Aalto University
School of Engineering

Pauli Putkiranta

Geometric calibration of rotating multi-beam lidar systems

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo, 31 Dec 2019

Supervisor: Prof. Matti Vaaja
Advisor: M.Sc. (Tech.) Heikki Hyyti

Aalto University
School of Engineering

Aalto University, P.O. Box 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author
Pauli Putkiranta

Title
Geometric calibration of rotating multi-beam lidar systems

Master's programme Geoinformatics **Code** ENG22

Thesis supervisor Prof. Matti Vaaja

Thesis advisor M.Sc. (Tech.) Heikki Hyyti

Date 31 Dec 2019 **Number of pages** 68+24 **Language** English

Abstract

The introduction of light-weight and low-cost multi-beam laser scanners provides ample opportunities in positioning and mapping as well as automation and robotics. The fields of view (FOV) of these sensors can be further expanded by actuation, for example by rotation. These rotating multi-beam lidar (RMBL) systems can provide fast and expansive coverage of the geometries of spaces, but the nature of the sensors and their actuation leave room for improvement in accuracy and precision. Geometric calibration methods addressing this space have been proposed, and this thesis reviews a selection of these methods and evaluates their performance when applied to a set of data samples collected using a custom RMBL platform and six Velodyne multi-beam sensors (one VLP-16 Lite, four VLP-16s and one VLP-32C). The calibration algorithms under inspection are unsupervised and data-based, and they are quantitatively compared to a target-based calibration performed using a high-accuracy point cloud obtained using a terrestrial laser scanner as a reference. The data-based calibration methods are automatic plane detection and fitting, a method based on local planarity and a method based on the information-theoretic concept of information entropy. It is found that of these, the plane-fitting and entropy-based measures for point cloud quality obtain the best calibration results.

Keywords Laser scanning, calibration, lidar, optimisation, point cloud

Aalto-yliopisto
Insinööritieteiden korkeakoulu

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi

Diplomityön tiivistelmä

Tekijä
Pauli Putkiranta

Työn nimi
Pyörivien monilaserkeilainjärjestelmien geometrinen kalibrointi

Maisteriohjelma Geoinformatiikka **Koodi** ENG22

Työn valvoja Prof. Matti Vaaja

Työn ohjaaja DI Heikki Hyyti

Päivämäärä 31.12.2019 **Sivumäärä** 68+24 **Kieli** englanti

Tiivistelmä

Kevyet ja edulliset monilaserkeilaimet tuovat uusia mahdollisuuksia paikannus- ja kartoitusaloille mutta myös automaatioon ja robotiikkaan. Näiden sensorien näköaloja voidaan kasvattaa entisestään esimerkiksi pyörittämällä, ja näin toteutettavat pyörivät monilaserkeilainjärjestelmät tuottavat nopeasti kattavaa geometriaa niitä ympäröivistä tiloista. Sensorien rakenne ja järjestelmän liikkuvuus lisäävät kuitenkin kohinaa ja epävarmuutta mittauksissa, minkä vuoksi erilaisia geometrisia kalibrointimenetelmiä onkin ehdotettu aiemmassa tutkimuksessa. Tässä diplomityössä esitellään valikoituja kalibrointimenetelmiä ja arvioidaan niiden tuloksia koeasetelmassa, jossa pyörivälle alustalle asennetuilla Velodyne-monilaserkeilaimilla (yksi VLP-16 Lite, neljä VLP-16:aa ja yksi VLP-32C) mitataan liikuntasalin geometriaa. Tarkasteltavat menetelmät ovat valvomattomia ja vain mittauksiin perustuvia ja niitä verrataan samasta tilasta hankittuun tarkkaan maalaserkeilausaineistoon. Menetelmiä ovat tasojen automaattinen etsintä ja sovitus, paikalliseen tasomaisuuteen perustuva menetelmä sekä informaatioteoreettiseen entropiaan perustuva menetelmä. Näistä tasojen sovitus ja entropiamenetelmä saavuttivat parhaat kalibrointitulokset referenssikalibraatioon verrattaessa.

Avainsanat Laserkeilaus, kalibrointi, lidar, optimointi, pistepilvi

Preface

The document you are eyeing appears to be the culmination of some $5\frac{1}{2}$ years at Aalto University and its accompanying community, but in reality I would argue that the take-away can be found completely elsewhere (why are you reading *this*, exactly?). This is, however, the longest single piece of literature—I use the term rather loosely—I have thus far produced and as such, an object of my pride. I have spent the last seven months putting it together, so enjoy, I suppose.

It is worthwhile to take the time to acknowledge some people without whom I would have either written something completely different—for better or worse—or I would have remained forever a mere Bachelor of Science (the horror!) or worse still, I would not have been paid. The obvious names are the ones already printed on the cover page; I want to thank Professor Matti Vaaja for his supervision and commentary and M.Sc. (Tech.) Heikki Hyyti for his guidance and patience. Heikki's research and code base made this work possible, for one, but also significantly easier.

Additionally, I want to extend my gratitude to Kartanonranta school for allowing us to use their premises for measurements and to Aimad El Issaoui for helping with the P40 part of those measurements. The research community at the Finnish Geodetic Institute (FGI) and specifically its department of remote sensing and photogrammetry also deserve a mention for providing me with the opportunity to work with them in good spirit.

My family has kept me on my toes by repeatedly inquiring how, and whether, this work was progressing, for which I am grateful. I also appreciate all the friends who have listened to me try to explain what this thesis is about only to reply with what amounts to a blank stare and an “Ok.” Your trials are over.

Kuluneet vuodet ovat tarjonneet lukuisia läheisiä ystäviä ja paljon lämpimiä muistoja – ja vaikka paljon on jo unohdettu, on vielä enemmän varmasti vielä unohtamatta. Kiitos Rakennusinsinöörikillan raadille 2016 sekä muille kanssaharrastajille! Lopuksia haluan kiittää vielä letkuneuvostoa, kadun retuperää ja Roope Hiirtä; unohtamatta tietenkään niitä neljää, joihin on aina voinut luottaa: siivooja, paari, konsulentti. Teidän kaikkien ja yhteisten tovereidemme suurenmoisessa seurassa olen ollut onnellinen. Jos taas Sinä, arvon lukija, nautit tämän teoksen onnellisesti kovikkeen kera, nautithan tarjoamanani myös loppuillan.

Masala, December 31, 2019,

Pauli Putkiranta

Contents

Symbols and abbreviations

Symbols

a	Scalar
a_i	ith scalar element of vector or set
$\mathbf{a}$	Vector
$\hat{\mathbf{a}}$	Vector of observations
$\mathbf{a}_i$	ith row of matrix or ith iteration of vector
$\mathbf{A}$	Matrix
$\hat{\mathbf{A}}$	Matrix of observations
E	Errorsum, object of minimisation in optimisation routines
H	Entropy of a discrete variable
h	Entropy value or derivative thereof for continuous variable
σ	Standard deviation

Operators

$\exp(a)$	Euler's constant e to the power of a
$\mathrm{E}[X]$	Expected value of X
$G(\mathbf{x} \mid \mu, \mathbf{\Sigma})$	The Gaussian probability density function with mean μ and covariance $\mathbf{\Sigma}$ for point x
$\int_S$	Integral over segment S
log	Logarithmic operator. Base 2, unless specified otherwise
O	Big O operator
$\mathbf{A}^T$	Transpose of matrix A
$\sum_i$	Sum over index i
$\cup$	Union of sets
$\leftarrow$	Assignment operator in algorithms
$\odot$	Element-wise multiplication of vector or matrix
$\oslash$	Element-wise division of vector or matrix
$\mathbf{A}^{\circ b}$	Matrix A element-wise exponentiation to b
$\lceil a \rceil$	a rounded up to nearest integer
$a \perp b$	a is perpendicular to b

