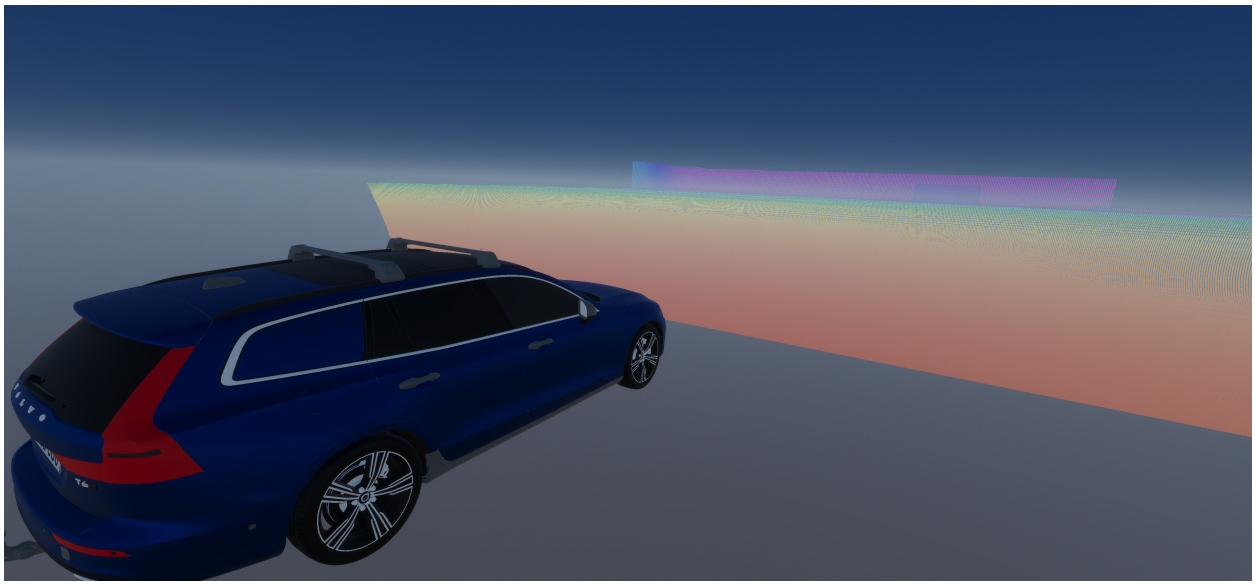


Extrinsic Lidar Calibration in a Workshop Environment

Evaluation of Concepts for Extrinsically Calibrating a Lidar sensor in an Automotive Workshop environment with respect to the Vehicle Coordinate System

Master's thesis in Systems Control and Mechatronics and Complex Adaptive Systems

MARCUS BERG
LUDVIG ERIKSSON

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Extrinsic Lidar Calibration in a Workshop Environment

Evaluation of Concepts for Extrinsically Calibrating a Lidar Sensor in an Automotive Workshop Environment with Respect to the Vehicle Coordinate System

MARCUS BERG
LUDVIG ERIKSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2023

Extrinsic Lidar Calibration in a Workshop Environment
Evaluation of Concepts for Extrinsically Calibrating a Lidar sensor in an Automotive Workshop Environment with Respect to the Vehicle Coordinate System
MARCUS BERG
LUDVIG ERIKSSON

Supervisors: Elías Marel, Daniel Åhlund, Volvo Car Corporation
Examiner: Jonas Fredriksson, Department of Electrical Engineering

Master's Thesis 2023
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Car and point cloud visualisation from Unity showing the point cloud covering the environment set up in the 3D software.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Extrinsic Lidar Calibration in a Workshop Environment
Evaluation of Concepts for Extrinsically Calibrating a Lidar sensor in an Automotive Workshop Environment with Respect to the Vehicle Coordinate System
MARCUS BERG, LUDVIG ERIKSSON
Department of Electrical Engineering
Chalmers University of Technology

Abstract

As the automotive industry moves toward an increasingly automated reality and thus is getting more reliant on sensor driven features, the need of accurate calibration methods is getting increasingly critical. To verify reliable performance during the life span of a vehicle, methods to recalibrate sensors during service are developed to accommodate for the limited equipment being present in workshop environments. Proof of concepts for extrinsic calibration of a lidar sensor are developed and evaluated on this premise to investigate how calibration methods can be adapted to work with limited resources.

Firstly, a target-based calibration algorithm is modeled to compute the extrinsic parameters of the sensor by computing the rigid transformation between the known position of calibration targets and the estimated position of the same targets seen by the lidar sensor. Secondly an alternative calibration algorithm is implemented that utilizes the internal IMU unit as well as the lidar sensor. The 3D points seen by the lidar sensor are accumulated with some estimated extrinsic parameters over a trajectory. The quality of the accumulated point cloud is then evaluated by comparing how the environment is seen by different scans over time. The algorithm optimizes over different extrinsic parameters using a grid search to find the estimated parameters of the lidar sensor.

The target-based method proved accurate when correct target position was found but is highly sensitive to poor target localization. Lidar-to-IMU calibration shows promising results in simulation with a high accuracy, few requirements on the environment and no additional equipment needed.

Keywords: lidar, extrinsic calibration, IMU, target-based, simulation.

Acknowledgements

We would like to thank our supervisors, Elías Marel and Daniel Åhlund for providing support and feedback throughout the thesis, both through the tools required and discussions about processes or results. We'd also like to thank our examiner Jonas Fredriksson for showing genuine interest and guiding us through the thesis writing process. We would also like to thank Volvo Car Corporation and the lidar team for giving us the opportunity to work with exciting new technology, and nice and helpful people.

Marcus Berg, Ludvig Eriksson, Gothenburg, June 2023

List of Acronyms

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

AD	Autonomous Drive
ADAS	Advanced Driver Assistance Systems
AV	Autonomous Vehicle
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
GNSS	Global Navigation Satellite System
GPU	Graphics Processing Unit
IMU	Inertial Measurement Unit
Lidar	Light Detection and Ranging
NIR	Near-InfraRed
RANSAC	Random Sampling Consensus
ROI	Region Of Interest
SRI	Spherical Range Images

Contents

1
Introduction

In an increasingly autonomous world sensors play a big part, being the eyes and ears of the machines. In the automotive world, where things are always moving, it is important to know the location of objects around the vehicle and their orientation in relation to the vehicle itself to know how and when to move to reach your destination while avoiding collisions with objects or people.

1.1 Background

Like many other sectors, the automotive industry is moving towards an automated and sensor-driven future. In order to make vehicles safer and optimize driving behaviour, control algorithms and safety functions are being utilized to an increasing degree. In this evolution, the need for new and better sensors are an ongoing challenge for manufacturers. As a result of added functionality and with more technology built into the cars, the service and reparation difficulties are increasing as well.

Recently Volvo Cars has introduced a lidar to their arsenal of sensors, located on the roof of the car, just above the windshield. In order to make use of the sensor readings it is of utmost importance that the sensor position and orientation in relation to the vehicle is well known. This implies that if some part of the vehicle in proximity to the lidar or the sensor itself were to be replaced or otherwise serviced, it might need to be calibrated to the vehicle coordinate system again.

To make the process of calibrating the sensor after replacement or service reliable, a robust and repeatable method of performing this task in workshop environments is needed.

1.2 Purpose

The purpose of this thesis is to compare concepts for extrinsic calibration of a lidar sensor in a workshop environment following possible service or replacement that might affect the position or orientation of the sensor. The purpose of the calibration is to ensure that the position and orientation in relation to the car coordinate system are well known. The methods or algorithms should be evaluated based on key parameters set in the scope of the project.

1.3 Objective

The main objective of the thesis is to develop a proof of concepts for performing extrinsic calibration on the lidar sensor in a workshop environment and compare their performance. The research questions that should be answered in the thesis are as follows:

- What methods are used today for sensor calibration, what do they require and under what circumstances do they have good performance?
- Which of these methods are best suited for the purpose of the thesis, in regards to the requirements set in the scope of the thesis?
- What is the expected precision for each of the chosen methods, what might impact the performance and are there possible improvements that can be implemented?

1.4 Scope

The thesis will cover methods on how extrinsic calibration of lidar can be performed and in what circumstances these methods can be utilized. Suitable concepts are then to be modeled, analyzed and compared based on their precision, speed, ease of use, cost and space efficiency. The thesis will only cover the development and evaluation of virtual concepts, and no physical concepts will be designed, realized or evaluated.

2

Ethics and sustainability aspects

One of the most prevalent problem with sensors on cars and other vehicles is the collection of data that may affect the privacy of both the owners of the vehicles, but also the people in their surroundings[1]. Especially when data might be saved for future diagnostics or improvements of the in place systems. As a bystander it might be hard to know what happens to the camera pictures or other sensor data that a car records when it passes by and it might feel a bit privacy intruding at times. However, with more data comes more possibilities, and that data might actually help to develop safer and more robust systems in the future, contributing to safer cars and environments, especially where traffic is a large factor, e.g. cities.

Another big problem that is quite well known is the ethical dilemmas that arise with self driving cars, which is a future use case of the sensor. Who's at fault for possible crashes or other accidents caused by or involving self driving cars? Is it the company designing and manufacturing the car, is it the programmer behind the code or other factors or individuals in the environment? These questions will probably not be answered for quite some time, as there are many different opinions about this.

The Trolley problem [2], or versions of it are also often discussed when it comes to the self driving world. Letting a machine choose actions that may be fatal to some humans, and perhaps in some cases valuing human lives against each other. How would these choices be made and who is responsible for the choices that the car makes? It's a highly relevant and difficult problem to deal with since it has no commonly accepted correct answer because of it's highly subjective nature.

There are both positives and negatives for autonomous vehicles (AVs) seen from an environmental perspective. Due to the increased computational load required by more algorithms running on an AV, more computational power will be needed, increasing emissions due to manufacturing of e.g. computational hardware. Easier and faster travel are also factors that might increase emissions caused by AVs, since people without drivers licenses might suddenly have the possibility to travel by themselves. However, AVs also come with better possibilities of vehicle pooling and vehicle sharing which is positive from an environmental perspective. Another benefit to AVs is their ability to communicate with each other and react in a much quicker manner than humans, giving rise to the possibility of platooning. Platooning is a concept involving vehicles driving very close to each other, for example on the highway, reducing the total amount of drag for all vehicles and thus their total emissions [3].

In the year 2018, 1.35 Million people died as a result of road accidents [4]. Autonomous driving could improve road safety substantially when implemented in the right way and at the right time [5]. This could reduce the number of accidents around the world, and thereby the amount of deaths related to road traffic. Because of this large advantage with autonomous vehicles, we feel that this thesis is ethically defensible.

3
Preparatory work

The preparatory work covers knowledge gathered through researching related works about existing calibration methods or sensor setups similar to those present in the thesis. Knowledge gathered is to be used to determine what methods should be further developed, analyzed and compared later in the thesis. This chapter is divided into different sections, covering information about general categories of calibration.

3.1 Related works

In broad terms, extrinsic lidar calibration can be summed up into two categories; target-based and non-target-based. Within each category, there are of course a great diversity of algorithms to perform the task. The group of non-target-based calibration algorithms studied in this chapter will handle methods performing lidar-to-IMU calibration.

3.1.1 Target-based methods

A common method for extrinsic calibration is to utilize calibration objects with known geometry and pose [6]–[8] to calibrate sensors, by mounting them in known positions and orientations relative to the vehicle. The object positions and orientations are then estimated using the sensor, and with that estimation, the sensor pose in the reference system can be computed using the estimated and known target positions and orientations.

In [6] a method for extrinsic calibration is presented which utilizes a single target board at close range to estimate the sensor orientation and horizontal position. The proposed method is designed to be used in the assembly process and it is assumed that the inspection environment is limited in space. Furthermore, it is assumed that the environment is static and contains some degree of infrastructure such as rails. Thus the goal is to integrate a small target into the existing infrastructure. The proposed method is conducted in four steps: Segment target region of interest (ROI), Outlier removal & Plane fit, Detect Feature Points and Pose Estimation. Since a single lidar scan produces a vast amount of 3D points the data is preferably pre-processed before any actual computation is conducted. Thus the 3D points are divided into regions of interest to reduce the computational load. The flat target board is detected by automatically segmenting it from the background points with a Euclidean clustering algorithm. When the points of the target surface are segmented

a plane model can be fitted by estimating it using the least-square method. Since the data contains depth noise and outliers, a random sample consensus (RANSAC) plane fitting algorithm is used to remove outliers [9]. The corner points of the calibration board are chosen as the points closest to each corner, which may due to low resolution give an incorrect position of the actual corner points. The pose components can then be estimated by minimizing the error between the known position of the board and the translated and rotated corner points detected by the beams. The paper uses the Levenberg-Marquardt algorithm [10], [11] to optimize the components iteratively. The paper simulates the method using the Velodyne VLP-16 lidar with the target board at 2.5 m distance from the sensor and achieved a precision of 0.11° in yaw angle and 4.9mm misalignment in $X-$direction. In addition, the method was verified with an experiment test bench, with a ground truth of sub-degree level of precision. In the experiments a precision of 0.06° in yaw was achieved and 3.5mm in $x-$direction.

