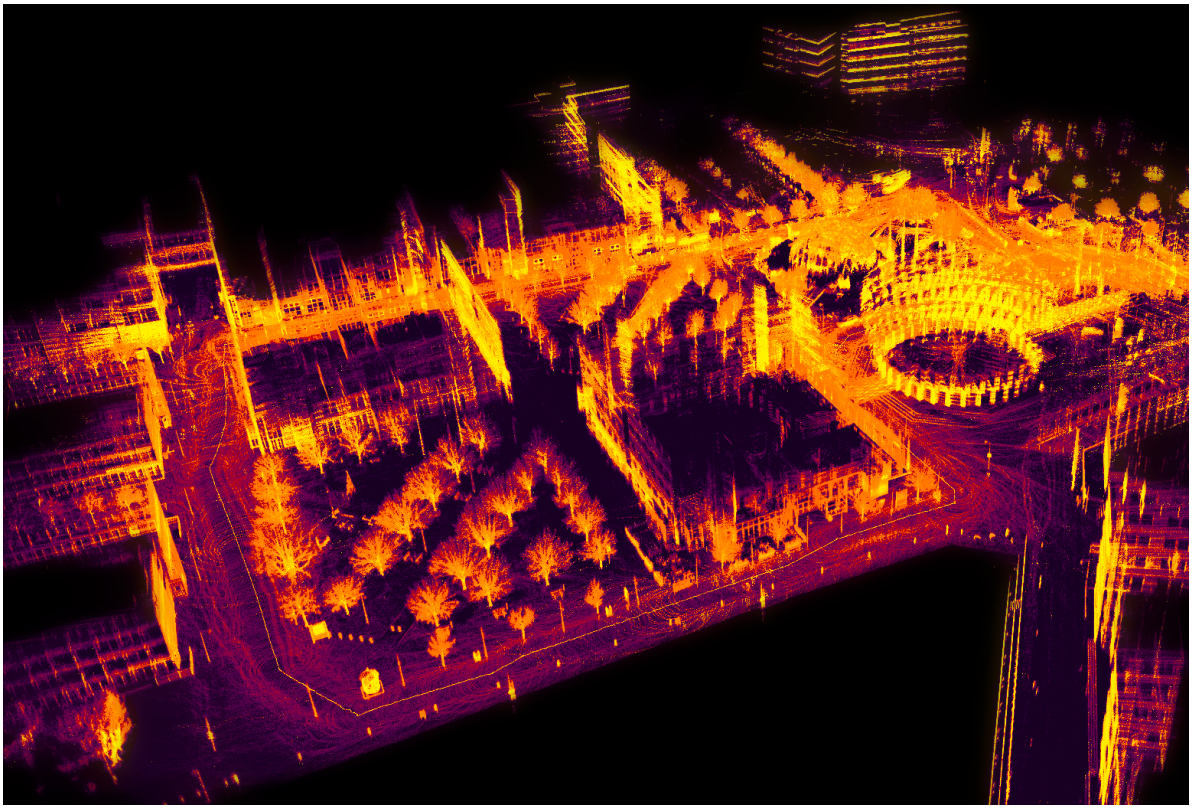
CHALMERS
UNIVERSITY OF TECHNOLOGY



Picking the raisins out of the cookie

LiDAR based odometry in dynamic urban environments

Master's thesis in Systems, Control and Mechatronics

ANDREAS ÅBERG

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025

Master's thesis 2025

Picking the raisins out of the cookie

LiDAR based odometry in dynamic urban environments

Andreas Åberg

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

Picking the raisins out of the cookie
LiDAR based odometry in dynamic urban environments
ANDREAS ÅBERG

© ANDREAS ÅBERG, 2025.

Supervisor: Lars Hammarstrand, Department of Electrical Engineering
Advisor: Ernesto Lozano Calvo, Hugo Delivery AB
Examiner: Lars Hammarstrand, Department of Electrical Engineering

Master's Thesis 2025
Department of Electrical engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: LiDAR point cloud visualization of Lindholmen campus, Gothenburg.

Typeset in LaTeX
Gothenburg, Sweden 2025

Picking the raisins out of the cookie
LiDAR based odometry in dynamic urban environments

ANDREAS ÅBERG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Reliable localization using LiDAR on pedestrian-scale delivery robots in urban environments poses distinctive challenges. Methods proven on larger vehicles with high-end sensors do not necessarily translate to compact platforms with simpler payloads. This thesis focuses on evaluating some state-of-the-art LiDAR odometry approaches on a small delivery robot equipped with a LiDAR and other sensors such as an IMU. By comparing trajectory accuracy, resilience to real-world motion profiles, and multiple sensor configurations, insight is provided into which techniques can serve as a reliable baseline and what additional capabilities are required to achieve practical localization on often more resource-constrained robots.
Building on this foundation, we explore targeted enhancements driven by real-world observations. We implement a real-time simple dynamic-object filtering approach that operates in parallel with odometry, based on a range images, where moving objects are subsequently identified and excised to maintain a static map. To further mitigate drift over extended runs, we integrate loop-closure adjustments, detecting and aligning previously visited areas to correct accumulated drift over larger distances. Together, these refinements demonstrate how a robust baseline can be augmented to meet the demands of odometry and mapping of a small delivery robot.

Keywords: LiDAR Odometry, Urban Localization, Dynamic Object Removal, SLAM.

Acknowledgements

I would like to thank Hugo Delivery AB for providing me with the opportunity and resources to conduct this thesis. Special thanks go to my industrial supervisor, Ernesto Lozano Calvo, whose insightful feedback have been invaluable throughout this project. I would also like to express my gratitude to my supervisor and examiner, Lars Hammarstrand, for his thorough guidance, support, and constructive feedback, which have significantly shaped this thesis. Furthermore, my sincere appreciation goes to my opponents, Anton Carlsson and Aron Enliden, for their thoughtful reviews and valuable suggestions, greatly improving the quality of this work. A heartfelt thank you goes to my friends and family, who have supported me tirelessly throughout my education. Lastly, I extend a special thanks to Chalmers Formula Student, for being a place to step away from thesis work during moments when a change of pace was needed.

Andreas Åberg, Gothenburg, June 13, 2025

List of Acronyms

Below is a list of acronyms that have been used throughout this thesis listed in alphabetical order:

BEV	Bird's-Eye View
BFS	Breadth-First Search
DLIO	Direct LiDAR–Inertial Odometry
FOV	Field Of View
GICP	Generalized ICP
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
ICP	Iterative Closest Point
LiDAR	Light Detection And Ranging
LOAM	LiDAR Odometry And Mapping
RANSAC	Random Sample Consensus
ROS	Robot Operating System
SLAM	Simultaneous Localization And Mapping

Contents

List of Figures

List of Tables

1

Introduction

Autonomous robots are rapidly transitioning from laboratory curiosities into real-world platforms for last-mile delivery, urban inspection, and on-demand mobility services. As fleets of compact, low-cost robots begin to navigate crowded sidewalks, robust and precise localization becomes the backbone for both operational efficiency and public safety. Unlike full-scale autonomous vehicles—equipped with high-end LiDAR arrays, multi-camera rigs, radar units, and powerful onboard GPUs, small delivery robots must contend with strict payload limits, tight power budgets, and the need to run all perception and planning onboard in real time on embedded hardware.

Global Navigation Satellite Systems (GNSS), though ubiquitous, suffer severe degradation beneath urban canopies and within the narrow corridors of modern cityscapes. Wheel-encoder odometry can quickly accumulate drift or suffer large errors due to wheel slip on uneven pavements or transient contacts with obstacles. By contrast, LiDAR sensors provide direct, high-resolution depth measurements that are largely invariant to ambient light, making them a strong option for mapping and localization in dynamic, unstructured urban environments.

Nevertheless, state-of-the-art LiDAR odometry algorithms designed for automotive-grade LiDAR scanners do not seamlessly transfer to the constraints of pedestrian-scale robots. Lighter, lower-resolution sensors produce sparser point clouds with limited computational headroom. Moreover, the urban environment itself presents unique challenges such as pedestrians, and cyclists, leading naive scan-matching to potentially latch onto transient features in a sparse lidar point cloud. Without explicit handling maps become cluttered with dynamic objects, undermining downstream tasks such as path planning and obstacle avoidance.

This thesis therefore investigates how to adapt and extend LiDAR odometry for pedestrian-scale delivery robots operating in dynamic urban settings. We first establish a LiDAR-odometry baseline by benchmarking three state-of-the-art solutions on our self-collected urban datasets. Next, we integrate a factor graph based loop-closure module with ICP matching, allowing long-term drift correction while preserving real-time performance on embedded platforms. Second, we show a dynamic-object filtering pipeline based on range-image clustering, which segments and excludes moving elements thereby maintaining a clean static map.

1.1 Objective

This thesis seeks to advance LiDAR-based odometry for pedestrian-scale delivery robots navigating dynamic urban environments. The goal is thesis to find a LiDAR-based odometry solution suited for pedestrian-scale delivery robots navigating dynamic urban environments, that is both highly accurate and robust enough to handle longer traversals without relying on GPS.

1.2 Limitations

The evaluation in this thesis is confined to self-collected datasets gathered using the target delivery-robot platform under its normal operating conditions. While numerous public LiDAR benchmarks exist, they offer little benefit for assessing performance on our specific hardware and sensor configuration, and thus are not included.

A further constraint arises from the absence of precise ground truth: generating centimeter-level trajectories would require high-precision motion-capture systems or survey-grade GNSS, neither of which is available on the platform. Consequently, we rely on end-to-end translational drift metrics and qualitative map inspections to gauge odometry accuracy. Additionally, we deliberately exclude deep-learning approaches, as they demand large annotated datasets beyond the scope of our self-collection efforts. Finally, though LiDAR-IMU fusion is explored to enhance robustness, we do not incorporate broader multi-sensor fusion (e.g., GPS or camera), since one key question is whether a minimal, GPS-denied setup can provide sufficient localization for pedestrian-scale robots.

1.3 Contributions & Results

Through experiments on our self-collected urban LiDAR datasets, this thesis delivers the following key results and contributions:

1. **Benchmarking of LiDAR Odometry Methods.** We compare three state-of-the-art scan-matching pipelines—point-to-point ICP, GICP, and feature-based on our delivery-robot platform. This benchmarking identifies the most robust baseline under typical operating scenarios and environment complexities (open plazas, narrow alleys).
2. **Real-Time ICP-Based Loop Closure.** We integrate an ICP-based loop-closure module into a lightweight pose-graph framework and demonstrate a significant reduction in end-to-end translational drift on longer trajectories over the original implementation.
3. **Dynamic-Object Filtering via Range-Image Clustering.** We adapt a previously offline filtering approach into a real-time pipeline that segments and removes moving objects (pedestrians, bicycles, delivery carts) by using range-images.

To capture the essence of our method, and to explain the thesis title, “Picking the Raisins Out of the Cookie”, we deliberately “cherry-pick” (as it conversely means in Swedish) elements from multiple LiDAR-odometry pipelines to assemble a more complete solution. This targeted selection of a LiDAR-odometry system combined with loop closure and dynamic-object removal, exemplifies how combining techniques can yield a minimal yet powerful system.

2
Background

To provide a foundation for this work, several key concepts related to LiDAR-based perception and odometry are introduced in this chapter. First, the principles of Light Detection and Ranging (LiDAR) technology and the platforms used for data collection are outlined. Next, methods for point cloud alignment are discussed, including classical algorithms such as Iterative Closest Point (ICP) and its variants, as well as modern feature extraction techniques. Building upon this, different approaches to LiDAR odometry are presented, ranging from LiDAR-only systems to both loosely and tightly coupled LiDAR-inertial configurations. To ensure mapping consistency, techniques for loop closure are described, including a graph-based formulation. Finally, strategies for dynamic object extraction are reviewed to account for moving elements in the environment.

2.1 Light Detection and Ranging

Light Detection and Ranging (LiDAR) is a remote sensing technique designed to measure distances and construct high-resolution 3D representations of environments and objects. The system functions by emitting laser pulses that reflect off surfaces and return to the sensor. By calculating the time it takes for each pulse to return and applying the known speed of light, the system determines the range of each return [1].

Each range measurement is then decoded using the known angular orientation of the emitter into a point in 3D space. When deployed across expansive areas, LiDAR accumulates vast numbers of such points into a "point cloud", a detailed 3D representation of the scene's geometry.

2.2 Platform

The solution will be deployed and partly evaluated on smaller autonomous robots depicted in Figure 2.1, which has a ROS2 based system that currently handles localization primarily by wheel-encoders, an IMU sensor and a GPS. The GPS can suffer from poor signal quality in urban and underground environments due to potential occlusion and signal multipath effects. The challenge lies in augmenting these systems with additional odometry data derived from sensors in a variety of environments while maintaining computational efficiency and real-time processing. Visual and LiDAR odometry have been successfully applied to address this limita-

tion, however, their integration into a robust localization framework for autonomous urban robots remains a complex challenge, as noted in a recent survey on LiDAR odometry approaches [2]. These findings underline the relevance and novelty of the proposed work in the robotics domain.

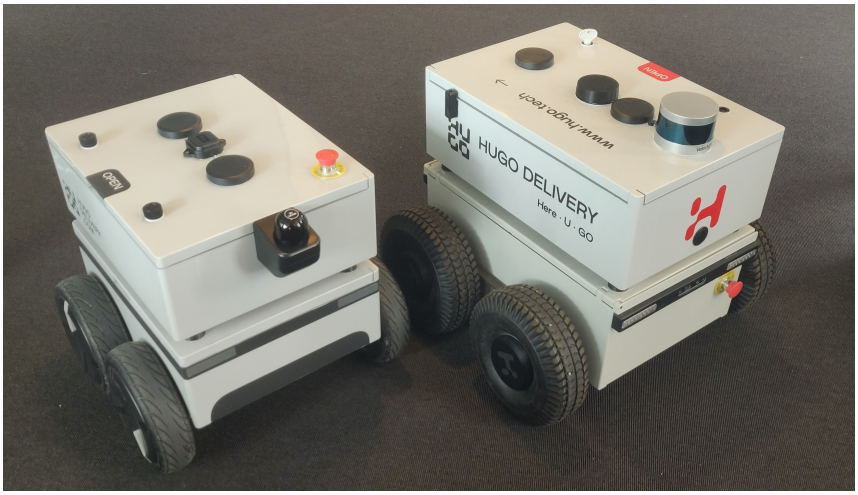


Figure 2.1: HUGO prototype robots

Data was collected using two different LiDAR units: Velodyne VLP-16 and Hesai JT16. The VLP-16 provides higher angular resolution and longer range, whereas the JT16 offers a lightweight, compact form factor with the cost of lower point density. Table 2.1 summarizes their key specifications.

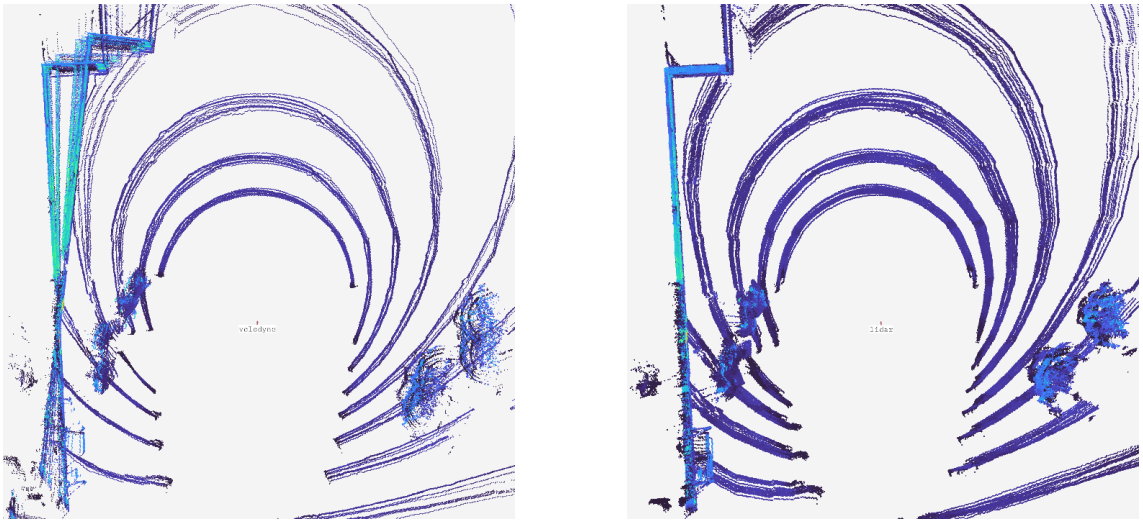
Metric	**VLP-16**	**JT16**
Channels	16	16
Range	100 m @ 10% refl.	30 m @ 10 % refl. (100m max)
Ranging Accuracy (typical)	± 3 cm	± 3 cm
FOV (Horizontal)	360°	360°
FOV (Vertical)	30° (±15°)	40° (0–40°)
Horizontal Angular Resolution	0.1–0.4°	0.6° (5 Hz), 1.2° (10 Hz)
Vertical Angular Resolution	2.0°	2.67°
Frame Rate / Rotation Speed	5–20 Hz	5 Hz / 10 Hz
Weight	830 g	$<$ 200 g

Table 2.1: Key specifications of Velodyne VLP-16 and Hesai JT16

2.3 Point Cloud Matching

Point cloud matching in lidar odometry fundamentally relies on aligning raw point clouds to estimate the relative motion between consecutive lidar scans. These approaches typically employ Iterative Closest Point (ICP) [3] algorithms and their variants to minimize the distance between points in successive frames, thus deriving the transformation parameters that describe the robot or sensor motion. The ICP algorithm iteratively pairs points from the source scan to the nearest points in the target scan and computes the optimal rigid transformation to minimize the cumulative distance between paired points.