Abbreviations

3D	Three-dimensional
ALS	Aerial laser scanning
CPU	Central processing unit
DOF	Degrees of freedom
FLANN	Fast library for approximate nearest neighbors
FOV	Field of view
GMM	Gaussian mixture model
HDR	High dynamic range
ICP	Iterative closest point
IMU	Inertial measurement unit
IRLS	Iterative reweighted least-squares
KDE	Kernel density estimation
Kd-tree	k-dimensional tree
kNN	k nearest neighbours
Lidar	Light detection and ranging[1]
LS	Laser scanning
MBL	Multi-beam lidar
MLS	Mobile laser scanning
NDT	Normal distribution transform
PDF	Probability density function
RANSAC	Random sampling consensus
RMBL	Rotating multi-beam lidar
RPM	Revolutions per minute
RSBL	Rotating single-beam lidar
RQE	Rényi quadratic entropy
SBL	Single-beam lidar
TLS	Terrestrial laser scanning
TOF	Time of flight

[1]Originally, *lidar* was a simple portmanteau of *light* and *radar* (James Ring, 1963; *lidar, n.*, 2019). The acronymic usage, though sensible, came later—albeit just a few years—from Collis (1965), and is sometimes given as *light imaging, detection and ranging* (e.g. by Ellis, 2019). There is no wide consensus on capitalisation (LIDAR, LiDAR, Lidar, lidar), and this thesis will use the noncapitalised *lidar*. Lidar, like radar, is sometimes used to refer to a specific sensor, rather than the technology more generally.

1. Introduction

Low-cost lightweight laser sensors have been gaining momentum in the fields of automation and mapping during the last decades. Light detection and ranging (lidar) instruments have been a staple in surveying for years, but increased mobility and flexibility enable faster measurements and real-time use cases such as autonomous driving. Multi-beam lidar (MBL) sensors especially are capable of capturing large fields of view (FOV) for real-time applications. Furthermore, the actuation of these sensors by, for example, rotation as well as fusion with other sensors can enable the observation of wider areas and more metrics. These actuated sensors form the various lidar systems that this thesis is concerned with—that is, a lidar system consists of a laser sensor, other possible sensors and mechanics exterior to these, such as a rotating platform.

However, affordability in sensors often means a compromise in areas such as precision and accuracy. The reported accuracy of one popular MBL device, the Velodyne VLP-16 Puck (Velodyne Lidar, 2015b), is approximately ± 3 cm at a range of 10 meters. Any actuation mechanism is likely to increase errors in measurements because of additional transformations, and cumulatively the need for improved accuracy grows. Self-calibration methods offer the possibility of a more accurate definition of both internal sensor parameters and external system parameters. With high-accuracy sensors such as lidar, small corrections in calibration can provide significant improvements in observation accuracy.

This thesis provides a synthesis of a selection of calibration methods proposed for lidar systems by reviewing their theoretical background and practical implementation and evaluating their performance in a test environment using a custom rotating MBL (RMBL) setup and a selection of MBL sensors. The calibration methods are only concerned with sensor geometry and do not consider reflectance values.

1.1 Goals

The main contribution of this thesis is an empirical and quantitative comparison of the performance of different calibration approaches. This is based on calibrations performed on a custom RMBL system using a set of measurements from a variety of sensors. In practice, all calibration routines are optimisation problems, where the method describes the cost function that is used as a measure of quality for the observations. The optimisation procedure then attempts to find a set of calibration parameters that evaluates the cost function optimally. The cost functions under scrutiny here are presented schematically and in pseudocode so as to provide replicable algorithms. In addition, where applicable, any related cost functions are also presented—for example if a method used here was generalised from another presented elsewhere. The methods are chosen so as to be generalisable to any MBL, RMBL or rotating single-beam lidar (RSBL) system and setup, including those with multiple sensors.

In addition to a quantitative comparison, a qualitative evaluation of algorithms is made to provide the reader with an understanding of the suitability of the methods in varying circumstances. This evaluation includes a look at computational complexity, behaviour in optimisation and reliability (e.g. with respect to varying levels of error in measurements). Additional properties, such as responsivity to different kinds of environments are also discussed but not evaluated. The evaluation and discussion are based on observed phenomena as well as experience gained through the implementation and testing of methods.

1.2 Limitations

The empirical section of this thesis only considers the calibration of a single static RMBL system. The sensor of the system is replaceable, and measurements are made using six separate sensors (four Velodyne VLP-16s, one Velodyne VLP-16 Lite and one Velodyne VLP-32C), all manufactured by Velodyne Lidar. The methods and results are not directly generalisable to moving systems, though a consideration for the moving coordinate frame should be the only necessary extension. The measurements for calibration method comparison were conducted in a single environment, which means that point clouds used in calibration were of this specific environment. Different environments may produce different results, and the results are not validated using a separate environment in this thesis. Rather, they are validated by comparisons with different methods.

Additionally, the selection of calibration algorithms presented here is not exhaustive, but the selection was made based on a review of previous literature on the topic. The effect of the optimisation algorithm used—in this case, the Nelder-Mead method (Nelder & Mead, 1965)—is not considered. In other words, it is possible that calibrations using different sensor setups, measurement environments, calibration or optimisation algorithms may produce results varying from those presented here.

1.3 Structure

This thesis is structured to first provide a wide overview of the laser scanning (LS) field in general, zooming in on MBL sensor systems. Additionally, Section 2 also provides the theoretical background for the calibration methods investigated. Section 3 describes the experimental setup—the devices and environment—used for the empirical part of this work. Next, further details on the calibration algorithms and implementations along with evaluation methodology are provided in Section 4. Results and observations are presented in Section 5, with a discussion in Section 6 and final remarks in Section 7.

2. Background

This section reviews the state of laser scanning at the time of writing. Special attention is paid to mobile laser scanning and (relatively) low-cost small scanners with applications in, for instance, robotics (e.g. C. Chen et al., 2017) and autonomous driving (e.g. Pfrunder, Borges, Romero, Catt, & Elfes, 2017), as the devices used for the experimental section of this thesis are being developed for automation in forestry and traffic. In addition, an overview of calibration methods that have been proposed by previous authors is given to provide background and insight into the choices made later in this thesis.

2.1 Laser scanning

Laser scanning refers to the active measurement of an environment using laser light and its reflection. Molebny, Kamerman, and Steinvall (2010) write that the origins of laser scanning can be found in military research and technology. Radar played a large role in World War II, after which interest in higher resolution and better accuracy motivated the development of devices using shorter wavelengths. Radar technology expanded into microwave area of the spectrum while a separate research area sprouted that employed lasers in the infrared area of the spectrum. By the 1960's, the first laser range finders were being used and developed in many countries, by both the military and by private enterprise. Around the same time, similar instruments were being used for atmospheric and ocean measurements, though properties (i.e. wave frequency) were changed to conform to the target or object of interest.

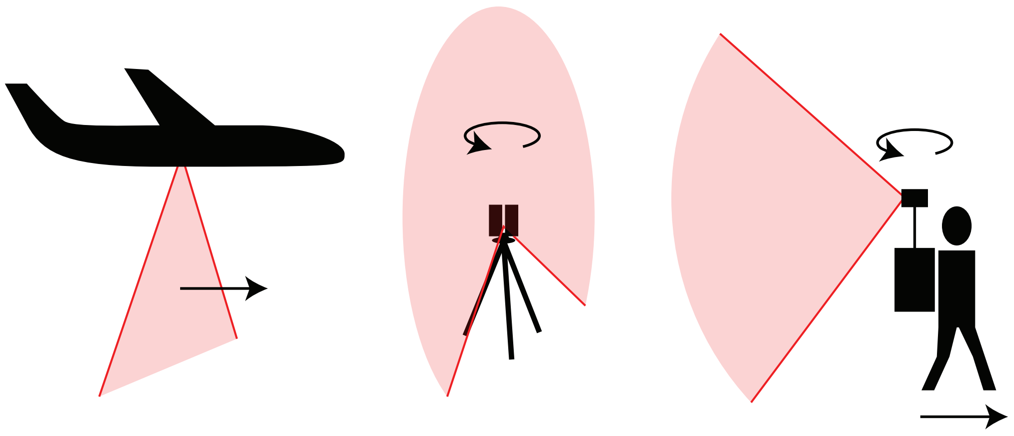


Figure 2.1. A simple sketch of ALS (left), TLS (middle) and MLS (right) in action.