In [7] the method from [6] is further developed to improve the precision. The proposed method builds upon the same concept with a single rigid target at a known position, however, it uses a cubic target instead of a planar board. A problem with the planar board used in [6] is that the corner points of the target are hard to accurately measure with a lidar since the beams are not likely to detect the very end of the board. This is especially true when a lidar with poor resolution is used. In an attempt to solve this limitation of the sensor, the method uses a cubic target with orthogonal sides. Similarly to the planar board, the sides of the cube can be detected with the Euclidean clustering algorithm. The orthogonal planes are then fitted using a RANSAC-based plane estimation to get rid of outlier points. Due to the random component in the RANSAC algorithm and errors in the sensor data, the detected planes are not likely to be perfectly orthogonal, which will lower the precision of the pose estimation. Thus the initial plane estimations are refined by minimizing the orthogonality error. One plane is used as a reference to estimate a second orthogonal plane with a nonlinear optimization before the last plane can be derived from the cross product of the first two planes. This is optimized iteratively using the Levenberg-Marquardt method. From the fitted planes a center vertex can be extracted from the intersection of the three planes, from which the remaining vertices of the cube can be calculated using prior knowledge of the physical cube. The method achieved a precision of 0.062° in rotation and 0.741 mm in translation in simulation. Furthermore, it achieved a precision of 0.039° in rotation and 0.91mm in translation in experiments with a ground truth precision of 0.01°.

Another solution to the problem of accurately detecting the edge of the calibration board was presented in [8]. The method uses a planar target board with near-infrared (NIR) photodetectors attached close to the corner points of the board. The photodetectors output a signal when a beam is projected on it, which can be used to match the position of the detector to a point in the lidar point cloud and thus get a more accurate relation between the point cloud and the target. The beams projected on the photodetectors can easily be distinguished in the point cloud by the reflective intensity of the returning beams since the beams projected on the de-

tectors have a much higher reflectance compared to points in its surrounding. By matching this point to the point on the photodetector outputting the highest signal, feature points can be accurately estimated. With the experimental setup used, the method achieved a precision of 0.05° in yaw angle and 2.3 mm in $x-$misalignment.

A highly researched version of target-based calibration is lidar-to-camera calibration, which uses targets to determine the extrinsic calibration between a lidar sensor and a camera sensor. It works by finding the target in sensor data from both sensors and computing the transform between the two estimations. [12] uses a planar target (a board) with a black circle on it, placed in (at least six) different positions to secure all six degrees of freedom. For each position Rodriguez, Fremont and Bonnifait estimate the target center and normal vector in both sensor data and using all estimations to compute the rigid transform between the two sensors. [12] present errors in the range of 46.1059 mm for translation and 3.4362° for rotation when using six different poses (the minimum) and running the iterative calibration for 401 iterations.

3.1.2 Non-target-based Calibration

Non-target-based calibration doesn't use any specific targets, as opposed to target-based calibration which has specific, known targets and one such concept is lidar-to-IMU calibration. Lidar-to-IMU calibration is a method whose purpose is to estimate the trajectory of a moving vehicle from the data collected by a sensor (e.g. a lidar) and match that to the data collected by a global position satellite system and inertial measurement unit (GNSS/IMU). By matching the trajectories from the two sensors, it is possible to compute the extrinsic parameters between the two, which is used to extrinsically calibrate the lidar sensor [13], [14].

The authors of [13] propose a method built on the assumptions that the lidar observes a generally static environment, containing some 3D features. In that environment, they drive the ego vehicle, onto which the lidar is mounted, as well as a GNSS/IMU unit. The lidar used by Levinson and Thrun [13] is a multi-beam lidar, which means that the lidar has several different emitters and receivers, mounted in different angles and heights inside the sensor head. The head is then spun around 360° to create the point cloud. Using the multi-beam property of the lidar, Levinson and Thrun [13] propose a calibration method in which they drive the ego vehicle in an environment fulfilling their requirements. After collecting data for 15-20 seconds they try to accumulate the point clouds for each frame into a cohesive map using the data from the GNSS/IMU. With an incorrect extrinsic calibration of the sensor, the resulting point cloud looks smeared and blurry. To combat this, Levinson and Thrun do a grid search optimisation of the extrinsic parameters to find the combination that results in the best point cloud. They define the quality of the point cloud as an energy function, dependent on small local planes in the point cloud. For each point in a single beam, a small local plane is estimated using the point and its 20 nearest neighbors, after which the plane normal is stored as the point normal. The distance is computed between each point and its nearest neighbor in neighboring beams in the point's normal direction and summed up to compute the energy func-

tion of the point cloud. Levinson and Thrun make the assumption that because the ego vehicle is moving several beams will hit the same spot, even if they're internally mounted offset from each other, which is the assumption upon which the method is built. Using this method, they get results that are very robust, and from different log files with the same sensor, they get extrinsic calibration results that are within 1 cm in translation and 0.03° in rotation of each other. It should be noted that the experimental results are based on manual measurements of the sensor misalignment due to the lack of a sophisticated calibration station and relatively high tolerances.

Li *et al.* [14] model the problem as a factor graph problem which they solve using the Levenberg-Marquadt algorithm. They start by constructing a point map of the environment based on lidar data, which they split into voxels. Recursively, each voxel is split into eight smaller voxels until requirements on the points in each voxel are met. All the points in a voxel should belong to the same plane, defined by a plane normal and curvature neighboring voxels containing parts of the same plane are combined to reduce voxel storage complexity. After defining the map and all point associations the graph is created, on which the Levenberg-Marquadt method is run and manage to reach an accuracy of 9.7 mm in translation and 0.5128° in rotation.

3.2 Algorithm selection

The methods found in existing literature consist mainly of three methods from two method groups, target-based calibration in the form of traditional target-based calibration and lidar-to-camera calibration, as well as non-target-based calibration in the form of lidar-to-IMU calibration. These three are the most common methods presented in the existing literature and traditional target-based calibration as well as lidar-to-IMU calibration will be modeled and evaluated during this project.

Traditional target-based calibration is a generally straightforward concept, in which you have a known target (some 3D object/body or picture with known patterns) placed in a known position in relation to the sensor or vehicle. With a known target shape and size, the position of the target in relation to the sensor can be determined. Combining this with the known exact position of the target in relation to the vehicle gives the offset of the sensor in relation to the vehicle frame.

Lidar-to-camera calibration combines data from cameras and lidar of an object seen by both sensors to compute their relative positions to each other. This method will not be considered, since it's undesirable to be relying on another sensor that might as well have been serviced or exchanged. This report will focus on target-based calibration as well as the lidar-to-IMU calibration proposed by Levinson and Thrun [13].

The exact method of lidar-to-IMU calibration can differ from case to case, where in some cases it is possible to match trajectories computed from lidar data to those computed from IMU data. In other cases, it is possible to combine point clouds from

lidar frames into one large point cloud and optimize the point cloud by comparing the points returned by different beams.

4
Concept Modeling

To be able to model the concepts as well as compare them to one another, they were run on data from a simulated environment. This allows for relatively rapid and easy testing of the different parameters and the overall structure of the environment, to enable testing of optimal scene structure and where to put specific targets to maximize performance. The 3D environment was built in Unity [15] and the lidar was simulated using a proprietary software package developed by the lidar supplier, which gave an output similar to that of the actual lidar. Using IMU data collected in a real test vehicle, the vehicle's trajectory can be re-simulated in the virtual environment to simulate real-world driving and replaying real test scenarios created on a test track. Being able to re-simulate trajectories recorded in the real world with a human driver makes it hard to recreate exactly, but makes the scenario more realistic, as the calibration will be carried out by a human technician and their trajectories won't be perfect.

4.1 Target-based Calibration

Target-based calibration is fairly straightforward to model virtually. A target model is constructed using four spheres attached to some form of rod to simulate the spheres being mounted on a larger contraption, which can be seen in Figure 4.1. Spherical targets are used since they can be estimated using only four points on their surface, and look the same from every angle [16]. The long rectangular beam is placed along the ground in the simulation environment, and the sphere targets are offset in x from their mounting rods to make segmentation easier. The rods sticking out under the target are not visible to the sensor when simulating, due to the target being placed on the ground. The position of the spheres in relation to the ideal location of the lidar is known, the lidar can then be translated and rotated to simulate a mounting error. After translating and rotating the lidar using predetermined perturbations, data can be collected to later be used when finding the extrinsic parameters of the lidar. Through this process, multiple values on the extrinsic parameters can be tested and used to evaluate the efficiency of the method.

The data is exported to MATLAB [17] where all the data can be combined to create a more dense point cloud. A density-based spatial clustering of applications with noise (DBSCAN) [18] clustering method is applied on the point cloud to find potential objects that could possibly be spheres.

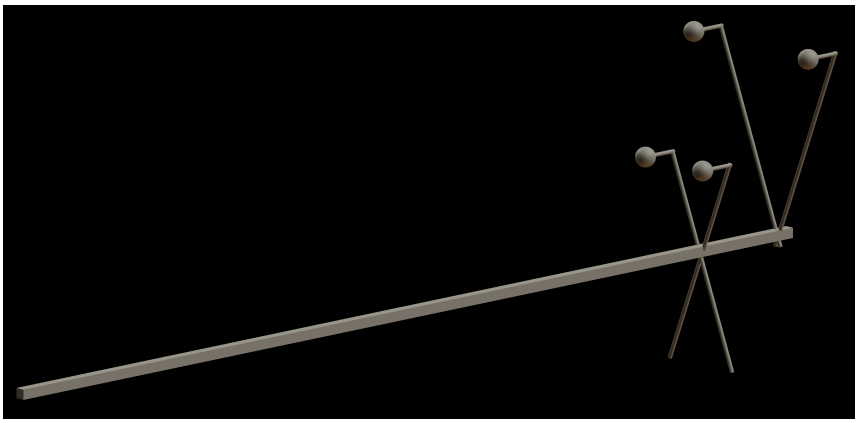


Figure 4.1: Target model used in Unity to simulate spheres mounted on a static target. The long rectangular beam is placed on the ground in the simulation environment, and the spheres are offset from their mounting rods to make segmentation easier. The rods sticking out under the target aren't visible to the sensor when simulating due to the target being placed on the ground.

To determine if a cluster is indeed a sphere, a random sample consensus (RANSAC) [9] method is used, wherein four points are selected, and used to solve the equation

$$\begin{vmatrix} x^2+y^2+z^2 & x & y & z & 1 \\ x_1^2+y_1^2+z_1^2 & x_1 & y_1 & z_1 & 1 \\ x_2^2+y_2^2+z_2^2 & x_2 & y_2 & z_2 & 1 \\ x_3^2+y_3^2+z_3^2 & x_3 & y_3 & z_3 & 1 \\ x_4^2+y_4^2+z_4^2 & x_4 & y_4 & z_4 & 1 \end{vmatrix} = 0, \tag{4.1}$$

where x_k, y_k, z_k are the coordinates for the k^{th} selected point, $k \in \{1,2,3,4\}$, and x, y, z are the coordinates for the sphere center. This determinant is only solvable if no three points are collinear, and if the four points are not coplanar. The determinant can be solved using the expansion by minors of the top row, which yields the equation

$$\begin{aligned} &(x^2+y^2+z^2)\begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix} - x\begin{vmatrix} x_1^2+y_1^2+z_1^2 & y_1 & z_1 & 1 \\ x_2^2+y_2^2+z_2^2 & y_2 & z_2 & 1 \\ x_3^2+y_3^2+z_3^2 & y_3 & z_3 & 1 \\ x_4^2+y_4^2+z_4^2 & y_4 & z_4 & 1 \end{vmatrix} + \\ &+ y\begin{vmatrix} x_1^2+y_1^2+z_1^2 & x_1 & z_1 & 1 \\ x_2^2+y_2^2+z_2^2 & x_2 & z_2 & 1 \\ x_3^2+y_3^2+z_3^2 & x_3 & z_3 & 1 \\ x_4^2+y_4^2+z_4^2 & x_4 & z_4 & 1 \end{vmatrix} - z\begin{vmatrix} x_1^2+y_1^2+z_1^2 & x_1 & y_1 & 1 \\ x_2^2+y_2^2+z_2^2 & x_2 & y_2 & 1 \\ x_3^2+y_3^2+z_3^2 & x_3 & y_3 & 1 \\ x_4^2+y_4^2+z_4^2 & x_4 & y_4 & 1 \end{vmatrix} + \\ &+ \begin{vmatrix} x_1^2+y_1^2+z_1^2 & x_1 & y_1 & z_1 \\ x_2^2+y_2^2+z_2^2 & x_2 & y_2 & z_2 \\ x_3^2+y_3^2+z_3^2 & x_3 & y_3 & z_3 \\ x_4^2+y_4^2+z_4^2 & x_4 & y_4 & z_4 \end{vmatrix} = 0, \end{aligned} \tag{4.2}$$

which can be written in terms of the minors M_{1j} as

$$(x^2+y^2+z^2)M_{11} - xM_{12} + yM_{13} - zM_{14} + M_{15} = 0. \tag{4.3}$$

Using the general equation of a sphere with radius r centered at (x_0, y_0, z_0)

$$(x - x_0)^2 + (y - y_0)^2 - (z - z_0)^2 = r^2 \tag{4.4}$$

and equating that with (4.2), it's possible to solve for the center and radius of the sphere

$$\begin{aligned} x_0 &= 0.5\frac{M_{12}}{M_{11}} \\ y_0 &= -0.5\frac{M_{13}}{M_{11}} \\ z_0 &= 0.5\frac{M_{14}}{M_{11}} \\ r^2 &= x_0^2 + y_0^2 + z_0^2 - \frac{M_{15}}{M_{11}}. \end{aligned} \tag{4.5}$$

It is important to note that these equations cannot be solved for the case where $M_{11} = 0$, which corresponds to no quadratic terms $(x^2 + y^2 + z^2)$ in which case the points do not lie on a sphere, and are either coplanar or collinear. Since the true radii of the targets are known and it is desired to fit the data to the known targets, the known radii are used when determining inlier points, instead of the radii found by solving (4.5).