Several variations of the ICP algorithm have been developed to improve accuracy, robustness, and computational efficiency. For instance, the Point-to-Point ICP method directly minimizes the distances between corresponding points, which is straightforward but can be sensitive to initial guesses and prone to local minima. Point-to-Plane ICP, on the other hand, minimizes the distance from points in one scan to the planes estimated from their corresponding points in another scan [4]. This approach typically exhibits better convergence properties and is more robust.



(a) Multiple pointcloud under motion **(b)** After deskewing

Figure 2.2: Point cloud motion deskewing. The most pronounced misalignment is visible on the wall at the left without deskewing

To improve the accuracy and efficiency of point cloud matching, especially under fast sensor motion, it is common to pre-process lidar scans through motion deskewing. This involves compensating for motion-induced distortions that arise due to the time it takes for a lidar sensor to complete a full scan. As shown in Figure 2.2, raw scans (left) can appear warped when motion is not corrected, while deskewed scans (right) are geometrically consistent and better aligned. Deskewing typically relies on prior motion information, either by assuming a velocity model or by incorporating data from additional sensors such as inertial measurement units (IMUs). This preprocessing step significantly speeds up the matching process by reducing

alignment errors and improving the convergence behavior of point cloud matching algorithms.

To enhance computational speed and robustness in dynamic environments, researchers have integrated point-based methods with probabilistic frameworks, such as Kalman filters or particle filters, providing real-time performance with improved noise handling. Furthermore, advanced variations, such as Generalized ICP (GICP), combine both point-to-point and point-to-plane strategies, adapting dynamically based on local surface properties. This hybrid method leverages the strengths of each ICP variant to yield higher accuracy and faster convergence.

Despite their popularity and simplicity, pure point-based methods suffer from inherent limitations, particularly sensitivity to dynamic objects, occlusions, and sparse environments. These challenges could lead to inaccuracies in the estimation process, necessitating the combination or integration of these methods with feature-based or deep-learning-based approaches to enhance overall robustness and reliability.

2.3.1 Iterative Closest Point (ICP)

The ICP algorithm [5] is a fundamental method in geometric computer vision and robotics for aligning two point clouds. It iteratively refines the transformation (rotation and translation) that best aligns a source point cloud to a target point cloud.

Known Data Association

In the case of known data association, correspondences between points $\boldsymbol{p}_n$ in the source and target point $\boldsymbol{q}_n$ are pre-established. That is, for each point in the source cloud, the corresponding point in the target cloud is known beforehand. This scenario simplifies the ICP algorithm significantly. Given these known correspondences, the goal is to find the rigid transformation, rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$, that minimizes the sum of squared distances between corresponding point pairs,

$$\bar{\boldsymbol{p}}_n = \boldsymbol{R}\boldsymbol{p}_n + \boldsymbol{t}, \tag{2.1}$$

such that,

$$\min \sum ||\boldsymbol{q}_n - \bar{\boldsymbol{p}}_n||^2. \tag{2.2}$$

This is the same as the Orthogonal Procrustes problem [6] which is solved by Singular value decomposition (SVD) of the cross-covariance matrix $\boldsymbol{H}$,

$$\boldsymbol{\mu}_p = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{p}_i, \quad \boldsymbol{\mu}_q = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{q}_i, \tag{2.3}$$

$$\boldsymbol{H} = \sum \left(\boldsymbol{q}_n - \boldsymbol{\mu}_q\right)\left(\boldsymbol{p}_n - \boldsymbol{\mu}_p\right)^\top, \tag{2.4}$$

$$\mathrm{svd}(\boldsymbol{H}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \tag{2.5}$$

where $\boldsymbol{\mu}$ denotes the center of mass of the points in both sets. The solution of the rigid body transform is given by

$$\begin{aligned} \boldsymbol{R} &= \boldsymbol{V}\boldsymbol{U}^{\top}, \\ \boldsymbol{t} &= \boldsymbol{\mu}_q - \boldsymbol{R}\boldsymbol{\mu}_p, \\ \bar{\boldsymbol{p}}_n &= \boldsymbol{R}\left(\boldsymbol{p}_n - \boldsymbol{\mu}_p\right) + \boldsymbol{\mu}_q. \end{aligned} \tag{2.6}$$

This procedure yields the least-squares optimal rigid body transformation aligning the source point cloud to the target point cloud under known correspondences.

Unknown Data Association

When the data association is unknown, there is no direct solution to the problem. The correspondences between points in the source and target must be estimated as part of the alignment process. An approach that tries to estimate the data association and then compute the transform is the ICP algorithm. This is done by iteratively associating the closest points between the sets and aligning the points with the rigid body transform. The basic ICP algorithm is summarized in Algorithm 1 below.

Algorithm 1: Basic ICP (Unknown Data Association)

Initialize $\bar{\boldsymbol{p}}_n = \boldsymbol{p}_n$
Set error $e = \infty$
while (e has decreased and $e >$ threshold) **do**
  $\mathcal{C} \leftarrow$ `calculate_correspondences`$(\bar{\boldsymbol{p}}_n, \boldsymbol{q}_n)$
  $(\boldsymbol{t}, \boldsymbol{R}) \leftarrow$ `compute_transformation_params`$(\mathcal{C})$
  $\bar{\boldsymbol{p}}_n \leftarrow \boldsymbol{R}(\boldsymbol{p}_n - \boldsymbol{\mu}_p) + \boldsymbol{\mu}_q$
  $e \leftarrow E(\boldsymbol{t}, \boldsymbol{R}) = \sum \left\|\boldsymbol{q}_n - \bar{\boldsymbol{p}}_n\right\|^2$
end
return $\{\bar{\boldsymbol{p}}_n\}$

2.3.2 Generalized Iterative Closest Point (GICP)

The Generalized ICP (GICP) algorithm [7] unifies the point-to-point and point-to-plane ICP variants within a single probabilistic framework. Like standard ICP, GICP operates iteratively, performing two key steps:

1. Identifying point correspondences via nearest-neighbor searches,
2. Computing the optimal rigid transformation $\boldsymbol{T} \in \mathrm{SE}(3)$ to align these correspondences.

Unlike classical ICP, which minimizes a sum of squared Euclidean distances, GICP incorporates uncertainty by associating covariance matrices with points, minimizing a sum of Mahalanobis distances.

The Mahalanobis distance accounts for the covariance of the distribution. For a point $\boldsymbol{p} \in \mathbb{R}^3$ drawn from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, it is defined as

$$d_{\boldsymbol{\Sigma}}(\boldsymbol{p}, \boldsymbol{\mu}) = \sqrt{(\boldsymbol{p} - \boldsymbol{\mu})^{\mathsf{T}}\, \boldsymbol{\Sigma}^{-1}\, (\boldsymbol{p} - \boldsymbol{\mu})}.$$

When comparing two Gaussian distributions, $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, the generalized Mahalanobis distance between their means is

$$d_{\text{gen}}(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) = \sqrt{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\mathsf{T} (\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)}.$$

Applying a rigid transformation $\boldsymbol{T} \in \text{SE}(3)$, defined by a rotation $\boldsymbol{R} \in \text{SO}(3)$ and translation $\boldsymbol{t} \in \mathbb{R}^3$, transforms a point $\boldsymbol{a} \in \mathbb{R}^3$ as follows

$$\boldsymbol{T}\boldsymbol{a} = \boldsymbol{R}\boldsymbol{a} + \boldsymbol{t}.$$

When transforming a covariance matrix Σ associated with a point, the transformed covariance becomes

$$\boldsymbol{\Sigma}' = \boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^\mathsf{T}.$$

Thus, the pairwise Mahalanobis distance used in GICP becomes

$$d_i(\boldsymbol{T}) = \sqrt{(\boldsymbol{b}_i - (\boldsymbol{R}\boldsymbol{a}_i + \boldsymbol{t}))^\mathsf{T} (C_i^B + \boldsymbol{R}C_i^A\boldsymbol{R}^\mathsf{T})^{-1} (\boldsymbol{b}_i - (\boldsymbol{R}\boldsymbol{a}_i + \boldsymbol{t}))},$$

where $\boldsymbol{a}_i, \boldsymbol{b}_i \in \mathbb{R}^3$ are corresponding points in source and target point clouds A and B, with covariances C_i^A and C_i^B.
The GICP algorithm seeks the optimal rigid transformation $\boldsymbol{T}$ minimizing the total sum of these distances,

$$\boldsymbol{T} = \underset{\boldsymbol{T}\in\text{SE}(3)}{\text{argmin}} \sum_{i=1}^{N} (\boldsymbol{b}_i - (\boldsymbol{R}\boldsymbol{a}_i + \boldsymbol{t}))^\mathsf{T} (C_i^B + \boldsymbol{R}C_i^A\boldsymbol{R}^\mathsf{T})^{-1} (\boldsymbol{b}_i - (\boldsymbol{R}\boldsymbol{a}_i + \boldsymbol{t})).$$

Consider two point clouds,

$$\{\boldsymbol{a}_i\}_{i=1}^N, \quad \{\boldsymbol{b}_i\}_{i=1}^N,$$

with unknown true alignment $\boldsymbol{T}^* \in \text{SE}(3)$. Each observed point is assumed to be a noisy measurement of an underlying true point:

$$\hat{\boldsymbol{b}}_i = \boldsymbol{T}^* \hat{\boldsymbol{a}}_i, \tag{2.7}$$

$$\boldsymbol{a}_i \sim \mathcal{N}(\hat{\boldsymbol{a}}_i, C_i^A), \tag{2.8}$$

$$\boldsymbol{b}_i \sim \mathcal{N}(\hat{\boldsymbol{b}}_i, C_i^B). \tag{2.9}$$

We define the residual at a current estimate $\boldsymbol{T}$ as

$$\boldsymbol{d}_i(\boldsymbol{T}) = \boldsymbol{b}_i - (\boldsymbol{R}\boldsymbol{a}_i + \boldsymbol{t}).$$

The optimal transformation solves the maximum-likelihood problem:

$$\boldsymbol{T} = \underset{\boldsymbol{T}\in\text{SE}(3)}{\text{argmin}} \sum_{i=1}^{N} \boldsymbol{d}_i(\boldsymbol{T})^\mathsf{T} (C_i^B + \boldsymbol{R}C_i^A\boldsymbol{R}^\mathsf{T})^{-1} \boldsymbol{d}_i(\boldsymbol{T}).$$

Special Cases

- **Point-to-point ICP:**
  Setting $C_i^A = 0$ and $C_i^B = I$, reduces the objective to the standard ICP,

$$\boldsymbol{T} = \underset{\boldsymbol{T} \in SE(3)}{\operatorname{argmin}} \sum_i \|\boldsymbol{b}_i - (\boldsymbol{R}\boldsymbol{a}_i + \boldsymbol{t})\|^2.$$

- **Point-to-plane ICP:**
  Letting $C_i^A = 0$ and $C_i^B = P_i^{-1}$, where $P_i = I - n_i n_i^\mathsf{T}$ projects onto the tangent plane at $\boldsymbol{b}_i$, gives the point-to-plane ICP cost,

$$\boldsymbol{T} = \underset{\boldsymbol{T} \in SE(3)}{\operatorname{argmin}} \sum_i (n_i^\mathsf{T}(\boldsymbol{b}_i - (\boldsymbol{R}\boldsymbol{a}_i + \boldsymbol{t})))^2.$$

Strictly speaking, P_i is singular and not invertible. Thus, it is approximated by an invertible $Q_i(\varepsilon) = P_i + \varepsilon I$, for $\varepsilon > 0$, ensuring $Q_i(\varepsilon) \to P_i$ as $\varepsilon \to 0$.

2.3.3 Plane-to-Plane (Fully Symmetric) GICP

The fully symmetric, plane-to-plane variant of GICP utilizes local surface structures from both point clouds. Each covariance C_i^A, C_i^B, reflects planar uncertainties, assigning small variance ϵ along normals and large variances in tangent directions. This symmetric approach enhances robustness, particularly in structured environments with clear planar features.

2.3.4 Feature Extraction Methods

Feature-based lidar odometry methods operate by identifying geometric features such as edges, corners, and planar surfaces, significantly reducing computational load and improving robustness compared to methods using entire point clouds [8]. Extracted features are matched across scans using nearest-neighbor searches, RANSAC-based techniques [9], or descriptor-based approaches. The optimal transformation is determined by minimizing distances between corresponding feature points.

These methods provide resilience against dynamic objects, sensor noise, sparse data, and occlusions. However, their effectiveness depends heavily on consistent feature extraction, which can be challenging in environments lacking distinctive features or exhibiting uniformity, potentially impacting odometry accuracy.

2.4 LiDAR-only Odometry

LiDAR-only odometry involves estimating the position and orientation of a vehicle or robot exclusively using LiDAR sensors. This approach primarily relies on aligning consecutive 3D LiDAR scans using algorithms such as Iterative Closest Point (ICP) and feature-based matching techniques. These methods are effective in structured environments due to the rich geometric information provided by LiDAR. However, LiDAR-only odometry may encounter difficulties in highly dynamic, sparse, or featureless environments, where data can be noisy or insufficiently dense, potentially compromising scan matching accuracy.

2.5 LiDAR-Inertial Odometry

LiDAR-inertial odometry fuses data from LiDAR additional sensors, typically an IMU, to improve localization accuracy and robustness. This combination leverages the complementary strengths of LiDAR (high accuracy, low frequency) and IMU (continuous but drift-prone, high-frequency data). Integrating these sensor modalities enhances performance in challenging environments [10], such as dynamic urban areas or rough terrain, enabling accurate state estimation even during rapid vehicle motion.

2.5.1 Loosely Coupled LiDAR-Inertial Odometry

In loosely coupled approaches, LiDAR and IMU data are processed separately before being fused at a higher level. Typically:

1. **LiDAR processing:** Relative motion estimates between successive scans are computed using ICP or feature-based methods.
2. **IMU processing:** IMU measurements are processed independently through inertial navigation algorithms or complementary filters to estimate orientation, velocity, and position.

These independent results are fused using filtering techniques, commonly Extended Kalman Filters (EKF) or Unscented Kalman Filters (UKF). In this architecture, IMU continuously estimates track motion, while LiDAR estimates periodically correct the accumulated drift from the IMU. For example, the LOAM (LiDAR Odometry and Mapping) system [11] incorporates IMU motion estimates to guide LiDAR scan matching, improving robustness by reducing uncertainty between scans. Nevertheless, LiDAR data remains the primary component for pose estimation.

2.5.2 Tightly Coupled LiDAR-Inertial Odometry

Tightly coupled methods integrate raw measurements from LiDAR and IMU sensors directly into a unified optimization or filtering framework. Rather than processing sensor data independently, these methods fuse sensor information at the measurement level, enabling simultaneous and continuous utilization of both sensors.

Typically, IMU data provides high-frequency predictions of system state between

LiDAR measurements. LiDAR data then corrects and refines these predictions by directly incorporating LiDAR points into the optimization cost function or filter update steps. Such direct integration gives robustness, particularly in challenging scenarios involving rapid motion, limited geometric features, or partial sensor occlusions [2].

Additionally, geometric observers [12] can be used within tightly coupled systems to further enhance robustness and accuracy by an observer that fuses position and orientation measurements with an IMU to generate a smooth trajectory [13].