Molebny et al. (2010) continue that the 1970's and 1980's saw the advent of imaging sensors which used various technologies (e.g. CO_2 lidar and Doppler-velocity imaging, to name a few) to extract more properties from targets than just range and velocity. These imaging methods are similar to flash lidar, which emerged in the 1990's alongside the first terrestrial and airborne laser scanners (TLS and

ALS, respectively). This also saw the shift towards civilian research, as such scanners were readily deployed for surveying and mapping purposes outside the military realm. ALS has been used to collect data about the earth's surface and create digital elevation and surface models. TLS has also been used for topography, but also monitoring, documentation and forensic purposes. Development has accelerated through the 21st century with increasingly affordable sensors and better capacities to process the large amounts of data that they produce. Mobile laser scanning (MLS) and real-time processing have expanded the applications of lidar into robotics and computer vision, and these areas are currently continuing to expand rapidly.

Scanning action	Terrestrial laser scanning (TLS)	Scanners that run at fixed positions, usually mounted on a tripod on the ground. Typically multiple scans are taken from an environment and combined. High accuracy, relatively slow.
	Aerial laser scanning (ALS)	Scanners used to map the ground from aerial vehicles. Low resolution, fast scanning of large areas.
	Mobile laser scanning (MLS)	Scanners that can be mounted on a moving platform and moved around while they scan their environments, e.g. handheld, backpack and non-aerial-vehicle-bourne scanners.
Measurement method	Time-of-flight	The scanner measures the time it takes for a pulse of light to travel to the target and back.
	Phase shift	Technically a TOF method where the phase of the laser is continuously shifted and the phase of the returning beam is used to determine TOF for a specific return.
	Triangulation	Laser source and detector are separate with precisely known displacement parameters. Geometry of target determined by observing the location of laser (typically a line) in sensor field of view.

Table 2.1. Types of laser scanning.

Table 2.1 and Figure 2.1 provide a simple classification of different kinds of laser scanning into terrestrial, aerial and mobile. The table also introduces three measurement methods—time-of-flight (TOF), phase shift and triangulation. However, the TLS-ALS-MLS classification is mainly useful in surveying applications. For example, triangulation sensors are often used in industrial quality control, which does not qualify as aerial or mobile laser scanning and is different from what is meant by TLS here.

2.2 Mobile laser scanning

MLS can also be considered to be a subcategory of TLS, rather than a group of its own (though all categorisations can be considered arbitrary). This work holds the distinction that MLS sensors are meant to move while measuring whereas TLS sensors are not, though they typically do rotate. The sensor systems investigated in this thesis use mobile TOF laser scanners, as the pulsed TOF method is almost ubiquitous in MLS. Mobile sensors tend to be lighter in construction than TLS equipment, making them more suitable for mobile applications. However, their light weight sacrifices accuracy. Additionally, combining these sensors with moving and rotating platforms can augment their ability to sense their environments. The reduced accuracy and sensor fusion possibilities introduce the need for precise calibration.

MLS sensors are mounted on some moving platform. Such platforms can include backpacks (e.g. Leica Geosystems, 2019), handheld devices (e.g. Paracosm, 2019) or vehicles (e.g. Xie, Xu, & Wang, 2019). The sensors themselves can be categorised as either single-beam or multi-beam lidar (SBL and MBL, respectively), the former producing 2-dimensional and the latter 3-dimensional data. In MLS solutions, SBL sensors are typically actuated in some rotating or nodding fashion to add a third dimension to the observations. In some cases, MBL sensors can also be rotated for an increased field of view (FOV). The actuated sensor systems are named rotating single- or multi-beam lidars (RSBL or RMBL, respectively).

An increasingly common modern application of mobile lidar sensors is in vehicles. As detailed by Hecht (2018), the development of autonomous vehicles creates a need for more sophisticated modelling instruments. To be able to steer itself, a vehicle needs to be able to understand the geometry of its constantly changing environment. In practice, there are two possible approaches to real-time remote geometric modelling - laser scanning and photogrammetry. Laser scanners have long been rather expensive for commercial applications, but a continued fall in pricing combined with high accuracy make them increasingly viable alternatives as autonomous vehicle sensors. Implementations typically also involve various other sensors, such as cameras, radar and ultrasound ranging.

2.3 Laser scanner configurations

In the fields of robotics and MLS, it is typical to employ various sensor configurations that displace sensors (laser scanners, GNSS receivers, inertial measurement units (IMUs), etc.) from one another in order to achieve optimal measurements for whatever purpose. For example, it may be useful to mount the scanner in an open position from which observations in all directions are possible while placing other sensors and hardware in a more centralised position. Additionally, sensors' fields of view can be augmented with actuation mechanisms. Jesús Morales et al. (2018) provide an overview of customised lidar systems that employ rotation, which is adapted and updated in Table 2.2.

Author	**Type**	**Sensor**	**Application**
Batavia (2002)	RSBL	Sick[1]	Obstacle detection
Wulf (2003)	RSBL	Sick LMS200	Density analysis
Weingarten (2006)	RSBL	2 Sick LMS200	Indoor scene reconstruction
Dias (2006)	RSBL	Sick LMS200	Sensor comparison
Sheh (2006)	RSBL	Hokuyo URG-04LX	Configuration analysis
Ueda (2006)	RSBL	Hokuyo URG-04LX	Mapping
Bosse (2009)	RSBL	SICK LMS291	Mapping
Yoshida (2010)	RSBL	Hokuyo UTM-30LX	Mapping
Morales (2011)	RSBL	Hokuyo UTM-30LX	Mapping and modelling
Sheehan (2012)	RSBL	3 SICK LMS-151	Calibration
Xiao (2013)	RSBL	Hokuyo UTM-30LX	Indoor robot
Neumann (2014)	RMBL	Velodyne HDL-64E	Underground mapping
Morales (2014)	RSBL	Hokuyo UTM-30LX	Boresight calibration
Alismail (2015)	RSBL	Hokuyo UTM-30LX-EX	Calibration for mapping
An (2015)	RSBL	Hokuyo URG-30LX	Indoor robot
Martínez (2015)	RSBL	Hokuyo UTM-30LX-EX	UGV and UAV environment modelling
Özbay (2015)	RSBL	Hokuyo UTM-30LX	UGV obstacle modelling
Moon (2015)	RSBL	SICK LMS511-pro	Cargo ship modelling
Öberlander (2015)	RSBL	Hokuyo UTM-30LX	System calibration
Shaukat (2016)	RSBL	Hokuyo UTM-30LX	Terrain modelling
Schubert (2016)	RSBL	Hokuyo UTM-30LX	Robot mapping
Leingartner (2016)	RMBL	Velodyne HDL-64E	Mapping
Neumann (2016)	RSBL	Hokuyo UTM-30LX-EW	RSBL-RMBL comparison
	RMBL	Velodyne VLP-16	
Kang (2016)	RSBL	Hokuyo UTM-30LX	6-DOF calibration
Droeschel (2017)	RSBL	Hokuyo UTM-30LX-EW	Robot mapping
Klamt (2017)	RMBL	Velodyne VLP-16	Robot mapping
Jesús Morales (2018)	RMBL	Velodyne VLP-16	Distribution analysis
Hyyti (2019)	RMBL	Velodyne VLP-16 Lite	Forestry automation

Table 2.2. RSBL and RMBL systems proposed by various authors over the last 20 years. (Adapted and modified from Jesús Morales et al., 2018, p. 3)

The most common laser scanner actuation mechanisms are rotation around some axis, either fully (e.g. Alismail & Browning, 2015; Sheehan et al., 2012) or in a nodding manner (e.g. McDaniel, Nishihata, Brooks, & Iagnemma, 2010). Often, it is SBL sensors that are rotated in order to achieve an extra dimension in field-of-view. For example, Sheehan et al. (2012) present a rotating lidar system with three SBL sensors as an affordable actuation mechanism for obtaining 3D point clouds (see Figure 2.2). However, also multi-beam lidar sensors can be actuated or combined to increase the size of the observed area. Gong, Wen, Wang, and Li (2018) use a non-rotating backpack setup of 2 MBL sensors (Velodyne VLP-16), where one is set at an angle for increased FOV (Figure 2.3). At the same time, Neumann et al. (2016) rotate a single Velodyne VLP-16 sensor to achieve a full FOV (Figure 2.4).