The number of iterations run by the RANSAC algorithm is defined by the RANSAC formula

$$T = \left\lceil \frac{\log(1 - \alpha)}{\log(1 - \epsilon^s)} \right\rceil \tag{4.6}$$

where T is the number of iterations required to ensure a correct fit with α confidence, ϵ is the portion of inliers in the current best fit and s is the number of points used to solve (4.5). When a solution with more inliers than the previous best solution is found, ϵ and T are both recomputed and the number of required iterations decreases.

After solving (4.5) and acquiring the four targets, the transform between the real known center points and the estimated center points found in the lidar data can be computed, which gives the estimated translation and rotation of the lidar sensor. The transform is computed using the built-in MATLAB function `estgeotransf3d` which is based on [19]. To find the least squares solution the Kabsch algorithm is used [20], [21].

First, the two sets of target center points (estimated and known) must be translated so that their centroids coincide with the coordinate system origin, which is done by subtracting the centroids from the points

$$\begin{aligned} \mathbf{P}_\mathrm{C} &= \mathbf{P} - \mathbf{C}_\mathrm{P}, \\ \mathbf{Q}_\mathrm{C} &= \mathbf{Q} - \mathbf{C}_\mathrm{Q} \end{aligned} \tag{4.7}$$

where $\mathbf{P}_\mathrm{C}, \mathbf{Q}_\mathrm{C}$ are the new centered points, $\mathbf{P}, \mathbf{Q}$ are the original uncentered points and $\mathbf{C}_\mathrm{P}, \mathbf{C}_\mathrm{Q}$ are the centroid coordinates for the respective sets of points. To compute

the transform for $\mathbf{P}$ into $\mathbf{Q}$ the cross-covariance matrix $\mathbf{H}$ is computed between the two centered sets of points

$$\mathbf{H} = \mathbf{P}_{\mathrm{C}}^{\top}\mathbf{Q}_{\mathrm{C}}. \tag{4.8}$$

After computing $\mathbf{H}$ it is possible to compute the rotation $\mathbf{R}$ using singular value decomposition (SVD)

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}. \tag{4.9}$$

To ensure a right-handed coordinate system, the correction term, d, is computed as

$$d = \operatorname{sign}\left(\det\left(\mathbf{V}\mathbf{U}^{\top}\right)\right). \tag{4.10}$$

Finally, combining (4.9) and (4.10) the rotation can be computed as

$$\mathbf{R} = \mathbf{V}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & d\end{bmatrix}\mathbf{U}^{\top}. \tag{4.11}$$

Using the rotation, the translation t can be computed as

$$\mathbf{t} = \mathbf{C}_{\mathrm{True}} - \mathbf{R}\mathbf{C}_{\mathrm{Estimate}}, \tag{4.12}$$

where $\mathbf{C}_{\mathrm{True}}$, $\mathbf{C}_{\mathrm{Estimate}}$ are the known target centers and the estimated target centers respectively.

4.2 Lidar-to-IMU calibration

To model lidar-to-IMU calibration virtually is a bit more difficult compared to target-based modeling. Using real IMU data collected in a test vehicle was the initial idea, but proved difficult due to problems with the data, because of IMU drift. Instead, simulated data was made in MATLAB to validate and test the virtual data collection without being affected by IMU drift. For data collection, the same procedure was followed as for the target-based method, with the same set of perturbations to allow for a comparison that is as fair as possible.

The lidar-to-IMU calibration method in [13] is originally proposed for another type of lidar than the one used in this work, however, the method can be applied to our lidar sensor. Data is collected through the virtual simulation environment, in which the vehicle drives along a known trajectory. Since the lidar sensor and the GNSS/IMU sensor are mounted in different locations in the vehicle, the lidar is modeled as being attached to an arm with a known length and angle from the GNSS/IMU sensor. Because of this, the trajectory of the lidar is different than the trajectory of the GNSS/IMU unit, and since the lidar is the sensor collecting point cloud data, it is the trajectory of the lidar that is most important. To compute the trajectory of the lidar, the origin of the lidar is treated as a point in the coordinate system of the GNSS/IMU sensor, which is moving in space. The lidar origin has a constant position in the GNSS/IMU coordinate system, but since the coordinate system origin (the GNSS/IMU sensor) is moving in space, so is the lidar origin. By

treating the lidar origin this way, the trajectory $\mathcal{T}_{\text{lidar}}$ can be computed by simply applying a transform $\mathbf{T}$ to the GNSS/IMU trajectory $\mathcal{T}_{\text{GNSS/IMU}}$:

$$\mathcal{T}_{\text{lidar}} = \mathbf{T}\mathcal{T}_{\text{GNSS/IMU}} \tag{4.13}$$

where

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{\text{lidar}} & \mathbf{t}_{\text{lidar}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.14}$$

where $\mathbf{R}_{\text{lidar}} \in \mathbb{R}^{3\times3}$ is the rotation matrix for the rotational offsets of the lidar in relation to the GNSS/IMU, $\mathbf{t}_{\text{lidar}} \in \mathbb{R}^{3\times1}$ is the translational offset of the lidar in relation to the GNSS/IMU and $\mathbf{0} \in \mathbb{R}^{1\times3}$ is a zero vector.

To combine the data points from the lidar into one big global map, the data points are transformed based on the position of the sensor at the time of data collection. For each frame collected by the lidar sensor, its position and orientation are extracted from the trajectory and used to transform the points and convert them from local lidar points to global points.

Because of the transformation, the new global map is subject to errors in the position and orientation of the lidar, which is the principle used to calibrate the sensor. By trying different combinations of extrinsic parameters when applying the transform from local lidar to global points and evaluating the quality of the resulting point cloud it is possible to find the true extrinsic parameters, also known as calibration.

To evaluate the optimal extrinsic calibration an energy-like cost function is used, which is similar to the energy function provided by Levinson and Thrun, [13]:

$$\begin{aligned} J &= \frac{1}{n}\sum_{b_i=1}^{B}\sum_{b_j=b_i-N}^{b_i+N}\sum_{k} w_k \left|\left|\boldsymbol{\eta}_k \cdot (\mathbf{p}_k - \mathbf{m}_k)\right|\right|^2, \\ w_k &= \begin{cases} 1, & ||\mathbf{p}_k - \mathbf{m}_k|| < d_{\max} \\ 0, & ||\mathbf{p}_k - \mathbf{m}_k|| \geq d_{\max} \end{cases} \end{aligned} \tag{4.15}$$

where n is the total number of points, B is the total number of beams in the lidar sensor, N is the number of neighbouring beams to consider, k iterates over the points returned by beam b_j, $\mathbf{p}_k$ is the k^{th} point in the transformed set, $\mathbf{m}_k$ is the closest point to $\mathbf{p}_k$ seen by beam b_i and $\boldsymbol{\eta}_k$ is the surface normal at point $\mathbf{m}_k$. The energy score is normalized by n to enable comparison between runs with different amounts of data and w_k is a threshold value filtering out contributions by points that are too far away, $d_{\max}$. By multiplying with the point normal, distances within the plane are neglected and only the distance in the normal direction is considered.

The energy value is lower for extrinsic parameters that give a better combination of point cloud frames. Thus the best extrinsic combination can be found through trial and error using an iterative grid search. For the first iteration of the iterative grid search, an initial set of evenly spaced values is generated, in the range of estimated possible mounting errors, for each parameter. All the values in the initial set are

then evaluated by stitching together the frames and computing the energy value. After the first iteration, the n parameter combinations with the lowest energy values are stored and used to generate the new combinations for the next iteration. For each of the best combinations, m values are generated around the previous value for each parameter, in the interval centered around the previous value with a width of the previous iteration step size. The new step size is constrained to not be smaller than $deg_{min}{}^{\circ}$ to shorten computation times and stop convergence when the method has reached satisfactory precision. There are $n \cdot m^3$ combinations to evaluate each iteration and because of high computational time, the method uses at most N iterations. However, the calibration may terminate prematurely if the solution has converged which is determined by saving the best solution after every iteration and terminating if the best score has not changed more than 0.001 units during the last l iterations.

The algorithm depends heavily on the surface normal of each point and it is therefore of great importance that they are accurately computed. In the algorithm presented in [13] the surface normals are computed for the points of each beam separately on the accumulated and transformed point cloud by fitting a plane to the 20 closest points. In this thesis, we propose a modification to the normal computation to achieve more accurate surface normals with fewer computations required. The modifications are inspired by the work [22], where a number of different modifications to the traditional least squares approach of computing surface normals are examined. Their work is presented to be used for Spherical Range Images (SRI) but the same method can be applied to a standard 3D point cloud as well.

The traditional least squares for a subset of k 3D points $\mathbf{p}_i, i = 1, 2, 3, ...k$ is used to find the normal vector $\mathbf{n} = (n_x, n_y, n_z)^T$ and the offset d that minimizes

$$e = \sum_{i=1}^{k} (\mathbf{p}_i^T \mathbf{n} - d)^2, \quad \text{subject to } |\mathbf{n}| = 1. \tag{4.16}$$

The normal vector can then be found by eigenvector analysis by finding the eigenvector that corresponds to the smallest eigenvalue of the covariance matrix

$$M = \frac{1}{k} \sum_{i=1}^{k} (\mathbf{p}_i \bar{\mathbf{p}})(\mathbf{p}_i \bar{\mathbf{p}})^T, \quad \bar{\mathbf{p}} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{p}_i. \tag{4.17}$$

The solution utilized for normal computation is the method described as Unconstrained Least Squares in [22]. This method completely disregards the need of an eigenvalue analysis and is computationally significantly more effective compared to the traditional least squares approach. Both side of the loss function Equation 4.16 is divided with the offset term d^2 that constrained the normal vector, resulting in the solution

$$\begin{aligned} \tilde{e} &= \sum_{i=1}^{k} (\mathbf{p}_i^T \tilde{\mathbf{n}} - 1)^2, \quad \tilde{\mathbf{n}} = \tilde{\mathbf{M}}^{-1} \tilde{\mathbf{b}} \\ \tilde{\mathbf{M}} &= \sum_{i=1}^{k} \mathbf{p}_i \mathbf{p}_i^T, \quad \tilde{\mathbf{b}} = \sum_{i=1}^{k} \mathbf{p}_i. \end{aligned} \tag{4.18}$$

Table 4.1: Extrinsic poses used for evaluation of the methods.

data set	ω	φ	κ
1	1.960°	1.140°	0.484°
2	0.840°	-1.240°	-0.936°
3	0.336°	-0.364°	-1.696°
4	1.756°	1.432°	-1.800°
5	0.916°	1.544°	-0.932°
6	-0.244°	0.912°	0.452°
7	-0.680°	1.360°	-0928°
8	-1.004°	-0.200°	1.984°
9	0.812°	-1.460°	-1.024°
10	0.992°	0.300°	-0.836°

The normal vector can then be obtained by normalizing the result $\mathbf{n} = \tilde{\mathbf{n}}/|\tilde{\mathbf{n}}|$.