Table 2.2 provides a consolidated overview of LiDAR odometry methods, categorized by their approaches such as direct, feature-based, or deep learning and the degree of sensor coupling involved. This summary highlights the evolution of techniques from classical ICP-based methods to modern tightly coupled and learning-based solutions.

Table 2.2: Unified overview of some LiDAR odometry methods [2]

Name	Year	Coupling	Method
Direct			
ICP [3]	1992	-	Iterative closest point (ICP), closest point matching.
Generalized-ICP [7]	2009	-	Combines point-to-point ICP and point-to-plane ICP in a probabilistic framework.
NICP [14]	2015	-	Extends Generalized-ICP by incorporating surface normals.
CT-ICP [15]	2022	-	Continuous-time ICP by interpolating positions.
KISS-ICP [16]	2023	-	Point-to-point ICP with adaptive thresholding.
FAST-LIO2 [17]	2022	Tightly	Registers point clouds directly using ikd-Tree without feature extraction.
DLIO [13]	2023	Tightly	Hierarchical geometrical observer for continuous-time state estimation.
Feature-based			
LOAM [11]	2014	Loosely	Extracts edge and planar feature points for registration.
LeGO-LOAM [8]	2018	Loosely	Uses ground segmentation within the LOAM framework.
DL-based			
LO-Net [18]	2019	-	Scan-to-scan LiDAR odometry neural network.

2.6 Deep Learning for LiDAR Odometry

Deep learning methods in lidar odometry represent an emerging and increasingly prominent direction [2], leveraging neural network architectures to automatically learn complex mappings between lidar scans and the associated transformations. Unlike traditional methods, deep learning approaches do not rely explicitly on predefined features or iterative optimization algorithms but instead exploit large datasets to learn robust and accurate motion estimation models. While deep learning-based approaches are emerging, they will not be the focus of this thesis. Nonetheless, it is important to acknowledge their existence, as they represent an active and growing area of research within LiDAR odometry. Deep learning methods typically require extensive labeled data and substantial computational resources for training. Additionally, interpretability and transparency remain concerns, as neural networks often function as "black-box" models with limited insight into their decision-making processes.

2.7 LiDAR Odometry Systems Overview

The literature on LiDAR odometry includes various methods broadly categorized into direct matching approaches, feature-based strategies, and tightly coupled sensor-fusion techniques. Three specific methods exemplify these distinct categories: the direct point-cloud matching method KISS-ICP, the feature-based, loosely coupled method LeGO-LOAM, and the tightly coupled LiDAR-inertial method DLIO. This section provides a concise description of these particular implementations.

2.7.1 KISS-ICP

KISS-ICP (Keep It Small and Simple ICP) [16] is a minimalist LiDAR odometry pipeline that revisits the classical point-to-point ICP algorithm and demonstrates that, with a few well-chosen enhancements, it can match or exceed the performance of more complex systems. Rather than relying on learned features or sensor fusion, KISS-ICP operates purely on raw 3D point clouds and employs a constant-velocity motion model for scan deskewing, adaptive thresholding for correspondence rejection, and a robust M-estimator [19] to mitigate outliers.
The pipeline consists of four main steps:

- **Motion Prediction and Deskewing:** Using the two most recent poses, translational and rotational velocities are estimated under a constant-velocity assumption. These velocities are then used to correct for intra-scan motion distortion by deskewing each point according to its relative timestamp within the sweep.
- **Double Voxel-Grid Subsampling:** To strike a balance between speed and accuracy, each incoming scan is first downsampled with a coarse voxel grid to form the merge set, and then further subsampled for registration. Unlike typical voxelization, subsampled points retain their original coordinates, instead of selecting the center of each voxel.

- **Adaptive Correspondence Threshold:** Rather than using a fixed maximum distance for nearest-neighbor matching, KISS-ICP computes a three-sigma bound on the observed scan-to-map displacement deviations, $\tau_t = 3\,\sigma_t$, where σ_t is the standard deviation of past corrections exceeding a minimum $\delta_{\min}$. This allows the algorithm to adapt online to varying motion profiles and scene dynamics.
- **Robust Point-to-Point Registration:** A Geman–McClure kernel [20] is applied within the ICP optimization to downweight outliers. Each iteration alternates between finding correspondences within the adaptive threshold and solving for the SE(3) update that minimizes the robustified point-to-point residuals, terminating when the pose correction falls below a small convergence bound.

2.7.2 Lightweight and Ground-Optimized LOAM (LeGO-LOAM)

LeGO-LOAM [8] is an extension of the LOAM framework tailored for real-time LiDAR odometry on ground vehicles, achieving high accuracy with reduced computational demands. Its key innovations are:

- **Range-Image Segmentation & Ground Separation:** Each incoming 3D scan is projected into a 2D range image. A column-wise ground-plane estimation labels ground returns, which are removed from the segmentation to prevent unreliable feature extraction on low-lying vegetation. Remaining points are clustered via an image-based method, and trivial clusters (fewer than 30 points) are discarded to focus on stable structures.
- **Feature Extraction:** The segmented point cloud is divided into horizontal sub-images (six segments of 300×16 for VLP-16). For each row in each sub-image, a roughness metric

$$c_i \;=\; \frac{1}{|S|\,\|r_i\|}\,\Big\|\sum_{j\in S, j\neq i}(r_j - r_i)\Big\|,$$

  is computed over the 10-point neighborhood S, where r_i and r_j denote the 3D coordinates of the center point and its neighbors respectively. Points with the highest c_i become *edge* features (excluding ground) and those with the lowest c_i become *planar* features (including ground).
- **Two-Step Optimization:** To reduce runtime, the 6-DOF pose between consecutive scans is estimated in two stages:
  (1) Match *planar* (ground) features to solve for vertical translation, roll and pitch.
  (2) With planar parameters fixed, match *edge* features to solve for horizontal translation and yaw.

  This decoupling yields similar accuracy to the original LOAM [11] two-algorithm fusion, while cutting odometry computation time by $\sim$35%.
- **Lightweight Mapping & Loop Closure:** Feature sets $\{F_e^t, F_p^t\}$ are stored per-scan, and a local map is built by selecting recent sets within a 100 m radius. A low-frequency mapping pass registers features to this map, refining

pose estimates. Optionally, LeGO-LOAM can integrate pose-graph SLAM with ICP-based loop closures using iSAM2[21] to eliminate drift over long trajectories.

By filtering out unreliable points early, leveraging ground constraints, and splitting the optimization, LeGO-LOAM runs in real time on embedded platforms (e.g., Jetson TX2) while achieving accuracy on par with or better than LOAM on standard benchmarks.

2.7.3 Direct LiDAR–Inertial Odometry (DLIO)

Direct LiDAR–Inertial Odometry (DLIO) [13] is a tightly-coupled LIO pipeline that achieves high-fidelity state estimation and dense mapping. DLIO's core innovation is a coarse-to-fine, continuous-time motion correction that enables per-point deskewing without iterative fitting, combined with a condensed scan-to-map registration and a provably-convergent hierarchical geometric observer [12].

Coarse IMU Integration & Fine Continuous-Time Deskewing

Aggressive maneuvers and uneven terrain induce severe motion distortion in spinning LiDAR scans. DLIO first integrates IMU measurements under a constant-jerk and constant-angular-acceleration model to produce a discrete pose sequence. For each LiDAR point p_k^n timestamped at $t_k + \Delta t_k^n$, an analytical, continous time solution is used to compute a unique SE(3) transform for each point, yielding per-point deskewing.

Keyframe–based Submapping

Rather than querying individual points within a fixed radius of the current pose, DLIO organizes the map as a sequence of *keyframes*, each associated with its raw LiDAR sweep. When constructing the local submap for scan–to–map registration, the algorithm selects a subset of these keyframes. Keyframes are chosen by:

- **Nearest–neighbor keyframes:** Select the k-NN keyframes to the current estimate.
- **Convex hull keyframes:** Extract L nearest keyframes that lie on the convex hull of all keyframe poses, ensuring that distant map boundaries remain represented.

Formally, if $\mathcal{K}$ is the set of all past keyframe poses, the submap $\widehat{S}_k$ is formed by concatenating the scans from

$$Q_k = \text{k-NN}(\mathcal{K},\, K), \quad H_k = \text{k-NN}(\text{hull}(\mathcal{K}), L).$$

To adapt to different environmental scales, DLIO uses adaptive keyframing. The insertion threshold for new keyframes is lowered in narrow areas to capture fine-grained features, whereas in open areas it increases to avoid redundancy.

Condensed Scan-to-Map Registration

Rather than performing an intermediate scan-to-scan alignment, DLIO embeds the continuous-time prior directly into a Generalized ICP (GICP) optimization that

registers each corrected scan to a local submap $\hat{S}_k^W$. The objective:

$$\Delta\hat{T}_k = \arg\min_{\Delta T} \sum_{c\in C} d_c^\top \left(C_{S_k,c} + \Delta T\, C_{P_k,c}\, \Delta T^\top\right)^{-1} d_c,$$

with $d_c = \hat{s}_{k,c} - \Delta T\, \hat{p}_{k,c}$ aligns scan and submap covariances in a plane-to-plane formulation (regularized to eigenvalues $(1, 1, \varepsilon)$).

Hierarchical Nonlinear Geometric Observer

The fused pose update $\Delta\hat{T}_k$ is passed to a two-stage geometric observer that guarantees global exponential convergence of orientation and translation estimates without iterative graph solvers. First, the spatial orientation estimate (a quaternion) is updated with a contracting observer ensuring fast attitude convergence; second, position, velocity, and IMU biases are corrected hierarchically [12]. This observer both initializes the next deskewing step and provides high-rate ($\geq$100 Hz) state outputs.

2.8 Loop Closure

In LiDAR-based localization, one often distinguishes between pure odometry (incrementally estimating motion from scan to scan) and a full Simultaneous Localization and Mapping (SLAM) system, which jointly builds or refines a map of the environment while estimating the sensor trajectory [22]. Loop closure, a core SLAM component, recognizes previously visited areas and injects constraints to correct accumulated drift in the estimated trajectory [23]. Many modern LiDAR odometry frameworks incorporate loop closure within a SLAM pipeline and adopt a graph-based formulation [24], in which each node denotes a sensor pose (often with its associated scan or submap) and each edge encodes a relative transformation—either from odometry or from loop-closure detection. To manage and optimize these graphs, such systems rely on graph-optimization libraries (e.g., GTSAM, g2o, Ceres, SE-Sync [24]), which offer efficient nonlinear least-squares solvers.

2.8.1 Graph Formulation

In a graph-based loop closure framework:

- **Nodes** represent poses (and optionally associated scans or submaps).
- **Odometry edges** connect sequential nodes, encoding relative motion derived from LiDAR odometry, inertial measurements, or their fusion.
- **Loop edges** are introduced when place recognition combined with geometric verification confirms a revisit.
- **Optimization** jointly adjusts all node poses by minimizing the total squared error over odometry and loop constraints, typically solved using nonlinear least-squares solvers provided by standard graph optimization libraries.

This abstraction facilitates seamless integration of heterogeneous sensor data (e.g., LiDAR, IMU, GPS) and supports large-scale, long-term mapping through techniques such as submap selection [25], marginalization [26], and hierarchical optimization [27].

2.8.2 Common Approaches with Loop Closure

CT-ICP [15] performs LiDAR-based loop detection by matching elevation-image projections in 2D, subsequently incorporating these loop constraints into its pose graph for global optimization. KISS-SLAM [28] identifies loop closures by extracting ground points, aligning the local map to the keypose's xy-plane, and projecting the resulting data into a bird's-eye-view (BEV) density grid. ORB descriptors [29] are computed from this 2D density image and matched against a database of previous keypose descriptors. Candidate loops undergo a RANSAC-based geometric validation step, producing a robust 2D alignment before insertion of validated loop edges into the graph. Similarly, frameworks such as LIO-SAM [30], LeGO-LOAM [8], and LOL [31] incorporate loop edges, often fused with additional sensor data like GPS, into their factor graphs for robust relocalization and drift correction.

2.9 Dynamic Object Extraction

Clustering, or object segmentation, is the crucial first step in dynamic LiDAR processing. By grouping points belonging to the same surface or object, we can isolate dynamic entities (e.g., vehicles, pedestrians) from the static environment. This separation not only prevents dynamic points from corrupting scan-to-map registration (thus in theory improving odometry accuracy) but also results in a cleaner global map that faithfully captures only the permanent structures, benefiting route planning and obstacle avoidance tasks.

An approach that relies on clustering is LeGO-LOAM [8], which directly adopts the fast range-image segmentation approach by Bogoslavskyi and Stachniss [32]. LeGO-LOAM first projects each LiDAR scan into a 2D range image. It then removes ground returns by detecting near-horizontal beams in each column, segmenting the remaining foreground clusters by comparing the incidence angle β between adjacent beams against a threshold θ. Connected components identified through this process correspond to individual objects or surfaces.

Crucially, LeGO-LOAM leverages this segmentation step explicitly for feature extraction: clusters identified as individual objects or surfaces are analyzed to classify their points into distinct planar and edge features. These extracted features serve as robust landmarks that enable precise scan-to-map matching and odometry estimation. By carefully distinguishing between planar surfaces and edges feature extraction is directly incorporated into scan registration.

Similarly, Removert [33] (Remove and Revert) employs range-image projection for dynamic object extraction. It projects both the transformed prior map and the current LiDAR scan into range images, flagging pixels as dynamic if their range differences exceed a predefined threshold. To avoid false positives in regions that appear differently due to viewpoint variations, Removert repeats the comparison at a lower resolution, reverting accidentally flagged static objects.

3
Method

3.1 Baseline Evaluation

In order to establish a reference method three LiDAR odometry methods are selected that span major algorithmic paradigms in the literature:

- **KISS-ICP** (ICP)
  A simple LiDAR-Odometry implementation based on classic Iterative Closest Point algorithm.
- **DLIO** (GICP)
  A tightly coupled LiDAR-Inertial Odometry approach that uses GICP.
- **LeGO-LOAM** (feature-based)
  A scan-matching system that extracts edge and planar features, performing frame-to-frame alignment in feature space and incorporating a lightweight map representation with loop-closure.

Each of these methods reflects a distinct design choice in LiDAR odometry: ICP, GICP, and feature based extraction. To select a baseline these are benchmarked on two complementary datasets in a pedestrian dynamic urban environment:

- **Square Loop**
  A single, closed-loop traverse of approximately 720 m, starting and ending at the same point. This compact circuit yields one clear loop-closure at the end of the run.
- **Out-and-back bridge**
  The platform departs from the start location, passes a narrow almost 100 m bridge and drives in total 450 m to a fixed turnaround, then retraces approximately the same path back, producing distinct outbound and inbound segments.

By benchmarking on the same datasets, we can isolate each paradigm, defining a robust baseline. These datasets are collected with the VLP-16, since its widely mentioned in LiDAR odometry litterature [8, 30, 34, 35].

3.2 LiDAR comparison

To evaluate and compare the performance of different LiDAR sensors in real-world conditions, datasets were collected using both the JT16 and VLP16 LiDAR sensors. Each dataset captures the same set of trajectories around Lindholmen to ensure consistency and comparability between the two platforms. The recorded trajectories

include only a single loop closure at the end, enabling the assessment of drift and loop closure accuracy. On average, the trajectories spans approximately 700 meters.

3.3 Graph-based Loop Closure

To account for drift over large distances a simple loop closure framework leveraging graph SLAM is implemented by interleaving keyframe selection, loop detection with pose-graph updates and map reconstruction.

3.3.1 Keyframe Selection

Deskewed LiDAR point clouds and pose outputs from the odometry node is subsampled into discrete keyframes. Each keyframe stores:

- A locally referenced timestamped point cloud.
- Its odometric pose estimate in SE(3).
- The cumulative path length along the trajectory.

3.3.2 Pose-Graph Formulation

The SLAM problem is formulated as a factor graph $G = \{\mathcal{X}, \mathcal{F}\}$, where:

- $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ is the set of pose variables, with each $X_i \in \mathrm{SE}(3)$ representing the 6-DOF pose of the sensor at keyframe i. Each pose is parameterized as a transformation matrix

$$X_i = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4}, \quad R_i \in \mathrm{SO}(3),\ t_i \in \mathbb{R}^3.$$

- $\mathcal{F}$ is the set of factors encoding relative pose constraints:
  1. **Prior factor:** Anchors the initial pose X_1 to the origin with high confidence (low covariance).
  2. **Odometry factors:** Between successive poses X_k and X_{k+1}, based on relative motion measurements $Z_{k,k+1} \in \mathrm{SE}(3)$. The residual is defined in the tangent space $\mathbb{R}^6$ as

$$r_{k,k+1} = \log\left(Z_{k,k+1}^{-1} X_k^{-1} X_{k+1}\right) \sim \mathcal{N}(0,\ \Sigma_{\mathrm{odom}}), \quad \Sigma_{\mathrm{odom}} \in \mathbb{R}^{6\times6},$$

     where $\log(\cdot)$ maps the pose difference to a minimal 6D vector (3 for rotation, 3 for translation), allowing a standard Gaussian noise model in the tangent space of SE(3).