Most applications of MBL sensors do not employ a rotation mechanism (and

[1]The paper does not specify the sensor model, but an image reveals it was produced by SICK.

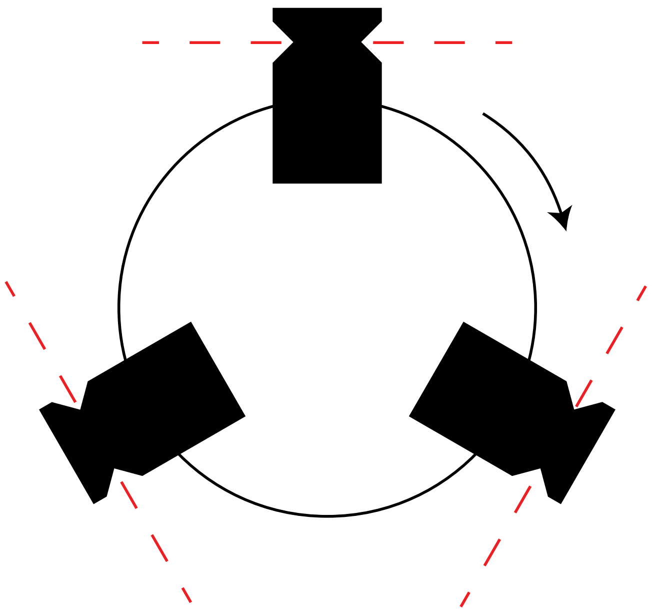


Figure 2.2. Rotating sensor system proposed by Sheehan et al. (2012), composed of three 2D SICK LMS-151 SBL sensors with 270° FOV and a rotating platform.

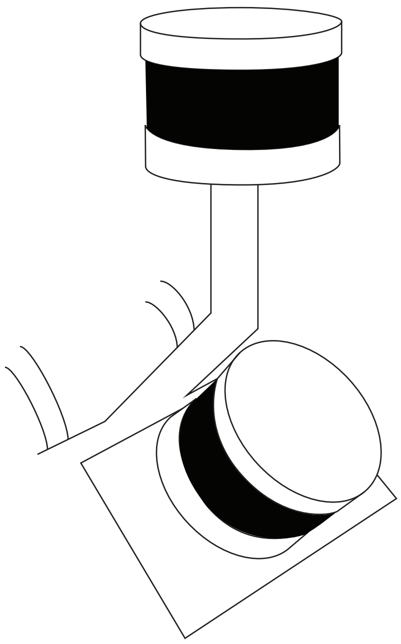


Figure 2.3. Backpack MBL setup with two Velodyne VLP-16 sensors. (Adapted from Gong et al., 2018)

are not included in Table 2.2). For example, many MLS setups mounted on vehicles make do with the FOV provided by the MBL sensor, with at least the exceptions of Maddern, Harrison, and Newman (2012), who mount a RSBL on

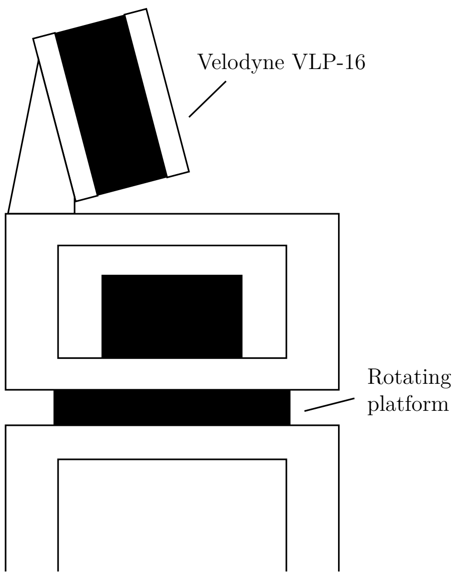


Figure 2.4. RMBL setup with Velodyne VLP-16. (Adapted from Neumann et al., 2016)

their Wildcat and Hyyti et al. (2019), who use a RMBL setup. MBL sensors have been mounted on vehicles without actuation by, for example, Levinson and Thrun (2014), Nouira, Deschaud, and Goulette (2015) and Pandey, McBride, Savarese, and Eustice (2015).

2.3.1 Measurements to point clouds

This section details the general mathematics for transforming lidar measurements into point clouds in Cartesian coordinates. The section is only concerned with geometry, like this thesis in general, and does not consider further properties of point clouds such as reflectance or colour.

A lidar sensor records polar coordinates for each observation. These consist of a range measurement r, calculated from TOF, and an angle encoding—let us call this the azimuth α—that determines the direction in which the range was measured. This applies to a single-beam sensor, which records two-dimensional data. A multi-beam sensor will additionally have varying elevation angles ω for the different laser beams. These are not measured, as they are constant. The transformation from these polar coordinates to Cartesian coordinates (in the sensor reference frame) is done by the familiar equations:

$$\begin{aligned}\hat{x}_s &= \hat{r}\cos\omega\sin\alpha \\ \hat{y}_s &= \hat{r}\cos\omega\cos\alpha \\ \hat{z}_s &= \hat{r}\sin\omega\end{aligned} \tag{2.1}$$

which results in what shall later be referred to as a single point observation $\hat{\mathbf{x}}_s$ $(= \hat{x}_s, \hat{y}_s, \hat{z}_s)$. The subindex s refers to the fact that the measurement is in the sensor reference frame. In an actuated environment, the sensor reference frame is displaced from the global coordinate system and an additional transformation is required. This six-DOF transformation can be described as:

$$\hat{\mathbf{x}} = \mathbf{R}_{\varphi,\theta,\psi}\hat{\mathbf{x}}_s + \mathbf{t}_e \tag{2.2}$$

where $\mathrm{R}_{\theta,\phi,\rho}$ is a rotation matrix based on the yaw, pitch and roll angles, respectively, and $\mathbf{t}_e$ is the translation vector describing the displacement between the origins of the sensor and global coordinate systems. The subindex e refers to the fact that these transformation parameters are *external* or *extrinsic* to the sensor.

The calibration parameters that are introduced in Section 2.4.2 affect equations (2.1) and (2.2) above by adding coefficients or constants to the recorded variables.

2.3.2 Point cloud distributions

Different scanners and scanner configurations produce different kinds of point clouds. In other words, the distribution of points in space varies largely due to the physical properties of the sensors. For example, typical TLS scanners produce very uniform point clouds where the point density is similar in both angular dimensions. At the same time, MBL devices such as Velodyne scanners produce point clouds that consist of layers—the vertical point density is sparse while the horizontal density can match that of TLS scanners. This is shown by the simulated pattern in Figure 2.5. Real examples of the pattern can be found online or in images presented by Levinson and Thrun (2014), for example. Similarly, SBL sensors produce a single 2D layer of measurements. When utilising actuated devices, the actuation mechanism also affects the point distribution. This can happen in obvious ways, such as a rotating system causing high point density near the axis of rotation, or in more unexpected ways, for example when a rotating system rotates at a rate relative to the internal rotation of the sensor. An example of the latter can be seen in Section 3.3.4.