Despite being computationally effective, the runtime of normal computations is still rather long using the method presented in [13], as the surface normals are recomputed for every parameter set tested in the grid search. To avoid this, another approach is being utilized where the surface normals only need to be computed once at the start of the calibration. The normals are computed separately for the points seen in each lidar frame and are transformed similarly but a bit differently than their points. When transforming the normals only the motion in the yaw rotation is considered since this is the most significant parameter for normal transformation, which leads to better results in practice. Despite not being theoretically optimal, it results in better performance in practical scenarios, since transforming a set of vectors is much faster than computing surface normals. The runtime of testing for each parameter setting is reduced by magnitudes using this modification of the algorithm.

4.3 Data generation

To evaluate the methods sensor readings from the different setups are needed. In order to have full control over the extrinsic pose of the lidar sensor and the precision achieved by the algorithms, synthetic data is generated as the base for evaluation. To evaluate the performance of the methods over the entire range of possible extrinsic poses of the lidar sensor, a set of ten different combinations of poses are randomly generated to be validated on. The angles are generated in the range $\left[-2,\ \ 2\right]$ as this is assumed to be the maximal possible mounting error possible to achieve, see Table 4.1.

To simulate the behaviour of a physical sensor a simple noise model will be used as described in Equation 4.19 since the simulation tool does not support any noise model. The noise model parameters are tuned to crudely resemble the real noise of

the sensor used in the thesis.

$$\begin{aligned} P &= \begin{bmatrix} x_{noisy} & y_{noisy} & z_{noisy} \end{bmatrix} \\ x_{noisy} &= x + a \cdot r \\ y_{noisy} &= y + b \cdot r \\ z_{noisy} &= z + c \cdot r \\ r &\sim \mathcal{U}(-0.5, 0.5) \end{aligned} \tag{4.19}$$

4.3.1 Target-based Calibration

Data generation for target-based calibration is a rather straightforward process. The environment including targets with their fixture, vehicle and other surrounding objects are imported to Unity. With the environment set up, lidar readings can be simulated using proprietary packages provided by the sensor supplier. The algorithm can then be applied to the point cloud P to attempt to estimate the extrinsic sensor pose used when recording the point cloud.

4.3.2 Lidar-to-IMU Calibration

In contrast to the target-based method, the lidar-to-IMU calibration requires both IMU data as well as lidar data. The IMU data used for evaluating the algorithm is a single synthetic trajectory generated in MATLAB that is used when recording all test scenarios in Table 4.1. The lidar data is then recorded in Unity similarly to the process described for the target-based method, with the difference that the vehicle is moved according to the GNSS/IMU trajectory when recording data. The noise model Equation 4.19 is then added to the recorded lidar point cloud before it can be used to evaluate the algorithm.

The recorded lidar data from Unity proved to be somewhat unreliable due to odd behaviour in the timestamps. The timestamps are used to synchronize the two sensors and time synchronization is an absolute necessity for the algorithm to work. Therefore an additional dataset is generated and bypasses the simulation process. A single frame from the recorded point cloud is exported and used to generate a synthetic alternative for the simulated sensor readings. This is done by stepping through the IMU data and for each reading transforming the point cloud with the lidar position and pose associated with the IMU reading at that time instance. The transformed point cloud is then assigned scan lines by dividing the sensor's vertical field of view into segments of angles. The visible points from each angle segment are considered to belong to a common scan line, which gives a quite similar result to the physical behaviour of the real sensor.

5
Results

The methods are evaluated in simulation in order to achieve complete control of the ground truth and the precision of the estimated extrinsic pose. A similar setup is used for both methods to allow for a fair comparison. As for the lidar sensor readings used when evaluating the calibration methods, simulated data is recorded with the same settings and scan patterns for both cases. The noise model used is constructed to resemble true sensor data but its accuracy is not known. The evaluation is performed on a set of ten randomly generated extrinsic poses of the lidar sensor with a precision of 0.001°, seen in Table 5.1. The poses are generated to be a misalignment in the range $[-2°, 2°]$ for each rotational axis as this is the range in which it is possible to mount the sensor.

For each data set the calibration is performed N_{runs} times to evaluate the robustness of the methods and to what degree they are deterministic. Since the lidar-to-IMU method is significantly more computationally heavy, it therefore has a longer run time and thus this number will vary between the two methods.

5.1 Evaluation of target-based calibration

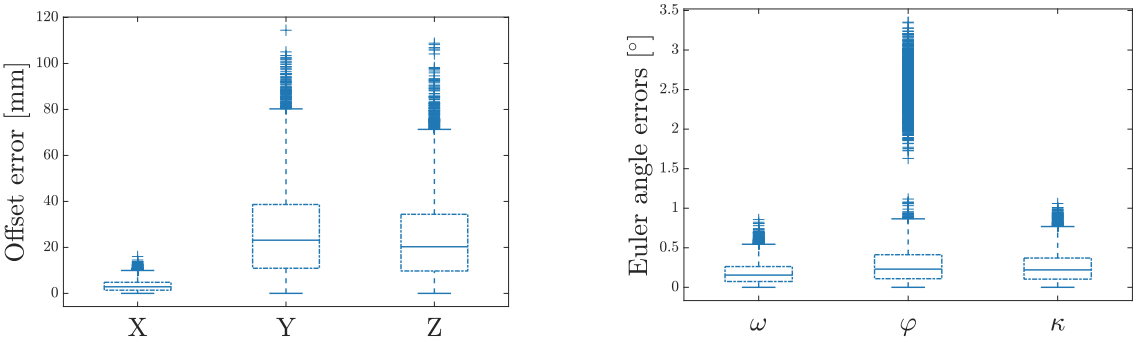
To evaluate the target-based calibration method, data were simulated and recorded with Unity using the target shown in Figure 4.1 with the sensor in several different poses, consisting of different rotations in roll, pitch and yaw. This section consists of two parts with the first part covering the results of simulated data using the targets estimated by the method, and the second part covering an analysis of where the method underperforms. The analysis lays the ground for how the method might be improved and is further discussed in chapter 6.

5.1.1 Results using estimated targets

After accumulating 10 frames into one dense frame, the DBSCAN algorithm is run using parameters $\varepsilon = 0.025$ m and $n = 5$ which defines the neighbour search radius around each point, and the number of neighbours within that radius for the point to be considered a core point respectively. Following the DBSCAN, the RANSAC algorithm is run using four points ($s = 4$) and the RANSAC formula (4.6) to determine how many iterations to run. RANSAC is run with a confidence value of $\alpha = 0.999$, an initial inlier portion estimate of $\varepsilon_0 = 0.1$ and a radius threshold of 0.01 m. After receiving the estimate from the RANSAC algorithm, each object is

Table 5.1: Extrinsic poses used for evaluation of the methods.

Data set	ω	φ	κ
1	1.960°	1.140°	0.484°
2	0.840°	-1.240°	-0.936°
3	0.336°	-0.364°	-1.696°
4	1.756°	1.432°	-1.800°
5	0.916°	1.544°	-0.932°
6	-0.244°	0.912°	0.452°
7	-0.680°	1.360°	-0928°
8	-1.004°	-0.200°	1.984°
9	0.812°	-1.460°	-1.024°
10	0.992°	0.300°	-0.836°



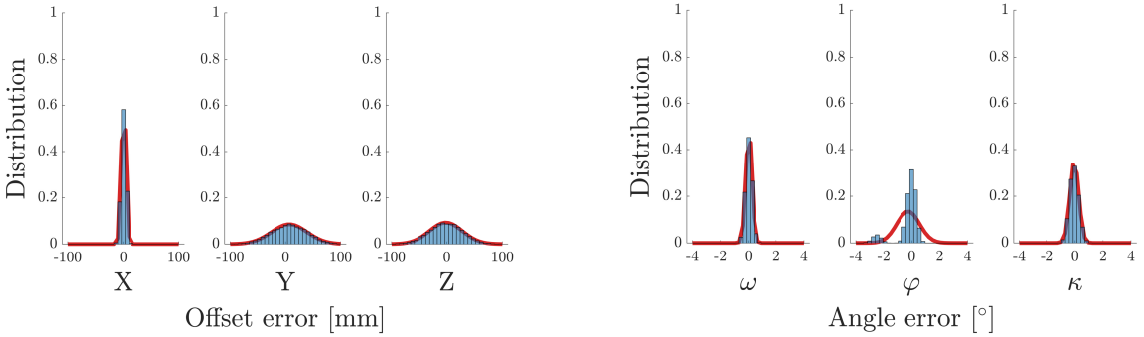
(a) Offset errors.

(b) Euler angle errors.

Figure 5.1: Box plot of absolute errors over the six parameters with X, Y and Z in (a) and ω, φ and κ in (b). The box plot shows the sizes of the errors and as can be seen, the size differs between the different parameters with e.g. X generally having smaller errors than Y and Z with the three angles having roughly similarly sized errors, apart from pitch (ω) having less spread. The solid line dividing the boxes is the median of the data, while the upper and lower edges of the boxes show the 25^{th} and 75^{th} percentiles, respectively. The difference between the 25^{th} and 75^{th} percentiles is called the interquartile range, and the whiskers extend to the farthest value, within 1.5 interquartile ranges, from the 25^{th} and 75^{th} percentiles. All values outside the whiskers are considered outliers, here shown with + markers, and it's interesting to note that the pitch angle (φ) has many more outliers and especially more outliers farther from the whisker compared to the other parameters.

checked using two criteria to try and determine whether it is a sphere or not. The first criterion checks the number of points in the cluster, which must be greater than 100 points per accumulated frame, and the other checks if the portion of inliers in the cluster is greater than the threshold of 0.8.

The method was run on ten different sensor poses, as seen in Table 5.1 and for each pose the method was run 1000 times. A box plot showing the sizes of the errors can be seen in Figure 5.1a for offsets and Figure 5.1b for Euler angles. The solid line



(a) Offset errors.

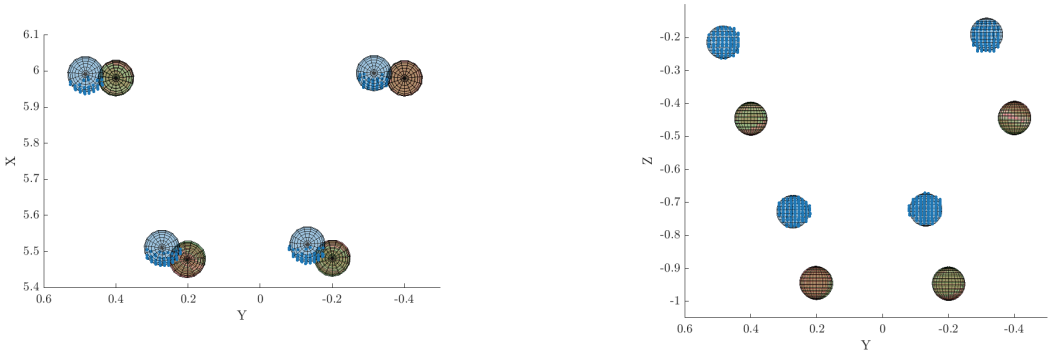
(b) Euler angle errors.

Figure 5.2: Histograms over (a) offset and (b) Euler angle errors. For X the errors are distributed quite close to zero, which is to be expected after viewing figure 5.1, and Y and Z are more normally distributed around zero. For the Euler angles in (b) it's hard to say whether the errors are normally distributed or not. Roll and yaw angle errors (ω and κ respectively) seem to be normally or close to normally distributed while the pitch angles (φ) have some outliers in the negative range.

dividing the boxes is the median of the data, while the upper and lower edges of the boxes show the 25^{th} and 75^{th} percentiles, respectively. The difference between the 25^{th} and 75^{th} percentiles is called the interquartile range, and the whiskers extend to the farthest value, within 1.5 interquartile ranges, from the 25^{th} and 75^{th} percentiles. All values outside the whiskers are considered outliers and are here shown with + markers. It is interesting to note that the pitch angle (φ) has many more outliers and especially many farther from the whiskers compared to the other Euler angles and the errors for the X parameter are smaller than the typical errors on the Y and Z parameters. The errors on the Euler angles are more even, with the roll angle (ω) having smaller errors, but not on the same scale as X.

Histograms over the distribution of the errors are shown in Figure 5.2 and from these, it appears that the offset errors in Figure 5.2a are approximately normally distributed with different standard deviations, with X errors having a smaller amplitude and standard deviation. A normal distribution seems to be accurate for the Euler angle errors in Figure 5.2b as well, at least for roll (ω) and yaw (κ). Pitch (φ) seems to have a bimodal normal distribution with some outliers in the negative ranges.

Overall, the errors are unreasonably large compared to what has been presented in the existing literature. To investigate why the method does not perform as desired, each module or part of the method was replaced by a cheat method which returns perfect data to the subsequent parts of the chain to determine if they are performing as they should.



(a) Top down view.

(b) Front view.

Figure 5.3: Sphere fit on selected clusters compared with ground truth and adjusted targets. Blue spheres are the estimated targets fitted to the blue points, the red spheres are the estimated targets adjusted with the found transform and the green spheres are the ground truth targets. Here it's apparent that the method finds the points belonging to the spheres by correctly segmenting and classifying the clusters. However, looking at the top-down view in (a) the estimated spheres seem to have different positions relative to each other, compared to the ground truths, which can be seen especially for the red and green spheres closest to the Y axis.

5.1.2 Investigation of method performance

The first part of the chain to be investigated is sphere localization which in itself consists of different parts. The clustering/segmentation part is already verified that it works, and results can be seen in Figure 5.3 where the blue spheres are estimated targets, which are fitted to the blue points, the red spheres are the blue spheres adjusted using the ground truth transform and the green spheres are the ground truth targets. From Figure 5.3 it can be determined that the method correctly finds the points corresponding to the spheres and managed to pair them to the correct ground truths. However, looking at the top-down view in Figure 5.3a it is apparent that the estimated spheres in blue do not have the same position relative to each other compared to the ground truth in green. This is especially clear comparing the red (adjusted estimated) spheres to the green spheres, and seeing that they are misaligned in different directions, mainly for the two spheres closest to the Y axis.

A cross-section of the upper left blue sphere is shown in Figure 5.4 visualizing that the points are not perfectly distributed on the sphere surface, but instead being grouped in the X dimension.

Because of the small errors in the localization part, it was bypassed by transforming the known ground truth locations $\mathbf{X}_{\text{Ground Truth}}$ as

$$\mathbf{X} = \mathbf{A}\mathbf{X}_{\text{Ground Truth}} \tag{5.1}$$

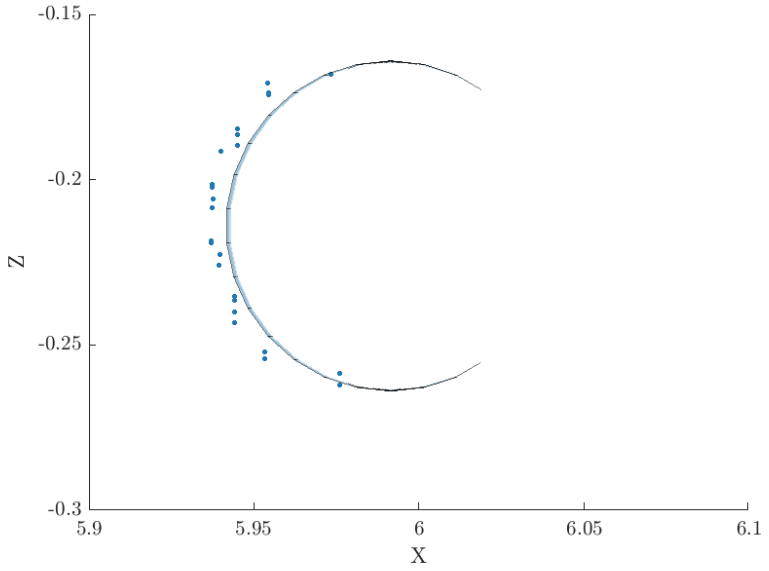


Figure 5.4: Cross section of the upper left blue sphere, showcasing the distribution of the points on the target. It is clear that the points on the sphere are not lying on a perfect sphere, but are grouped in the X dimension with several points having the same X value.

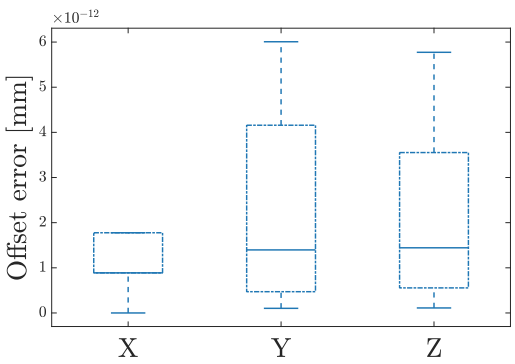
where

$$\mathbf{A} = \begin{pmatrix} \cos\varphi\cos\kappa & -\cos\varphi\sin\kappa & \sin\varphi & X_{\text{Offset}} \\ \cos\omega\sin\kappa + \cos\kappa\sin\omega\sin\varphi & \cos\omega\cos\kappa - \sin\omega\sin\varphi\sin\kappa & -\cos\varphi\sin\omega & Y_{\text{Offset}} \\ \sin\omega\sin\kappa - \cos\omega\cos\kappa\sin\varphi & \cos\kappa\sin\omega + \cos\omega\sin\varphi\sin\kappa & \cos\omega\cos\varphi & Z_{\text{Offset}} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.2}$$

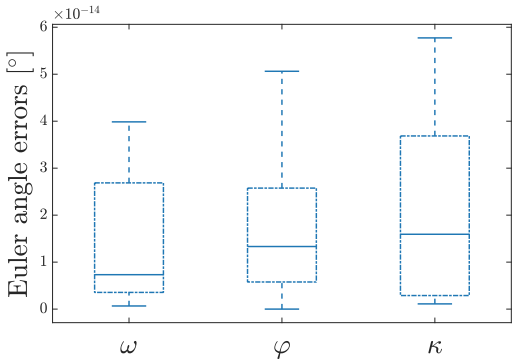
where ω, φ and κ are the lidar rotations around the X, Y and Z axes respectively and X_{Offset}, Y_{Offset} and Z_{Offset} are the offsets in the Cartesian dimensions. Using fake targets, with their own generated point clouds, yielded near-perfect performance.

Looking at Figure 5.5 shows that errors are practically nonexistent when using cheat targets, compared to using real targets. This means that computing the transform between the two sets of targets is not the issue, but rather the positions of the estimated targets in relation to each other. The distribution of the errors using cheat targets is more or less uniformly distributed as seen in Figure 5.6, but their distribution is not really relevant because of their small magnitude. This further reinforces that computing the transformation is very accurate when having perfect targets, and is probably not the cause of the large errors using the real targets.

In Figure 5.7 the varying values of the estimated sphere center can be seen, both between different frames and accumulated over several runs for each frame. The values for each parameter are the distances in each direction from the estimated center to the ground truth. The mean and standard deviation tell little, but from

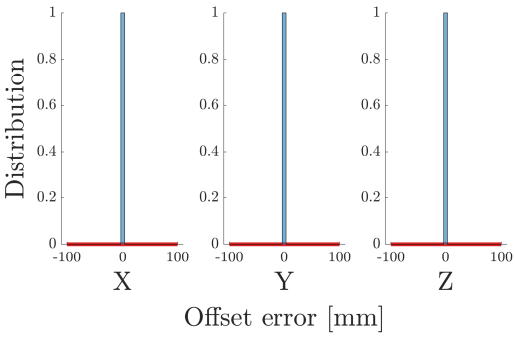


(a) Offset errors.

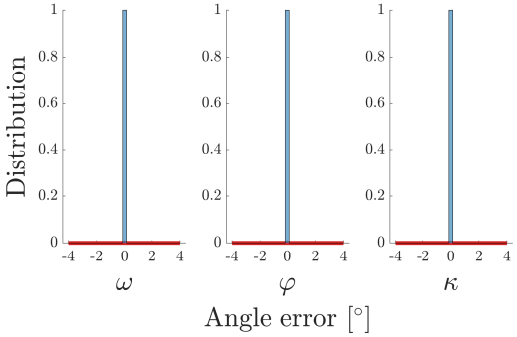


(b) Euler angle errors.

Figure 5.5: Boxplots of (a) offset and (b) Euler angle errors using cheat targets. When using cheat targets, the errors are practically nonexistent (note the coefficient on the error axes) for both offset and angle errors. This means that computing the transform from the correct targets are working correctly and that the fault lies within the localization algorithm.



(a) Offset errors.



(b) Euler angle errors.

Figure 5.6: Distribution of errors when using cheat targets to estimate the transform. The errors are practically nonexistent but there are some small errors, which seem to be uniformly distributed, but these errors can be neglected because of their magnitude.

the minimum and maximum lines, it is apparent that from one run to the other the estimated centers can differ as much as 20 millimetres in one dimension, even when running RANSAC with 99.9 % confidence. This means that the method to find the sphere centres is not working as well as is desired.

One big strength of the target-based method is the run time. The method was run on a laptop equipped with an Intel Core i7-11850H processor, an NVIDIA RTX A3000 mobile GPU and 32 GB of ram, which resulted in a mean run time of around 2.5 seconds when accumulating 10 frames. Accumulating more frames will result in more data, increasing the run time of the method.

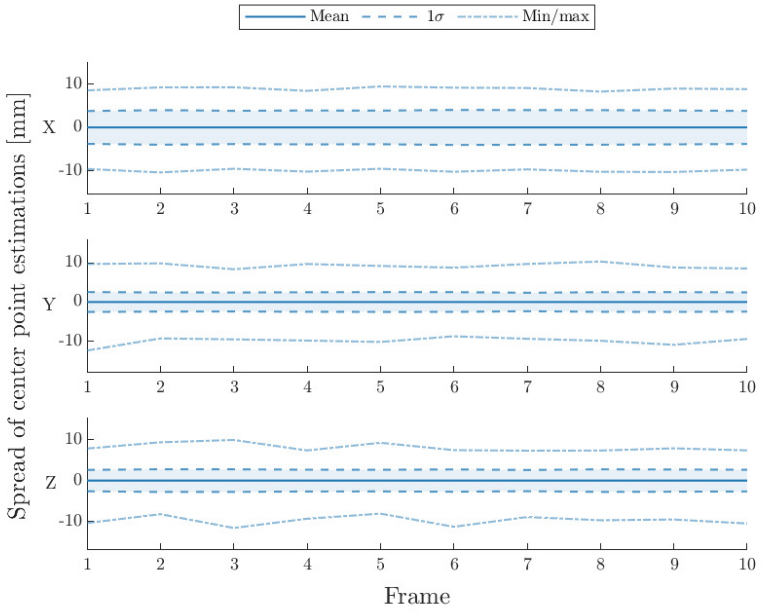


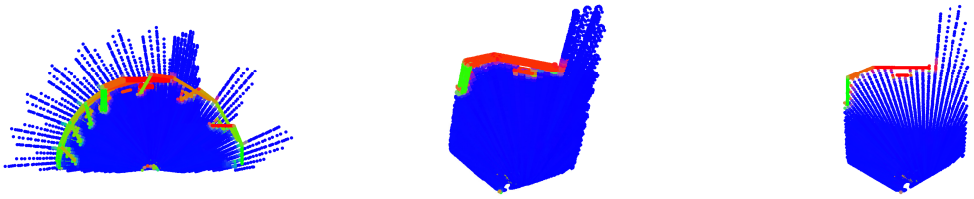
Figure 5.7: Spread of estimated centers in mm along each dimension, with the solid line representing the mean of the center estimations, the dashed lines representing one standard deviation from the mean, and the outer dash-dotted lines the minimums and maximums. The results are from ten different frames shown on the X-axis, and for each frame, the centers were estimated 1000 times with a RANSAC confidence of 99.9%. In a worst-case scenario, the estimated center from two different runs can differ 20 mm or more, in a single dimension.

5.2 Evaluation of Lidar-to-IMU calibration

The lidar-to-IMU calibration method is based on comparing points seen by different scan lines when the point cloud is accumulated while moving the vehicle. The calibration is done offline and after the data collection is done the process can be divided into three major parts; transforming and accumulating points, computing surface normals and finally computing an energy score. While evaluating the proof of concept in this thesis the data collection is performed completely synthetically. Each part of the calibration algorithm relies on the result from the previous step and is dependent on its performance. Thus each step will first be evaluated separately before the performance of the full calibration algorithm is evaluated.

5.2.1 Accumulate point cloud over trajectory

If the recorded point cloud were to be accumulated without first transforming the points according to the vehicle's motion, the resulting point cloud will look similar to the one in Figure 5.8a. In order for the calibration algorithm to work as intended each frame of points must be transformed according to the vehicle position by using the readings from the GNSS/IMU sensor. If the correct extrinsic pose of the lidar sensor is used in this process, the resulting accumulation of the point cloud will look like Figure 5.8c where the environment will have very crisp edges. However, if the



(a) No transformation **(b)** Incorrect extrinsic pose **(c)** Correct extrinsic pose

Figure 5.8: Accumulated point cloud over trajectory. In (a) the point cloud is accumulated without transforming the points according to the position of the vehicle at the time of the scan, resulting in a chaotic point cloud. In (b) the point cloud is accumulated using the IMU readings but with poorly estimated extrinsic parameters, resulting in a point cloud that resembles the environment but has quite a lot of noise. In (c) the point cloud is accumulated using IMU readings and the true extrinsic parameters of the sensor, resulting in a crisp point cloud.

point cloud is accumulated with a poor estimation of the extrinsic pose of the lidar sensor, the edges of the 3D objects in the environment will become very blurry and somewhat distorted as seen in Figure 5.8b.

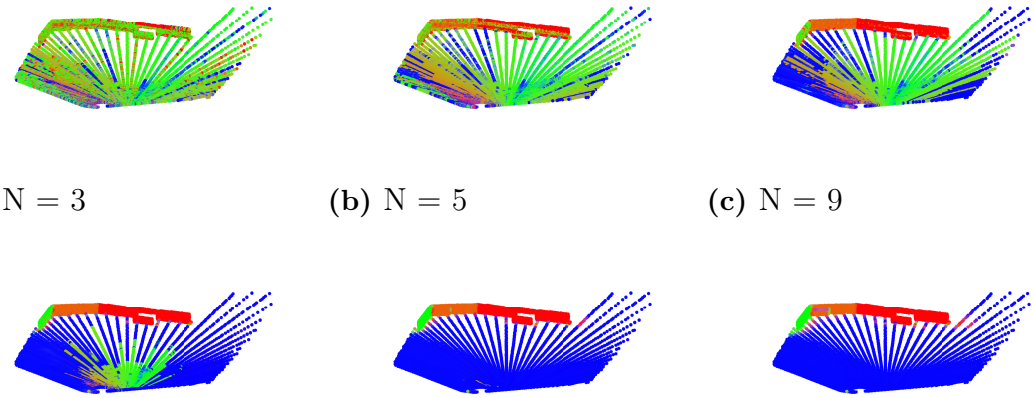
Since the two sensors have different mounting positions on the vehicle body, they thereby move along slightly different trajectories due to this offset. The transformation between the IMU and lidar trajectory is validated by confirming that the correct pose in Figure 5.8c yields a crisp point cloud even after the vehicle has moved along a turning trajectory.

5.2.2 Compute surface normals

In the original paper by Levinson and Thrun [13] the surface normals were computed separately for the points seen by each beam on the accumulated point cloud, by computing the plane that best fits the 20 closest points. In this thesis, a modification to this computation has been proposed to make the process faster and more robust in the setting of using a single-beam lidar.

In Figure 5.9 the surface normals have been computed using the N closest points for six different N. The normals are color coded such that the blue, green and red channel represents the normal x, y and z component respectively for each point. For small N this method clearly results in poorly estimated surface normals with the setup used in this thesis, even for $N = 20$ as used in the original paper. However, if more points are used when computing the surface normals the performance is greatly enhanced. For $N = 50$ the method have a satisfactory result for close to all points in the accumulated point cloud. Should N be chosen too large, edges between distinct surfaces will have a smooth transition of the normal vector rather than a discrete transition as can be seen close to the leftmost wall in Figure 5.9f.

In the paper by Levinson and Thrun surface normals are computed after which the



(a) N = 3 **(b)** N = 5 **(c)** N = 9

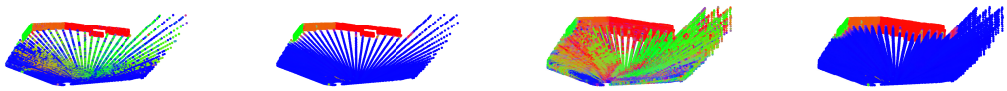
(d) N = 20 **(e)** N = 50 **(f)** N = 100

Figure 5.9: The surface normals are computed with Equation 4.18 using the N closest points. This is shown for different N where the blue, green and red channel represents the normal x, y and z component respectively for each point. With too small N surface normals are poorly estimated visualized by each segment not having a single common color. If N is chosen to large the border between two segments have a smooth transition in their normal vector rather than a discrete difference, as seen close to the left most wall in (f).

point cloud has been accumulated using only points seen by a single beam. This approach differs fundamentally from the one used in this thesis. Since the goal of the calibration algorithm is to compare the quality of the accumulated point cloud rather than the quality of the computed surface normals, another approach is presented that results in better estimated normals with the setup used. The proposed modification involves computing the normals separately for each lidar frame before accumulation and thereafter transforming them similarly to the position of the points. As seen in Figure 5.10 this greatly improves the quality of surface normals, especially when poor extrinsic poses are used to accumulate the point cloud. Accurately estimated surface normals are essential for the energy function to be accurate since it is computed by finding the distance to each point in the normal direction. An additional advantage of the method proposed is the great reduction of computing time. Since the surface normals only need to be computed once and then be transformed rather than computed once for every pose combination tested, the time for getting the normals are reduced substantially.

5.2.3 Compute energy score

To fully evaluate the energy score function, a large amount of extrinsic pose combinations would need to be compared. This is not only very time-consuming but additionally not trivial to understand the performance of the function. Therefore the function is first evaluated by altering a single extrinsic pose component, while



(a) CP, NPS **(b)** CP, NPF **(c)** IP, NPS **(d)** IP, NPF

Figure 5.10: Normals computation for both correct pose (CP) and incorrect pose (IP) when computing separately per scan line (NPS) and computing separately per frame (NPF). The blue, green and red channel represents the normal x, y and z component respectively for each point. Computing normals separately per scan line gives poorly estimated surface normals for both incorrect pose and correct pose as seen by the normal vector being inconsistent on each segment in (a) and (c) with the result being worse for the incorrect pose. As seen in (b) and (d) the normals are accurately estimated for both incorrect pose and correct pose when computing separately per frame.

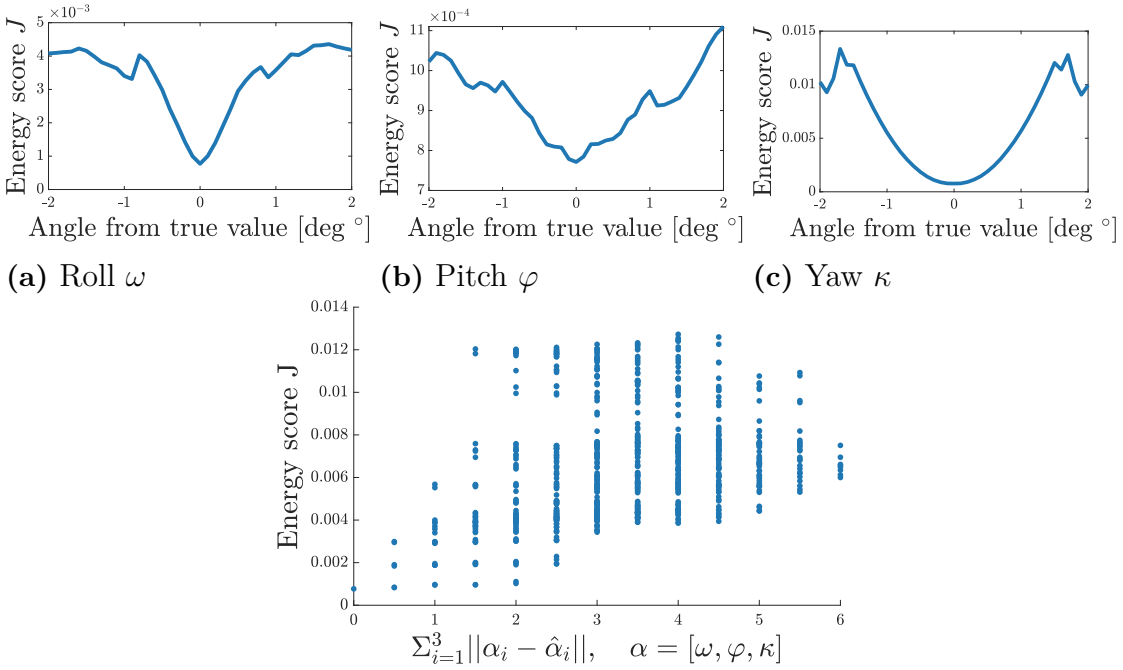
the other two components are kept constant at their true value. The result can be seen in Figure 5.11a-Figure 5.11c which shows that the energy function indeed has its global optima when the correct extrinsic pose is used.

The pose error clearly has a varying degree of impact on the energy score between the pose component and to verify that a lower total pose error yields a lower score this is visualized in Figure 5.11d. For every data point in the figure, the energy is plotted as a function over the total absolute error of each pose component, i.e. $\sum_{i=1}^{3} ||\alpha_i - \hat{\alpha}_i||$, where $\alpha = [\omega, \varphi, \kappa]$ and $\hat{\alpha}$ being the estimated pose component. The energy score is still decreasing with smaller error, however not as distinctly as in the case when only one pose component was varying at a time. Since the energy score does not decrease linearly with the total error, the calibration algorithm would get stuck at local optima very easily if only the best parameter settings were kept in each iteration. Thus an important note for performing the calibration algorithm is to keep a number of good parameter values in the grid search to make sure the global optima can be reached.

5.2.4 Calibration algorithm

When it comes to evaluating the actual calibration algorithm for lidar-to-IMU calibration synthetic data was created with the extrinsic poses in Table 5.1. For each pose combination, 10 test scenarios were created to validate that the algorithm is robust and gives similar results with slightly different data. Since the noise model used applies a random noise for each test scenario created, the data will be slightly different each time.

Every calibration was allowed to run for a maximum of 50 iterations and was prematurely terminated if none of the pose components had changed more than 0.001°for the best pose during the latest 5 iterations. For every iteration, the best 10 poses were kept, around each of which 4 new poses were created in the grid



(a) Roll ω **(b)** Pitch φ **(c)** Yaw κ

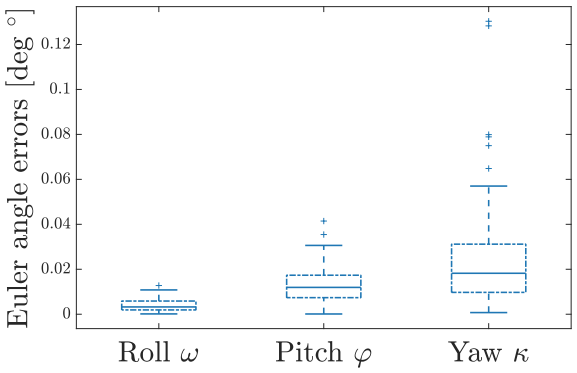
(d) Vary all three extrinsic pose components.

Figure 5.11: Top row shows how the energy score changes when one pose component is varying while the other two components are kept constant at their true value. The bottom row shows how the energy score changes when all three pose components are altered, the score is visualized as a function of the total pose error. Interesting to note is that when altering one parameter at a time (subfigures (a-c)), the energy function has a clear global (with respect to the search space) minimum for all variables. However, when varying all three at once (subfigure (d)), the minimum is not as obvious although there is some correlation between error and energy value.

search. When computing the energy function the point cloud was downsampled to only contain 100 000 points to save on computational time, as this should still leave a satisfactorily dense point cloud. The calibration errors for each pose component $\varepsilon = ||\text{true value} - \text{estimated value}||$ from every run are collected in Figure 5.12.

Clearly, the calibration error is not uniform over the three pose components, as the yaw error is almost 6 times as large as the roll error. Additionally, the standard deviation of the yaw error is about ten times as large as for the roll error. Furthermore, the yaw component has the most outliers as well, some of them having an error well above 0.1°. Since the yaw component of the pose clearly is the hardest to get right for the calibration algorithm, this is used to compute a confidence interval of the algorithm. By computing the 95% confidence interval of the yaw error and disregarding the lower limit, the algorithm gives an error not greater than 0.0283° with a 97.5% confidence.

As for the distribution of the calibration error, this follows the same pattern as for

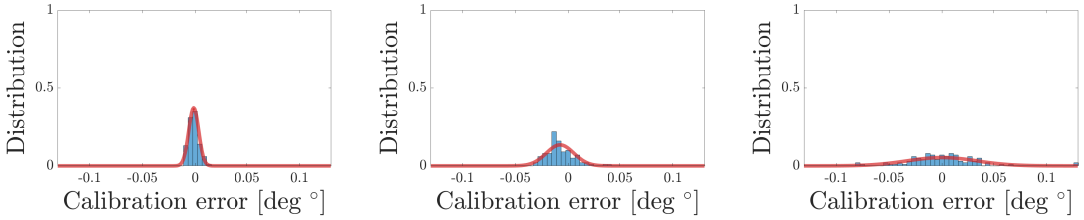


(a)

$\bar{\varepsilon}_\omega$	0.0040
$\bar{\varepsilon}_\varphi$	0.0127
$\bar{\varepsilon}_\kappa$	0.0238
σ_ω	0.0029
σ_φ	0.0081
σ_κ	0.0228

(b)

Figure 5.12: Performance of lidar-to-IMU calibration for all simulations. (a) shows the magnitude and deviation of the error. (b) shows the mean calibration error $\bar{\varepsilon}$ and standard deviation σ. It can be observed that the Roll error is both smaller in magnitude and standard deviation compared to the others and that Yaw is the largest, in both aspects. This is also reflected by the values in Figure 5.12b.