Incremental optimization is performed with iSAM2, which updates only affected portions of the graph.

3.3.3 Loop Constraint Estimation

Given a candidate loop between the current keyframe X_i and a past keyframe X_j that passed spatial and temporal filters, we estimate a robust loop-closure constraint as follows:

1. **Submap Extraction.**
   Choose N and extract past keyframes,

$$\mathcal{M}_j = \{X_{j-N}, \ldots, X_j, \ldots, X_{j+N}\}, \quad \mathcal{M}_i = \{X_i\}.$$

   Note that the submap M_j, consisting of X_j and its adjacent keyframes, lies strictly within the temporal domain preceding X_i.
2. **Voxel Downsampling.**
   Apply a voxel-grid filter of leaf size v to both submaps, producing downsampled clouds $\tilde{\mathcal{M}}_j$ and $\tilde{\mathcal{M}}_i$.
3. **ICP Alignment.**
   Initialize point-to-point ICP with the odometric guess

$$T_{\text{odom}} = X_i^{-1} X_j,$$

   register $\tilde{\mathcal{M}}_i$ to $\tilde{\mathcal{M}}_j$, yielding $Z_{i,j} \in \mathrm{SE}(3)$ and fitness ε_{ICP}.
4. **Constraint Addition.**
   If ICP converges with $\varepsilon_{\text{ICP}} \leq \varepsilon_{\max}$, define the 6-DOF residual,

$$r_{i,j} = \log\Big(Z_{i,j}^{-1}(X_i^{-1} X_j)\Big) \in \mathbb{R}^6,$$

   and insert it into the graph with a Cauchy-robust kernel

$$r_{i,j} \sim \text{Cauchy}(0,\ \sigma^2 I_6), \quad \sigma^2 = \varepsilon_{\text{ICP}}.$$

   The purpose of this is to downweight any spurious loop measurements.

Each downsampled keyframe cloud is transformed by its optimized pose and merged into a global point-cloud map.

3.4 Range Image Point Cloud Segmentation

To impose structure on the LiDAR data, we follow the implementations of [8] and [32] by converting each 3D scan into a 2D range image and applying image-based segmentation to isolate individual clusters. This is one approach that is crucial for downstream tasks such as object tracking or dynamic object removal.

3.4.1 Data acquisition and image dimensions

Assuming a spinning LiDAR, for each laser beam, a distance measurement d together with a known vertical angle ϕ and a timestamped yaw angle θ are arranged in a 2D array of size

$$\underbrace{N_{\text{beams}}}_{\text{rows}} \times \underbrace{N_{\text{steps per 360°}}}_{\text{columns}},$$

where $N_{\text{beams}} \in \{16, 32, 64, 128\}$ is typical LiDAR verticle resolutions, and the number of columns corresponds to the horizontal resolution (e.g., magnitude of 1000 to 2000 steps per revolution).

3.4.2 Projection from a 3D Point Cloud

To enable efficient image-domain processing of LiDAR data, we project the 3D point cloud into a 2D range image by mapping each point's spherical coordinates into pixel indices.

1. For each 3D point (x, y, z), compute its spherical coordinates (r, ϕ, θ), where

$$r = \sqrt{x^2 + y^2 + z^2}, \quad \phi = \arcsin\left(\frac{z}{r}\right), \quad \theta = \text{atan2}(y, x).$$

2. Map ϕ and θ to image row r_r and column c_c using

$$r_r = \left\lfloor \frac{\phi - \phi_{\min}}{\Delta\phi} \right\rfloor, \quad c_c = \left\lfloor \frac{\theta - \theta_{\min}}{\Delta\theta} \right\rfloor.$$

3. At each pixel (r_r, c_c), store the minimum range r among all points falling into that bin.

Adjacent pixels correspond to adjacent beams, so spatial continuity in the scene is directly encoded in the 2D grid. All downstream operations (ground removal, segmentation) can be implemented using simple image-domain traversals, avoiding costly nearest-neighbor searches in 3D space. While the projection step adds to the per-scan computation, it still remains on the order of milliseconds and thus supports real-time operation.

3.4.3 Ground Removal

Before attempting to segment objects, it is beneficial to strip away all returns originating from the ground plane. Ground removal not only reduces the number of points to process downstream, but also prevents large, contiguous ground returns from merging distinct objects. The range image representation and the known mounting orientation is used to perform a fast, per-column ground estimation and removal.

Assumptions and Sensor Orientation

Under the assumption that the robot is approximately level with respect to the ground, and that the LiDAR pitch and roll relative to the local ground plane are known, each column of the range image corresponds to a fixed horizontal angle θ (azimuth), and the vertical index r corresponds to an increasing elevation angle ϕ. Ground returns will therefore appear as a smooth, piecewise planar sequence of pixels in each column, starting from the lowest row and upwards.

Per-column plane approximation

For each image column c:

1. Initialize at the bottom row $r = 1$ (lowest elevation beam).
2. Iteratively examine the next row $r + 1$: compute the slope of the range difference between beams r and $r + 1$. Equivalently, form the angle
$$\gamma = \arctan\left(\frac{d_{r+1}\,\sin(\Delta\phi)}{d_r - d_{r+1}\,\cos(\Delta\phi)}\right),$$
where $\Delta\phi$ is the known elevation angle between adjacent beams and d_r, d_{r+1} are the corresponding measured ranges.
3. Compare γ to a small threshold γ_{ground} (empirically on the order of $5-10°$). As long as $\gamma \leq \gamma_{\text{ground}}$, the two beams are considered to lie on the same planar surface (i.e., the ground), and you advance to the next row.
4. Stop once $\gamma > \gamma_{\text{ground}}$: all rows $r' \leq r$ in this column are marked as ground and excluded from further processing; rows $r' > r$ remain for object segmentation.

This per-column, 1D "scanline" procedure visits each pixel at most once, yielding an $O(N)$ complexity across the entire range image (with $N = N_{\text{beams}} \times N_{\text{steps}}$). In practice, ground removal adds only a fraction of a millisecond per scan.
By working in the range image, the method naturally handles both sparse (e.g., 16-beam) and dense (64-beam) scans without additional tuning of voxel sizes or grid parameters. Only a single threshold γ_{ground} must be chosen (guided by the vertical beam spacing). Having stripped away ground returns, the resulting “clean” range image contains only those beams that hit potential objects above the ground [32].

3.4.4 Point cloud Segmentation

With ground points removed, segmentation of the remaining range image into individual object clusters is done by exploiting the implicit 2D neighborhood and a simple angular criterion.

Defining connectivity via the β–θ criterion

For any two adjacent pixels (r, c) and (r_n, c_n) in the range image (using an N_4 neighborhood: up, down, left, right), compute the angle

$$\beta = \arctan\left(\frac{d_{\min}\,\sin\alpha}{d_{\max} - d_{\min}\,\cos\alpha}\right),$$

where

$$d_{\max} = \max\left(d_{r,c},\, d_{r_n,c_n}\right), \quad d_{\min} = \min\left(d_{r,c},\, d_{r_n,c_n}\right),$$

and α is the known angular separation between the two beams. If $\beta > \theta$ (with θ typically set to 10° [32]), the two pixels are deemed to belong to the same object; otherwise they lie on different objects.

Connected-component labeling (CCL) via BFS

1. Initialize a label image L of the same size as the range image R, filled with zeros. Set current label to equal 1.
2. Scan R left-to-right, top-to-bottom. Whenever you encounter a pixel (r, c) with $L(r, c) = 0$ and a valid (non-ground) range:
   (a) Push (r, c) onto a queue.
   (b) While the queue is non-empty:
   - Pop (r', c'), set $L(r', c') =$ current_label.
   - For each N_4 neighbor (r_n, c_n) with $L(r_n, c_n) = 0$, compute β as above; if $\beta > \theta$, push (r_n, c_n) onto the queue.
3. After the BFS completes, increment current label and continue scanning for the next unlabeled pixel.

This pass-through CCL visits each pixel at most twice, once when first encountered, and once during neighbor checks, achieving $O(N)$ worst-case complexity (with N the number of valid pixels).

Back-projection to 3D clusters

After labeling, each connected component in the range image corresponds to one segment. We reconstruct the 3D points of segment k by back-projecting all pixels (r, c) with $L(r, c) = k$ into (x, y, z) space using the inverse spherical projection as in Section 3.4.2. This results in a fast fully image-based, no nearest-neighbor searches in 3D, that works equally well on both dense (64-beam) and very sparse (16-beam) data, since connectivity is defined in the 2D image domain [32].

3.4.5 Dynamic Object Removal

The dynamic object removal method presented here is inspired by the Removert framework [33]. It identifies and removes dynamic objects from point-cloud maps based on differences observed in sequential range images.

Ground Removal

All ground points are segmented from the point cloud using a previously mentioned ground-removal algorithm (as detailed in Section 3.4.3) and stored separately. Ground points exhibit high variability in range values even for small incidence-angle differences [33], and thus are preserved and will be reinstated after dynamic object removal.

Difference Image Computation

For each new LiDAR scan acquired within a small defined spatial distance d_Δ, both the current map cloud and the scan are transformed to the same reference frame. The resulting point clouds are then projected onto range images of equal resolution. Let $I_M(u, v)$ and $I_S(u, v)$ represent the map and scan range images, respectively. The absolute difference image I_Δ is computed as

$$I_\Delta(u, v) = \Big| I_M(u, v) - I_S(u, v) \Big|.$$

Pixels exceeding a predefined range-difference threshold $\tau_{u,v}$ generate a binary mask of dynamic regions,

$$M(u, v) = \begin{cases} 1, & I_\Delta(u, v) > \tau_{u,v}, \\ 0, & \text{otherwise.} \end{cases}$$

To mitigate isolated noise, an optional median filter with a $k \times k$ kernel can be applied to smooth range images, preserving significant structural differences while removing spurious pixels (Figure 3.1).

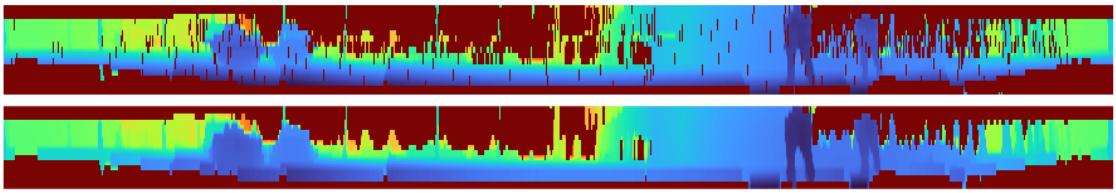


Figure 3.1: Range image before (top) and after (bottom) median filtering ($k = 5$)

Dynamic Pixel Indexing and Back-projection

Pixels marked as dynamic in the binary mask $M(u, v)$ are back-projected into 3D, yielding a set of candidate point indices,

$$I_D = \{(u, v) \mid M(u, v) = 1\}.$$

Dynamic Point Removal and Map Update

The indices in I_D are excised from the map. Let $\mathcal{D}$ be the set of keyframe point indices to be removed. The cleaned static map K_{clean} is then formed by

$$K_{\text{clean}} = \{k_i\}_{i=1}^{N} \setminus \{k_i \mid i \in \mathcal{D}\} \cup G,$$

where G is the set of stored ground points reintegrated after removal. Finally, the updated map cloud K_{clean} is reprojected back into the 3D point cloud frame and inserted back into the map.

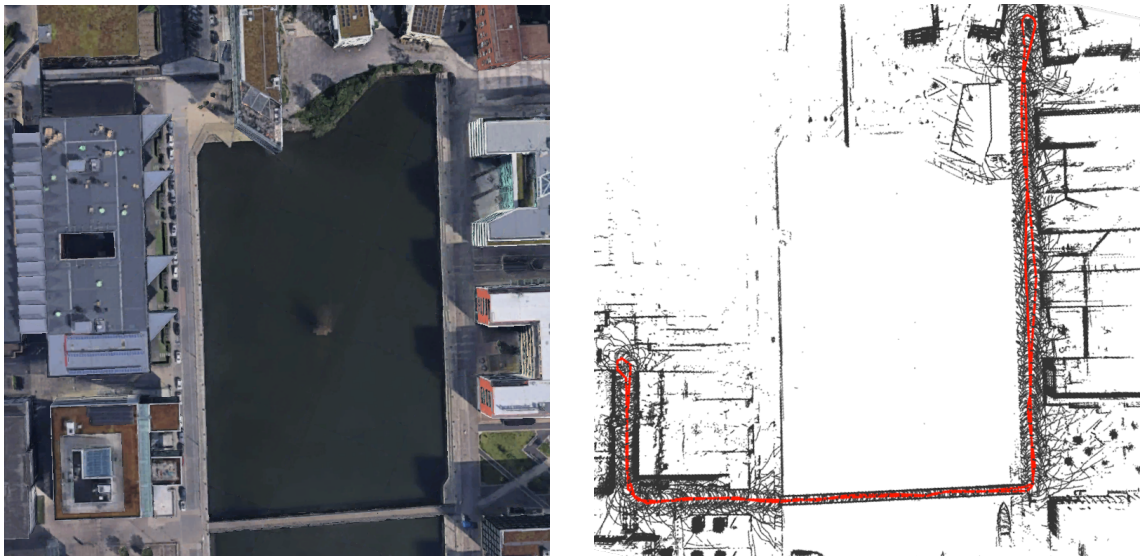
4

Results

This chapter presents both quantitative and qualitative results that compare various LiDAR odometry frameworks and assess specific enhancements implemented within the DLIO system. Key areas of evaluation include the impact of dynamic object removal, the integration of a loop closure module, and performance differences across the two LiDAR sensors. In addition, qualitative results are provided for range image–based segmentation.

4.1 Evaluation Data

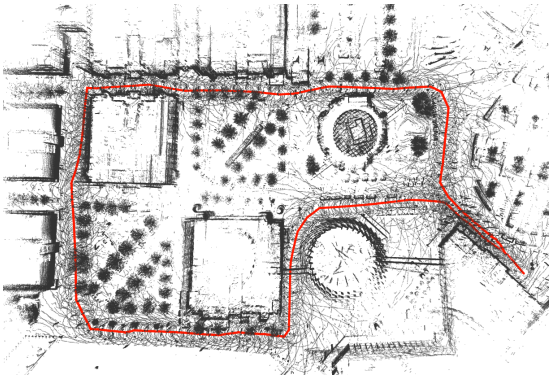
Figures 4.1 and 4.2 show satellite imagery alongside the LiDAR odometry trajectories using DLIO for the Out-and-Back Bridge and Big Square routes.



(a) Satellite image **(b)** DLIO trajectory

Figure 4.1: Out-and-Back Bridge: Satellite image and DLIO trajectory

(a) Satellite image



(b) DLIO trajectory

Figure 4.2: Big Square: Satellite image and DLIO trajectory

Due to lack of an accurate ground truth end-to-end translational error is used as a metric to evaluate baseline. Distance d is calculated between starting pose $(0, 0, 0)$ to the end pose of each framework. The translational error solely in the XY-plane is also calculated as d_{xy}[1][2] The table below shows that DLIO achieves by far the lowest end-to-end translational drift on both the Big Square and Out-and-Back Bridge trajectories.

Table 4.1: End-to-End Translational errors (m) for each algorithm

	Big Square					**Out-and-Back Bridge**				
Algorithm	x	y	z	d_{xyz}	d_{xy}	x	y	z	d_{xyz}	d_{xy}
KISS-ICP	2.269	2.277	9.410	9.944	3.215	−2.073	−11.459	58.207	59.360	11.645
LeGO-LOAM	−16.453	−2.584	15.571	22.800	16.655	−8.485	−0.108	0.228	8.489	8.486
DLIO	0.406	−1.626	2.213	3.624	1.675	0.045	−0.080	0.001	0.092	0.092

Specifically, DLIO exhibits the lowest translational drift on both routes. Conversely, LeGO-LOAM shows the largest error on the Big Square, and KISS-ICP suffers from significant vertical drift. Appendix A presents all three approaches with Google Earth imagery of each test area. Figures A.1a–A.1d show the Out-and-Back Bridge route, and Figures A.2a–A.2d show the Big Square route, with Satellite Image, KISS-ICP, LeGO-LOAM, and DLIO results side by side. Notably, the most pronounced discrepancy is observed in the Out-and-Back Bridge case, where LeGO-LOAM's truncation of the bridge path is particularly evident. LeGO-LOAM truncates the bridge length, whereas point-to-point methods do not; this behavior stems from its repetitive, feature-sparse representation, which can lead to spurious associations.