2.4 Calibration of laser scanner systems

Usually, laser scanners are calibrated using specialised calibration setups with accurately known target geometries and reflectances. This is typically done by the manufacturer of the scanner in question, and concerns the internal makeup of the device (i.e. laser angles and displacements). These are called the *intrinsic* calibration parameters. More complicated laser scanner systems with sensor fusion elements introduce additional calibration parameters (i.e. translations between coordinate systems) which require additional calibration procedures—these are called *extrinsic* calibration parameters. Moreover, black-box factory calibration parameters can often be improved upon with proper self-calibration

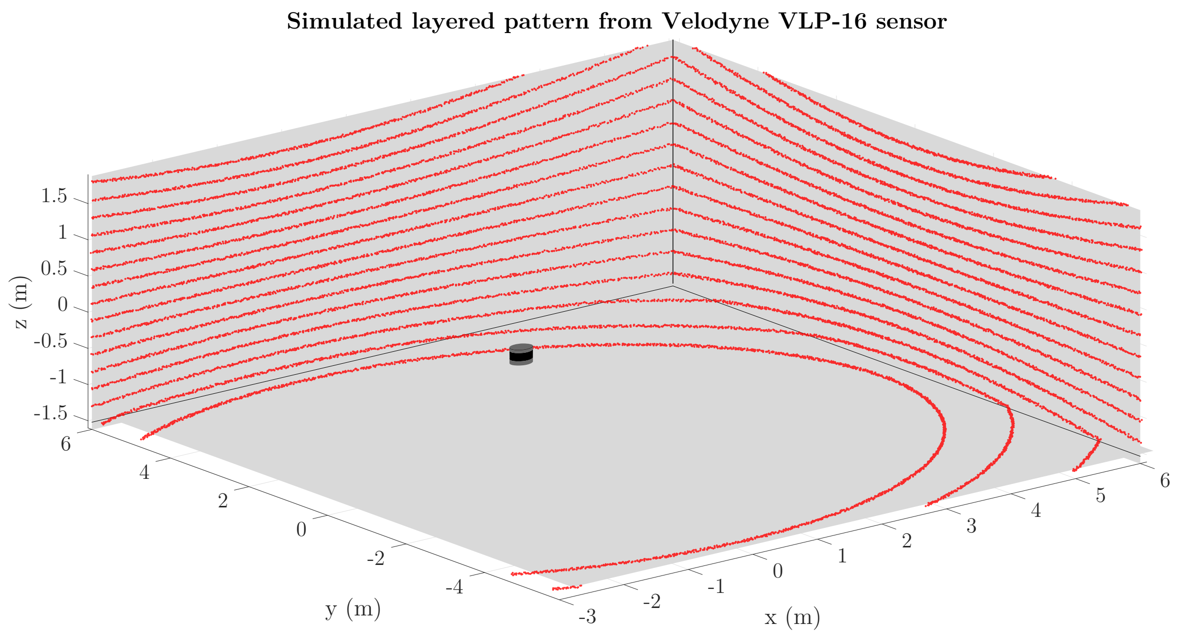


Figure 2.5. The layered distribution of MBL point clouds is caused by the multi-beam setup, where each beam observes the surroundings at a set elevation angle. The figure is a simulated scan of a 12×12 *metre room. The sensor is visualised at the origin at a height of 1.5 m.*

(e.g. Lichti, Stewart, Tsakiri, & Snow, 2000; Schulz, 2008, for TLS devices). Calibration out of the factory environment can be conducted in several ways. These can be classified into:

- Point-based calibration
- Calibration based on known features
- Calibration based on observed features
- Calibration based on some general presumed property of the point cloud

Point-based calibration, mainly used with TLS devices, involves the employment of multiple, typically spherical calibration targets that are placed in precisely known locations in the environment. This is really a subcategory of calibration based on known features, as these targets are known *a priori*. The distinction is that in point-based calibration, the targets are measured as single points instead of more complex geometries, while in what this thesis calls target-based calibration below (i.e. based on known features, targets) targets are three-dimensional (or at least two-dimensional, as in the case of planes). Typical targets contain very distinct corners and edges, and at least walls and boxes have been used, for example by Atanacio-Jiménez et al. (2011) and Muhammad and Lacroix (2010).

The last two types of calibration could be labelled as data-based, as they require no *a priori* information about the environment but rather use the measurements themselves as a starting point. Firstly, calibration can be based on features either manually (e.g. Hyyti et al., 2019) or automatically (e.g. Chan & Lichti, 2013) detected in the point cloud. Typically, these methods locate planes (walls, floors, ceilings) or cylinders (poles, water and gas pipes, columns) in the environment and try to reduce the amount of noise in these features. The final category of calibration methods makes some assumption about the point cloud. Examples of such assumptions could be that there is some underlying distribution to the points—in this case, the problem is in defining that distribution—or that the points form continuous surfaces. Data-based calibration methods are the main focus of this thesis.

2.4.1 Optimisation

A calibration procedure is an optimisation routine. In other words, the aim is to find the optimal set of calibration parameters, defined in some way. This definition is the concern of various calibration methods, which define a metric for the *quality* of the parameters. The expression of this metric is called a cost function (or objective or goal function), which takes as arguments the calibration parameters in order to minimise (or maximise, but the difference is trivial) its value. Some constraints can be applied on the values that parameters can take, if necessary.

The optimisation routine then consists of evaluating this function repeatedly with varying parameters in order to find an optimum. Venter (2010) provides an overview of optimisation algorithms, of which there are many. Different algorithms are suited for different types of problems, depending on the number of parameters, constraints, linearity, dimensionality, differentiability and other properties of the cost function. In the case of lidar calibration, some authors (e.g. Atanacio-Jiménez et al., 2011; C.-Y. Chen & Chien, 2012; Alismail & Browning, 2015) use the Levenberg-Marquadt algorithm (Marquardt, 1963), some (e.g. Morales et al., 2014; Hyyti et al., 2019) the Nelder-Mead algorithm (Nelder & Mead, 1965) and others (e.g. Maddern et al., 2012; Levinson & Thrun, 2014) a grid search approach. The last is applicable only with relatively few calibration parameters, since its complexity increases exponentially (to the power of 3) with parameters. All optimisations for this thesis are carried out using the Nelder-Mead algorithm as implemented in the Matlab function *fminsearch* (The MathWorks, Inc., 2019).

Shortly, the Nelder-Mead method uses a simplex, or polytope, of $n + 1$ vertices (in the n-dimensional optimisation space, n being the number of optimisation parameters), to gradually approach a local minimum. That is, it starts by evaluating the function at $n + 1$ points. The simplex then takes a series of steps, mostly moving the vertex with the highest value through the opposite face of the simplex. Such reflections preserve the volume of the simplex (which can be understood as the area where the optimum is currently being searched for), and in addition to them the simplex can expand or contract where possible or necessary, eventually converging on a minimum. As with other optimisation algorithms, the initial parameter values have an effect on the process, as the algorithm can converge on local minima, depending on the geometry of the cost function. Explanatory visualisations in low dimensions can be found online, for example on Wikipedia (*Nelder–Mead method*, 2019), wherefrom this brief description was also adapted.

2.4.2 Calibration parameters

As mentioned in Section 2.4, the calibration parameters of actuated lidar systems can be divided into intrinsic and extrinsic parameters. The parameters come into play in the derivation of the *kinematic chain* of the sensor in Section 3.2.1. There are many (especially intrinsic) parameters that can be defined, and it is not necessarily useful to try to optimise them all.

Extrinsic calibration parameters define the relationship between the sensor and some global coordinate system. Generally, we can define the transformation from one coordinate system to another in six degrees of freedom (DOF). In this case, all of these can be considered relevant parameters, though in any lidar system setup, certain parameters are likely to be more influential to the calibration. The parameter vector $\mathbf{\Theta}_e$ for extrinsic calibration of a single sensor can be expressed as:

$$\mathbf{\Theta}_e = (x, y, z, \alpha, \beta, \gamma) \tag{2.3}$$

where x, y, z are the translation parameters and α, β, γ the rotation parameters. Adding these parameters to equation (2.2) gives:

$$\hat{\mathbf{x}} = \mathbf{R}_{\varphi+\alpha,\theta+\beta,\psi+\gamma}\hat{\mathbf{x}}_s + \mathbf{t}_e + \mathbf{t}_c \tag{2.4}$$

where $\mathbf{t}_c$ is the calibration translation vector consisting of x, y, z. This is overly complicated, however, because instead of adding calibration parameters to transformation parameters, we can calibrate the transformation parameters themselves:

$$\mathbf{\Theta}_e = (t_x, t_y, t_z, \varphi, \theta, \psi) \tag{2.5}$$

$$\hat{\mathbf{x}} = \mathbf{R}_{\varphi,\theta,\psi}\hat{\mathbf{x}}_s + \mathbf{t}_e \tag{2.6}$$

where t_x, t_y, t_z are the components of the translation vector $\mathbf{t}_e$ that incorporate the x, y, z calibration.

