FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Study and Analysis of Indoor and Outdoor Mapping and Localization Solutions

Thiago Baldassarri Levin

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Filipe Neves dos Santos, PhD

Second Supervisor: Heber Sobreira, PhD

July 28, 2025

Resumo

A operação segura e eficiente de Robôs Móveis Autónomos (AMR) em ambientes internos, como fábricas e armazéns, depende fortemente da sua capacidade de se localizar com precisão e mapear o espaço envolvente. Nesses contextos, a ausência de GNSS e a dinâmica do ambiente tornam o problema de Mapeamento e Localização Simultâneos (SLAM) particularmente desafiador. Esta dissertação propõe a adaptação do VineSLAM — originalmente concebido para ambientes agrícolas — às particularidades geométricas e funcionais de cenários industriais, com o objetivo de melhorar a repetibilidade e a robustez dos sistemas SLAM. Para isso, foi desenvolvida uma framework de simulação e avaliação baseada no Webots e no ROS 2, que permite gerar conjuntos de dados sintéticos, controlados e reprodutíveis, utilizando uma plataforma robótica equipada com sensores LiDAR, IMU e câmaras. Diversos cenários simulados foram criados, incluindo obstáculos móveis, simetrias arquitetônicas e variações de terreno, de modo a testar o comportamento dos algoritmos em condições representativas dos ambientes industriais reais. Os resultados obtidos demonstram que o VineSLAM é eficaz na construção de mapas locais consistentes, embora apresente limitações na correção do erro acumulado, devido à sua dependência da odometria das rodas e à ausência de um sistema de reconhecimento de lugares previamente visitados. Para mitigar essas falhas, foi integrado um módulo de odometria baseado em LiDAR, o que resultou numa melhoria significativa na estimativa de pose. Este trabalho evidencia o potencial de adaptação do VineSLAM a ambientes industriais e reforça o papel essencial da simulação no desenvolvimento de soluções SLAM confiáveis. Abrem-se, assim, novas oportunidades de investigação futura, como a otimização da estimativa de pose com o auxílio de grafos, a fusão de sensores mais eficaz e a validação em cenários reais. Todos os dados e ferramentas desenvolvidos foram disponibilizados publicamente, com o objetivo de fomentar e acelerar a investigação nesta área.

Abstract

Reliable localization and mapping are fundamental to the safe and efficient operation of Autonomous Mobile Robots (AMR) in structured indoor environments such as warehouses and factories, where GNSS is unavailable and the environment is dynamic. This dissertation addresses the problem of analyzing and improving the repeatability and robustness of Simultaneous Localization and Mapping (SLAM) systems in these contexts, with a focus on adapting VineSLAM, a method initially developed for agricultural applications, to the geometric and operational characteristics of industrial environments. The study introduces a synthetic benchmarking framework built with Webots and ROS 2, enabling the generation of reproducible, controlled synthetic datasets using a custom robot equipped with LiDAR, IMU, and camera sensors. Multiple simulated scenarios, including dynamic obstacles, symmetric layouts, elevation changes, and low-traction zones, were designed to stress-test SLAM systems under industrial conditions. Benchmarking results reveal VineSLAM strengths in maintaining local map consistency but expose its global accuracy limitations due to reliance on wheel odometry and lack of loop closure. To address this, the LiDAR odometry module from LeGO-LOAM, adapted for possible IMU integration, was designed to enhance motion estimation and mitigate drift caused by unreliable wheel odometry. The outcomes confirm the feasibility of adapting VineSLAM for indoor structured environments and highlight the importance of simulation-based testing for iterative SLAM development. This work lays the foundation for future research, including the integration of pose graph optimization, tighter sensor fusion, and real-world validation to ensure robust deployment in industrial robotics. All datasets and tools are publicly released to foster open research.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. Adopted in 2015 by all UN Member States, the 17 goals aim to address major global challenges including poverty, inequality, urban development, innovation, climate change, and environmental degradation. Academic research and technological development are essential enablers for reaching these objectives.

This dissertation contributes to these efforts through the development and evaluation of a robust robotic mapping and localization platform, based on multi-modal sensor fusion and SLAM algorithms, for indoor and outdoor environments. The outcomes of this work support technological innovation and intelligent infrastructure that can be leveraged in sustainable urban planning, autonomous navigation, and industrial robotics.

The specific Sustainable Development Goals mentioned have the following names:

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 11 Make cities and human settlements inclusive, safe, resilient and sustainable

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.4	The system promotes the adoption of clean and smart technologies in infrastructure through robotics and SLAM-based automation.	Demonstrated localization accuracy and sensor integration in structured and unstructured environments.
	9.5	Supports scientific research and technological advancement in mobile robotics and perception.	Number of datasets, software modules, and SLAM benchmark results reported.
11	11.6	Contributes to the development of intelligent systems for sustainable urban mobility and mapping.	Application in smart city simulations and reduction of localization errors in GPS-denied spaces.
	11.7	Enhances the safety and usability of public spaces via autonomous mapping and environmental awareness.	Increased SLAM robustness under dynamic conditions (e.g., pedestrians, ramps).

Acknowledgements

This work would not have been possible without the guidance and support of my supervisors, Filipe Neves dos Santos and Héber Sobreira, who invited me to take part in this project. Their expertise and encouragement were essential throughout every stage of the research.

My deepest thanks go to André Aguiar, my supervisor at the host institution, whose daily support, technical insight, and time made a remarkable difference in both the development of this thesis and my personal growth during the process.

Sincere appreciation is also extended to the team at TRIBE, whose collaboration and openness contributed meaningfully to the progress of this work.

Above all, I am grateful to my family for their unconditional love, patience, and belief in me. Their support has always been the foundation of my strength. Without it, I would not be here today. Warm thanks also go to my friends for their encouragement, understanding, and consistent presence throughout this experience

Thiago Levin

"The greatest enemy of knowledge is not ignorance, it is the illusion of knowledge."

Stephen Hawking

Contents

List of Figures

List of Tables

List of Acronyms

AMR	Autonomous Mobile Robots
ATE	Absolute Trajectory Error
BoW	Bag of Wors
CRIIS	Centre for Robotics in Industry and Intelligent Systems
DSO	Direct Sparse Odometry
EKF	Extended Kalman Filter
GNSS	Global Navigation Satellite System
ICP	Iterative Closest Point
IEKF	Iterative Extended Kalman Filter
iiLab	industry and innovation Lab
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
LI	LiDAR-Inertial
LIS	LiDAR-Inertial System
LIV	Lidar-Inertial-Visual
LO	LiDAR Odometry
LOAM	LiDAR Odometry and Mapping
LV	LiDAR-Visual
NDT	Normal Distribution Transform
PF	Particle Filter
RRE	Relative Rotation Error
RTE	Relative Translation Error
ROS	Robot Operating System
RTK	Real-Time Kinematics
SLAM	Simultaneous Localization and Mapping
UGV	Unmanned Ground Vehicles
URDF	Universal Robot Description File
V-LOAM	Visual LiDAR Odometry and Mapping
VIS	Visual-Inertial System
VIO	Visual Inertial Odometry
VLP	Velodyne LiDAR Puck
VO	Visual Odometry

Chapter 1

Introduction

This chapter is divided into four sections. Section 1.1 presents the background and motivation for this work, discussing the role of Autonomous Mobile Robots (AMR) in industrial and agricultural environments, the challenges of reliable localization and mapping, and the motivation for adapting the VineSLAM framework to structured indoor settings. Section 1.2 defines the research problem and objectives, focusing on evaluating and improving the robustness and repeatability of VineSLAM. Section 1.3 outlines the main contributions of the dissertation, including the development of a benchmarking framework, system enhancements, and scientific dissemination. Finally, Section 1.4 describes the structure of the dissertation.

1.1 Context and Motivation

Autonomous Mobile Robots (AMR) have become increasingly important in manufacturing, logistics, and agriculture, enabling the automation of tasks like material transport, inventory monitoring, crop, and harvesting. These systems increase productivity and reduce operational costs by performing tasks that require precision and autonomy. However, effective navigation in real-world environments requires robust and reliable localization and mapping.

Traditional localization solutions such as the Global Navigation Satellite System (GNSS) present significant limitations. GNSS signals are often blocked or degraded in indoor environments such as warehouses or factories. Even outdoors, performance may be compromised by obstructions like trees or buildings. While differential GNSS and RTK can improve accuracy, they are still insufficient for high-precision tasks in complex environments.

3D Simultaneous Localization and Mapping (SLAM) has emerged as a powerful alternative. SLAM enables AMRs to build maps and localize themselves using only onboard sensors, such as LiDAR, cameras, and the Inertial Measurement Unit (IMU). Unlike 2D SLAM, which assumes planar motion, 3D SLAM captures the vertical structure of the environment, which is critical in scenarios with elevation changes, multi-level structures, or uneven terrain.

Despite its potential, deploying 3D SLAM in real-world environments presents significant challenges. Sensor noise, motion distortion, low-feature or repetitive environments, and dynamic

elements can degrade mapping quality and localization accuracy. Drift, in particular—the accumulation of minor errors over time—remains a significant limitation to the long-term robustness of SLAM systems.

This research is framed within the Centre for Robotics in Industry and Intelligent Systems[1] (CRIIS) at INESC TEC, which develops innovative solutions to advance robotics in industrial, agricultural, and forestry domains. Within CRIIS, the TRIBE Lab focuses on robotics and IoT for precision agriculture and agroforestry. At the same time, the industry and innovation Lab[2](iiLAB) provides testbed for the demonstration and validation of advanced production technologies in structured industrial environments.

The work presented in this dissertation builds on the VineSLAM framework [1], [2], a particle-filter-based 3D SLAM system developed at the TRIBE Lab. Originally designed for agricultural robotics—particularly in vineyard scenarios—VineSLAM has demonstrated strong performance in semi-structured, elevation-rich, and symmetric environments. Motivated by these capabilities, this research investigates its adaptation to structured indoor environments typical of industrial facilities. iiLab is now exploring its deployment for use in factory settings, where robust and reliable indoor localization is essential.

Figure 1.1 shows the Model E robot, developed by the TRIBE Lab, which served as a reference platform throughout this research.

Figure 1.1: Modular E robot, developed by the TRIBE Lab at INESC TEC, navigating through a vineyard environment.

[1] https://www.inesctec.pt/en/centres/criis
[2] https://www.inesctec.pt/en/laboratories/iilab-industry-and-innovation-lab

1.2 Goals

This dissertation addresses the challenge of improving the robustness and repeatability of 3D SLAM systems for autonomous mobile robots operating in indoor environments. The central goal is to evaluate and enhance the VineSLAM framework in scenarios involving dynamic obstacles, changes in elevation, and degraded odometry.

The following goals guide the research:

1. **Comprehensive Evaluation of SLAM Algorithms:** Analyze state-of-the-art 3D SLAM systems, identifying their respective strengths and limitations in structured indoor settings.
2. **Creation of Evaluation Tools and Methods:** Develop simulation environments and generate synthetic datasets to test SLAM algorithms under controlled and challenging conditions.
3. **Assessment of VineSLAM:** Evaluate VineSLAM's performance comparing with state of the art solutions across diverse scenarios, focusing on repeatability, adaptability, and resilience to environmental variability.
4. **System Improvement:** Identify weaknesses in VineSLAM and implement targeted enhancements based on experimental findings.

1.3 Contributions

The key contributions of this study were:

- **Simulation and Benchmarking Framework:** A modular, extensible simulation framework developed using Webots and ROS 2. This framework enables creation of synthetic datasets for benchmarking 3D SLAM algorithms under a variety of synthetic yet realistic conditions. Available at *Agrob4Simulator* [3] repository.
- **Public Dataset Release:** A synthetic 3D SLAM dataset was created and publicly released, featuring structured indoor environments with varying levels of complexity designed to evaluate SLAM performance under challenging conditions. The dataset includes synchronized LiDAR, RGB, IMU, odometry, and ground-truth data collected from simulation scenarios based on the iiLab industrial test environment. It is available at *Zenodo* [4][3], as part of the *TRIBE LAB @ INESC TEC datasets* collection [5].
- **Paper Submission:** The research resulted in the submission of a paper titled *"Synthetic Dataset Framework to Evaluate 3D SLAM Performance: A Webots and ROS 2 Approach"* to *8th Iberian Robotics Conference ROBOT 2025*, which details the proposed synthetic dataset generation for the benchmarking framework and comparative analysis of SLAM algorithms.

[3] https://github.com/thiago-b-levin/Agrob4Simulation
[4] https://zenodo.org/records/15274730
[5] https://zenodo.org/communities/criis-inesctec/

- **LiDAR Odometry Module with IMU Integration:** A lightweight LiDAR-Inertial odometry front-end was developed and integrated into VineSLAM to improve localization accuracy and reduce drift in environments where wheel odometry is unreliable.

All datasets, simulation environments, and code developed in this project are publicly released to support reproducible research and open collaboration within the robotics community.

1.4 Document Structure

The document is organized to systematically address the challenges and advancements in SLAM solutions for autonomous mobile robots operating in structured indoor environments. Chapter 2 presents a comprehensive literature review, discussing the SLAM problem, the roles of the front-end and back-end, current LiDAR-based SLAM methodologies, the VineSLAM framework, and relevant benchmarking datasets and simulation tools. Chapter 3 describes the methodology adopted in this work, including the development of a synthetic benchmarking framework using Webots and ROS 2, the design of custom test scenarios, and the evaluation pipeline. Chapter 4 reports the experimental results, analyzing the performance of selected SLAM algorithms under diverse conditions, with a particular focus on the integration and testing of VineSLAM enhancements. Chapter 5 concludes the dissertation by summarizing the main contributions, identifying limitations, and outlining opportunities for future research in improving SLAM robustness and deployment in industrial robotics.

Chapter 2

Background and Literature Review

This chapter is divided into seven sections. Section 2.1 explains the SLAM problem and shows the current SLAM architecture. Section 2.2 examines the role of the SLAM front end, focusing on sensor data processing techniques, including scan matching and loop closure. Section 2.3 transitions to the SLAM back end, exploring optimization strategies such as filter-based and graph-based methods that ensure accurate trajectory and map generation. Section 2.4 review of the lidar-based SLAM algorithms. Section 2.5 discuss the specialized VineSLAM method. Section 2.6 Discuss the current datasets to evaluate SLAM algorithms. Section 2.7 discuss current robot simulators.

2.1 SLAM

The SLAM problem involves estimating the position of a robot while simultaneously constructing a map of an unknown environment [4]. To formalize the SLAM problem, we define several key variables that describe the system.

At any given time step k, the pose of the robot, representing its position and orientation, is denoted as x_k. As time progresses, the sequence of all poses from the starting point up to time k forms the robot's trajectory, denoted as $X_{0:k} = \{x_0, x_1, \ldots, x_k\}$. A set of control inputs, $U_{0:k} = \{u_0, u_1, \ldots, u_k\}$, which typically includes data from wheel encoders or inertial sensors, influence the robot's movements between these poses. In parallel, the robot relies on onboard sensors such as LiDAR or cameras to collect environment observations, represented as $Z_{0:k} = \{z_0, z_1, \ldots, z_k\}$. These observations are instrumental in constructing a map of the environment, denoted as m.