[1] `https://github.com/PRBonn/kiss-slam/issues/6`.
Original author of KISS-ICP discussing "pitch-drift".

[2] `https://github.com/RobustFieldAutonomyLab/LeGO-LOAM/issues/154`.
Original author of LeGO-LOAM, LIO-SAM discussing the same problem.

4.2 DLIO Results

This section further investigates the DLIO framework by examining specific enhancements, including dynamic object removal, integration of loop closure, and sensor-specific performance comparisons. These evaluations aim to highlight the improvements each modification provides.

4.2.1 Impact of Dynamic Object Removal and Loop Closure

To examine improvements offered by dynamic object removal and loop closure integration, DLIO variants were tested on the most dynamic dataset (i.e., the one with the highest pedestrian-crossing frequency across all scans) collected at Lindholmen. Results of DLIO-Static (removal of dynamic objects), DLIO-LC (incorporating loop closure), and standard DLIO with and without point cloud voxelization, are displayed in Table 4.2 below.

Table 4.2: End-to-End Translational Errors (m) for DLIO Variants

	DLIO				**DLIO-Voxelized**			
	|x|	|y|	|z|	d_{xyz}	|x|	|y|	|z|	d_{xyz}
Error (m)	2.212	1.170	8.289	8.658	0.406	1.626	0.0397	1.676
	DLIO-Static				**DLIO-LC**			
	|x|	|y|	|z|	d_{xyz}	|x|	|y|	|z|	d_{xyz}
Error (m)	0.625	3.876	0.026	3.926	0.023	0.007	0.000	0.024

The DLIO-LC variant significantly outperformed the others, achieving the lowest overall translational error. By comparison, the dynamic-object removal method performed slightly worse than DLIO with voxelization enabled, yet it achieved the second-lowest z-error. Detailed inspections of the point clouds (see Appendix Figures B.1 and B.2) reveal a marked reduction in map misalignment after loop closure: duplicated structures, vegetation clusters, and other landmarks become correctly realigned. BEV overlays of the complete point cloud map in Fig. B.4 (overlaid on a satellite image) and Fig. B.3 (without background) offer a straightforward visualization of map coverage relative to real-world landmarks.

4.2.2 Performance Across Different LiDAR Sensors

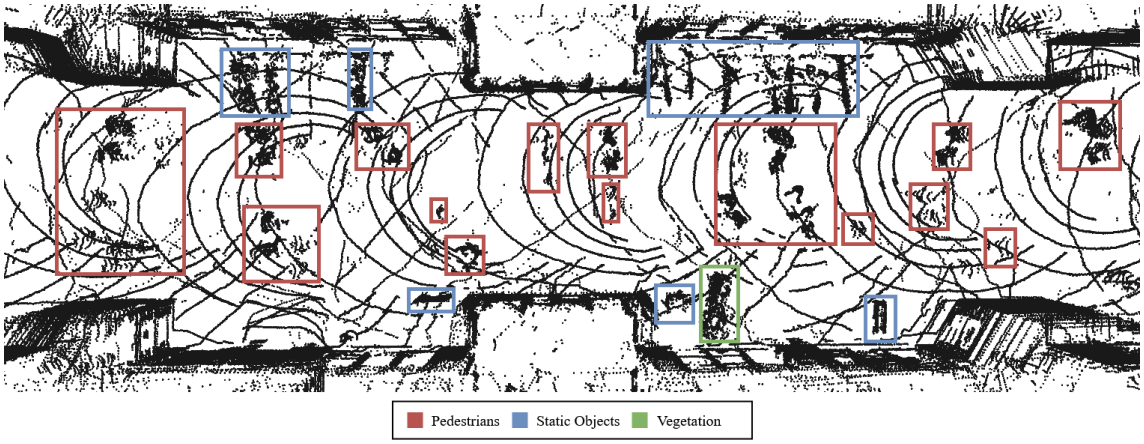
Experiments comparing performance with different LiDAR sensors, JT16 and VLP16, were also conducted to assess sensor-specific influences. Table 4.3 presents averaged translational errors across multiple self collected datasets. Loop closure integration is included due to yielding substantially lower errors compared to DLIO alone.

Table 4.3: End-to-End Translational Errors (m) for JT16 and VLP16 using DLIO Variants

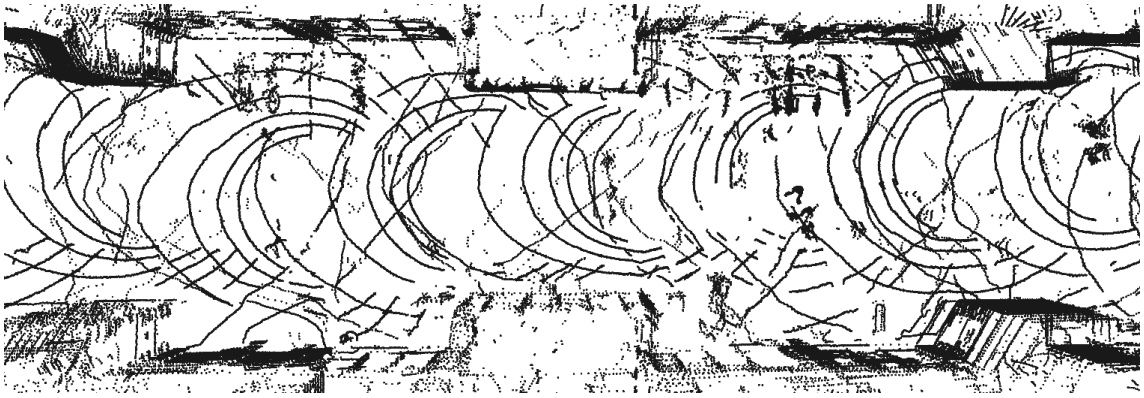
	JT16				**VLP16**			
Method	|x|	|y|	|z|	d_{xyz}	|x|	|y|	|z|	d_{xyz}
DLIO-Voxelized	5.194	5.478	7.20	12.325	0.156	1.325	0.039	1.335
DLIO-LC	0.098	0.046	0.024	0.111	0.019	0.011	0.001	0.022

4.3 Qualitative Evaluation: Dynamic Object Removal

Figure 4.3 shows the result of dynamic object removal applied in a narrow street scenario, with multiple pedestrians. The method operated in real time and removes dynamic objects from the environment representation. While some false positives may be removed, the overall structure of the scene is preserved. Mostly narrow or small objects are detected as false positive. Overall, the approach retains substantially more points compared to voxelization of the entire point cloud, preserving on 70.02% of the points versus 34.9% retained by voxelization.



(a) Before removal



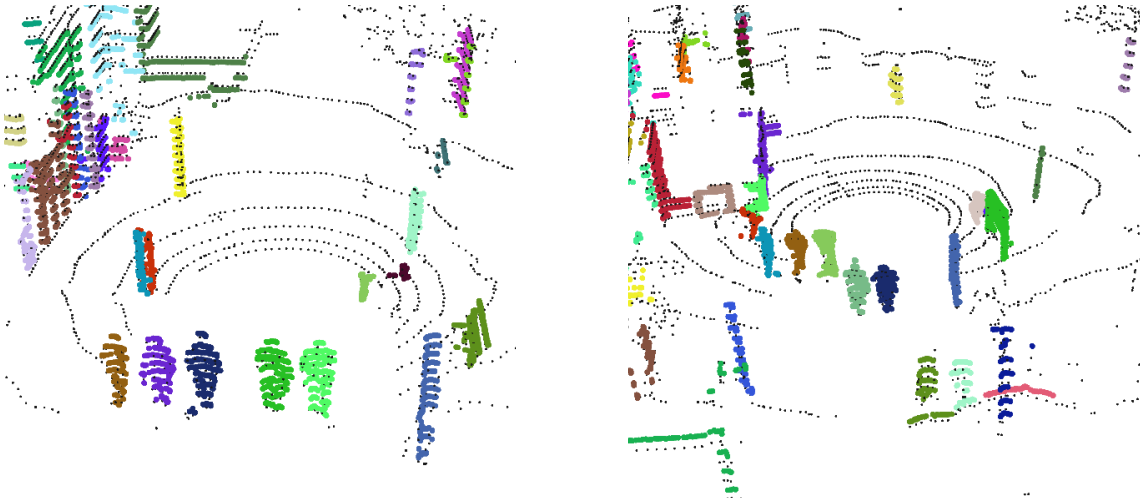
(b) After removal

Figure 4.3: BEV comparison of dynamic object removal in a narrow street

More visualizations illustrating dynamic object removal across various scenes are presented in Appendix C.

4.3.1 Qualitative Evaluation: Range-Image based clustering

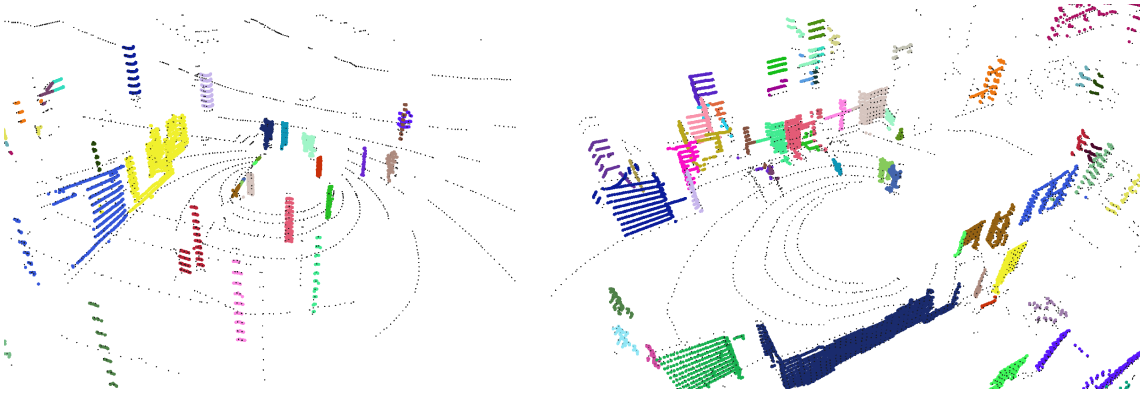
Building on the dynamic-object removal results (Fig. 4.3), we apply range-image–based clustering (re-implemented from LeGO-LOAM's feature extraction) to the same dataset within the DLIO pipeline.



(a) Cluster view 1 **(b)** Cluster view 2

Figure 4.4: Cluster-visualizations 1 & 2

Figures 4.4(a)–(b) and 4.5(a)–(b) show four different perspectives of the clustered point cloud, where each color denotes a distinct cluster, and black points contains the voxelized map scan.



(a) Cluster view 3 **(b)** Cluster view 4

Figure 4.5: Cluster-visualizations 3 & 4

In Figures 4.4a and 4.4b, many vertical, pole-like structures appear as coherent, narrow clusters, indicating that the method successfully isolates thin vertical features. Pedestrian clusters are also correctly separated, appearing centrally in these views. Figure 4.5b highlights larger planar surfaces such as building facades or wall segments. In this view, even distant pedestrian clusters can be seen near the edges of the scene.

5
Conclusion

This thesis was out to evaluate and enhance LiDAR-based odometry for compact, pedestrian-scale delivery robots operating in dynamic urban settings. First, it compared some state-of-the-art LiDAR odometry methods across metrics of trajectory accuracy, real-world motion resilience to identify a robust baseline . Building on that baseline, the work then integrated loop-closure adjustments to correct drift over extended runs. Lastly implemented a real-time, range image–based dynamic-object filter to excise moving elements from the map.

The baseline odometry approach demonstrated high accuracy across short and moderate runs but experienced some drift over prolonged trajectories. To address this, a graph-based loop-closure via ICP was applied, effectively correcting accumulated errors and stabilizing localization on longer paths. A separate real-time filter, driven by range-images, excised moving objects to preserve a clean global map. Additionally, the same range-image clustering method from LeGO-LOAM demonstrated potential for tracking individual clusters over time, being both fast and effective at distinguishing, for instance, pedestrians.

5.1 Discussion

The central question guiding this work was whether a LiDAR-based odometry pipeline for pedestrian-scale delivery robots can be made sufficiently accurate and robust in dynamic urban settings without relying on GPS. In the following discussion, we analyze how each component (baseline odometry, loop closure, dynamic-object removal, and range-image clustering) contributes to addressing this question under real-world conditions.

5.1.1 Baseline Comparative Analysis

In urban deployments, it is not surprising that robust odometry demands sophisticated approaches. KISS-ICP, for instance, assumes a constant-velocity motion model and uses distance-based submapping potentially holding millions of points that may not correspond correctly across scans, making it less efficient than DLIO leveraging GICP, IMU coupling and k-NN keyframe based submapping, which delivers superior odometry performance. In feature-driven approaches, LeGO-LOAM extracts edge and planar features to represent the environment sparsely; however, in repetitive feature sparse or under LiDAR occlusion, similar-looking features across different locations can lead to ambiguous associations.

5.1.2 LiDAR comparison

The comparative evaluation between the JT16 and VLP16 LiDAR sensors highlights several important considerations for odometry and mapping tasks. One of the most prominent limitations of the JT16 is its significantly reduced effective range. This limit impacts the ability to observe distant, stable features for robust motion estimation.

Interestingly, voxelization-based methods yielded improved performance, suggesting that horizontal resolution may not be as critical. Voxelization will tend to downsample more heavily in the horizontal direction, because more horizontally-close points will fall into the same voxel. This causes the VLP16 to end up in similar resolution ranges as the JT16. Due to its above-horizon beam configuration, the JT16 suffers from a lack of ground returns. This deficiency in near-ground observations can negatively affect trajectory estimation, especially in the vertical (z) direction since ground points could serve as reliable anchors.

5.1.3 Addition of loop closure

Integrating a loop-closure module into the odometry pipeline yields clear benefits by detecting when the robot revisits previously mapped areas and correcting accumulated relative drift through pose-graph optimization. This correction significantly improves consistency over trajectories that contain loops, reducing long-term error without additional manual intervention. However, loop closure provides no benefit on strictly linear or outward-bound paths, since no revisit events occur to trigger loop detection. Furthermore, while loop closure enhances relative accuracy, it does not establish an absolute reference frame; without external anchors such as GPS fixes, known landmarks, or global pose estimates, residual drift and map misalignment can persist over large-scale deployments. Therefore, for comprehensive localization robustness, loop-closure corrections must be complemented by other anchoring sources or absolute positioning methods.

5.1.4 Dynamic Object Removal

Online dynamic-object removal was implemented via a range-image–based filter, demonstrating the feasibility of excising moving elements during traversal. However, this approach incurs high amounts of false positives,especially under heavy occlusion,when static structures are misclassified as dynamic. To date, its direct benefit for improving odometry accuracy compared to using voxelization, remains inconclusive: no consistent reduction in pose error was observed in our experiments.

For applications that require a clean map but not real-time updating, offline post-processing of LiDAR data offers a better alternative, as of the original implementation of Removert. By filtering dynamic points after data collection, one could potentially leverage more sophisticated algorithms and global context to achieve better map cleanliness.

Where online filtering could be valuable is in scenarios where the LiDAR map feeds into downstream planning modules. In such cases, dynamically removing moving objects can prevent planners from treating transient obstacles as permanent, potentially enabling more efficient trajectory generation over previously traversed areas. Results show the overall map is still somewhat representative, and could potentially be used.

5.1.5 Range-Imaged based clustering

Although range-image–based clustering itself is not novel, it is the foundation of LeGO-LOAM's feature extraction, this work evaluated it due to proven computational efficiency compared to standard Euclidean clustering. By leveraging the organized structure of the range image, clusters can be formed more quickly and with lower processing overhead than for instance euclidean clustering in 3D point clouds. While these clusters were not used here for feature-driven odometry, they offer a promising first step toward real-time dynamic-object tracking: instead of merely excising moving points, the same range-image information generated by the range-image dynamic object removal can be retained.