Intrinsic parameters are concerned with the internal dimensions of the sensor. For this purpose, we can examine any single laser in a MBL individually, as in Figure 2.6. For any laser beam we can define six DOF—translations in three dimensions and rotations in three dimensions (yaw, pitch and roll). Additionally, we can define additive and proportional correction parameters to the distance metric, resulting in a total of up to eight calibration parameters per laser, leading to a total maximum of $8 \times 16 = 128$ intrinsic calibration parameters for a 16-beam device and $8 \times 32 = 256$ for a 32-beam device.

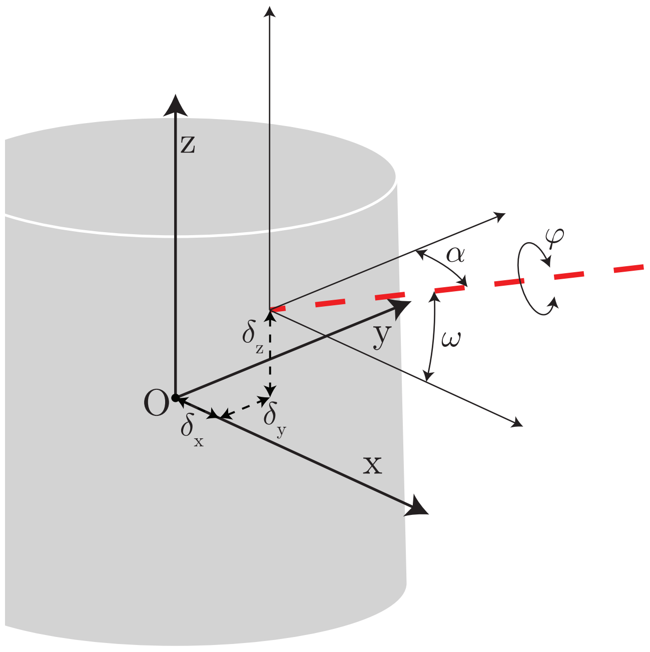


Figure 2.6. Intrinsic calibration parameters of a single laser beam. Sensor coordinate frame origin at O. *6-DOF translation to beam coordinates define calibration parameters.*

However, we can immediately dismiss some of these. First, roll angle η (as depicted in Figure 2.6) is irrelevant as it does not affect the measurement. Second, additive correction parameters to the range measurement can be incorporated into translation parameters t_x, t_y and t_z, as explained by Muhammad and Lacroix (2010). Finally, in the case of a static calibration (where the sensor does not move, as is the case in this thesis), a proportional correction parameter cannot be calculated. Using a moving platform with precise motion metrics would allow this calibration as the relative locations of features would change as a function of time, but this is not in the scope of this work. We thus have the intrinsic calibration parameters:

$$\mathbf{\Theta}_i = (\delta_x, \delta_y, \delta_z, \beta, \omega) \tag{2.7}$$

where the δs are the displacement parameters and β and ω are the horizontal and vertical angles, respectively. When dealing with small sensors such as the SICK, Hokuyo and Velodyne SBLs and MBLs mentioned so far, displacements inside the sensor are necessarily going to be small, and angular displacement is likely to be a larger cause of error. Disregarding the internal translation parameters can therefore be justified, but for completeness, they are kept in consideration here.

Before Section 3 delves into the role these calibration parameters play in the kinematic chain of the lidar system, Sections 2.4.3–2.4.6 provide the theoretical background of the calibration methods used in this thesis.

2.4.3 Target-based calibration

Many proposed calibration methods involve the construction (or existence) of a custom target. For example, Atanacio-Jiménez et al. (2011) and Muhammad and Lacroix (2010) utilise, respectively, a cubic space and a planar wall with accurately known dimensions for calibration. These methods are able to produce good results, but their accuracy is highly contingent on the accuracy with which the dimensions of the target and the sensor relative to the target are known. In addition, these methods require access to a suitable target and a highly sophisticated calibration setup. In many use cases, the construction of such a target and setup or transporting the scanner system to such a target is not practically feasible. Besides, transportation itself may cause small changes in calibration parameters, so a reliable on-site calibration method would be ideal.

An alternative approach to target-based calibration is employed in this thesis. This approach relies on creating a high-accuracy model of a target environment. In this case, the model is created using a high-accuracy TLS device, which is assumed to have good calibration. This removes the burden of building a custom target, but requires a comparatively tedious measurement process, access to a high-accuracy TLS device and a suitable environment of which a high-accuracy model can be retrieved. The use of a separate device allows for an objective reference that is external to the device used for experimentation and that will in this case be representing the so-called ground truth.

In practice, the point cloud observed by the TLS device is used as the target. The target-based calibration procedure then relies on the fact that points in the reference point cloud are evenly distributed and high in density. The cost function is a simple summation of the square distance to the nearest point in the reference point cloud. The assumption here is that the reducible part of this distance metric consists of noise mostly perpendicular to surfaces, so improving the calibration should improve the “sharpness” of the point cloud. Mathematically, we have:

$$E = \sum_{i=0}^{n} d_i^2 \tag{2.8}$$

where d is the distance to the nearest reference point and n is the number of points observed by the sensor being calibrated. Square distance is used for computational efficiency, as it saves having to calculate the square root (distance between two points being calculated as $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$).

2.4.4 Calibration using feature detection

Instead of knowing calibration features prior to conducting measurements, it is possible to use the measurements themselves to define features in the environment that can be used for calibration. The calibration parameters can typically be (and should be) estimated reasonably well before calibration, meaning that measured data should be a rather good approximation of the environment, likely with some excess noise resulting from remaining inaccuracies in calibration parameters. Therefore, it is possible to extract features from the measured data, and aim to improve the definition of these features during calibration.

Feature extraction can be performed manually or automatically. Manual feature detection requires some manual work, but can be rather efficient (see, for example, Hyyti et al., 2019). Manual detection also maintains a high level of confidence in the integrity of the features, as long as the detector is doing her job properly. Automatic feature detection tends to be quite efficient computationally and so can spare some time, but requires some tuning of feature detection parameters and does not always produce consistent results. It is highly dependent on the point distribution and thus the environment, and often some manual oversight is necessary to maintain confidence.

In practice, typical features to be detected are planes and cylinders (e.g. Chan & Lichti, 2013). Planes are simple to find and can be used to define more complicated and precise features such as corners, while locating cylinders is slightly more complex (it would also be possible to detect spheres, for example, but typical environments are less likely to have perfectly spherical than planar or cylindrical features, which are rather abundant in urban environments). However, many of the features in the physical world are not geometrically perfect, so approximating them with perfect shapes is problematic. Therefore, the use of feature detection and matching requires a non-arbitrary environment where there is high confidence in the geometric integrity of the available features.

This thesis employs a plane detection and fitting method. A random sampling and consensus (RANSAC) approach is adopted from Li et al. (2017) for plane detection, while least-squares methods are used for plane fitting. Feature detection in general is a widely studied field, and methods for plane detection that have been proposed mainly include numerous variations of RANSAC (e.g. Torr & Zisserman, 2000; Gallo, Manduchi, & Rafii, 2011) and the Hough transform (e.g. Tarsha-Kurdi, Landes, & Grussenmeyer, 2007; Borrmann, Elseberg, Lingemann, & Nüchter, 2011; Hulik, Spanel, Smrz, & Materna, 2014). RANSAC, in brief, takes a random sample from which it derives a feature, tests this feature against the set, compares it to the best feature so far, and iterates for either a set amount of trials or until a satisfactory feature is found. Efficiency and accuracy can be improved using different heuristic methods. The basic RANSAC algorithm for plane detection and an improved version are described further in Section 4.3.1.

