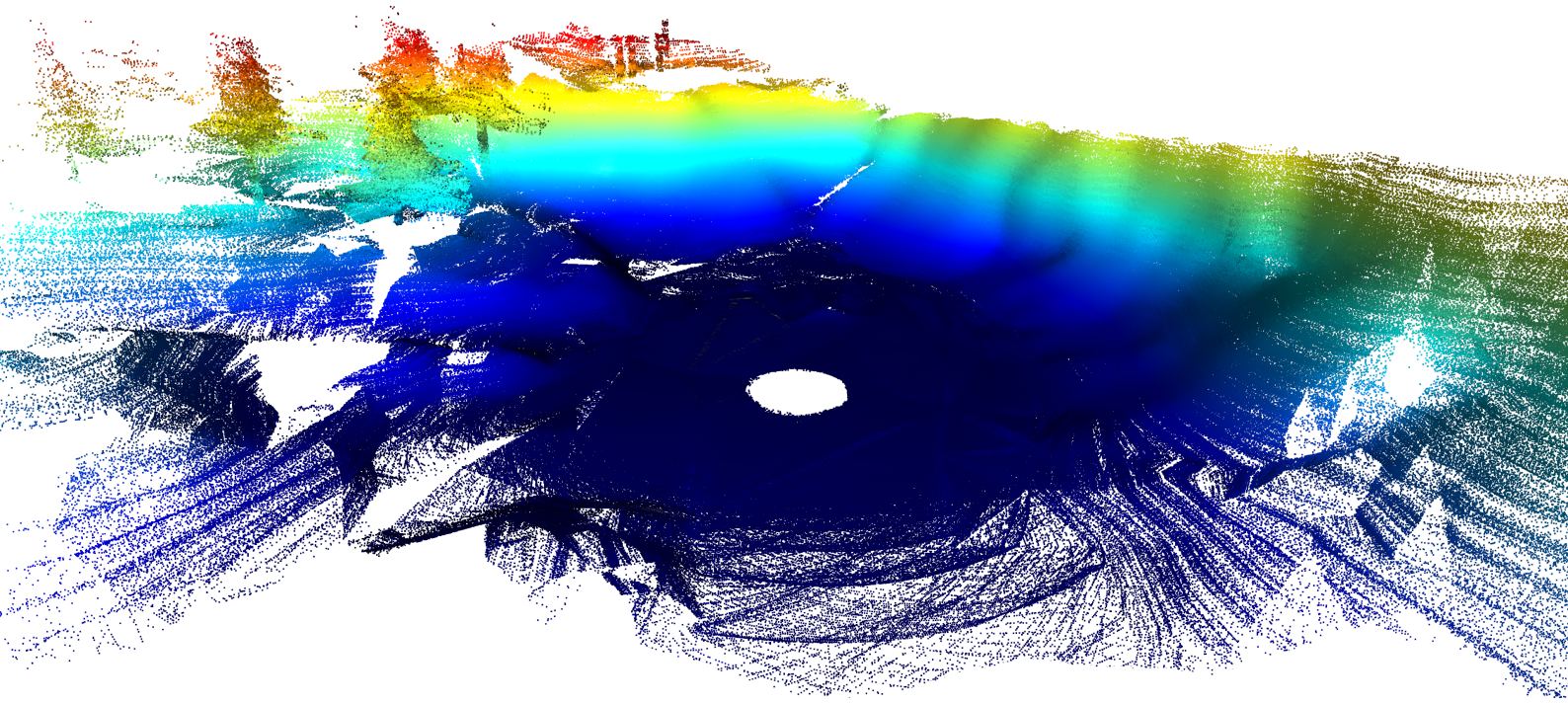


Extrinsic LiDAR Calibration for Autonomous Dumper

Master's thesis in Systems, Control and Mechatronics

ERIC DAHL

ANTON GUNNARSSON

Department of Electrical Engineering

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

Master's thesis 2023

Extrinsic LiDAR calibration for autonomous dumper

ERIC DAHL
ANTON GUNNARSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2023

Extrinsic LiDAR calibration for autonomous dumper
ERIC DAHL, ANTON GUNNARSSON

Supervisors: Anders Sunegård, Tomas Halleröd, Volvo Autonomous Solutions
Examiner: Lars Hammarstrand, Department of Electrical Engineering

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

Cover: LiDAR point cloud scan from one of the locations used in the thesis.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Extrinsic LiDAR calibration for autonomous dumper
ERIC DAHL, ANTON GUNNARSSON
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Autonomous driving has the potential to improve safety and efficiency, also in the mining industry. For this to happen, extrinsic calibration is a cornerstone to seamlessly and reliably fuse data from various sensors. For the autonomous dumper Volvo TA15 these sensors are a front and a rear LiDAR, whose fields-of-view are not intersecting. Fusing their measurements requires estimating the position and orientation (pose) of one LiDAR relative to the other; the pose can not be assumed static. Hence the need for automatic on-site extrinsic LiDAR calibration - a means to determine this pose - making use of existing and easily installed features.

This thesis presents two calibration algorithms: Daniliidis' State-Of-The-Art (SOTA) hand-eye problem solution and *MergeMaps*, a simpler yet more accurate and robust approach. Digital twins of the TA15 and a real quarry are used to test the algorithms over 200 calibrations in 25 scenarios, with the additional purpose of finding the key characteristics of good calibration environments. The 25 scenarios combine 5 quarry locations with 5 different landmark feature configurations (None, 5/10 boxes/cylinders), the latter 4 augmenting the existing environment.

Comparing the algorithms, *MergeMaps* produces 200 results with an average error of 0.27°/0.07m, never exceeding 0.98°/0.43m. Even that maximum error outperforms 75% of the hand-eye results, for which the average error is 8.16°/1.18m. The largest error is 146.84°/17.29m. The reason behind these huge errors is that the z-translation is unobservable for planar calibration trajectories, which should discourage using this algorithm for ground vehicles.

Adding landmarks reduces calibration errors by up to 45%/66.9% (rotations/translations) on average. The errors at the best location were 55.6%/64.5% lower than the worst. Moreover, boxes outperform cylinders by 25.7%/13.6%. Ten landmarks, compared to five, seem 19.1% better for rotations and 12.3% worse for translations. Advantageous environment characteristics include corners, edges, planar features, solid objects and large, proximate, irregular, near-vertical walls. Disadvantageous are vast, open and featureless areas.

Keywords: LiDAR, extrinsic calibration, quarry, hand-eye problem, ScLERP, ICP, sensor fusion, digital twin, landmarks, environment features.

Acknowledgements

We would like to thank our supervisors Anders Sunegård and Tomas Halleröd at Volvo Autonomous Solutions, whose help and support throughout the thesis have been invaluable. To Lars Hammarstrand, our examiner, we are thankful for all guidance along the way, academic advice and other life lessons learned.

Thanks to our peer reviewers Gustav Fåhraeus and Adam Thörnblom for valuable feedback. A special mention also goes to Oskar Wigström, Yufei Zhang, Hjalmar Egilsson and fellow thesis workers at CampX.

Finally, to our families and friends, it is thanks to you that we are able to write this. You are a never-ceasing source of inspiration and encouragement, and we are ever so grateful to have you by our side.

Eric Dahl, Anton Gunnarsson, Gothenburg, June 2023

List of Acronyms

CARLA Car Learning to Act
Open-source autonomous driving simulator. Objects such as vehicles sensors and environmental objects can be spawned in the world map. Light Detection and Ranging sensor (LiDAR) sensors are included by default. The software can also record when driving a route and save the data to a file for later use [1].

DOF degrees of freedom

FoV field-of-view

ICP Iterative Closest Point
Point cloud registration (alignment) algorithm. See 2.3.1.

IMU Inertial Measurement Unit
Sensor comprising accelerometers and gyroscopes. Measures acceleration and angular velocity. Some include a magnetometer.

LiDAR Light Detection and Ranging sensor
Sensor using laser beams to measure distance in a multitude of directions at a very high rate - resulting in point clouds - where each point represents such a measurement. Rotating LiDARs are configured so that a number of lasers are mounted on top of one another at slightly different angles. This creates channels of distance measurements as the lasers are rotated.

RANSAC Random Sampling And Consensus

RMSE Root Mean Square Error

ScLERP Screw Linear Interpolation

SVD Singular Value Decomposition

V.A.S Volvo Autonomous Solutions

Contents

List of Figures

List of Tables

List of Algorithms

1
Introduction

Automation and electrification are becoming increasingly popular in various fields and applications as businesses are trying to reduce environmental impact, increase profitability and remove humans from hazardous work environments. The mining industry is not an exception [2]. Volvo Autonomous Solutions (V.A.S) is currently developing a new autonomous, electric dumper (Volvo TA15) for the mining industry. Operating in quarries and mines, the TA15 is one step toward a more sustainable future within the industry. The dumper is supposed to operate autonomously, relying on the sensors mounted on the vehicle for localisation and perception of the environment. Thus it is equipped with a suite of sensors, including two LiDAR sensors - one at the front and one at the rear of the vehicle.

The relative pose (difference in position and orientation) of the two LiDARs has to be quite accurate to give acceptable measurements with regard to performance and safety. A tiny error in rotation may scale to large measurement errors for objects far away, and too large translation errors could prevent docking and autonomous charging. This pose between the LiDARs is by rule not static over time. Incidents such as mounts loosening due to vibrations, falling rocks and other unforeseen external events are the culprits. Neglecting this will sooner or later suppress the localisation capabilities and autonomy of the vehicle and could be dangerous if pursued extensively. Hence the position and orientation (pose) of the two sensors must be calibrated, and this is what is called extrinsic calibration.

Extrinsic calibration is usually performed at the end of the production line, and thereafter on behalf of the owner. This thesis will focus on the latter, which is often expensive and time-consuming for the owner and could therefore be neglected. The ability to easily re-calibrate the sensors on-site is therefore greatly appreciated. V.A.S therefore seek a way to calibrate the LiDARs on-site without the need for trained personnel. One enabler of this achievement is to know how to utilise the environment in the best possible way.

1.1 Purpose

The purpose of this thesis project is two-fold. Firstly, algorithms will be developed, able to estimate the transformation - the relative pose - between the two LiDARs of the TA15 in a simulation environment. Secondly, the same algorithms will be used

to investigate how simulated real-world environments may be augmented to improve pose estimation accuracy. This means placing various objects in the environment to see if that can help the algorithm to perform better. Here it is of interest to see what size, shape and amount of objects work best for the algorithm.

A good calibration algorithm would be able to output a sufficient pose estimate in any environment that a TA15 could be expected to encounter. A sufficient estimate here refers to the accuracy requested by V.A.S: errors lower than ± 5 cm (translation) and ± 0.2 degrees (rotation) are desirable but < 1 degree is acceptable. The rotational component is important while the translation component is secondary. The algorithm would also ideally be able to run without any preparations or altering of the surroundings.

The transformation estimate produced by the algorithm is highly dependent on the surroundings, naturally. To achieve high calibration accuracy, it is important to understand how different environmental factors affect the results. Using this knowledge, it will be possible to exploit the environments where the TA15 operates to their full potential. A calibration site may then be chosen more wisely and systematically, with benefits (direct and further down the line) such as:

- Improved calibration results
- Reduced need for environment augmentation
- Less calibration preparation
- Improved vehicle performance
- Improved vehicle safety
- Lower set-up times on new sites, and during calibration
- Lower calibration costs

It is therefore of interest to identify environments where a sufficient calibration may be performed, as well as what features in such environments contribute to that property.

1.2 Naming and coordinate frame convention

The result of the calibration is a transformation - a frequently recurring concept from here on. Hence, the following naming convention will be employed throughout the thesis, to systematically describe transformations between different frames of reference:

$$T_{\{Target\ frame\}\{Source\ frame\}}, \tag{1.1}$$

where T indicates a transformation matrix representation. The T is replaced by a $\check{\mathbf{q}}$ for dual quaternions. All frames of reference are defined as right-handed, Cartesian coordinate systems with the Euler angles (roll, pitch, yaw) extrinsic rotations on their respective axis. Both transformation matrices and dual quaternions are

thoroughly explained in the Theory section, including how source and target frames of reference translate to transformation matrices. The primary frames of reference used are front $\{F\}$ LiDAR, rear $\{R\}$ LiDAR, vehicle base frame $\{B\}$ and a static point in the world $\{W\}$, all displayed in Figure 1.1. The orientation of the frames fixed to the TA15 is further detailed in Figure 1.2. Additionally, these reference frames may carry subscripts, explained in Table 1.1.

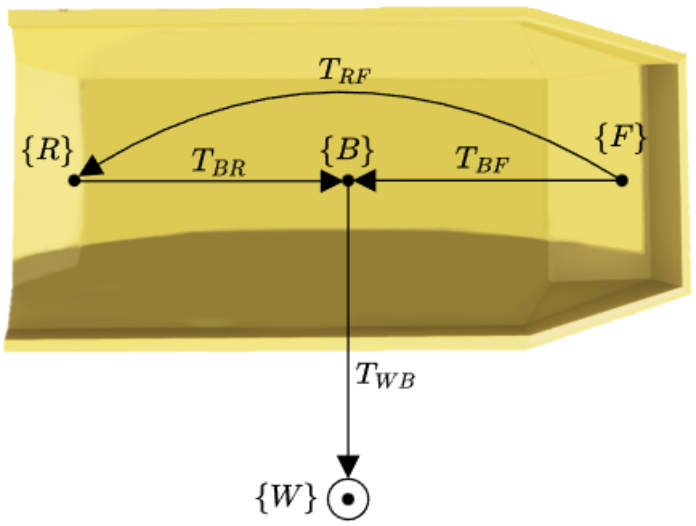


Figure 1.1: Top view of the dumper with the frames of reference used. The arrows represent transformations. Notice how the frame notation goes into the transformation subscripts, meant to be read right-to-left.

Subscript	Explanation
+	When two vehicle poses are present, denotes the latest sampled pose
0	The first sampled pose, pose 0
n	Pose n, in accordance with the previous row

Table 1.1: Subscripts used to denote the frames of reference located on the TA15. For example, T_{FF_+} denotes the transformation from the latest sampled front LiDAR pose to the preceding front LiDAR pose.

1.3 Research questions

Based on the purpose of the project, the following research questions have been formulated. The focus will be on the first question and on developing algorithms able to reliably output sufficient pose estimations in as many cases as possible without environment augmentation. The remaining questions will be addressed based on the results provided by these algorithms.

1. How accurately can the transformation T_{RF} (see Figure 1.2) relating the LiDARs of the TA15 be determined in different quarry environments?
   - Evaluation is carried out in 5 quarry locations, with and without inserted objects differing in amount and shape.
2. What difference in accuracy is obtained if using artificial landmarks to augment the environment?
   - As compared to the same environment without augmentation.
3. What environmental key characteristics, if any, render such environmental augmentation redundant in achieving sufficient accuracy?
   - Sufficient accuracy is defined in cooperation with V.A.S as ±(5 cm, 5 cm, 5 cm) in translation and ±(0.2°, 0.2°, 0.2°) in rotation in the (x,y,z-direction)

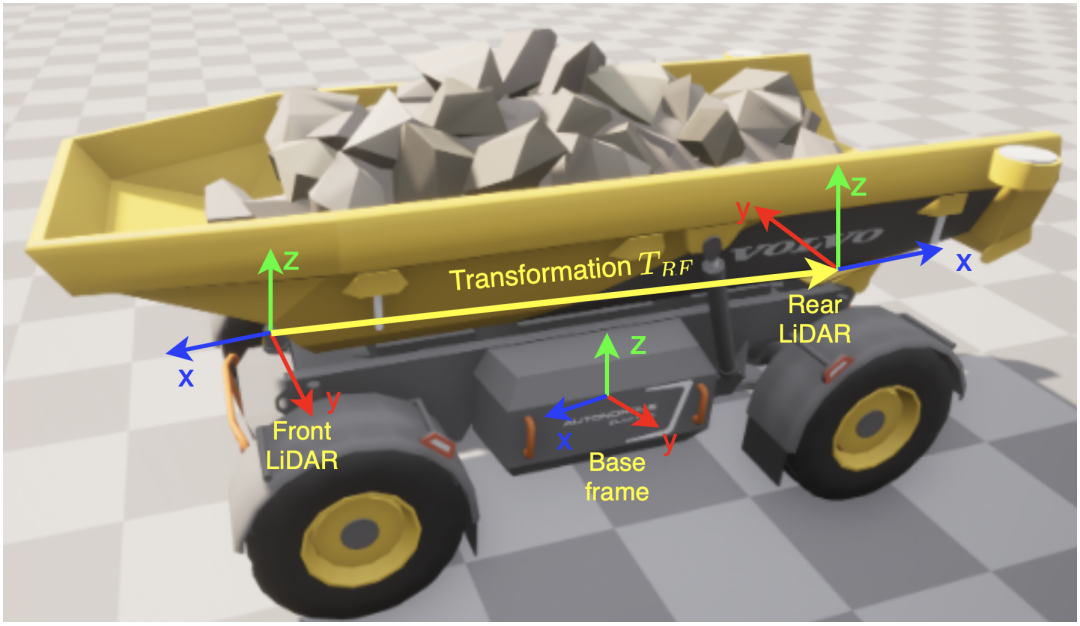


Figure 1.2: The LiDAR sensors are positioned beneath the dumper bed. The sought transformation goes from the coordinate frame of the front LiDAR to that of the rear LiDAR. To clarify, the transformation describes both the difference in translation between the two LiDAR sensors, as well as the difference in rotation between them. In the figure, the base frame is included as well for future reference

1.4 Problem formulation

The problem at hand is to develop an algorithm estimating T_{RF}, then use it to find environment characteristics that significantly affect the estimation accuracy. The inputs are LiDAR measurements and an odometry estimate based on sensor fusion. These will be further detailed, followed by how the problem is scoped.

The TA15 is equipped with two OS1-32 [3] LiDARs. Each has 32 vertical channels, a vertical field-of-view (FoV) of 45° (centred at the horizon) and a horizontal FoV of 180° (or less at some vertical angles). Thus, their FoVs are disjoint - not overlapping one another. The LiDARs rotate at a rate of 10 Hz, meaning that the point cloud sample rate is 0.1 seconds. With each channel producing 1024 points per revolution, one point cloud sample consists of (at most) 32768 distance measurements.

Each distance measurement - point in the point cloud - contains 4 values: x, y, z and i. The last value is the intensity, a value in the range [0, 1] representing signal strength. In simple terms, a higher value means a more certain reading. The first three are xyz-coordinates with respect to the LiDAR frame of reference ({F} or {R}).

The odometry estimate is provided as the transformation from the base frame {B} to a fixed point {W} in the world. In other words, the transformation T_{WB} is given. The estimate developed at V.A.S is an extended Kalman filter, which fuses data from onboard sensors such as GPS, Inertial Measurement Units (IMUs) and wheel encoders. The sample rate is 20 Hz, which is double that of the LiDARs. In this estimate, the front LiDAR and the rear LiDAR all sample asynchronously.

The *nominal* - in this context meaning intended - LiDAR poses relative to the vehicle midpoint and to each other can be viewed in Table 1.2. The rightmost column represents the front-to-rear LiDAR relation. This is also displayed in Figure 1.2. The robustness of the algorithm may be tested by intentionally configuring the LiDARs so that they are not in their nominal pose, seeing if the algorithm performs similarly despite the change in conditions.

Table 1.2: Nominal translations (t) and rotations (θ) for the LiDARs based on CAD model of the TA15. The first row represents the associated transformation.

	$T_{BF}^{nominal}$	$T_{BR}^{nominal}$	$T_{RF}^{nominal}$
$t_x\,[m]$	1.978	-1.958	-3.936
$t_y\,[m]$	0	0	0
$t_z\,[m]$	1.18	1.18	0
$\theta_x\,[^\circ]$	0	0	0
$\theta_y\,[^\circ]$	0	0	0
$\theta_z\,[^\circ]$	0	180	180

1.4.1 Scope

Parts of the thesis scope are defined as a result of the TA15 design, work already conducted and choices already made by V.A.S. Other assumptions and demarcations are results of requests from V.A.S, limitations in time and practicality reasons.

- The algorithm is not required to run in real-time; measurements can be made and then processed afterwards with no strict requirements with regard to time.
- The calibration procedure should not require any specialised personnel.
- The research will be conducted using data acquired from simulations in CARLA. This will enable comparison with ground truth and make for consistency and reproducibility of results.
- Data input provided to the algorithm will be limited to point cloud data acquired from a simulated OS1-32 LiDAR and the fused onboard odometry estimate of the TA15.
- The simulation is assumed to closely represent the real world.
- The simulated surroundings used for testing the algorithm will be based on real quarry scenes to which artificial landmarks will be added to create a more textured environment.
- Environment augmentation will be tested with 1-3 object types, each in 1-3 configurations.
- The minimum distance between the landmarks and the vehicle is not allowed to be less than 2 meters.
- The type of landmarks available is limited to the standard CARLA and V.A.S blueprint libraries.
- Preferably, the landmark configuration is feasible to reproduce in a real quarry.
- Preferably, the hand-eye calibration method is used as a starting point.

1.5 Related work

Previous work within the field of extrinsic LiDAR calibration has employed various methods and algorithms to solve similar calibration problems [4–7]. In [4], methods to find the extrinsic parameters for calibration are categorised as appearance-based methods, SLAM-based methods, or motion-based methods. However, only the last two methods seem to be used for finding the extrinsic parameters for non-overlapping FoV problems. One common motion-based method is to solve the "hand-eye problem" proposed by Tsai and Lenz [8]. The dual quaternion formulation by Daniilidis and Bayro-Corrochano [9] later improved the solution to the problem often also denoted as $AX = XB$. To solve the hand-eye problem for X, there exist different methods as described in [10]. However, using mentioned dual quaternion formulation seems to be the best starting point in this given case. This conclusion is supported in the literature [4, 6, 7, 11, 12]. For example, in [11], Daniliidis' solution is used as a baseline. There have also been papers that build on his work making small modifications and possible improvements, such as [12].

There are a wide variety of methods proposed to improve the localisation and perception capabilities of a robot or vehicle in various environments. Techniques such as [13] rely on matching geometric features in the environment to a map. This technique is interesting as it does not require modification of the environment. On the other hand, it is likely to fail for more complex environments. Other techniques have been suggested such as using artificial landmarks in the environment to improve perception and localisation. Often a marker is used in conjunction with a feature extraction algorithm [14]. These methods show good results but require advanced algorithms for extraction. Some have extended the technique of using artificial landmark extraction to also include reflexive markers to take advantage of the intensity metric associated with each point in a LiDAR point cloud. This is done in [15] and [16], among others. The former uses a cylinder covered in a reflexive membrane to improve precision. This requires a special material to construct the markers, however. The method used in [17] tries to simplify by using a selection of indistinguishable landmarks to decrease the overall ambiguity which seems promising.

1.6 Contributions

Implementation and evaluation of the hand-eye algorithm are performed for LiDAR calibration in a quarry environment, specifically with non-overlapping field of view LiDAR sensors mounted on a ground vehicle. Then a simpler method is tested for the same setup, the method aligns two large point clouds, one from each sensor that is first built up from multiple LiDAR scans.

