Eötvös Loránd University

Faculty of Informatics

Erasmus Mundus Joint Master in
Intelligent Field Robotic Systems

LiDAR-Based Mapping Module For Autonomous Navigation

Author:
Zewdie-Habtie Sisay
MSc Intelligent Field Robotic Systems (IFRoS)

Internal supervisor:
Zoltán Istenes
Professor

External supervisor:
Mohammad Albaja
Associate Professor, SMART

Budapest, 2025

Acknowledgements

Contents

Chapter 1

Introduction

1.1 Motivation

In recent years, robotics has evolved with the goal to develop intelligent robots that can assist or replace human beings in performing tasks. In order for these robots to operate effectively and efficiently in diverse environment, a set of modules that define the robot autonomy such as path planning, localization, mapping, perception, and manipulation are required. These modules are interconnected to each other in a sense that one can be an input to the other. In similar manner to how we humans understand and interact with our environment, robots also need to have these modules tightly integrated.

Map is an important prerequisite that helps a robot determine the pose of objects within its environment, enabling it to operate and perform tasks such as navigation, obstacle avoidance, and path planning. Both mapping and localization modules are usually combined and referred to as Simultaneous Localization and Mapping (SLAM) module where the robot builds the map while continuously localizing itself inside it. The planning module allows the robot to plan a path from its current pose (position, and orientation) to a goal pose. An algorithm that assists the robot to follow the planned path, a controller, is also considerably essential entity. Using perception module, robot senses and interpret the environment. In same way we humans use our sensing organs to sense our surrounding, robot uses its onboard sensors such as camera, Light Detection and Ranging (LiDAR), Radio Detection and Ranging (RADAR), Inertial Measurement Unit (IMU), Global Positioning System (GPS) and other sensors. The manipulation module enables the robot to interact with objects in its environment by performing tasks such as

grasping, picking, placing, and etc.

Map being an important prerequisite and SLAM being the state-off-the-art solution to generating it, there are several techniques to solving SLAM in the literature using different or similar methods. The two main techniques are online SLAM and full SLAM. These techniques rely on different sensor modalities with LiDAR SLAM and Visual SLAM being two the most widely used approaches.

1.2 Objectives

Objectives: what do you want to achieve? You can also briefly mention the methods, the way you are planning to achieve your objectives and the expected results.

1.3 Research Questions

Research questions: 3-5 specific questions you would like to answer in your thesis. Not too detailed, not too generic. Not too difficult, not too easy.See for example: `https://www.scribbr.com/research-process/research-questions/`
You can also mention the difficulties, why is it difficult to answer the questions, without entering into details of the literature review, helping the reader to evaluate, estimate the difficulty of your work.

1.4 Scope and Limitation

1.5 Thesis Structure

Chapter 2

Literature Review

2.1 Introduction

The purpose of this chapter is to provide background knowledge and review existing research related to LiDAR-Based SLAM, state estimation, and loop detection.

In mobile robotics, map is an important prerequisite that enables robots to operate and perform different tasks such as navigation, obstacle avoidance, path planning, etc. SLAM is the most popular method in the literature to generate maps. Among the different methods of SLAM, vision-based and LiDAR technologies are the most popular in the literature. Although each SLAM method has its advantages and drawbacks, they are essential for different tasks. LiDAR technologies will likely remain essential for years to come due to their numerous advantages: robust to various lighting conditions, accurate 3D measurements with higher Field of View (FoV), among others, despite their high cost and bulkiness to install in small-scale robots like drones. Solid-state LiDARs are cost-effective, lightweight, and precise, making them ideal for UAVs in mapping and complex environments. Although these solid-state LiDARs have good potential, they have presented new challenges to SLAM solutions: 1) When the robot is operating in a cluttered environment, there not be strong features with geometric shapes like edges and planes. This causes the LiDAR SLAM to degenerate easily. 2) Fusing many feature points with IMU is challenging. 3) LiDAR points are sequentially sampled resulting in each point having a different timestamp which causes motion distortion that severely affects the point registration [1].

Several works exist in the literature on LiDAR SLAM. Some of these works use filter-based (Kalman filter) approach while others use optimization-based(factor graphs) for the state estimation. Although filter-based approaches are simple in terms of implementation and computational requirements, they accumulate errors over time causing a larger drift. Optimization-based approaches on the other hand formulate the SLAM problem as non-linear problem and optimize the whole robot trajectory and the map minimizing the drift, more specifically during loop closure. In the following sections, we introduce basic probabilistic formulations, algorithms, and concepts that are used in the research. We also review the most related and relevant works.

2.2 Introduction to joint probability

In robotics, sensor measurements, controls, and robot states are modeled as random variables. This modeling helps to correctly incorporate the uncertainties that exist due to the sensor measurements, the motion model of the robot, or the environment itself. These variables can take multiple values following probabilistic laws. Probabilistic inference calculates the probability distributions of derived random variables, such as sensor data. Let $\mathbf{X}$ denote a random variable and x a possible event. The probability of $\mathbf{X}$ taking a value x is denoted as

$$p(X = x) \tag{2.1}$$

Equation 2.1 can also be written as *p(x)* for simplicity. The sum of all possible probabilities of discrete random variable $\mathbf{X}$ sum to 1.

$$\sum_{x} p(x) = 1 \tag{2.2}$$

To make estimation in a continuous space, a set of continuous random variables with probability density functions(PDFs) are used. Most commonly a normal distribution with mean μ and variance σ^2, shown in equation 2.3, is used.