Plane detection results in a parametrised set of planes and points belonging to these planes. The cost function for plane fitting tries to minimise the noise in each of these planes. The errorsum, or cost function value, is the sum of errors that is calculated using principal component analysis—that is, from the eigenvalues of the covariance matrix of the points that are considered to belong to the plane. The eigenvalues describe the variances (in the directions defined by the eigenvectors, with the largest eigenvalue corresponding to the direction of largest variance, the principal component) of the points in the plane. The smallest eigenvector, then, can be interpreted as the variance perpendicular to the plane or the noise of the points in the plane. Mathematically, we have:

$$E = \sum_{p=1}^{N} \lambda_p \tag{2.9}$$

where p refers to the current plane, N is the number of planes and λ is the smallest of the three eigenvalues satisfying the relationship:

$$\lambda \mathbf{v} = \boldsymbol{\Sigma}_{\mathrm{XX}} \mathbf{v} \tag{2.10}$$

where v is the corresponding eigenvector and Σ_{XX} is the covariance matrix for points $\hat{\mathrm{X}}$ in plane p:

$$\boldsymbol{\Sigma}_{\mathbf{XX}} = \mathrm{E}[\hat{\mathbf{X}}\hat{\mathbf{X}}^T] - \mu_{\mathbf{X}}\mu_{\mathbf{X}}^T. \tag{2.11}$$

2.4.5 Calibration using local planarity

In addition to trying to locate well-defined features in the point cloud, it is possible to make assumptions about the properties of the point cloud and use these in the calibration process. One such assumption is small-scale local planarity, which translates to contiguous and continuous surfaces. For example, the kind of RMBL solution such as is discussed in this work will have, at normal operating distances of perhaps 10 meters, a point density in the range of approximately 100-300 points per square meter per rotation. With this in mind, the assumption that any neighbourhood of, say, 20 nearest points will be nearly planar seems reasonable. It is this assumption that Levinson and Thrun (2014) and Nouira, Deschaud, and Goulette (2016) make in their proposed calibration algorithm.

The method is specifically designed for MBL sensors and systems, because it uses the beam-wise distribution of points as a key property of the point cloud. This is because local planarity (among a small neighbourhood), is compared to the closest point observed by a neighbouring beam. Levinson and Thrun (2014) employ a Velodyne HDL-64E sensor and Nouira et al. (2016) a Velodyne HDL-32E (64 beams with 26.9° vertical FOV (Velodyne Lidar, 2008) and 32 beams with 41.3° vertical FOV (Velodyne Lidar, 2015a), respectively). Both of these sensors, but especially the former, have—in MBL terms—relatively high point density in the vertical direction. The data collected by the Velodyne sensors consists of slices observed by individual lasers, which is why point density is nonetheless much higher in the horizontal direction than the vertical.

Levinson and Thrun (2014) first organise the point cloud by beam so that observations made by the same beam are in the same group. For each of these beams i, the algorithm considers the points $\hat{\mathbf{X}}_j$ in N neighbouring beams j on each side. First, it locates the nearest neighbour $\hat{\mathrm{x}}_i$ from the original beam i. Then, it calculates a local normal around this point $\hat{\mathrm{x}}_i$ based on kNN (here, $k = 20$). Finally, it calculates the squared inner product between this normal and the vector $\hat{\mathrm{x}}_i - \hat{\mathrm{x}}_i$, a line segment connecting the two points. Since $\mathrm{a} \cdot \mathrm{b} = 0$ if $\mathrm{a} \perp \mathrm{b}$ the inner product will be equal to zero if the point $\hat{\mathrm{x}}_j$ lies on the plane. It is assumed that the closer to the plane this point is, the more *contiguous* or uniform the point cloud is and the less noise it contains. Thus we have the cost function:

$$E = \sum_{b_i=1}^{B} \sum_{b_j=b_i-N}^{b_i+N} \sum_{k=1}^{n} w_k \|\mathbf{n} \cdot (\hat{\mathbf{p}}_{k,j} - \hat{\mathbf{m}}_{k,i})\|^2 \tag{2.12}$$

where B is the number of beams, N the number of adjacent beams that each beam is aligned to, n is the number of points observed by beam b_j, w_k is a weighting term, given a value of 0 or 1 depending on whether the distance between $\hat{\mathrm{p}}_{k,j}$ and $\hat{\mathrm{m}}_{k,i}$ is under some threshold, n is the normal at $\hat{\mathrm{m}}_{k,i}$, $\hat{\mathrm{p}}_{k,j}$ is the kth point and $\hat{\mathrm{m}}_{k,i}$ is its nearest neighbour observed by beam b_i.

The above formulation of the cost function makes it somewhat confusing due to the three summations. However, the underlying principle of surface continuity is rather straightforward and logical. In fact, we can formulate a simplified version of this cost function to generalise it to a case where points are not observed in a layered fashion. This is the case in this thesis, where the sensor is rotated at an angle and a continuous point cloud (within the FOV) is obtained.

The principle is similar. We can assume that a surface observed by a lidar sensor is continuous, or, more precisely, that the nearest neighbours to any point in a point cloud lie on a relatively smooth and continuous surface. There are, of course, blind spots in point clouds—so all surfaces are not continuous—but to the extent of the few nearest points, this seems reasonable. Additionally, at a local scale of a few to some tens of centimetres, these can be assumed to be nearly planar. To then formulate a cost function that penalises points that lie off of the local plane, we can use a similar method as described in Section 2.4.3. Using the nearest k neighbours of a point, we can use the eigenvalues of the covariance matrix to obtain a measure of planarity. The smallest eigenvalue describes variance in the normal direction to a best-fit plane, so we can directly sum up the smallest eigenvalues of the local covariance matrices as our cost function:

$$E = \sum_{i=1}^{n} \lambda_i \tag{2.13}$$

where n is the number of points in the cloud and λ_i is the smallest of the three eigenvalues of the covariance matrix of the nearest k points to point i (see equations (2.10) and (2.11)).

2.4.6 Calibration using entropy

Similarly to the assumption of local planarity, authors such as Sheehan et al. (2012), Sheehan, Harrison, and Newman (2014), Maddern et al. (2012) and Oberlander et al. (2015) start with the very general assumptions that the distribution of points in an observed point cloud is not random. Indeed, they attempt to define a probability density function (PDF) for the environment and use the information-theoretic measure of entropy as a quantity to be minimised for calibration. In essence, the observed point cloud is treated as a multivariate three-dimensional normal distribution of probabilities, and the result of the optimisation is the set of most-likely parameters.

Information theory is based on the work of Shannon (1948). In information theory, the concept of entropy refers to the expected amount of information delivered by a message or, when generalised to this case, a measurement. Entropy is maximised when the distribution of data points is even—that is, when the probability for a measurement being in any point in the observed space is the same. This is of course not the case when dealing with a physical environment. Points are not randomly distributed in space since their positions are strongly correlated with the actual physical geometry of the environment. The rationale behind the entropy-based approach to lidar system calibration, then, is that a distribution that contains minimum randomness is likely to be the most accurate. Many authors describe this sought-after quality—this lack of randomness—as point cloud *crispness*, though the term is rather vague and as such will not be used in this work.

In the foundational literature of information theory, Shannon (1948) introduces the concept of information entropy as analogous to entropy in physics and as a measure of the uncertainty involved in choosing a random sample from a set. Shannon defines entropy H mathematically as:

$$H = -K \sum_{i=1}^{n} p_i \log p_i \tag{2.14}$$

where K is some constant, n is the number of possible outcomes and p_i is the probability for each outcome. This concept of entropy was generalised by Rényi (1960) to the following:

$$H_\alpha = \frac{1}{1-\alpha} \log_2 \left(\sum_{i=1}^{n} p_i^\alpha \right) \tag{2.15}$$

where α is what Rényi calls the order of the entropy measure i.e. H_α is entropy of order α. Note that the logarithmic term is outside the summation, but this is a consequence of the relationship between the two entropy measures and can be verified by calculating the limit at $\alpha \to 1$ using L'Hôpital's rule (Shannon entropy).