With these definitions, the SLAM problem can be framed as estimating the posterior probability distribution of the robot's trajectory $X_{0:k}$ and the map m, given the sequence of observations $Z_{0:k}$ and control inputs $U_{0:k}$, starting from an initial pose x_0. This relationship can be expressed mathematically as:

$$p(X_{0:k}, m \mid Z_{0:k}, U_{0:k}, x_0) \tag{2.1}$$

In some instances, SLAM can be simplified to focus only on the current pose x_k and the map m,

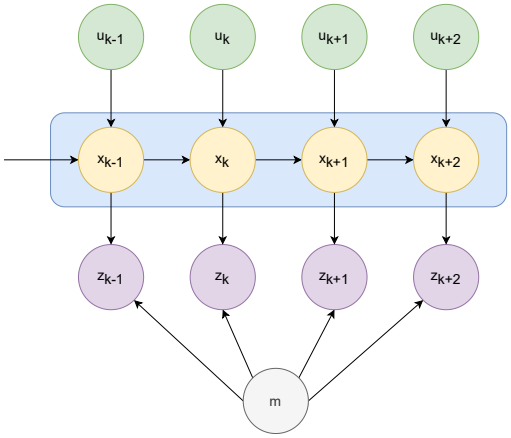


Figure 2.1: Graphical representation of simultaneous localization and mapping (SLAM) [5]. The robot moves from x_{k-1} to x_{k+2} through a sequence of controls. At each time instant, features are observed by the onboard robot sensors and are registered on the global map m.

which leads to the following formulation:

$$p(x_k, m \mid Z_k, U_k, x_0) \tag{2.2}$$

The interplay between these elements—poses, control inputs, observations, and the map—forms the foundation of SLAM. As illustrated in Figure 2.1, this interconnected system highlights the dynamic relationship between the robot's localization and mapping processes. The robot's pose is updated through a motion model that predicts its next state based on the current pose and control inputs:

$$p(x_k \mid x_{k-1}, u_k) \tag{2.3}$$

Simultaneously, sensor data refines this prediction using an observation model, which describes the probability of receiving a specific measurement given the robot's pose and the map:

$$p(z_k \mid x_k, m) \tag{2.4}$$

These models iteratively update estimates of the robot's position and the map as new observations are acquired, balancing accuracy with computational complexity. The key challenge in SLAM is maintaining precision while managing the estimation's computational demands, especially in dynamic environments.

A typical SLAM system Figure 2.2 is divided into two main components: the **front end** and the **back end** [4]. The front end processes raw sensor data, turning it into information suitable for estimation, while the back end optimizes the map and trajectory estimates based on the processed data.

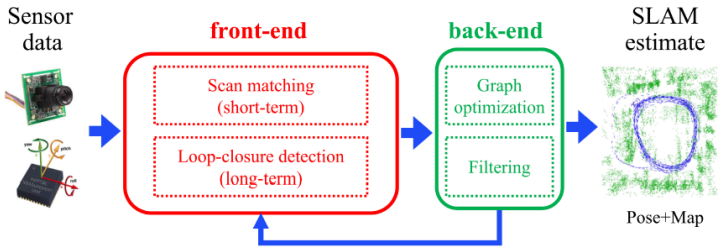


Figure 2.2: Framework of LiDAR SLAM [4]. A traditional system comprises two key components: a front end (red) and a back end (green). It takes sensor data as input and yields mapping and pose information.

2.2 SLAM Front End

The front end of a SLAM system plays a crucial role in processing raw sensor data in a form suitable for estimation. It begins with *data acquisition*, collecting information from various sensors, such as control inputs u_k (e.g., wheel encoders, IMUs) and observations z_k (e.g., LiDAR, cameras). This data provides the foundation for understanding the robot's movement and interactions with its environment.

Feature extraction follows data acquisition, identifying significant features from sensor measurements, such as landmarks, point clouds, or visual key points. This process transforms raw data into structured information for mapping and localization. Two critical techniques within this stage are *scan matching* and *loop closure detection*, described in detail below.

2.2.1 Scan Matching

Scan matching, often called point cloud registration, is a critical technique in LiDAR-based SLAM. This process involves aligning two sets of LiDAR scans to estimate their relative poses, forming the basis for data association and map updating. Among the primary methods, direct matching relies on raw scan points, exemplified by approaches like the Normal Distributions Transform (NDT) [6] models point clouds as probabilistic distributions. Its performance can be inconsistent, and computational requirements are substantial in correlation-based extensions.

Geometric methods, particularly the Iterative Closest Point (ICP) [7], refine correspondences between point clouds and compute the optimal rigid-body transformation to align them. Variants of ICP target improved efficiency or precision, addressing challenges such as partial scan overlap and environmental noise. However, methods based on ICP depend on initialization, making them vulnerable to local minima in complex settings scan matching.

Feature matching methods identify and utilize significant features from LiDAR scans to improve computational efficiency and robustness. Zhang et al. [8] proposed one of the most widely used feature-matching algorithms that extract linear and planar features from LiDAR data. Extensions such as [9] refine feature selection strategies by incorporating additional information such

as curvature and elevation. These methods effectively search for correspondences using closest points but remain sensitive to local extrema, requiring careful initialization. Modern feature matching also incorporates 3D feature descriptors inspired by image feature matching.

2.2.2 Loop Closure Detection

Loop closure detection is the ability of the robot to recognize a place that it has already visited before; this ability allows for position correction by creating constraints on the pose graph, reducing the amount of drift gained throughout the motion. The detection can be made using different sensor modalities.

Proposed by Nister et al. [10], the Bag-of-words(BoW) model generates a vocabulary of visual words that are a combination of features extracted from regions of an image and transformed into descriptors that are clustered using methods like k-means to form a visual vocabulary, each cluster represents a unique visual word, generating the BoW vector that allows images to match with previously seen images. These processes are widely used in visual loop closure methods. Building on that, FABMAP [11] is a method that relies on offline learning of visibility probabilities using a tree structure. Later, Galvez et al. [12] proposed a technique using binary descriptors BRIEF [13] and the FAST [14] feature detector. It efficiently retrieves loop candidates and compresses image data into compact representations, allowing rapid matching. However, these visual methods face challenges in dynamic or visually inconsistent scenarios, where changes in lighting, occlusions, or moving objects can alter the appearance of scenes. Such variations cause descriptors to misidentify revisited locations, reducing the robustness and reliability of visual SLAM in environments with significant feature changes or dynamic elements.

Differently, G.Kim et al. [15] and Uy et al. [16] proposed methods based on 3d point clouds. The first is called Scan Context [15], which converts point clouds into a cylindrical representation that discretizes the 3D space into bins, enabling similarity scores to compute distances between scans for place recognition. This method is primarily designed to estimate a relative yaw angle between LiDAR scans, which doesn't allow correcting the full 6-DoF pose. The second proposal is PointNetVlad[16], which leverages neural networks to generate global descriptors based on 3D point clouds. While these methods can handle specific loop detection tasks, they often lack the speed or robustness required for real-time applications. Additionally, neural network-based methods usually require significant computational resources and pre-training, which can limit their deployment in resource-constrained environments like mobile robots. Another key challenge across these methods lies in balancing efficiency and accuracy. For instance, while some approaches (e.g., Scan Context) offer fast retrieval, they may not provide the detail required for full-pose corrections.

To overcome these challenges, Cui et al. [17] proposed the BoW3D, which adapts the bag-of-words method to work with 3D features, using LinK3D [18] descriptors, which are lightweight, pose-invariant, and suitable for precise point-to-point matching, allowing to identify revisited locations efficiently and corrects the full 6-DoF loop pose in real-time. By dynamically building a

hash table-based database without requiring offline vocabulary training, BoW3D eliminates pre-training constraints and ensures efficient retrieval. Furthermore, its integration into systems like A-LOAM [8] has significantly improved loop closure accuracy and runtime performance.

2.3 SLAM Back End

The back end of an SLAM system aims to optimize the process and reduce drift. The typical back-end algorithms can be categorized into **filter-based methods** and **graph-based methods** [4].

2.3.1 Filter-Based Methods

Filter-based SLAM is grounded in Bayesian filtering principles, where the robot's state and the map are recursively estimated over time. This approach consists of two key stages: the prediction and update step [5].

In the **prediction step**, the next pose of the robot is estimated using a motion model that predicts the state x_k based on the previous state x_{k-1} and control inputs u_k. This can be expressed as:

$$p(x_k \mid x_{k-1}, u_k), \tag{2.5}$$

The motion model accounts for uncertainties in the control inputs, typically represented by Gaussian noise. The predicted joint posterior distribution of the robot's pose and map is updated through integration over all possible states of the previous pose:

$$p(x_k, m \mid Z_{0:k-1}, U_{0:k}, x_0) = \int p(x_k \mid x_{k-1}, u_k) \cdot p(x_{k-1}, m \mid Z_{0:k-1}, U_{0:k-1}, x_0)\, dx_{k-1}. \tag{2.6}$$

This step propagates the uncertainty in the robot's pose forward in time, relying on the motion model and previous belief.

The **update step** incorporates new sensor observations z_k to refine the prediction using an observation model, which relates the sensor measurements to the pose and the map. The observation model is represented as:

$$p(z_k \mid x_k, m), \tag{2.7}$$

where z_k depends on the current pose x_k and the map m. Using Bayes' theorem, the updated posterior distribution is computed as:

$$p(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) = \frac{p(z_k \mid x_k, m) \cdot p(x_k, m \mid Z_{k-1}, U_{0:k}, x_0)}{p(z_k \mid Z_{0:k-1}, U_{0:k})} \tag{2.8}$$

This process corrects the predicted state by incorporating sensor data, effectively reducing the uncertainty in the robot's pose and improving the map.

With all these fundamentals, several filter-based approaches are used to solve the SLAM problem [5], such as:

Extended Kalman Filter (EKF)

EKF-SLAM[19] is based on the Kalman filter method, adapted to handle nonlinear motion and observation models. The state, representing the robot pose and map, is approximated as a Gaussian distribution. The motion model is given by:

$$p(x_k \mid x_{k-1}, u_k) = f(x_{k-1}, u_k) + w_k, \tag{2.9}$$

where $f(\cdot)$ is a nonlinear function of the previous state and control inputs, and w_k represents Gaussian noise with covariance Q_k. Similarly, the observation model is linearized as:

$$p(z_k \mid x_k, m) = h(x_k, m) + v_k, \tag{2.10}$$

where $h(\cdot)$ is the observation function, and v_k is Gaussian noise with covariance R_k. While EKF-SLAM is computationally efficient for small environments, its quadratic complexity with respect to the state size makes it less suitable for large-scale applications.

Particle Filter

Particle filter (PF) based SLAM approaches have become essential in probabilistic SLAM research. A breakthrough in this field was the development of FastSLAM [20], which utilizes N particles. Each particle represents the robot's trajectory $X_{0:k}$ and a set of 2D Gaussian distributions corresponding to landmarks in the map. FastSLAM applies a method known as Rao-Blackwellization, where the path posterior $P(x_k^i \mid u_k, z_k)$ is sampled. The map $P(m \mid x_k^i, u_k, z_k)$ is modeled as a Gaussian distribution (here, i refers to the i-th particle). An enhanced version, FastSLAM 2.0 [21], introduced improvements to the proposal distribution, which led to more accurate trajectory sampling. FastSLAM and its successor leverage Rao-Blackwellization to compute the trajectory and map efficiently.

Building upon this foundation, Grisetti et al. [22] proposed a grid-based SLAM method to optimize performance by reducing the number of required particles. A key advancement in this work was incorporating an improved proposal distribution, which relies on a scan-matching procedure. The scan-matching algorithm identifies the optimal transformation that aligns, for example, a sensor scan with the current map. As a result, particles are sampled based not only on odometry inputs but also on the results of the scan-matching process. This significantly enhances the performance of the particle filter, reducing the computational cost and the need for a large number of particles.

2.3.2 Graph-Based SLAM

Graph-SLAM [23] approaches the problem by constructing a graph where nodes represent robot poses and map features, while edges encode constraints derived from control inputs and sensor observations. These methods optimize the entire trajectory and map by minimizing a cost function over the graph structure. Graph-SLAM aims to optimize the whole trajectory and the map jointly by minimizing a cost function that quantifies the errors in the graph structure.

The optimization problem for Graph-SLAM can be mathematically formulated as follows:

$$F(X,m) = \sum_{ij} e_{ij}(X,m)^{\top} \Omega_{ij} e_{ij}(X,m), \tag{2.11}$$

Here, e_{ij} represents the residual error between the predicted and observed constraints, and Ω_{ij} is the information matrix that encodes the uncertainty of the measurement. The optimal robot poses X^* and the map m^* are determined by minimizing the cost function:

$$(X^*, m^*) = \arg\min_{X,m} F(X,m). \tag{2.12}$$

The edges of the graph are associated with virtual measurements derived from the sensor data. Consider an edge between two nodes x_i and x_j, representing robot poses or map features. The virtual measurement z_{ij} is the relative transformation between these two nodes, often based on odometry or sensor data. The error term for this edge is expressed as:

$$e_{ij}(X,m) = z_{ij} - \hat{z}_{ij}(X), \tag{2.13}$$

Where $\hat{z}_{ij}(X)$ is the predicted measurement given the current configuration of the graph. Optimization aims to adjust the variables X and m to minimize these error terms across all graph edges.

As shown in Figure 2.3, the edge connection in Graph-SLAM illustrates the relationship between nodes x_i and x_j through a virtual measurement z_{ij}. The error e_{ij} is derived as the difference between the predicted and observed measurements, and the uncertainty is modeled by the information matrix Ω_{ij}.

To ensure real-time performance in practical scenarios, Graph-SLAM leverages advanced solvers that handle the sparse structure of the graph efficiently. Incremental approaches, such as iSAM2 [25] and g2o [26].

This formulation enables a robust and flexible framework for addressing the SLAM problem. It integrates diverse sensor modalities and accommodates nonlinear motion and measurement models.

2.4 Lidar-Based 3D SLAM Algorithms

SLAM methods can be broadly classified into two main categories [27]: visual-based and LiDAR-based techniques. Visual-based SLAM has made significant advancements by leveraging camera

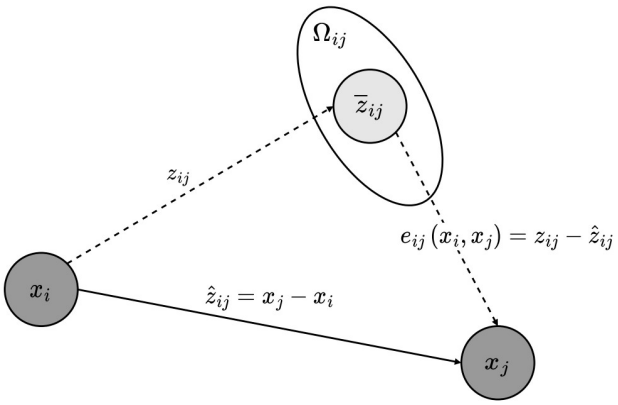


Figure 2.3: Edge connection in the graph formulation of the SLAM problem between vertexes x_i and x_j originated from a measurement z_{ij} [24].

systems such as monocular, stereo, and depth cameras. However, these methods are inherently sensitive to environmental factors such as ambient light and optical textures, limiting their performance in industrial and dynamic environments. LiDAR-based techniques address many of these limitations by providing robust and high-precision measurements through active sensing, unaffected by lighting or motion conditions. Nevertheless, LiDAR SLAM faces its own challenges, including degraded performance in geometrically sparse environments, motion distortion from sensor movement, high computational demands due to dense point clouds, and difficulty in handling dynamic or cluttered scenes. This section explores the current state-of-the-art in LiDAR-based 3D SLAM, categorized into four distinct approaches: LiDAR-only, LiDAR-inertial (LI) fusion, LiDAR-visual (LV) fusion, and LiDAR-inertial-visual (LIV) fusion methods.