Various locations in the quarry are then evaluated using the algorithms to find features in the environment that improve the accuracy of the extrinsic calibration parameters. Large landmark objects that stand out from the environment are also populated in the environment to evaluate if the algorithms benefit from this. Particularly objects with simple and distinct geometry compared to the rough and unstructured background environment. Flat surfaces and sharp corners of a box and curved shapes of a cylinder are properties that are of interest. These objects are placed out in the environment to see if point cloud alignment is improved and thus a better calibration result is obtained for this particular application.

1.7 Ethical and sustainability considerations

This thesis will contribute to the large-scale contemporary movements of automation and electrification, this is important to keep in mind. The work conducted within the boundaries of this thesis will primarily contribute to making the autonomous systems of TA15 more reliable by enabling more accurate measurements. People and property will hence be safer around the TA15. This is the small-scale picture of the ethical and sustainability aspects.

If taking one step back and looking at the entire mining and construction sector, the implications of the TA15 entering the market are safety benefits, possible short-term obsoletion of the workforce, increased efficiency and environmental benefits. Working in a mine or quarry is hazardous and with automation, fewer people will have to be exposed to this. On the other hand, the same people still have to make a living. Making the argument that automation makes people obsolete does not hold. Surely, the job position may be obsoleted but not the person holding the position. The history of automation proves that people adapt and new job opportunities always arise, with better working conditions than the former. Since this means long-term improvements for the workforce, on the whole, this would translate to social sustainability. Almost inarguably, automated and electrified vehicles are more efficient than their predecessors. From the point of ecological sustainability, both energy efficiency and the carbon intensity of the fuel contribute to a smaller carbon footprint. Moreover, time efficiency, cost of fuel and operation consistency contribute to economic sustainability.

One must nevertheless step back further to see the entire picture of how automation and electrification affect society as a whole. Looking further than the improvements within the mining and construction sector, one should be aware of possible implications within other industries. The characteristics described in the previous section - safety benefits, short-term obsoletion of the workforce, increased efficiency and environmental benefits - all hold when looking upon most industries. By contributing to the trends of automation and electrification, one should be aware that the research results may be used within any domain. Besides the immensely increased electricity demand resulting from the complete electrification of society, no huge societal ethical questions come to mind. When it comes to automation, there is one such significant ethical question that should be raised - automation that could be used for warfare. Applying this to the proposed thesis and the TA15, while there may be some connection, it is relatively vague.

2
Theory

In the following section, the theory used in the thesis is explained in more detail. First, motion in space using transformation matrices and dual quaternions is presented. This is followed by a guide to Iterative Closest Point (ICP)-based registration algorithms which serve the purpose of finding the transformation that best aligns two point clouds. Finally, the hand-eye problem will be posed building upon the two previous concepts.

2.1 Transformations in $\mathbb{R}^3$

Motion in 3D space can be described using transformations. A transformation is here defined as a rotation *followed by* a translation. Transformations are commonly described mathematically using transformation matrices. This thesis also uses dual quaternions, a more efficient representation which builds upon quaternions and dual number theory.

2.1.1 Rotation and transformation matrices

A rotation in $\mathbb{R}^3$ can be described by a 3x3 matrix $R \in SO(3)$ where rotations about the x (roll), y (pitch) and z (yaw) axes can be combined. $SO(3)$ denotes the special orthogonal group, the set of all possible rotations which can be performed in 3D-space, which is also all possible rotations which can be obtained by combining the rotations along the axes displayed in (2.1) below:

$$\overbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix}}^{\mathbf{R}_x(\theta_x)} \quad \overbrace{\begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix}}^{\mathbf{R}_y(\theta_y)} \quad \overbrace{\begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{\mathbf{R}_z(\theta_z)}. \tag{2.1}$$

About the x-axis by θ_x | About the y-axis by θ_y | About the z-axis by θ_z

Note that since matrix multiplication is non-commutative, the order of operations matters - $R_x R_y \neq R_y R_x$. In other words, rotating an object around its x-axis and then around its y-axis generally produces a different orientation than if performing the same rotations in reversed order. The object may as well be a single point.

Equation (2.2) showcases a point p being rotated from coordinate system A to coordinate system B. Notice (reading right-to-left, by matrix multiplication customs) how the subscripts align nicely, reading "p viewed in A, rotated from A to B, equals p viewed in B",

$$\mathbf{p}_B = \mathbf{R}_{BA}\mathbf{p}_A = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} p_A^x \\ p_A^y \\ p_A^z \end{bmatrix}. \tag{2.2}$$

Homogeneous coordinates can be used to extend the rotation matrix to also include translation. Simply explained, homogenous coordinates add a scaling factor k to points. For example, $\mathbf{x} = [x, y]^T \in \mathbb{R}^2$ would in homogeneous coordinates be $\mathbf{x} = [kx, ky, k]^T \in \mathbb{P}^2$, most frequently used in its normalized form $\mathbf{x} = [x, y, 1]^T$. More formally, the projective space $\mathbb{P}^n$ can be defined as $\mathbb{R}^{n+1}/\mathbf{0}$ where $\mathbf{0}$ represents the origin [18].

Translation can now be appended to the 3x3 rotation matrix to form a 3x4 matrix. This matrix will take a homogeneous coordinate as input and output the transformed point as a Euclidean coordinate. It is reasonable to construct the matrix so that transforming a homogeneous coordinate gives a homogeneous coordinate. Adding a row $[0, 0, 0, 1]$ at the bottom of the matrix gives these properties and we have a 4x4 transformation matrix $T \in SE(3)$;

$$T = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right] = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]. \tag{2.3}$$

The transformation matrix (within Computer Vision often called extrinsic camera matrix) may be broken down into rotation and translation separately [19]:

$$\left[\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right] = \left[\begin{array}{c|c} \mathbf{I}_3 & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right] \left[\begin{array}{c|c} \mathbf{R} & \mathbf{0} \\ \hline \mathbf{0} & 1 \end{array}\right]. \tag{2.4}$$

A point may be transformed in the same way as (2.2) rotated a point. It is however important to pay attention to the 2 ways to define a transformation: "camera to world" and its inverse, "world to camera". Imagine an empty world where only the world origin is defined (in a cartesian coordinate system), as well as a point p. Furthermore, imagine placing a "camera" L (for LiDAR) anywhere in this world using the Euler angles $(x, y, z, \theta_x, \theta_y, \theta_z)$ to describe its 6 degrees of freedom (DOF) in relation to the origin. Then, we may look at the point from either the camera frame of reference or the world frame of reference. In the same way as (2.2) we may transform the point as seen from the camera coordinate system to the world coordinate system using T_{WL}:

$$\mathbf{p}_W = \mathbf{T}_{WL}\mathbf{p}_L = \left[\begin{array}{ccc|c} & \mathbf{R}(\theta_x, \theta_y, \theta_z) & & \begin{matrix} x \\ y \\ z \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_L^x \\ p_L^y \\ p_L^z \\ 1 \end{bmatrix}. \tag{2.5}$$

Representation in terms of Euler angles can be more tangible since the angles are expressed clearly, yet possibly ambiguous. The rotation matrix $\mathrm{R}(\theta_x, \theta_y, \theta_z)$ may differ depending on the order of rotation about the axes (remembering $R_x R_y \neq R_y R_x$). This thesis uses the convention shown in (2.6). The order of rotations then becomes about the x, y, and then z-axis of the initial, fixed, coordinate frame. In other words, roll first, then pitch and lastly yaw,

$$R(\theta_x, \theta_y, \theta_z) = R_z(\theta_z) R_y(\theta_y) R_x(\theta_x). \tag{2.6}$$

The ambiguity can also be resolved in a simpler way by introducing yet another concept: dual quaternions. They also provide a more compact representation of transformations. To understand what a dual quaternion is, it is useful to first introduce the concepts that it builds upon: quaternions and dual numbers.

2.1.2 Quaternions

A quaternion q is a hypercomplex number, an extension of complex numbers, with one real part s and *three* imaginary parts (xi, yj, zk) - an imaginary vector $\mathbf{v}$. Similar to how complex numbers can be used to describe rotations in $\mathbb{R}^2$, quaternions efficiently represent rotations in $\mathbb{R}^3$. Discovered by Hamilton [20, 21] in 1843, a quaternion q can be defined as follows [22]:

$$\text{Hamilton's original notation: } \mathsf{q} = s + xi + yj + zk \tag{2.7}$$

$$\text{Ordered-pair notation: } \mathsf{q} = (s, \mathbf{v}) \qquad \mathbf{v} = [x, y, z]^T \tag{2.8}$$

$$\mathsf{q} \in \mathbb{H} \qquad s, x, y, z \in \mathbb{R},$$

where $\mathbb{H}$ is the set of all quaternions,

$$\mathbb{H} = \left\{ \; \mathsf{q} = s + xi + yj + zk \;\middle|\; s, x, y, z \in \mathbb{R}, \quad i^2 = j^2 = k^2 = ijk = -kji = -1 \; \right\}. \tag{2.9}$$

The imaginary units (i,j,k) all have the familiar property $i^2 = -1$, as well as some additional ones [22]:

$$i^2 = j^2 = k^2 = ijk = -1$$

$$\begin{array}{lll} ij = k & jk = i & ki = j \\ ji = -k & kj = -i & ik = -j \end{array} \tag{2.10}$$

$$\begin{aligned}
\text{Scalar multiplication: } & \lambda\mathsf{q} = (\lambda s, \lambda \mathbf{v}) \qquad \lambda \in \mathbb{R} \\
\text{Quaternion product: } & \mathsf{q}_1\mathsf{q}_2 = (s_1 s_2 - \mathbf{v}_1^T \mathbf{v}_2, \quad s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2) \\
\text{Quaternion conjugate: } & \bar{\mathsf{q}} = (s, -\mathbf{v}) \\
\text{Quaternion norm: } & \|\mathsf{q}\| = \sqrt{\bar{\mathsf{q}}\mathsf{q}} \\
\text{Quaternion inverse: } & \mathsf{q}^{-1} = \frac{\bar{\mathsf{q}}}{\|\mathsf{q}\|^2} \\
\text{Unit quaternion: } & \mathsf{q} \in \left\{ \; \mathsf{q} \in \mathbb{H} \;\middle|\; \|\mathsf{q}\| = 1 \; \right\} \\
\text{Pure quaternion: } & \mathsf{q} \in \mathbb{H}_p = \left\{ \; \mathsf{q} \in \mathbb{H} \;\middle|\; Re(\mathsf{q}) = 0 \; \right\}.
\end{aligned}$$

Analogous to rotation matrices, quaternions are non-commutative - the order of multiplication matters. Comparing the lower equations in (2.10), this becomes obvious. Also take notice of that notation may differ; (s, x, y, z) is often replaced for (a, b, c, d) [22] and $\mathbf{v}$ may be called $\vec{\mathsf{q}}$ [23].

2.1.3 Dual numbers

A dual number $\check{z}$ is another hypercomplex number with two parts. Where normal complex numbers have a real and an imaginary part, dual numbers instead have a real (a) and a *dual* (b) part:

$$\begin{aligned} \check{z} &= a + \epsilon b & a, b &\in \mathbb{R} \\ \check{z} &\in \Delta & \epsilon^2 &= 0 \\ & & \epsilon &\neq 0, \end{aligned} \tag{2.11}$$

where Δ^n is the set of all dual numbers of order n:

$$\Delta^n = \left\{ \; \check{z} = a + \epsilon b \;\middle|\; a, b \in \mathbb{R}^n, \quad \epsilon^2 = 0, \quad \epsilon \neq 0, \quad n \in \mathbb{N} \; \right\}. \tag{2.12}$$

Dual numbers differ from normal complex numbers solely by the properties of the dual unit ϵ. The same set of rules that apply to complex numbers apply to dual numbers. For multiplication and division, the procedure is the same - albeit the result differs due to the definition of ϵ. This also applies to more complex operations, such as dot and cross products.

$$\begin{aligned} \text{Dual conjugate:} \quad & \check{z}^* = a - \epsilon b \\ \text{Multiplication:} \quad & \check{z}_1 \check{z}_2 = a_1 a_2 + \epsilon(a_1 b_2 + a_2 b_1) \\ \text{Division:} \quad & \frac{\check{z}_1}{\check{z}_2} = \frac{a_1}{a_2} + \epsilon \frac{b_1 a_2 - a_1 b_2}{a_2^2} \qquad a_2 \neq 0. \end{aligned}$$

Functions taking a dual number as input make sense and may be broken down into functions of real variables. The Taylor expansion of the function evaluated around its real part is finite since $\epsilon^k = 0 \quad \forall k \geq 2$. This makes it possible to evaluate not only the trigonometric functions but any function that takes a dual number:

$$\begin{aligned} \text{Taylor expansion:} \quad f(a + \epsilon b) &= f(a) + \frac{f'(a)}{1!}\epsilon b + \frac{f''(a)}{2!}(\epsilon b)^2 + \cdots \\ f(a + \epsilon b) &= f(a) + \epsilon b f'(a) \end{aligned} \tag{2.13}$$

$$\begin{aligned} \text{Sine:} \quad & \sin(a + \epsilon b) = \sin(a) + \epsilon b \cos(a) \\ \text{Cosine:} \quad & \cos(a + \epsilon b) = \cos(a) - \epsilon b \sin(a) \\ \text{Tangent:} \quad & \tan(a + \epsilon b) = \tan(a) + \epsilon b \sec^2(a). \end{aligned}$$

Introduced by Clifford [24] in 1873, dual vectors in Δ^3 have shown useful for describing lines in space using Plücker coordinates. A line in Plücker coordinates is defined as

$$\begin{aligned} \check{\mathbf{l}} &= \mathbf{l} + \epsilon \mathbf{m} & \mathbf{l} \cdot \mathbf{m} &= 0 \\ \check{\mathbf{l}} &\in \Delta^3. \end{aligned} \tag{2.14}$$

The dot product of two lines represented in Plücker coordinates carries an interesting property,

$$\begin{aligned}
\check{\mathbf{l}}_a \cdot \check{\mathbf{l}}_b &= \begin{bmatrix} l_{a1} + \epsilon m_{a1} \\ l_{a2} + \epsilon m_{a2} \\ l_{a3} + \epsilon m_{a3} \end{bmatrix} \cdot \begin{bmatrix} l_{b1} + \epsilon m_{b1} \\ l_{b2} + \epsilon m_{b2} \\ l_{b3} + \epsilon m_{b3} \end{bmatrix} \\
\{\text{take dot product}\} &= \sum_{i=1}^{3} l_{ai} l_{bi} + \epsilon(l_{ai} m_{bi} + l_{bi} m_{ai}) \\
\{\text{as functions of } \theta\} &= \cos(\theta) - \epsilon d \sin(\theta) \\
\{\text{use eq. (2.13)}\} &= \cos(\theta + \epsilon d) \\
\{\text{define } \check{\theta}\} &= \cos(\check{\theta}),
\end{aligned} \tag{2.15}$$

since θ is the angle difference between the lines and d the corresponding shortest distance. Take notice of the so-called screw parameters $(\theta, d, \mathbf{l}, \mathbf{m})$[23], which will be relevant further on.

2.1.4 Dual quaternions

A dual quaternion $\check{\mathsf{q}}$ is a quaternion, its 4 components all being dual numbers. Its real part q is a quaternion representing rotation, and its dual q' is another quaternion representing translation. This composite can be viewed from multiple angles, all valid. Hence there are several ways to denote a dual quaternion. Quaternion notation and ordered-pair notation are the most common ones.

$$\text{Quaternion notation: } \check{\mathsf{q}} = \mathsf{q} + \epsilon \mathsf{q}' \qquad \mathsf{q}, \mathsf{q}' \in \mathbb{H} \tag{2.16}$$

$$\text{Ordered-pair notation: } \check{\mathsf{q}} = (\check{s}, \check{\mathbf{v}}) \qquad \check{\mathbf{v}} \in \Delta^3 \tag{2.17}$$

$$\text{Explicit notation: } \check{\mathsf{q}} = \check{s} + \check{x}i + \check{y}j + \check{z}k \qquad \check{s}, \check{x}, \check{y}, \check{z} \in \Delta \tag{2.18}$$

$$\check{\mathsf{q}} \in \mathcal{H},$$

where $\mathcal{H}$ is the set of all dual quaternions:

$$\mathcal{H} = \left\{ \; \check{\mathsf{q}} = \check{s} + \check{x}i + \check{y}j + \check{z}k \;\middle|\; \check{s}, \check{x}, \check{y}, \check{z} \in \Delta, \quad i^2 = j^2 = k^2 = ijk = -kji = -1 \; \right\}. \tag{2.19}$$

Since $\mathbb{H} \in \mathcal{H}$ and $\Delta \in \mathcal{H}$, dual quaternions may hence be viewed as either a dual number composed of quaternions or a quaternion composed of dual numbers. As a consequence, all rules that apply to quaternions and dual numbers also apply to dual quaternions. This implies that there exists three dual quaternion conjugates: the quaternion conjugate ($\bar{\check{\mathsf{q}}}$), the dual conjugate ($\check{\mathsf{q}}^*$) and their composition [25].

2.1.5 Transformation matrix ↔ DQ conversion

Considering a dual quaternion $\check{\mathsf{q}} = \mathsf{q} + \epsilon \mathsf{q}'$, it can be constructed from a transformation matrix T (defined in (2.3)) in two steps. The first is to construct the real part

$\mathsf{q} = s + xi + yj + zk$ from the rotation matrix with (2.20)-(2.23),

$$s = \frac{1}{2}\sqrt{1 + r_{1,1} + r_{2,2} + r_{3,3}} \tag{2.20}$$

$$x = \frac{1}{4s}(r_{3,2} - r_{2,3}) \tag{2.21}$$

$$y = \frac{1}{4s}(r_{1,3} - r_{3,1}) \tag{2.22}$$

$$z = \frac{1}{4s}(r_{2,1} - r_{1,2}), \tag{2.23}$$

$$\text{where } s \neq 0,$$

and then normalise the result. The expression $s^2 + x^2 + y^2 + z^2 = 1$ should always hold for a unit quaternion. The second step is to calculate the dual part q'. Notice that the translation is expressed as a quaternion,

$$\mathsf{q}' = \frac{\mathsf{tq}}{2}, \tag{2.24}$$

$$\text{where } \mathsf{t} = xi + yj + zk \in \mathbb{H}.$$

Going in the other direction, from a dual quaternion to a transformation matrix, the process is similar but inverted. The translation part is calculated as $\mathbf{t} = 2\mathsf{q}'\bar{\mathsf{q}}$, reordering (2.24) and remembering that $\bar{\mathsf{q}} = \mathsf{q}^{-1}$. The rotation matrix using the real part of the dual quaternion as input is displayed below,

$$R = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2zs & 2xz + 2ys \\ 2xy + 2zs & 1 - 2x^2 - 2z^2 & 2yz - 2xs \\ 2xz - 2ys & 2yz + 2xs & 1 - 2x^2 - 2y^2 \end{bmatrix}. \tag{2.25}$$

2.2 Pose interpolation

Screw Linear Interpolation (ScLERP) is a transformation interpolation method for dual quaternions that uses screw theory. In the most abstract sense, interpolating a transformation from $\check{\mathsf{q}}_a$ to $\check{\mathsf{q}}_b$ would look like:

$$\check{\mathsf{q}}_{ba}(\xi) = \check{\mathsf{q}}_a(\underbrace{\check{\mathsf{q}}_a^{-1}\check{\mathsf{q}}_b}_{\check{\mathsf{q}}_{ScLERP}})^{\xi}, \tag{2.26}$$

where $\xi \in [0, 1]$ determines transition "completion". For example, $\check{\mathsf{q}}_{ba}(0.7)$ would correspond to a dual quaternion 70% through the transformation from pose A to pose B. Since a dual quaternion can be written $\check{\mathsf{q}} = (\cos(\check{\theta}/2), \check{\mathbf{l}}\sin(\check{\theta}/2))$, the exponent ξ may be moved into the trigonometric expressions:

$$\check{\mathsf{q}}^{\xi}_{ScLERP} = \left(\cos\left(\frac{\check{\theta}}{2}\right), \check{\mathbf{l}}\sin\left(\frac{\check{\theta}}{2}\right)\right)^{\xi} = \left(\cos\left(\xi\frac{\check{\theta}}{2}\right), \check{\mathbf{l}}\sin\left(\xi\frac{\check{\theta}}{2}\right)\right), \tag{2.27}$$

$$\begin{aligned} \text{where} \quad & \check{\theta} = \theta + \epsilon d && \check{\theta} \in \Delta \text{ is the dual angle} \\ & \check{\mathbf{l}} = \mathbf{l} + \epsilon\mathbf{m} && \check{\mathbf{l}} \in \Delta^3 \text{ is the dual vector.} \end{aligned}$$

The variables $\theta, d, \mathbf{l}, \mathbf{m}$ are the previously described screw parameters. Hence, interpolation between two quaternions requires the screw parameters of the transformation $\check{\mathbf{q}}_{ScLERP} = \check{\mathbf{q}}_a^{-1}\check{\mathbf{q}}_b = (\check{s}, \check{\mathbf{v}})$. These can be obtained from (2.28) - (2.31) [9, 25]:

$$\theta = \arccos(s) \tag{2.28}$$

$$\mathbf{l} = \begin{cases} \frac{\mathbf{v}}{\sin(\frac{\theta}{2})}, & \text{if } \theta \neq 2\pi n, \qquad \forall n \in \mathbb{Z} \\ \frac{\mathbf{v}'}{|\mathbf{v}'|}, & \text{otherwise} \end{cases} \tag{2.29}$$

$$d = \mathbf{l} \cdot \mathbf{t} \qquad (\mathbf{t} = 2\mathbf{q}'_{ScLERP}\bar{\mathbf{q}}_{ScLERP}) \tag{2.30}$$

$$\mathbf{m} = \frac{1}{2}\left(\mathbf{t} \times \mathbf{l} + (\mathbf{t} - d\mathbf{l})cot\left(\frac{\theta}{2}\right)\right). \tag{2.31}$$

These parameters, used by (2.27) and thereafter (2.26) together enable interpolation between dual quaternions. In this thesis, ScLERP is used to obtain a sample of the vehicle pose at the same timestamp as the LiDAR point cloud sample. This is further described in Section 3.2.1.

2.3 Point cloud registration

The process of aligning two point clouds with one another is called point cloud registration. The alignment is carried out by finding the transformation that minimises the difference in position and orientation between two point clouds. By iteratively aligning new point clouds, multiple transformations are obtained that can be used to build a map in a world coordinate frame.

Point cloud registration methods are subdivided into local and global registration methods [26]. There are various types of global registration methods, such as Random Sampling And Consensus (RANSAC) and fast global registration [26, 27]. What they have in common is that they do not require an approximate initial transformation that roughly aligns the point clouds. These methods do not produce tight alignment either, and will thus not be discussed further.

Local registration methods on the other hand are often more precise when aligning point clouds, they usually require an approximate initial transformation that roughly aligns the point clouds. The most common algorithm today capable of merging point clouds is the ICP algorithm [28], which has numerous variants. The most common ones are point-to-point, point-to-plane and Generalized-ICP [29, 30]. All work quite similarly but the type of objective function that is used for minimisation is different.

2.3.1 Point to point ICP

The point-to-point ICP algorithm requires two point clouds that are supposed to be aligned. The target point cloud P consisting of points $\{p_i\}_{i=1,\dots,N}$ and the source point cloud Q consisting of points $\{q_i\}_{i=1,\dots,M}$. For the ICP algorithm to converge

to a proper solution, an approximate initial transformation T_0 that roughly aligns Q with P is required [31].

The goal is to find the transformation T that aligns P and Q. T is defined over the correspondence set $\mathcal{K} = \{(p, q)\}$. Thus, $\mathcal{K}$ is a set consisting of point pairs (p, q), where each point q in Q is associated with the nearest neighbour point p in P. These correspondences are reevaluated every iteration. The distance between p and Tq is called the correspondence distance. This is done for all points in Q. The transformation T can be calculated by minimising the objective function

$$E(T) = \sum_{(p,q)\in K} \|p - Tq\|^2. \tag{2.32}$$

The algorithm iterates until the two point clouds P and Q are tightly aligned as can be seen in Figure 2.1 or until a max iteration criterion is met. The main principle of the algorithm discussed above is also provided as pseudo-code in Algorithm 2.1.

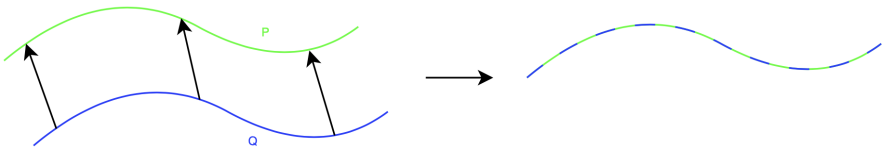


Figure 2.1: Principle of aligning the source point cloud Q to the target point cloud P. The ICP algorithm iterates until the two point clouds are tightly aligned.

Algorithm 2.1: ICP algorithm in pseudo-code.

```
# Input: P, Q, T0
# Output: T
T = T0
while(stopping_criteria):
    for q in Q:
        p = nearestNeighborInP(P, Tq)
        if(norm(p-q) < C_max^corrdist ):  # max correspondence distance
            K += {p, q}
    T = argmin_{T} E(T) # Defined in (2.32),(2.33),(2.35)
```

2.3.2 Point to plane ICP

The point-to-plane algorithm is different from point-to-point in the sense that another objective function is used [28]. By approximating planes instead, a better result is possible to obtain as long as there are surfaces that can be approximated as planes in the environment. Any pair of points p and q, which represent the same surface can be aligned by a rigid transformation. The goal is to find a transformation

that minimises the objective

$$E(T) = \sum_{(p,q)\in\mathcal{K}} ((p - Tq) \cdot n_p)^2. \tag{2.33}$$

Where n_p is the surface normal at point p. By projecting the point-to-point distance on n_p of the target point cloud P, a new error distance d can be found as can be seen in Figure 2.2. This is done for all corresponding point pairs and the procedure is then iterated until the two point clouds are tightly aligned or the max iteration criteria are met.

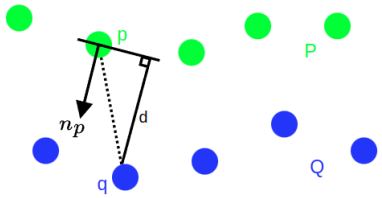


Figure 2.2: The point correspondence $p \in P$ to $q \in Q$ is projected on the surface normal n_p to find distance d

2.3.3 Generalized ICP

By combining the two methods mentioned above, i.e. point-to-point and point-to-plane, a better alignment can be achieved. Generalized-ICP is sometimes referred to as the plane-to-plane ICP, and the algorithm is explained in detail in [30] and is only shortly explained here. The author shows that the Generalized-ICP can outperform point-to-point and point-to-plane, but in [32] it is shown that Generalized-ICP does not work well in outdoor unstructured environments.

Generalized-ICP attaches a probabilistic model to the objective function (2.33). We assume that the corresponding point pairs have been found and the set of all points is indexed as $P = \{p_i\}_{i=1,\ldots,N}$ and $Q = \{q_i\}_{i=1,\ldots,N}$. The distance is then defined as $d_i^{(T)} = p_i - Tq_i$. The probabilistic model assumes that there exists a set of points in the target point cloud $\hat{P} = \{\hat{p}_i\}$ and source point cloud $\hat{Q} = \{\hat{q}_i\}$, which can generate P and Q with $p_i \sim \mathcal{N}(\hat{p}_i, C_i^P)$ and $q_i \sim \mathcal{N}(\hat{q}_i, C_i^Q)$. Here $\{C_i^P\}$ and $\{C_i^Q\}$ represent covariance matrices that are associated with the measured points. We assume that $\hat{p}_i = T^*\hat{q}_i$ where T^* represents the correct transformation. Assume p_i and q_i are drawn from independent Gaussian distributions [30]. Then $d_i^{(T^*)}$ is drawn from the distribution

$$\begin{aligned} d_i^{(T^*)} &\sim \mathcal{N}(\hat{p}_i - (T^*)\hat{q}_i, C_i^P + (T^*)C_i^Q(T^*)^T) \\ &= \mathcal{N}(0, C_i^P + (T^*)C_i^Q(T^*)^T). \end{aligned} \tag{2.34}$$

By using maximum likelihood estimation (MLE), T can be computed iteratively as

$$T = \underset{T}{\operatorname{argmax}} \prod_i \mathbf{p}(d_i^{(T)}) = \underset{T}{\operatorname{argmax}} \sum_i \log(\mathbf{p}(d_i^{(T)})). \tag{2.35}$$

2.4 The hand-eye problem

The hand-eye calibration problem is commonly formulated as $AX = XB$ and was proposed by Tsai and Lenz [8] in 1989. In the original setup, an industrial robot with a gripper (the hand) and a camera (the eye) are mounted fixed relative to each other, separated by some transformation X. Using the knowledge that X will remain the same, regardless of the robot pose, it is possible to calculate it provided the movements of the camera (A) and gripper (B) between 2 or more poses.

In this thesis, the transformation variables (A, B, X) will be named (T_{RR_+}, T_{FF_+}, T_{RF}) The equation will therefore read $T_{RR_+}T_{RF} = T_{RF}T_{FF_+}$. In words, it can be described as "making the transformation T_{RF} followed by T_{RR_+}, means ending up in the same pose as if applying T_{FF_+} followed by T_{RF}". This is illustrated in Figure 2.3. Multiple solutions to the problem have been proposed [8, 10, 23]. Most promising seems the simultaneous solution proposed by Daniilidis and Bayro-Corrochano [9]. The solution uses dual quaternions and Singular Value Decomposition (SVD).

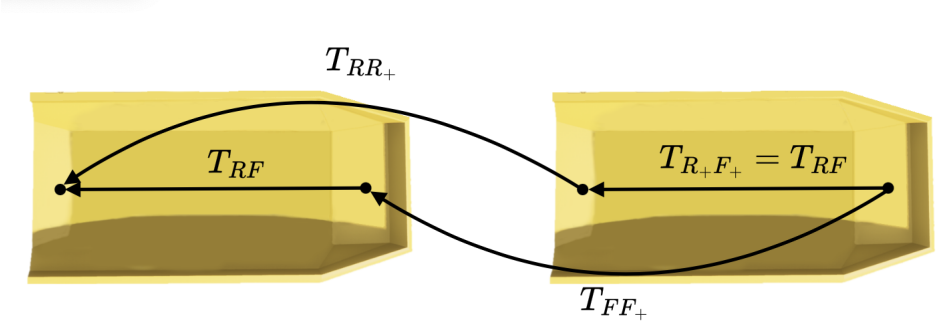


Figure 2.3: Intuition for the hand-eye problem equation $T_{RR_+}T_{RF} = T_{RF}T_{FF_+}$ can be obtained from following the transformations on either side of the equation, seeing that their poses indeed align.

2.4.1 Singular value decomposition

Singular Value Decomposition (SVD) provides a way of decomposing a matrix into its fundamental components [33]. Given a m×n matrix A, its SVD factorization is

$$A = U\Sigma V^*, \tag{2.36}$$

where U is a m×m matrix where the columns are the left singular vectors which are orthonormal eigenvectors of matrix AA^*, Σ is a m×n diagonal matrix with singular values, i.e. $\sigma_i(A) = \sqrt{\lambda_i(A^*A)}$ where $\lambda_i(A^*A)$ denotes the eigenvalues for A^*A, V^* is a n×n matrix where the columns are the right singular vectors which are orthonormal eigenvectors of matrix A^*A and $*$ denotes the complex conjugate transpose.

3
Method

In order to find the extrinsic calibration parameters for the LiDARs, the problem is divided into sub-problems. In Figure 3.1, the sub-problems are illustrated in a flowchart. The section outlines the two calibration methods used in the thesis, which are identical up until the mapping procedure. Input data collected from simulations of a digital twin of a quarry is pre-processed and used to create point cloud maps of the surroundings. Two point cloud maps are produced — one for each LiDAR. The two methods then use the mapping output differently to each give their own estimate of the relative LiDAR pose T_{RF}. Method 1, *Hand-Eye*, uses Daniilidis' [9] solution to the hand-eye problem. Method 2, *MergeMaps*, aligns the point cloud maps of the front and rear LiDARs using point-to-plane ICP.

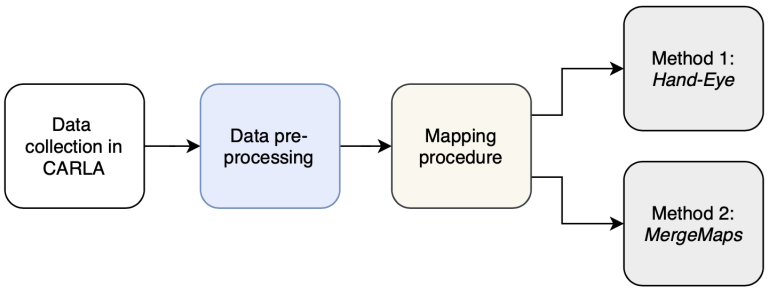


Figure 3.1: Sub-problems to solve to develop an extrinsic LiDAR algorithm for the dumper.

3.1 Data collection

The autonomous driving simulator Car Learning to Act (CARLA) [1] is used to collect data in a digital version of an existing quarry. The measured environment consists of mostly stones, mountain walls and rough roads. A rotating OS1-32 LiDAR is defined using the CARLA LiDAR sensor model, enabling also the simulated version to have a sample rate of 10 Hz and a horizontal FoV of 180°. Each point provided by the sensor model is an exact distance measurement from the LiDAR to the closest surface in a predefined direction relative to the LiDAR. The odometry estimate is fused from wheel odometry, IMU and GNSS data (all simulated in CARLA). When driving, measurement data from the simulated sensors is recorded and saved as mcap files.

3.2 Data pre-processing

The recorded LiDAR and odometry data requires some pre-processing. The flowchart in Figure 3.2 shows the most important steps. Time-synchronisation of LiDAR and odometry data is performed, followed by point cloud down-sampling. Next, point outliers are rejected, and lastly the normals required by the point-to-plane ICP algorithm are computed.

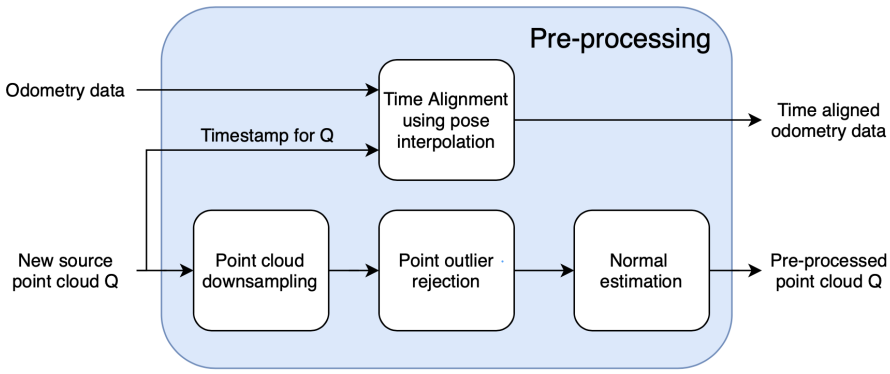


Figure 3.2: Breakdown of the measurement data pre-processing, displayed as a flowchart.

3.2.1 Time alignment using pose interpolation

The data read from the mcap file is not synchronised in time. More precisely, the two LiDARs and the odometry estimation algorithm provided by V.A.S all samples asynchronously. Using ScLERP, the odometry samples are interpolated and resampled to match the LiDAR samples. This is done for each LiDAR separately. Figure 3.3 illustrates this, exemplifying an interpolation of odometry data to match the sampling of the front LiDAR. The rear LiDAR follows the same procedure.

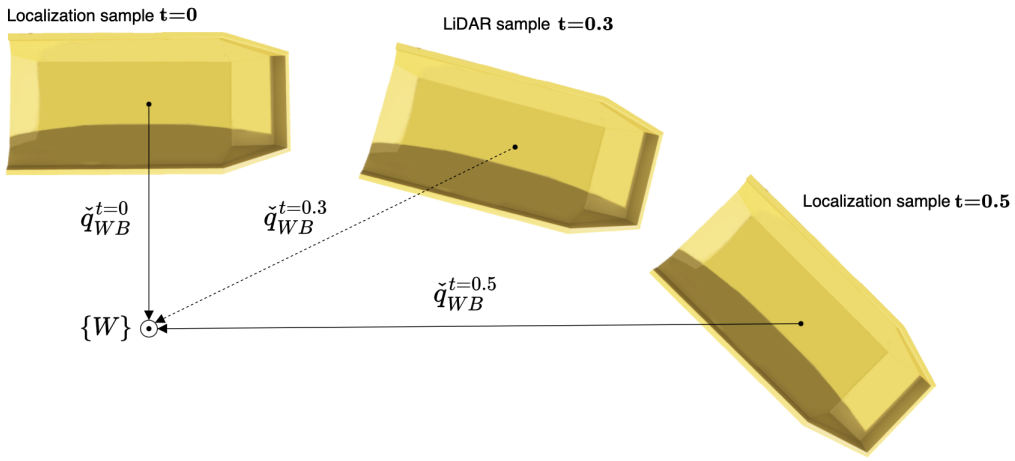


Figure 3.3: ScLERP used for time alignment. The front LiDAR samples at t=0.3, the pose (sampled at t={0, 0.5}) is interpolated at t=0.3 using ScLERP.

Continuing the same example, in (3.2) the pose is interpolated at $t = 0.3$ using (2.26). Since the transition "completion" at $t = 0.3$ is 60%, the argument ξ is 0.6. The expression is rewritten in terms of the dual angle $\check{\theta}$ and dual vector $\check{\mathbf{I}}$, derived from calculating the screw parameters of $\check{\mathbf{q}}_{ScLERP}$ using (2.26)-(2.31):

$$\check{\mathbf{q}}_{WB}^{t=0.3}(\xi) = \check{\mathbf{q}}_{WB}^{t=0}(\overbrace{(\check{\mathbf{q}}_{WB}^{t=0})^{-1}\check{\mathbf{q}}_{WB}^{t=0.5}}^{\check{\mathbf{q}}_{ScLERP}})^{\xi} \tag{3.1}$$

$$= \check{\mathbf{q}}_{WB}^{t=0}\left(\cos\left(\xi\frac{\check{\theta}}{2}\right), \check{\mathbf{I}}\sin\left(\xi\frac{\check{\theta}}{2}\right)\right). \tag{3.2}$$

The front and rear poses are then time aligned. For one front-rear pair of poses the transformation between them,

$$T_{R^{t=\{F\}}R^{t=\{R\}}} = T_{MR}^{-1}T_{WM^{t=\{F\}}}^{-1}T_{WM^{t=\{R\}}}T_{MR}, \tag{3.3}$$

is applied to the rear point cloud. The formulation $T_{R^{t=\{F\}}R^{t=\{R\}}}$ refers to the transformation from $\{R\}$ at the sample time of the rear LiDAR to $\{R\}$ at the sample time of the front LiDAR. The transformations T_{MR} surrounding the synchronisation transformation serve to transport the origin from the rear LiDAR to the base frame (the point at which odometry data is sampled) before the points are transformed and then back.

3.2.2 Point cloud downsampling

Voxel grid downsampling is used to downsample the point cloud. A voxel size of 10x10x10 [cm] is used. All points inside a voxel are averaged to form one new point representing all other points inside that voxel. The result is a new point cloud similar to the original one that uses a lower number of points. This reduces downstream computational complexity and creates a more equal distribution of points in space. This is beneficial for convergence of the ICP algorithm.

3.2.3 Point outlier rejection

The sampled point clouds from the LiDARs could be noisy, have sparse point density, and contain false measurement points, also known as outliers. While the simulation data does not contain any outliers, it is possible to analyse side effects of rejection if including it in the algorithm. Outliers pose a problem when aligning point clouds as some points could match with false measurements. Therefore, it is a good idea to remove outliers before aligning point clouds to make it easier for the ICP to converge to a good alignment. The problem is solved with a statistical outlier rejection algorithm [34].

For every point p_i in the point cloud, the average distance $\bar{d}_i$ from p_i to its n nearest neighbours is computed. By using all neighbouring average distances $\bar{d}_i$, the distance mean μ and standard deviation σ are calculated. A threshold $\gamma = \mu + \epsilon\sigma$ is formulated, where ϵ is a scalar value used to tune the filter. An outlier is found and

removed if $\bar{d}_i > \gamma$ holds.

The point cloud in Figure 3.4 originates from the real-world dataset, which was also used for tuning the outlier rejection algorithm. Outliers to be removed are visualised in red, gray points are retained. The tuned point cloud rejection consists of two passes using the algorithm. The first pass has the parameters 300 neighbours and $\epsilon = 2.0$, and the second pass has 80 neighbours and $\epsilon = 2.8$.

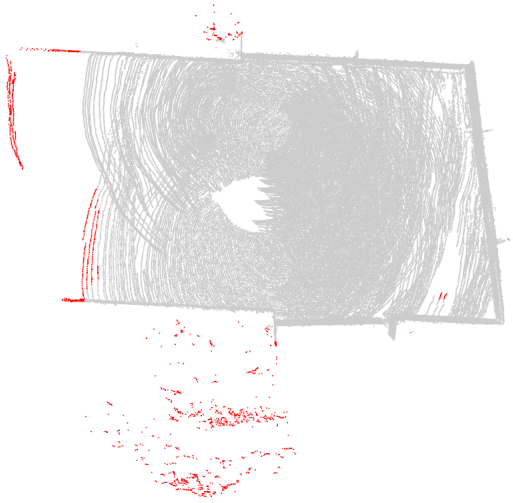


Figure 3.4: Top view of a point cloud which is obtained from real-world LiDAR data. The red points are determined to be outliers that will be removed from the point cloud. Note that the distinct features in grey as well as most of the ground plane inside are kept, but a lot of sparse points outside are considered outliers.

3.2.4 Normal estimation

The normal vector for each point in every sampled point cloud is computed using Algorithm 3.1, enabling the use of point-to-plane ICP for the mapping. Point clouds sampled by a rotating LiDAR tend to be very sparse in the vertical direction relative to the horizontal direction, as well as more sparse the further away from the sensor the points are. Algorithm 3.1 takes these properties into account. This makes it advantageous over for example the normal estimation algorithm proposed by Eberley [35] for the given type of point clouds.

Algorithm 3.1: Dynamic radius normal estimation algorithm

```
# cloud: structure representing point cloud of n points
# cloud.points:  array of n point coordinates [[x,y,z]_0,...,[x,y,z]_{n-1}]
# cloud.normals: array of point normals       [[x,y,z]_0,...,[x,y,z]_{n-1}]
def dynamicRadiusNormalEst(cloud)
    # Split point cloud in 2, based on the distance to LiDAR
    halfcloud[0], halfcloud[1] = splitPCin2(cloud)
    # Split resulting point clouds in 2
    partcloud[0], partcloud[1] = splitPCin2(halfcloud[0])
    partcloud[2], partcloud[3] = splitPCin2(halfcloud[1])

    LiDAR_vFOV = np.pi / 4  # LiDAR has 45° vertical FoV and
    LiDAR_CH = 32           #           32 channels
    alpha = LiDAR_vFOV/(LiDAR_CH-1)
    beta = 20   # 20\degree, Design parameter
    for i in [0, len(partcloud)]:
        # distance to farthest point in partcloud
        d_max = max(norm(partcloud[i].points))
        r_normal = d_max * sin(alpha) / sin(alpha+beta) # eq.
        ↪ (3.4)
        partcloud = computeNormals(partcloud[i], r_normal)

        cloud += partcloud[i]

    return cloud



def splitPCin2(cloud):
    r = norm(cloud.points) # r_i = sqrt(x_i^2 + y_i^2 + z_i^2)

    lowidx = r < median(r)
    return cloud[lowidx], cloud[not lowidx]
```

The proposed normal estimation algorithm extends the method proposed by Eberley [35]. Simplified, for each point p in a given point cloud, points within a static radius of p are used to estimate a plane, whose normal is assigned to p. This algorithm uses the same normal estimation radius for all points, regardless of the distance to the sensor. While this works well for point clouds where points are approximately equally distanced throughout space, this is not the case here. Dynamic normal estimation radius selection is implemented according to Algorithm 3.1, so that the search radius is small for points close to the LiDAR, and increases with increasing distance.

Going through Algorithm 3.1, the point cloud is split in 4, normals are estimated using different radii, then the cloud is reassembled. The choice of 4 partitions is due to that points within the normal estimation radius originate from two different vertical channels of the LiDAR, hence motivating a low number of point cloud partitions. To make a good normal estimate, each point requires at least two points within its radius, separated by a somewhat right angle. This can virtually only be obtained

if the mentioned points originate from at least 2 different channels. Points fulfilling this condition make for a robust plane estimate and thereby normal estimate, which is desirable.

For each of the 4 cloud partitions, the normals are estimated using [26], [35] and the distance to the farthest point in that partition. The latter is used to dynamically select a normal estimation radius. A desired radius is equal to or slightly greater than the distance to the closest points in a neighbouring LiDAR channel. To simplify how this distance varies in the environment, a flat surface is assumed. The normal estimation radius (r_{normal}) depends on the angle between the channels (α), the distance to the assumed surface (d_{max}) and the surface orientation (in pitch) relative to the LiDAR beam (β):

$$r_{normal} = d_{max} \frac{\sin(\alpha)}{\sin(\alpha + \beta)}. \tag{3.4}$$

Figure 3.5 illustrates the relation described in (3.4). The idea is that the radius should be large enough for all points to reach a point in the neighbouring channel. Hence the farthest point is used in (3.4). The angle of incidence of the LiDAR ray also matters; decreasing β means increasing r_{normal}. The design parameter β is set to 20°. This implies that the distance between points of neighbouring channels will be larger than r_{normal} for surfaces where the angle of incidence is lower than $90° - \beta$. If those points are a distance d_{max} away from the LiDAR, that is.

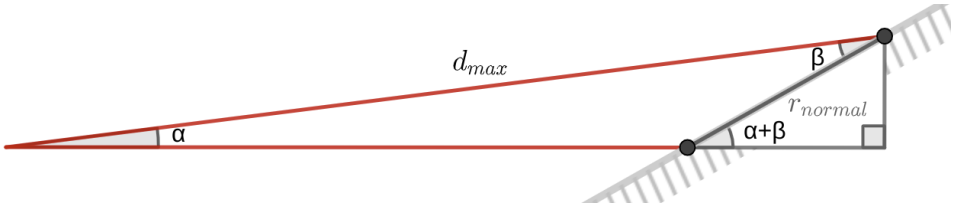


Figure 3.5: The normal estimation radius r_{normal} is selected based on the distance d_{max} to the farthest point in that point cloud partition. The red lines represent LiDAR beams of two neighbouring channels. The figure is not to scale.

3.3 Mapping procedure

The mapping procedure handles how a new point cloud Q is aligned with and integrated into the point cloud map P, which eventually becomes a high-fidelity point cloud representation of the environment. The steps of the procedure can be viewed in Figure 3.6 and each step will be discussed more in this section.

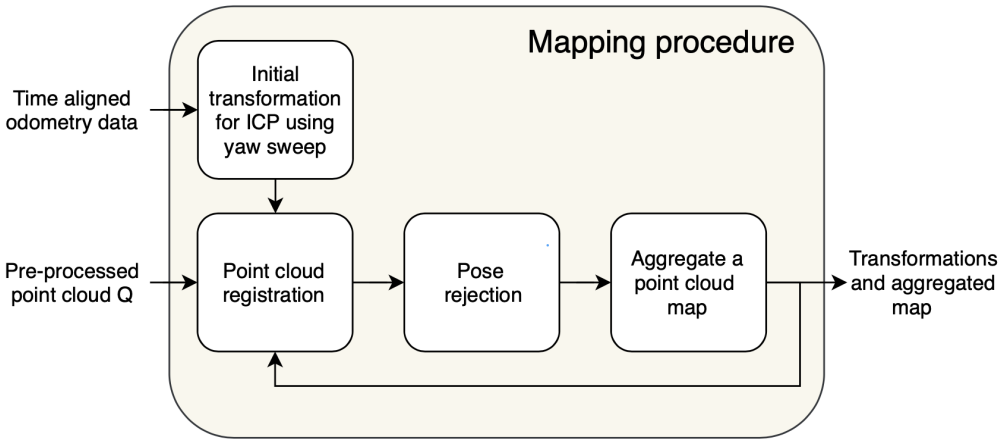


Figure 3.6: Mapping procedure inside the beige box. Each box describes one subtask for the procedure.

The hand-eye problem requires the pose transformations $T_{F_0F_n}$ and $T_{R_0R_n}$ and the *MergeMaps* method requires point cloud maps. The transformations and the maps are two faces of the same coin and can be calculated by aligning the point clouds corresponding to these poses. These point clouds are provided as LiDAR scans. The LiDAR point clouds are provided in their respective frame of reference ({F} or {R}). The broad idea is to transform these point clouds into two unified maps - one for the front LiDAR with origin in $\{F_0\}$ and one for the rear with origin in $\{R_0\}$.

In this section, Algorithm 3.2 and the calculation of a single transformation pair T_{F_0F}, T_{R_0R} will be detailed. First, an initial transformation approximately aligning the point clouds is estimated using odometry and a parameter sweep. Then this transformation is used to initialise the ICP algorithm, which outputs the desired transformation (T_{FF_+} or T_{RR_+}). If the alignment is poor, or if geometrical constraints are violated, the pose is rejected. If not, the current pose is saved and the point cloud is added to the map.

Algorithm 3.2: The transformation T_{LL_+} from a LiDAR in pose '+' to the previous pose, is obtained using ICP and a sweep in yaw sufficiently approximating T_{LL_+}.

```
for θ in range(yaw_param_sweep):
    T_BL = T(θ)
    dictOfTinit[θ] = T_BL^-1 T_WB^-1 T_WB+ T_BL  # T_BL==T_BL==T_B+L+
T_LL+^init = compareClouds(dictOfTinit, PC, PC+) # best T in dict
T_LL+ = ICP(T_LL+^init, PC, PC+) # ICP very likely converges

if(rejectPoseBadICP() or rejectPoseGeometry()):
    continue  # Pose never added to the cloud etc.
T_L0L+ = T_L0L+ T_LL+    # Saved as T_L0Ln
PC = T_LL+^-1 PC + PC+   # "Add" point cloud map PC to PC+
```

3.3.1 Initial transformations for ICP

When aligning point clouds with the ICP algorithm, the algorithm requires an initial transformation that roughly aligns the point clouds. This initial transformation is computed using odometry data, which yields the base-to-world transformations T_{WB} and T_{WB+}. Figure 3.7 illustrates two poses of the front LiDAR for the dumper, and Figure 3.8 shows two poses for the rear LiDAR.

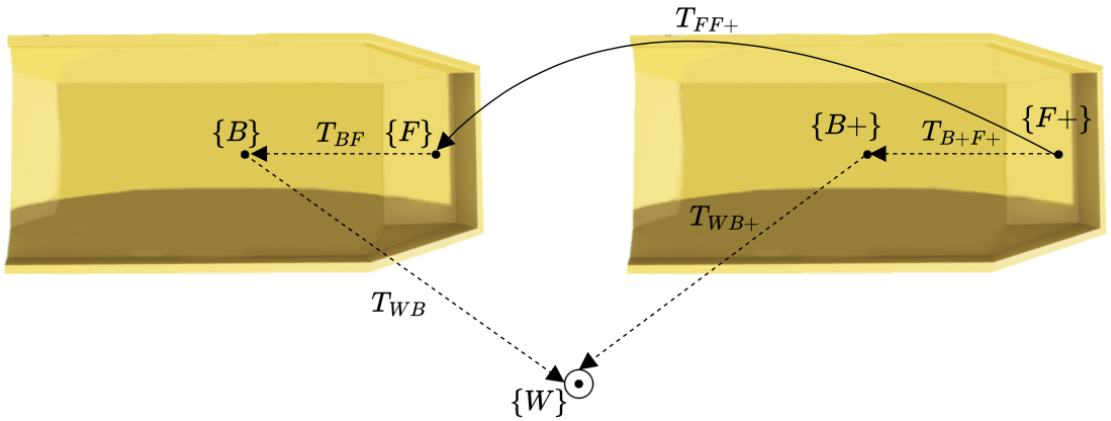


Figure 3.7: Top view of the dumper showing two consecutive poses for the dumper which are used to calculate the front LiDAR initial transformation T_{FF+}.

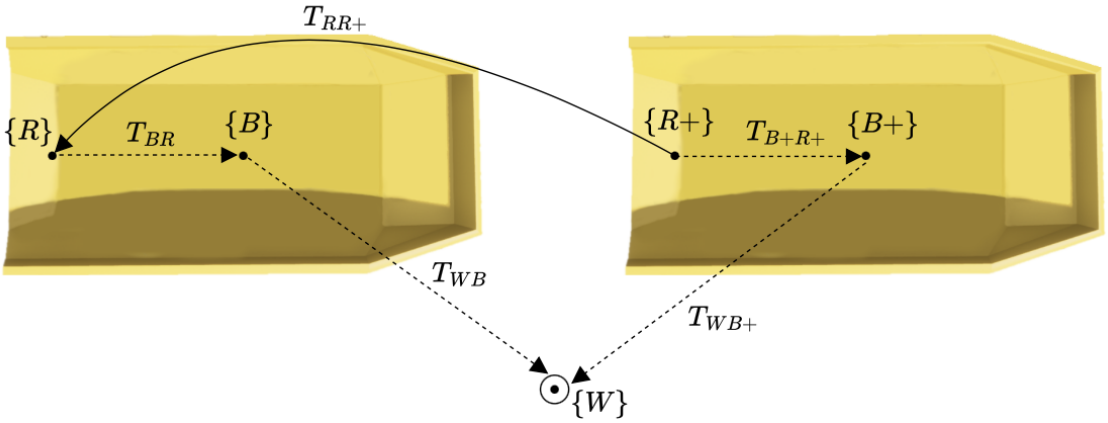


Figure 3.8: Top view of the dumper showing two consecutive poses for the dumper which are used to calculate the rear LiDAR initial transformation T_{RR+}.

By utilising two consecutive poses, a new transformation T_{FF+} can be calculated as

$$T_{FF+} = T_{BF}^{-1} T_{WB}^{-1} T_{WB+} T_{B+F+}, \tag{3.5}$$

to represent the transformation from the current front LiDAR pose to the previous front LiDAR pose. Similarly, the transformation T_{RR+} for the rear LiDAR can be calculated as

$$T_{RR+} = T_{BR}^{-1} T_{WB}^{-1} T_{WB+} T_{B+R+}. \tag{3.6}$$

These transformations are referred to as the initial transforms when aligning the point clouds. It should be noted that $T_{BF} = T_{B+F+}$ as well as $T_{BR} = T_{B+R+}$ are unknown. To solve this, they are estimated, this is described in Section 3.3.2. This estimate does not need to be very precise as it is only for calculating the initial transforms to make the ICP algorithm converge when aligning point clouds.

3.3.2 Find LiDAR yaw angle

The LiDAR to base frame transforms T_{BF} and T_{BR} are estimated to enable calculation of the initial transform for the ICP algorithm. A reasonable estimation for the translation component can be obtained from the nominal values found in Table 1.2. The roll and pitch are assumed to not deviate much from the nominal values, hence these values are also obtained from Table 1.2. The yaw angle has been observed to deviate more from its nominal mounting position than any other degree of freedom. Hence, only the yaw angle is estimated. More precisely, the sought angle is the relative yaw angle θ_z between a LiDAR frame of reference and the base frame. For T_{BF}, the angle appears in the matrix

$$T_{BF}(\theta_z) = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 1.978 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 1.18 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.7}$$

and the analogue is done for T_{BR}. The method to do this is to iterate through all angles in a determined range close to the nominal value. For the front LiDAR the yaw angle is supposed to be at 0°. The range will be $[-5, 5]$, and the rear is supposed to be rotated 180°, thus resulting in iterating through the range $[175, 185]$. A Root Mean Square Error (RMSE) value for point-to-point distance will be calculated between the two point clouds that are to be matched considering all errors less than 0.3 meters as can be seen in Algorithm 3.3. The transformation with the smallest point-to-point RMSE is selected. This transform then serves as the initial transformation required by the ICP algorithm to align point clouds.

Point-to-point distances larger than 0.3 meters are excluded from the error. The reason is that assuming that the vehicle turns, a small portion of the FoV will uncover new points that do not have correspondences in the older point cloud. This will produce large errors and make the algorithm more prone to select an angle which is slightly wrong but overlaps with the already covered FoV.

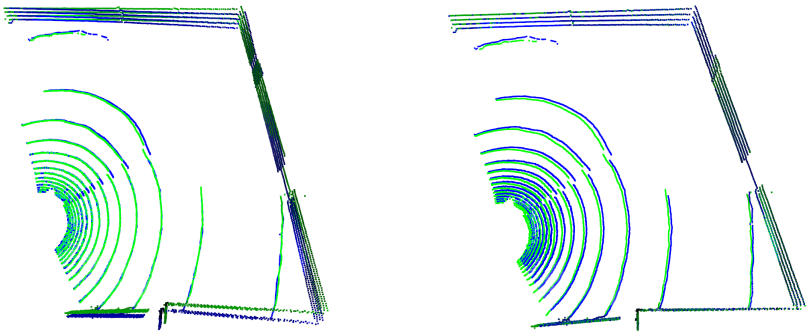
Algorithm 3.3: Yaw angle search

```
for θ, T^θ_{LL+} in dictOfTinit:

    error = (T^θ_{LL+} PC+).p2pDistances(PC)   # len(error)=len(PC+)

    rmse_theta = rms(filterIfAbove(0.3, error))   # Calc RMSE

    if(rmse_theta < rmse_best):
        rmse_best = rmse_theta
        T^init_{LL+} = T^θ_{LL+}
```

3.3.3 Point cloud registration

The implemented algorithm is able to use either of the three ICP methods, i.e. point-to-point, point-to-plane, and Generalized-ICP. Point-to-point does not produce tight alignment and is therefore abandoned for the other two methods that show significantly better results. The calibration algorithm uses the Open3D [26] implementation of ICP. The point-to-plane method is chosen over Generalized-ICP based on indications of ever so slightly better performance.

The ICP algorithm needs four parameters passed as arguments to the function - the source point cloud, target point cloud, maximum correspondence distance and initial transform. The maximum correspondence points-pair distance is tuned to 0.2 meters. The initial transformation applied to the source point cloud roughly aligns it with the target point cloud. This initial transformation is the one calculated in (3.5) and (3.6). The ICP algorithm iterates until the two point clouds converge (relative change in fitness and RMSE per iteration less than 10^{-6}), or until a stop criterion is met which is set to a maximum of 1000 ICP iterations. Two point clouds (green and blue colour) for consecutive poses can be seen in 3.9a and the merged point clouds can be seen in Figure 3.9b.



(a) Before alignment **(b)** After alignment

Figure 3.9: Top view of two front LiDAR point clouds. The blue point cloud is scanned one pose after the green one, thus they will not align. The aligning transformation is obtained from the ICP algorithm.

3.3.4 Pose rejection

Pose rejection is based on poor ICP convergence and violation of geometrical constraints. The ICP alignment is considered unsatisfactory if the fitness of the transformation is lower than a certain threshold. The geometrical constraints may be subdivided into rotational and translational constraints. Broadly, if the front and rear ICP results are geometrically incompatible, the pose is rejected. If either the front or rear transformation does not meet these criteria, both transformations are rejected altogether. Furthermore, the behaviour of the algorithm, if a pose is rejected, is the same as it would have been without the rejected pose in the input data.

The quality of the ICP result is analysed using fitness as the metric for pose rejection. A fitness lower than the threshold C_{fit}^{reject} results in the pose being rejected. The threshold is ramped up linearly for the first 10 poses and then kept at a constant level, in order to allow lower fitness for the first poses. This is reasonable since the ground points will not yet have any point correspondences in the point cloud map. Due to the vehicle turning, there will also be a fraction of the field of view being newly discovered for roughly the first half of the circulation. With this in mind, C_{fit}^{reject} is set to 0.4. That is, rejecting point clouds where less than 40% of the sampled point cloud corresponds to the map. Hence this filter is primarily a safeguard for really bad matches.

The geometrical constraints ensure that the difference in rotation and translation from one pose to the next is similar for both LiDARs. Algorithm 3.4 details the procedure. The difference in Euclidean distance travelled is chosen as metric. This works well as opposed to comparing each axis separately, which becomes problematic as soon as the LiDAR axes misalign. The analogue concept applies to the rotation constraint.

Algorithm 3.4: Pose rejection based on geometrical constraints.

```
1 # According to eq. (2.5):
2 Fxyz, Frpy = T2xyzrpy(T_{FF_+})  # F_xyz = [t_x, t_y, t_z]^T,  F_rpy = [θ_x, θ_y, θ_z]^T
3 Rxyz, Rrpy = T2xyzrpy(T_{RR_+})
4
5 # Compare the difference of norms
6 if(abs(‖Fxyz‖ - ‖Rxyz‖) > C_trans^reject or       # C_trans^reject = 0.05°
7    abs(‖Frpy‖ - ‖Rrpy‖) > C_rot^reject ):         # C_rot^reject = 0.4m
8     continue  # reject pose by continuing directly to the next
       ↪ pose
```

3.3.5 Aggregate point cloud map

The LiDAR frames of reference for the first vehicle pose ($\{F_0\}$, $\{R_0\}$) are used as starting points for building the point cloud map. For each new accepted pose, the map follows along. This is what line 10 in Algorithm 3.2 does. After the final pose has been processed, the map is transformed back to its initial frame of reference -

$\{F_0\}$ for the front LiDAR map, $\{R_0\}$ for the rear LiDAR map.

3.4 Method 1: *Hand-Eye*

In Figure 3.10 the transformations $T_{F_0F_n}$ for all n are produced from the mapping (and similar for $T_{R_0R_n}$) to calculate T_{RF} using Daniliidis' and Bayro-Corrochanos [9] dual quaternion solution to the hand-eye problem. The essence of their proposed 5-step solution to the hand-eye problem is explained here. For a more detailed explanation, the authors refer to Daniliidis' more comprehensive version [23]. The implementation by the Autonomous Systems Lab at ETH Zürich [12] is used.

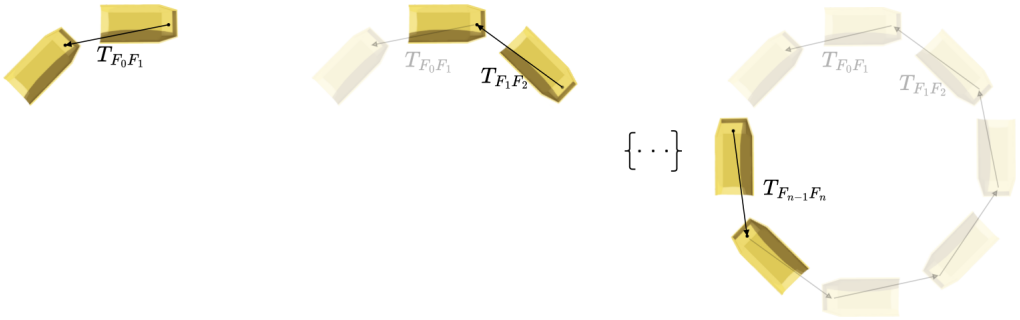


Figure 3.10: The transformations $T_{F_0F_n}$ are produced when mapping by chaining the transformations connecting adjacent poses. The same is done for $T_{R_0R_n}$. These transformations are then used in the hand-eye problem to find T_{RF}

Before step number one, the transformations $T_{F_0F_n}$ and $T_{R_0R_n}$ are converted to their dual quaternion versions $\check{\mathsf{q}}_{F_0F_n}$ and $\check{\mathsf{q}}_{R_0R_n}$ using (2.20)-(2.24).

1. Given $\check{\mathsf{q}}_{F_0F_n}$ and $\check{\mathsf{q}}_{R_0R_n}$, check if the scalar parts are close to equal using

$$\check{\mathsf{q}}_{F_0F_n}\bar{\check{\mathsf{q}}}_{F_0F_n} - \check{\mathsf{q}}_{R_0R_n}\bar{\check{\mathsf{q}}}_{R_0R_n} < C_{\check{s}=}^{threshold}, \tag{3.8}$$

   where $C_{\check{s}=}^{threshold}$ was set to 0.4. Then use (3.9) to construct the matrix T:

$$\underbrace{\begin{bmatrix} S_1 \\ \vdots \\ S_N \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} \mathsf{q}_{RF} \\ \mathsf{q}'_{RF} \end{bmatrix}}_{\check{\mathsf{q}}_{RF}} = 0 \tag{3.9}$$

   where

$$S_n = \begin{bmatrix} R_n - F_n & \left[R_n + F_n\right]_\times & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} \\ R'_n - F'_n & \left[R'_n + F'_n\right]_\times & R_n - F_n & \left[R_n + F_n\right]_\times \end{bmatrix}.$$

   Notice that only the 6 imaginary parts of the dual quaternions $\check{\mathsf{q}}_{F_0F_n} = (0, F_n + \epsilon F'_n)$ are used in (3.9), where $[.]_\times$ denotes the skew-symmetric matrix with the property $[a]_\times b = a \times b$.

2. Compute the SVD of T and check if only two singular values are less than 0.5. Take their corresponding right-singular vectors - columns of V - v_7, v_8. The result will be $\check{\mathsf{q}}_{RF} = \lambda_1 v_7 + \lambda_2 v_8$, a linear combination of these 8x1 column vectors. To find the values of λ_1 and λ_2, split v_7 and v_8:

$$\check{\mathsf{q}}_{RF} = \lambda_1 \underbrace{\begin{bmatrix} u_1 \\ v_1 \end{bmatrix}}_{v_7} + \lambda_2 \underbrace{\begin{bmatrix} u_2 \\ v_2 \end{bmatrix}}_{v_8}. \tag{3.10}$$

3. Inserting (3.10) into $\mathsf{q}_{RF}^T \mathsf{q}_{RF} = 1$ and $\mathsf{q}_{RF}^T \mathsf{q}'_{RF} = 0$ - requirements for the vector to be a unit dual quaternion - we get what is referred to as equations (34) and (35) in [23]. If also making the variable substitution $\lambda_1 = s\lambda_2$ these read as

$$\lambda_2^2(s^2 u_1^T u_1 + 2s u_1^T u_2 + u_2^T u_2) = 1 \tag{3.11}$$
$$\lambda_2^2(s^2 u_1^T v_1 + s(u_1^T v_2 + u_2^T v_1) + u_2^T v_2) = 0, \tag{3.12}$$

   the latter from which s_1 and s_2 can be calculated using the quadratic formula for second-order equations.
4. s_1 and s_2 are inserted into (3.11) to form two solutions. The s producing the highest function value, s_*, is selected. Then by rearranging (3.11), $\lambda_2 = \sqrt{1/s_*}$. Just as before, $\lambda_1 = s\lambda_2$.

5. Equation (3.10) gives the result $\check{\mathsf{q}}_{RF}$

3.5 Method 2: *MergeMaps*

The second method builds on the established [36], a quite simplistic approach that, once the front and rear point cloud maps exist, the sought transformation T_{RF} is also the one aligning these maps. Therefore, T_{RF} can be calculated using point cloud alignment.

The front and rear point cloud maps are aligned using point-to-plane ICP with a maximum correspondence distance of 0.3 meters, and the same voxel size and other tuning parameters as previously described. The initial transformation used for the ICP algorithm is the nominal relative transformation (Table 1.2) with a perturbation $T_{disturbance}$, defined as

$$T_{disturbance} = \begin{bmatrix} \cos(\pi + 10\delta_\theta) & -\sin(\pi + 10\delta_\theta) & 0 & -3.936 + 0.25\delta_x \\ \sin(\pi + 10\delta_\theta) & \cos(\pi + 10\delta_\theta) & 0 & 0.25\delta_y \\ 0 & 0 & 1 & 0.1\delta_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

where $\delta_{\{.\}} \in [-1, 1]$ is a uniformly distributed stochastic variable.

Tests suggest that the alignment algorithm is able to converge even when $T_{disturbance}$ is large. For example, it is able to converge from an error of $\pm 30°$ in yaw and ± 0.5 meters in x and y. This is compared to the nominal transformation.

3.6 Environment augmentation test design

The algorithms are tested by making 200 calibrations in 25 different scenarios. These scenarios (CARLA recordings) are composed of a combination of 5 calibration locations and 5 setups intended to augment the environment in each location. The tests are carried out in a digital twin of a quarry, simulated in CARLA. The 5 setups include 'no augmentations' and 4 different setups of landmarks placed in a ring-like pattern around the circling vehicle.

Each of the 200 calibrations get access to 155 poses - approximately one lap - of recorded data. The first 5 poses of each recording are ignored, due to the initial covariance of the odometry estimate being very high. In order to reduce the risk of ambiguous results due to some of the scenarios giving a very good calibration result by chance, 8 calibrations are made in each scenario. Each of these are engineered to have slightly different initial conditions by ignoring yet another pose for each calibration round. Hence the 8th calibration in each scenario would skip the first 12 poses of that recording.

3.6.1 Environment selection

Out of 9 available sites in the quarry, 5 locations are selected based on that they would have been the most likely starting points for calibration in reality. They are selected to see what different structures in the surrounding can be helpful when using measurement data from the sensors. An overview of the environment augmentation test locations can be seen in Figure 3.11.

(a) Locations #1 and #5

(b) Location #2

(c) Location #3

(d) Location #4

Figure 3.11: Locations used for the environment augmentation test.

In the list below, the test locations #1-#5 are briefly described.

- Location (#1): The environment consists of sides with gently sloping mountains near the driving route, one side is flat and has no significant features. Some trees exist. The vehicle drive route has an elevation of 0.7 m.

- Location (#2): The environment features a three-way cross with two small mountains and one low but broad stone wall. This area has fewer features since it is a three-way crossing. The vehicle drive route has an elevation of 0.4 m.

- Location (#3): This environment consists of two sides with vertical (close to 90° angle) mountains and one sloping mounting side close to the driving route, while the fourth side is flat and has no features. The vehicle drive route has an elevation of 0.6 m.

- Location (#4): The environment consists of a flat, open area where another mining vehicle is positioned nearby. There are two steep mountains, with vertical walls but at a larger distance compared to the other locations. The vehicle drive route has an elevation of 0.3 m.

- Location (#5): The environment is the same as location #1 but the vehicle spawns a little further away from the mountains and trees. The vehicle drive route has an elevation of 0.2 m.

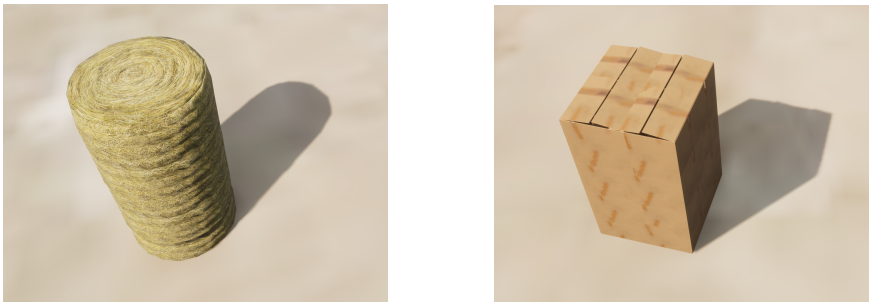
3.6.2 Augmentation object selection

The calibration accuracy could benefit from inserting additional artificial landmarks into the environment. To investigate this, landmarks are placed in a circle with a radius larger than the one the dumper is driving in, see Algorithm 3.5. The landmarks are placed at a distance of approximately 3 meters from the dumper, leaving some room for the limitation to keep a distance of 2 meters.

Algorithm 3.5: Generate position and number of objects in the algorithm in pseudo-code.

```
% Input: location centerpoint x0,y0,z0, Number of objects N,
↪  Circle radius R
% Output: list of objects' location

for n in N:
    x = x0 + R*cos(2*PI*n/N)
    y = y0 + R*sin(2*PI*n/N)
    z = z0
    objLocations[n] = [x, y, z]
```

Objects used as artificial landmarks are the CARLA blueprints "box01" and "haybale", which are grouped and stacked on top of each other so that the size of each landmark approximates to 1x1x2 m. These 2 types of objects are selected because of their differences in shape. The boxes have sharp corners and the hay bales are cylindrical as can be seen in Figure 3.12. Tests are conducted on cases using varying numbers of objects - none, 5 or 10.



(a) Hay bale **(b)** Box

Figure 3.12: Two different objects are used for augmentation - boxes and cylinders. These are composed from smaller objects

3.6.3 CARLA recording procedure

For each of the calibration scenarios investigated in the thesis, the simulation data acquisition is partly manual for each scenario. For consistency, the following procedure is used for recording the corresponding datasets:

1. Set simulation parameters
   - Spawn point: 5 different locations in the quarry are selected to see if different structures in the local environment have an impact on the calibration. All 5 selected locations have various features such as steep mountain walls, mountain walls with more gentle slopes, large stones, trees, and different elevations for ground terrain.
   - Throttle: The dumper is driving as slowly as possible at a constant velocity. Throttle is set to 0.05, this is a value between $[0.0, 1.0]$ in CARLA.
   - Steering: The dumper is driving in a circle with a radius of 6.375 m measured from the base frame to the circle centre point. Steering is set to (+/-0.7), this is a value between $[-1.0, 1.0]$ in CARLA.
   - If environment augmentation:
     - Add artificial landmark objects with different shapes and sizes.
     - Generate a list of all object's positions using Algorithm 3.5
2. Run CARLA simulation (automated process)
3. Start recording at a position pre-decided for that spawn point.
4. Stop recording after 1 and a half laps of driving. The algorithm will only look at the first 155 poses, of which the 5 first up to the 12 first are discarded. This approximately gives 1 lap of recorded data and ensures that each test has access to the same amount of data.

3.7 Real-world verification test design

Things may look fantastic in a simulation environment and still not work at all in reality. Reasons include over-fitting to the simulation environment and significant differences between simulation and real conditions. This is why it was also decided to test the algorithm on a real-world dataset. The dataset consists of one environment, shown in Figure 3.13. Since the real-world test primarily serves as a sanity check and due to limited data availability this single recording is considered sufficient for the purpose.

Figure 3.13: Real-world test site, containers are placed on an airport landing strip. Not shown in the picture are road barriers, approximately 1 meter tall, extending the containers at the time when the data was recorded.

The dataset is readily provided by V.A.S. The TA15 is driven in a U-turn, manually by remote. Since the data is provided, the exact methodology and circumstances for the recording remain unknown to the authors. That includes the exact relative pose of the LiDARs not being measured at the time of the recording. The results of this test are therefore primarily whether the calibration algorithm is able to map the environment accurately and then whether it converges to reasonable pose estimates.

3.8 Evaluation metrics

In order to reasonably enable comparing the 25 different scenarios, evaluation metrics are created. Each calibration result is a transformation T_{RF}^{alg}, which is first converted to the error transformation $T_{err} = T_{RF}^{nominal}(T_{RF}^{alg})^{-1}$. This is to make it easier to read the results. The error transformation can be described by 6 variables $(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z)$ by (2.5) and (2.6), one for each degree of freedom. This can further be boiled down to two numbers - the Euclidean distance error d_{xyz} and the angular distance error θ_{rpy}, defined as

$$d_{xyz} = \sqrt{{t_x}^2 + {t_y}^2 + {t_z}^2} \tag{3.14}$$

$$\theta_{rpy} = \sqrt{{\theta_x}^2 + {\theta_y}^2 + {\theta_z}^2}. \tag{3.15}$$

The Euclidean distance is simply the actual distance in space, disregarding the direction. The angular distance is the Euclidean distance analogue for angles. This makes it easier to conclude what natural environment features and artificial landmarks work best. Of course, a lower error corresponds to a better environment or augmentation.

4

Results

In this section, the test results for the *Hand-Eye* and *MergeMaps* methods are presented. The primary evaluation metrics are Euclidean distance error d_{xyz} and angular distance error θ_{rpy}, as defined in 3.8. Figure 4.1 shows a legend for the result tables.

Figure 4.1: The error transformation T_{err} for each calibration is broken down into two values - euclidean distance error [m] and angular distance error [°]. The left and right bars indicate how much each axis contributed to the error.

4.1 Method 1: *Hand-Eye*

In this subsection, the results from the *Hand-Eye* method are presented. Firstly, the results from calibration using simulation data for all selected locations in the quarry with various augmentation are shown. Then, the results for calibration using real sensor data are presented.

4.1.1 Environment augmentation test

Table 4.1 shows the result of 200 calibrations distributed over 25 combinations of different environments and augmentation types. By the table, it is clear that the *Hand-Eye* method is capable of producing somewhat accurate results, but may diverge. Due to frequent inaccuracies, it is difficult to evaluate patterns in the data.

Table 4.1: Environment augmentation test results for the *Hand-Eye* method. Each cell represents the LiDAR-to-LiDAR transformation error, averaged over 8 calibrations. Each row #1 − #5 represent one of the locations in Section 3.6.1.

		Augmentation Type				
		None	**Boxes**		**Cylinders**	
		0	5	10	5	10
Location Number	**#1**	2.605 22.565	0.387 2.279	0.229 1.480	0.133 0.519	0.159 0.931
	#2	4.801 35.108	1.812 7.371	0.719 2.319	2.436 14.032	1.002 5.636
	#3	0.140 0.618	0.447 3.252	0.536 4.343	0.067 0.426	0.560 4.610
	#4	1.195 9.359	0.297 2.317	0.362 1.722	0.332 2.063	0.188 1.154
	#5	7.006 52.328	0.519 2.824	0.381 2.600	2.583 20.047	0.516 4.097

4.1.2 Real-world verification test

The *Hand-Eye* algorithm seems to perform similarly for the real-world data as for the simulated data. Table 4.2 shows that rotations are reasonably but not sufficiently accurate and Euclidean translations seem off by far. Interestingly, the translation errors along axes x and y are small (0.22 and 0.013 meters respectively) while the error along the z-axis is huge (32 meters). Although several poses are rejected, the coherence of the accumulated maps in Figure 4.5 suggests that the error arises after their creation.

Table 4.2: Table describing real-world performance of *Hand-Eye* calibration algorithm.

	Calibration Result	Poses Accepted [#/tot]
Real-world location	32.1351 2.424	71/98

4.2 Method 2: *MergeMaps*

In this subsection, the results from the *MergeMaps* method are presented. Firstly, the results from calibration using simulation data for all selected locations in the quarry with various augmentation are shown. Then, the performance for calibration using real sensor data is shown.

4.2.1 Environment augmentation test

In Table 4.3 the results for the LiDAR-to-LiDAR transformation error $T_{err} = T_{RF}^{nominal}(T_{RF}^{alg})^{-1}$ is shown, averaged over 8 calibrations. This is done for every location #1 − #5 for no augmentation and for 5 and 10 boxes and cylinders respectively.

Here the RMSE is also calculated for every row and for every column in the table. Notice that location #3 has the lowest RMSE for both translation and rotation. Also, 5 and 10 boxes have lower RMSE for both translation and rotation compared to cylinders. It can also be seen that the RMSE is larger for the case when no objects are used.

Table 4.3: Environment augmentation test results for the *MergeMaps* method. Each cell represents the LiDAR-to-LiDAR transformation error, averaged over 8 calibrations. Notice, when looking at the coloured bar, the yaw angle contributes mostly to the error. However, for the translation error contribution, it is more varied, but z contribution is always small

		Augmentation Type				
		None	**Boxes**		**Cylinders**	
		0	5	10	5	10	**RMSE**
Location Number	**#1**	0.194 0.479	0.079 0.250	0.032 0.189	0.035 0.140	0.040 0.133	0.099 0.277
	#2	0.178 0.141	0.034 0.281	0.047 0.189	0.093 0.309	0.059 0.291	0.102 0.259
	#3	0.033 0.181	0.065 0.163	0.046 0.274	0.022 0.070	0.040 0.266	0.044 0.206
	#4	0.132 0.227	0.040 0.271	0.046 0.291	0.046 0.316	0.029 0.203	0.072 0.280
	#5	0.198 0.699	0.047 0.126	0.058 0.350	0.136 0.407	0.104 0.481	0.124 0.464
	RMSE	0.172 0.426	0.057 0.234	0.048 0.275	0.080 0.284	0.062 0.304	

In the box-plot diagrams in figures 4.2 and 4.3 the simulation result is shown to better visualise the results. In the upper diagram the Euclidean distance error d_{xyz} is shown and in the lower diagram the angular distance error θ_{rpy} is shown. In the diagrams, every box shows 8 simulations. This is done for each driving location $\#1 - \#5$ with no landmark object types and for 5 and 10 boxes and cylinders respectively. The average in every box is indicated by a horizontal line, whereas the median lies where the light grey and dark grey areas meet in the box.
Notice that location #5 always seems to give worse performance compared to the other 4 locations. It's also clear that no objects most often perform worse than if objects are presented. Other than that it seems that augmentation with boxes is the way to go by just looking at the diagrams, this is especially true for translation but not as clear for rotation.

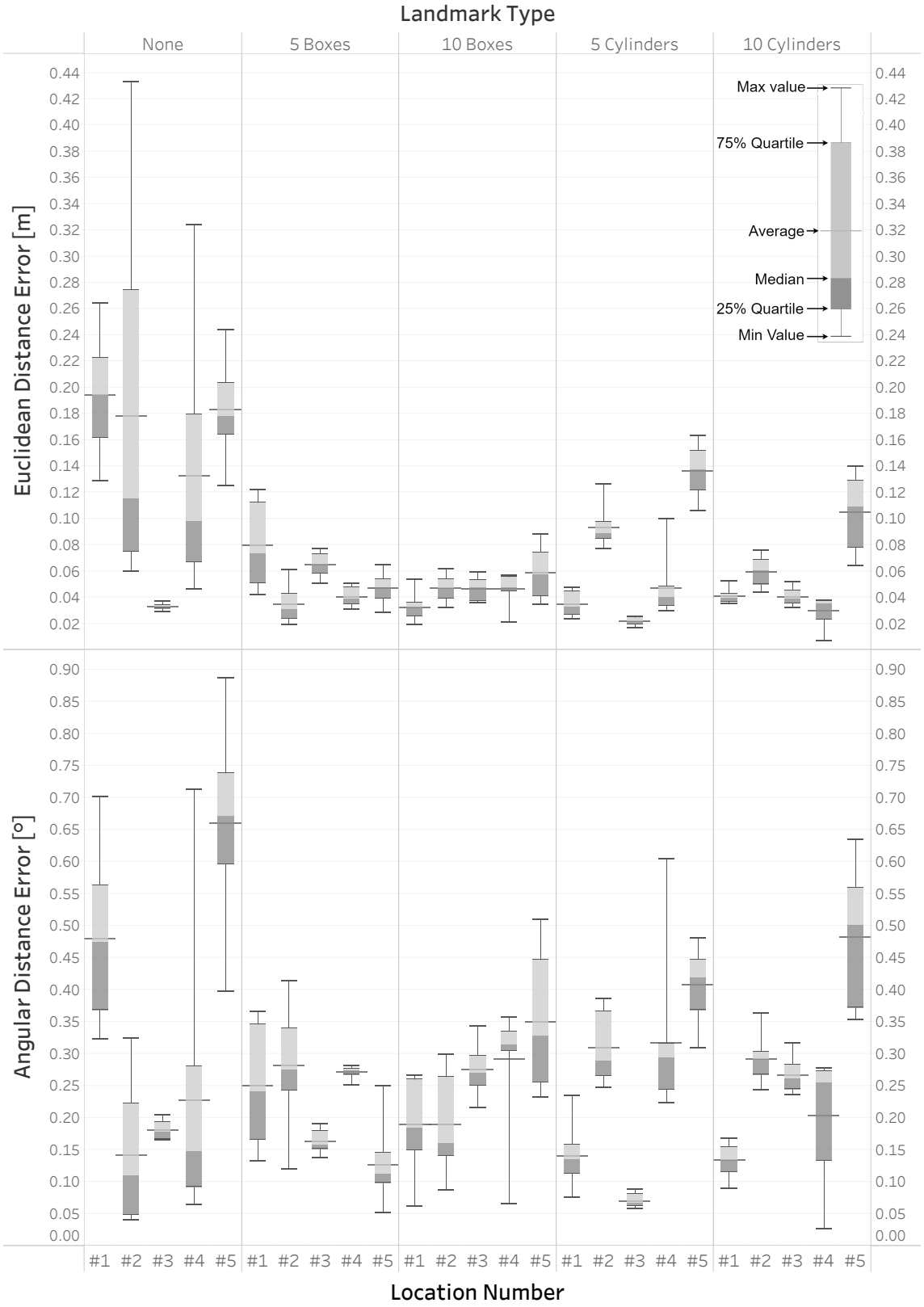


Figure 4.2: Box plot showing the calibration errors d_{xyz} (top) and θ_{rpy} (bottom). Each box represents 8 data points.

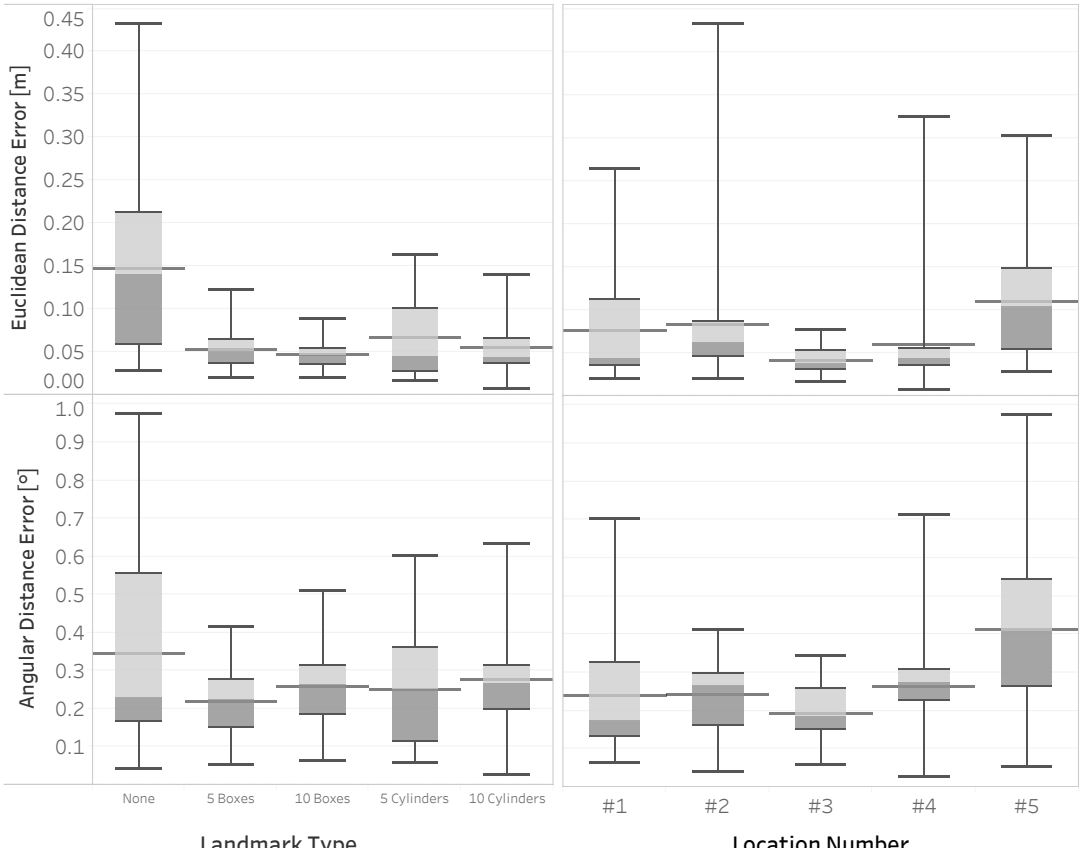
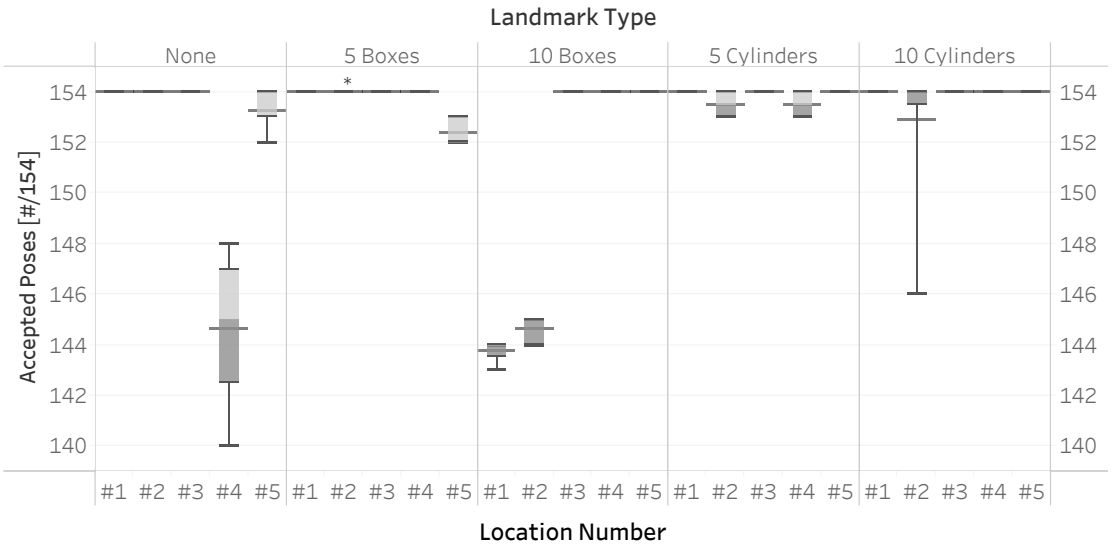


Figure 4.3: Box plot showing the calibration errors d_{xyz} (top) and θ_{rpy} (bottom) aggregated over the landmark and location categories respectively. Each box represents 40 data points.

In the boxplot diagram in Figure 4.4 the simulation result shows the number of poses accepted (not rejected) to use for the calibration. It can be seen that for some locations not all poses are accepted. The mapping procedure eliminates the poorly aligned poses from the mapping process.



* One outlier data point (21 accepted poses) was excluded for readability.

Figure 4.4: Box plot showing the number of accepted poses (out of 154). Each box represents 8 data points.

4.2.2 Real-world verification test

The calibration results for *MergeMaps* on real-world data (Table 4.4) converge toward a plausible estimate. This indicates that the algorithm may be promising also in reality.

Table 4.4: Table describing real-world performance of *MergeMaps* calibration algorithm.

	Calibration Result	Poses Accepted [#/tot]
Real-world location	0.1235 0.676	71/98

Figure 4.5 shows that the *MergeMaps* algorithm fulfils its intended function of aligning the point cloud maps. Although this is not equivalent to the output is a perfect calibration estimate, it further suggests that the simulation results may transfer well to the real world.

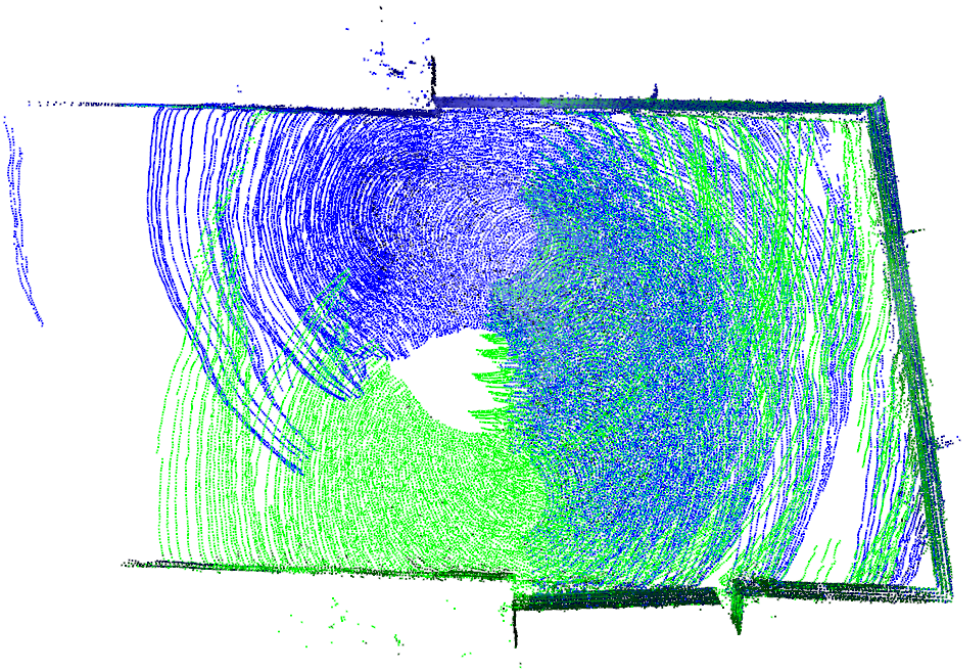


Figure 4.5: The front (blue) and rear (green) point cloud maps naturally align when using the transformation obtained from the *MergeMaps* algorithm.

5
Discussion

5.1 Method comparison

The outcome of using the *Hand-Eye* method for calibration can be read from Table 4.1. The results vary significantly, occasionally yielding reasonably good results. Yet only 50 out of 200 *Hand-Eye* calibration results are better than the worst *MergeMaps* result. Moreover, a mere amount of 6 *Hand-Eye* results are lower than 0.2°. The corresponding number for *MergeMaps* is 146. The average errors are 8.16°/1.18m, the largest 146.84°/17.29m and the smallest 0.016°/0.025m.

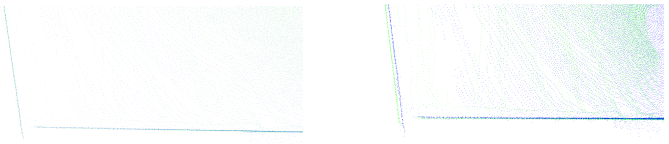
MergeMaps produces 200 results with an error of 0.27°/0.074m on average, a lowest error of 0.026°/0.007m and never exceeding 0.98°/0.43m. Even the latter outperforms 75% of the *Hand-Eye* results. These results show that the *MergeMaps* method outperforms the *Hand-Eye* method both in robustness as well as in accuracy. It should be said that *Hand-Eye* produced the overall best result with respect to rotation. However the amount of results with almost that accuracy is very low for *Hand-Eye* but high for *MergeMaps*, this is considered to have happened by chance.

5.1.1 Real-world performance

Seemingly, the performance properties of the algorithms are probable to transfer to reality as well. Accurate maps are produced also when the input is real data. The *MergeMaps* algorithm result on the real-world dataset is plausible and even on the same level of accuracy as in the simulations. While this is also true for the *Hand-Eye* algorithm if disregarding the 32 meter z-axis translation error, one should be more careful drawing any conclusions from this. It is nevertheless an indication that the behaviour of the *Hand-Eye* method is unreliable also in reality.

There are more differences between the two tests than one originating from simulations and the other from the real world. With this in mind, the fact that *MergeMaps* seems to show similar accuracy in reality as in simulation should be regarded with a high degree of uncertainty. It can be argued that the real-world environment is more forgiving since it consists of large, planar walls in a U-shape. On the other hand the vehicle is not driving in a full circle, which should make the mapping and merging of the maps more difficult. Last but not least, the exact ground truth transformation for the real-world data is unknown.

Some things can however be expected in terms of accuracy of the *MergeMaps* algorithm in reality. The accuracy with which the point cloud maps are merged is higher than the output error. This can be viewed in Figure 5.1, which shows how the point clouds align when using the algorithm output, compared to when using the "ground truth" $T_{RF}^{nominal}$. This clearly disregards earlier stages of the algorithm.



(a) T_{RF}^{alg} **(b)** $T_{RF}^{nominal}$

Figure 5.1: The mapping stage is more accurate than the indicated algorithm error for the real-world data.

On the whole, the results indicate that *MergeMaps* is capable of producing plausible LiDAR-to-LiDAR pose estimates also in reality. To what extent the accuracy of the results may transfer to a real setting is not possible to say with a high degree of certainty. The *Hand-Eye* algorithm seems as unreliable as previously suggested.

5.1.2 Calibration robustness

The *Hand-Eye* method frequently produces bad, sometimes even entirely nonsensical results. Out of the 200 calibrations that were performed, 6 results were in the latter category. The translation errors for these are several times greater than the distance between the LiDARs. The rotation errors are higher than 90°. An additional 22 measurements are wrong by more than 1 meter and 10°. That can be compared to the *MergeMaps* method, which in the 200 runs performed never superseded an angular distance error of 1°. Also, the translation errors were relatively small. Thus the *MergeMaps* method can be considered far more robust and reliable.

Still, there are patterns in the *Hand-Eye* results. The results with exceptionally high errors originate from the same recordings, and a majority of the 25 recordings do not produce errors even close to this magnitude. The recordings for which the *Hand-Eye* method frequently gets it wrong largely coincide with the recordings producing the most uncertain *MergeMaps* results. This is probably because the two methods make use of the same mapping results. The number of accepted poses seems unrelated to these less certain recordings.

One issue with the algorithm is that it is quite sensitive to inaccurate initialisation of the first two point clouds. If they do not align well, the algorithm may have a more difficult time converging to a good solution.

5.1.3 Verdict

Despite initial high hopes for the *Hand-Eye* calibration method, the results suggest that the *MergeMaps* method is the better calibration algorithm for the conditions given. It is significantly more accurate and by far more robust. The disproportionally large errors produced by the *Hand-Eye* algorithm further dissuade from using it in this context. Henceforward, only *MergeMaps* will be discussed unless otherwise indicated.

5.2 Environment Augmentation

Based on the newly made verdict, the following will primarily be based on the results from method 2, *MergeMaps*.

5.2.1 Comparing different environments

Despite the locations being rather similar, differences in the calculated RMSE values for the error evaluation metrics state that some locations exhibit better performance than others. Location #3, for example, stood out with the lowest RMSE values for both rotation (0.206°) and translation (0.044m). The errors in location #3 were 64.5%/55.6% lower than location #5, which had the highest RMSE values for both rotation and translation. In Figure 4.2 it can be seen that for example location #3 performs well even without environment augmentation with a mean distance translation error around 3 cm and a mean angular error just below 0.2°.

Adding landmarks does not reduce the calibration error in location #3 for two out of four augmentation variants - rather the opposite. In the two remaining cases, adding 5 boxes improves rotation but not translation, and adding 5 cylinders reduces the error for both, thus improving overall. From a probabilistic point of view, if selecting one of the four configurations at random, the landmarks would in this case be redundant, since using landmarks would more likely than not lead to a less accurate estimate. Moreover, just 5/25 scenarios give smaller rotation errors than location #3 without landmarks, and only 2/25 give smaller translation errors. One could conclude that location #3 is an environment where landmarks are largely redundant.

In order to further refine our understanding of the environment, we look at the result of Figure 4.2 when using no landmark augmentation, the mean distance and angular error values are ranked from lowest to highest. The distance error is ranked (#3, #4, #2, #1, #5) and the angular error is ranked as (#2, #3, #4, #1, #5). Location #2, #3 and #4 seem to give better performance than #1 and #5. This is interesting since #1 and #5 are located just a few meters away from each other. If comparing the features from all locations, some conclusions can be inferred. Common properties for different locations that seem to be favourable (+) and unfavourable (−) respectively follow here.

(+) Location #3 (and to some extent #4) have large, proximate, irregular, near-vertical mountain walls. This makes it easy to get a well-sampled surface with higher point density which can represent the surface more accurately. When downsampling the point cloud the new cloud will more accurately represent the true surface when it has access to more sampling points. Leading to more accurate approximations of local planes with the point-to-plane ICP algorithm.

(+) Location #1 is better than #5, the only difference is that #1 has higher driving route elevation. This may be beneficial for creating a point cloud map where points are evenly distributed, as opposed to grouped in channels.

(−) Location #4 and #5 have more distance to the surrounding mountains. This could be explained that the LiDAR beam diverges over a larger area resulting in lower point density. In reality, the returning beam would also be weaker.

(−) Location #4 exhibits a large, featureless, open area in one direction. Thus, there are not enough objects and structures reflecting the beams back to the LiDAR.

(−) Location #1 and #5 showcase gently sloping mountains in the surroundings, seemingly resulting in worse measurement data. The low angle of reflection of the slopes results in less dense point clouds.
h
(−) Location #1 and #5 have trees nearby. This may affect the measurements negatively, probably due to tree canopies being very unstructured and not point-dense. This may result in outliers.

5.2.2 Adding artificial landmarks

Observing RMSE values in Table 4.3, for example, if having 5 cylinders (which gives the largest RMSE value for translation with augmentation), the decrease in RMSE is 53.4% compared to having no object presented at all. Compared to the RMSE of the rotational component, it is a decrease of 33.3%. If looking at the RMSE values for 5 boxes, (which has the largest improvement in rotation error) to using no landmark augmentation. It can be said that landmarks for augmentation reduce calibration errors with up to 45% for rotation and 66.9% for translation on average. The largest errors reduce even more - 57.7% in rotation and 71.8% in translation - indicating that the calibration consistency also improves. Hence it can be concluded that in general landmark augmentation makes a difference compared to having no augmentation at all.

Boxes with flat surfaces, sharp edges and corners seem preferable over rounded shapes such as cylinders, giving the lowest error mean values for both 5 and 10 objects in Table 4.3. If taking the error reduction fraction for both amount categories and then averaging, the error reduction for boxes compared to cylinders is

25.7%/13.6%. It should be noted that this difference while not huge may be significant. The result suggests that the features of the box are more helpful over round objects for increased algorithm performance. Even if the alignment seems to be quite good for both objects in Figure 5.2, it can be seen that not all points align in the cylinder as well as they should. The points are a bit more spread compared to the box shape, this could be one explanation for why calibration with boxes has better performance.

(a) Box shape **(b)** Cylinder shape

Figure 5.2: Top-view of an aggregated point cloud for a box corner and a cylinder. The reconstruction of the cylinder shape is pretty good, but not as good as the box corner.

Regarding the number of objects, it can be determined that the translation component will be 12.3% smaller with 10 objects but the rotational component will be 19.1% smaller for 5 objects regardless of object type used, as can be seen in Table 4.3. This behaviour is a bit unclear, but since orientation is more important than translation, opting for 5 objects seems to be the better alternative. Also, having to use fewer objects in real physical driving scenarios is preferable.

One may argue that object size is important. Although the size of the objects was not varied in the study, the objects were 2 meters high and 1.5 meters wide. They were placed as close to the vehicle as possible within the scoped limits, plus margins. Considering larger objects a physical constraint, the prospect of differently sized objects can thus be considered reasonably covered. What can be gained from testing smaller landmark versions is quite likely bounded within the presented calibration results.

5.3 Method comments

No method is perfect, therefore this section exists - a critical reflection on the method used. The following sections intentionally reflect the sections of Chapter 3: Method, discussing choices made, notes on preconditions and other relevant commentaries.

5.3.1 Data collection

While CARLA simulations are not identical to reality, they come with the great benefit of making ground truth available. From a research perspective this is vital. Regardless, what is known to differ between the artificial world and the real one should be kept transparent.

General modelling errors worth mentioning can be divided into world modelling errors and vehicle modelling errors. Both will be kept brief in relation to the amount of work put into them internally at V.A.S prior to this work. To keep things relevant and disclosable, the focus will be on simplifications in the world model and inputs to the calibration algorithm.

World modelling errors are straightforward - the world in which the simulation occurs is a triangle mesh, a surface built up by small, planar, triangular elements. In this context, the scale of these elements is often on the level of meters, as can be seen in Figure 5.3. This has implications for vehicle movement and, more importantly, LiDAR measurements and mapping capabilities. Since quarry environments generally have a rough texture, this would implicate that LiDAR beams sometimes hit small indents in the rock and sometimes hit small protrusions. If those indents and protrusions are modelled as one flat surface, the simulation measurements will be more consistent as a result. Mapping capabilities should also be enhanced by the features that the planar elements and intersecting edges provide, since such features ought to help the ICP to converge.

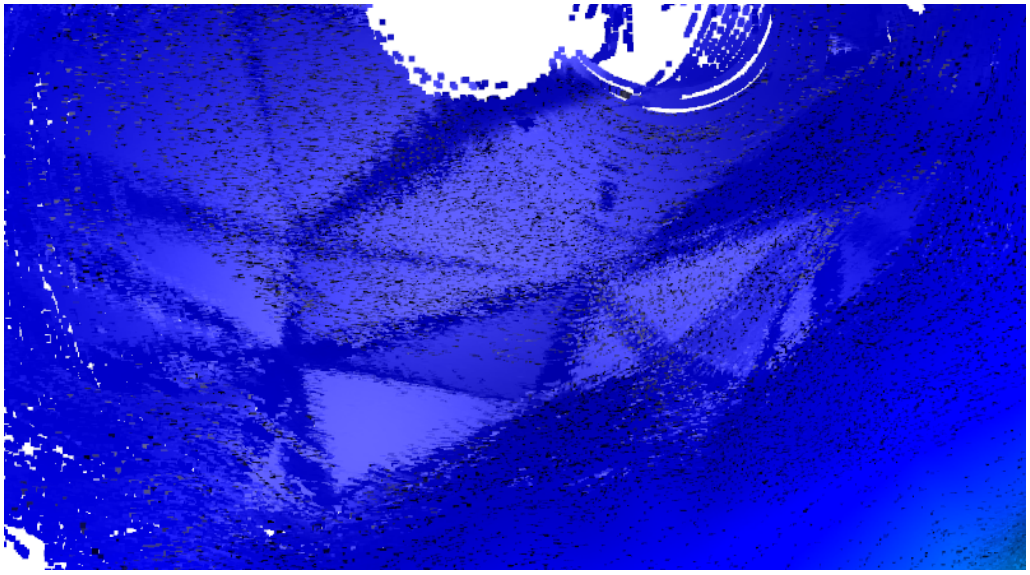


Figure 5.3: The underlying triangle mesh that builds up the map is visible even after applying the mapping algorithm. For reference the diameter of the half circle at the top of the picture is roughly 3 meters in diameter.

The odometry data can be compared to a ground truth pose provided by CARLA. During the tests conducted, the accuracy is almost exclusively below ±1 cm and

±1°. Since the odometry data only serves as an input to the ICP algorithm this is unlikely to contribute to any errors in the algorithm at all. Then again, it is important to keep in mind that the odometry estimate is based on a fuse of data from other sensors, that are also simulated. These are sampled by taking the ground truth value and then adding some Gaussian distributed noise. So while some noise is modelled, the measurements still likely suffer more from disturbances and hardware errors in reality than the simulations indicate.

The simulated LiDAR data differs from the real data by always measuring the distance perfectly, without disturbances, hardware errors or sample timing errors. This preserves the meshing effect displayed in Figure 5.3 and makes it stand out clearer than it would have done otherwise. Figure 5.4 shows the effect from another perspective.

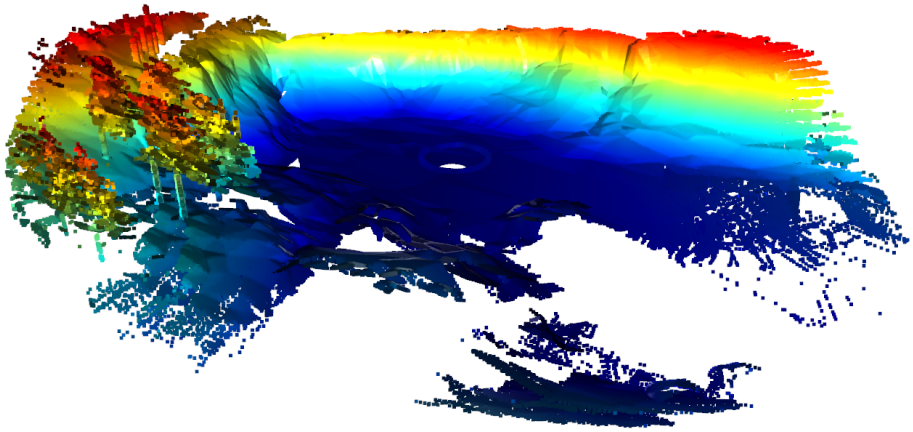


Figure 5.4: The triangle mesh clearly visible in a point cloud picture showing location #5.

5.3.2 Data pre-processing

The data pre-processing step was effective and well-motivated. There is mainly only one algorithm simplification that needs commentary - the normal estimation algorithm. While the algorithm is by no means ideal, it proved fully decent at estimating normals, and so served its purpose. Therefore no further work was put into optimising this step.

The normal estimation exhibits two slightly undesired properties. As it seems, neither of them has a profound impact on the ICP algorithm. The first is that the boundaries where the point cloud has been divided and then merged again clearly show. This is viewed in Figure 5.5. When combined with point outlier rejection this could potentially have a significant impact. The other undesired property is related - it is clear from Figure 5.5 that several normals of the green points are weighted downwards due to some ground points being inside their normal estimation radius.

The same phenomenon, while more limited, can be seen also for normals of blue points close to the ground plane. Overall this leads to the normals around corners being averaged, as if the corners were rounded. This may have a minor impact on the point cloud alignment.

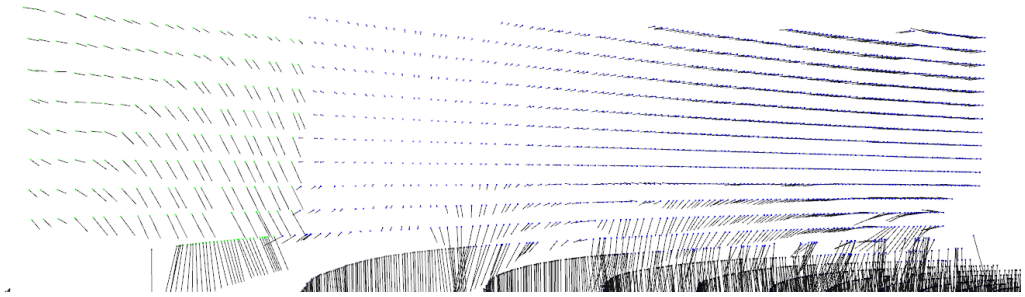
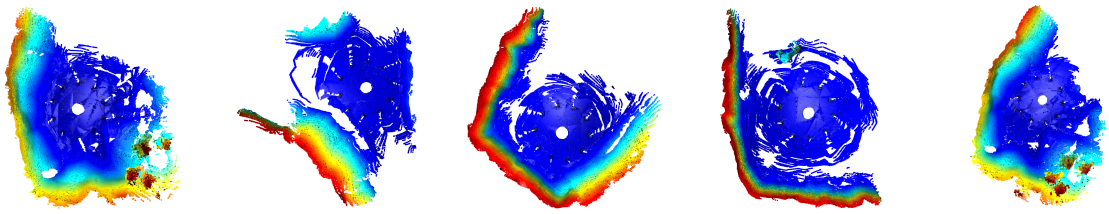


Figure 5.5: Some minor side effects arise when normals are estimated separately for the blue and green partitions of the point cloud. The normal estimation radius is smaller for the blue partition since it is closer to the vehicle.

5.3.3 Mapping procedure

The mapping algorithm produces well-aligned maps of high quality. That is determined from visual inspection of the maps, analysis of the ICP fitness results and analysis of potential drift of the map. Over 200 calibrations, only one map was left incomplete, yet accurate. Moreover, the sweep algorithm as well as pose rejection and its implications will be discussed.



(a) Location #1 **(b)** Location #2 **(c)** Location #3 **(d)** Location #4 **(e)** Location #5

Figure 5.6: Mapping results for the front LiDAR for each location. Top view, color representing height. More mapping results are attached in Appendix D.

Although visual inspection does not give all the details of how good an alignment is, it is still a simple and quick method. It can be used to detect complete misalignments and also relatively subtle ones, for example the one shown in Figure 5.7. Although not all 400 point cloud maps (200 per LiDAR) were meticulously scrutinised for the kind shown in the picture, the ones known to be prone to such misalignments were. None contained such misalignments. That is, it was not possible to distinguish one stitched point cloud from the rest in the merged map.

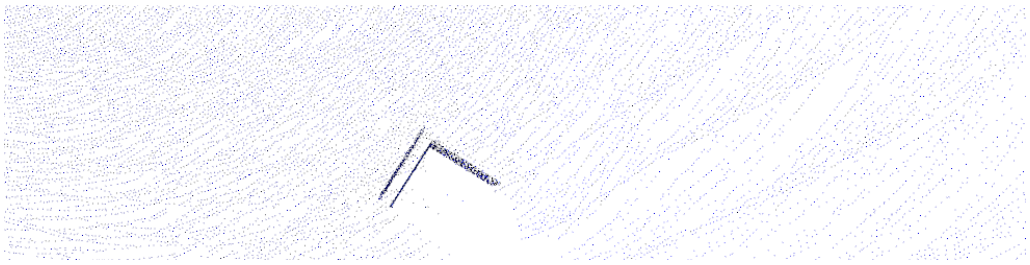


Figure 5.7: Close-up of a point cloud map with one misaligned point cloud. This is very clear by the side of the box ending up slightly off from the other readings of the box.

The fitness output of the ICP algorithm can also be used to say something about the quality of the map. The fitness average for each calibration is almost exclusively above 90%. While the maximum correspondence distance parameter of the ICP algorithm plays a role here, the number speaks for the mapping working well.

Accumulating and downsampling a point cloud map repeatedly poses a risk of having a point cloud drift. In other words, the map produced may look very accurate, but a comparison to the first point cloud used to build the map shows that it no longer aligns. Only indicative tests have been conducted on this issue. These show that there may be a small drift of less than one centimetre in translation.

The sweep algorithm is largely unnecessary for the case when the LiDARs are in their nominal position. The prestudy included in Appendix A with LiDARs in other positions than the nominal shows that the algorithm works as intended. The sweep range may be altered after need. There is room for developing the sweep algorithm to also estimate the LiDAR-to-base transformations T_{BF} and T_{BR}. This was considered interesting but out of scope due to time limitations.

No poses were rejected in a majority of the 200 calibrations made. This can be read from Figure 4.4. For 3 of the 25 recordings, around 10 poses were consistently rejected. Almost all of them were due to geometrical constraint violation. No explicit investigations were made regarding to what extent the pose rejection contributes to the calibration results. If the geometrical constraints are violated, at least one of the point clouds captured should reasonably not align with the rest. This was considered enough to motivate pose rejection.

The downside of pose rejection is that a full map does not accumulate if too many poses are rejected. It does not come as a surprise that the more subsequent poses that are rejected, the more likely the next pose is to be rejected. This way the mapping may at some point diverge and not register any new poses (at some point rejection will also be based on low fitness) until the last few poses before completing the circle.

This happened once during the environment augmentation test, resulting in a mere 21 accepted poses at location #2 and 5 boxes. The underlying reason is likely tied to the environment opening up for the front LiDAR, simultaneous to a geometrical pose rejection, making registration more difficult. Regardless, the transformation estimate of that calibration is not bad at all. This should however be taken with more than a grain of salt. Since the point cloud maps do not overlap in their initial configuration (remembering that the initial transformation is a perturbed version of $T_{RF}^{nominal}$), no alignment is performed. This is important to take notice of when using the algorithm.

5.3.4 Method 1: *Hand-Eye*

The *Hand-Eye* method produces somewhat puzzling results. For the circumstances given it performs poorly both in terms of accuracy and robustness. Yet it is frequently used for sensor calibration for other ground vehicles in the literature [4, 5, 7]. As pointed out in [7], planar movement renders translations along the z-axis unobservable. This is the Achilles heel for using the *Hand-Eye* algorithm to calibrate LiDARs on a dumper.

The unobservability can be explained in two steps. The first is simple - the *Hand-Eye* method does not work if the vehicle is just driving in a straight line. This is hopefully made clear by Figure 5.8. Notice that the LiDARs can be placed anywhere on the vehicle as long as they point in the same direction and still the value of T_{F_+F} will be the same. Hence all translations (x, y and z) are unobservable when driving straight ahead.

Now for the second step. Even though ground vehicles turn, since they only move in two directions - x and y - they are victims of a special case of the same phenomenon as when driving straight. The LiDARs may be placed on different heights on the TA15, and still produce transformations T_{F_+F} very similar to what they would have been if they were level. The z-translation is unobservable due to the lack of changes in roll and pitch. Conversely, small disturbances in roll and pitch in the input data scale immensely when the algorithm optimises the z-translation based on how roll and pitch intervene with the other degrees of freedom.

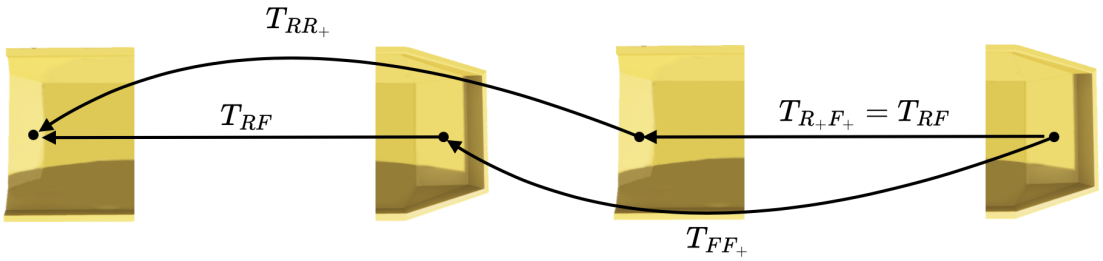


Figure 5.8: The hand-eye problem is not solvable if the vehicle drives in a straight line. The length of the TA15 may be altered freely without affecting T_{FF_+} and T_{RR_+}.

In [7], the z-translation is set to 0 in table III showing calibrated extrinsics. Still the calibration error is 1.433 meters and 3.217° - similar to if not larger than the results in this thesis.

5.3.5 Method 2: *MergeMaps*

The *MergeMaps* method is based on keeping things simple. It is not obvious what contribution this stage has to the calibration error, however it is possible to see errors on the level of a few centimetres by visual inspection (as visualised in Figure 5.1). Since such visual flaws have not been seen throughout the development of the *MergeMaps* algorithm, it is likely to be small. A very conservative upper bound would be 5 cm and 0.2°.

For the kind of off-line on-site calibration researched in this thesis, the only significant limitation may be that it requires both LiDARs to get a full overview of their surroundings, 360° horizontally. As long as there is a space large enough for the vehicle to circle around, this does not limit either.

5.3.6 Test setup

Scrutinising the design and execution of the environmental augmentation tests, it seems to the authors that the methodology was consistent and motivated by the problem formulation and scope. Still some points are worth discussing. These are the selection of environments, augmentations and other operating conditions for the tests. Also, some inconsistencies arising from the manual recording procedure in CARLA are brought up.

The selection of environment is considered well-grounded since all simulations were made in a digital twin of a real quarry. While the possibility exists that there are other better environments in this (or other similar quarries), the *MergeMaps* method will likely not perform significantly differently. This is motivated by the consistent results over the different locations. For Daniliidis' method, there is a good chance that utilising the three dimensions more would result in more consistent results and maybe also improve accuracy. A concrete example of this would be driving in a circle penetrated by a ridge, which would result in two uphills and two downhills in one lap. The steeper the ridge the better in theory, however maybe not in practice, since sensor readings will not be as accurate. For odometry estimates, this is very likely due to velocity changes and slipping. For LiDAR estimates it is not unlikely due to parts of the environment being occluded by the ridge.

The selection of landmark objects was calculated, considering that the usage of reflexive markers was out of scope. The selection of two landmark types (box, cylinder) in two configurations (5, 10) seemed appropriate and enabled evaluation of object shape and amount. The possibilities with regards to size can also be considered covered as discussed in Section 5.2.2. The size of the selected landmarks was adjusted to be similar. Since the difference in width of the boxes depending on viewing

angle is larger than the difference between the shapes, the influence it has on the comparison of object shape should be considered small.

Other operating conditions such as throttle, turning radius and runtime clearly affect the results. The prestudy included in Appendix A allowed the results to be more relevant, since close-to-optimal operating conditions would also be used for calibration in reality. One can argue that the comparison between spawn points would have been fairer if all data had been recorded with a constant setpoint for velocity instead of a constant throttle. This is a fair point, however the speed controller would also mean yet another source of potential errors. Also, from an accelerations point of view, the constant throttle is likely less aggressive resulting in a smoother behaviour of the vehicle. So while a constant throttle may result in laptime deviations for different spawn points the choice can still be justified.

Finally, there are some inherent consistency problems involving CARLA. Two simulations (#2, 10 cylinders and #5, 5 boxes) had to be re-recorded due to partially corrupt files. Analysis of the mcap file showed at least signs of inconsistent sampling. The calibration results of the re-recorded versions were significantly better as well, level to the other results. Other than that, the manual recording procedure also resulted in two errors tied to the initial position of the vehicle. Firstly the vehicle spawn position relative to the landmarks was not exact but approximated in each location. Secondly the steering and throttle parameters were not set upon spawn, and the amount of time elapsed before they were set was different every time. Hence, if the vehicle spawned in a slope, the likelihood of making a non-centred circle was significantly higher.

5.4 Implementation

MergeMaps has potential, but what does it mean further down the line? First, this question is asked from an industrial point of view. It is followed by a discussion regarding how the method may generalise to other setups.

5.4.1 On-site implementation requirements

Implementing a calibration station on-site would not require a lot, and the calibration in itself would be fairly easy. The bulk of activities and development surrounding the calibration is likely the bigger hurdle. To perform the calibration, a few things are required: space, time, personnel, software and perhaps physical landmark objects.

- Space: Roughly 25 meters in diameter, driveable terrain, with the type of surroundings concluded preferable in this thesis. Most importantly, the surroundings are largely static and there are no large, vast, featureless strips along the horizontal FoV.
- Time: Most time could potentially be spent on having the TA15 break its normal routine and initiate the calibration. Driving the TA15 to the calibration site could be done manually via remote. The data collection itself takes less

than a minute, then the processing of the data less than 10 minutes. Also, it is advisable to have a confirmation step where personnel concludes that enough poses have been accepted and that the calculated transformation is plausible. Depending on how user-friendly the system is this could probably take anything from one minute to one hour.
- Personnel: Maybe needed for manually driving the TA15 to/from the calibration site. Needed to set up props, if any. Personnel also starts the calibration and checks the calibration result.
- Software: To reduce the calibration time, user-friendly software is very important. The questions to address are:
  - How should the TA15 drive to/from the calibration site?
  - How is the calibration algorithm started?
  - How do the personnel get to analyse and accept/reject a calibration?
- Landmarks: Maybe not needed. The size of the ones in the thesis was approximately 1x1x2 meters. The landmarks can be made hollow and without rear surfaces since only surfaces facing the vehicle will matter.

In the end, the results presented in the thesis suggest that the rotation provided by the algorithm would always be useful for calibrating the LiDARs, while the translation not necessarily is accurate enough to always rely upon. This however correlates well with the fact that the LiDAR orientation likely will be subject to change over time to a much larger extent than the position. Therefore, a reasonable strategy could be to calibrate the orientation and leave the position at its nominal values (determined by the factory calibration). One exception where position calibration would be useful is when the position is known to have altered from its nominal values. Examples include if the LiDAR mount gets bent or if an entirely different LiDAR mount is installed.

5.4.2 Transferability to other setups

The proposed *MergeMaps* algorithm (referring to only Section 3.4) can be applied to calibrate sensors that sample point clouds, or through processing can turn their samples into point clouds. This includes but is not limited to LiDARs, RaDARs and depth cameras of various kinds. It can be used for sensors with or without overlapping FoVs. Likely, it is even possible to calibrate a RaDAR and a depth camera with non-overlapping FoVs. Extending the algorithm to work for more than two sensors is also possible. The simplest way this can be done is by chaining the point cloud merges, in the same way as the mapping was done here. For the case of 2D-LiDARs it may well work provided the measurements are combined with motion data from an IMU or similar.

There should be no limitations on the algorithm regarding the type of setting, as long as reasonable point cloud maps are obtained. Ground, aerial, water and submersible vehicles could use it in theory. The most promising use is likely for ground vehicles since the ranges are short enough for the point cloud sensors to provide useful data to build a map out of. For more urban environments, one might consider a feature-based registration algorithm instead of point-to-plane ICP.

6
Conclusion

The developed *MergeMaps* algorithm reliably estimates the relative pose T_{RF} between the two LiDARs on the TA15 with sub-degree accuracy. Reasonably, this is good enough to be useful also in reality. The *Hand-Eye* method is considered not suitable for the posed problem, and therefore the presented findings of the environmental study disregard the results associated with that method. All errors mentioned below refer to RMSE values, except for overall averages, minimum values and maximum values. Finally, thoughts on further work are briefly brought to light.

6.1 Algorithm capabilities and limitations

MergeMaps is a simple but reliable calibration method, estimating T_{RF}. Over the 200 calibrations of the environment augmentation test, the rotational calibration error never exceeded 1 degree. Averaging these results, *MergeMaps* yields a calibration error of 0.27°/0.074m. The highest errors are 0.98°/0.43m and the lowest are 0.026°/0.007m. The algorithm generally performs better with landmarks present, but not necessarily. The aggregated rotational calibration error is approximately 0.5° and a translation error of around 0.17m without landmarks. A possible limitation of the algorithm is that it requires both LiDARs to individually map their entire surroundings, for example by making a 360° turn.

The *Hand-Eye* method is outperformed in 150 out of 200 calibrations by the least accurate *MergeMaps* result. Thus, *Hand-Eye* is the least robust method by far. The average error for *Hand-Eye* is 8.16°/1.18m. The largest errors are 146.8°/17.29m and the smallest are 0.016°/0.025m. Although this is the lowest value for rotations overall, it is not too unlikely to be coincidental; a mere amount of 6 of *Hand-Eye* results are lower than 0.2°. The corresponding number for *MergeMaps* is 146. The mostly low accuracy and poor performance may be explained by the z-translation of T_{RF} being unobservable for planar calibration trajectories. This suggests that the *Hand-Eye* method is unsuitable to use as a means to estimate T_{RF}.

6.2 Environment augmentation

Using landmarks reduces calibration errors (RMSE) with up to 45% in rotation and 66.9% in translation for *MergeMaps* when using 5 boxes placed around the vehicle in a circle. The largest difference is however in how the maximum error value is reduced - by 57.7% in rotation and 71.8% in translation. Overall, the box-shaped landmarks give lower RMSE than the cylinders by 25.7%/13.6%. They also give more consistent results, being bounded by the best and worst calibrations using cylinders. For both the box and cylinder averages, it seems that 5 landmarks give better rotations by 19.1% and that 10 landmarks give better translations by 12.3%.

The best location (#3) gave calibration errors of 0.206°/0.044m. Compared to the worst location (#5), this error was 55.6% lower for rotations and 64.5% lower for translations. Still, all locations were promising in terms of producing a good calibration result. The selection of a great calibration location over a good one can therefore not be understated - it may render landmarks redundant. This is demonstrated by location #3, which outperforms most calibrations in other locations and for which adding landmarks decreased accuracy in half of the cases.

The environmental key characteristics that make up a great calibration environment are suggested to be large, proximate, irregular, near-vertical walls such as those in location #3. The landmark comparison speaks for corners, edges and planar features. On a much larger scale, these are also properties of location #3. Naturally, solid objects such as rocks are preferred over canopies. It is also not strange that characteristics to avoid include vast, open and featureless areas, such as that in #4. One can however notice that all locations but #2 include such areas, to some extent.

6.3 Further work

For V.A.S, the *MergeMaps* method should work for their purpose since they are most interested in finding the relative yaw rotation of the LiDAR. An even simpler algorithm would likely work since orientation and especially yaw are of significantly higher interest than the other DOF. The used method estimates 6 degrees of freedom which is a waste of computation power and more complexity is introduced.

For research purposes, it would be interesting to see if the hand-eye problem can perform better on a bent torus-shaped path. Since the sparsity of measurements in z-height seems to be one of the problems with the current implementation. Another method that would be interesting to look into is pose graph optimisation when driving in a circle since constraints on the transform can be determined when driving in the circle, and then closing the loop when a full lap has been driven. Another way to improve the calibration algorithms could be to also include measurements from RaDAR, IMU, and camera sensors.

Bibliography

[1] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López, and V. Koltun, "CARLA: An Open Urban Driving Simulator," *ArXiv*, vol. abs/1711.03938, 2017.

[2] N. Ertugrul, A. P. Kani, M. Davies, D. Sbarbaro, and L. Morán, "Status of Mine Electrification and Future Potentials," in *2020 International Conference on Smart Grids and Energy Systems (SGES)*, 2020, pp. 151–156. DOI: 10.1109/SGES51519.2020.00034.

[3] *OS1 - Mid-Range High-Resolution Imaging Lidar*, [Online]. Available: https://invensense.tdk.com/download-pdf/iam-20680ht-datasheet/.

[4] S. Das, N. Mahabadi, A. Djikic, C. Nassir, S. Chatterjee, and M. Fallon, "Extrinsic Calibration and Verification of Multiple Non-overlapping Field of View Lidar Sensors," in *Proceedings - IEEE International Conference on Robotics and Automation*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 919–925, ISBN: 9781728196817. DOI: 10.1109/ICRA46639.2022.9811704.

[5] M. He, H. Zhao, F. Davoine, J. Cui, and H. Zha, "Pairwise LIDAR calibration using multi-type 3D geometric features in natural scene," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 1828–1835, ISBN: 9781467363587. DOI: 10.1109/IROS.2013.6696597.

[6] J. Jiao, Y. Yu, Q. Liao, H. Ye, and M. Liu, "Automatic Calibration of Multiple 3D LiDARs in Urban Environments," May 2019. [Online]. Available: http://arxiv.org/abs/1905.04912.

[7] J. Jiao, H. Ye, Y. Zhu, and M. Liu, "Robust Odometry and Mapping for Multi-LiDAR Systems with Online Extrinsic Calibration," Oct. 2020. [Online]. Available: http://arxiv.org/abs/2010.14294.

[8] R. K. Lenz, "A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, 1989, ISSN: 1042296X. DOI: 10.1109/70.34770.

[9] K. Daniilidis and E. Bayro-Corrochano, "The Dual Quaternion Approach to Hand-Eye Calibration," Tech. Rep., 1996.

[10] M. Shah, R. D. Eastman, and T. Hong, "An overview of robot-sensor calibration methods for evaluation of perception systems," in *Workshop on Performance Metrics for Intelligent Systems*, 2012.

[11] A. Dekel, L. Härenstam-Nielsen, and S. Caccamo, "Optimal least-squares solution to the hand-eye calibration problem," *CoRR*, vol. abs/2002.10838, 2020. [Online]. Available: https://arxiv.org/abs/2002.10838.

[12] F. Furrer, M. Fehr, T. Novkovic, H. Sommer, I. Gilitschenski, and R. Siegwart, "Evaluation of Combined Time-Offset Estimation and Hand-Eye Calibration on Robotic Datasets," in *Field and Service Robotics*, M. Hutter and

R. Siegwart, Eds., Cham: Springer International Publishing, 2018, pp. 145–159, ISBN: 978-3-319-67360-8. DOI: https://doi.org/10.1007/978-3-319-67361-5{_}10. [Online]. Available: https://github.com/ethz-asl/hand_eye_calibration.

[13] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 376–382, Jun. 1991, ISSN: 1042296X. DOI: 10.1109/70.88147.

[14] A. Howard, M. Mataric, and G. Sukhatme, "Relaxation on a mesh: a formalism for generalized localization," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, IEEE, pp. 1055–1060, ISBN: 0-7803-6612-3. DOI: 10.1109/IROS.2001.976308.

[15] G. Kurz, S. A. Scherer, P. Biber, and D. Fleer, "When Geometry is not Enough: Using Reflector Markers in Lidar SLAM," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2022, pp. 4880–4887, ISBN: 978-1-6654-7927-1. DOI: 10.1109/IROS47612.2022.9981522. [Online]. Available: https://ieeexplore.ieee.org/document/9981522/.

[16] Q. Zeng, Y. Kan, X. Tao, and Y. Hu, "LiDAR Positioning Algorithm Based on ICP and Artificial Landmarks Assistance," *Sensors*, vol. 21, no. 21, 2021, ISSN: 1424-8220. DOI: 10.3390/s21217141. [Online]. Available: https://www.mdpi.com/1424-8220/21/21/7141.

[17] D. Meyer-Delius, M. Beinhofer, A. Kleiner, and W. Burgard, "Using artificial landmarks to reduce the ambiguity in the environment of a mobile robot," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 5173–5178, ISBN: 9781612843865. DOI: 10.1109/ICRA.2011.5980111.

[18] R. Szeliski, "Computer Vision: Algorithms and Applications 2nd Edition," Tech. Rep., 2021. [Online]. Available: https://szeliski.org/Book,.

[19] K. Simek, *Dissecting the Camera Matrix, Part 2: The Extrinsic Matrix*, [Online]. Available: https://ksimek.github.io/2012/08/22/extrinsic/, *Sightations - A Computer Vision Blog*.

[20] W. R. Hamilton, "On quaternions; or on a new system of imaginaries in algebra," *London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 25, no. 3, pp. 489–495, 1844.

[21] W. Hamilton, *On Quaternions; or on a new System of Imaginaries in Algebra*.

[22] J. Vince, *Quaternions for Computer Graphics*. London: Springer London, 2021, pp. 81–81, ISBN: 978-1-4471-7508-7. DOI: 10.1007/978-1-4471-7509-4.

[23] Konstantinos Daniilidis, "Hand-Eye Calibration Using Dual Quaternions," Tech. Rep., 1999.

[24] W. K. Clifford, "Preliminary sketch of bi-quaternions," *Proceedings of the London Mathematical Society*, vol. s1-4, no. 1, pp. 381–395, 1873.

[25] Y.-B. Jia, "Dual Quaternions," Tech. Rep.

[26] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," Intel Labs, Tech. Rep. [Online]. Available: http://www.open3d..

[27] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast Global Registration," in 2016, pp. 766–782. DOI: 10.1007/978-3-319-46475-6{_}47.

[28] Y. Chen and G. G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, 1992.

[29] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, IEEE Comput. Soc, pp. 145–152, ISBN: 0-7695-0984-3. DOI: 10.1109/IM.2001.924423.

[30] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems V*, Robotics: Science and Systems Foundation, Jun. 2009, ISBN: 9780262514637. DOI: 10.15607/RSS.2009.V.021.

[31] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992, ISSN: 01628828. DOI: 10.1109/34.121791.

[32] G. Agamennoni, S. Fontana, R. Y. Siegwart, and D. G. Sorrenti, "Point Clouds Registration with Probabilistic Data Association," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2016, pp. 4092–4098, ISBN: 978-1-5090-3762-9. DOI: 10.1109/IROS.2016.7759602.

[33] *"Singular Value Decomposition (SVD) tutorial"*, [Online]. Available: https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm, *Accessed: May. 29, 2023.*

[34] Radu Bogdan Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," Ph.D. dissertation, TECHNISCHE UNIVERSITÄT MÜNCHEN, München, 2009, pp. 42–43. [Online]. Available: https://mediatum.ub.tum.de/doc/800632/941254.pdf.

[35] D. Eberly, "A Robust Eigensolver for 3 × 3 Symmetric Matrices," Geometric Tools, Redmond WA, Tech. Rep., Aug. 2021. [Online]. Available: https://www.geometrictools.com/.

[36] P. Trybała, J. Szrek, F. Remondino, J. Wodecki, and R. Zimroz, "Calibration of a multi-sensor wheeled robot for the 3D mapping of underground mining tunnels," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLVIII-2/W2-2022, pp. 135–142, Dec. 2022, ISSN: 2194-9034. DOI: 10.5194/isprs-archives-XLVIII-2-W2-2022-135-2022.

A

Prestudy: Operating conditions for tests

Test results may only be valid within certain operating ranges. Therefore a prestudy was made to make sure that the operating conditions of the tests conducted were relevant - in other words, close to optimal. By trial and error, the conclusions that were made were that the amount of laps did not seem to make a difference. Velocity and turn radius had a large impact on the other hand, set through the parameters throttle and steering, which both are values in the range $[-1, 1]$. The final values decided upon were 0.05 for throttle and ± 0.7 for steering.

A.1 Perturbed LiDARs

How is the performance of the algorithm affected by the actual relative pose of the LiDARs? When altering the nominal values for the poses between the LiDARs in CARLA, for example changing yaw to 175 deg for the rear LiDAR. The performance from the calibration is supposed to work as well as it does as in the nominal case. In the mentioned case the transform should output 175 deg. This is important since the purpose of the algorithm is to find the correct calibration parameters regardless of how "bad" the LiDARs are mounted (within a certain limit). This is because when calibrating with real-data, the calibration parameters is unknown.

In Table A.1 below, the calibration result for different simulation runs are listed. When writing for example 0/180 we mean front LiDAR is rotated 0 degrees in yaw and the rear LiDAR is rotated 180 degrees in yaw. In the last column all 6 DOF for both LiDARs are perturbed a small amount to see how well the algorithm can handle that case. The driving parameters that were used in the environment augmentation test were used also in these tests.

Table A.1: Table viewing results when LiDARs are not where they are assumed to be. Since the pose errors for perturbed LiDARs even are smaller than the nominal case, this would motivate that the true relative pose of the LiDARs is largely unimportant to the estimate accuracy. Therefore all further results will use the nominal configuration unless otherwise specified. After the environment augmentation test, the average test result over the 8 calibrations was inserted into the table, since it should be more representative than one single calibration. For "xyzrpy5", the rear LiDAR has been misplaced 5 cm in x, y and z and 5° in roll, pitch and yaw.

	Nominal	**Not nominal**	
	yaw 0/180	yaw 0/175	xyzrpy5
#5	0.198 0.699	0.1544 0.480	0.1512 0.262

B

Hand-Eye non-aggregated results

Table B.1: Run 1, the first 5 poses discarded in each of the 25 recordings. Environment augmentation test results using the *Hand-Eye* method.

		Augmentation Type				
		None	**Boxes**		**Cylinders**	
		0	5	10	5	10
Location Number	**#1**	4.5322 39.007	0.2616 1.986	0.0861 0.547	0.0605 0.517	0.2318 1.910
	#2	17.2925 124.758	1.9327 12.997	0.8867 0.452	2.1894 17.533	1.2208 9.364
	#3	0.2595 1.103	0.3470 1.618	0.5515 4.539	0.0638 0.493	0.4885 3.923
	#4	3.2849 28.018	0.1778 1.411	0.1625 0.745	0.7116 5.647	0.2408 1.223
	#5	1.6582 5.379	0.4631 1.335	0.5251 3.321	0.7430 2.105	0.8804 6.770

Table B.2: Run 2, the first 6 poses discarded in each of the 25 recordings.

		Augmentation Type				
		None	**Boxes**		**Cylinders**	
		0	5	10	5	10
Location Number	**#1**	4.0006 34.672	0.1787 1.506	0.1121 0.763	0.0393 0.211	0.1542 0.166
	#2	15.8452 138.982	1.9262 12.218	0.8916 0.366	2.6875 20.125	0.2420 2.021
	#3	0.2193 0.905	0.3755 2.054	0.6106 5.011	0.0253 0.016	0.4934 3.999
	#4	2.4128 20.800	0.1639 1.021	0.4782 1.692	0.7793 5.771	0.1101 0.816
	#5	2.7237 8.559	0.1794 1.361	0.2962 1.933	0.9535 1.228	0.3218 2.325

Table B.3: Run 3, the first 7 poses discarded in each of the 25 recordings.

	Augmentation Type				
	None	**Boxes**		**Cylinders**	
Location Number	0	5	10	5	10
#1	3.4042 29.659	0.0849 0.105	0.7199 4.895	0.0880 0.678	0.1695 0.695
#2	1.1464 5.139	1.8819 12.634	1.3037 4.992	2.7612 14.786	1.1941 8.571
#3	0.1014 0.478	0.1902 1.563	0.6182 5.030	0.0470 0.350	0.6172 5.084
#4	0.9641 7.696	0.1587 1.282	0.3533 1.787	0.1828 0.836	0.0772 0.164
#5	2.2644 5.188	0.2809 2.353	0.2252 0.701	1.0360 4.639	0.6539 5.477

Table B.4: Run 4, the first 8 poses discarded in each of the 25 recordings.

	Augmentation Type				
	None	**Boxes**		**Cylinders**	
Location Number	0	5	10	5	10
#1	3.1208 27.245	0.7463 4.376	0.1437 0.915	0.0983 0.587	0.1588 0.594
#2	0.3632 0.519	1.4960 6.183	0.6868 3.049	2.9720 12.994	1.3092 7.508
#3	0.0784 0.251	0.3843 2.835	0.6372 5.160	0.0673 0.449	0.7268 6.002
#4	0.5811 2.656	0.2105 1.720	0.5892 0.319	0.3506 0.528	0.1383 1.139
#5	2.4784 9.750	0.5488 4.030	0.0866 0.288	1.4506 9.516	0.6406 5.314

Table B.5: Run 5, the first 9 poses discarded in each of the 25 recordings.

	Augmentation Type				
	None	**Boxes**		**Cylinders**	
Location Number	0	5	10	5	10
#1	1.8805 16.311	0.9495 5.360	0.1657 0.956	0.1610 0.035	0.1732 1.168
#2	0.5114 0.351	1.2113 4.206	0.5088 2.818	2.6193 12.036	1.3366 7.867
#3	0.0896 0.446	0.4758 3.368	0.3179 2.501	0.0608 0.486	0.6050 5.016
#4	0.6658 4.885	0.2816 2.360	0.2571 1.295	0.1349 0.563	0.2212 1.861
#5	2.3861 13.446	0.6458 3.702	0.2884 2.295	3.0749 25.549	0.6853 5.636

Table B.6: Run 6, the first 10 poses discarded in each of the 25 recordings.

		Augmentation Type				
		None	**Boxes**		**Cylinders**	
		0	5	10	5	10
Location Number	**#1**	1.6167 13.993	0.6063 3.222	0.0667 0.286	0.2324 1.157	0.1620 1.367
	#2	0.4991 0.624	2.3668 8.264	0.6297 0.832	2.4497 9.328	1.0055 3.643
	#3	0.1634 0.804	0.6110 4.840	0.4011 3.226	0.0926 0.362	0.6794 5.639
	#4	0.9593 5.501	0.4590 3.670	0.3155 2.347	0.1726 0.735	0.4534 3.236
	#5	13.6104 146.840	0.8331 6.107	0.5710 4.488	12.2651 109.928	0.6275 5.177

Table B.7: Run 7, the first 11 poses discarded in each of the 25 recordings.

		Augmentation Type				
		None	**Boxes**		**Cylinders**	
		0	5	10	5	10
Location Number	**#1**	1.3902 11.988	0.2028 1.313	0.1056 0.822	0.2277 0.712	0.1117 0.746
	#2	0.6742 0.427	0.8844 1.085	0.4741 3.477	1.7480 10.745	0.9837 3.396
	#3	0.1063 0.498	0.5937 4.821	0.5365 4.329	0.1103 0.927	0.4243 3.526
	#4	0.1926 1.380	0.6446 4.778	0.3763 3.024	0.2693 2.038	0.0814 0.639
	#5	15.5947 121.994	0.8049 3.330	0.5969 4.104	0.6339 4.661	0.2432 1.715

Table B.8: Run 8, the first 12 poses discarded in each of the 25 recordings.

		Augmentation Type				
		None	**Boxes**		**Cylinders**	
		0	5	10	5	10
Location Number	**#1**	0.8977 7.642	0.0690 0.368	0.4293 2.654	0.1536 0.254	0.1115 0.801
	#2	2.0742 10.065	2.7996 1.377	0.3723 2.563	2.0613 14.710	0.7258 2.718
	#3	0.0997 0.460	0.5946 4.914	0.6118 4.952	0.0701 0.327	0.4423 3.687
	#4	0.5026 3.935	0.2804 2.292	0.3663 2.566	0.0569 0.383	0.1829 0.154
	#5	15.3355 107.467	0.3939 0.375	0.4601 3.672	0.5087 2.750	0.0721 0.363

C

MergeMaps non-aggregated results

Table C.1: Run 1, the first 5 poses discarded in each of the 25 recordings.

Location Number	*Augmentation Type*					
	None	**Boxes**		**Cylinders**		
	0	5	10	5	10	**RMSE**
#1	0.2644 0.700	0.0499 0.156	0.0240 0.180	0.0368 0.127	0.0394 0.090	0.1232 0.338
#2	0.4329 0.323	0.0251 0.373	0.0316 0.265	0.1261 0.256	0.0756 0.298	0.2052 0.306
#3	0.0345 0.204	0.0504 0.189	0.0594 0.243	0.0199 0.088	0.0340 0.267	0.0420 0.208
#4	0.2111 0.712	0.0388 0.273	0.0566 0.313	0.0994 0.603	0.0377 0.250	0.1101 0.470
#5	0.1248 0.397	0.0284 0.250	0.0806 0.455	0.1058 0.310	0.1172 0.532	0.0978 0.401
RMSE	0.2524 0.510	0.0399 0.259	0.0544 0.305	0.0880 0.331	0.0687 0.321	

Table C.2: Run 2, the first 6 poses discarded in each of the 25 recordings.

Location Number	*Augmentation Type*					
	None	**Boxes**		**Cylinders**		
	0	5	10	5	10	**RMSE**
#1	0.2300 0.588	0.0519 0.132	0.0189 0.062	0.0233 0.126	0.0385 0.168	0.1077 0.287
#2	0.3356 0.251	0.0343 0.413	0.0350 0.155	0.1015 0.380	0.0608 0.244	0.1606 0.304
#3	0.0288 0.204	0.0597 0.191	0.0487 0.216	0.0192 0.059	0.0323 0.299	0.0404 0.209
#4	0.0953 0.125	0.0459 0.273	0.0210 0.066	0.0453 0.296	0.0330 0.224	0.0544 0.215
#5	0.1699 0.569	0.0463 0.100	0.0881 0.509	0.1281 0.372	0.1395 0.567	0.1222 0.459
RMSE	0.2021 0.397	0.0483 0.248	0.0493 0.260	0.0770 0.279	0.0732 0.331	

Table C.3: Run 3, the first 7 poses discarded in each of the 25 recordings.

		Augmentation Type					
		None	**Boxes**		**Cylinders**		
		0	5	10	5	10	**RMSE**
Location Number	**#1**	0.2150 0.536	0.0758 0.211	0.0267 0.261	0.0270 0.075	0.0369 0.129	0.1047 0.291
	#2	0.2131 0.194	0.0223 0.279	0.0430 0.166	0.0856 0.288	0.0595 0.263	0.1083 0.243
	#3	0.0318 0.182	0.0560 0.169	0.0449 0.258	0.0222 0.069	0.0364 0.316	0.0400 0.216
	#4	0.1006 0.169	0.0342 0.261	0.0463 0.332	0.0509 0.334	0.0129 0.026	0.0569 0.252
	#5	0.1802 0.673	0.0410 0.121	0.0680 0.234	0.1154 0.364	0.1393 0.634	0.1196 0.460
	RMSE	0.1645 0.410	0.0495 0.216	0.0476 0.256	0.0699 0.260	0.0718 0.343	

Table C.4: Run 4, the first 8 poses discarded in each of the 25 recordings.

		Augmentation Type					
		None	**Boxes**		**Cylinders**		
		0	5	10	5	10	**RMSE**
Location Number	**#1**	0.1879 0.471	0.1211 0.366	0.0536 0.257	0.0262 0.098	0.0352 0.117	0.1046 0.298
	#2	0.1327 0.117	0.0334 0.235	0.0464 0.299	0.0870 0.247	0.0693 0.289	0.0815 0.246
	#3	0.0338 0.173	0.0629 0.158	0.0532 0.299	0.0241 0.061	0.0393 0.264	0.0448 0.209
	#4	0.3238 0.370	0.0346 0.251	0.0555 0.335	0.0404 0.293	0.0359 0.273	0.1497 0.308
	#5	0.1779 0.670	0.0541 0.109	0.0654 0.438	0.1503 0.414	0.1184 0.552	0.1228 0.475
	RMSE	0.1952 0.413	0.0691 0.240	0.0552 0.331	0.0813 0.258	0.0677 0.330	

Table C.5: Run 5, the first 9 poses discarded in each of the 25 recordings.

		Augmentation Type					
		None	**Boxes**		**Cylinders**		
		0	5	10	5	10	**RMSE**
Location Number	**#1**	0.2000 0.478	0.1222 0.362	0.0302 0.266	0.0435 0.235	0.0414 0.150	0.1090 0.319
	#2	0.0974 0.052	0.0189 0.306	0.0571 0.146	0.0843 0.274	0.0666 0.270	0.0702 0.230
	#3	0.0370 0.165	0.0668 0.137	0.0364 0.259	0.0252 0.073	0.0376 0.259	0.0429 0.192
	#4	0.1476 0.190	0.0310 0.276	0.0545 0.310	0.0328 0.293	0.0379 0.278	0.0752 0.272
	#5	0.1577 0.622	0.0369 0.051	0.0484 0.367	0.1443 0.424	0.1012 0.469	0.1092 0.430
	RMSE	0.1397 0.369	0.0664 0.254	0.0465 0.279	0.0794 0.283	0.0620 0.303	

Table C.6: Run 6, the first 10 poses discarded in each of the 25 recordings.

	Augmentation Type					
	None	**Boxes**		**Cylinders**		
Location Number	0	5	10	5	10	**RMSE**
#1	0.1667 0.375	0.1025 0.329	0.0343 0.188	0.0281 0.144	0.0525 0.156	0.0928 0.257
#2	0.0850 0.041	0.0285 0.250	0.0472 0.134	0.0911 0.290	0.0514 0.294	0.0651 0.225
#3	0.0327 0.168	0.0715 0.157	0.0353 0.280	0.0164 0.063	0.0447 0.239	0.0440 0.196
#4	0.0771 0.064	0.0381 0.274	0.0479 0.300	0.0390 0.248	0.0363 0.259	0.0500 0.244
#5	0.2265 0.802	0.0489 0.095	0.0345 0.287	0.1635 0.481	0.0847 0.385	0.1332 0.472
RMSE	0.1366 0.405	0.0637 0.236	0.0403 0.247	0.0867 0.283	0.0564 0.277	

Table C.7: Run 7, the first 11 poses discarded in each of the 25 recordings.

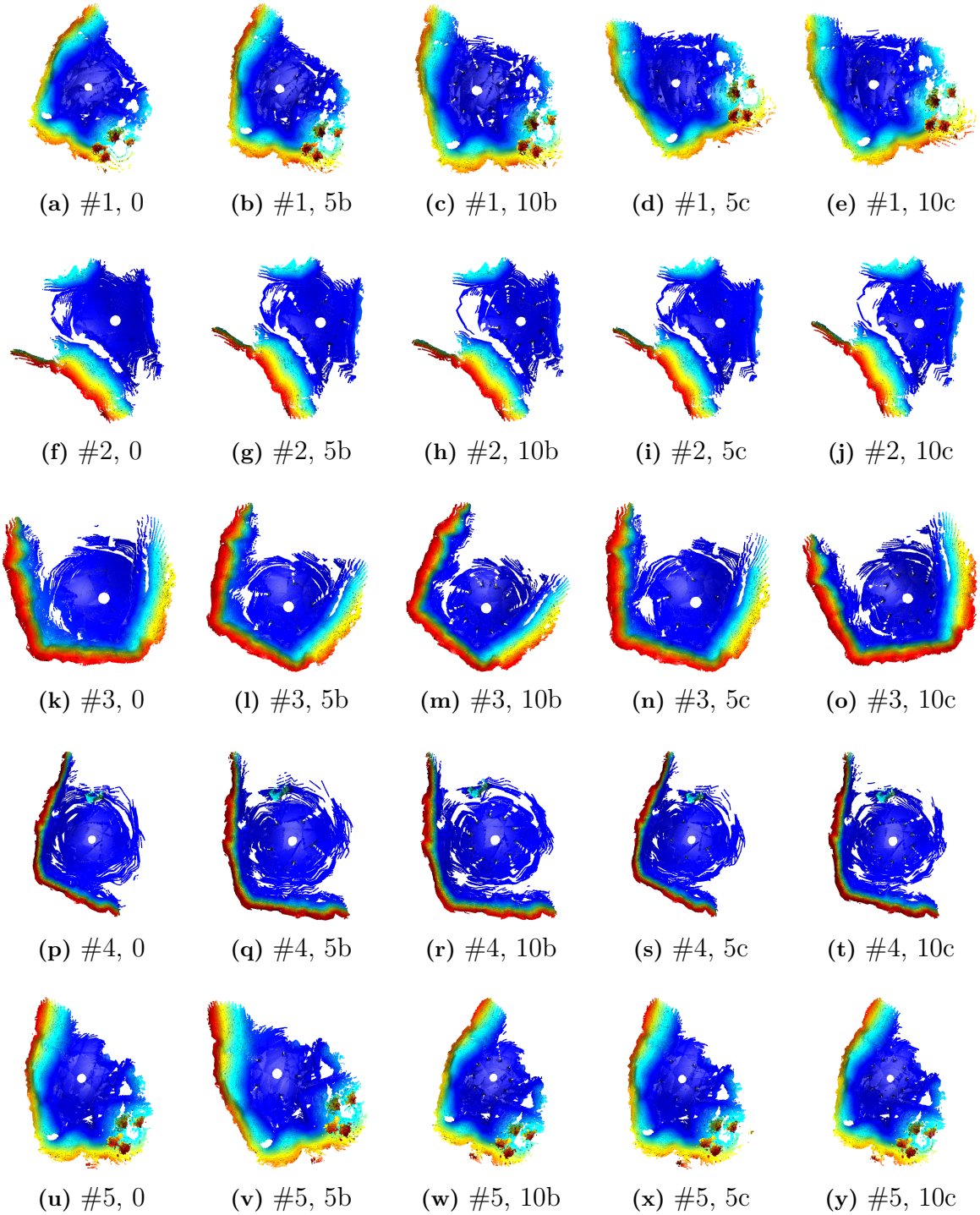
	Augmentation Type					
	None	**Boxes**		**Cylinders**		
Location Number	0	5	10	5	10	**RMSE**
#1	0.1568 0.360	0.0709 0.270	0.0366 0.173	0.0448 0.153	0.0436 0.139	0.0835 0.235
#2	0.0598 0.045	0.0609 0.120	0.0507 0.087	0.0937 0.386	0.0436 0.308	0.0641 0.231
#3	0.0323 0.182	0.0767 0.146	0.0373 0.293	0.0207 0.061	0.0459 0.236	0.0466 0.200
#4	0.0569 0.087	0.0485 0.276	0.0436 0.315	0.0331 0.238	0.0068 0.043	0.0416 0.220
#5	0.2440 0.886	0.0540 0.115	0.0401 0.274	0.1525 0.471	0.0714 0.353	0.1359 0.494
RMSE	0.1356 0.438	0.0631 0.199	0.0420 0.244	0.0843 0.301	0.0470 0.243	

Table C.8: Run 8, the first 12 poses discarded in each of the 25 recordings.

	Augmentation Type					
	None	**Boxes**		**Cylinders**		
Location Number	0	5	10	5	10	**RMSE**
#1	0.1288 0.323	0.0418 0.173	0.0351 0.125	0.0476 0.161	0.0355 0.113	0.0679 0.194
#2	0.0643 0.102	0.0515 0.270	0.0616 0.260	0.0767 0.351	0.0477 0.363	0.0612 0.285
#3	0.0305 0.166	0.0743 0.156	0.0532 0.343	0.0246 0.087	0.0517 0.250	0.0501 0.219
#4	0.0461 0.096	0.0508 0.281	0.0443 0.357	0.0296 0.224	0.0343 0.273	0.0418 0.261
#5	0.3022 0.975	0.0650 0.170	0.0414 0.232	0.1299 0.423	0.0639 0.358	0.1538 0.518
RMSE	0.1517 0.470	0.0578 0.217	0.0480 0.277	0.0728 0.278	0.0479 0.286	

D

Visual mapping results



(a) #1, 0 **(b)** #1, 5b **(c)** #1, 10b **(d)** #1, 5c **(e)** #1, 10c

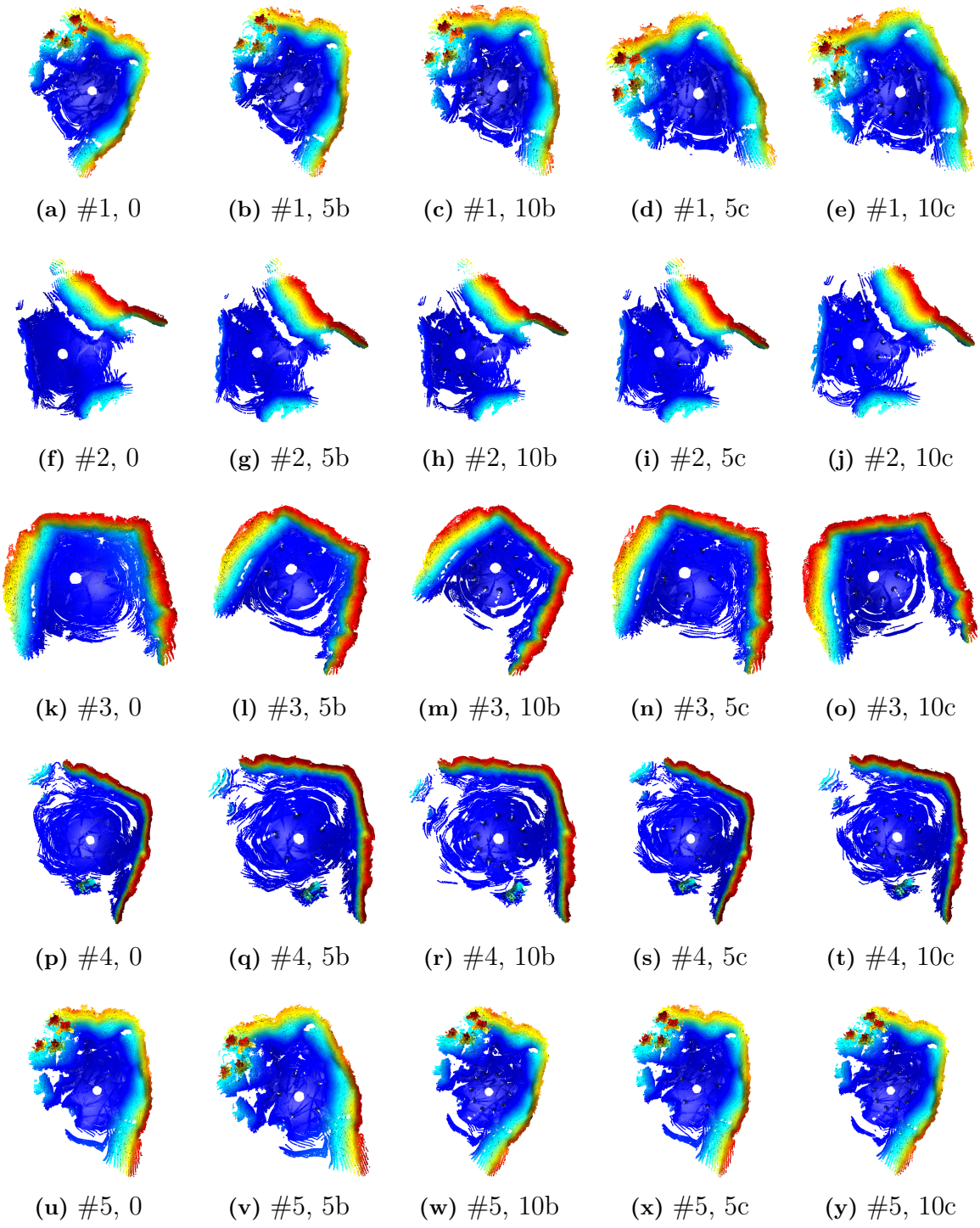
(f) #2, 0 **(g)** #2, 5b **(h)** #2, 10b **(i)** #2, 5c **(j)** #2, 10c

(k) #3, 0 **(l)** #3, 5b **(m)** #3, 10b **(n)** #3, 5c **(o)** #3, 10c

(p) #4, 0 **(q)** #4, 5b **(r)** #4, 10b **(s)** #4, 5c **(t)** #4, 10c

(u) #5, 0 **(v)** #5, 5b **(w)** #5, 10b **(x)** #5, 5c **(y)** #5, 10c

Figure D.1: Mapping results for the front LiDAR for each scenario. Top view, color representing height, rotation of initial pose. First 5 poses skipped. (b=Box, c=Cylinder)



(a) #1, 0 **(b)** #1, 5b **(c)** #1, 10b **(d)** #1, 5c **(e)** #1, 10c

(f) #2, 0 **(g)** #2, 5b **(h)** #2, 10b **(i)** #2, 5c **(j)** #2, 10c

(k) #3, 0 **(l)** #3, 5b **(m)** #3, 10b **(n)** #3, 5c **(o)** #3, 10c

(p) #4, 0 **(q)** #4, 5b **(r)** #4, 10b **(s)** #4, 5c **(t)** #4, 10c

(u) #5, 0 **(v)** #5, 5b **(w)** #5, 10b **(x)** #5, 5c **(y)** #5, 10c

Figure D.2: Mapping results for the rear LiDAR for each scenario. Top view, color representing height, rotation of initial pose. First 5 poses skipped. (b=Box, c=Cylinder)

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

CHALMERS
UNIVERSITY OF TECHNOLOGY