(a) Roll ω **(b)** Pitch φ **(c)** Yaw κ

Figure 5.13: The distribution of the calibration error is shown separately for each pose component. As indicated by the bell shape of the normal distribution curve in red, the errors are normally distributed closely around zero. There are differences between the three parameters, which could also be seen in Figure 5.12 where the errors in Roll are smaller both in amplitude and spread, while the errors grow in both amplitude and spread for Pitch and with Yaw having the largest errors with the biggest spread.

the magnitude of the errors as seen in Figure 5.13. The yaw component has a significantly larger spread in its distribution compared to the roll component. However, all three components seem to be normally distributed as the distribution follows the bell shape of its corresponding probability density function.

The simulations were made on a laptop equipped with an Intel Core i7-11850H processor, a NVIDIA RTX A3000 Laptop GPU and 32 GB of ram on which each calibration converged in about 25 minutes. The entirety of the calibration algorithm is written as a proof of concept in MATLAB without any major considerations about code optimization or parallel computing. Thus the run time could most likely be reduced significantly if some simple optimizations were made.

Table 5.2: Summary of performance for Target-Based Calibration (TB) using Cheat Targets (CT) and Real Targets (RT) as well as Lidar to Imu Calibration (LI).

[deg °]	TB CT	TB RT	LI
$\bar{\varepsilon}_\omega$	0.0000	0.2016	0.0040
$\bar{\varepsilon}_\varphi$	0.0000	0.5073	0.0127
$\bar{\varepsilon}_\kappa$	0.0000	0.2835	0.0238
σ_ω	0.0000	0.1924	0.0029
σ_ε	0.0000	0.7127	0.0081
σ_κ	0.0000	0.2119	0.0228

5.3 Analysis of results

Since the lidar-to-IMU calibration algorithm only estimates the extrinsic pose and not the offset in the current implementation, the following comparison of the two calibration methods will only be based on the precision of this estimation. The two algorithms have different prerequisites and each has their advantages and disadvantages. The most essential aspect when comparing them however is how well the extrinsic parameters can be estimated with the calibration algorithm, which is summarized in Table 5.2 where the mean error ε and standard deviation σ of the estimated pose components are presented.

Clearly, the lidar-to-IMU calibration has significantly smaller errors compared to the target-based calibration when real targets are used in the algorithm. Nonetheless, it has been proven that the target-based approach estimates the pose parameters with a near-zero error if the true target positions are found in the point cloud as seen in the table. Thus with an improved algorithm for target detection, the result could be very different. As previously discussed the errors for both methods follow a normal distribution with a near-zero center point. Not surprisingly the standard deviation follows the same pattern as the magnitude of the errors, as the lidar-to-IMU calibration has a significantly lower standard deviation compared to when real targets are used for the target-based method.

Despite both algorithms being purely proofs of concepts and little code optimization have been done for both cases, it is obvious that there is a rather large difference in run time. Since the target-based method computes the pose from estimated target positions, whereas the lidar-to-IMU method tries to estimate the pose by testing different pose parameters, the run time for target-based calibration is a fraction of the run time for lidar-to-IMU calibration. With a run time of about 2.5 seconds, the target-based algorithm is approximately 600 times faster than the lidar-to-IMU algorithm which has a run time of about 25 minutes. These values are not directly comparable since proper code optimization has not been done and parameters might not be set up for a completely fair comparison and it should be considered as a crude indicator of the difference.

As for the space-saving aspect, the lidar-to-IMU calibration clearly has the upper hand. It has been shown that the method has the potential for good performance with as few objects as the road and a few walls, using nothing but sensors already present in the vehicle. The target-based method however requires special equipment that must be constructed and supplied to every workshop that should be able to perform the calibration.

6
Discussion

The investigation performed in this thesis has given important knowledge about possible extrinsic calibration methods suitable for use in a workshop environment. However, there are still some areas that the work has not been able to fully cover. The planned setup of developing and testing the calibration algorithms was to set up a virtual environment in Unity where needed 3D objects could be imported. From this setup, realistic lidar readings can be recorded using a proprietary sensor package provided by the sensor supplier. Despite this tool being very useful in a development process it was not without problems to use it in this work.

The premise of the lidar-to-IMU calibration method investigated in this thesis is that the vehicle moves along a trajectory while recording its position and orientation with the GNSS/IMU sensor and a 3D point cloud with the lidar sensor. If the data from the IMU readings should be used to accumulate the point cloud from the lidar, it is an absolute necessity that both sensors can be accurately time synchronized. The intended test setup for developing the calibration algorithm does not have an automatic time synchronization between sensors but requires manual work to achieve this. This process is highly manual and prone to errors. A common procedure for time synchronization between sensors in this setup is to perform some exaggerated events at the beginning or at the end of the data log such as a hard acceleration or deceleration. The idea of this method is that this event should be easily detectable in the log of both sensors and can be used as the starting point for time synchronization. However, this method obviously depends on a having reliable and consistent frequency of data readings. Even if the distinct event can be accurately detected on both data logs, the accumulation of points will not be accurate over a longer time interval if the sensor frequencies are not consistent. The primary intended approach of data collection for the lidar-to-IMU calibration was to record a log with a GNSS/IMU sensor while driving a physical test vehicle to simulate realistic behaviour. This trajectory could then be imported into Unity where the lidar data could be recorded with full control over all six degrees of freedom for the sensor. The GNSS/IMU units used when recording the position and orientation of the vehicle had some odd behaviour in the orientation data which made the process of accumulating points infeasible and rendered the use of a physical test vehicle impractical for the development process. Despite not being fully investigated this behaviour seemed to be caused by drift in the GNSS/IMU sensor. The position and orientation accuracy decrease over time which usually is an indicator of drift, possibly caused by the fact that the test vehicle moved at low speed while recording data.

Without data from a physical sensor unit, this data had to be synthetically generated. Thus a trajectory of the vehicle movement was created in MATLAB to be imported into Unity where the lidar data was recorded. However, this process was not without issues either. The timestamps of the lidar data recorded in Unity proved to be inconsistent. The frequency set to be used when recording data was not the frequency in the frames received. Furthermore, the frequency was not even consistently incorrect but had different times between frames during the log, thus the IMU data was unusable for accumulating the point cloud frames. The solution used in this thesis is to not rely on the recording from Unity but to synthetically generate the lidar data as well. This is done by exporting a single frame of lidar data from Unity. For each data point in the IMU log this single frame of lidar data is then transformed with the IMU data such that it corresponds to how it would be seen from that position. Since the scan lines associated with each 3D point are essential to the calibration algorithm, they have to be changed each time the point cloud is transformed. This is done by dividing the field of view of the lidar sensor into sectors and assigning the points seen by a single sector to a common scan line. When the vehicle moves along the trajectory the points seen by each scan line will change over time, simulating the behaviour of the real sensor. Despite this solution being very helpful in the development of the calibration algorithm, it is rather limited. Since a single frame of 3D points is used in the generation, no new objects are seen along the trajectory. This is obviously not a realistic scenario if data from a physical test setup were deployed for data collection.

Since the development of the calibration algorithms is mainly done in the context of finding a proof of concept to the problem description, some aspects of the evaluation phase are somewhat limited. The test setup used does not utilize previously tested and validated platforms for AD & ADAS. Thus the process of collecting data is rather manual and slow compared to other methods utilizing such platforms where the data collection is better controlled and automated. Because of this, it is not feasible in the limited project time to collect a huge amount of data for testing and validation of performance and robustness. Furthermore, the nature of being a proof of concept implies that the algorithms are not optimized for speed. The code is written in a high-level interpreted language with more thought on interpretability than code optimization. Despite many of the computations being highly parallelizable GPU computing have not been utilized to a high degree in the current implementation. Considering the extensive run time this implies it is not feasible to perform high volumes of simulations in the limited project time, making any statistical result produced inadequate for a final verdict on the performance of the algorithms. This is particularly true for the lidar-to-IMU calibration since its run time is significantly longer than for the target-based calibration.

6.1 Discussion of results

To allow for a fair comparison between the two calibration methods some initial comments should first be made on the performance of each method. Since both data collection from physical sensors and synthetically generated sensor simulations

were used during the validation phase, not all results can be compared directly.

6.1.1 Target-based Calibration

The performance of the target-based method wasn't nearly as good as desired, with mean errors on the scale of tens of millimeters, with some extremes reaching an error of 160+ millimeters. Errors on the offset parameters are not the most important as rotation errors will contribute more over distance, however, the rotation errors are also unacceptably big. The main suspected reason for the bad performance is the sphere localization since when cheating that part of the toolchain and using perfectly estimated targets, the method performs flawlessly.

When estimating the targets, the found targets don't have the same position in relation to each other as the ground truths. This can be observed in Figure 5.3a where the red spheres are the transformed estimations and the green spheres are the ground truths. The red spheres are generally aligned with the green spheres, but looking closely, the red spheres are slightly misaligned with the green spheres, especially for the two lower spheres in Figure 5.3a. If the misalignments were in the same direction it would be reasonable to question the computation of the transformation, however, in this case, the misalignments are in different and almost opposite directions, while the two spheres in the top of Figure 5.3a are very well aligned. Since the method used is trying to fit a rigid body transform between the estimated body (comprised of the estimated centers) and the true body (comprised of the ground truth centers) the shape of the two bodies is important. If the estimated body has a different shape than the ground truth body, there might be an incorrect transformation that gives a better result than the true transformation despite being incorrect in terms of extrinsic calibration. Thus the relative position of the estimated spheres is the cause of the incorrect transformation, and not the transformation computation itself.

When looking into the spread in estimated centers, in Figure 5.7, the centers can differ 20 mm from two edge cases and if the method compensates for this by changing the transformation estimation it gives rise to errors in offset and rotation. The reason for the method not finding the right targets is not completely clear, but since it is based on RANSAC one reason could be the imperfect point distribution in the X dimension, see Figure 5.4. When only selecting four points from which to estimate the sphere, it is possible to select points all lying at the lower end of each column and thus get an estimation that might be offset from the true location. Another contributing factor could be that the simulation is very deterministic and points in different frames end up in the exact same position, meaning that accumulating more frames does not necessarily improve performance, while that might not be the case for real world data.

Even if the method performs perfectly when using perfect data, that tells very little about the robustness of the method, since even improving the localization method substantially would not provide perfect targets. However, with fewer spread-out estimations for each frame, the estimation on accumulated frames might become more

robust. The results presented are derived from clean simulated data, i.e. without any sort of noise. In reality, noise will play an important part in the performance of each method, but because of a lack of realistic noise models, and already poor performance from the target-based model, no noise model was implemented. It is suspected that introducing noise will only further decrease the performance of the model, and won't provide interesting results until other parts of the method have been improved.

Target-based calibration is sensitive to manufacturing and placement tolerances of the targets used, which isn't prevalent in the simulated data, where everything is perfectly located in relation to each other. This means that when using target-based calibrations there are always two more tolerances to account for compared to lidar-to-IMU calibration. The target itself will have some manufacturing tolerances from its own manufacturing process in addition to the tolerance or error from placing the target, either by itself or mounting it to the ego vehicle in any way. It's important to account for this when using non-ideal environments to collect data, i.e. in realistic or real environments.

6.1.2 Lidar-to-IMU Calibration

One of the largest contributing factors to the poor performance of the lidar-to-IMU calibration algorithm during the development phase was poorly estimated surface normals. Initially, the normals were computed using the 20 closest points seen by the same scan line after accumulation along the trajectory, similar to the work by Levinson and Thrun. As presented in Figure 5.10 this resulted in very poorly estimated normals, especially for points on the ground. This might be a consequence of the way the sensor scans the environment and the sparsity of the point cloud since road points tend to lie on distinct lines if the beams are transmitted at the same angles every scan.

The solution for estimating normals used in this work is to perform the computation separately per lidar frame. Since every full scan by itself produce a crisp representation of the environment, only limited by the accuracy of the sensor itself, normals can be accurately computed for that particular scan using commonly used algorithms such as least squares using neighboring points. That still leaves a big problem about how the normals should be transformed when multiple scans are accumulated over a trajectory though. Of course, one could simply apply the same transformation matrix used for transforming the position of the 3D points, though this will result in very poor estimations when extrinsic parameters far from the true values are used when accumulating the points. Although this method would theoretically give perfect normal vectors when correct extrinsic parameters are used, the calibration algorithm requires accurate surface normals even for poorly chosen extrinsic parameters for good performance.

Most locally planar surfaces in the static local environment for a vehicle could generally be described crudely as strictly horizontal or strictly vertical, be it a road,

sidewalks, buildings or traffic signs. Thus those surfaces will be most important to compute surface normals for. If we do not care about small variations in the z-component of the normal vector, the extrinsic roll and pitch components of the transformation could be omitted entirely resulting in the rotation purely taking place in the XY 2D plane. Even though this is not theoretically accurate, it performs well in practical applications where a small variance in the z-component of the normal vector is not of absolute importance.

Unfortunately, the process for collecting data from physical sensors used in this thesis did not provide accurate enough data, making it impossible to evaluate the performance of the calibration algorithm on real data. Possibly the best solution to solve this problem is to utilize an existing AD & ADAS platform for data collection that has already been tested and validated and better corresponds to the setup of the vehicles of interest. This would not only make sure accurate IMU reading can be recorded but additionally, such a setup would also completely omit the need for manual time synchronization. The platform is set up with the intention of sensor fusion being used and hence time synchronization has already been automated between sensors. The data collection that was made utilized the GNSS reading from the GNSS/IMU unit, thus another possible solution would be to investigate if less drift is caused when integrating the accelerations measured with the internal accelerometer of the GNSS/IMU unit.

Because no data from the test vehicle could be used the IMU data used for validation is synthetically generated with a simple model. The generated trajectory is without noise and has piece-wise constant velocity making it somewhat unrealistic. Despite the algorithm to accumulate the point cloud will work equally well when the velocity is non-constant and when the trajectory is not as smooth, as long as the position and orientation data are accurate, it should still be considered when evaluating the result of the calibration algorithm.

Furthermore, it should also be considered how realistic the lidar data is when evaluating the results. One could argue the data is equally realistic to what is used to evaluate the target-based calibration since a simulated scan from Unity is used for data generation. However, it is important to consider the fact that purely synthetic data has been used, leading to certain limitations. For example, the way the data is generated, using one single frame from one perspective, only one set of objects are observed and only from one perspective, contrary to a real world scenario.

The performance of the algorithms shows good potential with promising errors in pose estimation. As presented in Figure 5.11 the energy function has its optima for the correct pose parameters meaning it should be theoretically possible to find the true pose parameters if all possible combinations were to be tested in the algorithm. Even though it has not been fully validated, the algorithm tended to yield smaller errors when more points were used in the energy score. The results presented used 100000 points but also fewer points were used during the development process which did result in larger errors. This indicates that it could be of interest to run further

simulations where more points are used in the computation to validate whether better performance is achievable or not. Despite the method having promising results, it should be noted that the yaw component has the largest errors in the calibration, which is the most important parameter to accurately estimate since it has the highest impact on the sensor performance.

With the current implementation of the calibration algorithm, only the extrinsic pose components are estimated and not the translational misalignments. This was an intentional choice to keep the run time down when developing the algorithm since the pose parameters affect the sensor performance far more than the offset. However, the offset could easily be estimated in the algorithm as well by switching between optimizing for offset and pose to keep the number of parameters to test to a minimum. It should be noted that this would obviously increase the run time of the algorithm further.

6.1.3 Comparison of methods

Depending on what aspects are most important for the calibration different arguments could be made on which algorithm is best suited to solve the problem handled in this thesis. In this chapter, the performance of the two methods will be discussed in relation to each other, highlighting factors that affect or might alter the performance in the future.

One significant factor, relevant to all calibration methods, is precision in which lidar-to-IMU calibration currently outperforms target-based calibration. However, if a better algorithm for localizing the targets is found one might have to reconsider the final verdict since the target-based method have other advantages. Additionally, the precision presented for the lidar-to-IMU calibration is the final precision that could be expected from the method provided equally realistic data is used. This is not entirely true for the target-based method since the precision presented does not take into account the tolerances to which the target fixture is mounted or in relation to the vehicle. Even if accurate measures of mounting would be developed, some additional errors would be introduced decreasing the overall precision of the method.

The target-based method has a lot shorter run time if only the calibration algorithm is considered. However, the target fixture would need to be prepared and mounted to the vehicle before calibration which should be included in the total run time. As for the lidar-to-IMU calibration, the data collection could be somewhat time-consuming since the vehicle would have to be taken out from the workshop and driven along a trajectory. Because the algorithms both are executed offline the run time of them should in most cases not be too much of a restricting factor since no manual workload is required during this process and other tasks can be performed simultaneously.

Despite the target-based method being dependent on a target fixture, no additional requirements are set on the environment to make the algorithm work. In contrast, the lidar-to-IMU calibration requires some 3D object to be present in the environ-

ment from which the data is collected, even though the method made promising results with only the ground and a few walls being present.

6.2 Possible improvements and future work

Although the evaluation showed promising results on the lidar-to-IMU calibration some work is yet to be done before a final verdict can be made on the performance of the method. As of now, the calibration algorithm does only handle calibration of the extrinsic pose of the sensor and not its position. This would be rather simple to implement as the algorithm already handles translational offsets, but is currently set to neglect them during the grid search optimization. To minimize the number of parameter combinations to test in the grid search, this additional optimization is preferably implemented such that the algorithm alters between optimizing over pose and position.

The evaluation has been performed on a rather small batch of synthetic data without any major work on introducing realistic noise to the models. Thus further validation on larger batches of data should be performed with the setup expected in a workshop environment. Evaluation of physical hardware should be done to verify that the method performs satisfactorily even outside the controlled environment that is simulations.

Should the implementation of the target-based method be investigated further, the primary work effort should be made to improve the performance of the localization of targets. The biggest flaw of the current algorithm is that poor estimations are made when the localizations of the targets are inaccurate. However as shown in Table 5.2 the errors are practically non-existent if correct localization is found making the method a theoretically valid solution.

Future work should have a way of introducing better realistic noise to the synthetic data, either directly during simulation or infusing afterward. Implementing realistic noise will probably require a thorough study of the noise produced by the sensor and might be suitable for a thesis report in and of itself. Having a realistic noise model would be very useful for future development and simulation of the sensor, and would provide more interesting results.

7
Conclusion

Based on the evaluation performed in this thesis the potential of extrinsic calibration of a lidar sensor can be confirmed. Using a target-based approach has a simplicity that could be sought after and it has been one of the most widespread solutions applied in industry. The work in this paper shows that the method is highly accurate when the true position of the targets in the point cloud is known, however very sensitive to the localization of targets. It is thus reliant on very high accuracy when mounting the targets in relation to the vehicle in addition to the localization algorithm itself.

Using a lidar-to-IMU based approach utilizing the internal GNSS/IMU unit of the vehicle has a independency that could be sought after when spatial and monetary restrictions are in place. This method shows promising precision on the extrinsic pose of the lidar sensor as no pose component has an error larger than 0.0283° with a 97.5% confidence level in simulations. Furthermore, the calibration is performed without requiring any external sensors or additional equipment. The algorithm utilizes 3D objects or features in the environment for calibration and has shown high accuracy with only a road and a few walls being present. The run time of the algorithm can be somewhat extensive but the computation is performed offline and could be done simultaneously as other work is being done to the vehicle.

Between the two methods, there are both advantages and disadvantages for either method, with the largest being performance, where lidar-to-IMU calibration has been found to be the best-performing algorithm. Regarding run time, target-based calibration has a massive advantage, with one calibration run taking approximately 2.5 seconds for ten accumulated frames compared to approximately 25 minutes for lidar-to-IMU calibration. The requirements on the environment are significantly less for lidar-to-IMU calibration and practically negligible compared to target-based calibration, which in itself is a big advantage. Lidar-to-IMU calibration does not require any special tools or setup compared to target-based calibration, since the only requirement is that the vehicle should be driven in an environment containing three-dimensional features (i.e. walls or objects) and is thus cheaper and more accessible.

Bibliography

[1] A. Taeihagh and H. S. M. Lim, "Governing autonomous vehicles: Emerging responses for safety, liability, privacy, cybersecurity, and industry risks," *Transport reviews*, vol. 39, no. 1, pp. 103–128, 2019.

[2] J. J. Thomson, "Killing, Letting Die, and The Trolley Problem," *The Monist*, vol. 59, no. 2, pp. 204–217, Apr. 1976, ISSN: 0026-9662. DOI: 10.5840/monist197659224. eprint: https://academic.oup.com/monist/article-pdf/59/2/204/4172191/monist59-0204.pdf. [Online]. Available: https://doi.org/10.5840/monist197659224.

[3] M. Massar, I. Reza, S. M. Rahman, S. M. H. Abdullah, A. Jamal, and F. S. Al-Ismail, "Impacts of autonomous vehicles on greenhouse gas emissions—positive or negative?" *International Journal of Environmental Research and Public Health*, vol. 18, no. 11, p. 5567, 2021.

[4] Geneva: World Health Organization, "Global status report on road safety 2018," 2018. [Online]. Available: https://www.who.int/publications/i/item/9789241565684 (visited on 06/14/2023).

[5] National Highway Traffic Safety Administration, "NHTSA Enforcement Guidance Bulletin 2016–02: Safety-Related Defects and Automated Safety Technologies," *Federal Register*, vol. 81, no. 185, 2016. [Online]. Available: https://www.govinfo.gov/app/details/FR-2016-09-23.

[6] S. Oh, J.-H. You, A. Eskandarian, and Y.-K. Kim, "Accurate alignment inspection system for low-resolution automotive lidar," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11 961–11 968, 2021. DOI: 10.1109/JSEN.2020.3049091.

[7] H.-S. Song, J.-H. You, J.-E. Park, A. Eskandarian, and Y.-K. Kim, "Automatic inspection system for automotive lidar alignment using a cubic target," *IEEE Sensors Journal*, vol. 22, no. 3, pp. 2793–2801, Feb. 2022, ISSN: 1558-1748. DOI: 10.1109/JSEN.2021.3133008.

[8] J.-H. You, S. T. Oh, J.-E. Park, A. Eskandarian, and Y.-K. Kim, *Automatic lidar extrinsic calibration system using photodetector and planar board for large-scale applications*, 2020. DOI: 10.48550/ARXIV.2008.10542. [Online]. Available: https://arxiv.org/abs/2008.10542.

[9] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[11] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[12] V. Fremont, P. Bonnifait, *et al.*, "Extrinsic calibration between a multi-layer lidar and a camera," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, IEEE, 2008, pp. 214–219.

[13] J. Levinson and S. Thrun, "Unsupervised calibration for multi-beam lasers," in *Experimental Robotics: The 12th International Symposium on Experimental Robotics*, O. Khatib, V. Kumar, and G. Sukhatme, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 179–193, ISBN: 978-3-642-28572-1. DOI: 10.1007/978-3-642-28572-1_13. [Online]. Available: https://doi.org/10.1007/978-3-642-28572-1_13.

[14] S. Li, L. Wang, J. Li, B. Tian, L. Chen, and G. Li, "3d lidar/imu calibration based on continuous-time trajectory estimation in structured environments," *IEEE Access*, vol. 9, pp. 138 803–138 816, 2021. DOI: 10.1109/ACCESS.2021.3114618.

[15] Unity Technologies. "Unity." (2023), [Online]. Available: https://unity.com/ (visited on 06/07/2023).

[16] Y. Shi, G. Zhao, M. Wang, Y. Xu, and D. Zhu, "An algorithm for fitting sphere target of terrestrial lidar," *Sensors*, vol. 21, no. 22, p. 7546, Nov. 2021, ISSN: 1424-8220. DOI: 10.3390/s21227546. [Online]. Available: http://dx.doi.org/10.3390/s21227546.

[17] The MathWorks, Inc. "MATLAB." (2023), [Online]. Available: https://se.mathworks.com/products/matlab.html (visited on 06/07/2023).

[18] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *kdd*, vol. 96, 1996, pp. 226–231.

[19] 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, 1987. DOI: 10.1109/TPAMI.1987.4767965.

[20] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, no. 5, pp. 922–923, 1976. DOI: https://doi.org/10.1107/S0567739476001873. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1107/S0567739476001873. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1107/S0567739476001873.

[21] W. Kabsch, "A discussion of the solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 34, no. 5, pp. 827–828, 1978. DOI: https://doi.org/10.1107/S0567739478001680. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1107/S0567739478001680. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1107/S0567739478001680.

[22] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 3084–3091.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY