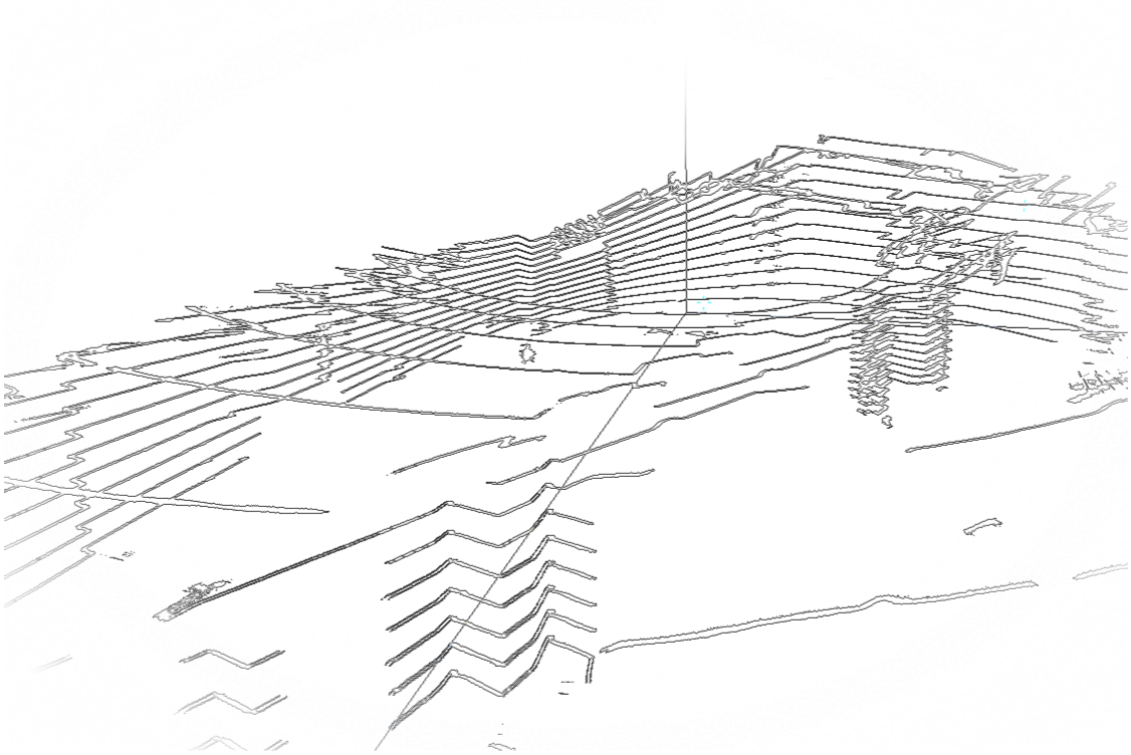
CHALMERS
UNIVERSITY OF TECHNOLOGY



Ego-Motion Tracking with LiDAR Point Clouds

An Evaluation of Utilizing LiDAR Point Clouds in Verification Tools for Ego-Motion

Master's thesis in Systems, Control and Mechatronics

ERIK HENNING LARSSON
JONATHAN JANSSON

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

Ego-Motion Tracking with LiDAR Point Clouds

An Evaluation of Utilizing LiDAR Point Clouds in Verification Tools for Ego-Motion

ERIK HENNING LARSSON
JONATHAN JANSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Ego-Motion Tracking with LiDAR Point Clouds
An Evaluation of Utilizing LiDAR Point Clouds in Verification Tools for Ego-Motion
ERIK HENNING LARSSON
JONATHAN JANSSON

Supervisors:
Karl Granström, Department of Electrical Engineering
Andreas Andersson, Aptiv Contract Services Sweden AB
Karin Brötjefors, Aptiv Contract Services Sweden AB

Examiner:
Karl Granström, Department of Electrical Engineering

Master's Thesis 2019
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visually edited LiDAR point cloud gathered in an indoor garage.

Typeset in LaTeX
Gothenburg, Sweden 2019

Ego-Motion Tracking with LiDAR Point Clouds
An Evaluation of Utilizing LiDAR Point Clouds in Verification Tools for Ego-Motion
ERIK HENNING LARSSON
JONATHAN JANSSON
Department of Electrical Engineering
Chalmers University of Technology

Abstract

In the automotive industry safety is a critical aspect. It is crucial that autonomous features are thoroughly tested and their functionality verified. In order to perform testing and verification objectively a reference for correct behaviour or performance is required. This is often referred to as the *ground truth*. Commonly ground truth of ego-motion is obtained with Inertial Measurement Units (IMUs) and Global Positioning Systems (GPSs). This thesis investigates the feasibility of using Light Detection And Ranging (LiDAR) point clouds for verification of ego-motion parameters speed and angular rates.

The investigation is performed by evaluating the popular Iterative Closest Point (ICP) algorithm and two of its extensions. Furthermore, different methods to increase robustness are evaluated. In addition, Bayesian filtering techniques are utilized to improve upon the resulting estimates.

The results show that estimation of speed and yaw rate is indeed possible. However, estimations of pitch and roll rates prove to be difficult. Further analysis shows that a LiDAR's procedure for sampling point clouds can cause issues in the process of estimating ego-motion, and that handling such issues can increase performance significantly. In conclusion, the thesis shows promising results for estimating yaw rate and speed of an ego-vehicle while presenting the importance of different procedures, as well as discussing possible causes of the difficulties in estimating pitch and roll rates.

Keywords: LiDAR, Velodyne, Point Clouds, IMU, Ego-Motion, Verification, Iterative Closest Point, Bayesian Filtering

Acknowledgements

We would like to express our gratitude towards our supervisors Andreas Andersson and Karin Brötjefors at Aptiv Contract Services Sweden AB for their engagement in the project on all levels. We would also like to thank our supervisor at Chalmers University of Technology Karl Granström for keeping us on track during all complications throughout the project. Lastly, we would like to thank Mats Björnerbäck, Jacob Gideflod, Jacob Andrén, Oscar Pantzare, Hugo Drakeskär and Adam Pettersson for their involvement in the hardware related parts of the project and more.

Erik Henning Larsson & Jonathan Jansson, Gothenburg, June 2019

Contents

List of Figures

List of Tables

Abbreviations and wording

LiDAR - Light Detection And Ranging
IMU - Inertial Measurement Unit
GPS - Global Positioning System
ICP - Iterative Closest Point
G-ICP - Generalized Iterative Closest Point
Mb-ICP - Metric-based Iterative Closest Point
V-ICP - Velocity Updating Iterative Closest Point
NDT - Normal Distributions Transform
NDT-OM - Normal Distributions Transform Occupancy Map
SLAM - Simultaneous Localization And Mapping
LOAM - LiDAR Odometry And Mapping
kd-tree - k-dimensional tree
SVD - Singular Value Decomposition
PCA - Principal Component Analysis
KF - Kalman Filter
RTSS - Rauch-Tung-Striebel-Smoother
RMSE - Root Mean Square Error

Transformation - Simplification of *proper rigid transformation* for convenience
Reading (In the context of LiDAR) - The distance calculated to a point from one laser firing to its corresponding detector receiving
Measurement (In the context of algorithms) - The observation of a state made by an algorithm.
Scan - The vertical array of lasers/detectors observing one measurement
Scan-cycle - One revolution of scans
Point cloud - The points sampled during a scan-cycle
Source (cloud) - The point cloud sampled during the previous scan-cycle
Target (cloud) - The point cloud sampled during the current scan-cycle
Match - The pair of points from the source and target cloud which is seen as corresponding points
Point correspondence - The ideal notion of two sequentially measured points representing the same point in space

1

Introduction

Currently, there is a large interest in automation in the automotive industry. Many vehicles today utilize automated features where collision warning, lane keeping assistance and emergency braking are examples of increasingly common features. For a vehicle to act autonomously it is critical that every aspect of the system has been thoroughly tested and verified, such that any action performed is never a possible safety hazard.

Testing and verification can be done either live, or offline with sampled data. The advantage of an offline process is that scenarios can be replayed and exchanged to quickly test the behaviour and performance of implemented features. Thus, enabling a measure for how performance change throughout development.

To evaluate the feature's output a *ground truth* is required, i.e, a reference for evaluating performance against. By having a reliable ground truth each function can have its performance evaluated objectively and automatically. A ground truth can be created manually or generated using a calibrated sensor with low influence of noise (or some combination thereof). However, manually creating a ground truth can be time consuming, and sensors with low influence of noise can be expensive.

In the area of autonomous drive, both the required amount of information and its accuracy of the ego-vehicle and surrounding objects grow quickly with the level of autonomy. Having individual high-quality sensors for verifying all measured parameters might not be economically feasible, given a large fleet of test vehicles. However, information gathered from one sensor can often be used for multiple purposes, thus minimizing the number of sensors required in each test vehicle.

1.1 Purpose

Using some combination of an Inertial Measurement Unit (IMU), Global Positioning System (GPS) and a wheel-speed sensor is likely the most accurate way to estimate the ego-motion parameters speed and angular velocities. Meanwhile, Light Detection And Ranging (LiDAR) sensors are known for their high accuracy in 3D mapping. Even though they do not inherently measure ego-motion it is likely that ego-motion

can be derived with reasonable accuracy using LiDAR generated point-clouds.

An accurate IMU is generally quite expensive. Even though accurate LiDARs are also expensive they are trending towards lower prices [1, 2]. Evaluating if estimates from a LiDAR are comparable with estimates of an IMU, it could result in that an IMU no longer is a necessity for verification of ego-motion. Furthermore, if cost is not an important parameter in a validation setup, a LiDAR's estimate could either be fused with an IMU's estimate, or used for redundancy. Thus, this thesis aims to evaluate the feasibility and accuracy of using LiDAR generated point clouds for estimating ego-motion.

1.2 Scope

The goal and objective of this thesis is to evaluate the accuracy of ego-motion estimations using LiDAR generated point clouds. To achieve this, the scope of the project is defined as:

- Investigate the feasibility of achieving high-quality estimates of the ego-motion parameters speed and angular rates.
- Perform estimations in an offline setting.
- Perform estimations on data generated in a static environment with clear landmarks/way-points.
- Evaluate the quality of estimations in relation to IMU and wheel speed sensors.

The investigation includes finding appropriate models and algorithms for tracking, filtering as well as correction for point-cloud distortions.

1.3 Related Work

Localization and 3-D mapping is useful in many areas, not least for autonomous drive. Thus, there exists a lot of research regarding odometry estimation using LiDAR point clouds. Solutions vary, but a common approach is the relatively simple, yet effective, Iterative Closest Point (ICP) algorithm [3].

ICP matches the closest points of two clouds and attempts to find the relative pose, or *transformation*, that aligns them by minimizing the distance between corresponding points. Some crude assumptions are made for optimal performance, such that there exists a perfect point correspondence between the clouds, which is generally not true due to noise, occlusion etc.

Naturally there exists a flurry of extensions to ICP trying to solve this (> 2300 hits on IEEE when searching for "Iterative Closest Point", where more than 200 were published in 2018). In the review of ICP algorithms in [4] it is described how there exists many options, as well as modules which can be added and combined to fit different applications.

Some extensions of ICP attempts to solve the problem of not having perfect point to point correspondence by minimizing the distance between a point and a geometric feature where its corresponding point is contained, such as point to plane[5]. It is shown in [6] that this type of minimization increases the accuracy in the estimates.

Further the popular algorithm Generalized Iterative Closest Point (G-ICP) [7] extends this approach by minimizing the distance between two planes, where each plane contains a corresponding point. It adds a probabilistic framework to the minimization in an attempt to model the uncertainties generated by measurement noise of the sensor. There also exists fully probabilistic frameworks as proposed in the probabilistic Iterative Correspondence[8] and is said to increase robustness in unstructured environments.

A different approach is performed by the Iterative Dual Correspondence [9] algorithm, which address that the correspondence search does not consider the rotational components, which could lead to large residual errors. It improves the matching process by using two sets of correspondences; rotational and translational components. The approach is similar in the Metric-based Iterative Closest Point (Mb-ICP) [10] algorithm where the rotational components is included in their proposed metric. This paper has been generalized into three dimensions by [11]. Along with the new metric a point to facet approach is used, where the geometry of the plane containing the point is limited to only span a facet.

Another interesting approach to matching point-clouds is by utilizing the measured intensity of a reflected laser scan. This is used in [12] where a mixed correspondence matching approach is proposed.

Apart from ICP and its extensions, a common approach for point-cloud matching is the Normal Distributions Transform (NDT) [13]. The scanned area is divided into cells, where each cell is assigned a normal distribution of the probability of measuring a point. Two clouds can then be matched without establishing point correspondences.

As for ICP, NDT has many extensions. An example is Normal Distributions Transform Occupancy Map (NDT-OM) [14], which combines the cell based approach of NDT with an occupancy grid method. NDT-OM and G-ICP is compared for odometry tracking in the masters thesis by Lindén [15], where similar performance is obtained. Worth noting is that NDT-OM requires a local point cloud map to match against.

The family of Simultaneous Localization And Mapping (SLAM) algorithms has several high performance variants where some is amongst the top versions of KITTI's

odometry benchmark charts [16]. SLAM algorithms tries to find the pose of the sensor while simultaneously updating a local map of the environment. The method LiDAR Odometry And Mapping (LOAM) [17] uses feature extraction of edges and planes, similar to to some ICP versions, which is run in parallel with an ego-motion estimation algorithm to correct for point cloud distortions. A special case called Velodyne SLAM is proposed in [18], where the method is specifically constructed for a Velodyne HDL-64E in an attempt to capture the sensors specific characteristics. Both of these SLAM approaches uses some ICP variant for pose estimation. Versions using a local map of the environment is discarded from the evaluation, due to them only gaining accuracy if located in a previously visited area.

1.4 Thesis Outline

First of all, the context of the thesis is presented in the Preliminaries chapter in terms of hardware setup. Next, the theory behind different point cloud matching methods used for estimating ego-motion is introduced, followed by theory behind utilized filtering techniques. Then, the Implementation chapter describes how previously described point cloud matching methods and filtering techniques are composed into a set of proposed solutions. Following section describes the environment and the tests which were performed as well as how evaluation is made. Finally, results of the different proposed solutions are presented and discussed, followed by a summarizing Conclusion chapter with findings from performed tests.

2

Hardware

This chapter defines the context of the thesis in terms of hardware. It presents the testing vehicle and different sensors used throughout gathering data, and describes the sensors individual characteristics.

2.1 Data Gathering Platform

The platform to be used when gathering data for evaluation is an electric auto rickshaw called *Zbee* and is manufactured by *Clean Motion* [19].

Figure 2.1: Image of the auto rickshaw used for testing and verification.

On the auto rickshaw's roof a *VLP-16* LiDAR from *Velodyne LiDAR* [20] is mounted. Further, an *HG1120BA50* IMU from *Honeywell Aerospace* [21] is mounted close to the auto rickshaw's center of rotation, and will be used as reference when estimating angular velocities.

2.2 LiDAR

The acronym LiDAR comes from *Light Detection And Ranging*. A LiDAR is a sensor measuring distances through emitting pulses of light through a laser. The light bounce on surrounding objects and returns to the sensors detector, which also registers the returned signals intensity. Distances are then calculated by measuring time between emitting and receiving. The VLP 16 used is a so called *Rotational LiDAR*, which consists of a vertical array of lasers and performs 360 degree scans by rotating about its z-axis. A point cloud is hereby defined as one lap of sampled points.

2.3 IMU

An Inertial Measurement Unit (IMU) is a combination of sensors measuring accelerations (accelerometer), angular rates (gyroscope) and sometimes the magnetic field (magnetometer) acting on the device. Commonly three sets of each sensor type is included and oriented to make observations of the device axes x, y and z. This setup can be used to calculate the absolute orientation of the sensor by fusing data from the sensors, as well as calculating velocity through integration of accelerometer values.

2.4 Wheel Speed - Hall Effect Sensors

Calculating speed with the accelerometer in an IMU is performed by integrating the sensors acceleration readings. Thus, the noise from the accelerometers readings will be integrated and adds up to a bias over time. By measuring the magnetic field of the brush-less DC motors in the auto-rickshaw using Hall Effect sensors the wheel-speed can be calculated without integration errors [22].

3

Point Cloud Matching

The general approach for estimating ego-motion with LiDAR sensors in this thesis is to find a transformation which aligns two consecutive sampled point clouds. The following chapter presents theory behind the point cloud matching algorithm ICP and its modified versions; Mb-ICP and G-ICP, as well as a range of additions attempting to improve upon the algorithm. All of which provides the foundation for the final analysis of using LiDARs for ego-motion estimation. First concepts of rigid transforms are introduced, as they make out the format of which the different point cloud matching algorithms yield an estimation.

3.1 Rigid Transformation

A local right handed Cartesian coordinate system is defined with x pointing forwards, y horizontally orthogonal to x and z pointing up with the sensor in origin. Rotations around the coordinate systems axes x, y and z are described by the angles (ϕ, θ, ψ) called *roll, pitch* and *yaw* and can be observed in Figure 3.1 below.

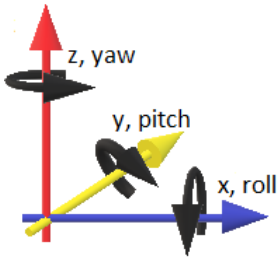


Figure 3.1: Cartesian coordinate system x, y, z with rotations (ϕ, θ, ψ) around its corresponding axis called *roll, pitch, yaw*, respectively.

The relative pose difference (position and orientation) between two points $a \in \mathcal{A}$ and $b \in \mathcal{B}$ in the sensors coordinate system is found through alignment of the points and can be described by a *proper rigid transformation*, further called a *transformation* for convenience. Transformations preserve the structure of the data set that is being

transformed and is defined as

$$T = \begin{bmatrix} R_{3x3} & t_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}, \tag{3.1}$$

where $t_{3x1} = [t_x, t_y, t_z]^\top$ is a translation vector and R_{3x3} is a rotation matrix. Rotations in the project is described by Tait-Bryan angles with the zyx convention defined as

$$R_{zyx} = R_z R_y R_x, \tag{3.2}$$

where

$$\begin{gathered} R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix}, \quad R_y = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}, \\ R_z = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \end{gathered} \tag{3.3}$$

describes elemental rotations around the axis indicated by its subscript.

Using homogeneous coordinates, i.e, scalable representations of points, a point is defined as $a' = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^\top$ indicated by the prime notation. One can describe the alignment of point a' to point b' with transformation T as

$$b' = Ta', \tag{3.4}$$

where consecutive transformations can easily be applied by sequential multiplications

$$b' = T_1 T_2 \dots T_n a'. \tag{3.5}$$

Some operations require the normalized point representation, described as

$$a = \frac{a'_{1:3}}{a'_4}, \tag{3.6}$$

where subscript indicates position in the vector.

3.2 Iterative Closest Point

The Iterative Closest Point (ICP) algorithm was made famous by Besl and McKays paper in [3] and is used for alignment of point clouds. ICP can be described in

two major steps; find the closest points between two clouds, then minimize the distance between each point pair. Finding the closest points between the clouds is a *nearest neighbor* problem, which can be accelerated by using a *k-dimensional tree* (kd-tree)[23, 24].

The alignment process is done iteratively, hence the name, with the aim of finding the transformation which is the optimum of the minimization problem. For each iteration the transformation is applied to the source cloud such that convergence is reached when the source and target cloud are aligned.

A visualization of ICP's process is displayed in Figure 3.2, where the top right, and bottom left image demonstrates the iterative process, and the bottom right demonstrates the result.

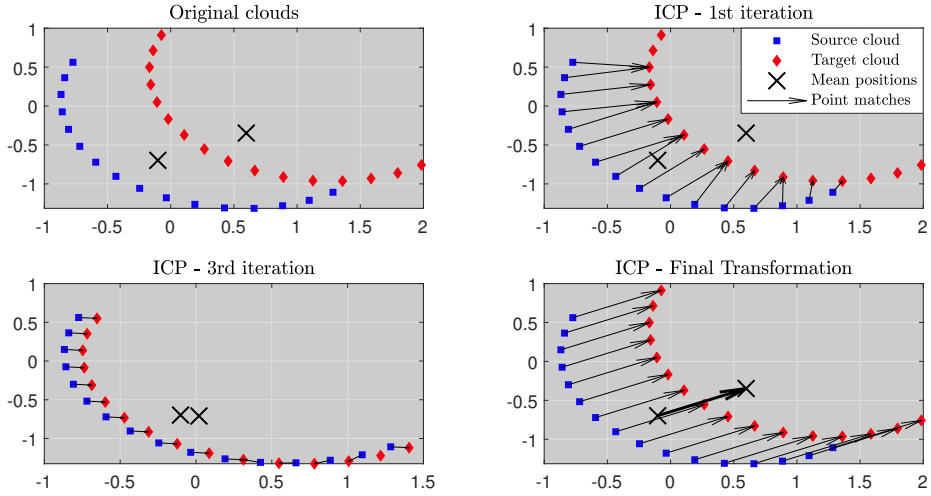


Figure 3.2: A visual presentation of how the iterative process of ICP. First, the points in the two clouds are matched based on distance. Then the transform which minimize the distance between the correspondences is applied. These two steps are iterated until convergence, when the final transform is found by summarizing the iteratively applied transforms.

Besl and McKay suggests to solve the minimization problem with the quaternion approach of Horn [25] for lower dimensions to avoid reflections, though for dimensions of $n > 3$ Singular Value Decomposition (SVD) is suggested. The authors of [26] states that there is no noticeable difference in accuracy for three dimensions between the two, while SVD being faster in general.

The minimization method used is the SVD based Least-squares fitting method presented in [27], which can be read for the complete derivation and proof. Given two sets of points $\mathcal{A} = \{a_i'\}$ and $\mathcal{B} = \{b_i'\}$, $i = 1, 2, \ldots, N_p$ where corresponding points share index. The aim is to find the transformation T (as defined in (3.1)) which

minimize the distance

$$d = \sum_{i=1}^{N_p} ||b'_i - Ta'_i|| = \sum_{i=1}^{N_p} ||b_i - (Ra_i + t)||, \tag{3.7}$$

such that $\mathcal{A}$ is aligned onto $\mathcal{B}$. Initially the centroid of each set is computed as

$$\bar{\mathcal{A}} = \frac{1}{N_p} \sum_{i=1}^{N_p} a_i, \ \bar{\mathcal{B}} = \frac{1}{N_p} \sum_{i=1}^{N_p} b_i. \tag{3.8}$$

The non-normalized covariance matrix C is then computed as

$$C = \sum_{i=1}^{N_p} (a_i - \bar{\mathcal{A}})(b_i - \bar{\mathcal{B}})^\top. \tag{3.9}$$

Then the SVD of C is found

$$C = U\Lambda V^\top, \tag{3.10}$$

where U is the eigenvectors of $CC^\top$, V the eigenvectors of $C^\top C$ and Λ contains the singular values (square root of eigenvalues) of $CC^\top$. Rotation R can be calculated as

$$R = UV^\top, \tag{3.11}$$

and translation t as

$$t = \bar{\mathcal{B}} - R\bar{\mathcal{A}}, \tag{3.12}$$

if $det(R) = 1$. Meanwhile, if $det(R) = -1$ the method fails as the solution of the rotation represents a reflection, although it is stated as unlikely [27].

In order for the residual distances between matched points to be completely removed ICP require perfect point to point correspondence in the two clouds to be aligned. For real world applications this is unlikely due to measurement noise, occlusion or distortions from motion during sampling [28].

Further, ICP is prone to converge to a local minimum given large rotations between the point clouds [29, 28]. To aid the alignment process an initial transformation can be passed as a guess such that the algorithm is less likely to converge to a local minimum [4].

A visualization of the motion distortion is displayed in Figure 3.3, where the difference in positions, which is unknown for the sensor throughout the sampling process, results in a point-cloud displaying a distorted realization of the environment. This is due to rotational LiDARs sample environments in a sweeping manner over time, rather than instantly. Addressing this issue is important as suggested in [15, 18, 30, 31].

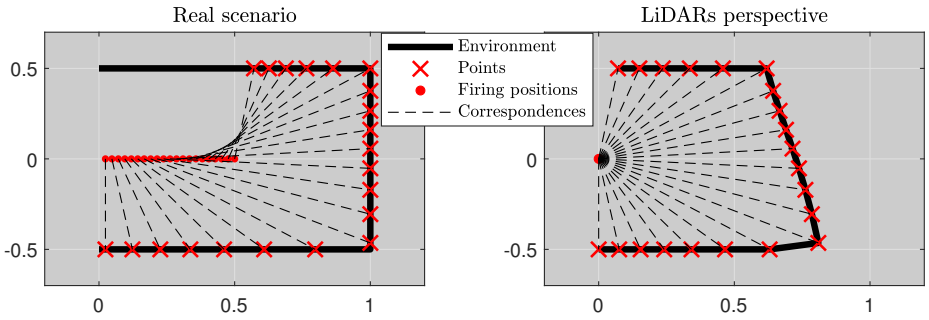


Figure 3.3: Demonstration of how gathering a point cloud over time during movement cause distortions in the LiDARs interpretation of the surrounding environment.

The authors of [4, 6] describe ICP as having distinct computation steps, where different methods or combinations thereof can be implemented in each step to tackle some of the issues stated in section 1.3. Due to its modular behaviour it is suitable to describe the computation steps as *modules*. In this thesis the alignment process is described by the following modules:

1. **Pre-Processing:** Selecting a subset of points from each cloud, either to enhance performance or lower computational cost.

2. **Cloud Rectification:** Minimize distortions in the source cloud.

3. **Point Correspondence search:** Find the points in the source cloud which corresponds to points in the target cloud.

4. **Matching Rejection:** Reject pairs of corresponding points classified as outliers.

5. **Minimization:** Minimization of a distance metric in order to align the source and target point cloud.

where point 3 and 5 are core modules of the algorithm. If the core modules are modified then the algorithm is described as an *extension* to ICP. Some extensions such as Metric-based ICP defines a new distance metric to be minimized, and as a result modifies both the point correspondence search as well as the minimization step. Others such as the Generalized ICP changes only the minimization step, leaving the point correspondence search unchanged. The theory for these modules and extensions is described in the following sections.

3.3 Extensions to ICP

There exists many versions of ICP, where some only differ slightly from the base algorithm, while others redesign the entire core structure. The following ICP exten-

sions were chosen such that a module based approach can be used and extensions which require specific structures which does not comply with the other modules are discarded. Thus, two different implementations were chosen which do not significantly interfere with the other modules, namely the Metric-based Iterative Closest Point and the Generalized Iterative Closest Point algorithms which are described in the following subsections.

3.3.1 Metric-based ICP

Originally the Metric-based Iterative Closest Point algorithm was proposed for two dimensional matching problems [10]. In order to better handle rotational components which distorts the correspondence search, as seen in Figure 3.4, it introduces a new distance metric called *metric-based distance*.

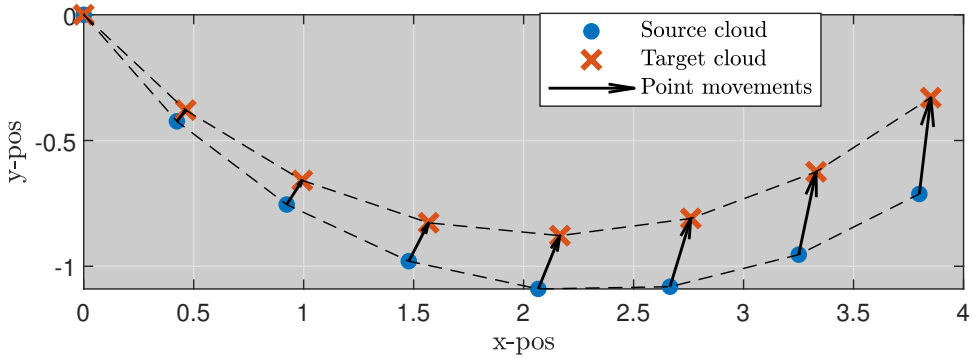


Figure 3.4: An arc of points before and after a rotation. The figure displays the difference in distance between the individual points when a rotation is applied.

This new distance attempts to consider both translational and rotational components. Further, a generalization to 3-dimensional point clouds is proposed by [11].

ICP assumes the existence of true point correspondences in the two clouds, which is in general false when sampling the clouds with a LiDAR. Mb-ICP solves this by creating a local structure in which the true corresponding point is likely to lie within. More specifically, Mb-ICP attempts to minimize the distances between the each point in the source cloud and the corresponding facet defined by the three closest points in the target cloud.

The new metric is used to calculate point-to-point distances and is defined as

$$d_{ap}(p_1, p_2) = \sqrt{||\delta||_2^2 - \frac{||p_1 \times \delta||_2^2}{k}}, \ \delta = p_1 - p_2, \ k = ||p_1||_2^2 + L^2, \tag{3.13}$$

where L is a weighting factor between rotation and translation.

The metric-based distance is then used instead of euclidean distance to perform a point correspondence search. For each point in the source cloud, Mb-ICP finds the three closest points, $\{v_1, v_2, v_3\}$, in the target cloud. These three points defines a facet v where the true corresponding point $p(\lambda^*)$ is likely to lie.

Next, Mb-ICP computes the true closest point within the facet. This is performed conditionally depending on whether the point is found to lie on the edge of or within v. Further, as presented in Figure 3.5, depending on along which edge segment the closest point lies its position is derived slightly different.

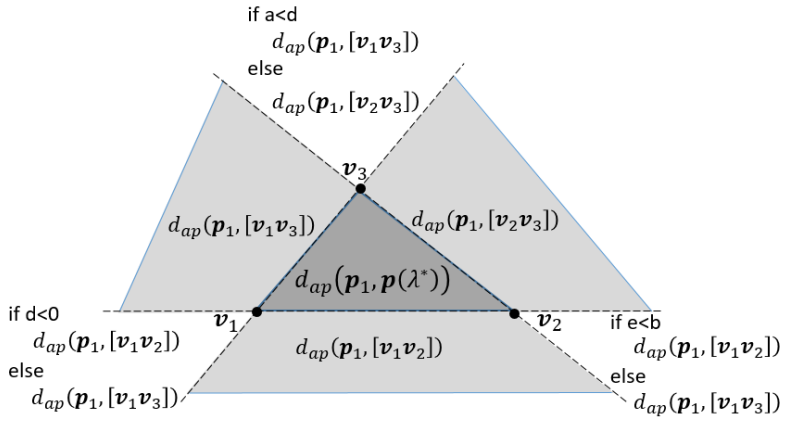


Figure 3.5: Depiction of how the distance from p_1 to the closest point $p(\lambda^*)$ is computed depending on its position. $d_{ap}(p_1, [v_1, v_2])$ defines the distance from the point p_1 to the segment between point v_1 and v_2. $d_{ap}(p_1, p(\lambda^*))$ defines the distance form the point p_1 to the plane spanned by v.

Consequently, the point is derived either by finding the closest point to one of the edge segments or plane spanned by v. Mb-ICP solves this search by first deriving the closest point in the plane spanned by v, and later evaluating where that point lies relative to the facet.

The closest point in the plane from point p_1 in the source cloud is derived by finding the minimum squared point-to-plane distance as

$$d^2(\lambda) = \lambda^\top A\lambda - 2B^\top\lambda + C, \tag{3.14}$$

where $\lambda = [\lambda_1, \lambda_2]^\top$, $A = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$, $B = \begin{bmatrix} d \\ e \end{bmatrix}$, $C = f$, in which

$$\begin{aligned}
a &= ||v_3 - v_1||_2^2 - \frac{||p_1 \times (v_3 - v_1)||_2^2}{k}, \\
b &= ||v_2 - v_1||_2^2 - \frac{||p_1 \times (v_2 - v_1)||_2^2}{k}, \\
c &= (v_3 - v_1)^\top (v_2 - v_1) - \frac{(p_1 \times (v_3 - v_1))^\top (p_1 \times (v_2 - v_1))}{k}, \\
d &= -(v_1 - p_1)^\top (v_3 - v_1) + \frac{(p_1 \times (v_1 - p_1))^\top (p_1 \times (v_3 - v_1))}{k}, \\
e &= -(v_1 - p_1)^\top (v_2 - v_1) + \frac{(p_1 \times (v_1 - p_1))^\top (p_1 \times (v_2 - v_1))}{k}, \\
f &= ||(v_1 - p_1)||_2^2 - \frac{||p_1 \times (v_1 - p_1)||_2^2}{k}.
\end{aligned} \tag{3.15}$$

The minimum of (3.14) is $\lambda^* = A^{-1}B$ which results in the corresponding point being computed as $p(\lambda^*) = v_1 + \lambda_1(v_3 - v_1) + \lambda_2(v_2 - v_1)$.

If $p(\lambda^*)$ is not contained within the facet, the closest point to one of the edge segments is computed. Finding a closest point on such a segment defined as $s = [s_1, s_2]$, $s_1, s_2 \subseteq \{v_1, v_2, v_3\}$, to point p_1, is performed by minimizing the squared distance between the point and segment. The squared distance from point to segment is defined as

$$d^2(\mu) = a_s\mu^2 - 2b_s\mu + c_s, \tag{3.16}$$

where

$$a_s = ||s_2 - s_1||_2^2 - \frac{||p_1 \times (s_2 - s_1)||_2^2}{k}, \tag{3.17}$$

$$b_s = -(s_1 - p_1)^\top (s_2 - s_1) + \frac{(p_1 \times (s_1 - p_1))^\top (p_1 \times (s_2 - s_1))}{k}, \tag{3.18}$$

$$c_s = ||(s_1 - p_1)||_2^2 - \frac{||p_1 \times (s_1 - p_1)||_2^2}{k}. \tag{3.19}$$

The minimum of (3.16) is $\mu^* = \frac{b_s}{a_s}$, and the point on the segment is calculated as

$$p^* = (1 - \mu)s_1 + \mu s_2. \tag{3.20}$$

In [11] an efficient way to determine if the closest point lies within the facet or along which edge segment is presented. This is performed by utilizing information gained from deriving the closest point to the plane spanned by v as

$$
\begin{aligned}
[v_1, v_2] \quad & if\ \lambda_1^* < 0\ and\ \begin{cases} \lambda_2^* > 0\ and\ \lambda_1^* + \lambda_2^* < 1, \\ or\ \lambda_2^* < 0\ and\ d < 0, \\ or\ \lambda_1^* + \lambda_2^* > 1\ and\ e < b \end{cases} \\
[v_1, v_3] \quad & if\ \lambda_2^* < 0\ and\ \begin{cases} \lambda_1^* > 0\ and\ \lambda_1^* + \lambda_2^* < 1, \\ or\ \lambda_1^* < 0\ and\ d \geq 0, \\ or\ \lambda_1^* + \lambda_2^* > 1\ and\ d < a \end{cases} \\
[v_2, v_3] \quad & if\ \lambda_1^* + \lambda_2^* > 1\ and\ \begin{cases} \lambda_1^* > 0\ and\ \lambda_2^* > 0, \\ or\ \lambda_1^* < 0\ and\ e \geq b, \\ or\ \lambda_2^* < 0\ and\ d \geq a \end{cases} \\
[v_1, v_2, v_3] \quad & otherwise
\end{aligned}
\quad , \tag{3.21}
$$

where $[v_1, v_2]$ defines the edge segment between v_1 and v_2 (etc.), and $[v_1, v_2, v_3]$ defines the plane spanned by v.

In addition to modifying the point correspondence search, the minimization step of ICP is altered to fit the new distance function. This is done by using a least-squares estimator to minimize the squared distance of the corresponding points, where a_i, b_i denotes points in the source cloud and target cloud, respectively. That is, by minimizing

$$E_{dist}(q) = \sum_{i=1}^{N_p} d_{ap}^2(a_i, b_i), \tag{3.22}$$

where N_p is the total amount of points, $b_i = R(n, \theta)a_i + t$, in which θ is the rotation angle around unit vector n and $q = [t_x, t_y, t_z, \theta n_x, \theta n_y, \theta n_z]^\top$. However, this expression does not have a closed form solution. This is solved by linearizing about $\theta = 0$ resulting in

$$E_{dist}(q) = \sum_{i=1}^{N_p} \delta_i^\top(q) M(a_i) \delta_i(q), \tag{3.23}$$

where $\delta_i(q) = \delta_i + [a_i]_x \theta n - t$ and

$$M(a_i) = I - \frac{[a_i]_x^\top [a_i]_x}{k}, \tag{3.24}$$

in which $\delta_i = b_i - a_i$ and $[a_i]_x$ is the skew-symmetric matrix

$$[a_i]_x = \begin{bmatrix} 0 & -a_{i,z} & a_{i,y} \\ a_{i,z} & 0 & -a_{i,x} \\ -a_{i,y} & a_{i,x} & 0 \end{bmatrix}. \tag{3.25}$$

This can be expanded as

$$E_{dist}(q) = q^\top \alpha q - 2\beta^\top q + \gamma, \tag{3.26}$$

$$\alpha = \sum_{i=1}^{N_p} \begin{bmatrix} M(a_i) & -M(a_i)[a_i]_x \\ -[a_i]_x^\top M(a_i) & [a_i]_x^\top M(a_i)[a_i]_x \end{bmatrix}, \tag{3.27}$$

$$\beta = \sum_{i=1}^{N_p} \begin{bmatrix} M(a_i) \\ M(a_i)[a_i]_x \end{bmatrix} \delta_i, \;\; \gamma = \sum_{i=1}^{N_p} \delta_i^\top M(a_i)\delta_i, \tag{3.28}$$

from which the vector q which minimize the least squares problem can be found as $q^* = \alpha^{-1}\beta$, where q contains rotational and translational components.

3.3.2 Generalized ICP

Generalized Iterative Closest Point, as described in the original article [7], modifies the minimization step of ICP by attaching a probabilistic model of local geometric structures in the point clouds. This aims to resolve not having perfect point correspondences by assuming corresponding points still exist in the same local plane.

Given two sets of points $\mathcal{A} = \{a_i\}$ and $\mathcal{B} = \{b_i\}$, the probabilistic model assumes existence of an underlying true set of corresponding points $\hat{\mathcal{A}} = \{\hat{a}_i\}$ and $\hat{\mathcal{B}} = \{\hat{b}_i\}$. In other words, it assumes that $\mathcal{A}$ and $\mathcal{B}$ are sampled according to $a_i \sim \mathcal{N}(\hat{a}_i, C_i^{\mathcal{A}})$ and $b_i \sim \mathcal{N}(\hat{b}_i, C_i^{\mathcal{B}})$, where $C_i^{\mathcal{A}}$ and $C_i^{\mathcal{B}}$ are covariance matrices associated with the sampled points.

Extraction of covariance matrices can be done through Principal Component Analysis (PCA) of the matched points. The local geometry is captured for each point in the two sets using a *bounded radius neighbor search*, where neighbors are defined as the points contained within the neighborhood bounded by a given radius. As discussed in [32], this radius has to be chosen such that it captures the structure of the local area, but small enough not to be skewed from including surrounding features.

Once a neighborhood of point ρ is established the sample covariance of the resulting neighbors is calculated (similar to (3.9)) as

$$C = \frac{1}{M-1} \sum_{j=1}^{M} (p_j - \mathbb{E}[NN])(p_j - \mathbb{E}[NN])^\top, \tag{3.29}$$

where $NN = \{p_1, p_2, \ldots, p_M\}$ is the set of neighboring points of ρ, the point $p_j \in NN$, M is the number of neighbors within the given radius and $\mathbb{E}[NN]$ the sample mean of NN.

The eigenvalue equation

$$C \cdot \mathbf{v}_j = \lambda_j \cdot \mathbf{v}_j, \quad j \in \{1, 2, 3\} \tag{3.30}$$

is solved through eigendecomposition where $\mathbf{v}_j$ is the eigenvectors of unit length and λ_j the eigenvalues of the covariance. The eigenvectors form an orthogonal frame of the neighborhoods principal components.

The order of the eigenvalues is defined as $\lambda_1 > \lambda_2 > \lambda_3$. The eigenvector $\mathbf{v}_3$, which corresponding to the smallest eigenvalue represents the principal component with the smallest variance, which in structured environments is likely to be a surface normal [33].

The distributions are chosen such that it models high covariance in a plane, and low covariance in the points surface normal direction

$$C_n = \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.31}$$

where ϵ is a small constant describing the covariance in the plane. Given the surface normal vector $\mathbf{v}_3$ for a point p_i the corresponding covariance matrix C_i is rotated such that ϵ describes the uncertainty in the respective surface normals as

$$C_i = R_{\mathbf{v}_3} C_n R_{\mathbf{v}_3}^\top, \tag{3.32}$$

where $R_{\mathbf{v}_3}$ is the rotation that transforms the standard basis vector $e_1 \to \mathbf{v}_3$.

Assuming perfect correspondence between two clouds they are associated by the correct transformation T^* such that

$$\hat{b}_i = T^* \hat{a}_i, \tag{3.33}$$

where T^* contains the correct rotation R^* and translation t^*. Note that the authors of G-ICP simplify the transform notation, where a transformation is applied to a point e.g, Ta_i, it instead implies $Ra_i + t$, although dimensions of the operation does not agree.

However, the assumption of perfect correspondences is contradicted by sampling clouds with a LiDAR. Thus, a disparity of the association for an arbitrary T is defined as

$$d_i(T) = b_i - Ta_i. \tag{3.34}$$

Since a_i and b_i are assumed to be drawn from independent Gaussian distributions the disparity for the transform can be expressed as

$$d_i(T) \sim \mathcal{N}(b_i - (T)a_i, C_i^{\mathcal{B}} + (R)C_i^{\mathcal{A}}(R)^\top), \tag{3.35}$$

where the distribution of the correct transform T^* is expressed as

$$d_i(T^*) \sim \mathcal{N}(0, C_i^{\mathcal{B}} + (R^*)C_i^{\mathcal{A}}(R^*)^\top). \tag{3.36}$$

The transform which minimize this disparity is found by using Maximum Likelihood Estimation (MLE) to iteratively compute T by setting

$$T = \underset{T}{argmax} \prod_i p(d_i(T)) = \underset{T}{argmax} \sum_i log(p(d_i(T))), \tag{3.37}$$

which can be simplified to

$$T = \underset{T}{argmin} \sum_i d_i(T)^\top (C_i^{\mathcal{B}} + TC_i^{\mathcal{A}}T^\top)^{-1} d_i(T). \tag{3.38}$$

In this thesis the minimization problem is then solved with Scipy's [34] implementation of the nonlinear conjugate gradient method in [35] and replaces the minimization step in the standard ICP implementation.

3.4 Pre-Processing

Applications where ICP is implemented generally utilize some pre-processing on the point clouds. The computational cost of ICP can be large even with a low computational complexity due to the sheer amount of data. One approach to this is to down-sample data, as it lowers the computational cost significantly. Common approaches to this are random and uniform sampling, although it implies loss of potentially useful information [4].

To ensure that calculations are made on useful information some points in the clouds can be removed only based on distance from the sensor. Such filters discards points based on Euclidean or coordinate distance. A lower Euclidean bound can be set such that readings from the ego-vehicle are neglected. Due to the sensor not measuring

at the exact same angle for each scan-cycle point correspondences at large distances likely result in large disparities, especially throughout motion. Thus, an upper bound could also be effective. Additionally, as the LiDAR is likely to observe readings from the ground, a bound in the z-coordinate can be set as well to remove feature-less points.

3.5 Matching Rejection

A core assumption of ICP is that for each point in the source cloud there exists a true point correspondence in the target cloud. This assumption is fundamentally contradicted when sampling the point-clouds with a LiDAR, as with noise each scan is a unique sampling of the surrounding environment rather than the same, but transformed, cloud.

Furthermore, if the source and target clouds are sampled from different positions some occlusion of the environment is likely. Thus, the unique samples also consists of partly unique environments. There exists different options to handle these issues, such as defining a threshold for alignment error, or by rejecting matched pairs which can be ruled out as corresponding points, i.e. outlier rejection. This section describes the methods used to find and reject these types of points.

3.5.1 Disparity Based Rejection

A simple and intuitive method for measuring the reliability of point correspondences is to observe the euclidean distance between the matched points [36]. Given a threshold for what qualifies for a probable match one can reject less probable correspondences. In this thesis this is implemented by using the already computed distances based on the individual distance metrics for the given ICP algorithm.

3.5.2 Surface Normal Based Rejection

As a perfect point correspondence is unlikely, extra information of the sampled underlying surface can be extracted to ensure that points are sampled from common features [6]. The local geometry is captured with PCA (as described in Section 3.3.2). For each matched pair the surface normal orientation is compared, where orientational deviations over a threshold is rejected.

A pair $\{p^a, p^b\}$ is rejected if the relationship

$$cos^{-1}(\mathbf{v}_3^a \cdot \mathbf{v}_3^b) < \alpha \tag{3.39}$$

does not hold, where $p^a \in \mathcal{A}$, $p^b \in \mathcal{B}$, α is a set threshold, and $\mathbf{v}_3^a$,$\mathbf{v}_3^b$ is the eigenvector corrsponding to the smallest eigenvalue drawn from a surface in the cloud

represented by its superscript. The method was highly recommended in [37] due to its general increase in performance for different configurations and scenarios.

3.6 Cloud Rectification - Velocity Updating ICP

Velocity updating Iterative Closest Point (V-ICP) is an enhancement to the family of ICP algorithms by compensating for distortions in clouds caused by motion while sampling, proposed in [38]. The nested structure of V-ICP uses the estimated transformation of ICP to transform individual points in a gathered cloud such that they are synced to a common point in time, as if the cloud were sampled instantaneously. A visualization of this is presented in Figure 3.6, where the left image displays two distorted sequential point-clouds, $\mathcal{A}$ and $\mathcal{B}$, and the estimated motion throughout gathering the clouds. The right image displays the desired result of V-ICP, where the rectification process successfully restored the point-clouds to accurately represent the environment.

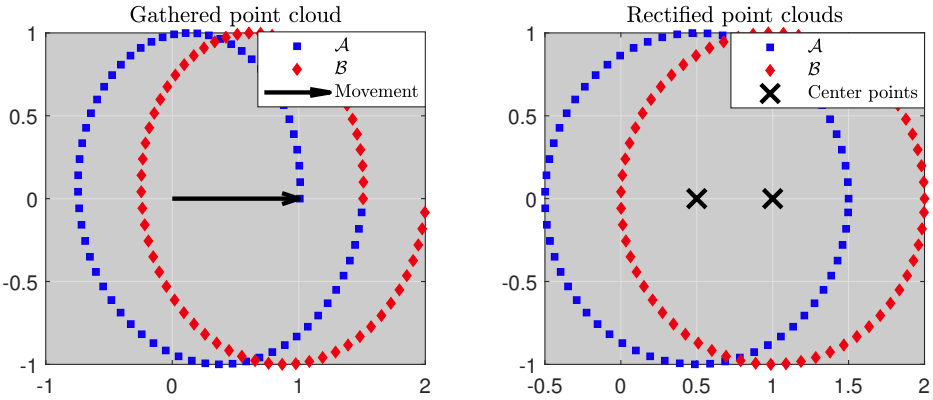


Figure 3.6: Two point clouds gathered in a cylindrical room at a constant velocity of 0.5 units/lap in the x-direction which is rectified to compensate for the movement while sampling the points.

In order to perform individual transformations to each point in a point cloud, the discrete transformation estimate from the ICP algorithm is converted to a continuous transformation. The report in [39] presents a multitude of ways to perform such a conversion. The version used here is stated as

$$T_c = \frac{ln(T_d)}{\Delta t}, \ \Delta t = t_i - t_{i-1}, \tag{3.40}$$

where T_d is the discrete transform, and Δt the time between the current source and target point clouds. The continuous transformation matrix is then converted to a state transition matrix as

$$\Phi(t) = e^{t \cdot T_c}. \tag{3.41}$$

This is then applied to transform each individual point "forward in time" to the time of the last sampled point, under the assumption that the transformation is constant throughout gathering the cloud. A visualization of the algorithm is presented in Figure 3.7, where the transformation estimation with ICP and cloud rectification is alternated until convergence. Further, a detailed description of the algorithm is stated in Algorithm 1.

Algorithm 1: V-ICP
x_i^k is a set of points from the cloud X_k, sampled at time k, and $\hat{X}$ denotes a cloud that has been rectified.

$T_{c,k} = T_{c,k-1}$
while $||T_{c,k}^- - T_{c,k}|| > \epsilon$ **do**
  $\Phi_i(t) = e^{t \cdot T_{c,k}}$
  for *i=1:n* **do**
    $\hat{x}_i^k = \Phi_i(t_n^k - t_i^k) x_i^k$
  $T_{d,k} =$
   $ICP(\hat{X}_{k-1},\ \hat{X}_k,\ T_{d,k-1})$
  $T_{c,k}^- = T_{c,k}$
  $T_{c,k} = \frac{ln(T_{d,k})}{\Delta t}$
Return $T_{d,i}$

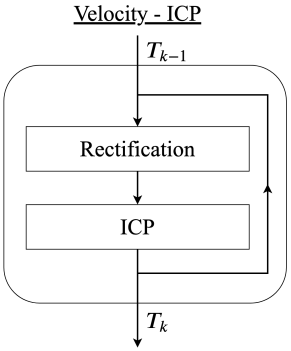


Figure 3.7: Simplification of the Velocity Updating Iterative Closest Point algorithm

4

Filtering

Bayesian statistics is a statistical approach created in the 18th century, and was introduced in the topic of filtering in mid 20th century [40]. Bayesian statistics treat results as approximations of parameters or their probability distributions by modeling uncertainties in observations and systems. This approach will be used to improve the ego-motion parameters derived from the point cloud matching algorithms and IMU readings.

The theory of filtering and smoothing in the following sections will be described with a Bayesian approach and the fundamental framework is based on the book by Simo Särkkä [41]. For further understanding, the reader is recommended to consult this book.

4.1 Bayesian Filtering

The term *optimal filtering* regards statistical optimality of a filtering process. *Bayesian filtering* is a set of methods which with a Bayesian approach estimates the state $\boldsymbol{x_k}$ of a system for a given time k with statistical optimality.

In *Bayesian filtering* the state is generally expressed as a vector containing the parameters of interest for a given application. The system is indirectly observed through measurements $\boldsymbol{y_k}$ affected by noise.

The aim of *Bayesian filtering* is to for a sequence of unknown states

$$\boldsymbol{x}_{0:k} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\}, \tag{4.1}$$

estimate the joint *posterior distribution* of the states given the measurements

$$\boldsymbol{y}_{1:k} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k\}. \tag{4.2}$$

This can be performed through *Bayesian inference*, i.e, by applying *Baye's rule*,

$$p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k}) = \frac{p(\boldsymbol{y}_{1:k}|\boldsymbol{x}_{0:k})p(\boldsymbol{x}_{0:k})}{p(\boldsymbol{y}_{1:k})}, \tag{4.3}$$

where $p(\boldsymbol{x}_{0:k})$ is the *prior* distribution and $p(\boldsymbol{y}_{1:k}|\boldsymbol{x}_{0:k}) = l(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k})$ is the *likelihood function*. Further, the *normalization constant* $p(\boldsymbol{y}_{1:k})$ is defined as

$$p(\boldsymbol{y}_{1:k}) = \int p(\boldsymbol{y}_{1:k}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}. \tag{4.4}$$

However, this requires a recalculation of the full posterior distribution for each new observation. By modelling a system as a *Markov process*, i.e. the future state of a system is assumed to only be dependent on the current state and none of the preceding sequence of states, while noise vectors are assumed independent, this requirement is dropped. Thus, (4.3) can be compute up to proportionality as

$$p(\boldsymbol{x}_k|\boldsymbol{y}_k) \propto p(\boldsymbol{y}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}), \tag{4.5}$$

where we seek to compute $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$.

In *Bayesian filtering*, three different distributions are of particular interest;

- *Predictive distributions*, given the current and previous measurements, predict distributions of future states $\boldsymbol{x}_{k+n}$, n steps after state $\boldsymbol{x}_k$

$$p(\boldsymbol{x}_{k+n}|\boldsymbol{y}_{1:k}). \tag{4.6}$$

- *Filtering distributions*, given the current and previous measurements, estimate distributions of state $\boldsymbol{x}_k$

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}). \tag{4.7}$$

- *Smoothing distributions*, given measurements from an interval $[1:N], N > k$, estimate distributions of state $\boldsymbol{x}_k$

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:N}). \tag{4.8}$$

These distributions are utilized in algorithms such as the Kalman Filter (KF) and the Rauch-Tung-Striebel-Smoother (RTSS) where there exists a closed form solution. These algorithms will be discussed further in the coming sections.

4.2 Kalman Filtering

The Kalman Filter [42] is a closed form solution for Bayesian filtering equations of linear and Gaussian models defined as

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{A}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{q}_{k-1}, \\ \boldsymbol{y}_k &= \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{r}_k, \end{aligned} \tag{4.9}$$

where $\boldsymbol{x}_k$ is the state, $\boldsymbol{y}_k$ is a measurement, $\boldsymbol{q}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}_{k-1})$ is the process noise, $\boldsymbol{r}_k \sim \mathcal{N}(0, \boldsymbol{R}_k)$ is the measurement noise, $\boldsymbol{x}_0 \sim \mathcal{N}(\hat{\boldsymbol{x}}_0, \boldsymbol{P}_{0|0})$ is the Gaussian prior distribution, $\boldsymbol{A}_{k-1}$ is the transition matrix and $\boldsymbol{H}_k$ is the measurement model.

A linear model can be evaluated in closed form with the Bayesian filtering equations to obtain the following Gaussian distributions:

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) &= \mathcal{N}(\boldsymbol{x}_k|\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{P}_{k|k-1}), \\ p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) &= \mathcal{N}(\boldsymbol{x}_k|\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k}), \\ p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}) &= \mathcal{N}(\boldsymbol{y}_k|\boldsymbol{H}_k\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top + \boldsymbol{R}_k). \end{aligned} \tag{4.10}$$

Kalman filters recursively compute a predicted mean and covariance $\hat{\boldsymbol{x}}_{k|k-1}$, $\boldsymbol{P}_{k|k-1}$ through a *prediction step*, then a posterior mean and covariance $\hat{\boldsymbol{x}}_{k|k}$, $\boldsymbol{P}_{k|k}$ through a *update step*:

Prediction step: (Computes predicted mean and covariance, $\hat{\boldsymbol{x}}_{k|k-1}$, $\boldsymbol{P}_{k|k-1}$)

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k-1} &= \boldsymbol{A}_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1} \\ \boldsymbol{P}_{k|k-1} &= \boldsymbol{A}_{k-1}\boldsymbol{P}_{k|k-1}\boldsymbol{A}_{k-1}^\top + \boldsymbol{Q}_{k-1} \end{aligned} \tag{4.11}$$

Update step: (Computes posterior mean and covariance, $\hat{\boldsymbol{x}}_{k|k}$, $\boldsymbol{P}_{k|k}$)

$$\begin{aligned} \boldsymbol{v}_k &= \boldsymbol{y}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_{k|k-1} \\ \boldsymbol{S}_k &= \boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top + \boldsymbol{R}_k \\ \boldsymbol{K}_k &= \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top\boldsymbol{S}_k^{-1} \\ \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\boldsymbol{v}_k \\ \boldsymbol{P}_{k|k} &= \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{S}_k\boldsymbol{K}_k^\top \end{aligned} \tag{4.12}$$

4.3 Bayesian Smoothing

For applications where measurements beyond time k is available it is possible to compute a marginal distributions conditional on the entire measurement sequence up until time N, where $N > k$, with the use of *Bayesian smoothing*. There exists a few

different types of smoothing, although only *fixed-interval* smoothing is considered here.

The Bayesian optimal smoothing equations for the smoothing distributions in (4.8) is formulated as

$$\begin{aligned} p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k}) &= \int p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})d\boldsymbol{x}_k, \\ p(\boldsymbol{x}_k|\boldsymbol{y}_{1:N}) &= p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) \int \frac{p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:N})}{p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k})} d\boldsymbol{x}_{k+1}, \end{aligned} \tag{4.13}$$

and can recursively be computed backwards after the filtering distributions are obtained.

4.4 Rauch-Tung-Striebel Smoothing

Rauch-Tung-Striebel-Smoother [43], also called *Kalman smoother*, is a so called *forward-backward smoother*. RTSS is based on the Kalman filtering equations, and computes the closed form smoothing solution

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:N}) = \mathcal{N}(\boldsymbol{x}_k; \hat{\boldsymbol{x}}_{k|N}, \boldsymbol{P}_{k|N}) \tag{4.14}$$

to linear and Gaussian filtering models, where $N > k$.

RTSS perform the same forward recursion as a KF, and further utilize backward recursion for re-evaluating the posterior densities as

$$\begin{aligned} \boldsymbol{G}_k &= \boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top \boldsymbol{P}_{k+1|k}^{-1}, \\ \hat{\boldsymbol{x}}_{k|N} &= \hat{\boldsymbol{x}}_{k|k} + \boldsymbol{G}_k(\hat{\boldsymbol{x}}_{k+1|N} - \hat{\boldsymbol{x}}_{k+1|k}), \\ \boldsymbol{P}_{k|N} &= \boldsymbol{P}_{k|k} - \boldsymbol{G}_k(\boldsymbol{P}_{k+1|k} - \boldsymbol{P}_{k+1|N})\boldsymbol{G}_k^\top, \end{aligned} \tag{4.15}$$

where the moments $\boldsymbol{x}_{k+1|k}$, $\boldsymbol{P}_{k+1|k}$, $\boldsymbol{x}_{k|k}$ and $\boldsymbol{P}_{k|k}$ are stored during the forward pass.

5

Implementation

Chapters 3 and 4 describe theory behind alignment of point clouds and how Bayesian statistics is used to improve estimates. The following chapter explains how estimates of ego-motion are made by utilizing these proposed methods such that an evaluation can be made. With the modular toolbox for point cloud matching described in chap-

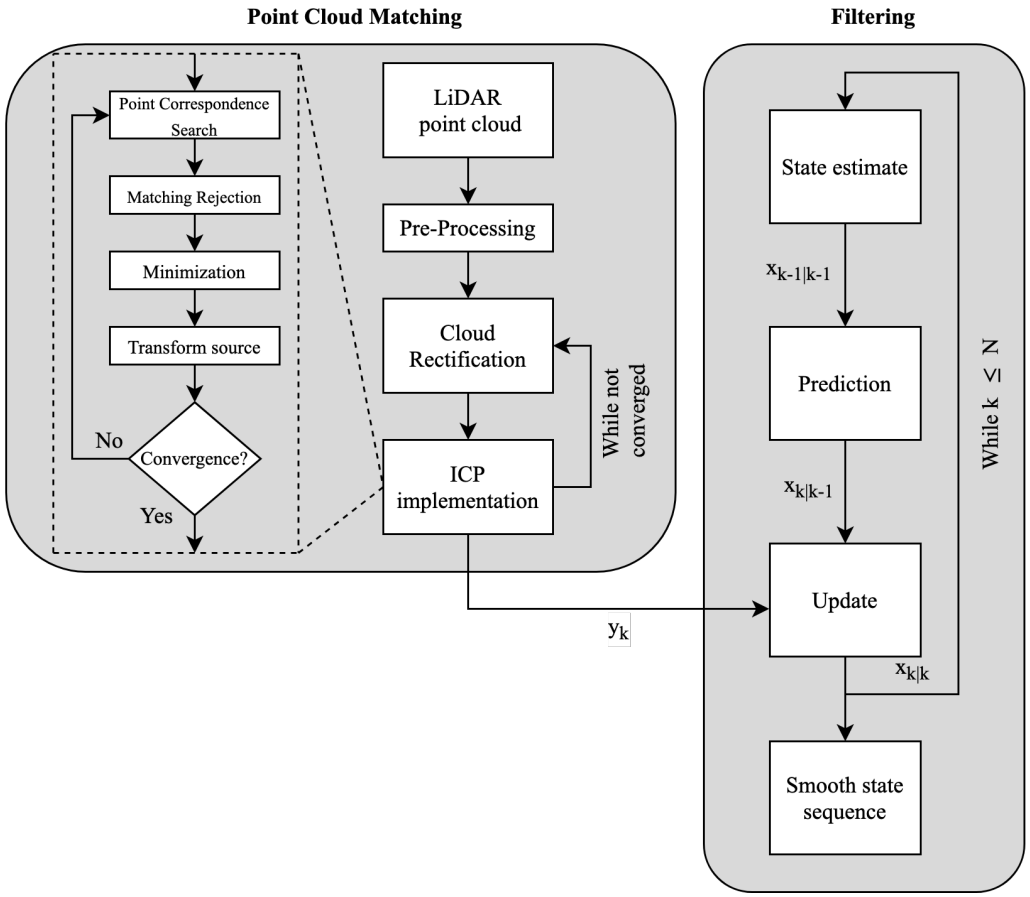


Figure 5.1: Flowchart of the proposed algorithm. Measurements are obtained through the process in the *Point Cloud Matching* block to the left, and is used as noisy observations of the state in the Filtering block to the right.

ter 3 the modules of the point cloud matching algorithm can easily be interchanged, creating different configurations. The algorithm begins with sampling a point cloud,

defining the first frame as the source cloud, and the following frame as the target cloud. These clouds are passed through the point cloud matching algorithm, yielding a measurement y_k. The measurement is then processed by a Kalman filter as a noisy observation of the state x for each time instance k. When the complete sequence is processed, further improvements of the ego-motion estimations are made with the backwards pass of an RTSS. An overview of the algorithm can be seen in Figure 5.1.

The following sections will describe the different implemented configurations of the point cloud matching block, followed by the choice of models and parameters in the Filtering block.

5.1 Point Cloud Matching

The *Point Cloud Matching* block presented in Figure 5.1 illustrates the flow through different modules. All modules are subject to certain tunable parameters which are defined in the following sections. Further, all modules are not necessary for the algorithm to function and can be combined into different configurations. Thus, a set of configurations which are implemented and evaluated is also presented.

5.1.1 Pre-Processing

Pre-processing is the first module applied to a newly sampled point cloud. As discussed in Section 3.4 distance based removal of points can be performed in different ways to handle different issues. First a lower bound on sampled distance is set to 0.5 m in order to discard all points from the ego-vehicle. Secondly, an upper bound is set to 40 m in order to neglect points with a higher probability to result in larger disparities.

Further, in Figure 5.2 two LiDAR point clouds gathered at different locations and times are visualized, where the circular patterns in the center of the images consist of points which origins from floor and ceiling.

In the sensors local coordinate frame these points appear very similar in the two frames. Thus, ICP is likely to match the circular patterns generated by the rotational nature of the sensor itself. In other words, the circular patterns become false features for ICP. Such false features could cause the ICP algorithm to converge to a local minimum. In attempt to omit such features a lower and upper bound in z-direction is set to -0.7 m and 0.3m respectively. This ensures that no readings originates from floor or ceiling. However, it does neglect small portions of other, relevant, points as well.

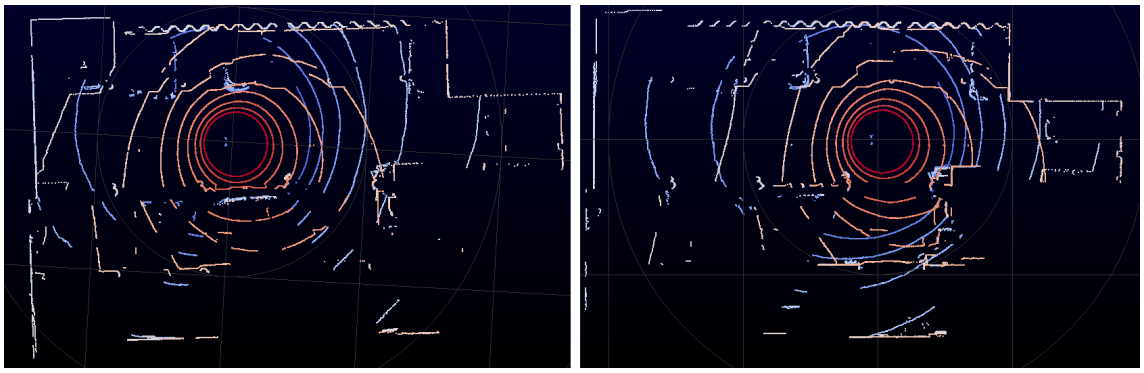


Figure 5.2: Visualization of two point clouds sampled at different locations. Rotational LiDARs create circular patterns in the floor and ceiling, demonstrating how these circles create similar features at different locations.

5.1.2 Cloud Rectification

After pre-processing a new cloud, it is passed into the cloud rectification module in order to reduce distortions caused by ego-motion while sampling. This is performed based on an initial guess of the true transformation. The initial guess is set to $T_{k,init} = T_{k-1}$, as vehicular motion is expected to be smooth, where the target cloud is rectified thereafter.

Next, one of the three ICP implementations (i.e. ICP, Mb-ICP or G-ICP) is processed which yields a new estimate of the transformation T_k at time k. Cloud rectification is hereafter re-iterated until T_k converges, i.e. until rectification no longer affects the estimate returned by the ICP implementation. In order to allow for noise and imperfections a threshold for convergence is set on the change of T_k between iterations. Iteration is continued until reaching the threshold quantified as

$$||T_k^{i-1} - T_k^i||_2 > RT, \tag{5.1}$$

where T_k^i and T_k^{i-1} is the current and previous transformation estimate for time k. Convergence threshold for rectification, RT, is set to 0.01, and is complemented with a bound of a maximum of 10 iterations.

5.1.3 ICP Implementation

The ICP implementation starts off with a point correspondence search, which for each point in the source cloud, finds the nearest neighbor in the target cloud. If Mb-ICP is the implementation of choice, a modified version using metric-based distance is used instead. This metric contains a design parameter L which weighs the rotational and translational component of a transformation between two points.

The authors of the article introducing Mb-ICP in [10] suggests using $L = 3$, which is used here.

Corresponding points are matched and passed through the matching rejection module in an attempt to find and discard any outliers. In this module, disparity based rejection algorithm is implemented with a disparity threshold of 0.2 m. Meanwhile, surface-normal based rejection is implemented to analyze the structure of the local geometry of each point in order to extract its surface normal. Due to rotational LiDARs having a lower vertical than horizontal resolution the radius parameter has to be driven by the vertical resolution. That is, the radius has to be large enough to capture the structure in vertical direction such that measurements from two different lasers are obtained. A radius of 0.5 m is chosen, which with the Velodyne VLP 16 LiDAR used throughout testing corresponds to being able to capture readings from multiple lasers on orthogonal surfaces up to 11.5 m away from the sensor. The threshold for surface normal deviation α was set to $\alpha = 45°$, where matched points with a larger deviation are rejected.

Remaining corresponding points are then subject to a minimization step, where the minimization method depends on which ICP implementation was chosen. This step attempts to find the transformation which minimize the distance between matched point.

If G-ICP is the implementation of choice, a radius based neighborhood search is used to compute the structure of the local area around both points in each matched point pair. The radius for the search was set to 0.5m, similar to surface-normal based rejection. Meanwhile, ϵ which describes the uncertainty in the surface normal was set to $\epsilon = 0.001$.

The resulting transformation is applied to the source cloud and an alignment error is computed. If this alignment error is reduced more than a given tolerance, the process is re-iterated until convergence is reached. The alignment error is defined as

$$AE = \sum_{i}^{N} \frac{||b'_i - T_k a'_i||_2}{N}, \tag{5.2}$$

where the point a'_i from the source cloud is aligned to its corresponding point b'_i in the target cloud with the transformation T_k at time k. The threshold for convergence of the alignment error is set to 0.01, and is complemented with a bound of a maximum of 10 iterations.

5.1.4 Conversion from Transformation to Measurement

When convergence is reached, the resulting transformation matrix T_k is converted into the measurement y_k, which is further refined by the Filtering block as described

in the upcoming section. This conversion is performed as

$$v = \frac{||t||_2}{\Delta t}, \tag{5.3}$$

$$(\phi, \theta, \psi) = \begin{cases} if \ -1 < R_{(3,1)} < 1: & \begin{cases} \phi = \frac{arctan2(R_{(3,2)}, R_{(3,3)})}{\Delta t} \\ \theta = \frac{arcsin(-R_{(3,1)})}{\Delta t} \\ \psi = \frac{arctan2(R_{(2,1)}, R_{(1,1)})}{\Delta t} \end{cases} \\ if \ R_{(3,1)} \leq -1: & \begin{cases} \phi = 0 \\ \theta = \frac{\pi}{2\Delta t} \\ \psi = \frac{-arctan2(-R_{(2,3)}, R_{(2,2)})}{\Delta t} \end{cases} \\ otherwhise: & \begin{cases} \phi = 0 \\ \theta = \frac{-\pi}{2\Delta t} \\ \psi = \frac{arctan2(-R_{(2,3)}, R_{(2,2)})}{\Delta t} \end{cases} \end{cases}, \tag{5.4}$$

where v is the speed, ϕ the roll rate, θ the pitch rate, ψ the yaw rate and where R and t are the rotational and translational components of the transformation, as defined in Section 3.1.

5.1.5 Point Cloud Matching Configurations

All modules are not necessary for the algorithm to function, as some rather improves upon the chain of operations. A set of configurations is set up in order to analyze the feasibility of using the best performing configuration and how individual modules effect performance. The combination of modules which makes out ICP, Mb-ICP and G-ICP are the basis for all evaluation and is extended with additional modules in iterations. The final configurations are listed in Table 5.1. Pre-processing was found to be necessary to yield reasonable estimates, therefore it is included in most combinations.

Table 5.1: The different configurations tested and analyzed, where each configuration is added onto ICP, Mb-ICP and G-ICP respectively.

	Filtering	Pre-Processing	Matching Rejection	Cloud Rectification
Config. 1				
Config. 2				
Config. 3				
Config. 4				
Config. 5				

5.2 Filtering

By assuming a system is linear and discrete, one finds the discrete state transition of the systems state as

$$x_k = Ax_{k-1} + Dq_{k-1}, \tag{5.5}$$

where x defines the state, A the discrete state transition matrix and q the process noise of the system.

Here, the motion parameters $v, \dot{\theta}, \dot{\phi}$ and $\dot{\psi}$ are modeled as independent from each other while dependent on their respective derivative. Thus, a motion model for each individual parameter and their derivatives can be expressed as

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \; D = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \tag{5.6}$$

where Δt defines the discrete time step of the system which was set to $\Delta t = 0.105$. This motion model can be seen as a discretized *constant acceleration model* where accelerations are modeled as constant.

Meanwhile, the process noise q for each parameter is unknown, but can be estimated based on a set of assumptions. The approach in this thesis is to assume the noise to be white and zero mean, with the distribution as

$$q_k \sim \mathcal{N}(0, Q_k), \tag{5.7}$$

where Q_k is the process noise covariance.

Further, as described in [44, p.269], process noise covariance in the continuous domain can be expressed as the auto-correlation of the process noise as

$$Q(t) = E[\tilde{q}(t)\tilde{q}(\tau)^\top] = \tilde{\rho}(t)\delta(t-\tau). \tag{5.8}$$

Assuming $\tilde{\rho}$ to be time-invariant, this is simplified as

$$Q = E[\tilde{q}(t)\tilde{q}(\tau)^\top] = \tilde{\rho}\delta(t-\tau) \; \forall t. \tag{5.9}$$

Meanwhile, the discrete-time domain correspondence of the process noise can be expressed as

$$q_k = \int_0^{\Delta t} e^{A_c(\Delta t - \tau)} D\tilde{q}(k\Delta t + \tau)\ d\tau, \tag{5.10}$$

where A_c is the continuous-time state transition matrix [44, p.270]. From this, the process noise covariance can be derived as

$$\begin{aligned} Q = E[q_k q_k^\top] &= E[\int_0^{\Delta t} e^{A_c(\Delta t-\tau)} D\tilde{q}(k\Delta t+\tau)\tilde{q}(k\Delta t+\tau)^\top D^\top e^{A_c(\Delta t-\tau)^\top}\ d\tau] \\ &= ADD^\top A^\top E[\int_0^{\Delta t} \tilde{q}(k\Delta t+\tau)\tilde{q}(k\Delta t+\tau)^\top\ d\tau] \\ &= ADD^\top A^\top \tilde{\rho}. \end{aligned} \tag{5.11}$$

This results in a process noise covariance as

$$Q = \begin{bmatrix} \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} & \Delta t \end{bmatrix} \tilde{\rho}, \tag{5.12}$$

which consists only of a single design parameter, the *power spectral density* $\tilde{\rho}$.

Given a state consisting of all ego-motion and accelerations as

$$x = \left[v, a, \dot{\theta}, \dot{\phi}, \dot{\psi}, \ddot{\theta}, \ddot{\phi}, \ddot{\psi}\right]^\top, \tag{5.13}$$

a combined motion model for the system can be expressed as

$$A = \left[\begin{array}{cc:cccccc} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right], \quad D = \left[\begin{array}{cc:cccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]. \tag{5.14}$$

Further, noise cross-covariance between individual motion parameters is assumed to be negligible, resulting in a combined process noise covariance matrix as

$$Q = \left[\begin{array}{cc:cccccc} \frac{\Delta t^3}{3}\tilde{\rho}_v & \frac{\Delta t^2}{2}\tilde{\rho}_v & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\Delta t^2}{2}\tilde{\rho}_v & \Delta t\tilde{\rho}_v & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & \frac{\Delta t^3}{3}\tilde{\rho}_{\dot{\theta}} & 0 & 0 & \frac{\Delta t^2}{2}\tilde{\rho}_{\dot{\theta}} & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^3}{3}\tilde{\rho}_{\dot{\phi}} & 0 & 0 & \frac{\Delta t^2}{2}\tilde{\rho}_{\dot{\phi}} & 0 \\ 0 & 0 & 0 & 0 & \frac{\Delta t^3}{3}\tilde{\rho}_{\dot{\psi}} & 0 & 0 & \frac{\Delta t^2}{2}\tilde{\rho}_{\dot{\psi}} \\ 0 & 0 & \frac{\Delta t^2}{2}\tilde{\rho}_{\dot{\theta}} & 0 & 0 & \Delta t\tilde{\rho}_{\dot{\theta}} & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^2}{2}\tilde{\rho}_{\dot{\phi}} & 0 & 0 & \Delta t\tilde{\rho}_{\dot{\phi}} & 0 \\ 0 & 0 & 0 & 0 & \frac{\Delta t^2}{2}\tilde{\rho}_{\dot{\psi}} & 0 & 0 & \Delta t\tilde{\rho}_{\dot{\psi}} \end{array}\right], \tag{5.15}$$

which contains four design parameters, the individual power spectral densities $\tilde{\rho}$ corresponding to $v, \dot{\theta}, \dot{\phi}$ and $\dot{\psi}$, respectively. To simplify the tuning process each parameter was set to the same value, such that $\tilde{\rho}_v = \tilde{\rho}_{\dot{\theta}} = \tilde{\rho}_{\dot{\phi}} = \tilde{\rho}_{\dot{\psi}} = \tilde{\rho}$.

The power spectral density was estimated by non-linear parameter search for all configurations and extensions, and was chosen to minimize the Root Mean Square Error (RMSE) of resulting estimates. However, the resulting power spectral density for the different extensions was vaguely different. Thus, the values for the power spectral density was defined only by which configuration was used, with the resulting values presented in Table 5.2.

	$\tilde{\rho}$
Configuration 1:	7.4
Configuration 2:	20
Configuration 3:	403
Configuration 4:	20
Configuration 5:	403

Table 5.2: The power spectral density value for the different configurations of the ICP modules.

Meanwhile, measurements of the state is obtained through the point matching process visualized in Figure 5.1, and are noisy observations of the speed and angular velocities. The measurement vector is stated as

$$y = \left[v, \dot{\theta}, \dot{\phi}, \dot{\psi}\right]^\top. \tag{5.16}$$

Measurements and state are related through the *measurement model*, and describes noisy observations of the state as

$$y_k = Hx_k + r_k, \tag{5.17}$$

where the measurement model H is defined as

$$H = \left[\begin{array}{cc:cccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right], \tag{5.18}$$

and the measurement noise $r_k \sim \mathcal{N}(0, R_k)$. Measurement noise covariance matrices R_k for all configurations were sampled with the sensor being stationary and with no moving objects within the sensors field of view. Thus, the true state is known to be zero for all parameters, and the characteristics of the noise can be estimated as well as any potential bias. This was performed for each configuration of the algorithm to ensure as accurate estimate of the error characteristics as possible. The resulting measurement noise covariance matrices is listed in Appendix C.

6

Testing and Evaluation

This chapter begins by describing the tests set up and the environment they were performed in. Further, it describes all evaluation metrics and graphs used to quantify the results.

6.1 Testing Environment

The main goal is to implement a functioning algorithm during strict circumstances where there exist visible and static landmarks. This will be set up by using a hand-picked designated area for gathering data which is later evaluated offline.

The chosen environment is the garage floor of an office building, as it is a highly controllable environment such that tests that follow the scope can be performed with ease. The garage contains pillars and vehicles, which is in line with the types of features which was sought for. A few scenarios are staged to test different aspects of the algorithms and are presented in the following sections.

6.1.1 Test Case 1

In test case 1 the vehicle moves in a circle around a pillar of the garage floor, attempting to keep a constant speed and yaw-rate. An illustration of test case 1 can be seen in Figure 6.1. The motion is performed as fast as possible to observe how the algorithms handle hard cornering.

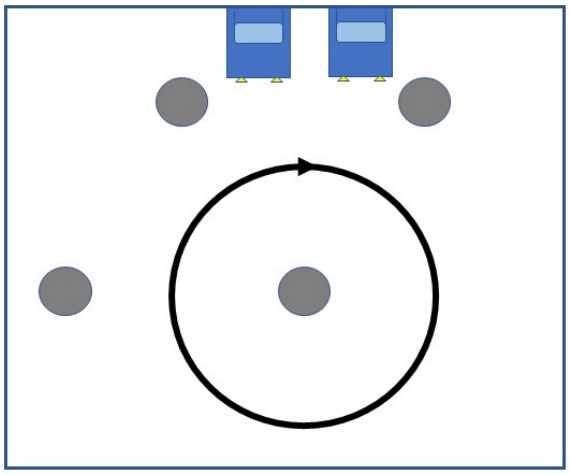


Figure 6.1: Illustration of the motion performed in Test case 1, where pillars and cars present are included.

6.1.2 Test Case 2

Test case 2 starts from standing still, accelerating for a few seconds, then braking until standing still. The resulting straight motion can be observed in Figure 6.2. Through this motion only translational components should be observed, where the ability to cope with changes in speed is of interest.

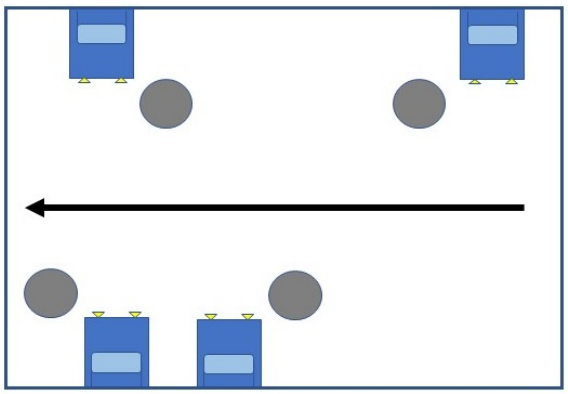


Figure 6.2: Illustration of the motion performed in Test case 2, where pillars and cars present are included.

6.1.3 Test Case 3

Test case 3 is performed similarly to test case 2, where the ego-vehicle started from standing still. Although throughout the acceleration a turn is made, completing the motion by braking until standing still. The motion can be seen in Figure 6.3. The scenario tests similar features as in test case 2 with the addition of rotational components.

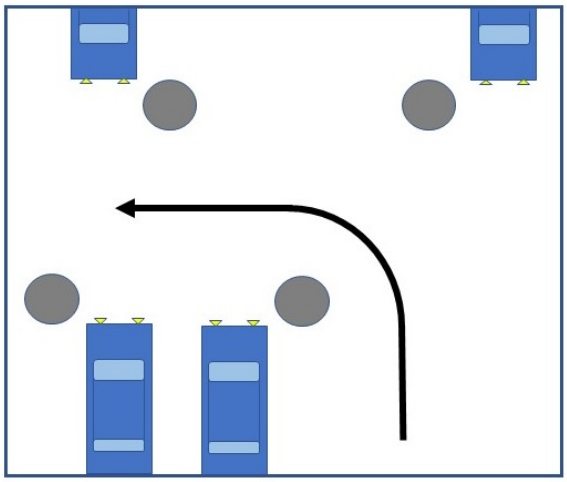


Figure 6.3: Illustration of the motion performed in Test case 3, where pillars and cars present are included.

6.1.4 Test Case 4

For test case 4 a similar motion is performed as in test case 3, although with an additional turn as can be seen in Figure 6.4. Starting from standing still, accelerating while turning left, then right, then braking until standing still. The features tested are similar to test case 3, adding a change of sign to the rotational components.

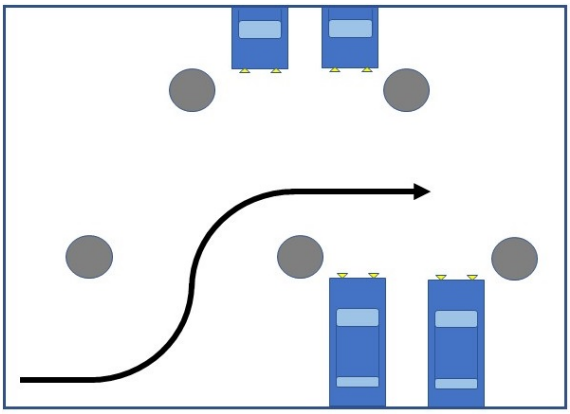


Figure 6.4: Illustration of the motion performed in Test case 4, where pillars and cars present are included.

6.1.5 Test Case - Roll & Pitch

The tests in section 6.1.1-6.1.4 are performed on flat surfaces which will not give any significant change in roll and pitch rates. Due to the LiDARs difference in vertical and horizontal resolution, the performance for estimating pitch and roll rates can be

interesting. Thus, an additional test presented in Figure 6.5 was set up to observe how changes in roll and pitch rates are handled.

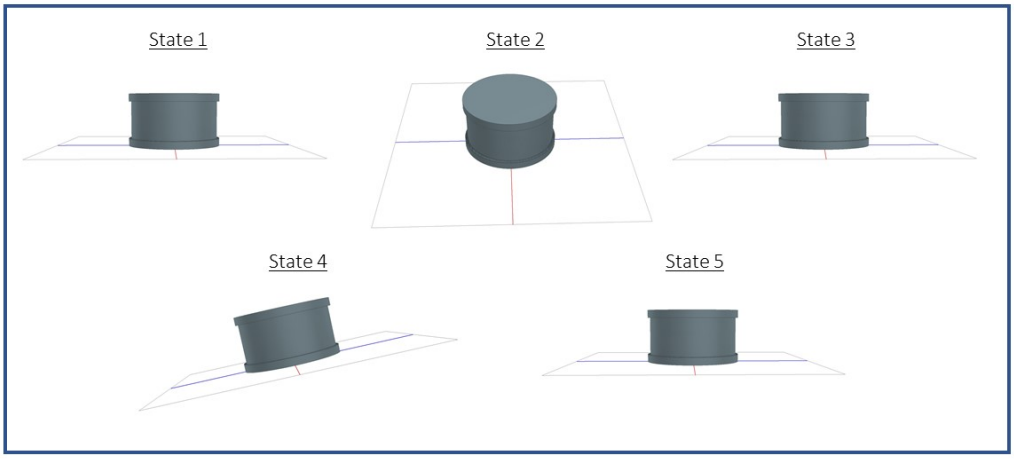


Figure 6.5: Visualization of the motion throughout the roll and pitch rate test case.

The test starts with tilting the sensor forward, and then back to flat. Second the sensor is tilted to the side, and then back to flat again.

6.2 Evaluation

In order to evaluate the performance of the proposed LiDAR-based solution the filtered and smoothed IMU data will be used as a reference in combination with wheel speed measured by the Hall-sensors. The process of selecting the noise covariance matrices of the IMU was performed as in Section 5.2. The power spectral density was set to $\tilde{q} = 200$, the sample rate of 600Hz leading to $\Delta t \approx 0.00167$ and the measurement noise can be seen in Appendix C.

The resulting performance of the different configurations will be presented in two ways. First, by a *Box-Plot*, which visualize the distribution of the relative errors. Secondly, by the Root Mean Square Error metric, in order to quantify performance and make it easy to compare performance in-between different configurations.

6.2.1 Box Plots

In order to present a qualitative comparison of the results it is important that they are presented in a fashion where drawing unambiguous conclusions is straight forward. By merely comparing line plots the result can be hard to evaluate.

To provide insight of the algorithms performance, the results will be presented in the form of *Box plots*.

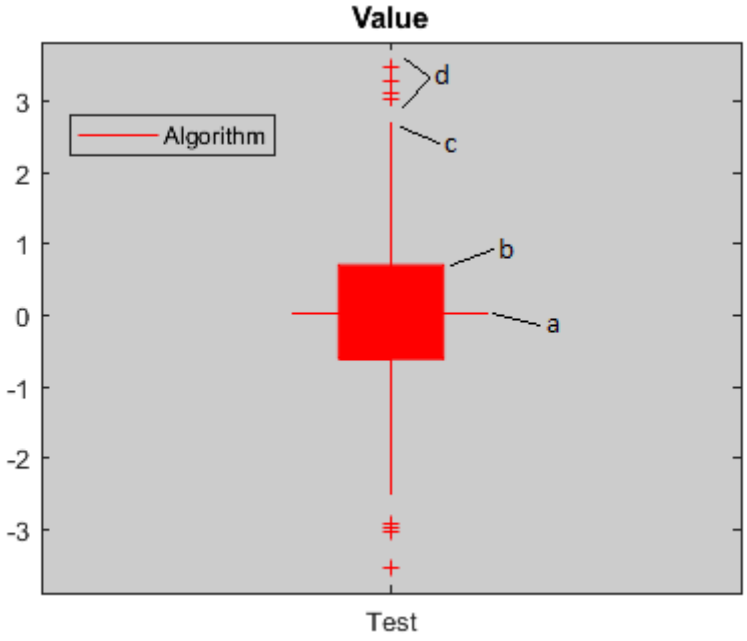


Figure 6.6: Figure depicts a box plot of a normal distribution $\mathcal{N} \sim (0, 1)$ where **a** is the median, **b** the top 75th percentile, **c** the top 99th percentile and **d** are outliers.

An illustration of a box plot can be seen in figure 6.6 to provide some aid in how to interpret the results. Some of the plots have some outliers visually removed as the scale of the figures would otherwise result in difficulty interpreting the plots.

6.2.2 Root Mean Square Error

Root Mean Square Error is a commonly used metric for deviations in estimates or predictions of values. It presents the mean of the deviation over a sequence of predicted values. The RMSE of a set of n predicted values $\hat{x}$, given the true vales x is stated as

$$RMSE(\hat{x}) = \sqrt{\frac{\sum_{k=1}^{n}(\hat{x}_k - x_k)^2}{n}}. \tag{6.1}$$

7

Results

This chapter presents the different results gathered based on the tests defined in Chapter 6. The results are presented in order, such that additional modules are added upon previously presented configurations and such that the effect of an individual module can be evaluated. Finally, the test case for roll and pitch is presented separately, as assumptions made in some modules are contradicted under the tests circumstances.

To begin with, the filtered and smoothed values from the gyroscope in the IMU and the wheel-speed sensor gathered during tests from Sections 6.1.1-6.1.4 is presented in Figure 7.1. These values were used as reference throughout all test cases.

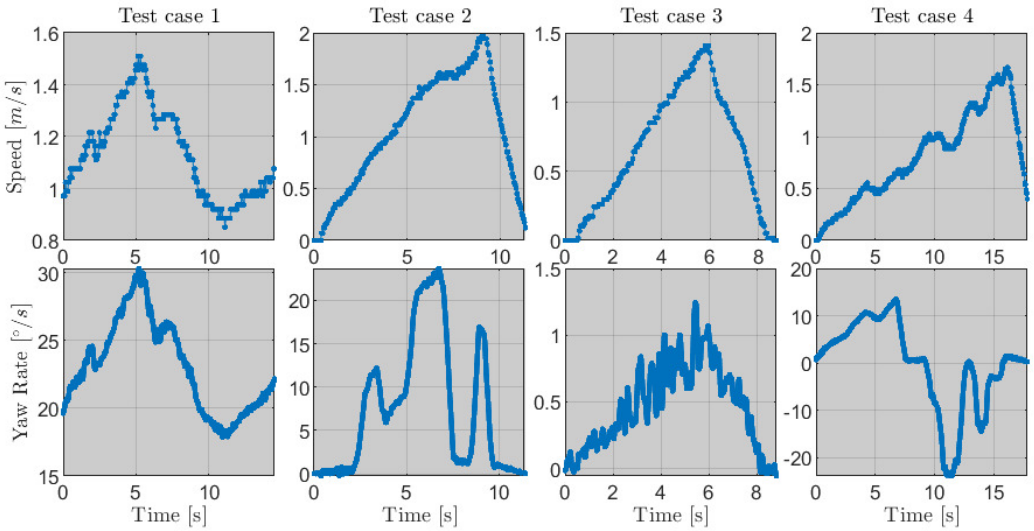


Figure 7.1: Wheel speed readings and smoothed gyroscope values for test cases 1-4.

7.1 Configuration 1 - No Additions

In the first configuration no matching rejection, cloud rectification or pre-processing was performed. The results for the different test cases are presented in Figure 7.2,

displaying the estimates of ICP, Mb-ICP and G-ICP.

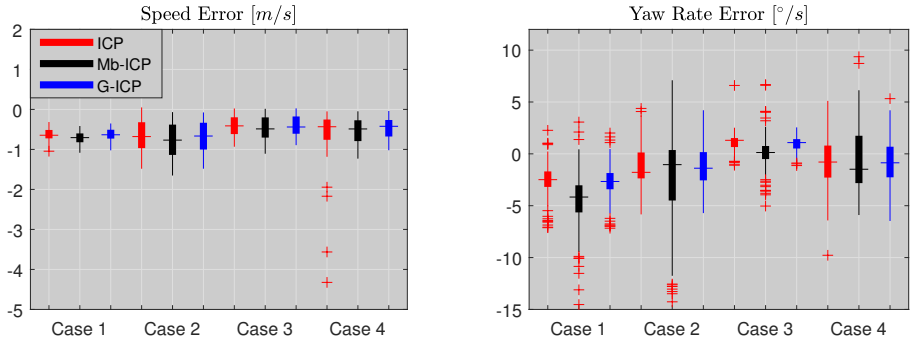


Figure 7.2: Box plots presenting the Speed and Yaw Rate error distributions for the different test cases and algorithms for Configuration 1. Occasional outliers for ICP excluded due to their magnitude.

Further, the RMSE for the different algorithms and test cases are presented in Table 7.1.

		Case 1	Case 2	Case 3	Case 4
Speed [m/s]	ICP	0.6642	0.4879	0.7917	0.6342
	Mb-ICP	0.7393	0.5636	0.9014	0.6332
	G-ICP	0.6644	0.4862	0.7896	0.5382
Yaw Rate [°/s]	ICP	3.0150	1.2815	2.5058	2.2662
	Mb-ICP	5.3260	2.1532	5.0345	3.2347
	G-ICP	3.1084	1.2108	2.4602	2.2531

Table 7.1: The RMSE for the parameters Speed and Yaw Rate for the different test cases and algorithms for Configuration 1.

From the results it can be seen that the three algorithms struggle, especially with yaw-rate estimation. ICP and G-ICP perform equally poor with Mb-ICP being the worst in general. Looking closer at Test Case 1 presented in Figure 7.3 there is a clear bias in both parameters, where speed estimates appear to be affected heavily.

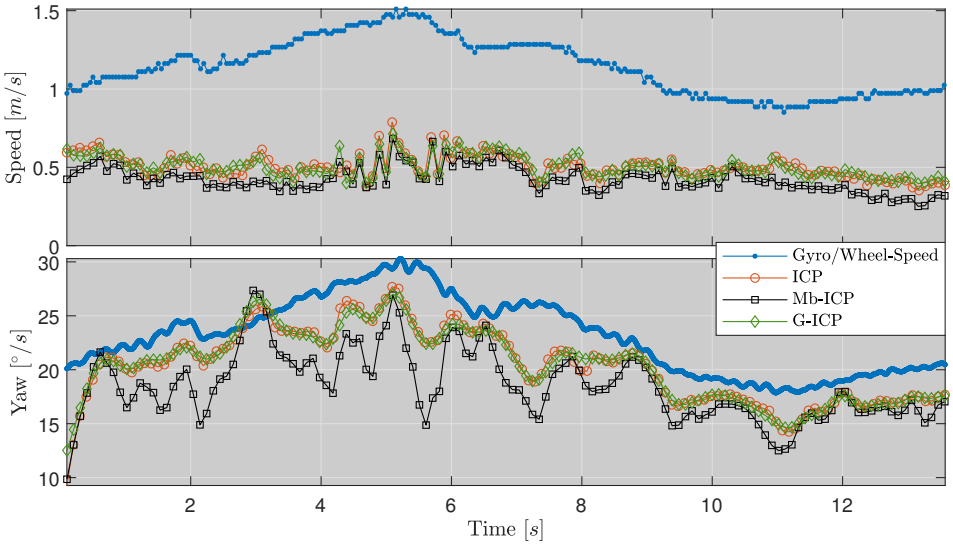


Figure 7.3: Estimates of speed and yaw rate compared with the ground truth for Configuration 1 - Test case 1.

7.2 Configuration 2 - Pre-Processing

For the second configuration, a pre-processing step is applied in attempt to process higher quality clouds in ICP. By applying the pre-processing step the results presented in Figure 7.4 were achieved.

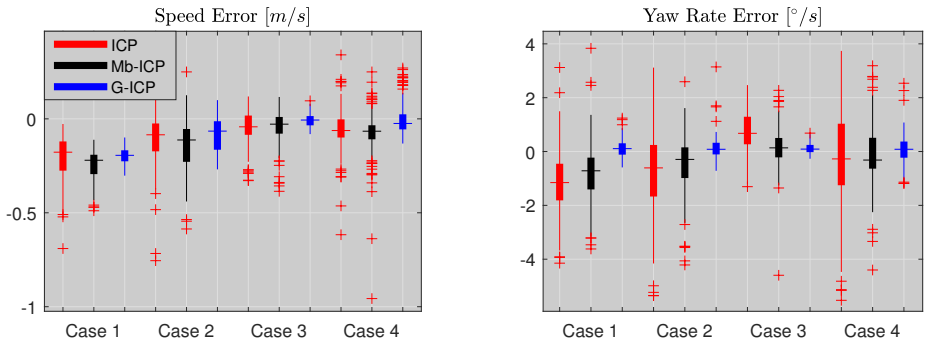


Figure 7.4: Box plots presenting the Speed and Yaw Rate error distribution for the different test cases and algorithms for Configuration 2.

Further, the RMSE for the different algorithms and test cases is presented in Table 7.2.

		Case 1	Case 2	Case 3	Case 4
Speed [m/s]	ICP	0.2470	0.1192	0.1977	0.1276
	Mb-ICP	0.2621	0.1229	0.1950	0.1474
	G-ICP	0.2048	0.0365	0.1311	0.0878
Yaw Rate [°/s]	ICP	1.6912	0.9846	1.8336	1.8343
	Mb-ICP	1.3950	0.9258	1.5585	1.1287
	G-ICP	0.3593	0.2253	0.5064	0.5699

Table 7.2: The RMSE for the parameters Speed and Yaw Rate for the different test cases and algorithms for Configuration 2.

The results clearly shows how the estimation errors were reduced for the three ICP implementations, especially considering speed estimates. Both yaw rate and speed estimates were best performed by G-ICP, with Mb-ICP and ICP being less accurate. Test case 1 appears to be more difficult than the other cases considering speed estimates.

7.3 Configuration 3 - Matching Rejection

To further enhance the quality of the data in the final steps of ICP, matching rejection is added in addition to the pre-processing step. This in order to neglect influence from unique points in the source and target clouds. In other words, remove points which origin from features which is not present in both clouds. The result of adding a matching rejection step is presented in Figure 7.5.

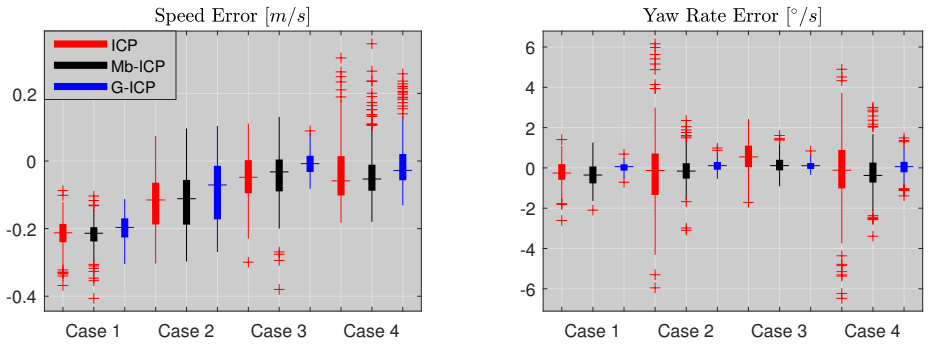


Figure 7.5: Box plots presenting the Speed and Yaw Rate error distribution for the different test cases and algorithms for Configuration 3.

Further, the RMSE for the different algorithms and test cases are presented in Table 7.3.

		Case 1	Case 2	Case 3	Case 4
Speed [m/s]	ICP	0.2232	0.0912	0.1551	0.1001
	Mb-ICP	0.2231	0.1068	0.1577	0.0964
	G-ICP	0.2056	0.0359	0.1335	0.0870
Yaw Rate [°/s]	ICP	0.6477	0.9096	2.2291	1.9679
	Mb-ICP	0.7287	0.5701	0.8969	1.1351
	G-ICP	0.2331	0.2384	0.3163	0.4636

Table 7.3: The RMSE for the parameters Speed and Yaw Rate for the different test cases and algorithms for Configuration 3.

The results does not show a clear general improvement from adding the matching rejection step. However, as seen both from the result from adding a pre-processing step and from adding a matching rejection step; G-ICP seem to outperform ICP and Mb-ICP under equal conditions in all test cases, and in estimation of both Speed and Yaw Rate. Looking closer on the results only from G-ICP, a slight but seemingly general improvement can be seen in adding the matching rejection step.

7.4 Configuration 4 - Cloud Rectification

The last module tested is the cloud rectification step, in the form of the V-ICP loop. This in order to compensate for ego-motion induced distortions in LiDAR sampled point clouds. In this configuration the matching rejection step is excluded in order to evaluate the effect of the two modules under equal conditions. The result of adding the cloud rectification step is presented in Figure 7.6.

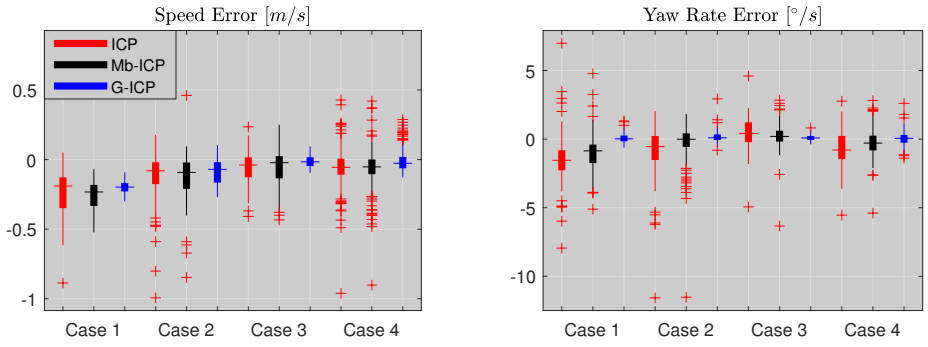


Figure 7.6: Box plots presenting the Speed and Yaw Rate error distribution for the different test cases and algorithms for Configuration 4.

Further, the RMSE for the different algorithms and test cases is presented in Table 7.4.

		Case 1	Case 2	Case 3	Case 4
Speed [m/s]	ICP	0.2806	0.1350	0.2411	0.1654
	Mb-ICP	0.2803	0.1461	0.2158	0.1667
	G-ICP	0.2061	0.0416	0.1329	0.0877
Yaw Rate [°/s]	ICP	2.3445	1.2143	2.0748	1.3917
	Mb-ICP	1.6963	1.2012	1.7226	1.0706
	G-ICP	0.3504	0.2226	0.4930	0.5277

Table 7.4: The RMSE for the parameters Speed and Yaw Rate for the different test cases and algorithms for Configuration 4.

Contrary to the result of adding the matching rejection step, adding the cloud rectification step resulted in a worse behaviour in general, with a few small exceptions for G-ICP.

7.5 Configuration 5 - Combining Modules

The final configuration tested is combining the matching rejection and cloud rectification step. The result of combining the two modules on top of the pre-processing step is presented in Figure 7.7.

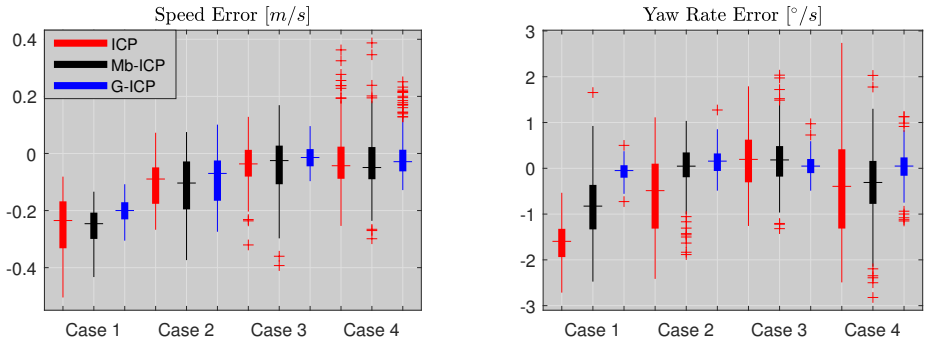


Figure 7.7: Box plots presenting the Speed and Yaw Rate error distribution for the different test cases and algorithms for Configuration 5.

Further, the RMSE for the different algorithms and test cases is presented in Table 7.5.

		Case 1	Case 2	Case 3	Case 4
Speed [m/s]	ICP	0.2816	0.0979	0.1312	0.1045
	Mb-ICP	0.2654	0.1231	0.1565	0.1076
	G-ICP	0.2077	0.0427	0.1349	0.0863
Yaw Rate [°/s]	ICP	1.6569	0.7346	1.0623	1.1581
	Mb-ICP	1.1212	0.7537	0.6116	0.9108
	G-ICP	0.2367	0.2445	0.3365	0.3794

Table 7.5: The RMSE for the parameters Speed and Yaw Rate for the different test cases and algorithms for Configuration 5.

The result of the two modules seem to differ between scenarios and parameters, and in comparison to the different configurations. However, one clear difference which can be observed for G-ICP is that the estimates seem to improve for Yaw Rate estimates in test case 4. Meanwhile, the effect seems to be larger for ICP and Mb-ICP, but resulting in both better and worse behaviour for different parameters and test cases. In summary, the effect of combining the two modules seems to have larger influence on ICP and Mb-ICP than on G-ICP, but with a varying effect for different scenarios and parameters.

7.6 Provoking Roll & Pitch

In previous test cases pitch and roll rates were in general close to zero but with some small oscillation. In these cases all algorithms estimated the roll and pitch rates to be zero, with small deviations. The last test case, with the attempt to evaluate how a provocation in pitch and roll is handled, is presented in Figure 7.8. In this test case the ground and ceiling filter was removed as it does not function properly when the sensor is tilting. However, the distance-based filter of the pre-processing step and the matching rejection step is included.

As seen in the figure, none of the algorithms handle the provocation very well and does not in general follow the trends of the ground-truth. Meanwhile, G-ICP seems to handle the deviations in pitch and roll rates better than ICP and Mb-ICP. Further, an interesting aspect of the result is that the yaw rate estimates seem to deviate from the ground-truth in an unusual manner during periods of larger roll rates. Four additional test cases for roll and pitch can be found in Appendix A.

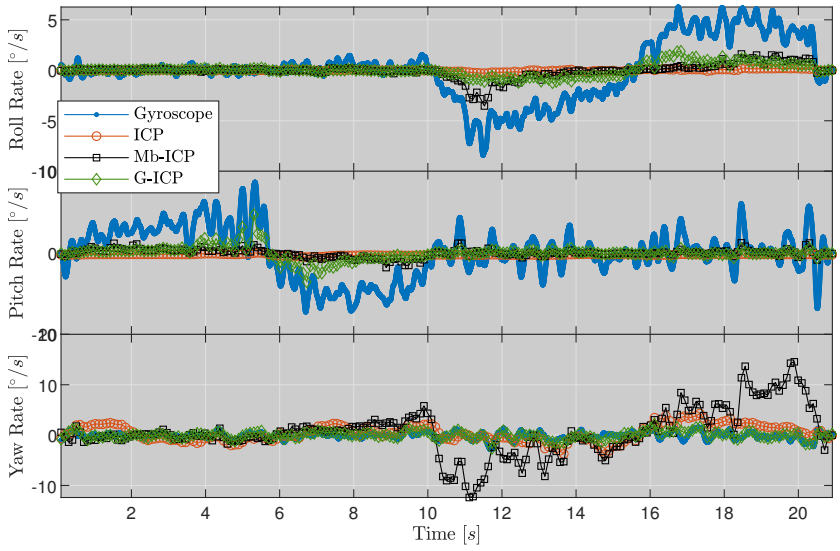


Figure 7.8: Roll and pitch rate estimates and ground truth for the Roll and Pitch test case.

8

Discussion

As stated in Section 1.2, the goal of the thesis is to "evaluate ego-motion estimates from using LiDAR generated point clouds". The results presented in Chapter 7 with all its components will be discussed in this chapter in such that conclusions regarding the goal can be drawn.

8.1 ICP and ICP Extensions

The Iterative Closest Point algorithm is in a sense naive, although effective. ICP's limitations become obvious as a core assumption for optimal performance is that there always exists true point correspondence. This might be true for some types of problems, but certainly not for LiDAR point clouds due to measurement noise. In Table 7.2 the algorithms are compared without any additions except filtering, and it can be seen that ICP with its minimization of point-to-point distance performs poorly. This is in line with the comparison performed in [37], which suggested other methods of minimization. Worth mentioning is that ICP has a computational cost that was significantly lower than its extensions, and still performs almost as well as Mb-ICP in many cases.

Metric-based ICP utilize an interesting concept of not only removing the assumption of true point correspondences, but also changing the metric for which correspondences are assigned by. The core reason for this is, according to the authors of Mb-ICP [10], to increase alignment performance during rotation. Therefore it was expected to heavily outperform at least ICP in test case 1 where rotation is applied throughout the whole scenario. As shown in the results there are some improvement compared to ICP in yaw rate, which appears to come with the cost of speed estimations.

The parameter L in Mb-ICP is the trade-off between rotation and translation [11]. The authors suggested to set $L = 3$, where it is possible that tuning of this parameter is necessary to achieve greater results.

Generalized ICP is a commonly used algorithm within the ICP family, and from the results one can clearly see why, as it performed best in almost every test case.

However, the G-ICP algorithm is by no means perfect and there exists many papers attempting to improve upon it. The approach of using structural information is interesting, although the implementation in G-ICP contains some flaws in that a uniform distribution of points is assumed. Due to how rotational LiDARs operate, the vertical resolution becomes very low in comparison to the horizontal resolution, i.e, points are not sampled uniformly on surfaces, but rather in horizontal lines as can be seen in Figure 8.1. This becomes an issue as the nearest neighborhood method of

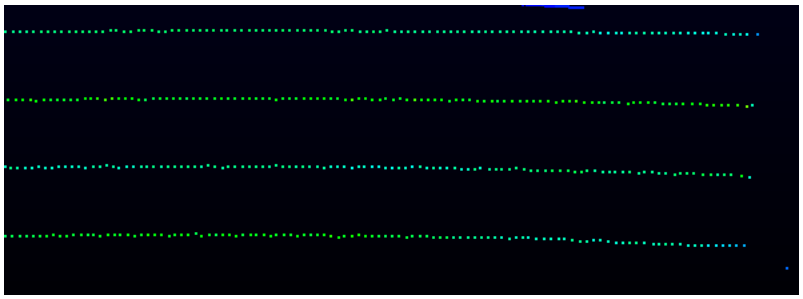


Figure 8.1: Depiction of points sampled on a flat surface, where it can be seen that the samples are in lines and not uniformly distributed.

selecting the k-nearest neighbors in G-ICP possibly only finds a set of points which origin from the same line. More specifically, the resulting surface normal computed from PCA might be sampled from a line rather than a surface. Thus, PCA will only capture the distribution of the measurement noise of the sampled line such that the estimated normals may differ heavily. The same issue applies to other similar methods such as surface normal rejection.

The simple workaround to bad neighborhood selection implemented, which defines neighbors as the points within some radius, seemingly works for the evaluated test cases. However, the radius has to be set to large enough to capture the measurements from the next channel of the sensor, but low enough to capture the unique structure in the area. As the resolution of point clouds gathered by this type of LiDARs deprecate over distance, an interesting area of further development is to evaluate new approaches to this issue, for example by implementing a dynamic radius. This would likely increase the performance as well as generalize the algorithm for multiple environments.

As a conclusion, G-ICP performed better than Mb-ICP and ICP for all parameters in all test cases and for all configurations. However, both extensions to ICP is in general more computationally heavy than ICP. Thus, the choice of algorithm naturally becomes a trade-off between computational cost and performance.

8.2 ICP Modules

The impact of pre-processing is obvious for featureless areas such as ground and ceiling which can be seen when comparing the results of configurations in section 7.1

and 7.2. By removing the repetitive measurements captured in the top and bottom of each frame the improvements were very clear. This is a problem that can occur for many scenarios, such as going through a tunnel where the walls might be identical, as well as driving in a round-about where the center can appear identical for each frame to distort the results. An example of this can be seen with the central pillar in Test Case 1 where the algorithms speed estimations are less accurate in general. This observation concurs with the ones presented in [45, 37] where pose estimations in repetitive or featureless environments was troublesome.

The approach for ground and ceiling removal by simply removing points above and below a set z-coordinate threshold is a naive approach and only works well under good circumstances. Due to the evidently large impact of the pre-processing step it would be interesting to evaluate more qualified approaches to ground and ceiling removal such as the gradient threshold based method in [46]. In addition, it would be interesting to analyze other methods and ideas for removal of featureless areas.

Another solution to removing the false features which occurs on the ground and ceiling is to use a sensor which does not suffer from the low vertical resolution of rotational LiDARs. An example is the LiDAR sensor used in [47], which rotates in multiple directions and therefor does not generate the same differentiable false features on the ground and ceiling.

Matching rejection generally had a positive impact on the results, as can be seen comparing the results in Sections 7.2 and 7.3. This is in line with the results from [37]. Which of the two implemented methods that had the largest impact were not investigated. There are other matching rejection methods which could potentially affect the results even more. The distance based rejection method from Section 3.5.1 has a tuneable parameter in the maximum acceptable distance between corresponding points. There exists some methods with a similar approach, although the maximum distance is based on the distribution of the points. It is likely to perform better for different scenarios where the range distributions varies, without having to be reconfigured e.g. travelling on a highway soon to enter a city.

The concept of cloud rectification seems intuitive at higher speeds. Although as can be seen when comparing Sections 7.2 and 7.4 there is no clear improvement. By observing the results of the configurations performance solely on test case 4, as can be seen in Appendix B, there appears to be some added oscillations for ICP and Mb-ICP. Tuning of the convergence threshold might be necessary for more accurate estimates. The speeds which the test cases were run with where relatively low, with the top speed measured to approximately $2m/s$, which translates to a maximum translational distortion of the point cloud by 20 cm. This would likely not distort the point clouds severely enough such that rectification is necessary, therefore it is hard to draw any reasonable conclusion regarding the V-ICP addition. Though as previously mentioned there are several authors who proves attest to its importance [18, 30, 15, 47].

Combining both cloud rectification and matching rejection had some expected con-

sequences. The result in Section 7.5 appears to be something in-between the result of applying the modules individually as in Sections 7.4 and 7.3. Analyzing whether or not adding the rectification module on top of the matching rejection module is an improvement or not in general would require further testing under higher speeds. However, it seems that the rectification process has a tendency to introduce some oscillations and lower the general performance during lower speeds.

In conclusion, the pre-processing module seems to not only be the most important module out of the evaluated modules, but its ground and ceiling removal step seems to be vital for high performance in speed and yaw rate estimates. The matching rejection module had a positive effect on the estimates, but the magnitude of its effect results in the question of whether or not the added computational complexity is worth the trade-off. Lastly, the cloud rectification module has a questionable effect on performance in the test cases presented, but could possibly have larger importance when sampling under higher speed.

8.3 Roll & Pitch Rates

Roll and pitch rates estimates presented in Section 7.6 can be seen to perform poorly. This was expected due to the low vertical resolution of the sensor which can not properly capture the transformation between the two frames. A possible reason could be due to that a core assumption of the ICP algorithms is that there is a proper rigid transformation which aligns the source and target point cloud. Imagining a simple case of a single channel rotational LiDAR in a square room, where the source cloud is captured horizontally and the target cloud when the sensor is tilted 30°. Naturally the source cloud will represent a square while the target cloud will have a more rectangular shape in the coordinate frame of the LiDAR, such that a stretch and/or shear transformation is required to properly align the two clouds. Another observation that can be made from Figure 7.8 and the figures in Appendix A is that yaw rate can be affected. This will likely cause faulty estimations when roll and pitch rates are applied, i.e, driving over some steep curvature such as a speed bump or a hill.

As previously discussed, the ground and ceiling removal algorithm used in the pre-processing step simply removes all points below or above a set threshold in z-direction. This however, is performed under the coordinate-frame of the LiDAR sensor itself. Thus, if the sensor is tilting in pitch or roll in reference to the ground or ceiling some faulty points will be removed in the direction in which it is currently tilting. In other words, the algorithm does not work properly when a pitch or roll is applied. At the same time, as previously discussed the ground and ceiling removal algorithm seems to be important for speed and yaw rate estimates, and it would be interesting to investigate how a properly functioning algorithm would affect the estimates for pitch and roll rates as well.

Potentially, a LiDAR sensor with a higher vertical resolution and/or field of view

would solve the problems which arose in the presented results. There exists many other models and types of LiDAR sensors with higher vertical resolution than the 2 degrees for the Velodyne VLP 16 or its 30° field of view. It would be interesting to investigate what vertical resolution and/or field of view is necessary in order to yield high quality roll and pitch estimates.

Apart from traditional rotational LiDARs, there exists LiDAR sensors such as discussed for solving issues of false features in the ground and ceiling in Section 8.2. Possibly, both the issues with estimating roll and pitch rates, and the false features in ground and ceiling would be solved by simply utilizing a different type of LiDAR sensor.

Meanwhile, a better solution could for example be to extract the ground plane of the point clouds and estimate the angles to such a plane instead. In addition, such an approach would yield an absolute pitch and roll estimate as well, which could possibly be used to handle yaw rate and speed estimates better.

8.4 Estimating Ego-Motion with LiDAR Data and ICP

As a summary, the G-ICP extension combined with a pre-processing and matching rejection module showed the best general performance out of the different configurations. Adding on the cloud rectification module did not show a general improvement but might be more important when sampling at higher speed. While the resulting algorithm showed promising performance for estimating speed and yaw rate, the tests for pitch and roll deemed the algorithms unfeasible for estimating pitch and roll rates under the circumstances.

Setting the threshold of convergence even lower proved to increase the yaw-rate accuracy of ICP and Mb-ICP even further, which was expected. G-ICP did not have much room for improvement as convergence was quick. Lowering the threshold can increase the general performance of the algorithms to some extent, though it does not solve the problems stated.

The area of active safety and autonomous drive spans to many different types of vehicles, all of which have partly unique behaviour on the road. An example is trucks, which in comparison to, e.g, cars tend to tilt their cabin to a much more significant degree while turning or braking. Thus, the pitch and roll rates might be a more interesting parameter to track in a truck than in a car. Going back to answer on the problem stated in the purpose of the thesis, ego-motion in terms of speed and yaw-rate can be derived from LiDAR data, while pitch and roll rates seem to be more troublesome with the proposed algorithms. In other words, depending on which subset of parameters of ego-motion are interesting for the use-case, the approach presented in this thesis could be useful.

An interesting topic of further analysis is the neighborhood method of G-ICP, which could make the algorithm both more accurate and robust. Moreover, the difference in accuracy between speed, yaw rate and pitch rate, roll rate is an interesting result, though partly expected due to the structure of the data gathered from a rotational LiDAR. However, even though the tests performed show an underwhelming performance for roll and pitch, the results leave room for further analysis as there are many other approaches which could handle such estimations better. Lastly, it seems that some issues for estimating all parameters origin from how rotational LiDARs sample clouds, and it would be interesting to investigate which problems persist if using a different type of LiDAR sensor.

9

Conclusion

The proposed algorithm for estimating ego-motion from LiDAR data show some conclusive results under the defined test cases. Under the circumstances of the test cases and with the proposed algorithm, it is feasible to estimate speed and yaw rate of the ego-vehicle from LiDAR data. Meanwhile, in order to estimate pitch and roll rates, further analysis of other modules, completely different algorithms, or other types of LiDAR sensors is required. Further, a ground and ceiling removal method is shown to be a vital part of the ICP based algorithms in order to estimate both speed and yaw rate without any significant bias, at least when using data from a rotational LiDAR in an enclosed space.

Depending on which parameters are interesting for a specific application, the algorithm could be useful in its presented state. As previously discussed, the algorithm might not be suitable in applications where significant pitch or roll occur, such as in trucks. Meanwhile, it could be useful for estimating speed and yaw rate in for example a car.

As a conclusion, the results of the performed tests indicates feasibility for using LiDAR generated point clouds for ego-motion estimations. However, further analysis into handling false features and estimating roll and pitch rates has to be done for robust, and accurate estimations.

Bibliography

[1] L. Nissen, "Velodyne announces order from ford motor company for its next-gen solid-state lidar sensor designed for adas safety and autonomous driving," 2016, accessed on: February 26, 2019. [Online]. Available: https://velodynelidar.com/docs/news/Velodyne%20Announces%20Order%20From%20Ford%20Motor%20Company%20for%20its%20Next-Gen%20Solid-State%20LiDAR%20Sensor%20Designed%20for%20ADAS%20Safety%20and%20Autonomous%20Driving.pdf

[2] T. B. Lee, "Why experts believe cheaper, better lidar is right around the corner," 2018, accessed on: February 25, 2019. [Online]. Available: https://arstechnica.com/cars/2018/01/driving-around-without-a-driver-lidar-technology-explained/

[3] P. J. 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, Feb. 1992.

[4] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Found. Trends Robot*, vol. 4, no. 1, pp. 1–104, May 2015. [Online]. Available: http://dx.doi.org/10.1561/2300000035

[5] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, Apr. 1991, pp. 2724–2729 vol.3.

[6] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, May 2001, pp. 145–152.

[7] A. Segal, D. Hähnel, and S. Thrun, "Generalized-icp," in *Robotics: Science and Systems V, University of Washington, Seattle, USA, June 28 - July 1, 2009*, 2009. [Online]. Available: http://www.roboticsproceedings.org/rss05/p21.html

[8] L. Montesano, J. Minguez, and L. Montano, "Probabilistic scan matching for motion estimation in unstructured environments," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug. 2005, pp. 3499–3504.

[9] F. Lu and Milios, "Robot pose estimation in unknown environments by matching 2d range scans," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 1994, pp. 935–938.

[10] J. Minguez, L. Montesano, and F. Lamiraux, "Metric-based iterative closest point scan matching for sensor displacement estimation," *Robotics, IEEE Transactions on*, vol. 22, pp. 1047 – 1054, Nov. 2006.

[11] L. Armesto, J. Minguez, and L. Montesano, "A generalization of the metric-based iterative closest point technique for 3d scan matching," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 1367–1372.

[12] H. Yoshitaka, K. Hirohiko, O. Akihisa, and Y. Shin'ichi, "Mobile robot localization and mapping by scan matching using laser reflection intensity of the sokuiki sensor," in *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*, Nov. 2006, pp. 3018–3023.

[13] P. Biber and W. Strasser, "The normal distributions transform: a new approach to laser scan matching," *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, Oct. 2003.

[14] J. P. Saarinen, H. Andreasson, T. Stoyanov, and A. Lilienthal, "3d normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments," *The International Journal of Robotics Research*, vol. 32, pp. 1627–1644, Dec. 2013.

[15] L. Richard, "Estimation of ego vehicle motion from lidar point cloud data," Master's thesis, Chalmers University of Technology, 2015.

[16] A. Geiger, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ser. CVPR '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 3354–3361. [Online]. Available: http://dl.acm.org/citation.cfm?id=2354409.2354978

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

[18] F. Moosmann and C. Stiller, "Velodyne slam," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2011, pp. 393–398.

[19] "Clean motion zbee product specification," Lerum, Sweden: Clean Motion, n.d, accessed on: February 4, 2019. [Online]. Available: https://cleanmotion.se/zbee/specification/

[20] "Velodyne vlp16 product page," San Jose, U.S.A: Velodyne Lidar, n.d,

accessed on: February 5, 2019. [Online]. Available: https://velodynelidar.com/vlp-16.html

[21] "Honeywell aerospace hg1120 product specification," Honeywell International Inc., n.d, accessed on: February 4, 2019. [Online]. Available: https://aerospace.honeywell.com/en/products/navigation-and-sensors/hg1120

[22] Electronics Tutorials, "Hall effect sensor," n.d, accessed March 22, 2019. [Online]. Available: https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html

[23] S. Maneewongvatana and D. M. Mount, "Analysis of approximate nearest neighbor searching with clustered point sets," *CoRR*, vol. cs.CG/9901013, 1999. [Online]. Available: http://arxiv.org/abs/cs.CG/9901013

[24] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975. [Online]. Available: http://doi.acm.org/10.1145/361002.361007

[25] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, Apr. 1987. [Online]. Available: http://josaa.osa.org/abstract.cfm?URI=josaa-4-4-629

[26] D. Eggert, A. Lorusso, and R. Fisher, "Estimating 3-d rigid body transformations: a comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, no. 5, pp. 272–290, Mar. 1997. [Online]. Available: https://doi.org/10.1007/s001380050048

[27] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987.

[28] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Autonomous Robots*, Apr. 2013.

[29] J. Lv, K. Yukinori, A. A. Ravankar, A. Ravankar, and T. Emaru, "A solution to estimate robot motion with large rotation by matching laser scans," in *2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Jul. 2015, pp. 1083–1088.

[30] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "Lidar point clouds correction acquired from a moving car based on can-bus data," *CoRR*, vol. abs/1706.05886, 2017. [Online]. Available: http://arxiv.org/abs/1706.05886

[31] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low drift, robust, and fast," in *IEEE International Conference on Robotics and Automation(ICRA)*, Seattle, WA, May 2015.

[32] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," *KI - Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, Nov. 2010. [Online]. Available: https://doi.org/10.1007/s13218-010-0059-6

[33] M. Magnusson, A. Nuchter, C. Lorken, A. J. Lilienthal, and J. Hertzberg, "Evaluation of 3d registration reliability and speed - a comparison of icp and ndt," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 3907–3912.

[34] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, accessed on: March 5, 2019. [Online]. Available: http://www.scipy.org/

[35] E. Polak and G. Ribiere, "Note sur la convergence de méthodes de directions conjuguées," *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, vol. 3, no. R1, pp. 35–43, 1969. [Online]. Available: http://www.numdam.org/article/M2AN_1969___3_1_35_0.pdf

[36] Z. Zhang, "Iterative point matching of free-form curves and surfaces," *International Journal of Computer Vision - IJCV*, vol. 13, Oct. 1994.

[37] F. Donoso, K. Austin, and P. McAree, "How do icp variants perform when used for scan matching terrain point clouds?" *Robotics and Autonomous Systems*, vol. 87, pp. 147 – 161, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889016301282

[38] S. Hong, H. Ko, and J. Kim, "Vicp: Velocity updating iterative closest point algorithm," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 1893–1898.

[39] L. Shieh, H. Wang, and R. Yates, "Discrete-continuous model conversion," *Applied Mathematical Modelling*, vol. 4, no. 6, pp. 449 – 455, 1980. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0307904X80901778

[40] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. New York: Academic Press, 1970.

[41] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[42] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering (ASME)*, vol. 82D, pp. 35–45, Jan. 1960.

[43] H. E. Rauch, C. T. Striebel, and F. Tung, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, 1965. [Online]. Available: https://doi.org/10.2514/3.3166

[44] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation*. Wiley-Interscience, 2001.

[45] G. Bolmvall and M. Östman, "Precision localization in dense point cloud maps," Master's thesis, Chalmers University of Technology, 2018.

[46] H. Vu, H. T. Nguyen, P. M. Chu, W. Zhang, S. Cho, Y. W. Park, and K. Cho, "Adaptive ground segmentation method for real-time mobile robot control," *International Journal of Advanced Robotic Systems*, vol. 14, no. 6, 2017.

[47] J. Zhang and S. Singh, "Laser-visual-inertial odometry and mapping with high robustness and low drift," *Journal of Field Robotics*, Aug. 2018.

A

Roll & Pitch Tests

Further tests of roll and pitch rates which was not presented in the results chapter are presented here.

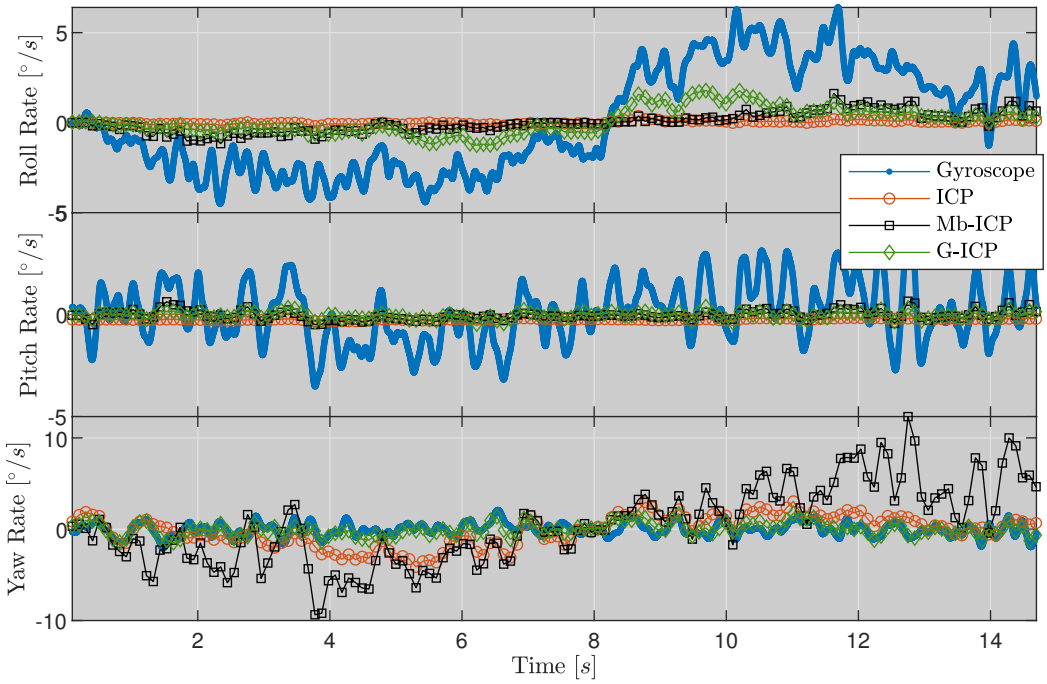


Figure A.1: Roll and pitch rate estimates and ground truth for a test case where the platform was rolled -45° then return to horizontal.

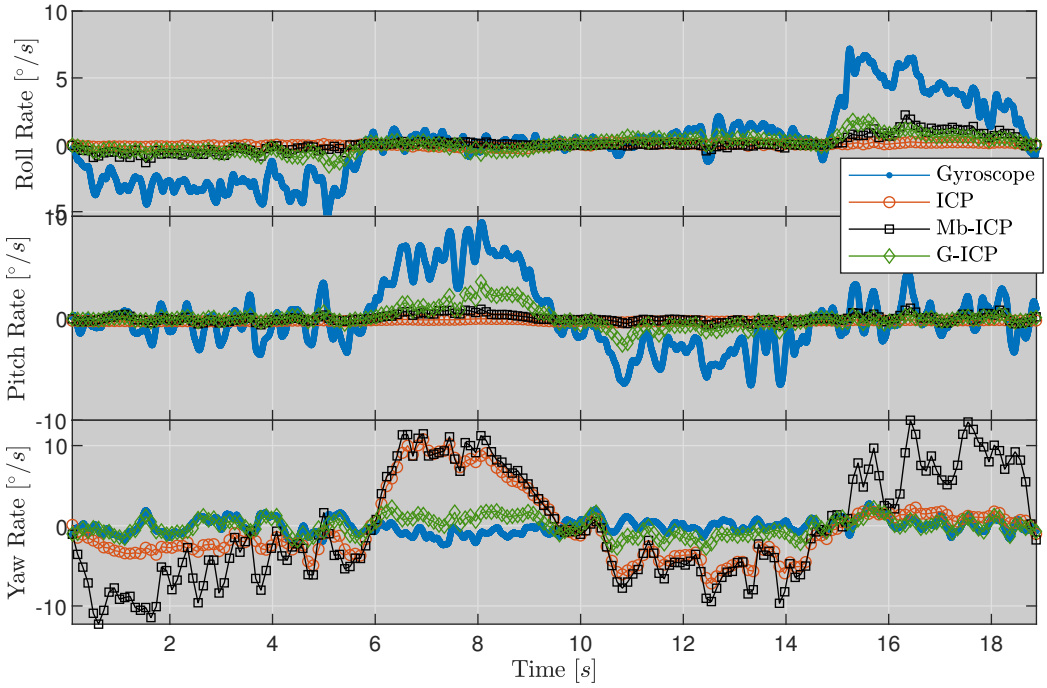


Figure A.2: Roll and pitch rate estimates and ground truth for a test case where the platform was rolled -45°, pitched 45°, pitched back and then return to horizontal.

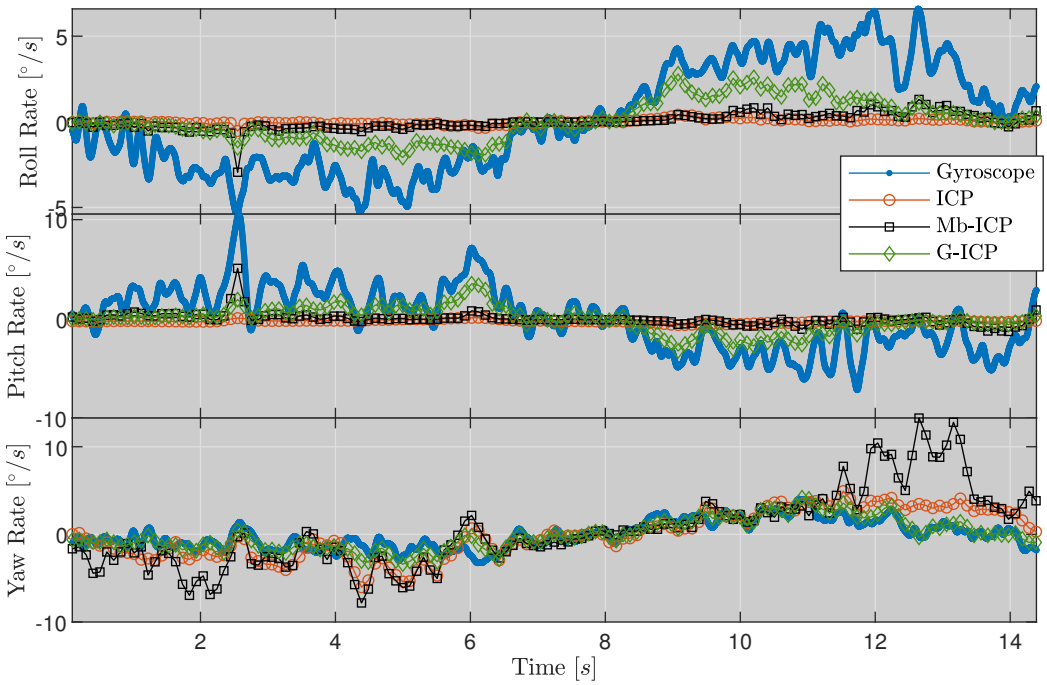


Figure A.3: Roll and pitch rate estimates and ground truth for a test case where the platform was rolled -45° and pitched 45° simultaneously and then return to horizontal.

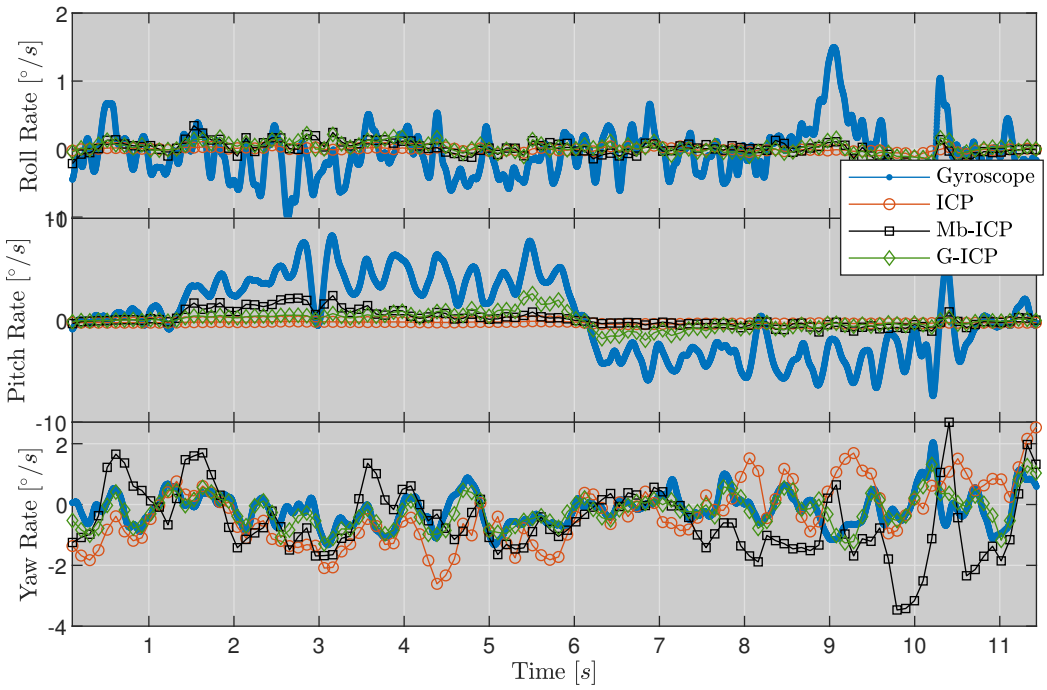


Figure A.4: Roll and pitch rate estimates and ground truth for a test case where the platform was pitched 45° then return to horizontal.

Noticeable in the figures is that almost nothing is captured in the roll and pitch rates, except for with G-ICP where small hints which follow the general trends can be seen. It appears that the roll and pitch rates influences the yaw-rate quite much.

B

Test Case 4

The resulting graphs for all configurations for test case 4 is presented below.

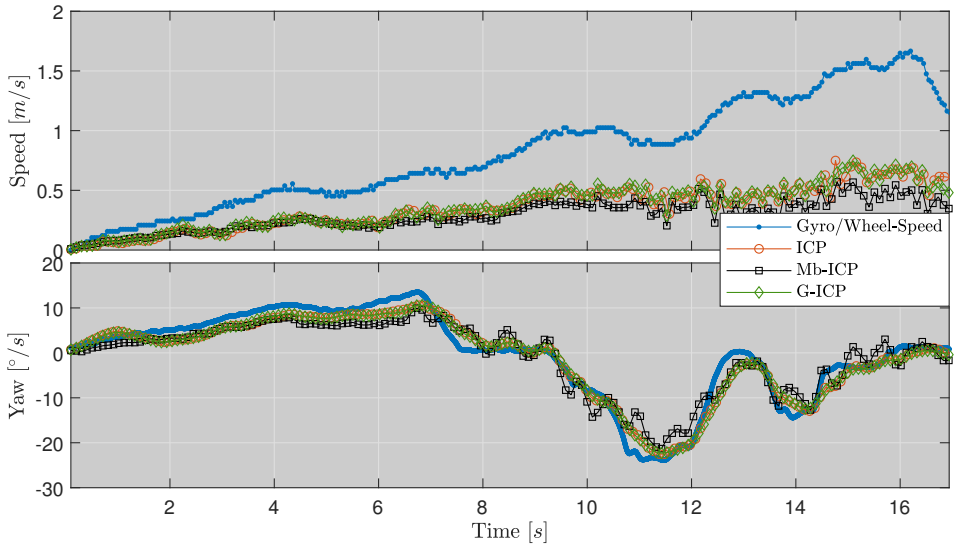


Figure B.1: Filtered speed and yaw rate estimations for test case 4 for the algorithms ICP, Mb-ICP and G-ICP without any additions.

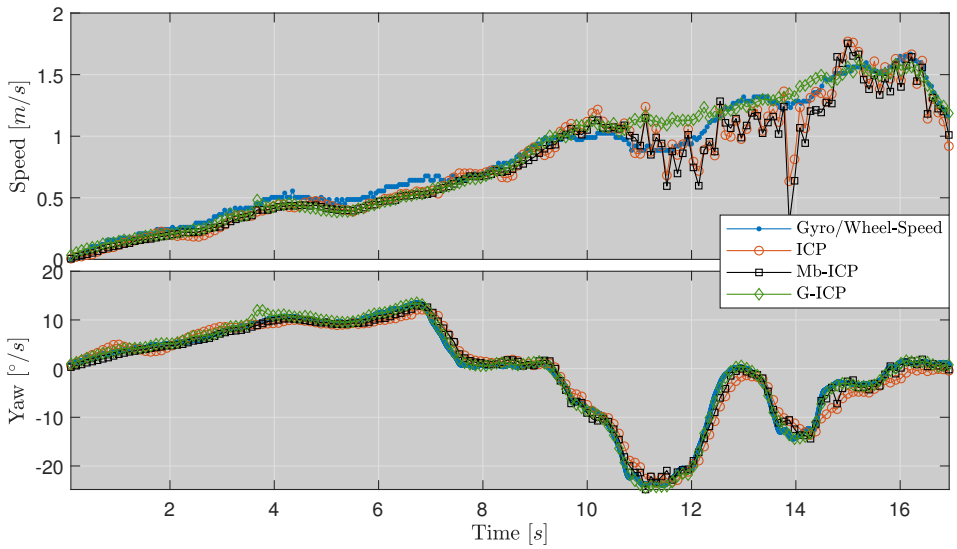


Figure B.2: Filtered speed and yaw rate estimations for test case 4 for the algorithms ICP, Mb-ICP and G-ICP where false features such as ground and ceiling are removed.

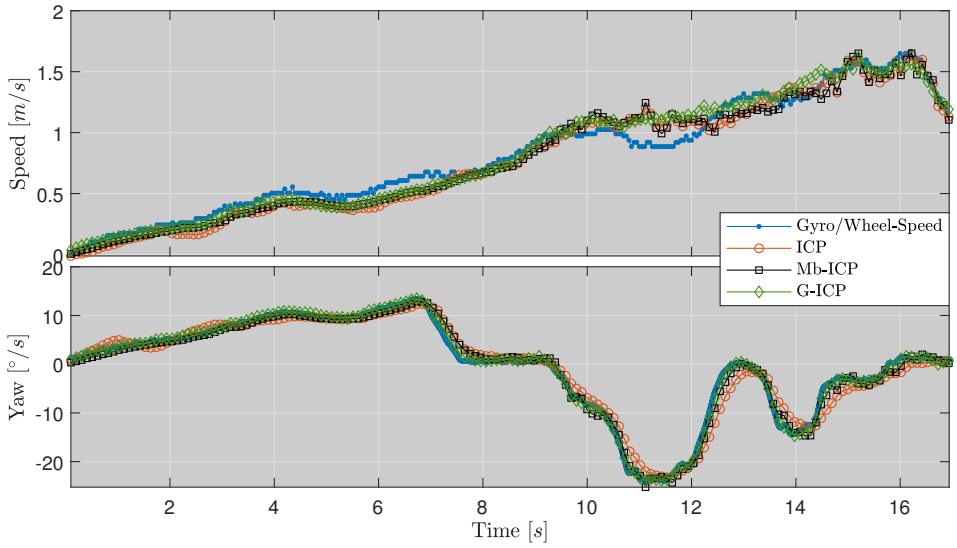


Figure B.3: Filtered speed and yaw rate estimations for test case 4 for the algorithms ICP, Mb-ICP and G-ICP where false features are removed and outliers are rejected in the correspondence search.

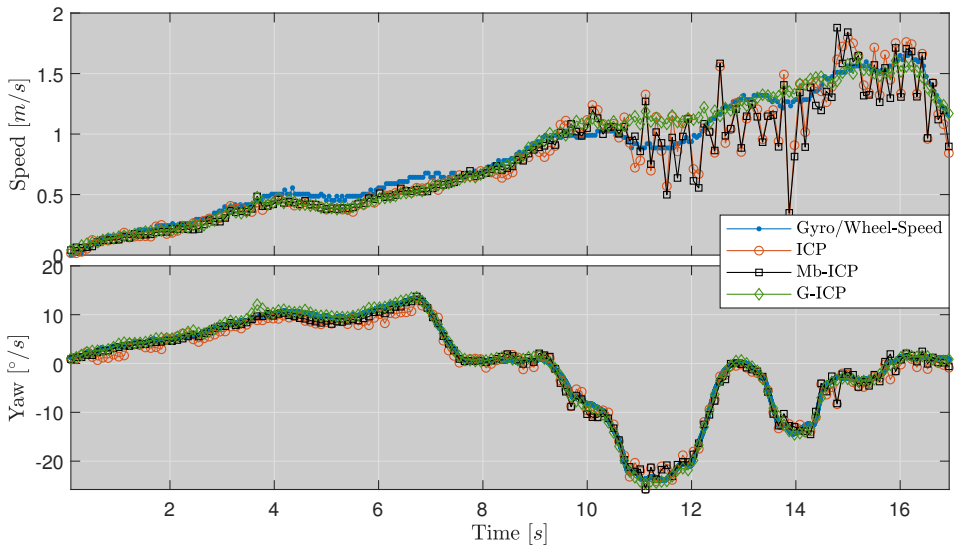


Figure B.4: Filtered speed and yaw rate estimations for test case 4 for the algorithms ICP, Mb-ICP and G-ICP where false features are removed and the point clouds are rectified.

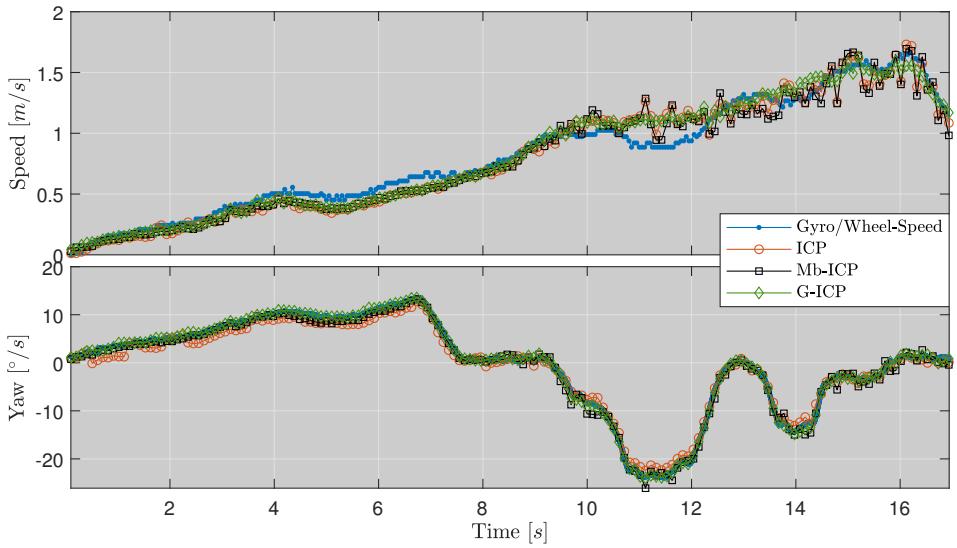


Figure B.5: Filtered speed and yaw rate estimations for test case 4 for the algorithms ICP, Mb-ICP and G-ICP where false features are removed, outliers are rejected in the correspondence search and the point clouds are rectified.

C

Measurement Noise Characteristic

The measurement noise covariance matrices for the different algorithms and configurations used in the filtering process is presented below. These were sampled with a stationary LiDAR in an enclosed room such that $\mathbb{E}(y) = [0, 0, 0, 0]^\top$.

$$R_{IMU} = \begin{bmatrix} 57.4563 & 1.1818 & 1.1156 \\ 1.1818 & 75.0010 & -2.6799 \\ 1.1156 & -2.6799 & 99.8949 \end{bmatrix} 10^{-4} \tag{C.1}$$

$$R_{ICP-Conf.-1} = \begin{bmatrix} 17.5817 & 172.3560 & -1.0683 & 0.9663 \\ 172.3560 & 1689.6285 & -10.4734 & 9.4748 \\ -1.0683 & -10.4734 & 16.4318 & 18.4323 \\ 0.9663 & 9.4748 & 18.4323 & 65.6044 \end{bmatrix} 10^{-5} \tag{C.2}$$

$$R_{ICP-Conf.-2} = \begin{bmatrix} 14.1032 & 138.2638 & 0.3573 & 0.7296 \\ 138.2638 & 1355.5034 & 3.4910 & 7.1519 \\ 0.3573 & 3.4910 & 8.2432 & 0.3109 \\ 0.7296 & 7.1519 & 0.3109 & 2.7740 \end{bmatrix} 10^{-5} \tag{C.3}$$

$$R_{ICP-Conf.-3} = \begin{bmatrix} 32.9479 & 1245.4638 & 18.2649 & 2.9004 \\ 1245.4638 & 67603.1733 & 1143.4261 & 162.9006 \\ 18.2649 & 1143.4261 & 22.4236 & 2.5445 \\ 2.9004 & 162.9006 & 2.5445 & 3.9503 \end{bmatrix} 10^{-5} \tag{C.4}$$

$$R_{ICP-Conf.-4} = \begin{bmatrix} 9.4134 & 92.2853 & -0.2162 & -0.0113 \\ 92.2853 & 904.7310 & -2.1206 & -0.1070 \\ -0.2162 & -2.1206 & 8.0170 & -0.0027 \\ -0.0113 & -0.1070 & -0.0027 & 1.9728 \end{bmatrix} 10^{-5} \tag{C.5}$$

$$R_{ICP-Conf.-5} = \begin{bmatrix} 9.4798 & 92.9381 & -0.1301 & 0.1169 \\ 92.9381 & 911.1495 & -1.2759 & 1.1496 \\ -0.1301 & -1.2759 & 6.8466 & 0.2338 \\ 0.1169 & 1.1496 & 0.2338 & 1.8128 \end{bmatrix} 10^{-5} \tag{C.6}$$

(C.7)

$$R_{Mb-ICP-Conf.-1} = \begin{bmatrix} 2.9965 & 29.3746 & -1.4526 & -3.5028 \\ 29.3746 & 287.9538 & -14.2397 & -34.3364 \\ -1.4526 & -14.2397 & 6.7355 & 4.6584 \\ -3.5028 & -34.3364 & 4.6584 & 17.2495 \end{bmatrix} 10^{-5} \tag{C.8}$$

$$R_{Mb-ICP-Conf.-2} = \begin{bmatrix} 14.1032 & 138.2638 & 0.3573 & 0.7296 \\ 138.2638 & 1355.5034 & 3.4910 & 7.1519 \\ 0.3573 & 3.4910 & 8.2432 & 0.3109 \\ 0.7296 & 7.1519 & 0.3109 & 2.7740 \end{bmatrix} 10^{-5} \tag{C.9}$$

$$R_{Mb-ICP-Conf.-3} = \begin{bmatrix} 14.3558 & 140.7386 & 0.5157 & 0.7857 \\ 140.7386 & 1379.7490 & 5.0499 & 7.7001 \\ 0.5157 & 5.0499 & 6.9395 & 0.4968 \\ 0.7857 & 7.7001 & 0.4968 & 2.5948 \end{bmatrix} 10^{-5} \tag{C.10}$$

$$R_{Mb-ICP-Conf.-4} = \begin{bmatrix} 13.8362 & 135.6179 & -0.6991 & 0.0040 \\ 135.6179 & 1329.2855 & -6.8515 & 0.0386 \\ -0.6991 & -6.8515 & 23.6742 & 0.2208 \\ 0.0040 & 0.0386 & 0.2208 & 6.1537 \end{bmatrix} 10^{-5} \tag{C.11}$$

$$R_{Mb-ICP-Conf.-5} = \begin{bmatrix} 0.0420 & 0.4116 & 0.1196 & -0.0090 \\ 0.4116 & 4.0350 & 1.1725 & -0.0881 \\ 0.1196 & 1.1725 & 22.7582 & -0.0851 \\ -0.0090 & -0.0881 & -0.0851 & 5.2612 \end{bmatrix} 10^{-5} \tag{C.12}$$

(C.13)

$$R_{G-ICP-Conf.-1} = \begin{bmatrix} 0.3676 & 3.6033 & -0.1264 & -1.3635 \\ 3.6033 & 35.3239 & -1.2392 & -13.3663 \\ -0.1264 & -1.2392 & 326.3219 & 152.6730 \\ -1.3635 & -13.3663 & 152.6730 & 789.6554 \end{bmatrix} 10^{-5} \tag{C.14}$$

$$R_{G-ICP-Conf.-2} = \begin{bmatrix} 0.3639 & 3.5676 & 0.3416 & -1.5491 \\ 3.5676 & 34.9785 & 3.3344 & -15.1955 \\ 0.3416 & 3.3344 & 167.1495 & 15.0715 \\ -1.5491 & -15.1955 & 15.0715 & 256.0464 \end{bmatrix} 10^{-5} \tag{C.15}$$

$$R_{G-ICP-Conf.-3} = \begin{bmatrix} 0.3672 & 3.6005 & 0.2433 & -1.5750 \\ 3.6005 & 35.3015 & 2.3725 & -15.4485 \\ 0.2433 & 2.3725 & 169.9439 & 12.1531 \\ -1.5750 & -15.4485 & 12.1531 & 255.5510 \end{bmatrix} 10^{-5} \tag{C.16}$$

$$R_{G-ICP-Conf.-4} = \begin{bmatrix} 0.3526 & 3.4572 & 0.3107 & -0.5417 \\ 3.4572 & 33.8955 & 3.0303 & -5.2982 \\ 0.3107 & 3.0303 & 140.6032 & 10.9047 \\ -0.5417 & -5.2982 & 10.9047 & 263.6631 \end{bmatrix} 10^{-5} \tag{C.17}$$

$$R_{G-ICP-Conf.-5} = \begin{bmatrix} 0.3549 & 3.4801 & 0.3313 & -0.5332 \\ 3.4801 & 34.1215 & 3.2350 & -5.2134 \\ 0.3313 & 3.2350 & 144.7536 & 8.2088 \\ -0.5332 & -5.2134 & 8.2088 & 263.8649 \end{bmatrix} 10^{-5} \tag{C.18}$$