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \tag{2.3}$$

The PDF for multivariate random variable, usually expressed as vector form, is shown in equation 2.4 where $\mu \in \mathbb{R}^n$ is the mean and Σ is the *covariance matrix*.

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{2.4}$$

To express the joint probability distribution of two random variables $\mathbf{X}$ and $\mathbf{Y}$, we use the notation $p(x,\ y) = p(\mathbf{X} = x \text{ and } \mathbf{Y} = y)$. If $\mathbf{X}$ and $\mathbf{Y}$ are independent, $p(x,y) = p(x)\ p(y)$.

The conditional probability of x given y is denoted as:

$$p(x \mid y) = \frac{p(x,y)}{p(y)} \tag{2.5}$$

Leveraging the conditional probability shown in equation 2.5, Bayes rule can be derived and put as in the following equation 2.6.

$$p(x \mid y) = \frac{p(y \mid x)\ p(x)}{p(y)} \tag{2.6}$$

It is also possible to condition the Bayes rule described in equation 2.6 on arbitrary other variables, z, as shown in equation 2.7**.**

$$p(x \mid y, z) = \frac{p(y \mid x, z)\ p(x \mid z)}{p(y \mid z)} \tag{2.7}$$

Similarly, probabilities of independent random variables x and y can be combined on other variable z as in equation 2.8.

$$p(x, y \mid z) = p(y \mid z)\ p(x \mid z) \tag{2.8}$$

2.3 LiDAR Odometry and LiDAR Inertial Odometry

move text to methodology or literature review???

Accurate 3D mapping with LiDARs requires efficiently registering the point clouds to a global map frame. This efficiency of registration depends on accurate estimation of the LiDAr pose in continuous motion. Independent motion estimation systems such as Global Navigation Satellite Systems (GNSS) or Inertial Navigation Systems(INS) can be used for registration,however GNSS systems do not provide continuous signal in diverse

environments and INS accumulate drifts over time resulting in inconsistency in the 3D map.

LiDAR odometry (LO) is a continuous estimation of the motion of the vehicle by analyzing successive 3D point clouds captured by LiDAR sensor mounted on it. This works by tracking the changes in the structure of the environment that is achieved aligning the current scan to the previous and computing the relative motion. LO methods are robust in environments with rich features (planes, and edges). The most widely used registration method Iterative Closest Point (ICP) is one example that can be used to estimate motion with LO.

However, LO suffers from drift and degeneration specially in challenging environments such as repetitive structures and dynamic objects. Fusing high frequency IMU data with LiDAR can rescue LO and by reducing the drift thereby improving the motion estimation. This fusion of LiDAR and IMU for motion esstimation is known as LiDAR Inertial Odometry (LIO). Several researches [2, 3, 1] have been conducted demonstrating the effectiveness LIO for motion estimation and 3D mapping.

2.3.1 State estimation??

2.4 Introduction to Bundle Adjustment

Bundle adjustment (BA) is a technique used to jointly solve both the 3D structure and camera poses in many visual applications such as structure from motion (SfM) and Visual-SLAM (V-SLAM). It works to minimize the re-projection error, see figure 2.1, between the observed 2D features in the images and projected 3D points(features), changing both the camera poses and the pose of 3D points to minimize the error.

One important thing to consider when working with BA is its cost. The computational complexity of BA is $\mathbf{O}(qN+lM)^3$ where N is the number points, M is the number of poses, q and l the number parameters for the points and the camera respectively. To mitigate this drawback of BA, different ways can be incorporated including using windowed BA with small window size, and optimizing only on poses and keeping 3D feature points fixed[**bibid**] paraphrase this paragraph, it was copied from Nunos lecture notes

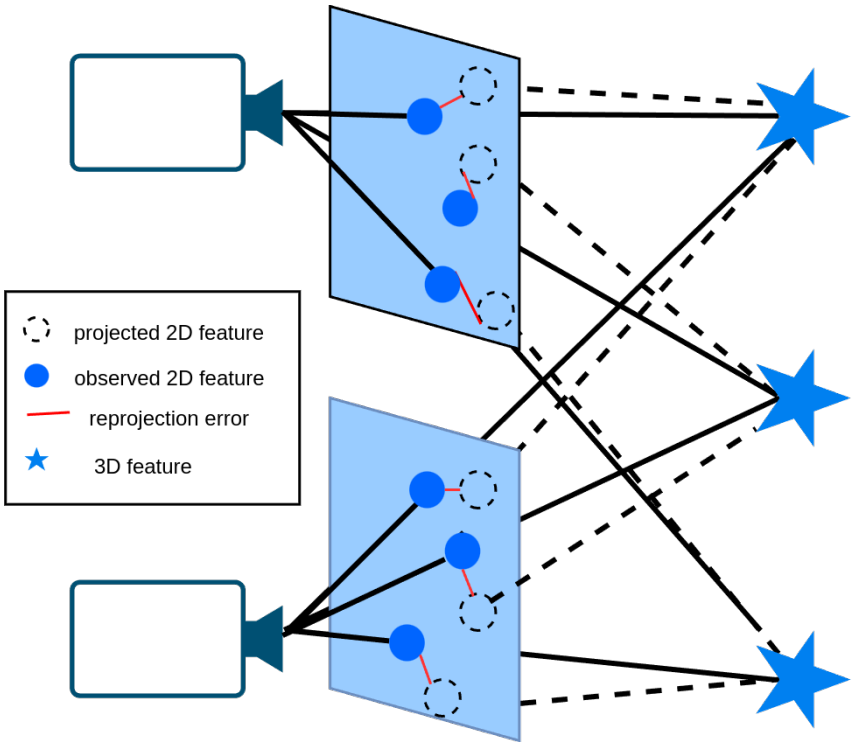


Figure 2.1: Visual BA

2.4.1 LiDAR Bundle Adjustment

In large-scale SLAM applications—such as at the city scale—map inconsistencies are common, even after Pose Graph Optimization (PGO), since PGO optimizes only relative poses. Additionally, loop closure errors can significantly impact the global consistency of the map [4]. To address this, LiDAR BA (LBA) can be employed to improve global consistency. Furthermore, vertical misalignments, which most LIO systems are vulnerable for, can be corrected using LBA, although GPS data, if available, may also help mitigate vertical drift.

Similar to visual BA, the LBA problem can be formulated to jointly determine the LiDAR poses and the global 3D point cloud map thereby decreasing the drift caused by accumulation of scan registration errors. Even though LBA seems simpler than visual BA due to accurate depth measurements, its formulation is not trivial. Due to the sparse and non repetitive nature of LiDAR point clouds, the natural formulation used in visual BA to minimize the re-projection error does not apply to LBA [5].

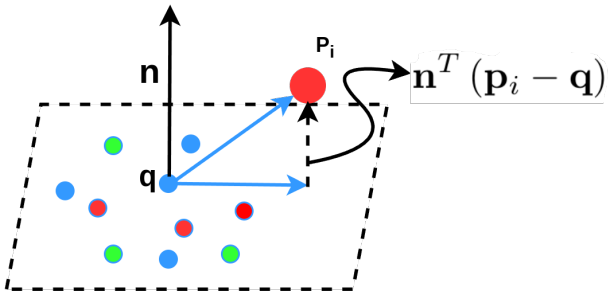


Figure 2.2: A plane in space and corresponding feature points drawn from different scans

A work by Liu et al. [6] introduced a formulation of BA on sparse LiDAR feature points with edges and planes to refine map incrementally built with LiDAR SLAM. In this formulation, unlike visual BA, the plane or edge parameters (the normal vector $\mathbf{n}$, and point in the plane $\mathbf{q}$, Eq. 2.9) are first analytically solved such that the BA problem is only related to LiDAR poses and such parameters are eliminated in each iteration of optimization [7]. This drastically reduces the cost of BA by enforcing it to only perform the so called motion-only BA without the need to optimize the 3D points.

$$\min_{\mathbf{n},\mathbf{q}} \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{n}^T(\mathbf{p}_i - \mathbf{q})\right)^2 \tag{2.9}$$

Equation 2.10 shows the direct formulation of LBA that minimizes the sum of squared distances of each LiDAR point to the plane where $\mathbf{p}_i$ is a point observed from pose $\mathbf{T}_i$, and $\mathbf{A}$ is the covariance matrix of the observed points, λ_k is the k-th smallest eigenvalue of $\mathbf{A}$, $\mathbf{u}_k$ is the corresponding eigenvector.

$$\begin{aligned}
(\mathbf{T}^*, \mathbf{n}^*, \mathbf{q}^*) &= \arg\min_{\mathbf{T},\mathbf{n},\mathbf{q}} \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{n}^T(\mathbf{p}_i - \mathbf{q})\right)^2 \\
&= \arg\min_{\mathbf{T}} \left(\min_{\mathbf{n},\mathbf{q}} \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{n}^T(\mathbf{p}_i - \mathbf{q})\right)^2 \right) \\
&= \lambda_3(\mathbf{A}), \quad \text{if } \mathbf{n}^* = \mathbf{u}_3,\ \mathbf{q}^* = \bar{\mathbf{p}} \\
\text{where} \quad \bar{\mathbf{p}} &= \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i, \quad \mathbf{A} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^T
\end{aligned} \tag{2.10}$$

The optimization on the scan poses $\mathbf{T}$ finally corresponds to minimizing the eigenvalues of matrix $\mathbf{A}$ in Eq. 2.10. The formulation defined in Eq 2.10 requires the set of points, $\mathbf{p}_i$ acquired from set of scan poses $\mathbf{T}$, belonging to the same plane feature to be defined. Adaptive voxelization technique [6] is used to define the points that belong to the plane. If all points in the voxel lie on the plane by examining the eigenvalue of the point covariance

matrix, that voxel is kept in memory otherwise the voxel is further subdivided into eight octants until a certain minimum threshold voxel size is reached. This generates voxel map with voxels of different sizes adapted to the environment.

The voxilization process described above extends to non-planar features such as curved surface when the voxel map is performed at a finer level. The maximum depth of a tree and the minimum number of points contained in a voxel are the two conditions used to stop the recursive subdivision of a voxel.

2.5 Optimization

2.6 Overview and types of SLAM

There are several types of SLAM solutions. Depending on the type of exteroceptive sensors used, LiDAR SLAM and V-SLAM are the most commonly used types. LiDAR SLAM uses laser sensors to measure distances and create precise 3D maps, making it highly effective in dark or low-feature environments. However, its high cost and power consumption make it less suitable for low-cost robotics. On the other hand, V-SLAM depends on monocular, stereo, or RGB-D cameras to detect visual features and track motion, providing a more affordable and lightweight solution. While V-SLAM captures detailed textures and colors, it struggles in low-light conditions and areas with few visual features. Despite these differences, both types are essential in robotics and autonomous systems. To improve accuracy, many modern applications combine both techniques using sensor fusion, taking advantage of their strengths to enhance localization and mapping performance. Based on the solution methodology used, SLAM can be further categorized into online SLAM and Full SLAM.

2.6.1 Online SLAM

Online SLAM methods estimate the current state x_t and the whole map m based on all past observations $z_{1:t}$ and controls $u_{1:t}$. These methods use Kalman Filter(KF), Extended Kalman Filter (EKF), Particle Filter (PF), and Unscented Kalman Filter (UKF)[8]. EKF-SLAM is one of the most popular online SLAM algorithms that uses Bayes filter to estimate the robot poses and landmark locations, assuming Gaussian distributions and local linearity. Bayes filter is a recursive filter for state estimation that includes two steps,

prediction and update as shown in Eq (2.11), where $\bar{belief}(x_t)$ represents the prior belief about the robot state at time t before incorporating the latest observation (prediction step) and $belief(x_t)$ is a posterior belief that defines the robots refined state after incorporating latest observation (update step).

$$\begin{aligned} \bar{belief}(x_t) &= \int p(x_t|x_{t-1}, u_t)\, belief(x_{t-1})\, dx_{t-1}, \\ belief(x_t) &= p(z_t|x_t)\bar{belief}(x_t). \end{aligned} \tag{2.11}$$

Map generated with online SLAM is not accurate due to the linearization errors accumulated over time, local state estimation and absence of loop closure constraint to refine accumulated errors.

2.6.2 Full SLAM

FuLL SLAM aims to solve the problem of finding the most probable sequence of robot poses $x_{1:t}$ and the map m, given all sensor measurements $z_{1:t}$ and control inputs $u_{1:t}$[9]. It is mostly used to build consistent map using point clouds and solve SfM problems, specially to handle loop closure events.

Bayesian network

One approach to solving the Full SLAM problem is using Bayesian network where the problem is formulated as joint distribution of all the variables($x_{1:t}, z_{1:t}, u_{1:t}$, and m). It is a probabilistic directed graph model for solving full SLAM where the entire problem is formulated as joint distribution of the robot poses ($x_{1:M}$), the measurements ($z_{1:K}$), controls ($u_{1:M}$), and landmarks ($l_{1:N}$). The joint probability distribution of the whole SLAM is defined as in equation 2.12 where the variables X, L, U, and Z are all robot poses, all landmarks, all control inputs, and all measurements respectively (Eq.2.13). Figure 2.3 shows a typical Bayes network where every node encodes a conditional probability on the variable and its parent nodes.

$$P(X, L, U, Z) \propto P(\mathbf{x}_0) \prod_{i=1}^{M} P(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \mathbf{u}_i) \prod_{k=1}^{K} P(\mathbf{z}_k \mid \mathbf{x}_{i_k}, \mathbf{l}j_k) \tag{2.12}$$

$$\begin{aligned} X &= \{\mathbf{x}_i\}, \quad i \in 0 \dots M \\ L &= \{\mathbf{l}_j\}, \quad j \in 1 \dots N \\ U &= \{\mathbf{u}_i\}, \quad i \in 1 \dots M \\ Z &= \{\mathbf{z}_k\}, \quad k \in 1 \dots K \end{aligned} \tag{2.13}$$

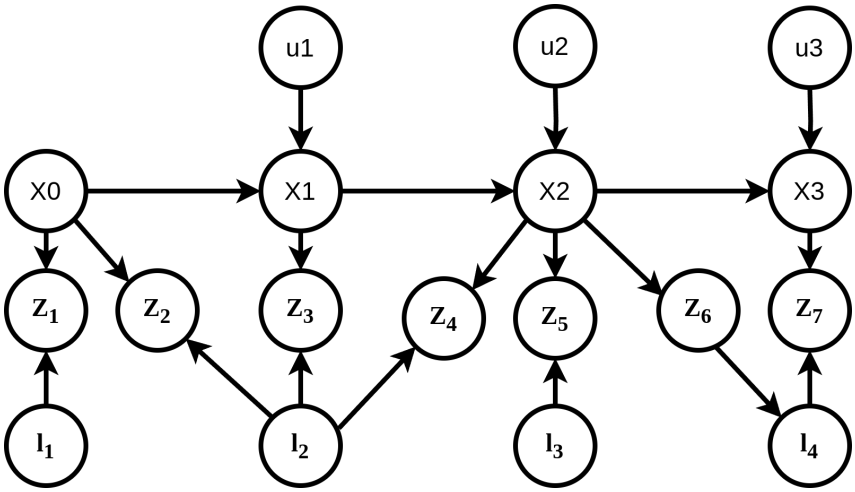


Figure 2.3: Probabilistic Bayesian net SLAM model

Factor Graph

Bayesian network uses only arrows to define dependency between variables which does not encode much of information. Effortlessly, Bayesian network can be converted to factor graphs by conditioning on sensor data [10]. Figure 2.4 shows the factor graph obtained by converting the Bayesian network shown in figure 2.3. Each node in the Bayesian network is split in both variable node and factor node in the graph. The factor is connected to a variable node and parent variable nodes in the graph.

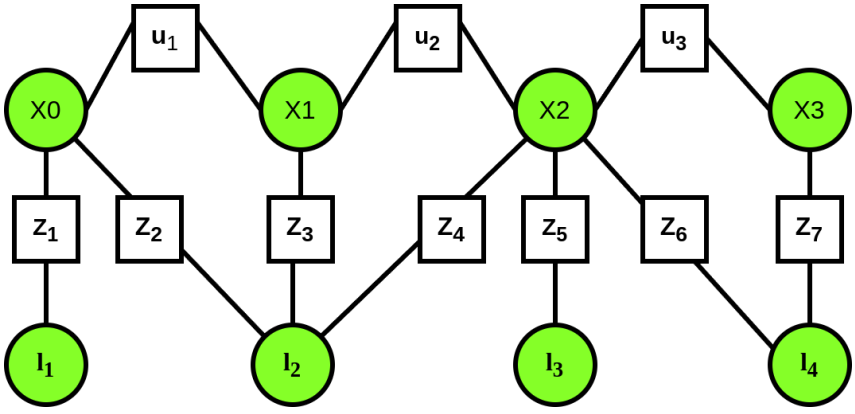


Figure 2.4: Full SLAM model with factor graph

Factor Graphs represent the probability distribution as a bipartite graph with variable nodes (green circles in Fig. 2.4) and factor nodes (white squares in Fig. 2.4) where edges can only connect variable nodes via factor nodes. These factor nodes encode information that enters into the system and the graph captures the propagation of these information to the hidden states that we intend ot estimate. We denote variable nodes as $\mathbf{x}$ and factor

nodes as $\boldsymbol{\phi}$. The factors define the probabilistic constraint motion model ($\boldsymbol{\phi}_i$) and the likelihoods measurements ($\boldsymbol{\phi}_\mathbf{k}$). The joint probability distribution of the problem is then computed as the product of all factors in the graph(Eq. 2.14). This graph structure allows efficient optimization via methods such as Gaussian Belief Propagation or Nonlinear Least Squares Optimization. Factor graph is the core of the prevalent SLAM technique known as pose graph optimization (PGO). Most of the LiDAR SLAM works [7, 3, 11], even the most recent KISS(Keep It Small and Simple) SLAM [12] use PGO with factor graph to generate 3D map of the environment from LiDAR point clouds.

$$p(\mathbf{X} \mid \mathbf{Z}) \propto \prod_{k=1}^{K} \phi_k \quad \text{where } \phi_k \geq 0 \quad \forall K \tag{2.14}$$

Solving the SLAM problem with factor graph is exactly determining the values of the unknown variable nodes($\mathbf{X}$) that maximize the agreement with the uncertain measurements($\mathbf{Z}$). This is known as maximum a posteriori estimate (MAP). Solving for the states (variable nodes) that maximize the posterior $p(X \mid Z)$ (Eq. 2.15) can be performed by exploiting non-linear optimization methods.

$$X^{MAP} = \arg\max_X p(X|Z) = \arg\max_X \frac{p(Z|X)p(X)}{p(Z)} \tag{2.15}$$

2.7 Related works

Ji Zhang et al. [13] proposed a real-time method for LiDAR odometry and mapping. They achieved low-drift results without the need to use IMU and high-accuracy range measurements. The costly SLAM problem is divided into to algorithms where one algorithm computes the odometry at a higher frequency and the other algorithm runs at a much lower frequency to register point cloud. While it achieves impressive accuracy and real-time performance, it lucks inertial fusion and loop closures are not considered.

A method by Xu et al. [1] presented a novel approach to tightly-coupled LiDAR-inertial odometry by proposing a new formula for computing the Kalman gain, which demonstrates equivalence to the conventional formula but reduces computation complexity based on state dimensions rather than measurement dimensions. An Iterative Kalman Filter(IKF) is used to minimize the point-to-plane distance during the registration of scan to map. The authors implemented their method into a robust LiDAR-inertial

odometry package called FAST-LIO that operates effectively on a small-scale quadrotor. Experiments conducted in varied environments validate the system's resilience against fast motion and vibration noise. Despite its resilience to fast motion and cluttered environments, FAST-LIO does not include loop closure or global pose graph optimization causing accumulation of drift over long trajectories.

Kim et al. [14] presented Scan Context, a novel spatial descriptor designed for outdoor place recognition using 3D LiDAR scans. Unlike traditional histogram-based methods [15, 16, 17], Scan Context encodes the entire point cloud into a matrix format, preserving the absolute geometrical structure of the environment. This approach utilizes a cosine distance measure for similarity scoring and implements a two-phase search algorithm to enhance loop detection efficiency, particularly in urban settings. Experimental results demonstrate that Scan Context significantly outperforms existing global descriptors in various datasets, providing robust performance against noise and viewpoint changes.

A work by Xiyuan Liu et al. [7] presented a novel approach for large-scale LiDAR mapping by introducing a hierarchical LiDAR bundle adjustment (HBA) framework. The method aims to improve the consistency and accuracy of LiDAR-based maps by formulating a global optimization problem that jointly refines multiple scans. Unlike conventional SLAM techniques that rely on pairwise constraints, HBA leverages a hierarchical structure to balance local precision and global consistency, mitigating drift accumulation over extended trajectories. The authors demonstrate the effectiveness of their approach through extensive experiments on real-world datasets, showcasing its advantages in handling large-scale environments with minimal loop closures. Their results indicate that HBA outperforms existing methods in terms of mapping accuracy and robustness, particularly in scenarios with sparse revisits. A key contribution of this work is the introduction of an efficient optimization strategy that scales well with the number of LiDAR scans. By organizing the LiDAR data into hierarchical layers, the proposed framework enables efficient refinement without requiring excessive computational resources. The authors also incorporate a robust outlier rejection mechanism to enhance data association, ensuring reliable pose estimation in challenging conditions. Furthermore, the method integrates seamlessly with existing LiDAR-based odometry pipelines, making it adaptable to various robotic platforms. However, BHA map consistency degrades if there are more revisits and loop closures.

Wei Xu et al. [2] significantly improved upon the original work [1] by introducing two key advancements: direct point registration and the incremental k-D tree (*ikd-Tree*). Unlike [1], which relies on hand-crafted feature extraction, the new approach registers raw LiDAR points directly to the map, increasing accuracy and adaptability to different LiDAR sensors. The newly introduced ikdTree enables efficient incremental updates, dynamic re-balancing, and on-tree downsampling, reducing computational load while maintaining real-time performance. These improvements allow a new approach to achieve higher accuracy at a lower computational cost, making it more robust for diverse environments, including those with small FoV LiDARs and aggressive motion. As showing Figure **??** and Figure **??**, ikd-Tree data structure achieves best overall performance both for point insertion time and point search time.

Chapter 3

Methodology

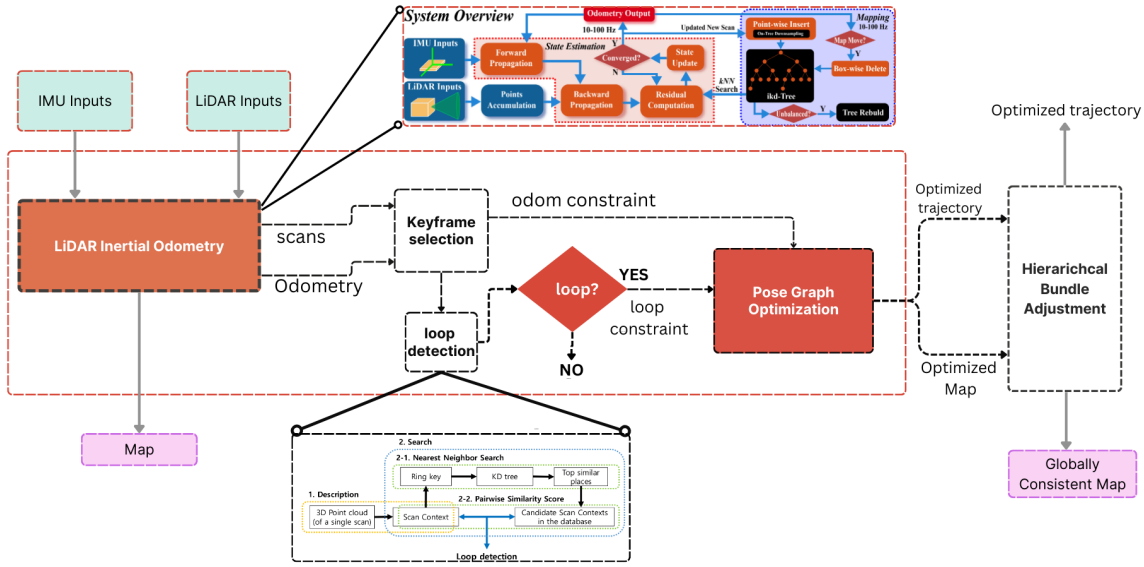


Figure 3.1: System overview of FAST-LIO with loop closure detection and optimization.

3.1 System Overview

The diagram illustrates a LiDAR-based FAST-LIO mapping system with loop closure detection and pose graph optimization. The key components include:

- **FAST-LIO**: This module integrates IMU and LiDAR data to estimate the system's state in real time. It employs forward propagation for state prediction and backward propagation for refinement. Point cloud data is accumulated and processed using

kNN search and residual computation to update the system state. The odometry output is generated at 10-100 Hz, while mapping is performed using an ikd-Tree for efficient point-wise insertion and deletion.

- **Scan Context**: This block is responsible for detecting loop closures by comparing newly acquired scans with previously stored scan contexts. It involves two main steps: (1) Nearest Neighbor Search, where a KD-tree and ring key are used to retrieve the most similar scan locations, and (2) Pairwise Similarity Score computation, which evaluates the resemblance between scan contexts to identify loop closures.
- **PGO (GTSAM)**: Optimizes the detected loop closures to refine the trajectory and map.
- The final output is an optimized trajectory and an improved map representation.

3.1.1 FAST-LIO2

3.1.2 Overview of loop detection and closure techniques

Scan context establishes compact representation by capturing highest points from the structure in the absence severe disturbances in the roll-pitch directions.

Sequential state estimation from odometry suffer from drift over long trajectory due to inherent noise in the robot motion, in the sensor measurements, data association problems, or/and dynamics in the environment [18]. Due to such a drift, when the robot revisits the same place again, its belief may indicate totally different location than the one it already knows before. Efficiently detecting loop closure overcomes this problem by correcting the drift resulting drastically improved state estimate by integrating pose constraint in pose graph.

3.1.3 Loop closure

Overview of Loop Closure Techniques and explanation about scan context showing sample loop detec

- Feature-based methods (e.g., Scan Context, ORB-SLAM)....
- Geometric-based methods() ICP, GTSAM-based approaches....
- Learning-based approaches (e.g., deep learning for place recognition)

Gupta et al. proposed a robust loop closure detection pipeline for LiDAR-based SLAM that effectively handles variations in LiDAR sensors, including different scanning patterns, fields of view, and resolutions. Their approach generates local maps from consecutive LiDAR scans and employs a ground alignment module to ensure consistency across diverse motion profiles. A key feature of their method is the use of density-preserving Bird's Eye View (BEV) projections, from which ORB feature descriptors are extracted for efficient place recognition. These descriptors are stored in a binary search tree, enabling fast retrieval while mitigating perceptual aliasing through self-similarity pruning.

3.1.4 Outlier rejection in Pose Graph Optimization > SC-Dist

Description of the parameter to control the loop detection in scan context supported with equations. Also provide overlayed trajectory of the trajectories of odom, pgo, and hba. In addition add error at the loop closure

3.2 Dynamic Objects removal

TODO!!!!

Chapter 4

Experiments and Results

4.1 Dataset

put desciption of the dataset and the sensors used in each and parameters things like clearly.

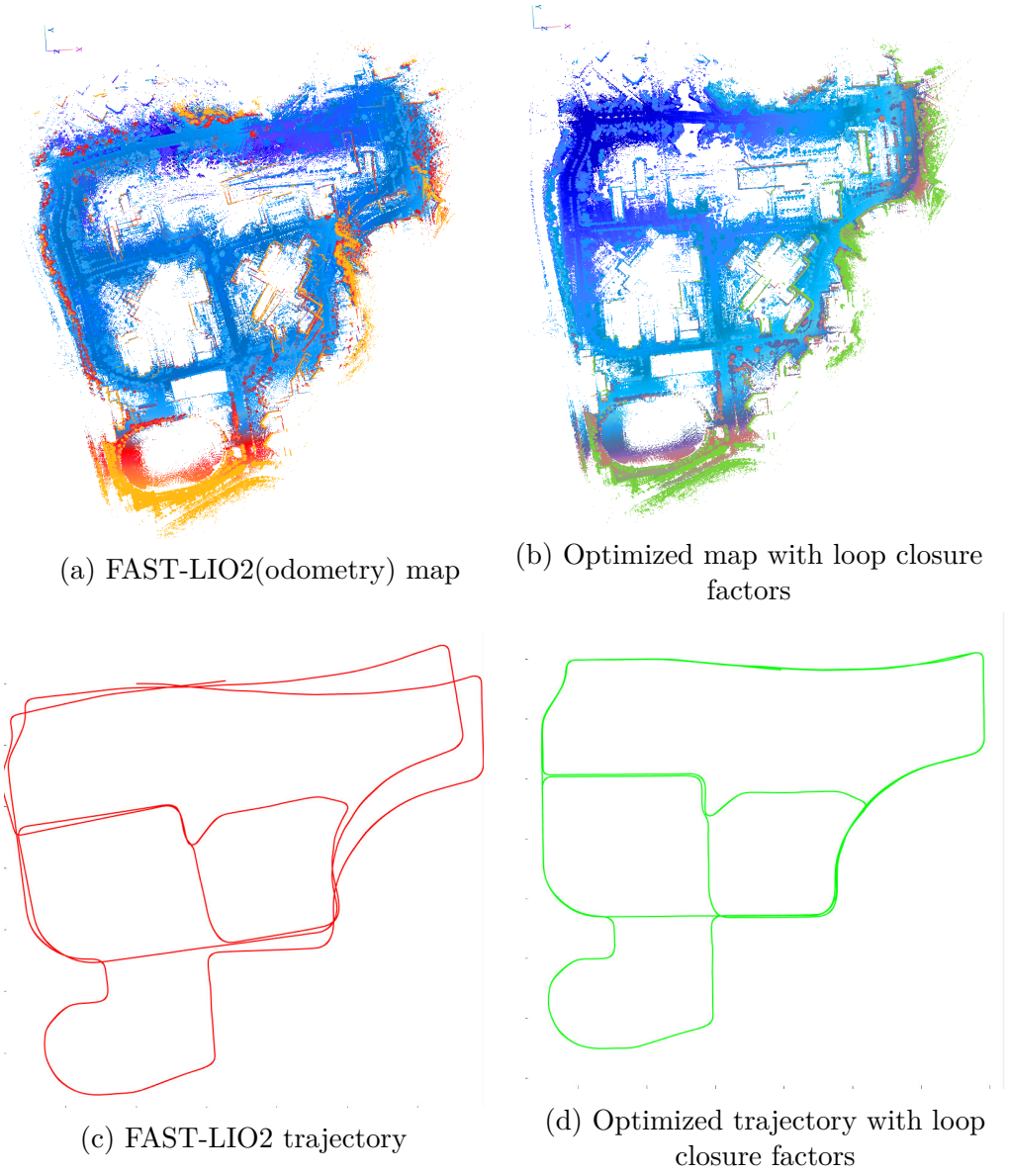
To evaluate the performance of our mapping module, we performed experiments with various open source dataset and our own dataset collected around Saxion building. The experiments performed with the open source dataset were performed to evaluate the robustness of our mapping module to different variety of sensors types, seasonal and environmental changes. opensource dataset contains LiDAR data, IMU data, gps data, ZED camera data. the LiDAR type is Ouster with 128 scan lines and 360deg FoV. The rate of the IMU, and LiDAR .....

Table 4.1: Dataset used for evaluating performance of our mapping module

Dataset	LiDAR	IMU	GPS	Ground Truth	Size	Duration(s)
Saxion-fEight	OS-128	xsens	CL	✗	15GB	???
MulRan-KAIST02	OS-64	xsens	CL	✓	23GB	???
HeLiPR DCC06	OS-128	RTK	RTK	✓	47GB	1074
KITTI 183	V-64??	???	CL	✓	18GB	???

In order for the SLAM to work correctly for each dataset, an extrinsic calibration between the IMU and LiDAR sensors is required. Table 4.1 the calibration matrix for each dataset used in the experiments.

4.1.1 Analysis of Accuracy



(a) FAST-LIO2(odometry) map

(b) Optimized map with loop closure factors

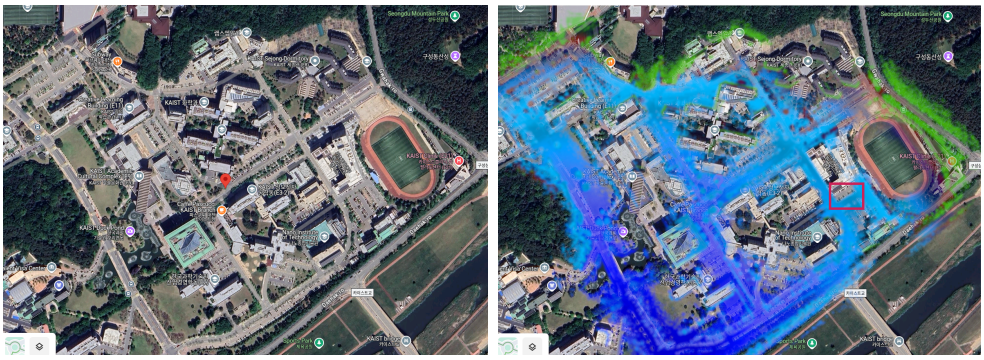
(c) FAST-LIO2 trajectory

(d) Optimized trajectory with loop closure factors

Figure 4.1: Map and path comparison of odometry vs factor graph for MulRan KAIST02 sequence

Figure 4.1 shows the result of the map and trajectory before and after applying optimization to the output of the LiDAR odometry based poses. It can be clearly seen in 4.2b that the map generated by odometry is not accurate and it has duplication in most part of the map specially on the top right corner side.

As initial step, we tested on the Mulran KAIST02 sequence using the odometry from FAST-LIO2 and generated map both from the odometry and pose-graph optimization.



(a) KAIST google map (b) Map overlayed to google map

Figure 4.2: Odometry showing direction of driving(not fully optimized because both optimization and publishing the odometry run in separate threads so the published odometry is not final optimized result)

The map shown in 4.1b is generated after the FAST-LIO2 poses are optimized by ISAM2 after adding loop closure constraints to the factor graph. Since this map is generated from the optimized poses, the duplications are now removed and is more consistent than the previous map. The drift accumulated by FAST-LIO2 odometry can be seen in figure 4.1c and figure 4.1d shows the trajectory after the drift is corrected with ISAM2.

To further analyze the accuracy of the optimized map, we overlayed the generated map onto the google map, and it can be seen that some small inconsistencies are visible because unlike LiDAR bundle adjustment methods [**bibid**], pose graph optimization does not optimize point cloud consistency directly[19]. Instead, it focuses solely in optimizing relative pose errors. Hence the divergence is not completely eliminated in the map.

Table 4.2: Metric comparison of our SLAM with state-off-the-art for indoor custom indoor dataset at Saxion

Method	avg runtime	avg freq.	Num loop closures
KISS-SLAM	30ms	34.00 HZ	77
MOLA SLAM	29.72ms	33.65HZ	-
PROPOSED			

Table 4.2 shows the comparison of our method with most recent SLAM methods, and it is shown that KISS-SLAM results in many false positive loop closures resulting inaccuracy in the generated map. The average runtime of both SLAM methods is comparably real-time.

Chapter 5

Conclusion

5.1 Future work

Use deep learning methods for loop closure detection to improve efficiency, and improving time efficiency of the overall system. Geo-referencing the SLAM map for easy evaluation and accuracy checking

Bibliography

[1] Wen Xu et al. “FAST-LIO: A Fast, Robust LiDAR-inertial Odometry Package by Tightly-coupled Iterated Kalman Filter”. In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 7270–7276.

[2] Wei Xu et al. “Fast-lio2: Fast direct lidar-inertial odometry”. In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2053–2073.

[3] Tixiao Shan et al. “Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping”. In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2020, pp. 5135–5142.

[4] Davide Scaramuzza and Friedrich Fraundorfer. “Visual odometry [tutorial]”. In: *IEEE robotics & automation magazine* 18.4 (2011), pp. 80–92.

[5] Jiarong Lin and Fu Zhang. “Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV”. In: *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2020, pp. 3126–3131.

[6] Zheng Liu and Fu Zhang. “Balm: Bundle adjustment for lidar mapping”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3184–3191.

[7] Xiyuan 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.

[8] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. “Analysis and improvement of the consistency of extended Kalman filter based SLAM”. In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 473–479.

[9] Sebastian Thrun. “Probabilistic robotics”. In: *Communications of the ACM* 45.3 (2002), pp. 52–57.

[10] Frank Dellaert, Michael Kaess, et al. “Factor graphs for robot perception”. In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.

[11] 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.

[12] Tiziano 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).

[13] Ji Zhang, Sanjiv Singh, et al. “LOAM: Lidar odometry and mapping in real-time.” In: *Robotics: Science and systems*. Vol. 2. Berkeley, CA. 2014, pp. 1–9.

[14] Giseop Kim and Ayoung Kim. “Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4802–4809.

[15] Marian Himstedt et al. “Large scale place recognition in 2D LIDAR scans using geometrical landmark relations”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 5030–5035.

[16] Naveed Muhammad and Simon Lacroix. “Loop closure detection using small-sized signatures from 3D LIDAR data”. In: *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. IEEE. 2011, pp. 333–338.

[17] Walter Wohlkinger and Markus Vincze. “Ensemble of shape functions for 3D object classification”. In: *2011 IEEE international conference on robotics and biomimetics*. IEEE. 2011, pp. 2987–2992.

[18] S. Gupta et al. “Effectively Detecting Loop Closures using Point Cloud Density Maps”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2024.

[19] Yue Pan et al. “MULLS: Versatile LiDAR SLAM via Multi-metric Linear Least Square”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021.

List of Figures