2.4.1 LiDAR-Only Approaches

LiDAR-only approaches to SLAM leverage the precision of LiDAR sensors to estimate poses and construct maps with high accuracy, particularly in challenging environments. Zhang et al. [8] proposed LiDAR Odometry and Mapping (LOAM) that set the foundation by separating the odometry and mapping tasks, achieving a balance between computational efficiency and mapping precision; it introduced a significant advancement by identifying effective edge and planar feature points within intricate point cloud data. Additionally, he utilized point-to-line and point-to-plane distances to formulate an error function, which was then employed to address the nonlinear optimization problem for pose estimation. However, reliance on edge and plane feature extraction can be computationally intensive in large-scale settings. Building upon this, Wang et al. [28] proposed F-LOAM that optimizes the computational pipeline and enhances real-time performance. Yet, it remains vulnerable to cumulative errors over extended trajectories. Referring to ground optimization, Shan et al. [29] proposed LeGO-LOAM that introduced ground segmentation to stabilize odometry in uneven and outdoor terrains, outperforming traditional methods in such scenarios but struggling in cluttered environments. Meanwhile, optimized SC-F-LOAM [30] addressed drift

through a Scan context loop closure mechanisms and adaptive thresholds, showcasing improved scalability and reduced drift in large-scale operations.

Recent advancements enhance LiDAR odometry by introducing domain-specific knowledge and kinematic constraints tailored to wheeled mobile robots. Guadagnino et al. [31] proposed Kinematic-ICP, which directly integrates a robot's cinematic model into the point-to-point ICP optimization framework. This approach focuses on wheeled platforms operating on planar surfaces, such as those in warehouses or offices. By constraining the motion estimation to the kinematics of the robot, Kinematic-ICP ensures natural and smooth trajectories, effectively correcting errors from wheel odometry while maintaining high accuracy in pose estimation. An adaptive weighting mechanism dynamically balances the influence of LiDAR and wheel odometry data, allowing the system to handle degenerate scenarios, such as feature-poor environments, effectively. Experimental evaluations in large-scale warehouse operations demonstrated that Kinematic-ICP achieves state-of-the-art performance with reduced computational overhead, showcasing its robustness and applicability in real-world scenarios.

Additionally, Chen et al. [32] proposed a semantic approach SLOAM to integrate object-level understanding, improving robustness in feature-rich but cluttered settings like forests. SLOAM semantic segmentation of trees and ground provides a distinct advantage in environments where purely geometric methods falter. Overall, LiDAR-only SLAM has evolved through iterative enhancements, with each technique addressing the specific limitations of its predecessors, paving the way for highly adaptive and precise localization and mapping frameworks.

Solid-state LiDAR, a subset of 3D LiDAR technology, offers notable advantages such as stable performance and lower costs than traditional multi-line LiDAR systems like the VLP-16 and VLP-32. However, its reliance on irregular scanning patterns and a narrower field of view can lead to motion blur. Solid-state LiDAR-based SLAM has become a significant area of research, with LOAM-Livox [9] being one of the most prominent examples. Leveraging the unique scanning approach and sensor characteristics of the Livox radar, the authors developed a SLAM system tailored for Livox, using LOAM [8] as a foundational reference. This system enhances performance by filtering out unsuitable points in the point cloud and extracting line and surface features. Pose estimation is performed iteratively by minimizing the residuals of line and surface distances.

2.4.2 LiDAR-Inertial Fusion

LiDAR-Inertial (LI) fusion leverages the complementary capabilities of LiDAR and IMU sensors to enhance the SLAM system's robustness, accuracy, and adaptability. By combining high-precision spatial data from LiDAR with high-frequency motion data from IMUs, LI fusion enables accurate state estimation even in challenging environments where single-sensor approaches might fail.

Loosely coupled LiDAR-inertial methods balance computational efficiency and motion estimation accuracy by independently processing LiDAR and IMU observations and then fusing their results. This modularity makes these approaches adaptable and easy to implement in real-time applications. Loosely coupled methods often incorporate IMU data to correct motion distortions, as

seen in IMU-aided LOAM, LeGO-LOAM, and similar methods. An exciting development by Kim et al. [33] is the adaptive keyframe approach, which refines loosely coupled designs by dynamically selecting keyframes based on the complexity of the scene and motion characteristics. This reduces computational load while maintaining accuracy by focusing map updates on critical moments. For example, fewer keyframes are chosen in feature-dense or high-motion environments, ensuring the system remains efficient. However, this strategy introduces potential limitations, such as increased temporal sparsity, which can degrade performance in rapidly changing scenarios if not carefully managed.

LiDAR-Inertial Observability-Aware Navigator(LION) [34] further innovates within this method by removing traditional feature extraction altogether. Instead, it relies on direct scan-to-scan matching and IMU preintegration, simplifying computations and making it particularly effective in feature-sparse environments like tunnels or degraded areas. Its use of a fixed-lag sliding window smoother allows for localized optimization, enabling robust performance in real-time. Nevertheless, this design also has drawbacks. The fixed-lag smoother limits optimization to a predefined temporal window, which can hinder the system's global consistency over long trajectories. Furthermore, while LION reliance on observability metrics to switch odometry sources improves adaptability, it may lead to information loss by ignoring specific state quantities, such as velocity, thus underutilizing the full potential of LiDAR and IMU integration.

Tightly coupled LiDAR-inertial methods integrate LiDAR and IMU data within a unified process, enabling enhanced performance in scenarios where either sensor alone may fail to provide accurate estimates. These methods are filter-based and graph-based, each with unique design considerations and associated trade-offs.

Filter-based tightly coupled methods leverage recursive estimation methods, such as Kalman filters, for fusing LiDAR and IMU data. Building upon this foundation, Wei et al. [35] employ an Iterated Extended Kalman Filter (IEKF), addressing computational inefficiencies by scaling the complexity with the state vector rather than the feature dimension. Fast-LIO2 [36] extends this with direct scan-to-map matching using an incremental k-d tree structure for map maintenance, significantly enhancing accuracy and efficiency and making it suitable for complex environments. However, the lack of explicit loop closure mechanisms limits its global consistency over extended trajectories. To further improve computational efficiency, *Faster-LIO* [37] introduces sparse incremental voxels to replace the k-d tree structure. This innovation supports high-speed operations, but its focus on computational speed degrades accuracy in large-scale deployments.

Graph-based tightly coupled methods take a global optimization perspective by leveraging factor graphs to model LiDAR and IMU data relationships. The authors of LeGO-LOAM [29] introduced its successor, LIO-SAM [38], integrating IMU-based methodologies. This system constructs LiDAR-inertial odometry using a factor graph method, incorporating multiple relative and absolute measurements, such as loop closures, as illustrated in Figure 6. A key innovation of LIO-SAM lies in marginalizing older pose and point cloud data, replacing the process of matching scans to global maps with local map matching. This approach significantly enhances real-time performance. Additionally, the system incorporates a GPS-based absolute positioning factor [62]

to correct long-term drift. Despite these advancements, the method's reliance on geometric features for extraction limits its effectiveness in open environments over extended periods.

2.4.3 LiDAR-Visual Fusion

LiDAR-Visual (LV) fusion algorithms combine the complementary strengths of visual and LiDAR sensors to mitigate the limitations of standalone systems. Cameras provide cost-effective solutions with rich visual features for tasks like loop closure detection, but they are sensitive to changes in lighting and lack robustness in texture-deficient environments. LiDAR sensors, on the other hand, excel at providing accurate depth information and perform well under diverse environmental conditions, though they face challenges in structureless scenarios such as long corridors. Integrating these modalities addresses their limitations, creating robust and accurate navigation systems.

Loosely coupled methods process visual and LiDAR data independently and fuse their outputs at a later stage to improve pose estimation and mapping. This modular design simplifies implementation and reduces computational demands, making it ideal for real-time applications. Zhang et al. [39] introduced DEMO, a loosely coupled method where LiDAR depth information complements visual odometry by mapping depth values onto visual features, creating a unified representation. However, this approach only partially utilizes LiDAR information, limiting its performance in sparse or unreliable feature environments. Similarly, Shin et al. [40] extended this concept by integrating LiDAR points as additional features within the Direct Sparse Odometry(DSO) method, enhancing robustness in texture-deficient environments. While these approaches improve performance, separating sensor processing can lead to inefficiencies, such as redundant computation or missed optimization opportunities. Learning-based methods further advance loosely coupled designs. Huang et al. [41] used deep learning to filter out dynamic features, enhancing robustness in cluttered environments. LIV-LAM [42] introduces unsupervised learning for object detection, leveraging detected objects as landmarks for mapping. While these innovations enhance adaptability, they also introduce challenges like increased computational demands and potential inaccuracies in object detection under extreme conditions.

Tightly coupled methods integrate visual and LiDAR data into a unified optimization method, enabling comprehensive sensor utilization for enhanced accuracy and robustness. The creators of LOAM advanced their algorithms by integrating monocular camera feature tracking with IMU in their work on Visual LiDAR Odometry and Mapping(V-LOAM) [43]. This integration linked the depth information of visual feature points with LiDAR point clouds, enabling the development of a visual-inertial odometry system to assist in LiDAR scan matching. Nevertheless, this approach's visual odometry (VO) is utilized solely to provide an initial pose estimation for the LiDAR odometry (LO). Since no full integration of vision and LiDAR is achieved, the final error calculation remains the same as in LOAM.

2.4.4 LiDAR-Inertial-Visual Fusion

LiDAR, inertial, and visual data fusion provide a comprehensive solution for addressing the limitations of standalone sensors. LiDAR excels in providing dense 3D geometry and robustness to environmental lighting variations but struggles in degenerate scenarios such as tunnels or wide-open spaces. IMUs complement this with high-frequency motion capture but suffer from long-term drift. On the other hand, visual sensors add rich texture and feature-based information, enabling loop closure detection and improving localization accuracy. By integrating these complementary modalities, LiDAR-Inertial-Visual(LIV) SLAM systems achieve superior performance in challenging environments.

Loosely coupled LIV SLAM architectures prioritize modularity by processing visual-inertial and LiDAR-inertial data streams independently and merging their results later in the pipeline. While this simplifies implementation and ensures flexibility, it does not fully exploit the interdependencies between the sensor modalities.

Zhang et al. [44] introduced an updated iterative version of [43], incorporating a sequential parallel processing method to refine motion estimation through a coarse-to-fine approach. This improved system leverages visual-inertial coupling to estimate motion more precisely, enhancing scan matching to achieve greater motion estimation and mapping accuracy. As a result, the system delivers high-frequency, low-latency motion estimation and generates dense, high-precision 3D map representations.

To harness the benefits of loosely coupled and highly coupled approaches, Zhao et al. [45] proposes Super Odometry that adopts an IMU-centric data processing method composed of three core components: IMU odometry, visual-inertial odometry, and LiDAR-inertial odometry. In this system, IMU bias is constrained using pose priors provided by the visual-inertial and LiDAR-inertial odometry modules, which, in turn, rely on motion predictions from the IMU odometry. Additionally, a dynamic octree structure is employed to maintain high real-time performance. The design is rooted in the principle that IMU odometry can achieve high accuracy when auxiliary sensors effectively constrain its bias drift, as the IMU provides smooth measurements with minimal noise and few outliers.

Tightly coupled LIV SLAM methods overcome the limitations of loose coupling by integrating LiDAR, visual, and inertial data into a unified optimization method. This approach enables mutual constraints between modalities, enhancing accuracy and robustness at the cost of higher computational demand. Shan et al.[46] introduced LVI-SAM, a publicly accessible system. Built on a factor graph method, LVISAM consists of two interconnected subsystems: the Visual-Inertial System (VIS) and the LiDAR-Inertial System (LIS). Unlike Super Odometry, LVISAM allows optional extraction of feature depth from LiDAR scans through a depth association technique. For loop closure, candidate matches are initially identified by the VIS and subsequently refined by the LI. The system leverages a factor graph to optimize all constraints simultaneously, including contributions from the VIS, LIS, IMU preintegration, and loop closure processes.

Building upon these advancements, Lin and Zhang [47] proposed R3LIVE, a tightly coupled

LiDAR-Inertial-Visual SLAM method that introduces a novel Visual-Inertial odometry (VIO) architecture. The system combines a LiDAR-Inertial odometry (LIO) subsystem, which reconstructs the geometric structure of the environment, with a VIO subsystem that renders the map's texture by minimizing photometric errors. This architecture achieves real-time reconstruction of dense, RGB-colored 3D maps while maintaining robustness and accuracy in degenerate environments. For instance, the system excels in scenarios with limited LiDAR geometry, such as featureless corridors, by leveraging RGB color information directly from visual data. Experimental evaluations demonstrated R3LIVE capacity for high-precision mapping in large-scale indoor and outdoor environments, with minimal drift even over trajectories exceeding 1.5 km.

Methods such as LVI-SAM and R3LIVE represent the state of the art in LIV SLAM, delivering exceptional accuracy and robustness. However, their high computational demands and dependence on precise multi-sensor calibration underscore areas for further refinement.

2.5 VineSLAM

VineSLAM [1][2] is an SLAM framework designed for the challenges of agricultural robotics, particularly in vineyards, where environmental symmetry, variable foliage, and unstructured terrain complicate localization and mapping. Introduced by Aguiar et al. [1], the system combines point and planar features from 3D LiDAR data to estimate a full six degrees-of-freedom (6-DoF) robot pose. It employs a novel Particle Filter (PF) approach that jointly evaluates point-level and semiplane-level observations, enabling the robot to localize itself using both dense geometric points and broader structural features, such as ground and vegetation planes.

The perception pipeline extracts edge and planar point features using a smoothness descriptor. At the same time, semiplanes—defined by their normal vector, centroid, and convex boundary—are computed using RANSAC for plane fitting, followed by Convex Hull extraction. These semiplanes capture large surfaces, such as the ground and lateral canopy walls, and are particularly valuable in environments lacking diverse structural landmarks. The mapping component of VineSLAM maintains a 3D voxel grid to register point features and implements an efficient semiplane merging strategy, ensuring continuous refinement of the map. Localization is achieved through a PF that estimates pose likelihood based on both the distance between corresponding point features and the alignment of observed and mapped semiplanes in terms of orientation and position.

Building on this foundation, a subsequent refinement proposed by the same authors [2] introduces a cluster-based PF enhancement that reduces computational overhead while improving localization accuracy. This extension integrates stereo camera data to generate dense, colored point clouds and applies scan matching within clustered particle groups. As a result, the SLAM system supports multi-layer mapping—metric, visual, and semantic—while maintaining real-time feasibility in high-dimensional state spaces.

Although developed for agricultural fields, the methodology behind VineSLAM is highly adaptable to structured industrial environments such as warehouses or manufacturing floors. The semiplane representation aligns well with the geometric regularity of such settings, where walls,

floors, and machinery form dominant planar structures. VineSLAM's semiplane merging and refinement algorithms could perform even more reliably in these environments due to their reduced variability and noise. Moreover, the PF-based localization, designed to tolerate ambiguities in natural environments, would offer robustness against occlusions and dynamic elements often present in industrial facilities, such as moving personnel or automated vehicles.

The system's modular and sensor-agnostic architecture also supports integration with a wide range of perception hardware commonly deployed in industry, including depth cameras, laser scanners, and stereo rigs. Its voxel grid-based map structure and scan-matching refinement further contribute to its scalability in large indoor spaces. While adjustments would be needed—for instance, adapting the feature extraction thresholds to account for the denser, more uniformly spaced features in factories—the core architecture of VineSLAM offers a promising blueprint for industrial-grade SLAM systems. Its fusion of global structural understanding with local geometric detail makes it well-suited to bridge the gap between outdoor field robotics and indoor automation.

2.6 Datasets and Benchmarking for SLAM Systems

The development of autonomous mobile robots and Unmanned Ground Vehicles (UGV) depends heavily on quality datasets to evaluate SLAM algorithms. Datasets like KITTI [48] have become a standard for testing algorithms in urban driving. They provide high-resolution images, Velodyne LiDAR scans, and precise GNSS/IMU data. This combination supports advances in visual odometry, 3D mapping, and object detection, making KITTI a critical tool for improving autonomous navigation systems.

For mobile robotics, datasets such as TUM RGB-D [49] and EuRoC MAV [50] offer a variety of scenarios. TUM RGB-D is focused on indoor environments and includes synchronized RGB-D sequences with accurate ground-truth poses. EuRoC MAV is geared more toward aerial and outdoor use, testing SLAM systems in dynamic and complex conditions. Together, these datasets help researchers evaluate performance across a range of real-world challenges.

That said, these datasets come with notable limitations. Many depend on external systems such as motion capture setups (VICON[1] and OptiTrack[2]) or GNSS/INS systems to provide ground-truth data [51], which restricts their use to either outdoor spaces with reliable GNSS reception or confined indoor environments where a motion capture system can be installed. They also often lack real-world complexities such as unpredictable lighting conditions, dynamic agents, and varied sensor configurations. In systems that integrate LiDAR, cameras, and IMU, synchronization errors between sensors can further degrade performance and complicate fair algorithm benchmarking.

To get around these issues, recent projects have introduced synthetic datasets built using robot simulators to overcome these issues. Wang et al. [51] created the *TartanAir* dataset using Unreal Engine and AirSim [52]. It features photorealistic scenes with challenging lighting, fast motion,

[1] https://www.vicon.com
[2] https://optitrack.com

and adverse weather conditions. The dataset provides stereo RGB images, depth maps, optical flow, and ground-truth trajectories—everything needed to benchmark visual odometry and feature-based SLAM systems. Similarly, Park et al. [53] introduced the *CSE dataset*, aimed at collaborative SLAM in settings such as hospitals and offices. Built with NVIDIA Isaac Sim[3], it supports multi-robot setups with dynamic human agents, offering synchronized stereo RGB-D, IMU, and ground-truth data in static and dynamic scenes.

Simulated environments offer several advantages beyond scalability and diversity. They enable precise sensor synchronization, reproducibility, and systematic control over environmental variables such as lighting, occlusion, terrain complexity, and sensor noise. This allows researchers to rigorously test algorithm robustness under ambiguous or adverse conditions. Moreover, simulations facilitate exploring SLAM functionalities often absent in real-world datasets, including loop closure, map merging, and multi-agent coordination.

Despite their strengths, these simulation-based datasets have practical limitations. Both TartanAir and CSE rely on resource-intensive or proprietary engines that demand specialized hardware and expertise, limiting accessibility and reproducibility. Furthermore, their focus on vision-centric pipelines makes it challenging to generalize findings to SLAM systems involving LiDAR or hybrid sensor fusion.

These constraints underscore the need for a more accessible, flexible, and general-purpose synthetic dataset generation framework—particularly one designed for indoor environments, multi-sensor configurations, and real-time adaptability. Such a tool would democratize benchmarking capabilities and enable the development and testing of next-generation SLAM solutions in industrial and human-centered domains.

2.7 Robot Simulators

Robot simulators play a crucial role in SLAM research, offering safe, controllable, and repeatable environments for algorithm development and synthetic dataset generation. As SLAM systems increasingly incorporate diverse sensors and operate in complex environments, the ability to simulate accurate perception, realistic dynamics, and flexible interactions becomes essential [54]. Simulation environments are particularly valuable during early development stages, allowing researchers to test control logic and perception pipelines before deployment on physical hardware. As robotic systems grow in complexity, simulators are routinely used to emulate the behavior and motion of 3D robots across various virtual environments [55].

A range of simulators has emerged to meet these demands, each with distinct priorities—from lightweight tools optimized for rapid iteration to photorealistic engines focused on high-fidelity rendering and accurate sensor emulation.

Among the more accessible simulators, *Webots* [56][4] stands out as a light-weight, open-source platform developed by Cyberbotics for modeling and controlling complex robotic systems in fully

[3] https://developer.nvidia.com/isaac/sim
[4] https://cyberbotics.com

simulated three-dimensional environments. It provides a rich suite of components, including sensors, actuators, and controllers, and supports physically accurate simulation of robot-environment interactions. Webots offers an intuitive interface, seamless integration with ROS and ROS 2, and built-in tools for trajectory logging, ground-truth extraction, and multi-sensor synchronization. These features make it particularly well-suited for indoor robotics applications involving AMRs and UGVs, where fast development cycles, structured environments, and moderate photorealism are sufficient for evaluating SLAM algorithms and generating synthetic datasets.

At the other end of the fidelity spectrum, *Isaac Sim*, NVIDIA's advanced platform built on Omniverse, offers photorealistic rendering, ray-traced lighting, and high-precision sensor models. Its features include domain randomization and dynamic scene control, making it particularly effective for sim-to-real transfer studies. However, its high computational demands and reliance on proprietary infrastructure limit accessibility and slow down development workflows.

Gazebo [57][5], widely adopted in the ROS community, provides a robust middle ground between fidelity and flexibility. It supports physics-based simulations with gravity, inertia, and collision modeling and extensive customization of environments, sensors, and robot models. While not as visually realistic as Unreal-based engines, Gazebo excels in geometry-based SLAM tasks and supports multi-robot scenarios. Its open-source nature, moderate hardware requirements, and strong ROS compatibility have made it a leading choice for robotics research. However, it offers limited support for appearance-based SLAM and presents a moderate learning curve. It can also be resource-intensive on lower-end hardware and presents a moderate learning curve for new users working with complex models or plugins.

Unreal Engine-based simulators such as *AirSim* [52] and *CARLA* [58] push the boundaries of visual realism and dynamic visual complexity. AirSim, initially developed for drones, now supports ground vehicles and provides comprehensive sensor suites, including LiDAR, IMU, and depth cameras. CARLA specializes in autonomous driving scenarios, simulating traffic agents, road networks, and photometric conditions. Both platforms are valuable for visual SLAM under challenging lighting, motion blur, or occlusion. However, they remain less suited to indoor robotics and are computationally demanding.

Table 2.1 summarizes key characteristics of these simulators, including platform support, ROS compatibility, licensing model, computational demand, and available sensors.

Nevertheless, despite the diversity of simulation tools, few platforms are designed to generate structured, annotated datasets tailored to SLAM in indoor environments. This gap motivated the core contribution of this dissertation: the design and implementation of a lightweight, ROS-compatible synthetic dataset generation framework purpose-built for mobile robots operating in indoor spaces.

[5] https://gazebosim.org

Table 2.1: Comparison of Robotic Simulators for SLAM and Dataset Generation

Simulator	Platforms Supported	ROS Support	Open Source	Computational Demand	SLAM-Relevant Sensors
Webots	Wheeled Robots, Manipulators, Drones	✓	✓	Low	RGB and Stereo Cameras, LiDAR, IMU, GPS, Encoders
Gazebo	Wheeled Robots, Manipulators, Drones	✓	✓	Medium	RGB and Depth and Stereo Cameras, LiDAR, IMU, GPS, Encoders
Isaac Sim	Wheeled Robots, Manipulators, Drones	✓	✓	Very High	RGB, Depth, and Stereo Cameras, RTX LiDAR, Radar, IMU, GPS
AirSim	Drones, Cars	Limited	✓	High	RGB, Depth Cameras, LiDAR, IMU, GPS
CARLA	Cars	✓	✓	High	RGB, Depth, Semantic Cameras, LiDAR, Radar, IMU, GPS

Chapter 3

Methodology

This chapter presents the design and implementation of a synthetic dataset and benchmarking framework developed using Webots and ROS 2, aimed at overcoming limitations in existing SLAM datasets. The primary objective is to provide a reproducible, extensible, and well-instrumented simulation environment for evaluating 3D SLAM algorithms under realistic yet controlled conditions—including dynamic obstacles, perceptual aliasing, and degraded motion scenarios.

The chapter begins by describing the modular simulation architecture used to generate synchronized multi-sensor datasets with ground-truth trajectories. It then details the mobile robotic platform, including its sensor suite and ROS 2 plugin integration, followed by the construction of challenging test scenarios designed to probe SLAM robustness. A standardized benchmarking framework is also introduced, outlining the evaluation metrics, protocols, and selected SLAM systems.

In addition, the chapter introduces a LiDAR odometry module adapted from the LeGO-LOAM [29] architecture. This module focuses solely on the LiDAR odometry pipeline, adding an optional IMU integration, and is designed to provide a better motion estimation front-end for the VineSLAM system. Its development was motivated by performance bottlenecks observed during evaluation, aiming to improve pose tracking in feature-sparse or dynamic environments. The module's design, implementation, and integration are described in detail, forming the basis for the performance analysis presented in Chapter 4.

The chapter is organized as follows. Section 3.1 describes the dataset generation process, including robot and sensor modeling and scenario setup. Section 3.2 outlines the benchmarking methodology. Section 3.3 details the LiDAR odometry module. Section 3.4 explains the designed IMU integration to LiDAR odometry.

3.1 Synthetic Dataset Generation

Designing an accessible, flexible, and general-purpose dataset generation framework requires supporting the creation of diverse environments, various sensor configurations, and reproducible experimental processes. Among the platforms evaluated in Section 2.7, *Webots* was selected as the

simulation foundation due to its lightweight architecture, native integration with ROS and ROS 2, broad sensor support, and open-source licensing [56]. Compared to more resource-intensive alternatives such as Isaac Sim and Unreal Engine-based tools, it provides a favorable trade-off between visual realism and computational efficiency, making it suitable for a wide range of SLAM scenarios. Although *Gazebo* offers similar capabilities—such as realistic physics, ROS compatibility, and extensible world-building—its steeper learning curve and higher computational demands can hinder rapid prototyping and broad accessibility. In contrast, Webots offers a more approachable and efficient development environment while retaining sufficient fidelity for SLAM benchmarking across indoor and outdoor settings.

To further enhance usability and portability, the entire framework was containerized using Docker [1] (Figure 3.1), allowing seamless deployment across different computing environments. This decision facilitates reproducibility, reduces system configuration overhead, and enables consistent experimentation across platforms. The complete simulation stack, along with example worlds and launch files, is publicly available at the *Agrob4Simulator* [2] repository.

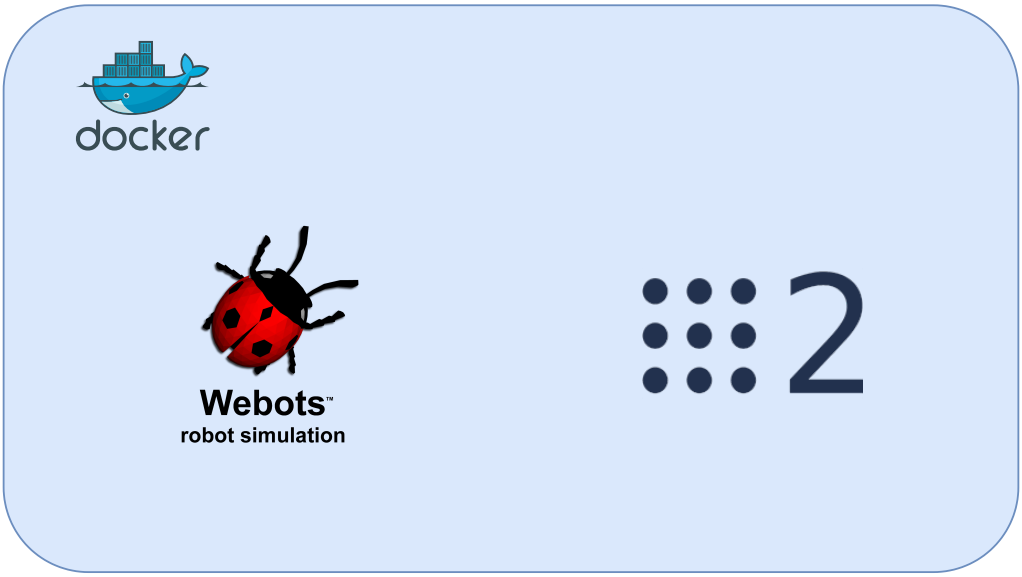


Figure 3.1: Webots-ROS 2 Container.

The framework extends the official `webots_ros2`[3] interface with custom launch configurations, modular world definitions, and a standardized output format tailored to SLAM benchmarking. We generated each dataset by simulating a mobile robot navigating in a baseline environment derived from a 3D reconstruction of a iiLAB facility. This environment was augmented with various perturbations to evaluate SLAM robustness, including dynamic obstacles, symmetric layouts, inclined surfaces, and low-friction zones—conditions that challenge localization, data association, and loop closure in real-world indoor deployments.

[1] https://docs.docker.com
[2] https://github.com/thiago-b-levin/Agrob4Simulation
[3] https://github.com/cyberbotics/webots_ros2

3.1.1 Simulation Framework: Webots and ROS 2 Integration

The simulation architecture leverages Webots and ROS 2 to enable bidirectional real-time communication of control commands, synthetic sensor data, and ground-truth information. A schematic overview is presented in Figure 3.2.

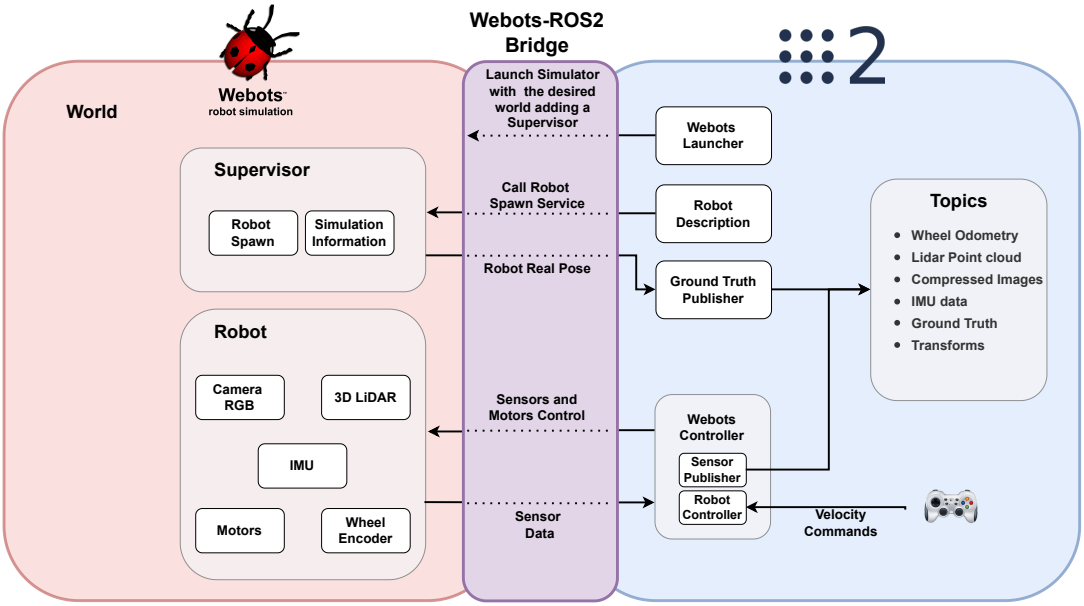


Figure 3.2: Overview of the Webots-ROS 2 integration framework for synthetic dataset generation. A mobile robot equipped with multiple sensors is spawned through a ROS 2 service. Synthetic sensor data are published to ROS 2 topics, while ground-truth information is continuously retrieved from the Webots Supervisor and published.

The simulation proceeds as follows:

- **Webots Launcher:** A ROS 2 launch pipeline initializes the Webots simulator with a specified world and instantiates a *Supervisor Node*. The Supervisor manages simulation timing and provides access to internal simulation states.

- **Robot Spawning:** The mobile robot is dynamically instantiated using a ROS 2 service. The robot's configuration—including kinematic structure, sensor suite, and control interfaces—is defined in a Universal Robot Description File (URDF), which is internally converted to a Webots-compatible `PROTO` file.

- **Ground-Truth Publisher:** A dedicated *Ground-Truth Supervisor Node* continuously queries the simulation environment for the true six degrees-of-freedom (6-DoF) pose of the robot. This node publishes ground-truth data using standard `nav_msgs/Odometry` messages and is also broadcast as a transform between the `base_link` and `world`. Ground-truth information is inherent and is employed for benchmarking and evaluation.

- **Webots Controller Node:** Manages the bidirectional communication interface between the simulator and ROS 2. This node hosts sensor publisher and actuator control plugins, ensuring real-time feedback and closed-loop control.

  - **Sensor Publishers:** These plugins extract data from the simulation (e.g., LiDAR, RGB camera, IMU, and encoders) and publish them to ROS 2 topics using standard message types such as `sensor_msgs/PointCloud2` and `sensor_msgs/Imu`.
  - **Robot Controller:** This plugin receives actuation commands, sent as `geometry_msgs/Twist` messages, that are interpreted and translated into individual wheel velocities applied via Webots motor API.

The simulation operates at the standard time step of 32 ms (approximately 31.25 Hz), synchronized with the ROS 2 simulation clock to maintain temporal consistency across sensors and control channels. Sensor data and ground-truth information were published to dedicated ROS 2 topics to ensure modularity, interoperability, and reproducibility of the synthetic datasets. Table 3.1 summarizes the devices or publishers responsible for each topic, along with the corresponding message types used during data acquisition.

Table 3.1: Published Topics and Message Types

Device	**Topic Name**	**Message Type**
3D LiDAR	`/scan`	`sensor_msgs/PointCloud2`
RGB Camera	`/camera/compressed`	`sensor_msgs/CompressedImage`
IMU	`/imu/data`	`sensor_msgs/Imu`
Wheel Encoders	`/odom`	`nav_msgs/Odometry`
Ground-Truth Supervisor	`/ground_truth/odom`	`nav_msgs/Odometry`
—	`/tf`	`tf2_msgs/TFMessage`

Standardized ROS 2 message types facilitate compatibility with a wide range of SLAM and perception algorithms. At the same time, explicit topic segregation enables independent processing of noisy sensor data and ground truth measurements.

3.1.2 Mobile Robotic Platform

A model of a real mobile robot, designated *Modular E* (Fig. 3.3), was used for dataset generation. The platform was designed by the TRIBE Lab—INESC TEC Laboratory of Robotics and Internet-of-Things (IoT) for Smart Precision Agriculture and Forestry. It adopts a two-module chassis architecture, employing front-wheel differential drive for locomotion and passive rear wheels for stability. For this study, the complete mechanical, inertial, and sensor configurations were described in the URDF. Upon simulation launch, this file was used to spawn the robot, configure its

sensors, and establish the control interfaces through ROS 2 and Webots plugins, enabling reproducible and systematic benchmarking of SLAM algorithms under indoor conditions and synthetic dataset generation.

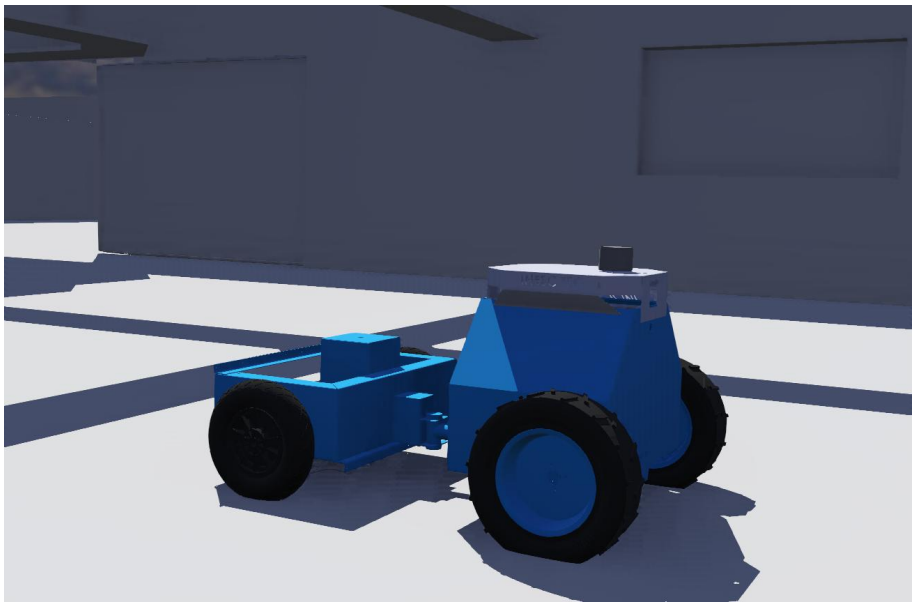


Figure 3.3: Rendered model of the Modular E robot used in simulation. The platform features a front-mounted sensor suite and a differential drive architecture.

The robot was operated using a Logitech F710 wireless joystick, with user inputs translated into velocity commands published on the `/cmd_vel` topic. A custom plugin driver interpreted these commands and employed a differential drive controller to compute left and right wheel velocities, accounting for physical parameters such as wheelbase and wheel radius.

3.1.3 Sensor Suite Configuration

The Modular E robot was equipped with a multi-modal sensor suite designed to support a variety of SLAM and perception algorithms. The suite includes a 3D LiDAR, RGB camera, inertial measurement unit (IMU), and wheel encoders. Each sensor was interfaced via custom ROS 2 plugins that ensured compatibility with standard message types and topic structures.

3D LiDAR

Mounted a simulated Velodyne VLP-16[4] (Figure 3.4), which is a 3D LiDAR sensor from Ouster, widely adopted in various solutions. This model provides a 360° horizontal field of view and a vertical field of view of approximately 40°, distributed across 16 channels. With a maximum data rate of 300,000 points per second, it is well-suited for generating dense point clouds in dynamic scenarios.

[4] https://ouster.com/products/hardware/vlp-16

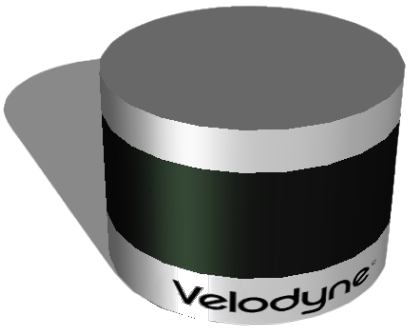


Figure 3.4: Velodyne VLP-16 Webots Model.

The sensor was selected for its robustness, extensive ROS ecosystem support, and availability within simulation environments. To interface the simulated Velodyne VLP-16 LiDAR with ROS 2, a developed custom plugin is part of the implemented Webots sensor integration layer. The plugin derives from the base `Ros2SensorPlugin` class provided by the `webots_ros2_driver` framework. It manages sensor initialization, data acquisition, message formatting, and the publication of 3D point cloud data in a ROS-compliant format.

During operation, the plugin retrieves raw LiDAR data from the Webots API and constructs `sensor_msgs/msg/PointCloud2` messages. Using `PointCloud2Iterator`s, it populates the `x`, `y`, `z`, and `intensity` fields for each point in the cloud. Since Webots does not inherently provide intensity values, these are synthetically computed based on the normalized distance of each point relative to the sensor's maximum range, producing higher intensity for closer points and lower for distant ones. These messages are published at 10 Hz—matching the real VLP-16 scan rate —on the topic `/scan`, enabling seamless integration into ROS 2-based SLAM and perception pipelines with realistic sensor behavior.

RGB Camera

Next to the LiDAR, the robot features a forward-facing standard RGB camera configured with a resolution of 800 $\times$ 800 pixels, a horizontal field of view of approximately 80° (1.396 radians), and a near-to-far clipping range between 0.02 m and 300 m. Gaussian noise is applied with zero mean and a standard deviation of 0.007. The camera operates at a frame rate of 30 Hz, providing a sufficient temporal resolution for tasks such as visual odometry, scene segmentation, and object recognition.

To interface the camera with ROS 2, a custom plugin named `camera_publisher` was developed. This plugin derives from `Ros2SensorPlugin` and handles both raw and compressed image streams. At each simulation step, the plugin acquires the raw BGRA image buffer from Webots, copies it into a `sensor_msgs/msg/Image` message, and publishes it on the topic

`/camera/image_raw`. Simultaneously, it converts the image to BGR format using OpenCV, encodes it as JPEG, and publishes it as a `sensor_msgs/msg/CompressedImage` on the topic `/camera/compressed`. This compression significantly reduces storage requirements—shrinking datasets from approximately 20 GB (raw) to 2–3 GB (compressed)—while maintaining visual quality sufficient for downstream SLAM and perception tasks.

Inertial Measurement Unit (IMU)

The IMU sensor used in this setup provides linear acceleration, angular velocity, and orientation data, all of which are essential for motion estimation, sensor fusion, and pose correction in SLAM systems. It is simulated using three distinct Webots components: an accelerometer, a gyroscope, and an inertial unit. These are combined and interfaced with ROS 2 through a native plugin provided by the official `webots_ros2_driver` package.

Unlike the LiDAR and camera plugins, this IMU plugin was not custom-developed. Instead, it is implemented as the `Ros2IMU` class and internally manages the initialization, enabling, and data acquisition of each individual sensor. It aggregates the sensor data into a single `sensor_msgs/msg/Imu` message, including quaternion-based orientation from the inertial unit, angular velocity from the gyroscope, and linear acceleration from the accelerometer. This message is published at each simulation step using a sensor-data QoS profile on a configurable topic (typically `/imu/data`). The plugin activates only when subscribed to, ensuring efficient resource use during simulation.

Wheel Encoders

The robot's odometry is estimated using two wheel encoders mounted on the front wheels, which measure the accumulated rotation in radians since the beginning of the simulation. A custom ROS 2 plugin, was developed to process these encoder readings and publish odometry data using the `nav_msgs/msg/Odometry` message type. The plugin calculates the robot's pose and velocity using a differential drive kinematic model based on the wheel radius r and the distance between the wheels (wheel base) b.

At each simulation step, the plugin retrieves the angular positions of the left and right wheels, computes their angular velocities, and derives the linear and angular velocities of the robot:

$$v = \frac{r}{2}(v_r + v_l), \qquad \omega = \frac{r}{b}(v_r - v_l), \tag{3.1}$$

where v_r and v_l are the angular velocities of the right and left wheels, respectively. The robot's position (x, y) and orientation θ are then updated by integrating over time:

$$x \leftarrow x + v\cos(\theta)\cdot\Delta t, \qquad y \leftarrow y + v\sin(\theta)\cdot\Delta t, \qquad \theta \leftarrow \theta + \omega\cdot\Delta t \tag{3.2}$$

This estimated pose is published on the `/odom` topic and is also broadcast as a transform between the `odom` and `base_link`.

3.1.4 SLAM Evaluation Dataset Collection

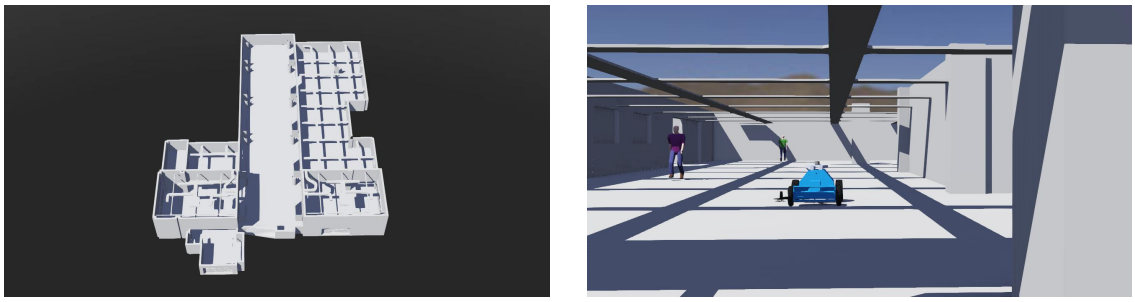
Datasets were collected by recording ROS 2 bags that captured all relevant topics, including LiDAR point clouds, RGB images, IMU measurements, wheel encoder odometry, and ground-truth trajectories. Each bag was timestamped using the synchronized simulation clock to ensure temporal consistency across testing. The sequences created are briefly described in Table 3.2.

Table 3.2: Summary of Synthetic Sequences for SLAM Evaluation

Sequence	**Description**	**Duration (s)**	**Distance (m)**	**Laps**
Static	Baseline sequence capturing a typical environment without additional challenges (see Fig. 3.5a).	705.77	530	2
Dynamic	Introduces walking people in the environment to evaluate the algorithm's robustness to dynamic objects (see Fig. 3.5b).	765.72	567	2
Ramp 15 °	Features a 15 ° ramp that the robot ascends and descends during the sequence, introducing non-planar motion with vertical displacement and pitch rotation (see Fig. 3.6a).	713.49	540	2
Ramp 30 °	Includes a 30 ° ramp. The robot attempts to ascend but fails, causing wheel slippage—a challenge for odometry-reliant SLAM methods (see Fig. 3.6b).	802.21	583	2
Symmetric Corridor	Two walls are added to the central corridor through which the robot navigates, creating a low-feature, symmetrical environment that challenges SLAM performance (see Fig. 3.7a).	316.15	237	2
Slippery Zone	Introduces a low-friction zone that causes wheel slippage, challenging SLAM systems that depend on wheel odometry (see Fig. 3.7b).	788.54	589	2

For the simulation environment, a 3D model (Fig. 3.5a), based on the iiLab – industry and innovation Lab[5], indoor facilities that simulate an industrial environment and are used to test solutions. This simulation environment was progressively modified to systematically introduce a series of scenarios designed to stress the critical components of SLAM pipelines. The baseline sequence was recorded in this unmodified environment. This setting serves as a control case to establish reference SLAM performance under ideal conditions. To test robustness to dynamic changes, three human agents in different locations were inserted into the scene using animated pedestrian models available in Webots. These agents followed predefined paths, creating intermittent occlusion and feature tracking interference (Fig. 3.5b).

[5] https://www.inesctec.pt/en/laboratories/iilab-industry-and-innovation-lab

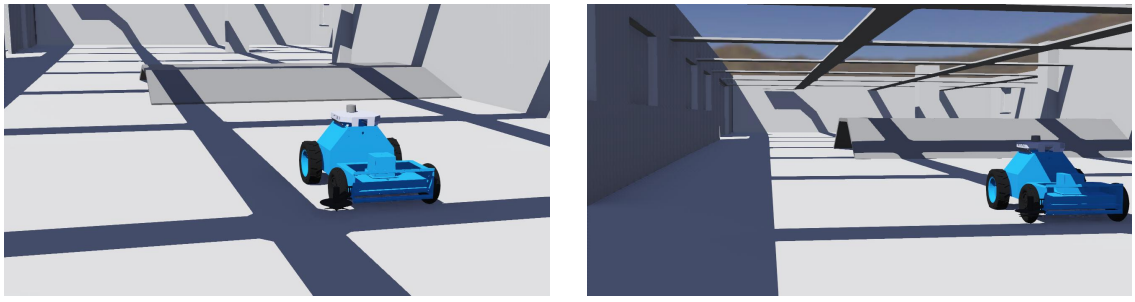


(a) Static iiLab environment.

(b) Dynamic agents in simulation.

Figure 3.5: Initial baseline and dynamic variation of the iiLab environment.

To introduce vertical motion, a 15-degree ramp was added as a sloped plane within the simulation world and placed on the main navigation path of the robot. The robot ascended and descended this ramp during the sequence (Fig. 3.6a), which challenged the vertical motion estimation and SLAM modules that rely on planar assumptions. To simulate a situation where the robot's wheels keep moving but it can't advance, an additional configuration raised the difficulty by replacing the ramp with a 30-degree slope. In this case, the robot failed to climb fully, resulting in wheel slippage due to the steep incline, which challenges wheel-odometry-based solutions the most (Fig. 3.6b).

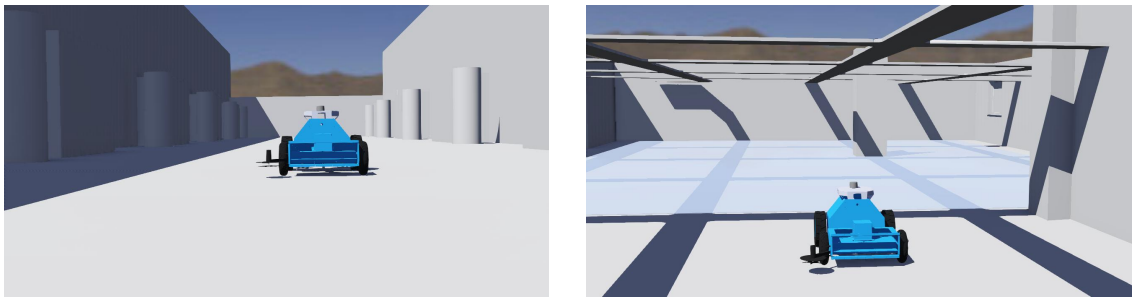


(a) 15° ramp incline.

(b) Failed climb at 30° slope.

Figure 3.6: Scenarios involving inclined surfaces and vertical motion.

Another variation focused on perceptual aliasing: two symmetric walls were introduced into a central corridor to create a visually ambiguous environment (Fig. 3.7a). The geometry and textures of both walls were intentionally mirrored to challenge loop closure mechanisms and feature-based localization, which often rely on unique appearance cues. Lastly, a low-traction region was defined by altering the surface friction coefficient of the floor material within a specific zone. The robot's differential-drive system experienced frequent slippage when traversing this area (Fig. 3.7b), making this sequence particularly challenging for odometry-dependent SLAM pipelines.



(a) Symmetric corridor geometry.

(b) Low-traction surface zone.

Figure 3.7: Scenarios introducing perceptual ambiguity and slippage.

The resulting publicly available dataset collection [3] enables reproducible performance evaluation of SLAM algorithms under controlled yet diverse conditions. Each sequence is temporally aligned and synchronized with a simulation clock to facilitate accurate benchmarking.

3.2 Benchmarking Framework

Quantitative evaluation of SLAM algorithms is essential to objectively compare performance across different systems and environments [59], [60]. To this end, a benchmarking framework was established using synthetic datasets generated in controlled simulation scenarios. A set of trajectory-based error metrics was employed to assess the accuracy and drift characteristics of the estimated poses. State-of-the-art 3D SLAM algorithms were selected for evaluation, and a consistent benchmarking protocol was applied to ensure fair and reproducible comparisons.

3.2.1 Evaluation Metrics

To quantitatively assess SLAM performance, a set of standard trajectory-based error metrics was adopted. These metrics provide complementary insights into the global and local accuracy of the estimated trajectories with respect to the ground truth.

Absolute Trajectory Error (ATE)

The Absolute Trajectory Error measures the global consistency of the estimated trajectory after alignment with the ground truth. It is defined as the root mean squared error (RMSE) of the translational differences between corresponding poses:

$$ATE_{rmse} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{p}_i^{est,aligned} - \mathbf{p}_i^{gt}\right\|^2}, \tag{3.3}$$

where:

- N is the number of synchronized pose pairs,

- $\mathbf{p}_i^{gt}$ is the ground truth position at timestamp i,
- $\mathbf{p}_i^{est,aligned}$ is the aligned estimated position at timestamp i.

Relative Translation Error (RTE)

The Relative Translation Error captures local drift by comparing relative translations between pairs of poses separated by a fixed distance Δ (e.g., 10 meters). It is computed as:

$$RTE = \frac{1}{M} \sum_{(i,j)\in\mathscr{P}} \left\| \left(\mathbf{p}_j^{rel,est} - \mathbf{p}_j^{rel,gt} \right) \right\|, \tag{3.4}$$

where:

- $\mathscr{P}$ is the set of pose index pairs (i, j) such that the ground truth distance between them is approximately Δ,
- $\mathbf{p}_j^{rel,gt}$ and $\mathbf{p}_j^{rel,est}$ are the ground truth and estimated relative translations between poses i and j,
- M is the number of valid pose pairs.

In this work, the RTE is further normalized by Δ and reported as a percentage error over the traveled distance.

Relative Rotation Error (RRE)

The Relative Rotation Error quantifies local angular drift between corresponding relative rotations. It is computed as:

$$RRE = \frac{1}{M} \sum_{(i,j)\in\mathscr{P}} \frac{\theta(R_{ij}^{est} R_{ij}^{gt^{-1}})}{\Delta}, \tag{3.5}$$

where:

- R_{ij}^{gt} and R_{ij}^{est} are the relative rotation matrices between poses i and j for ground truth and estimated trajectories, respectively,
- $\theta(\cdot)$ denotes the rotation angle in degrees of the composed rotation,
- Δ is the distance interval over which the rotation is measured,
- M is the number of valid pose pairs.

Together, these metrics provide a comprehensive evaluation of SLAM system performance. ATE reflects the overall alignment error and is especially sensitive to loop closure and large-scale drift. In contrast, RTE and RRE offer localized views of odometric stability and are particularly useful for analyzing short-term errors in environments with symmetry, noise, or degraded motion estimation. To ensure fairness, all metrics were computed using fixed segment lengths and consistent parameters across all systems and sequences.

3.2.2 SLAM Algorithms Benchmarked

To establish a comprehensive baseline for assessing 3D SLAM performance, three representative algorithms were evaluated on synthetic datasets. These systems reflect a spectrum of design philosophies, including geometric registration, feature-based methods, and hybrid probabilistic models. The benchmarking focuses on quantifying the strengths and limitations of each approach under standardized conditions and provides the foundation for extending and adapting VineSLAM [1], [2] introduced in Section 2.5,to industrial applications characterized by structural regularity, sparse texture, and sensor noise.

KISS-ICP [61] A geometry-driven LiDAR odometry system that revives classical point-to-point Iterative Closest Point (ICP) methods using a reduced-complexity design. KISS-ICP applies adaptive thresholding for correspondence rejection, voxel-based downsampling, and robust kernel optimization. Motion prediction is performed via a constant velocity model, enabling scan deskewing without inertial measurements. The algorithm generalizes across a broad range of LiDAR sensors and motion profiles, achieving high temporal efficiency and competitive accuracy. Notably, it excludes loop closure and global map refinement, prioritizing simplicity, generalizability, and real-time capability.

LEGO-LOAM [29] A real-time SLAM framework optimized for ground-based operation on uneven terrain. LEGO-LOAM introduces ground-aware point cloud segmentation and extracts curvature-based edge and planar features from structured subregions of the scan. A two-stage Levenberg-Marquardt optimization first estimates roll, pitch, and vertical translation from ground-plane correspondences, followed by refinement of the full 6-DoF pose using edge features. Loop closure is supported, although full graph-based backend optimization is not implemented. The system balances computational efficiency and localization robustness, particularly in semi-structured environments with prominent planar geometry.

3.2.3 Benchmarking Protocol

All SLAM systems were evaluated under identical synthetic simulation conditions to ensure consistency and reproducibility. The benchmarking protocol involved the following steps:

- Uniform initialization for all sequences to eliminate bias from initial conditions.
- Standardized sensor noise models and motion profiles across all datasets.
- Trajectory alignment using Umeyama's method prior to metric computation [62].
- Metric computation standardized using publicly available evaluation tools [60].

All trajectory evaluations were performed using the IILABS 3D Toolkit[6], which provides standardized utilities to evaluate SLAM trajectories against ground truth, including frame correction and accuracy metric computation based on the `evo`[7] library.

This benchmarking protocol ensures that the comparative analysis is both fair and representative of algorithmic performance under controlled and repeatable synthetic conditions.

3.3 LiDAR Odometry Module

To address the performance bottlenecks observed in VineSLAM during benchmarking (see Chapter 4), a dedicated LiDAR odometry front-end was developed. This module is adapted from the odometry subsystem of LeGO-LOAM [29], a widely used geometric SLAM framework that builds on the original LOAM architecture [8].

While LeGO-LOAM was originally designed as a full odometry and mapping system, in this work only the odometry component is retained. The mapping back-end was completely removed, and the remaining codebase was refactored and fully documented to facilitate integration and maintainability within the VineSLAM stack. As shown in Figure 3.8, the odometry pipeline follows four main stages: (1) image projection, (2) feature extraction, (3) feature association, and (4) transform integration. These stages provide a balance between computational efficiency and motion estimation precision.

Each LiDAR scan is projected into a 2D range image, preserving the angular structure of the point cloud while enabling efficient spatial processing. From this image, edge and planar features are extracted based on local curvature metrics. Feature association is performed across consecutive scans using segmentation-aware matching to establish geometric correspondences. Finally, a two-step Levenberg–Marquardt optimization estimates the 6-DOF transform between scans, decoupling orientation from translation to improve convergence.

The resulting pose estimate is published at 10 Hz and replaces wheel odometry in the motion prediction step of VineSLAM's particle filter. Unlike wheel encoders, which are prone to drift and slippage, LiDAR odometry provides geometry-consistent motion estimates that are more robust in both agricultural and indoor industrial environments.

While the core processing structure follows LeGO-LOAM, this implementation introduces several practical adaptations. In addition to removing the mapping subsystem, the code was thoroughly documented to ease integration. Furthermore, as detailed in the next section, a novel IMU integration scheme was added to enhance motion compensation and pose stability under fast or abrupt movement—an extension not present in the original LeGO-LOAM.

[6] https://github.com/JorgeDFR/iilabs3d-toolkit
[7] github.com/MichaelGrupp/evo

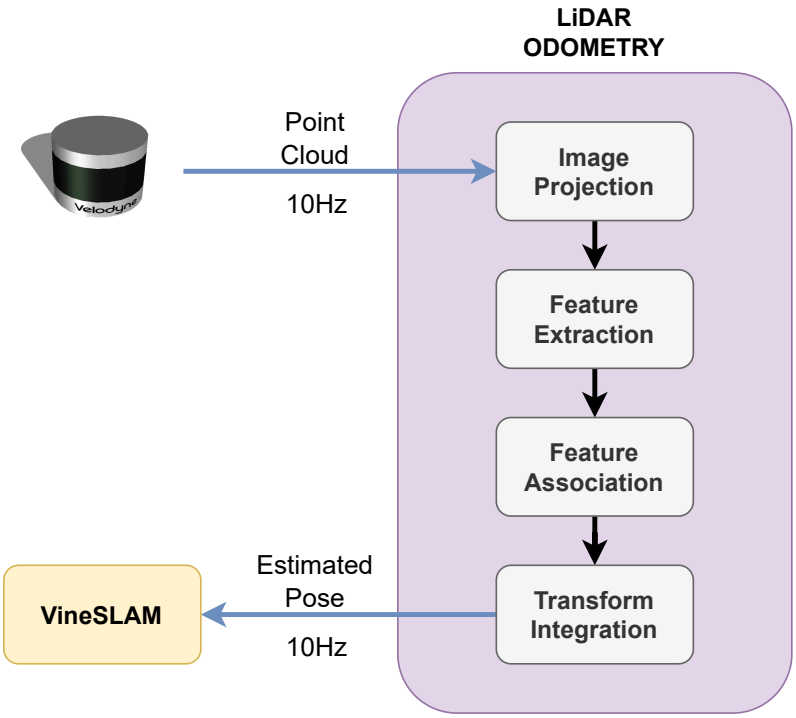


Figure 3.8: Overview of the LiDAR Odometry Module, showing the four-stage process: image projection, feature extraction, feature association, and transform integration.

3.4 LiDAR Odometry IMU Integration

In LiDAR-based odometry systems such as LeGO-LOAM, a major challenge lies in correcting motion distortion that occurs during LiDAR scan acquisition. Unlike cameras, which typically capture frames nearly instantaneously, LiDAR sensors accumulate points sequentially over tens of milliseconds as the sensor rotates. If the platform is in motion—especially undergoing rotation—this results in a geometrically distorted point cloud, which compromises the accuracy of feature extraction and scan matching. To address this, angular velocity measurements from an IMU (Figure 3.9) are integrated to estimate the platform's rotation during a scan and correct for these distortions in a process known as *deskewing*.

Each incoming IMU message provides angular velocity $\omega_{\text{ros}} \in \mathbb{R}^3$ in the ROS coordinate frame. To align with the LeGO-LOAM internal coordinate system, these values are transformed using a fixed rotation matrix corresponding to sequential 90-degree rotations about the Z and X axes:

$$\mathbf{R}_{\text{ros}\to\text{lego}} = \mathbf{R}_x\left(\frac{\pi}{2}\right) \cdot \mathbf{R}_z\left(\frac{\pi}{2}\right), \tag{3.6}$$

$$\omega_{\text{lego}} = \mathbf{R}_{\text{ros}\to\text{lego}} \cdot \omega_{\text{ros}}, \tag{3.7}$$

where $\mathbf{R}_x(\cdot)$ and $\mathbf{R}_z(\cdot)$ are standard rotation matrices around the X and Z axes, respectively.

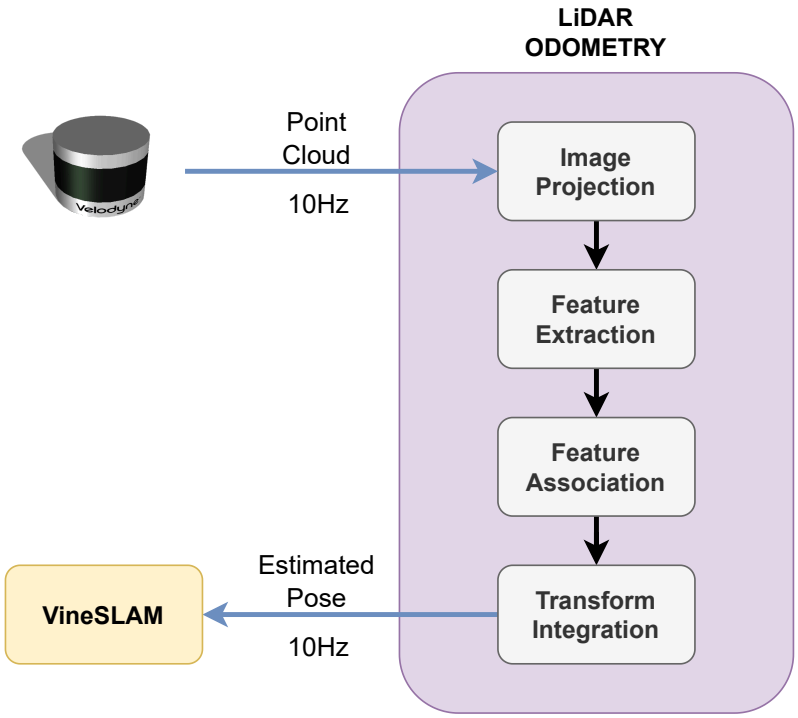


Figure 3.9: Overview of the LiDAR-Inertial Odometry Module.

These transformed angular velocities are then integrated over time to yield a rolling estimate of the platform's orientation in roll, pitch, and yaw (RPY) form. Assuming sufficiently small angular increments between IMU samples, the integration is approximated as:

$$\begin{aligned} \theta_{\text{roll}}(t+\Delta t) &= \theta_{\text{roll}}(t) + \omega_x \cdot \Delta t, \\ \theta_{\text{pitch}}(t+\Delta t) &= \theta_{\text{pitch}}(t) + \omega_y \cdot \Delta t, \\ \theta_{\text{yaw}}(t+\Delta t) &= \theta_{\text{yaw}}(t) + \omega_z \cdot \Delta t. \end{aligned} \tag{3.8}$$

The resulting RPY orientation is then converted into a rotation matrix via Z–Y–X Euler composition. Each orientation matrix is timestamped and stored in a rotation queue that maintains a temporal history over recent measurements.

When a new LiDAR scan is received, the system resets its internal buffers and begins processing the raw point cloud. If IMU integration is enabled via the `use_imu` parameter, the most recent orientation estimate is used to populate the fields `imu_roll_init`, `imu_pitch_init`, and `imu_yaw_init` in the customized outgoing `CloudInfo` message. These values provide two essential functions: (1) they serve as an initial guess for the scan-to-scan transformation optimization in the `FeatureAssociation` module, and (2) they establish a reference frame for deskewing individual LiDAR points.

Deskewing is performed by rotating each point according to its precise acquisition time within the scan. Let t_0 denote the scan start time and t_p the time at which a point p was captured. The corrected position of the point is computed as:

$$\mathbf{p}_{\text{corrected}} = \mathbf{R}^{-1}(t_p) \cdot \mathbf{R}(t_0) \cdot \mathbf{p}_{\text{raw}}, \tag{3.9}$$

where $\mathbf{R}(t)$ is the IMU-derived rotation matrix at time t. This transformation removes the accumulated rotational distortion between t_p and t_0, aligning all points to a common reference frame.

The inclusion of the `use_imu` parameter allows the system to dynamically enable or disable this correction. If disabled, the pipeline works as explained in Section 3.8.

Overall, IMU should improve the geometric consistency of LiDAR scans, enhance the convergence rate of non-linear odometry estimation, and boosts system robustness under aggressive motion or degraded feature environments. Although the method corrects only for rotational motion—and is subject to IMU drift and sensor bias over time—it offers a computationally efficient and practically effective strategy for front-end LiDAR odometry in robotics applications.

Chapter 4

Experimental Results

This chapter presents the experimental validation of the tools, methodologies, and SLAM enhancements developed throughout this dissertation. Aligned with the overarching goal of improving the repeatability and robustness of VineSLAM in industrial environments, the results focus on three core contributions: benchmarking state-of-the-art SLAM systems, evaluating VineSLAM against them, and identifying and integrating improvements to its pipeline. All experiments were conducted using the synthetic dataset and benchmarking framework introduced in Chapter 3.

The chapter is structured as follows. Section 4.1 presents the benchmarking results for all evaluated SLAM systems, providing both analysis and discussion. Section 4.2 examines the influence of different motion estimation strategies by comparing wheel-based, LiDAR-based, and LiDAR-Inertial odometry under identical conditions. Together, these results form the basis for the conclusions drawn in the final chapter.

4.1 SLAM Benchmarking

This section presents a comparative evaluation of SLAM systems under diverse conditions using the benchmarking methodology established earlier. The focus is on analyzing performance across six synthetic sequences, each crafted to isolate specific challenges commonly encountered in indoor robotics—such as dynamic obstacles, ambiguous environments, and degraded traction.

Performance is assessed using standard trajectory-based metrics: Absolute Trajectory Error (ATE), Relative Translational Error (RTE), and Relative Rotational Error (RRE). Qualitative assessments of the reconstructed maps complement these to highlight structural accuracy and consistency.

The results aim to contextualize VineSLAM's behavior alongside established SLAM algorithms and expose the strengths and limitations of each system when deployed in controlled but realistic scenarios. This study produced a large number of trajectory and map reconstructions. These outputs serve as the basis for the visual evaluations presented in the following sections. As an illustration, Figure 4.1 displays selected trajectory plots from the 30-degree ramp and slippery zone sequences, along with reconstructed maps from the slippery zone. These cases were

selected because they exhibited the most visually distinctive features among the SLAM systems under evaluation.

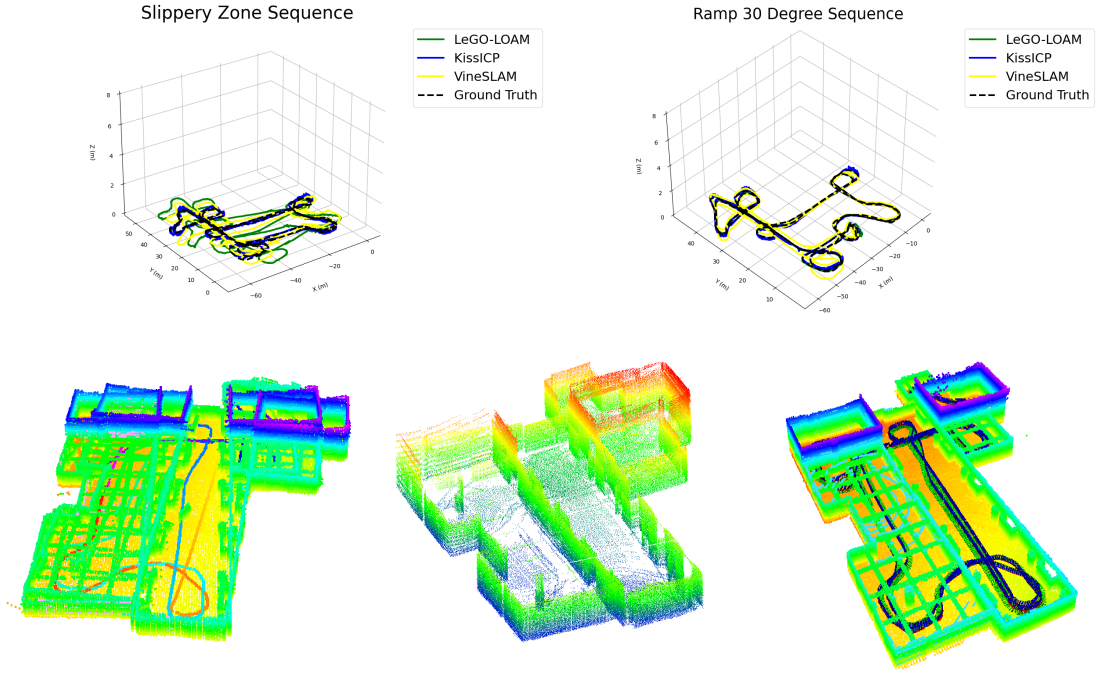


Figure 4.1: Representative outputs used in the evaluation. Top: estimated trajectories in the slippery zone (left) and 30° ramp (right) sequences. Bottom: map reconstructions in the slippery zone by LeGO-LOAM (left), VineSLAM (center), and KISS-ICP (right). These sequences were chosen for their clear visual differences across SLAM systems.

Quantitative Evaluation

The comparative results highlight distinct behavioral patterns across the evaluated SLAM systems, shaped by their design principles and adaptation to varying environmental challenges. KISS-ICP achieves the lowest trajectory error in static and dynamic scenes. It benefits from its lightweight, scan-to-map ICP design that emphasizes local geometric alignment. However, its purely odometric nature leads to significant degradation in environments with ambiguous or repetitive geometry, such as the symmetric corridor, where it accumulates drift due to the absence of any global correction.

LeGO-LOAM maintains moderate accuracy across structured and semi-structured settings by leveraging LiDAR-specific edge and planar features. It performs reliably on flat and moderately dynamic sequences but exhibits instability on the steep ramp and slippery trajectories. These failures correspond to a breakdown in ground segmentation and feature extraction under non-ideal surface conditions, revealing sensitivity to vertical motion and terrain irregularities.

VineSLAM, while not achieving the lowest error in ideal conditions, demonstrates relatively balanced performance across diverse sequences. Its multi-modal particle filter enables integration

Table 4.1: Benchmark results of SLAM algorithms on each sequence. Metrics include Absolute Trajectory Error (ATE, in meters), Relative Translational Error (RTE, in %), and Relative Rotational Error (RRE, in degrees per meter).

SLAM Algorithm	Static	Dynamic	Ramp 15°	Symmetric Corridor	Ramp 30°	Slippery Zone
LeGO-LOAM	0.109 m	0.110 m	0.098 m	0.108 m	0.349 m	3.754 m
	1.85%	1.81%	1.67%	1.46%	3.18%	2.29%
	0.177°/m	0.173°/m	0.146°/m	0.138°/m	0.239°/m	0.236°/m
VineSLAM	0.216 m	0.159 m	0.193 m	0.164 m	0.448 m	1.752 m
	2.11%	1.97%	1.77%	1.56%	2.89%	2.68%
	0.177°/m	0.178°/m	0.164°/m	0.130°/m	0.221°/m	0.213°/m
KISS-ICP	0.086 m	0.090 m	0.108 m	0.236 m	0.150 m	0.112 m
	1.12%	1.08%	1.09%	1.84%	1.46%	1.17%
	0.108°/m	0.106°/m	0.106°/m	0.096°/m	0.123°/m	0.115°/m

of multiple odometry sources, including wheel encoders and inertial measurements, which stabilizes pose tracking under dynamic or degraded motion conditions. However, the system lacks global consistency mechanisms; drift is not corrected over time since it does not perform loop closure or trajectory optimization. This limitation becomes evident in long or ambiguous trajectories where accumulated error in the fused odometry cannot be compensated by local surfel alignment alone.

The dense surfel map and probabilistic fusion improve local robustness, but without global spatial awareness or loop detection, VineSLAM underperforms in scenes where revisiting poses or correcting accumulated drift is crucial. Notably, its rotational errors remain modest, suggesting that its projective alignment strategy and IMU integration are effective for angular stabilization. Nonetheless, consistent overestimation in translational drift under challenging terrain indicates that the particle filter fusion alone is insufficient for long-term mapping in the absence of global error correction.

Overall, while VineSLAM provides a strong foundation for odometry fusion and dense mapping, the benchmarking results indicate that enhancements such as loop closure detection may be necessary to extend its capabilities toward robustness.

Qualitative Map Evaluation

Qualitative inspection of the reconstructed maps was conducted across all test environments to complement the trajectory error metrics. These visualizations reveal systematic differences in mapping fidelity, structural consistency, and artifact presence, which help explain the observed performance trends and highlight architectural limitations in each system.

Across all methods, a common error was identified as z-axis misalignment (i.e., inaccurate height estimation), where flat surfaces appeared sloped or vertically inconsistent. While LeGO-LOAM and KISS-ICP both exhibited mild inclination artifacts, the phenomenon was amplified

in VineSLAM when operating with wheel odometry, which lacks direct observation of elevation, roll, and pitch. This limitation reduces pose observability in 6-DoF space, especially during non-planar motion. The dense map accumulated vertical drift due to uncorrected pose errors in pitch and elevation. This limitation reflects the reliance of VineSLAM's fusion strategy on the accuracy of the incoming odometry stream and the lack of a global correction mechanism.

To address this issue, the frame-to-frame LiDAR odometry from LeGO-LOAM was substituted as the motion input to VineSLAM. The resulting maps demonstrated markedly improved structural consistency and reduced z-axis distortion. This supports the conclusion that VineSLAM would benefit from an integrated LiDAR odometry module, providing higher-fidelity motion priors and mitigating the limitations of wheel-based estimation, especially in scenarios with degraded traction or poor contact stability.

These map-based evaluations reinforce the quantitative trends and suggest that while VineSLAM's architecture enables general robustness, its performance is tightly coupled to the reliability of its odometry input. Enhancing its front-end with LiDAR-based motion estimation or integrating lightweight backend correction could significantly improve mapping fidelity in challenging environments.

4.2 Odometry Benchmarking

In VineSLAM, the odometry source plays a central role in the particle filter's motion update stage, where it provides the prior for particle propagation before observation-based correction. Initially, this role was filled by wheel odometry derived from encoder data. However, the qualitative analysis of reconstructed maps revealed consistent drift and structural distortion—particularly along the vertical axis—when relying on wheel odometry in environments with dynamic motion, poor traction, or elevation change. These observations motivated the integration of LiDAR-based odometry and its extension to LiDAR-Inertial fusion as alternative motion models within the framework.

To assess the effectiveness of each strategy, odometry outputs were evaluated across the same six synthetic environments used in the SLAM benchmarking. The evaluation was performed using the `evo` toolkit, with trajectories aligned to ground truth. Given that odometry modules are primarily responsible for short-term motion prediction rather than long-term global consistency, the analysis focuses on Relative Translation Error (RTE) and Relative Rotation Error (RRE), which are computed over 1-meter intervals to better reflect frame-to-frame accuracy. Absolute Trajectory Error (ATE) is reported for reference but is not the main evaluation criterion.

Table 4.2 presents the updated results. As expected, wheel odometry showed the highest errors in all conditions, with particularly large rotational drift (RRE > 7°/m) in the ramp30 and slippery zone sequences. LiDAR odometry significantly improved both translational and rotational accuracy, achieving RRE below 2.5°/m in all sequences. The inclusion of IMU data further enhanced short-term rotational precision, with RRE values dropping below 1°/m in most sequences and RTE

Table 4.2: Odometry performance across environments: ATE (m), RTE (%), RRE (°/m). Metrics were computed with a trajectory sampling interval of 1 m.

Odometry Method	Static	Dynamic	Ramp 15°	Symmetric Corridor	Ramp 30°	Slippery Zone
Wheel	21.681 m	22.510 m	19.059 m	40.675 m	19.129 m	22.229 m
	13.37%	13.54%	12.82%	12.88%	14.39%	15.30%
	7.108°/m	7.040°/m	6.082°/m	5.049°/m	7.750°/m	8.052°/m
LiDAR	9.661 m	9.556 m	4.039 m	7.080 m	10.504 m	22.749 m
	12.34%	12.56%	10.29%	10.61%	13.85%	12.27%
	1.708°/m	1.612°/m	1.428°/m	1.359°/m	2.250°/m	1.955°/m
LiDAR-Inertial	7.774 m	9.659 m	7.155 m	3.223 m	48.027 m	7.305 m
	9.94%	11.16%	9.47%	9.91%	18.38%	10.48%
	0.874°/m	0.799°/m	0.788°/m	0.663°/m	4.341°/m	0.905°/m

consistently under 11%. This enhanced short-term consistency is particularly beneficial for VineSLAM, where the accuracy of the prediction step directly impacts the alignment of LiDAR features and the reliability of the observation likelihood computation.

LiDAR and LiDAR-Inertial methods both demonstrated strong performance in structured environments such as the static and symmetric corridor sequences, where consistent geometry supports reliable scan registration. In these cases, the addition of inertial data offered incremental improvements in rotational precision but was not critical for accurate pose estimation. In more complex environments involving degraded traction or dynamic interference—such as the slippery zone—the LiDAR-Inertial fusion method yielded more stable orientation tracking and lower short-term drift, highlighting the benefit of combining motion priors from multiple sensing modalities. However, the performance degradation observed in the ramp30 sequence—where LiDAR-Inertial odometry exhibited the highest overall error—suggests that this fusion approach remains sensitive to vertical motion and IMU bias accumulation. These findings indicate that while inertial fusion can enhance robustness in many scenarios, its integration must be carefully tuned to avoid introducing new sources of error under steep elevation changes.

Overall, these results confirm that geometric odometry sources (LiDAR and LiDAR-Inertial) are more suitable than wheel odometry for driving the motion model in VineSLAM's prediction step. Moreover, the improved RTE and RRE values directly translate to more reliable and consistent particle propagation, which in turn supports improved map consistency and pose estimation throughout the SLAM pipeline.

Chapter 5

Conclusion and Future Work

This dissertation explored the challenges and opportunities of deploying robust SLAM systems for autonomous mobile robots operating in structured industrial environments. Grounded in the limitations of traditional GNSS-based localization and the unreliability of wheel odometry in dynamic or non-planar indoor settings, the work focused on adapting and improving VineSLAM, a particle-filter-based SLAM framework developed for agricultural robotics. A central goal was to evaluate the system performance and scalability when applied to structured indoor environments, and to develop tools and methods that facilitate repeatable, systematic SLAM benchmarking.

To achieve this, a major contribution of the work was the design and implementation of a modular simulation and benchmarking framework based on Webots and ROS 2. The system featured a robot model equipped with LiDAR, camera, IMU, and wheel encoders, and operated in procedurally generated environments modeled after real-world laboratory settings. The simulation sequences were crafted to isolate critical challenges such as dynamic objects, symmetric corridors, inclined ramps, and low-traction surfaces—features that are common in industrial facilities yet difficult to control or reproduce in physical experiments. The resulting synthetic dataset was designed with reproducibility, temporal consistency, and compatibility in mind, enabling the testing of 3D SLAM pipelines in a controlled, repeatable manner. This framework provided a scalable and efficient testbed for evaluating algorithmic robustness and identifying failure modes early in the development cycle.

Three SLAM algorithms—KISS-ICP, LeGO-LOAM, and VineSLAM—were evaluated using this framework. KISS-ICP proved to be effective for short-range odometry in geometric environments but showed limitations in extended or ambiguous spaces. LeGO-LOAM demonstrated high efficiency and accuracy on flat terrains but was sensitive to environmental irregularities such as ramps or loss of traction. VineSLAM, while developed for agricultural settings, demonstrated good performance in symmetric and structurally repetitive indoor environments. Its semiplane representation and particle-filter-based localization offered resilience in situations where perceptual aliasing degraded other algorithms.

To mitigate these weaknesses, this dissertation introduced the idea of using LeGO-LOAM LiDAR odometry, with optional IMU integration, as motion estimation for VineSLAM particle filter,

to replace its wheel odometry motion estimation with a more reliable scan-to-scan matching approach fused with IMU data. This enhancement significantly reduced drift in vertical and dynamic scenarios and improved the overall structural integrity of the generated maps. However, while this integration addressed the system's most immediate source of error, VineSLAM still lacks a global optimization component. Without loop closure detection or a back-end graph optimizer, the system remains vulnerable to cumulative drift in extended trajectories.

The synthetic benchmarking platform itself also revealed broader insights into SLAM system evaluation. Controlled simulation allowed precise diagnosis of algorithmic weaknesses, such as sensitivity to terrain, loss of traction, or perceptual aliasing. While synthetic environments cannot fully capture the noise and unpredictability of the real world, they provide an essential sandbox for stress-testing algorithms and facilitating rapid iteration. The synthetic dataset benchmarking framework developed here offers a foundation for future research, especially in the sim-to-real domain, and is designed to be extensible with new sensors, environments, and coordination scenarios.

Future work should focus on three major areas. First, integrating loop closure and pose graph optimization into VineSLAM is a natural next step to address long-term drift and improve mapping consistency. Methods such as Scan Context, BoW3D, and factor graph optimizers, including g2o [26] and iSAM2 [25], offer promising paths for this enhancement. Second, the odometry could be further improved through tightly coupled LiDAR-Inertial or LiDAR-Inertial-Visual fusion techniques, incorporating advances from systems like LIO-SAM [38] and R3LIVE [47] to handle degenerate geometry or dynamic conditions better. Third, real-world deployment and evaluation in industrial testbeds are needed to validate the sim-to-real transfer of the system. This includes assessing mapping accuracy and robustness to real sensor noise, calibration error, and unexpected environmental changes.

Additionally, the benchmarking platform can be expanded to include semantic and RGB-D data and support for multiple robots and tasks such as collaborative mapping or dynamic object filtering. These extensions will help align the framework with the emerging needs of industrial automation and multi-agent systems while providing a controlled, reproducible space for algorithm development and testing.

In summary, this dissertation provides system-level contributions and methodological tools for advancing SLAM in structured indoor settings. By proposing a new motion estimation source to VineSLAM and validating its performance in controlled synthetic environments, this work confirms that the system has potential for industrial adaptation. The combination of benchmarking infrastructure and algorithmic refinements lays a strong foundation for future research and deployment in real-world automation applications.

References

[1] A. S. Aguiar, F. Neves dos Santos, H. Sobreira, J. Boaventura-Cunha, and A. J. Sousa, "Localization and mapping on agriculture based on point-feature extraction and semiplanes segmentation from 3d lidar data," *Frontiers in Robotics and AI*, vol. 9, 2022, ISSN: 2296-9144. DOI: 10.3389/frobt.2022.832165. [Online]. Available: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2022.832165.

[2] A. S. Aguiar, F. N. dos Santos, H. Sobreira, J. B. Cunha, and A. J. Sousa, "Particle filter refinement based on clustering procedures for high-dimensional localization and mapping systems," *Robotics and Autonomous Systems*, vol. 137, p. 103 725, 2021, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2021.103725. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889021000105.

[3] T. Levin, *Synthetic dataset generation for 3d slam evaluation*, Zenodo, Apr. 2025. DOI: 10.5281/zenodo.15274730. [Online]. Available: https://doi.org/10.5281/zenodo.15274730.

[4] C. Cadena, L. Carlone, H. Carrillo, *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016. DOI: 10.1109/TRO.2016.2624754.

[5] A. S. Aguiar, F. N. dos Santos, J. B. Cunha, H. Sobreira, and A. J. Sousa, "Localization and mapping for robots in agriculture and forestry: A survey," *Robotics*, vol. 9, no. 4, 2020, ISSN: 2218-6581. DOI: 10.3390/robotics9040097.

[6] P. Biber and W. 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) (Cat. No.03CH37453)*, vol. 3, 2003, 2743–2748 vol.3. DOI: 10.1109/IROS.2003.1249285.

[7] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992. DOI: 10.1109/34.121791.

[8] J. Zhang and S. Singh, "Loam : Lidar odometry and mapping in real-time," *Robotics: Science and Systems Conference (RSS)*, pp. 109–111, Jan. 2014.

[9] J. Lin and F. 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)*, 2020, pp. 3126–3131. DOI: 10.1109/ICRA40945.2020.9197440.

[10] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 2161–2168. DOI: 10.1109/CVPR.2006.264.

[11] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth, "Openfabmap: An open source toolbox for appearance-based loop closure detection," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 4730–4735. DOI: 10.1109/ICRA.2012.6224843.

[12] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012, ISSN: 1552-3098. DOI: 10.1109/TRO.2012.2197158.

[13] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792, ISBN: 978-3-642-15561-1.

[14] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443, ISBN: 978-3-540-33833-8.

[15] G. Kim and A. 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)*, 2018, pp. 4802–4809. DOI: 10.1109/IROS.2018.8593953.

[16] M. A. Uy and G. H. Lee, *Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition*, 2018. arXiv: 1804.03492 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1804.03492.

[17] Y. Cui, X. Chen, Y. Zhang, J. Dong, Q. Wu, and F. Zhu, "Bow3d: Bag of words for real-time loop closing in 3d lidar slam," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2828–2835, May 2023, ISSN: 2377-3774. DOI: 10.1109/lra.2022.3221336. [Online]. Available: http://dx.doi.org/10.1109/LRA.2022.3221336.

[18] Y. Cui, Y. Zhang, J. Dong, H. Sun, X. Chen, and F. Zhu, "Link3d: Linear keypoints representation for 3d lidar point cloud," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–8, Mar. 2024. DOI: 10.1109/LRA.2024.3354550.

[19] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3562–3568. DOI: 10.1109/IROS.2006.281644.

[20] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, *Fastslam: A factored solution to the simultaneous localization and mapping problem*, 2002. [Online]. Available: www.aaai.org.

[21] "Fastslam 2.0," in *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 63–90, ISBN: 978-3-540-46402-0. DOI: 10.1007/978-3-540-46402-0_4. [Online]. Available: https://doi.org/10.1007/978-3-540-46402-0_4.

[22] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Apr. 2005, pp. 2432–2437. DOI: 10.1109/ROBOT.2005.1570477.

[23] S. Thrun, "Simultaneous localization and mapping," in *Robotics and Cognitive Approaches to Spatial Mapping*, M. E. Jefferies and W.-K. Yeap, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 13–41, ISBN: 978-3-540-75388-9. DOI: 10.1007/978-3-540-75388-9_3.

[24] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010. DOI: 10.1109/MITS.2010.939925.

[25] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "Isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3281–3288. DOI: 10.1109/ICRA.2011.5979641.

[26] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613. DOI: 10.1109/ICRA.2011.5979949.

[27] Q. Zou, Q. Sun, L. Chen, B. Nie, and Q. Li, "A comparative analysis of lidar slam-based indoor navigation for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6907–6921, 2022. DOI: 10.1109/TITS.2021.3063477.

[28] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-loam : Fast lidar odometry and mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4390–4396. DOI: 10.1109/IROS51168.2021.9636655.

[29] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4758–4765. DOI: 10.1109/IROS.2018.8594299.

[30] L. Liao, C. Fu, B. Feng, and T. Su, "Optimized sc-f-loam: Optimized fast lidar odometry and mapping using scan context," in *2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, 2022, pp. 1–6. DOI: 10.1109/CVCI56766.2022.9964574.

[31] T. Guadagnino, B. Mersch, I. Vizzo, *et al.*, *Kinematic-icp: Enhancing lidar odometry with kinematic constraints for wheeled mobile robots moving on planar surfaces*, 2024. arXiv: 2410.10277 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2410.10277.

[32] S. W. Chen, G. V. Nardari, E. S. Lee, *et al.*, "Sloam: Semantic lidar odometry and mapping for forest inventory," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 612–619, 2020. DOI: 10.1109/LRA.2019.2963823.

[33] B. Kim, C. Jung, D. H. Shim, and A.-a. Agha–mohammadi, "Adaptive keyframe generation based lidar inertial odometry for complex underground environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3332–3338. DOI: 10.1109/ICRA48891.2023.10161207.

[34] A. Tagliabue, J. Tordesillas, X. Cai, *et al.*, "Lion: Lidar-inertial observability-aware navigator for vision-denied environments," in *Experimental Robotics*, B. Siciliano, C. Laschi, and O. Khatib, Eds., Cham: Springer International Publishing, 2021, pp. 380–390, ISBN: 978-3-030-71151-1.

[35] W. Xu and F. Zhang, *Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter*, 2021. arXiv: 2010.08196 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2010.08196.

[36] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022. DOI: 10.1109/TRO.2022.3141876.

[37] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-lio: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4861–4868, 2022. DOI: 10.1109/LRA.2022.3152830.

[38] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5135–5142. DOI: 10.1109/IROS45743.2020.9341176.

[39] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2174–2181. DOI: 10.1109/ICRA.2015.7139486.

[40] Y. S. Shin, Y. S. Park, and A. Kim, "Dvl-slam: Sparse depth enhanced direct visual-lidar slam," *Autonomous Robots*, vol. 44, pp. 115–130, 2 Jan. 2020, ISSN: 15737527. DOI: 10.1007/s10514-019-09881-0.

[41] S.-S. Huang, Z.-Y. Ma, T.-J. Mu, H. Fu, and S.-M. Hu, "Lidar-monocular visual odometry using point and line features," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1091–1097. DOI: 10.1109/ICRA40945.2020.9196613.

[42] R. Radmanesh, Z. Wang, V. S. Chipade, G. Tsechpenakis, and D. Panagou, "Liv-lam: Lidar and visual localization and mapping," in *2020 American Control Conference (ACC)*, 2020, pp. 659–664. DOI: 10.23919/ACC45564.2020.9148037.

[43] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," vol. 2015, May 2015. DOI: 10.1109/ICRA.2015.7139486.

[44] J. Zhang and S. Singh, "Laser–visual–inertial odometry and mapping with high robustness and low drift," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1242–1264, 2018. DOI: https://doi.org/10.1002/rob.21809. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21809. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21809.

[45] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, "Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8729–8736. DOI: 10.1109/IROS51168.2021.9635862.

[46] T. Shan, B. Englot, C. Ratti, and D. Rus, "Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5692–5698. DOI: 10.1109/ICRA48506.2021.9561996.

[47] J. Lin and F. Zhang, "R3live: A robust, real-time, rgb-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 672–10 678. DOI: 10.1109/ICRA46639.2022.9811935.

[48] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[49] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[50] M. Burri, J. Nikolic, P. Gohl, *et al.*, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. DOI: 10.1177/0278364915620033. eprint: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.full.pdf+html. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract.

[51] W. Wang, D. Zhu, X. Wang, *et al.*, "Tartanair: A dataset to push the limits of visual slam," 2020. arXiv: 2003.14338 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2003.14338.

[52] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," 2017. arXiv: 1705.05065 [cs.RO]. [Online]. Available: https://arxiv.org/abs/1705.05065.

[53] H. Park, I. Lee, M. Kim, H. Park, and K. Joo, "A benchmark dataset for collaborative slam in service environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 12, pp. 11 337–11 344, 2024. DOI: 10.1109/LRA.2024.3491415.

[54] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 23–30.

[55] A. Farley, J. Wang, and J. A. Marshall, "How to pick a mobile robot simulator: A quantitative comparison of coppeliasim, gazebo, morse and webots with a focus on accuracy of motion," *Simulation Modelling Practice and Theory*, vol. 120, p. 102 629, 2022, ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2022.102629. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X22001046.

[56] O. Michel, "Cyberbotics ltd. webots™: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004. DOI: 10.5772/5618.

[57] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[58] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., ser. Proceedings of Machine Learning Research, vol. 78, PMLR, 13–15 Nov 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html.

[59] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006. DOI: 10.1109/MRA.2006.1638022.

[60] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580. DOI: 10.1109/IROS.2012.6385773.

[61] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "Kiss-icp: In defense of point-to-point icp – simple, accurate, and robust registration if done the right way," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1029–1036, 2023. DOI: 10.1109/LRA.2023.3236571.

[62] S. Umeyama, “Least-squares estimation of transformation parameters between two point patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991. DOI: 10.1109/34.88573.