5.2 Concluding remarks and future work

This thesis demonstrates that even a robust LiDAR-odometry pipeline can be augmented to meet the challenges of dynamic urban environments by "picking the raisins out of the cookie", that is, cherry-picking modules from existing methods. Starting from a reliable baseline, a simple graph based loop-closure based on ICP was added to compensate relative drift over long trajectories, a capability DLIO lacked. Second, although LeGO-LOAM employs range images for edge-and-plane feature extraction, we leverage range-images to instead remove moving objects in real time to keep the global map clean. In addition, range-image clustering used by LeGO-LOAM proved a lightweight way to generate object clusters that could serve future tracking or planning tasks. Together, these enhancements show how combining selective components, rather than reinventing the wheel, yields a more extensive framework.

Looking ahead, the next step is to turn range-image clusters into true dynamic-object trackers, maintaining identities and motion estimates over time. Such tracking would enable the robot not only to ignore moving obstacles but also to predict their paths for safer navigation. Another promising direction is to introduce occasional GPS or other absolute-position measurements into the pose-graph optimization, anchoring the map to a global frame. Even sparse, intermittent GPS fixes could correct residual drift on straight, loop-free routes and further improve long-distance localization accuracy. Moreover, global localization using satellite imagery or georeferenced maps represents a compelling direction for research. With an initial GPS-based pose estimate, the robot's locally accurate LiDAR points could be aligned to overhead satellite images or vector maps by matching large-scale geometric features such as building outlines, road boundaries, and open spaces. This could enable map-to-map alignment and provide a globally consistent frame even in GNSS-challenged environments.

Bibliography

[1] Mohammed Khader and Shine Cherian. *An Introduction to Automotive LiDAR*. https://www.ti.com/lit/wp/slyy150d/slyy150d.pdf?ts=1748396073117. 2020.

[2] Dongjae Lee et al. *LiDAR Odometry Survey: Recent Advancements and Remaining Challenges*. 2023. arXiv: 2312.17487 [cs.RO]. URL: https://arxiv.org/abs/2312.17487.

[3] P.J. Besl and Neil D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.

[4] Kok-Lim Low. "Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration". In: (Jan. 2004).

[5] P.J. Besl and Neil D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.

[6] John C Gower and Garmt B Dijksterhuis. *Procrustes Problems*. Oxford University Press, Jan. 2004. ISBN: 9780198510581. DOI: 10.1093/acprof:oso/9780198510581.001.0001. URL: https://doi.org/10.1093/acprof:oso/9780198510581.001.0001.

[7] Aleksandr Segal, Dirk Hähnel, and Sebastian Thrun. "Generalized-ICP". In: June 2009. DOI: 10.15607/RSS.2009.V.021.

[8] Tixiao Shan and Brendan 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.

[9] Jonathan Lichtenfeld, Kevin Daun, and Oskar von Stryk. *Efficient Dynamic LiDAR Odometry for Mobile Robots with Structured Point Clouds*. 2024. arXiv: 2411.18443 [cs.RO]. URL: https://arxiv.org/abs/2411.18443.

[10] Prashant Anand Srivastava. "Demystifying Sensor Fusion and Multi-Modal Perception in Robotics". In: *European Journal of Computer Science and Information Technology* 13.26 (Apr. 2025), pp. 76–90. ISSN: 2054-0965. DOI: 10.37745/ejcsit.2013/vol13n267690. URL: http://dx.doi.org/10.37745/ejcsit.2013/vol13n267690.

[11] Ji Zhang and Sanjiv Singh. "LOAM : Lidar Odometry and Mapping in real-time". In: *Robotics: Science and Systems Conference (RSS)* (Jan. 2014), pp. 109–111.

[12] Brett T. Lopez. *A Contracting Hierarchical Observer for Pose-Inertial Fusion*. 2023. arXiv: 2303.02777 [cs.RO]. URL: https://arxiv.org/abs/2303.02777.

[13] Kenny Chen, Ryan Nemiroff, and Brett T. Lopez. *Direct LiDAR-Inertial Odometry: Lightweight LIO with Continuous-Time Motion Correction*. 2023. arXiv: 2203.03749 [cs.RO]. URL: https://arxiv.org/abs/2203.03749.

[14] Jacopo Serafin and Giorgio Grisetti. "NICP: Dense normal based point cloud registration". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 742–749. DOI: 10.1109/IROS.2015.7353455.

[15] Pierre Dellenbach et al. "CT-ICP: Real-time Elastic LiDAR Odometry with Loop Closure". In: *CoRR* abs/2109.12979 (2021). arXiv: 2109.12979. URL: https://arxiv.org/abs/2109.12979.

[16] Ignacio Vizzo et al. "KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way". In: *IEEE Robotics and Automation Letters* 8.2 (Feb. 2023), pp. 1029–1036. ISSN: 2377-3774. DOI: 10.1109/lra.2023.3236571. URL: http://dx.doi.org/10.1109/LRA.2023.3236571.

[17] Wei Xu et al. "FAST-LIO2: Fast Direct LiDAR-inertial Odometry". In: *CoRR* abs/2107.06829 (2021). arXiv: 2107.06829. URL: https://arxiv.org/abs/2107.06829.

[18] Qing Li et al. *LO-Net: Deep Real-time Lidar Odometry*. 2020. arXiv: 1904.08242 [cs.CV]. URL: https://arxiv.org/abs/1904.08242.

[19] Fumio Hayashi. *Econometrics*. Princeton, NJ: Princeton University Press, 2000. Chap. Extremum Estimators. ISBN: 0-691-01018-8.

[20] Nived Chebrolu et al. *Adaptive Robust Kernels for Non-Linear Least Squares Problems*. 2021. arXiv: 2004.14938 [cs.RO]. URL: https://arxiv.org/abs/2004.14938.

[21] Michael Kaess et al. "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.

[22] H. Durrant-Whyte and T. Bailey. "Simultaneous localization and mapping: part I". In: *IEEE Robotics & Automation Magazine* 13.2 (2006), pp. 99–110. DOI: 10.1109/MRA.2006.1638022.

[23] Xu Xu et al. "When-to-Loop: Enhanced Loop Closure for LiDAR SLAM in Urban Environments Based on SCAN CONTEXT". In: *Micromachines* 15 (Sept. 2024), p. 1212. DOI: 10.3390/mi15101212.

[24] Anđela Jurić et al. “A Comparison of Graph Optimization Approaches for Pose Estimation in SLAM”. In: *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*. 2021, pp. 1113–1118. DOI: 10.23919/MIPRO52101.2021.9596721.

[25] Kai Ni, Drew Steedly, and Frank Dellaert. “Tectonic SAM: Exact, Out-of-Core, Submap-Based SLAM”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 1678–1685. DOI: 10.1109/ROBOT.2007.363564.

[26] T. Bailey and H. Durrant-Whyte. “Simultaneous localization and mapping (SLAM): part II”. In: *IEEE Robotics & Automation Magazine* 13.3 (2006), pp. 108–117. DOI: 10.1109/MRA.2006.1678144.

[27] Giorgio Grisetti et al. “Hierarchical optimization on manifolds for online 2D and 3D mapping”. In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 273–278. DOI: 10.1109/ROBOT.2010.5509407.

[28] Tiziano Guadagnino et al. *KISS-SLAM: A Simple, Robust, and Accurate 3D LiDAR SLAM System With Enhanced Generalization Capabilities*. 2025. arXiv: 2503.12660 [cs.RO]. URL: https://arxiv.org/abs/2503.12660.

[29] Ethan Rublee et al. “ORB: an efficient alternative to SIFT or SURF”. In: Nov. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[30] Tixiao Shan et al. “LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5135–5142. DOI: 10.1109/IROS45743.2020.9341176.

[31] Dávid Rozenberszki and Andras Majdik. “LOL: Lidar-Only Odometry and Localization in 3D Point Cloud Maps”. In: *CoRR* abs/2007.01595 (2020). arXiv: 2007.01595. URL: https://arxiv.org/abs/2007.01595.

[32] Igor Bogoslavskyi and Cyrill Stachniss. “Fast range image-based segmentation of sparse 3D laser scans for online operation”. In: Oct. 2016, pp. 163–169. DOI: 10.1109/IROS.2016.7759050.

[33] Giseop Kim and Ayoung Kim. “Remove, then Revert: Static Point cloud Map Construction using Multiresolution Range Images”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 10758–10765. DOI: 10.1109/IROS45743.2020.9340856.

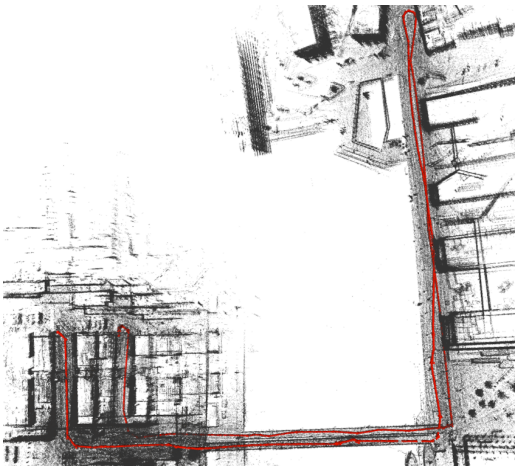
[34] Darwin Mick et al. *LiPO: LiDAR Inertial Odometry for ICP Comparison*. 2024. arXiv: 2410.08097 [cs.RO]. URL: https://arxiv.org/abs/2410.08097.

[35] Austin Nicolai et al. “Deep Learning for Laser Based Odometry Estimation”. In: 2016. URL: https://api.semanticscholar.org/CorpusID:30369588.

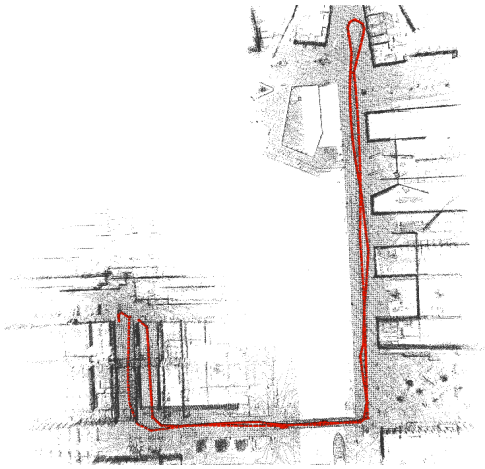
A

BEV Satellite Imagery and LiDAR Odometry Trajectories

(a) Google Earth satellite image



(b) KISS-ICP



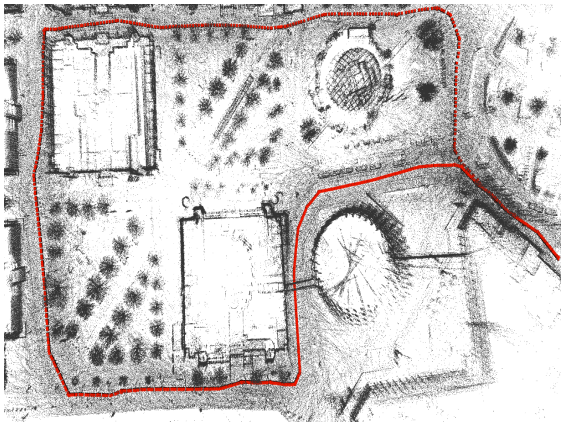
(c) LeGO-LOAM

(d) DLIO

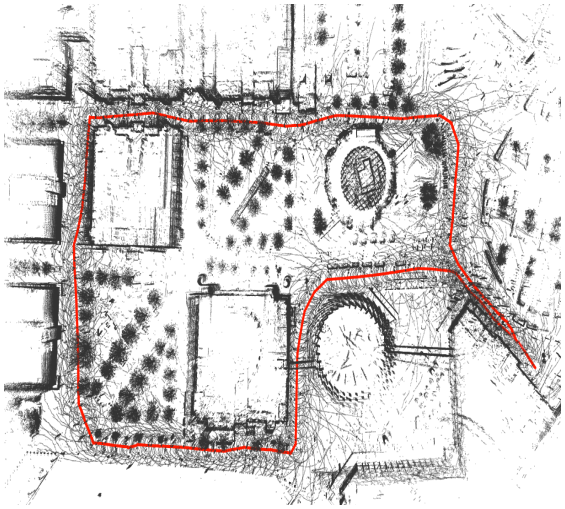
Figure A.1: BEV: Out-and-back bridge

(a) Google Earth satellite image

(b) KISS-ICP



(c) LeGO-LOAM



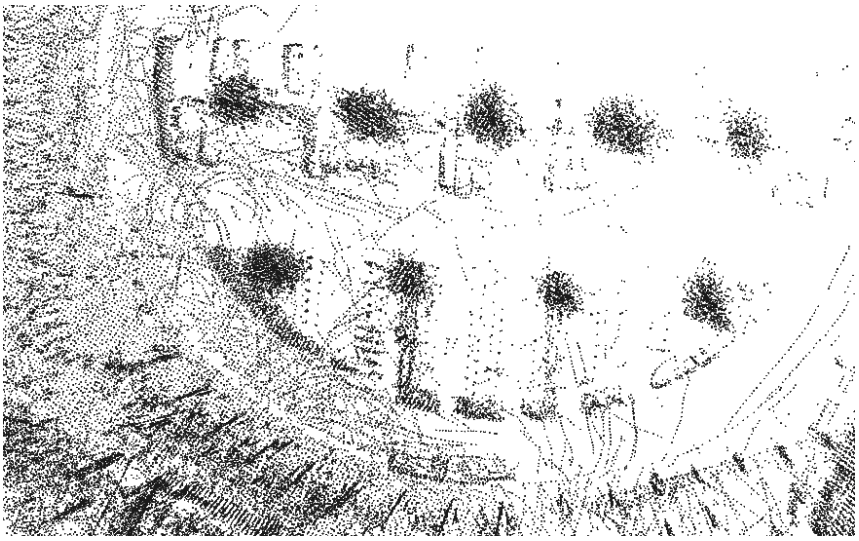
(d) DLIO

Figure A.2: BEV: Big square

B

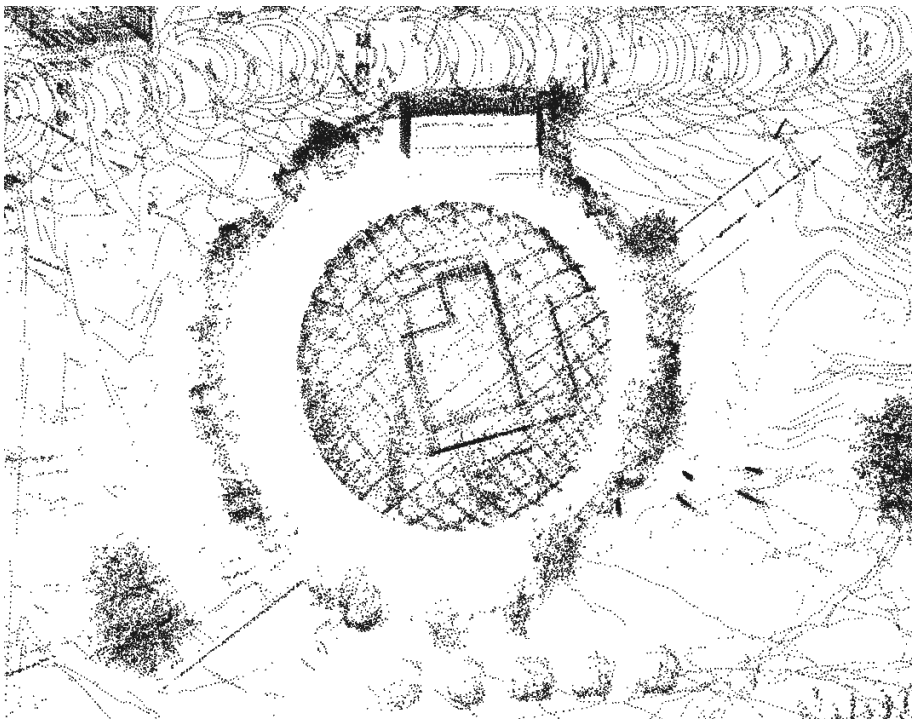
Point Cloud Maps Before & After Loop Closure

(a) Before Loop closure

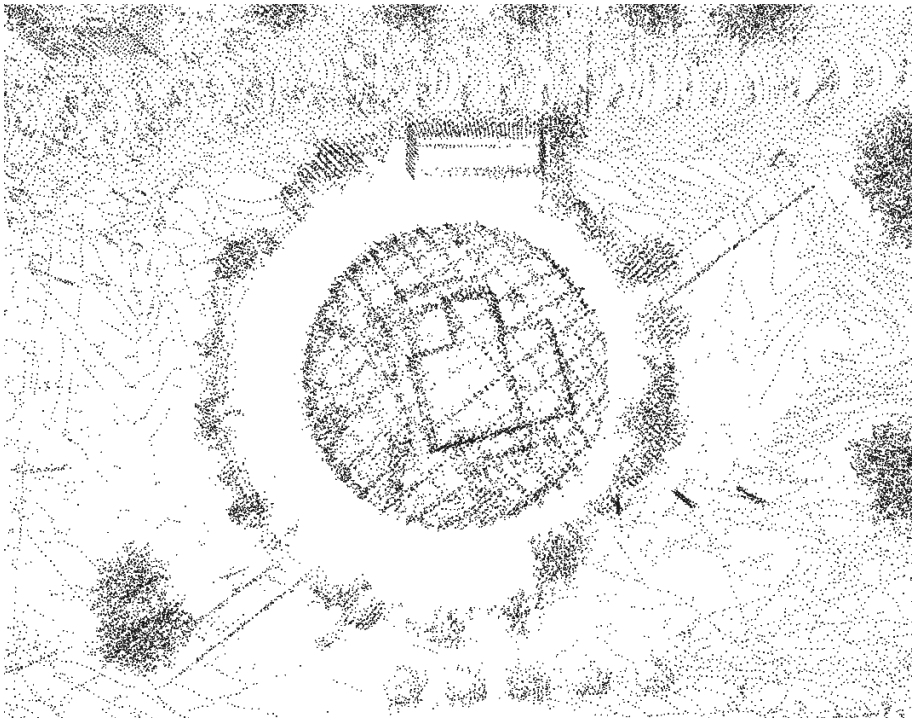


(b) After loop closure

Figure B.1: Point cloud alignment before and after loop closure of nearby trees

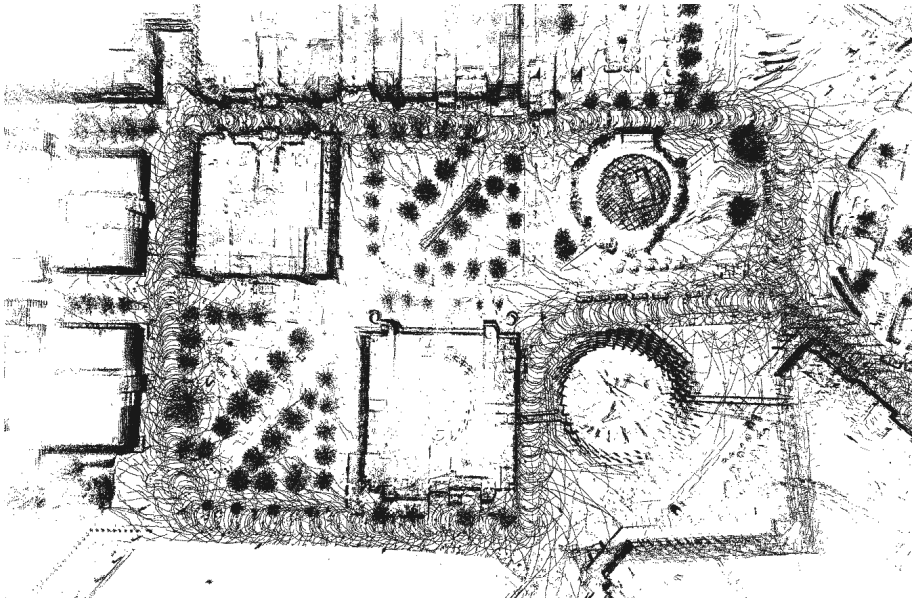


(a) Before loop closure

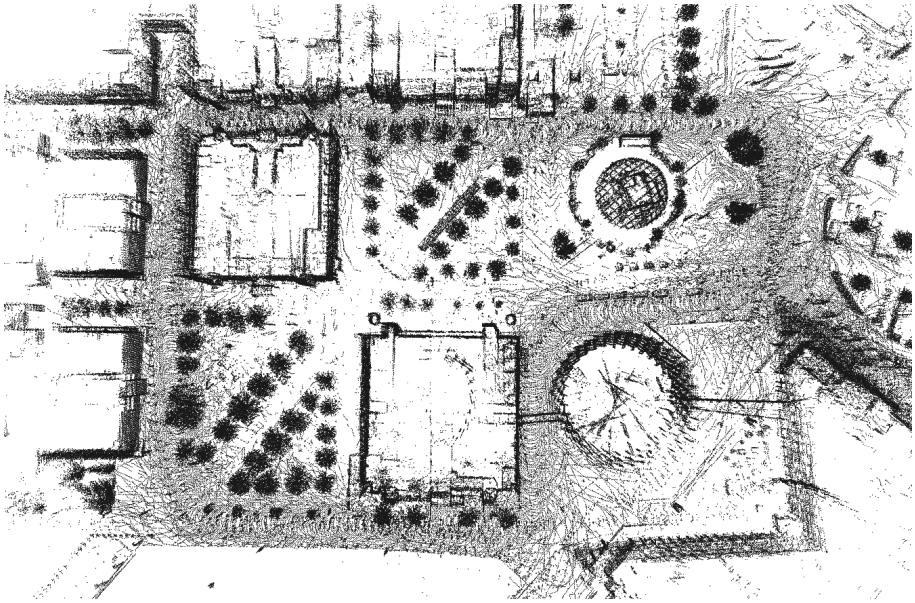


(b) After loop closure

Figure B.2: Dome of Visions at Lindholmen Campus

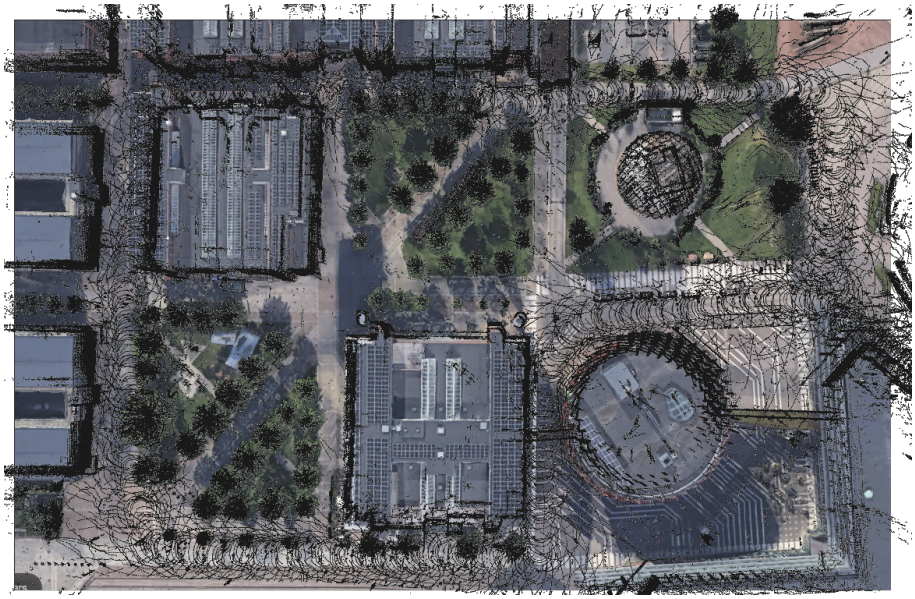


(a) Before loop closure

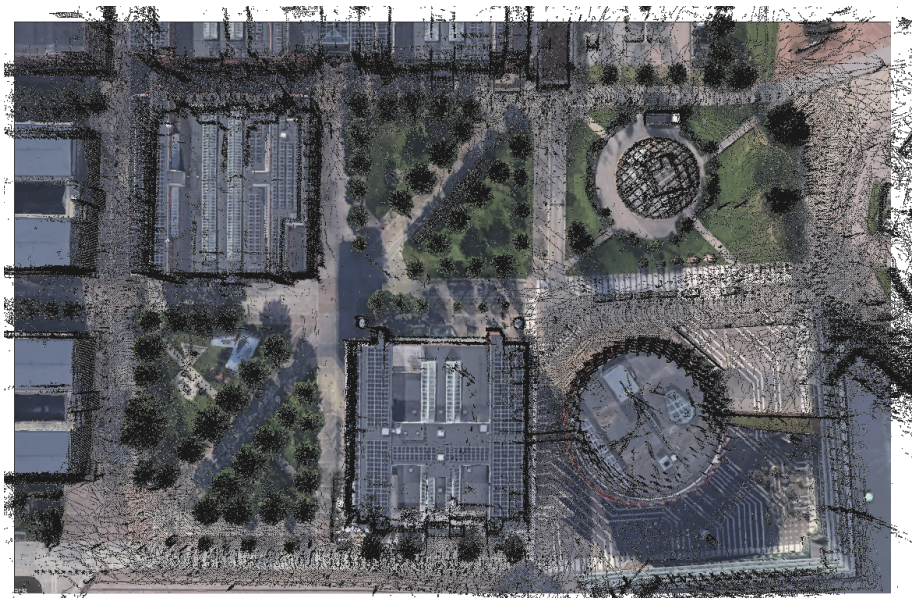


(b) After loop closure

Figure B.3: Point cloud map of Lindholmen



(a) Before loop closure

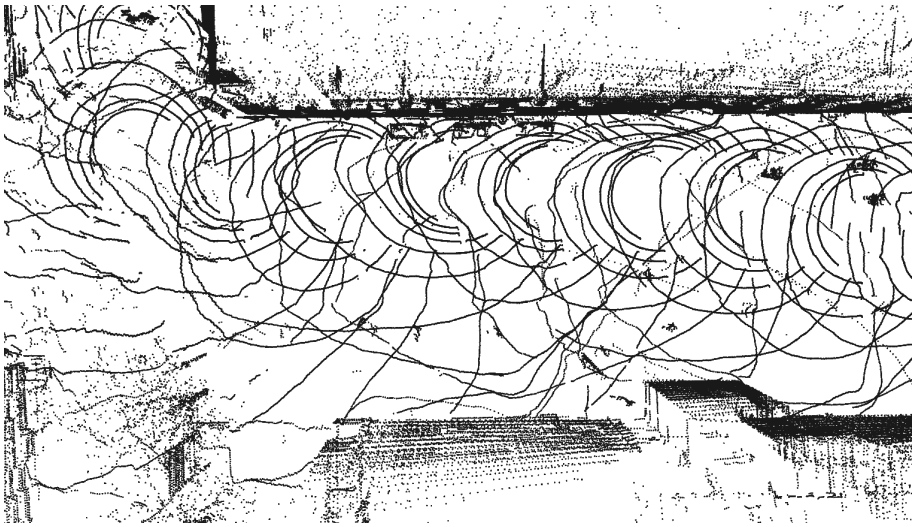


(b) After loop closure

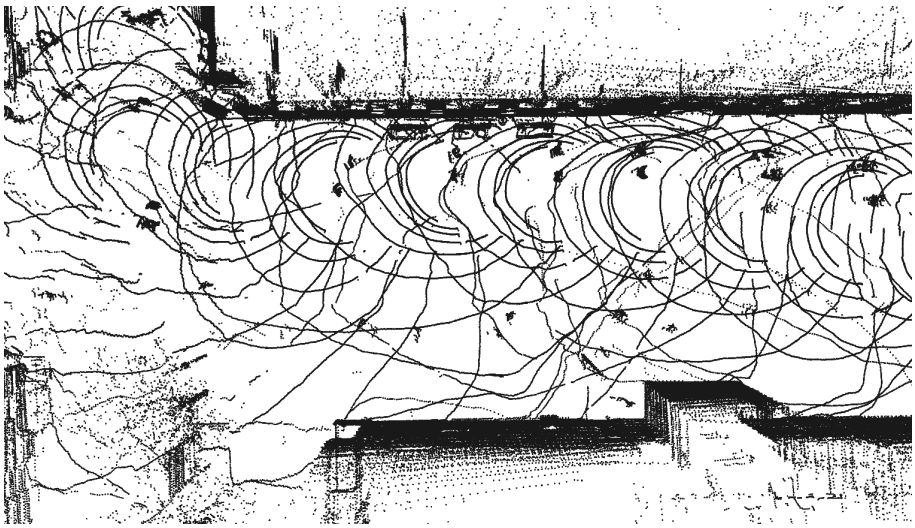
Figure B.4: Point cloud map from fig B.3 overlayed on a Google Earth satellite image

C

Qualitative Results Dynamaic Object Removal

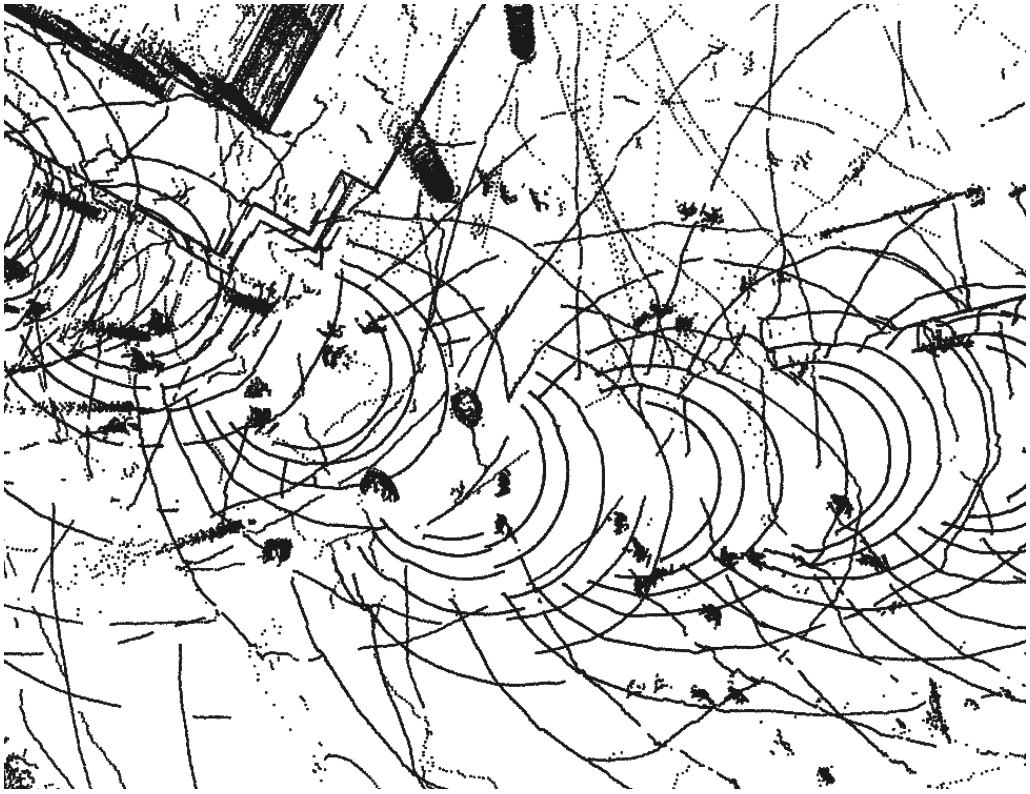


(a) Before

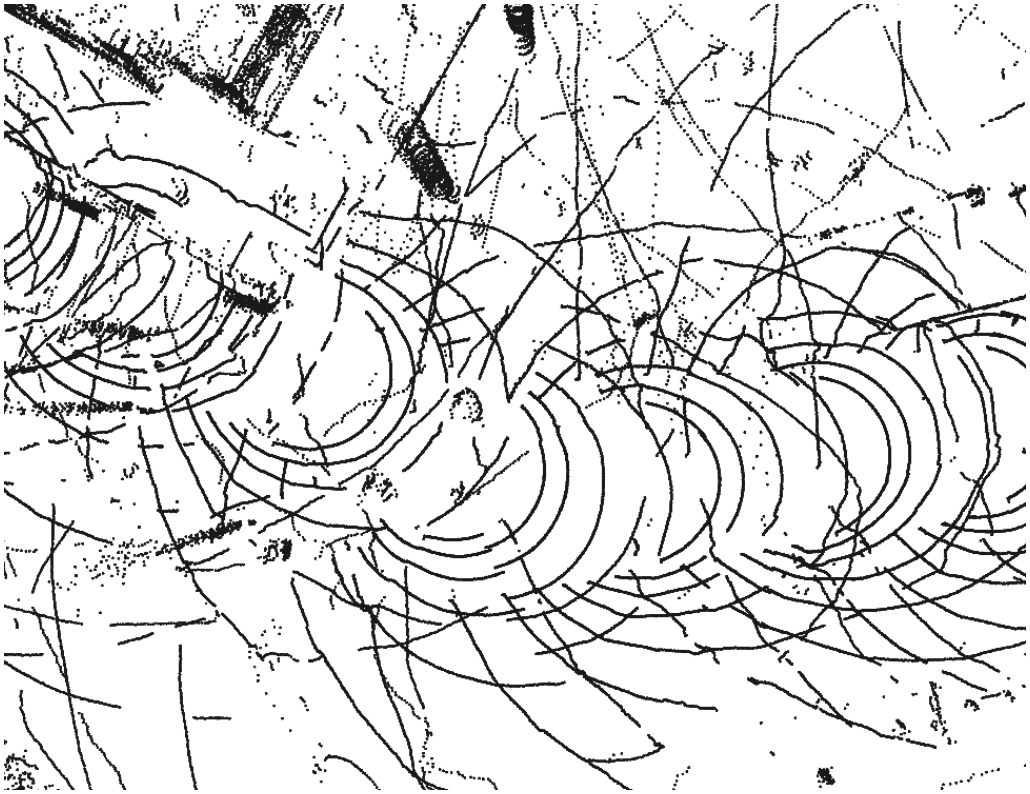


(b) After

Figure C.1: Narrow street with few pedestrians

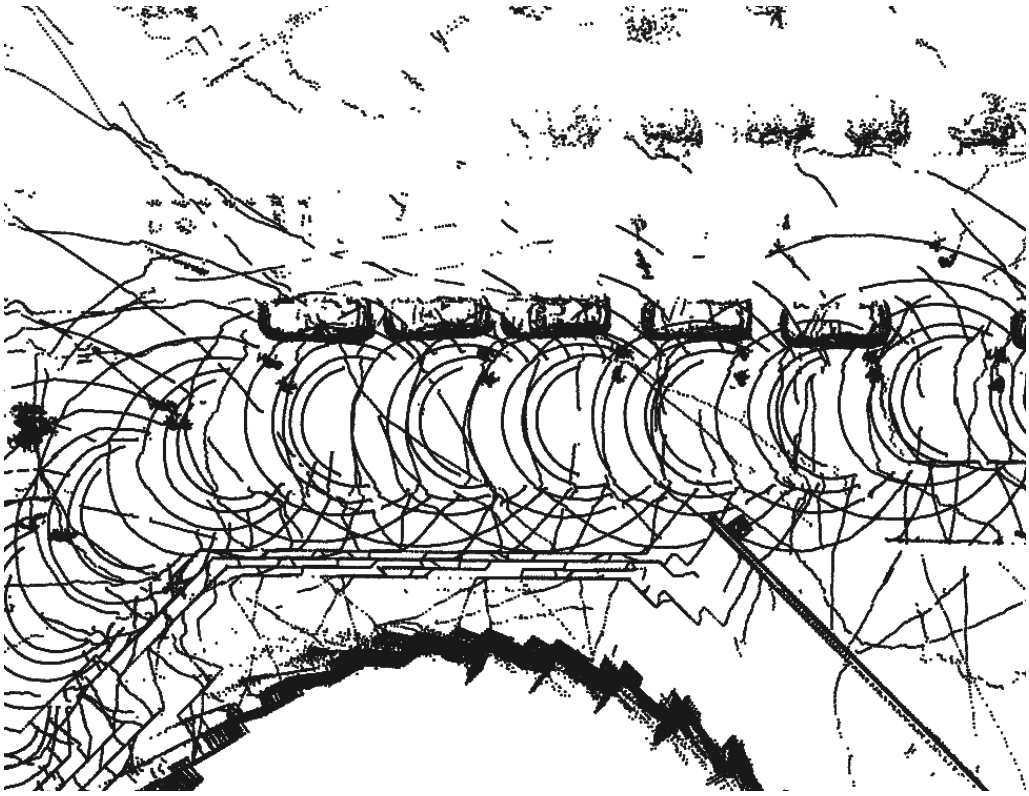


(a) Before

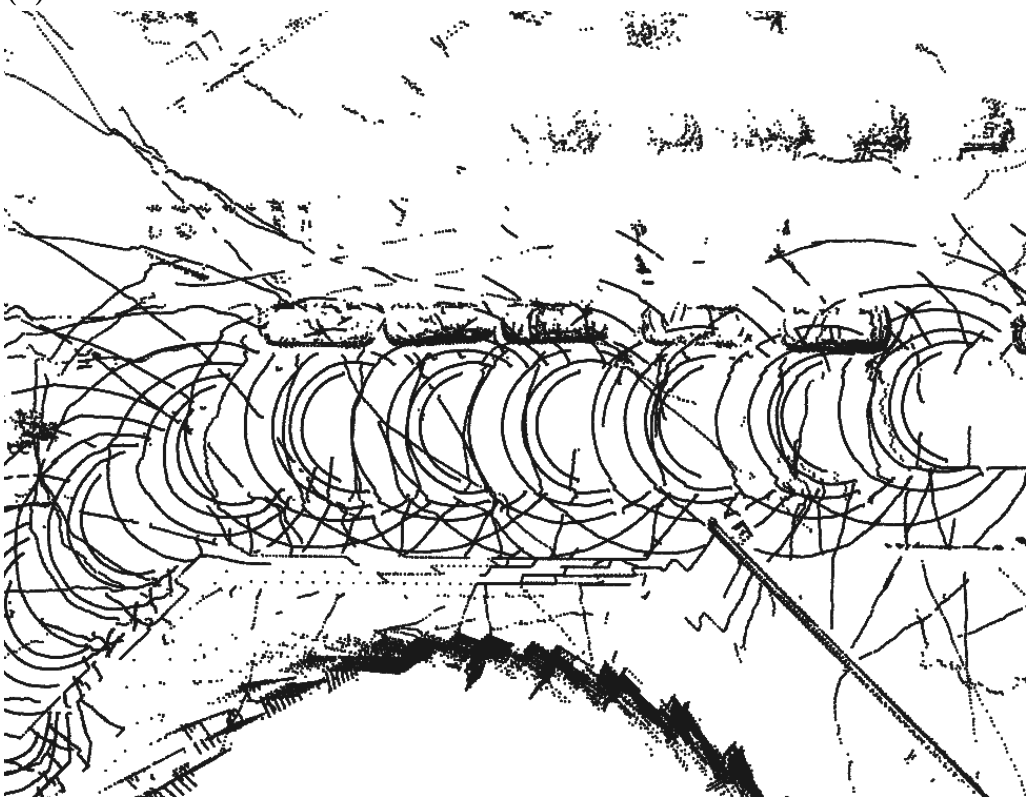


(b) After

Figure C.2: Intersection with multiple pedestrians

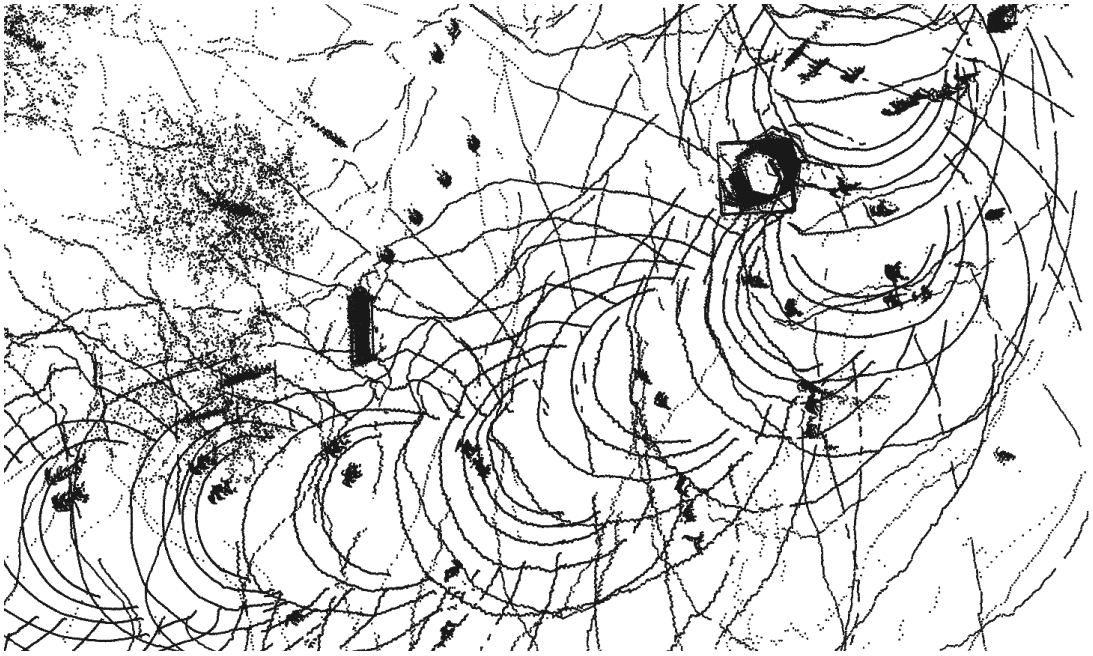


(a) Before

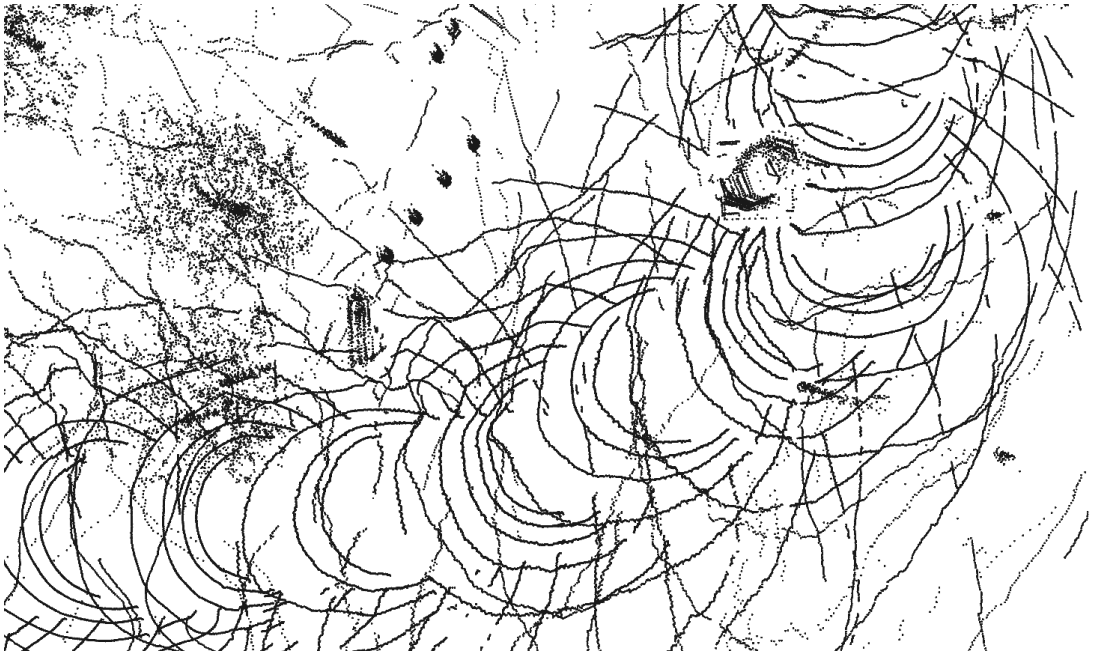


(b) After

Figure C.3: Path with parked cars

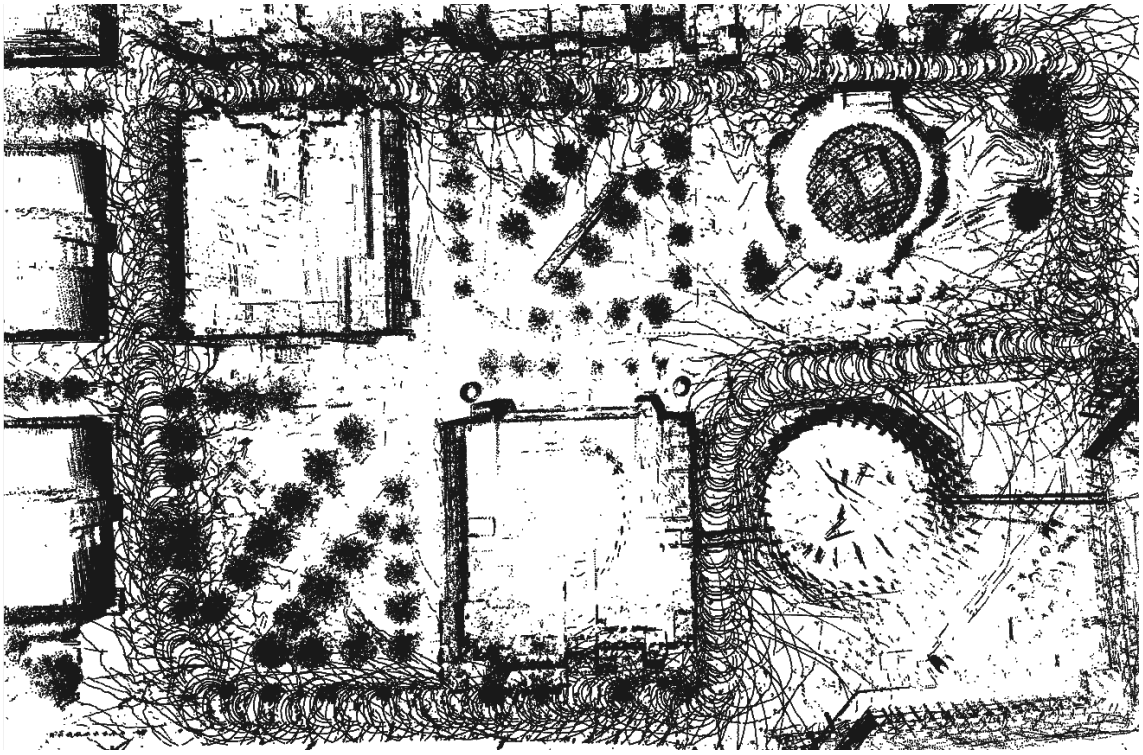


(a) Before

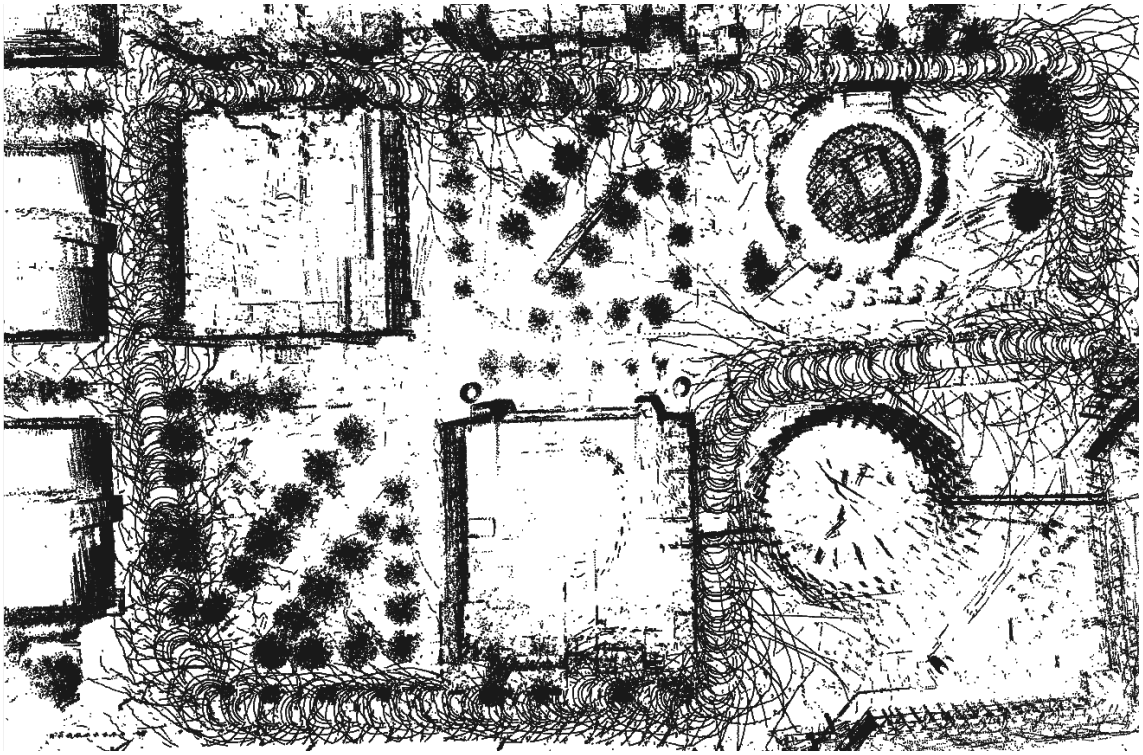


(b) After

Figure C.4: Highly Dynamic Area



(a) Map without cleaning



(b) After

Figure C.5: Cleaned Map

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden