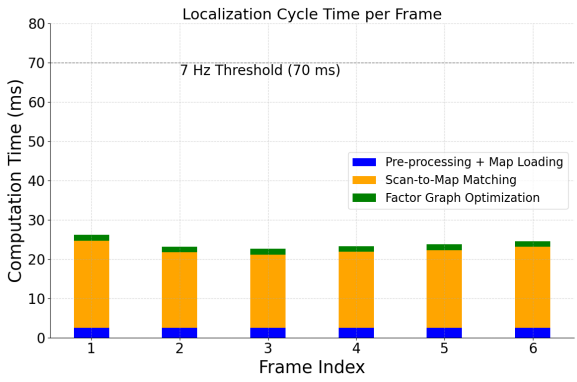
We evaluate scan-to-map matching by varying the radius of the local sub map and compare multi-threaded NDT (NDT-OMP) , standard NDT and classic ICP (Figure 4.5).As shown in the Table 4.5 While both NDT and ICP maintain convergence on small radii ($>$ 100 m), their per-scan runtime is exceed 100 ms and increases as map grows.In contrast, NDT-OMP maintains sub-20 ms mean latency with over 95 % convergence up to 200 m, and still achieves under 25 ms runtimes and 90 % convergence at 350 m (with occasional failures). This demonstrates that NDT-OMP's parallelization, coupled with an adaptive local-map radius based on the robot's pose, significantly improves scan-matching efficiency without sacrificing robustness.

Table 4.5: Scan-Matching Performance vs. Local Map Radius

Map Radius(m)	Method	Mean Time(ms)	Converged(%)	Remarks
100	NDT	90	90	
	ICP	>100	90	
	NDT-OMP(8 thread)	15	99	
200	NDT	>100	<50	failures
	ICP	>100	<50	failures
	NDT-OMP(8 thread)	20	97	
350	NDT	>100	<50	failures
	ICP	>100	<50	failures
	NDT-OMP(8 thread)	25	90	some failures



(a) Computation time trend of sliding-window vs full-batch optimizer.



(b) Computation time breakdown per frame (stacked components).

Figure 4.4: Computation characteristics of the proposed localization system: (a) Sliding-window optimization scalability and (b) real-time frame-wise timing breakdown.

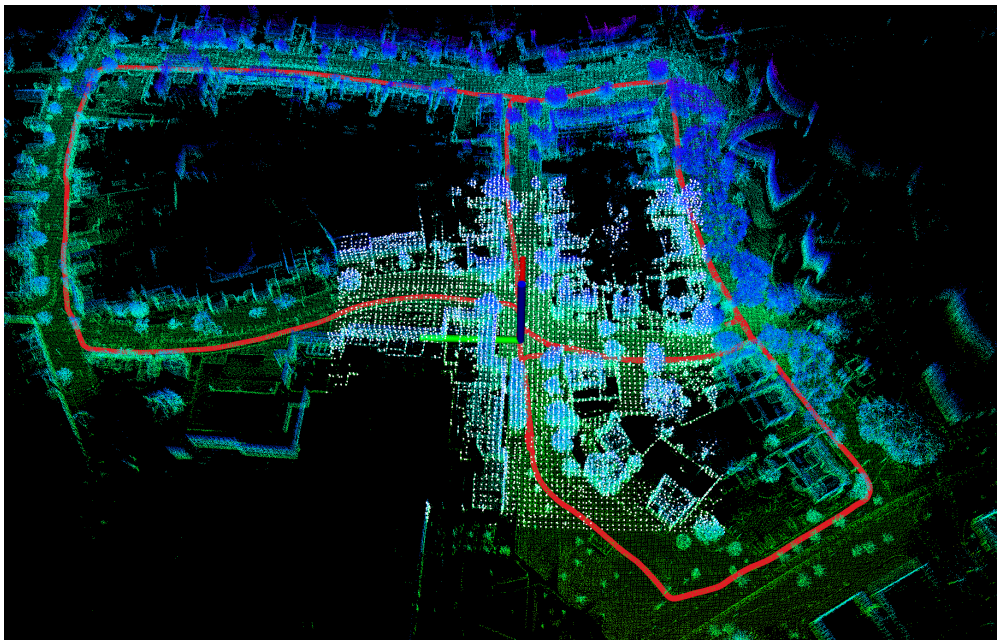


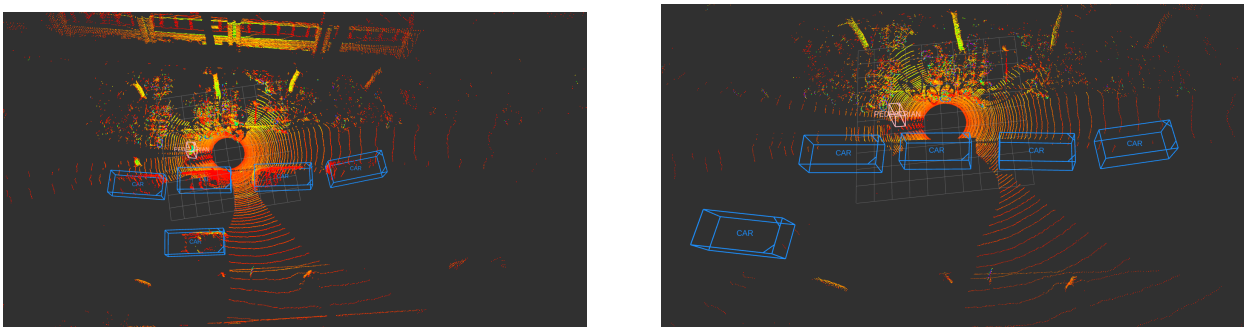
Figure 4.5: Top-down view of the localization trajectory overlaid on the pre-built 3D map. The red line indicates the estimated trajectory, while the white rectangle structures represent dynamically loaded local submaps within a fixed radius during runtime. This demonstrates how local map tiling supports scalable scan-to-map registration.

Considering a high-frequency LiDAR-inertial odometry (LIO) stream and periodic map-matching updates, we compare a sliding-window factor-graph optimizer against a full-batch solver. By restricting the graph to the most recent 5–50 seconds of data,as shown in Figure 4.4a the sliding-window approach maintains nearly constant solve times under 1 ms per update regardless of localization duration. In contrast, the full-batch solver's complexity grows superlinearly as more keyframes accumulate, eventually fail in real-time as the numer of nodes increase.

Figure 4.4b further breaks down the per-frame latency of the proposed system. With point-cloud pre-processing and map loading taking 2 ms, scan-to-map matching 20 ms, and factor-graph optimization 1 ms, the total remains below 23 ms. This comfortably satisfies the real-time 10 Hz requirement (100 ms), achieving 43 Hz processing frequency.

Impact of Dynamic Object Removal on Registration Performance

In this evaluation, we assess the impact of incorporating dynamic object removal into the localization pipeline.Example of detected 3D bounding boxes and the corresponding retained point cloud after dynamic object removal are illustrated in Figure 4.6.The objective is to analyze how removing transient objects from LiDAR scans influences registration accuracy, localization robustness, and computational efficiency. Key performance indicators include registration convergence rate, average iteration count and execution time are evaluated.



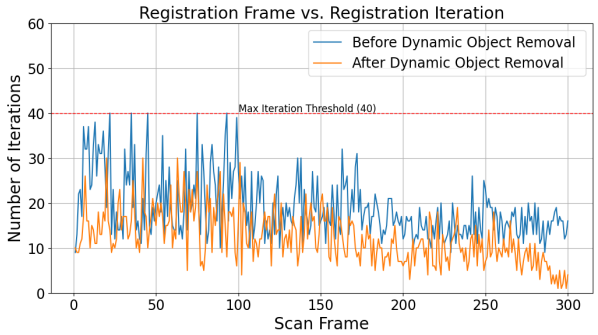
(a) 3D bounding box of detected dynamic objects

(b) Point-cloud retained after removing points with 3d bounding box.

Figure 4.6: Detecting and removing dynamic objects from 3d LIDAR point-cloud.

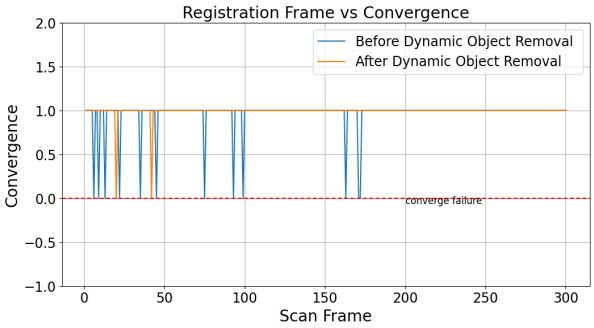
Table 4.6: Comparison of Point-cloud Registration Metrics Before and After Dynamic Object Removal

Metric	Without Removal	With Removal	Improvement
Avg. Registration Iterations	19	12	↓ 7
Avg. Execution Time (ms)	14.41	9.73	↓ 4.68
Registration Failures	7	2	↓ 5



(a) Registration iterations required per scan frame.

(b) Registration Execution time per frame.



(c) Convergence status (1 = success, 0 = failure) across frames.

Figure 4.7: **Registration performance before and after dynamic object removal.** Each subplot shows the impact of removing dynamic objects on and iteration count (a) , execution time (b), and convergence (c). Across 300 frames, the system becomes more stable, efficient, and reliable after filtering out dynamic elements from the scan.

The point-cloud registration performance was evaluated across 300 scan frames , selected based on a high number of detected dynamic objects. As summarized in Table 4.6, filtering dynamic objects results reduction in average registration iterations (from 19 to 12) and execution time (from 14.41 ms to 9.73 ms). Additionally, the number of convergence failures dropped from 7 to 2. The performance plots in Figure 4.7 further illustrate this trend. Dynamic object removal improves stability and efficiency by eliminating noisy inputs that otherwise increase iteration count, processing time, or lead to failed alignments.

Table 4.7 presents the computational overhead introduced by integrating dynamic object detection and filtering into the localization pipeline. While this module enhances registration robustness and accuracy, it incurs additional processing time. Specifically, the 3D object detection step (using ONNX inference) adds approximately 28 ms, and the point cloud filtering (using a CropBox filter) adds another 3 ms per frame. Although the registration and optimization stage benefits from a reduction in processing time from 15 ms to 10 ms the overall per-frame runtime increases from 15 ms (without removal) to 41 ms (with removal), resulting in a net overhead of 26 ms. Despite this increase, the system remains within acceptable real-time performance bounds for medium-speed mobile robotics applications.

Table 4.7: Per-frame runtime analysis with and without dynamic object removal.

Component	Without Removal (ms)	With Removal (ms)	Overhead(ms)
Object Detection	—	28.0	+28.0
Point Cloud Filtering	—	3.0	+3.0
Registration and other	15.0	10.0	↓ 5
Total	15.0	41.0	+26.0

Localization on Map Side Feature Sparse Environment

Map-side feature-sparse environments refer to areas where the current LiDAR observations contain few or no correspondences in the prebuilt map. This situation typically arises at the boundaries of a mapped area, in unmapped zones, or at transitions between separately built or merged submaps. In such regions, the prior map lacks persistent geometric features required for reliable scan-to-map registration, resulting in degraded or failed localization if scan matching is used in isolation.

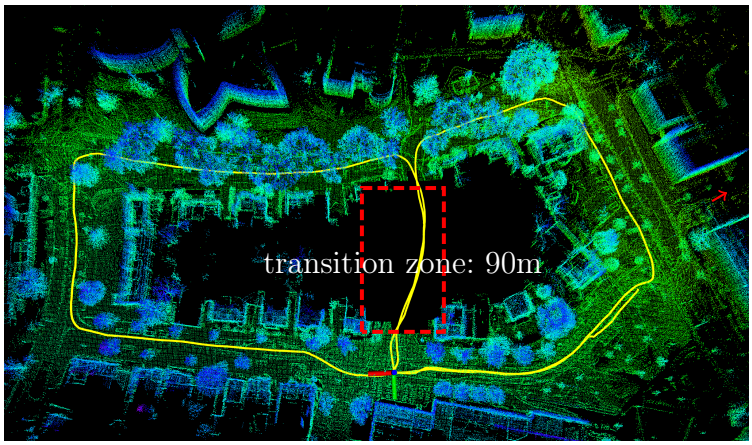


Figure 4.8: **Localization in transition zone.** The robot enters a transition region (highlighted in red) where LiDAR scan matching fails due to the absence of stable features in the prebuilt map

In this scenario, the robot traverses an approximately 90-meter-long transition corridor connecting two main streets. This zone, highlighted in red in Figure 4.8, contains both partially mapped and unmapped segments. The traversal was performed twice to assess consistency. As shown in Figure 4.11, the NDT-based map matching exhibited two prominent error spikes(each exceeding 2 m) corresponding precisely to the robot's entry into the unmapped region during both passes. These spikes highlight NDT's reliance on strong map overlap and its vulnerability in feature-deprived zones.

In contrast, the proposed fusion pipeline maintained stable localization throughout both traversals, as shown in Figure 4.11. This stability is attributed to the FAST-LIO front-end, which provides high-frequency odometry estimates unaffected by the absence of map

data. By combining this with map-based corrections via factor graph optimization, the system successfully bridges the gap between locally consistent and globally drift-resilient localization.

Table 4.8 summarizes the APE statistics. The proposed method limited the maximum APE to 0.455 m, compared to 1.278 m with NDT. The RMSE was also significantly lower—0.095 m versus 0.295 m. These results demonstrate the robustness of the fusion approach under degraded mapping conditions.

For comparison, under fully mapped conditions (Saxion Sequence 1, Table 4.2), the same system achieved a maximum APE of 0.395 m and an RMSE of 0.067 m. This indicates that the proposed method maintains localization accuracy even when operating in feature-sparse or partially unmapped environments.

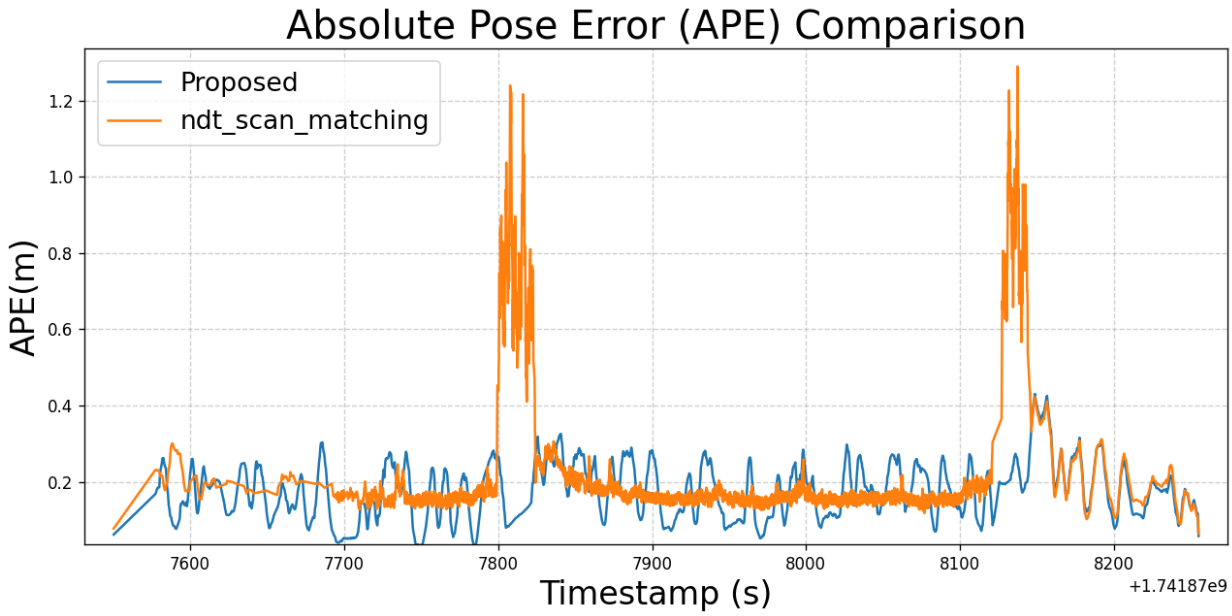


Figure 4.9: **Error comparison in transition zone.** The APE for NDT map matching shows two large spikes (a), while the proposed fusion approach remains stable (b), benefiting from the FAST-LIO front-end.

Table 4.8: Absolute Pose Error (APE) Statistics in Transition Corridor

Method	Max APE (m)	Mean APE (m)	RMSE (m)	Std Dev (m)
Proposed Fusion	**0.455**	**0.078**	**0.095**	**0.054**
NDT Map Matching	1.278	0.131	0.295	0.264
FAST-LIO2	7.59	1.456	2.197	1.645

Note: Bold values indicate the best performance across each metric.

Localization Under Fog Degradation with a Recent Map

We tested the localization pipeline on the **Saxion Sequence 1** using a one-month-old prebuilt map and simulated fog at three visibility levels: *Mild (100 m)*, *Moderate (60 m)*,

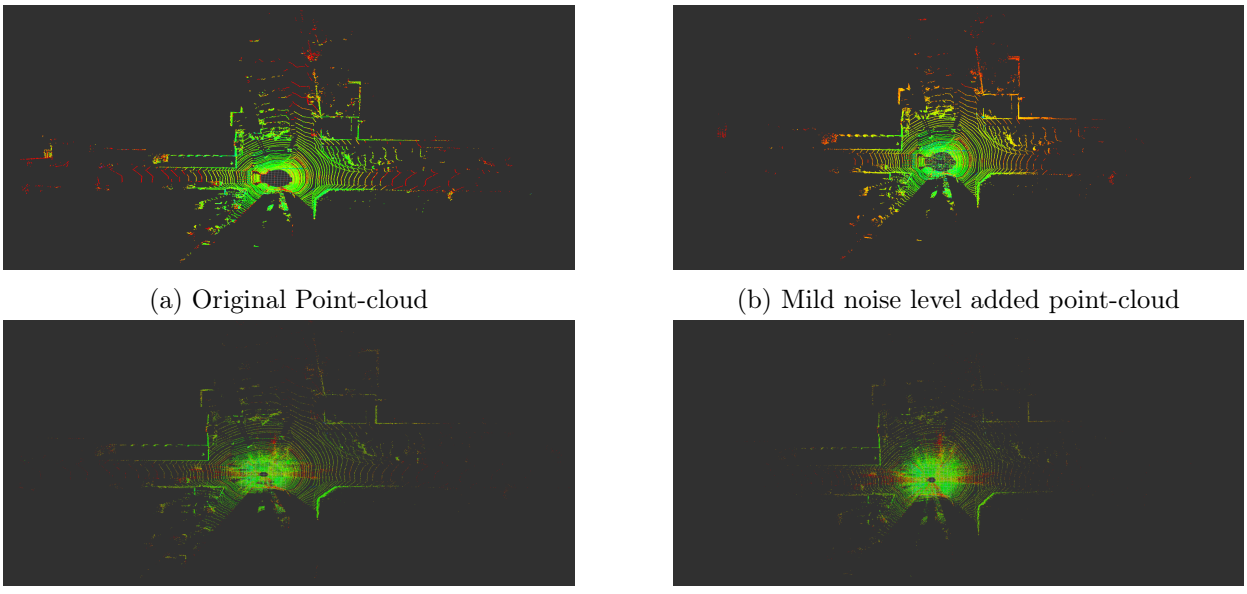
and *Severe (30 m)* see Table 4.9. Figure 4.10 illustrates fog-induced degradation effects such as point dropouts, false near-field returns, and spatial clutter. The resulting Absolute Pose Error (APE) was compared against a baseline under clear conditions.

Under Mild and Moderate fog Table 4.10, the proposed fusion and NDT map matching module methods show limited degradation compared to the baseline (Table 4.2).APE increses to 0.10-0.28 m , with occasional peaks over 1.4 m, but RMSE remains stable, confirms the robustness of NDT under moderate visibility loses.In contrast,FAST-LIO2 drifts significantly, with higher mean RMSE values, indicating its sensitivity to fog when operating without map-based correction.

Under Severe fog (visibility 30 m), all methods show significant degradation. The NDT map matching module becomes unstable and fails to converge consistently due to severe point dropout and noise. Fast-LIO also exhibits large drift, and the combined effect causes the fusion pipeline to fail repeatedly, unable to maintain consistent localization beyond a short distance.

Table 4.9: Fog severity levels based on visibility range

Fog Level	**Visibility V (m)**
Mild	100
Moderate	60
Severe	30



(a) Original Point-cloud

(b) Mild noise level added point-cloud

(c) Moderate noise level added point-cloud

(d) Severe noise level added point-cloud

Figure 4.10: Point-cloud snapshots under increasing composite geometric noise levels: (a) Original scan; (b) Mild noise; (c) Moderate noise; (d) Severe noise.

Table 4.10: Translation error (APE) statistics under Severe and Moderate fog visibility

Condition	Method	Max	Mean	Min	RMSE	Std Dev
Mild	Proposed	**0.3682**	**0.1238**	**0.0250**	**0.1356**	**0.0583**
	NDT Map Matching	0.4089	0.1474	0.0164	0.1585	0.0581
	Fast-LIO2	12.0253	4.2643	0.581	5.7521	3.2145
Moderate	Proposed	**0.3792**	**0.1326**	**0.0092**	**0.1452**	**0.0593**
	NDT Map Matching	1.3442	0.1712	0.0193	0.2178	0.1346
	Fast-LIO2	17.7823	6.2.3521	0.0610	12.9345	5.3614

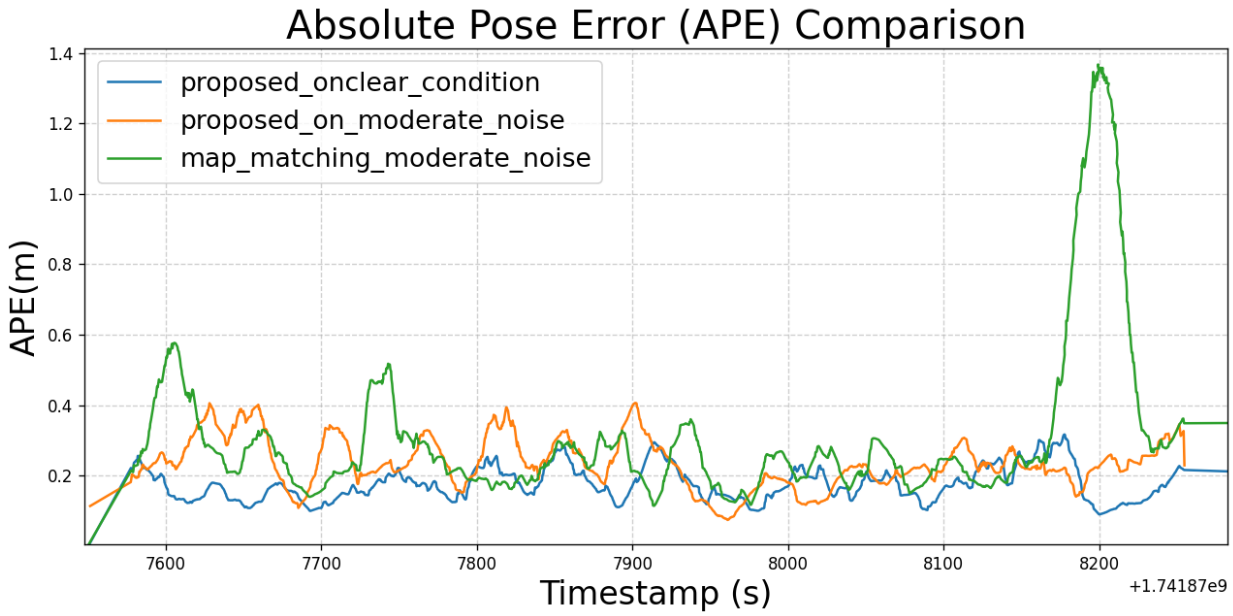


Figure 4.11: The APE comparison at different noise level

Benchmarking on Public Dataset

To evaluate the effectiveness of the proposed method, we compare its mean translation and rotation errors against recent map-based localization techniques using the KITTI Sequence 05 dataset. The comparison includes methods based on stereo vision and prior maps [1], visual point cloud priors [45], and LiDAR-only map alignment [23]. Table 4.11 summarizes the results, where our approach demonstrates superior accuracy and consistency, with lower mean error and deviation across both translation and rotation metrics.

To highlight the benefits of using a prior map, we provide a contextual comparison with SLAM-based methods, which follow a fundamentally different paradigm. While SLAM systems rely on loop closures and global optimization to reduce drift, this is often insufficient to fully correct long-term error—as seen in KITTI Sequence 05. In contrast, our method leverages a prior map to maintain globally consistent localization without requiring revisit-based correction. As shown in Table 4.12, 4.13, and Figure 4.12, SLAM baselines such as

Table 4.11: Comparison of mean translation and rotation errors (± standard deviation) on KITTI Sequence 05

Method	**Translation Error (m)**	**Rotation Error (°)**
Proposed (Ours)	**0.121 ± 0.077**	**0.30 ± 0.144**
Youngji Kim et al.	0.15 ± 0.14	0.34 ± 0.40
D. Rozenberszki et al	∼2.5 ± 2.0	–
Xiaohu Lin et al.	3.18 ± 5.58	1.27 ± 1.97

Note: Bold values indicate the best performance across each metric.

KISS-SLAM and MOLA still exhibit several meters of drift, whereas our approach remains tightly aligned with the reference trajectory.

Table 4.12: Translation (APE) and Rotation Error statistics for **KITTI Sequence 05**

Method	**Metric**	**Max**	**Mean**	**Median**	**Min**	**RMSE**	**Std Dev**
Proposed	APE (m)	**0.483**	**0.121**	**0.102**	**0.007**	**0.143**	**0.077**
	Rot. (deg)	**1.914**	**0.3**	**0.356**	**0.078**	**0.402**	**0.144**
KISS-ICP	APE (m)	5.067	1.460	1.445	0.281	1.604	0.666
	Rot. (deg)	2.696	1.376	1.441	0.000	1.465	0.504
MOLA SLAM	APE (m)	6.817	1.539	1.447	0.306	1.793	0.920
	Rot. (deg)	3.861	1.931	1.886	0.000	2.015	0.577

Note: Bold values indicate the best performance across each metric.

Table 4.13: Translation (APE) Error statistics for **Mulran-KAiST-03**

Method	**Metric**	**Max**	**Mean**	**Median**	**Min**	**RMSE**	**Std Dev**
Proposed	APE (m)	**0.74**	**0.16**	**0.12**	**0.03**	**0.19**	**0.1**
MOLA	APE (m)	19.98	8.83	6.85	2.79	9.99	4.68
KISS	APE (m)	75.82	33.47	32.27	4.49	36.57	14.72

Note: Bold values indicate the best performance across each metric.

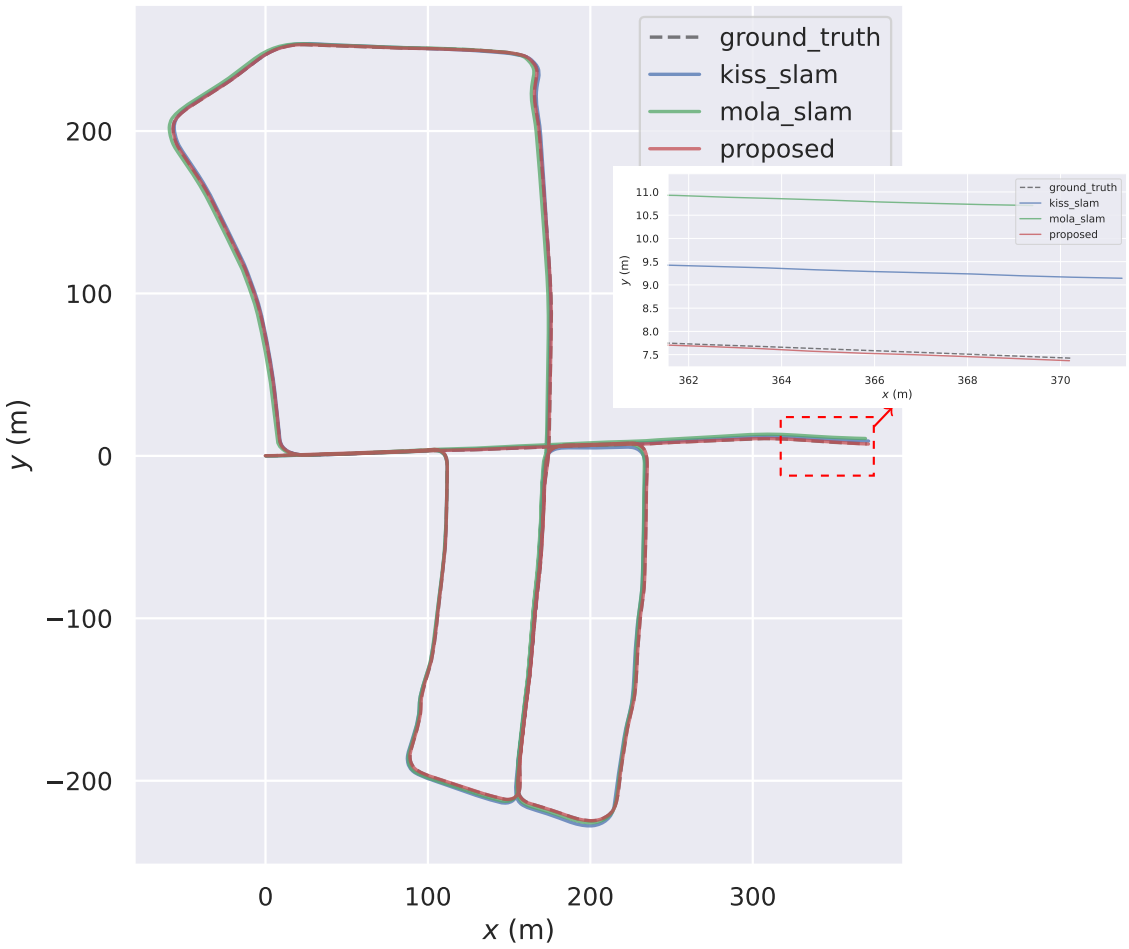


Figure 4.12: **KITTI Sequence 05 – Trajectory Alignment.** The figure shows the estimated trajectories from KISS-SLAM, MOLA SLAM, and the proposed method, overlaid against ground truth.The red dashed box indicates the zoomed region..

4.3 Summary of Results

The overall performance of the proposed localization system across all evaluated datasets is summarized in Table 4.14. The system consistently achieves mean absolute pose errors between 5–16 cm and RMSE below 20 cm, demonstrating strong drift resilience and accuracy. Standard deviation values remain low across sequences, indicating stable and consistency.

Table 4.14: Summary of translation error metrics (APE) for the proposed method across all sequences.

Sequence	**Mean APE (m)**	**RMSE (m)**	**Std. Dev (m)**
Saxion Sequence 1	0.052	0.067	0.042
Saxion Sequence 2	0.090	0.107	0.058
Saxion Sequence 3	0.097	0.135	0.094
Saxion Sequence 4	0.04	0.05	0.033
KITTI Sequence 05	0.121	0.143	0.077
MulRan KAIST 03	0.160	0.190	0.100

Chapter 5

Discussion

This chapter reflects on the results presented in Chapter 4 and discusses how they address the research questions, the implications of the findings, and possible directions for future work.

5.1 Interpretation of Results

5.1.1 Accuracy and Robustness

The results demonstrate that integrating a prior 3D map into a LiDAR-inertial localization framework significantly enhances both accuracy and robustness.By aligning real-time LiDAR scans with globally consistent reference map , the system reduces drift that typically accumulates in odometry-only methods.Experimental results from multiple datasets show that the proposed system achieves centimeter- to decimeter-level translational RMSE(see Table 4.2, 4.3, 4.4), while standalone LiDAR-Inertial Odometry exhibits drift ranging from meter-level to over ten meters in extended trajectories.Unlike SLAM approaches that rely on loop closures, this method consistently aligns live LiDAR scans to a static reference map, maintaining accuracy even in environments with sparse or unreliable loop opportunities(Table 4.12).

To effectively combine high-frequency LiDAR-Inertial Odometry (FAST-LIO2) with NDT-based map matching, the proposed system employs a factor graph optimization framework.The factor graph fuses odometry and scan-matching constraints into a consistent probabilistic model, reducing accumulated linearization errors compared to filtering methods.The sliding-window strategy further ensures real-time performance by limiting the optimization to a fixed temporal window, maintaining constant computational cost regardless of trajectory length (see Figure 4.4a).This design effectively mitigates drift in odometry and

compensates for scan-matching failures, preserving both short-term precision and long-term consistency in challenging environments.

5.1.2 Real-Time Performance

Results show that while standard NDT and ICP struggle to meet real-time constraints as map size increases often resulting in slow convergence or failure, the proposed approach maintains reliable performance (Table 4.5). By operating on locally segmented submaps and leveraging multithreaded NDT-OMP, the system ensures fast and stable scan matching.To manage computational load in large-scale environments, the system incorporates dynamic submap loading and tile-based map management. Instead of operating on a full global map, only local map tiles within a defined radius are loaded and used for scan-to-map matching. This reduces memory usage and significantly improves convergence speed of multithreaded NDT-OMP matching.The proposed pipeline operates at an average rate of 48 Hz (23 ms per frame), as illustrated in Figure 4.4b.

5.1.3 Environmental Adaptability

Under moderate fog (visibility 60m), both NDT and the fusion method exhibit stable performance, with APE increasing only marginally compared to the baseline(Table 4.10). This confirms that NDT scan matching retains resilience in moderately degraded visibility. However, under severe fog (visibility 30m), scan matching becomes unreliable due to poor feature correspondences, and FAST-LIO2 suffers from high drift. As a result, the fusion pipeline repeatedly fails, unable to maintain reliable pose estimates.

In sparse map regions and transition zones, scan-to-map registration frequently fails due to a lack of correspondences. Nevertheless, the fusion method maintains localization thanks to high-rate odometry updates and prior structure in the factor graph(Figure 4.11).

Dynamic object removal further enhances registration performance. Filtering out transient objects improves convergence, reduces iteration counts, and lowers overall computational load during scan matching(see Table 4.7). While the object detection module introduces some processing overhead, it remains feasible for real-time operation, particularly on GPU-accelerated platforms. However, the system may face limitations on resource-constrained devices without hardware acceleration.

5.2 Implications

These findings underscore the value of combining prior maps, real-time odometry, and probabilistic optimization for autonomous navigation. The approach enables accurate and robust pose estimation in GNSS-denied and moderately degraded visual environments, which are common in urban outdoor and indoor settings. The use of dynamic submap loading and multithreaded NDT further supports the system's deployment in real-time and large-scale environments, particularly for high-accuracy, repetitive navigation tasks.

5.3 Limitations and Future Work

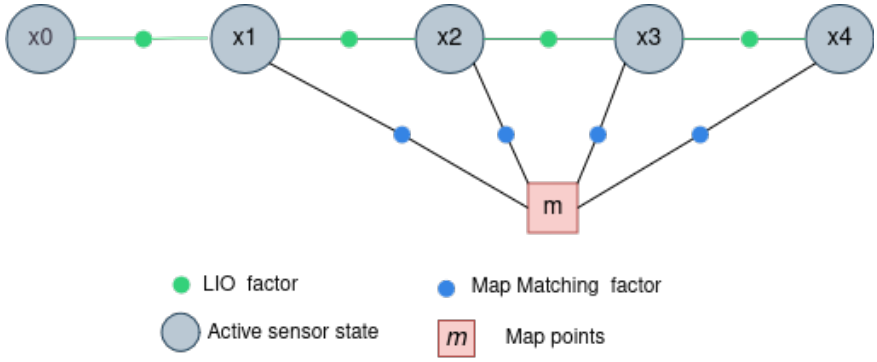
While the proposed localization system demonstrates high accuracy and robustness across diverse scenarios, several limitations remain. First, it assumes a known initial pose, which restricts its ability to perform global localization or recover from kidnapped robot scenarios. Second, the system relies on a static prior map without support for online map adaptation, which limits its effectiveness in long-term or evolving environments.

A further A limitation arises in highly degenerate environments—such as corridors, open plains, or fog—where LiDAR lacks sufficient geometric richness. While FAST-LIO2 and multithreaded NDT are resilient under moderate degradation, their performance degrades in cases of extreme sparsity or planar dominance. FAST-LIO2 relies on consistent local structure for correction, and even NDT-OMP may fail to converge under heavy noise.

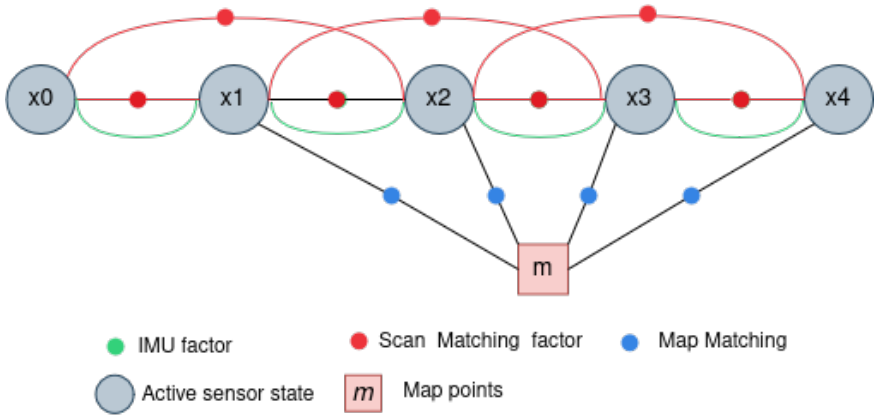
To address this, a future enhancement proposes replacing the tightly coupled LIO front-end with a modular graph-based framework, treating IMU data as a pre-integration factor. This would enable pose propagation during LiDAR degradation and support selective correction when reliable map constraints are available. The flexible structure also allows integration of visual or radar factors for improved localization in perceptually degraded scenes. The current and proposed formulations are shown in Figure 5.1, panels (a) and (b), respectively. In summary, future work may proceed along the following directions:

- **Global localization and recovery:** Integration of coarse initial pose estimation methods for startup or failure recovery.
- **Adaptive mapping:** Support for online map updates or hybrid SLAM fusion during long-term deployments.
- **Modular multi-sensor fusion:** A graph-based architecture incorporating visual, radar, and inertial factors to enhance robustness across sensor failures and environmental degradation.

These enhancements would strengthen the system's applicability in real-world autonomous navigation, particularly in complex, dynamic, and partially observable environments.



(a) Current formulation: tightly coupled LiDAR-Inertial Odometry (FAST-LIO2) with direct map matching via NDT-OMP.



(b) Proposed formulation: modular factor graph where IMU is treated as a pre-integration factor, allowing flexible integration of additional sensors.

Figure 5.1: Comparison of sensor fusion architectures. (a) The current design integrates FAST-LIO2 with NDT-based map matching in a tightly coupled manner. (b) The proposed approach restructures the system into a modular factor graph, supporting IMU pre-integration and extensibility to visual or radar factors for increased robustness in degraded environments.

Chapter 6

Conclusion

6.1 Summary of Findings

This thesis presented a real-time, map-based localization system that combines LiDAR-Inertial Odometry (FAST-LIO2), multithreaded Normal Distributions Transform (NDT) scan matching, and a fixed-lag factor graph optimization framework. The system was designed to provide accurate and drift-resilient 6-DoF pose estimation in large-scale, GNSS-denied, and partially dynamic environments.

Evaluations conducted on benchmark and custom datasets demonstrated that the system consistently achieved centimeter- to decimeter-level translational RMSE, outperforming standalone odometry methods such as FAST-LIO2, which exhibited meter-level drift in long trajectories.The proposed method achieved similar or superior localization accuracy compared to recent map-based approaches.Furthermore,compared to SLAM-based techniques that rely on loop closures, the proposed method maintained better global consistency by continuously aligning LiDAR scans with a pre-built 3D map.

Real-time performance was sustained with an average processing latency of under 23 ms per frame, enabled by a multithreaded NDT implementation and dynamic submap loading. This ensured scalability across large environments while maintaining memory and computational efficiency. Additionally, dynamic object removal using a deep learning-based detection model improved registration stability in semi-dynamic scenes, and NDT demonstrated resilience under moderate visibility degradation such as fog. The system also maintained robust localization in feature-sparse map boundaries and unmapped transition zones, where scan-to-map registration typically fails. By relying on high-frequency odometry within the factor graph, it preserved pose continuity and minimized drift.

These findings confirm that the proposed localization pipeline is well-suited for high-accuracy, real-time navigation in complex, unstructured, and dynamic environments.

Appendix A

Trajectory Plot

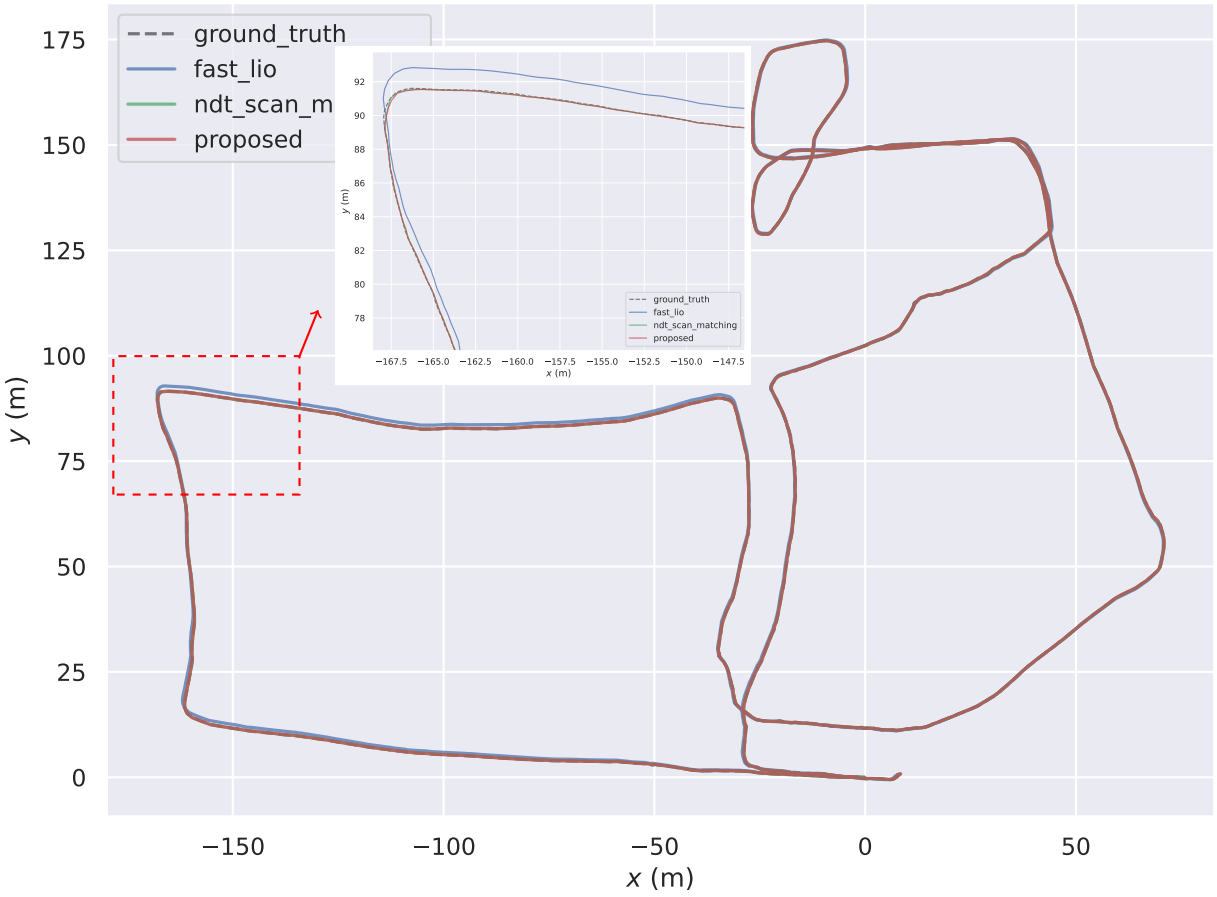


Figure A.1: **Saxion Sequence 02 – Trajectory Alignment.** The figure shows the estimated trajectories from FAST-LIO2, low frequency NDT map matching , and the proposed method, overlaid against ground truth. The red dashed box indicates the zoomed region.

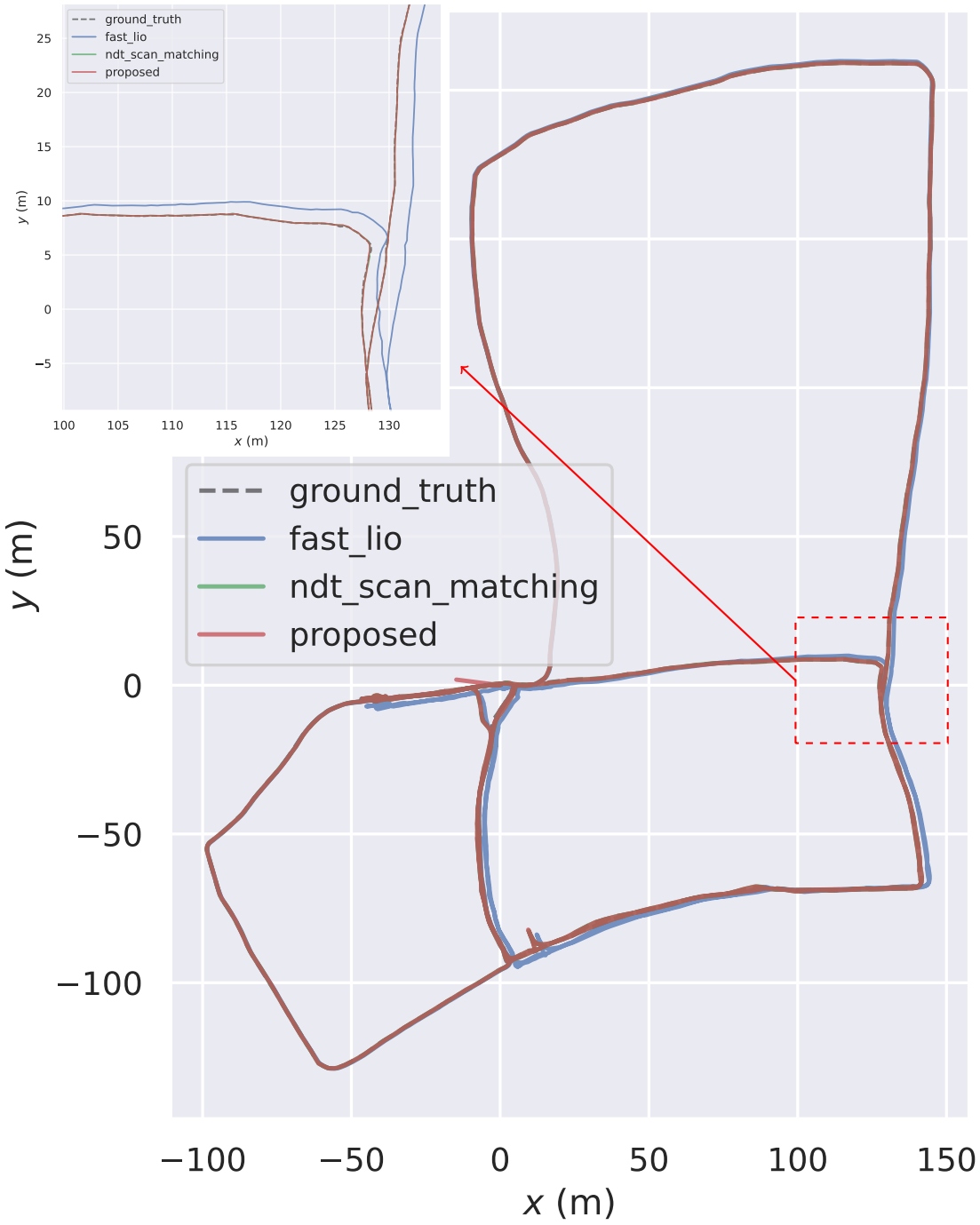


Figure A.2: **Saxion Sequence 03 – Trajectory Alignment.** The figure shows the estimated trajectories from FAST-LIO2, low frequency NDT map matching , and the proposed method, overlaid against ground truth. The red dashed box indicates the zoomed region.

Appendix B: Declaration Form

The signed declaration form is attached as a separate document titled `Eliyas-TDF.pdf`.

Bibliography

[1] Y. Kim, J. Jeong, and A. Kim. "Stereo camera localization in 3D LiDAR maps". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, Oct. 2018, pp. 1–9.

[2] H. Carvalho et al. "Optimal nonlinear filtering in GPS/INS integration". In: *IEEE Transactions on Aerospace and Electronic Systems* 33 (1997), pp. 835–850. DOI: 10.1109/7.599911.

[3] H. Liu et al. "A Precise and Robust Segmentation-Based Lidar Localization System for Automated Urban Driving". In: *Remote Sensing* 11.12 (2019), p. 1348. DOI: 10.3390/rs11111348.

[4] J. Levinson, M. Montemerlo, and S. Thrun. "Map-based precision vehicle localization in urban environments". In: *Proceedings of Robotics: Science and Systems*. Vol. 4. Cambridge, MA, USA, 2007, pp. 1–8.

[5] A. Mohamed et al. "A Review on Visual and Non-visual Odometry Methods". In: *Journal of Robotic Systems* (2020).

[6] X. Wang et al. "Comprehensive Survey of Visual Odometry: Trends and Evaluation". In: *IEEE Access* (2019).

[7] Paul J. Besl and Neil D. McKay. "A Method for Registration of 3-D Shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1992.

[8] Ji Zhang and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time". In: *Robotics: Science and Systems*. 2014.

[9] J. Tang et al. "A Loosely Coupled LiDAR-Inertial Fusion Method". In: *Sensors* (2015).

[10] T. Shan et al. "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2020).

[11] Tong Qin, Peiliang Li, and Shaojie Shen. "LINS: A LiDAR-Inertial State Estimator for Robust and Fast Localization". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3867–3874.

[12] Wei Xu et al. "FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3317–3324.

[13] Wei Xu et al. "FAST-LIO2: Generalized Lightweight LIO system on ROS2". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2022, pp. 5958–5964.

[14] Cesar Cadena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. DOI: 10.1109/TRO.2016.2624754.

[15] Wolfgang Hess, Stefan Kohlbrecher, et al. "Real-Time Loop Closure in 2D LIDAR SLAM". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278.

[16] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters". In: *IEEE Transactions on Robotics*. 2007, pp. 34–46.

[17] Jose Luis Blanco-Claraco. "A flexible framework for accurate LiDAR odometry, map manipulation, and localization". In: *The International Journal of Robotics Research* 0.0 (2025), p. 02783649251316881. DOI: 10.1177/02783649251316881. eprint: https://doi.org/10.1177/02783649251316881. URL: https://doi.org/10.1177/02783649251316881.

[18] T. Guadagnino et al. "KISS-SLAM: A Simple, Robust, and Accurate 3D LiDAR SLAM System With Enhanced Generalization Capabilities". In: *arXiv preprint* arXiv:2503.12660 (2025). URL: https://arxiv.org/pdf/2503.12660.

[19] Kenji Koide et al. "Tightly Coupled Range Inertial Localization on a 3D Prior Map Based on Sliding Window Factor Graph Optimization". In: *arXiv preprint arXiv:2402.05540* (2024).

[20] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[21] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. "Map-Based Precision Vehicle Localization in Urban Environments". In: *Proceedings of the Robotics: Science and Systems (RSS)*. 2007.

[22] Xiaohu Lin et al. “Autonomous Vehicle Localization with Prior Visual Point Cloud Map Constraints in GNSS-Challenged Environments”. In: *Remote Sensing* 13.3 (2021), p. 506. DOI: 10.3390/rs13030506.

[23] Dávid Rozenberszki and András L. Majdik. “LOL: Lidar-only Odometry and Localization in 3D Point Cloud Maps”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[24] Peter Biber and Wolfgang Straßer. “The normal distributions transform: A new approach to laser scan matching”. In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*. Vol. 3. IEEE. 2003, pp. 2743–2748.

[25] Martin Magnusson et al. “Scan registration for autonomous mining vehicles using 3D-NDT”. In: *Journal of Field Robotics* 24.10 (2007), pp. 803–827.

[26] Kenji Koide, Jun Miura, and Emanuele Menegatti. “A Portable 3D LIDAR-based System for Long-term and Wide-area People Behavior Measurement”. In: *Advanced Robotic Systems* (2019).

[27] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. “Center-based 3D Object Detection and Tracking”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11784–11793.

[28] Alex H Lang et al. “PointPillars: Fast Encoders for Object Detection from Point Clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12697–12705.

[29] Yan Yan, Yuxing Mao, and Bo Li. “SECOND: Sparsely Embedded Convolutional Detection”. In: *Sensors* 18.10 (2018), p. 3337.

[30] Holger Caesar et al. “nuScenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11621–11631.

[31] R. E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45. DOI: 10.1115/1.3662552.

[32] Anastasios I. Mourikis and Stergios I. Roumeliotis. “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2007, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024.

[33] Simon J. Julier and Jeffrey K. Uhlmann. "A New Extension of the Kalman Filter to Nonlinear Systems". In: *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*. 1997, pp. 182–193.

[34] Frank Dellaert. *Factor Graphs and GTSAM: A Hands-On Introduction*. https://doi.org/10.5281/zenodo.14117. Accessed: 2024-04-01. 2017.

[35] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. "Factor Graphs and the Sum-Product Algorithm". In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 498–519. DOI: 10.1109/18.910572.

[36] Michael Kaess et al. "iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree". In: *The International Journal of Robotics Research* 31.2 (2012), pp. 217–236. DOI: 10.1177/0278364911430419.

[37] Simon Herbert, Maurizio Scalea, and Cyrill Stachniss. "Fusion of stereo vision and lidar in large-scale urban environments". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2013, pp. 2559–2565.

[38] MMDetection3D Contributors. *MMDetection3D: OpenMMLab next-generation platform for general 3D object detection*. https://github.com/open-mmlab/mmdetection3d. 2020.

[39] Autoware Foundation. *Autoware Universe: An Open-source Software Stack for Autonomous Driving*. https://github.com/autowarefoundation/autoware.universe. Accessed: 2024-04-29. 2023.

[40] Johannes Teufel, André Meißner, and Frank Wagner. "Realistic Rain, Snow, and Fog Variations for Comprehensive Performance Characterization of LiDAR Perception". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 567–574. DOI: 10.1109/IV53394.2022.00086.

[41] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3354–3361.

[42] Giseop Kim et al. "MulRan: Multimodal Range Dataset for Urban Place Recognition". In: *The International Journal of Robotics Research* 40.12-14 (2021), pp. 1394–1407. DOI: 10.1177/02783649211012805.

[43] Xinyu Liu et al. "Large-Scale LiDAR Consistent Mapping Using Hierarchical LiDAR Bundle Adjustment". In: *IEEE Robotics and Automation Letters* 8.3 (2023), pp. 1523–1530. DOI: 10.1109/LRA.2023.3238902.

[44] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM.* https://github.com/MichaelGrupp/evo. 2017.

[45] X. Lin et al. "Autonomous Vehicle Localization with Prior Visual Point Cloud Map Constraints in GNSS-Challenged Environments". In: *Remote Sensing* 13.506 (2021). DOI: 10.3390/rs13030506.