Shannon entropy, equation (2.14), is thus a special case of Rényi entropy at $\lim_{\alpha \to 1}$. Another special case is called Rényi Quadratic Entropy (or just Quadratic Entropy), when $\alpha = 2$. In this case, there is total reliance on the quadratic form of probabilities p_k^2. Principe and Dongxin Xu (1999) point out that quadratic entropy is appealing in the case of a continuous variable as it leads to more simple calculations. In the case of measurements of a physical environment, our variables are naturally continuous.

The generalisation of Shannon's entropy measure to continuous variables was proposed by Shannon himself as:

$$h(x) = -\int_S p(x) \log p(x) dx \tag{2.16}$$

However, as shown by Jaynes (1963), this is not correct—strictly speaking—as a limit of Shannon's expression. Nonetheless, it has become the measure known as differential entropy, though, unlike its predecessor, it can take negative values and is not invariant to a change of variables. For these reasons it is denoted here as $h(x)$. It is also the measure adopted by Sheehan et al. (2012), Maddern et al. (2012) and Oberlander et al. (2015) for calibration purposes. In fact, these authors turn to a similar generalisation of the quadratic measure of entropy as in equation (2.15), written explicitly below:

$$h(x) = -\log_2 \int_S p(x)^2 dx \tag{2.17}$$

The main difference between the discrete and continuous versions of the entropy equation is the use of a probability density function $p(x)$ instead of probability p_k. When applied in the case of laser scanner measurements, we have a discrete number n of measurements $\hat{\mathrm{x}}$ (constituting $\hat{\mathrm{X}}$) of a continuous random variable with PDF $p(x)$. We are then faced with the task of determining this PDF. Previous authors choose to represent the distribution as a Gaussian mixture model, obtained by applying a kernel density estimation (KDE) method utilising a Gaussian kernel at each measurement point. The KDE method (called Parzen or Parzen-Rosenblatt Window estimation after Rosenblatt (1956) and Parzen (1962) by some of the authors) is a data smoothing method used for this purpose, to estimate the PDF of a random variable. Mathematically it is expressed as:

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_\xi(x - x_i) = \frac{1}{n\xi}\sum_{i=1}^{n} K\Big(\frac{x - x_i}{\xi}\Big) \tag{2.18}$$

where K is the kernel, ξ is a positive smoothing parameter—the bandwidth—and K_ξ denotes a scaled kernel. The bandwidth can be set to 1, which eliminates it from the equation. The Gaussian kernel for data point x is an implementation of the multivariate Gaussian distribution function:

$$G(\mathbf{x} \mid \mu, \Sigma) = \frac{e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)}}{\sqrt{(2\pi)^k|\Sigma|}} = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}}e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)} \tag{2.19}$$

where the distribution is defined by μ and Σ, the mean vector and covariance matrix, respectively. The following notation is also prevalent (Principe & Dongxin Xu, 1999):

$$G(\mathbf{x} - \mu, \Sigma) = G(\mathbf{x} \mid \mu, \Sigma) \tag{2.20}$$

However, for clarity, let us adhere to the right-hand side form.

In previous laser scanner calibrations, equation (2.19) is plugged into the KDE equation (2.18) with bandwidth 1. Additionally, Σ is defined as $\Sigma = \sigma^2\mathrm{I}$. Diagonalising the covariance matrix implies independent variables while equalising the variances implies an isotropic distribution. These are intuitively rather reasonable assumptions in this case, since there is no *a priori* knowledge of the environment that is being modelled. Additionally, any biases in the measurement equipment will not translate consistently into biases in Cartesian coordinates (x, y, z variances being in the diagonal) as measurements are recorded in polar coordinates and only later transformed. Thus we have the PDF:

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} G(\hat{\mathbf{x}}_i \mid \mathbf{x}, \sigma^2 \mathbf{I}) \tag{2.21}$$

Now, we can inject this function into Rényi's entropy measure from equation (2.17):

$$h(\mathbf{x}) = -\log_2 \int_S \Big(\frac{1}{n} \sum_{i=1}^{n} G(\hat{\mathbf{x}}_i \mid \mathbf{x}, \sigma^2 \mathbf{I}) \Big)^2 dx \tag{2.22}$$

Expanding and rearranging:

$$h(\mathbf{x}) = -\log_2 \Big(\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \int_S G(\hat{\mathbf{x}}_i \mid \mathbf{x}, \sigma^2 \mathbf{I}) G(\hat{\mathbf{x}}_j \mid \mathbf{x}, \sigma^2 \mathbf{I}) dx \Big) \tag{2.23}$$

Here we see the convolution of two Gaussians, which—following Bromiley (2013) and Principe and Dongxin Xu (1999)—can be expressed as:

$$\int_S G(\hat{\mathbf{x}}_i \mid \mathbf{x}, \sigma^2 \mathbf{I}) G(\hat{\mathbf{x}}_j \mid \mathbf{x}, \sigma^2 \mathbf{I}) dx = G(\hat{\mathbf{x}}_i \mid \hat{\mathbf{x}}_j, 2\sigma^2 \mathbf{I}) \tag{2.24}$$

This, when combined with equations (2.23) and (2.19), yields the cost function:

$$h(\hat{\mathbf{x}}) = -\log_2 \left(\frac{1}{n^2 \sqrt{(2\pi)^k |2\sigma^2 \mathbf{I}|}} \sum_{i=1}^{n} \sum_{j=1}^{n} e^{-\frac{1}{2}(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)^T (2\sigma^2 \mathbf{I})^{-1} (\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)} \right) \tag{2.25}$$

For optimisation purposes, we can simplify the function further by removing the monotonic logarithmic operator as well as the constant coefficient. This is because optimisation aims to minimise the cost function, and the exact value of the cost function is irrelevant, as long as it behaves similarly when parameters are adjusted. Thus, we are left with the cost function:

$$\begin{aligned} E &= -\sum_{i=1}^{n}\sum_{j=1}^{n} e^{-\frac{1}{2}(\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j)^T(2\sigma^2\mathbf{I})^{-1}(\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j)} \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{n} e^{-\frac{1}{\sigma^2}(\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j)^T(\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j)} \end{aligned} \tag{2.26}$$

from which we can see that the entropy-based measure relies purely on pairwise distances $(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)$ between points in the observed measurement set $\hat{\mathrm{X}}$ and free variable σ. In other words, optimisation via entropy aims to minimise point–point distances across the entire point cloud with heavy emphasis on nearby points (due to σ).

3. Device and experimental setup

This section presents the RMBL device that is being calibrated using the methods presented in Section 2.4 and the experimental setup, i.e. the environment being scanned and the methods related to scanning. Though the tests are conducted in the context detailed here, the calibration methods can be generalised to other laser scanner systems as long as the calibration parameters being optimised can be sufficiently defined from the observations.

3.1 Reference scanner

Figure 3.1. Leica ScanStation P40. (Leica Geosystems, 2018)

The reference measurements are conducted by a Leica ScanStation P40 TLS device (Leica Geosystems, 2018), pictured in Figure 3.1. It is a TOF device that ships with a wavelength of either 1550 nm or 658 nm. The device employed here was equipped with the invisible wavelength 1550 nm. According to the manufacturer, the P40 is capable of an accuracy of 0.8 mm at a distance of 10 m. The scanner is capable of measuring up to a million points per second, and functions at a range of up to 120 or 270 metres (depending on the settings). The FOV of the scanner is 360° horizontally and 270° vertically. There is also an integrated camera that captures high dynamic range (HDR) images. In this case, the scanner was used with resolution setting 3.1 mm at 10 m, normal sensitivity and speed mode.

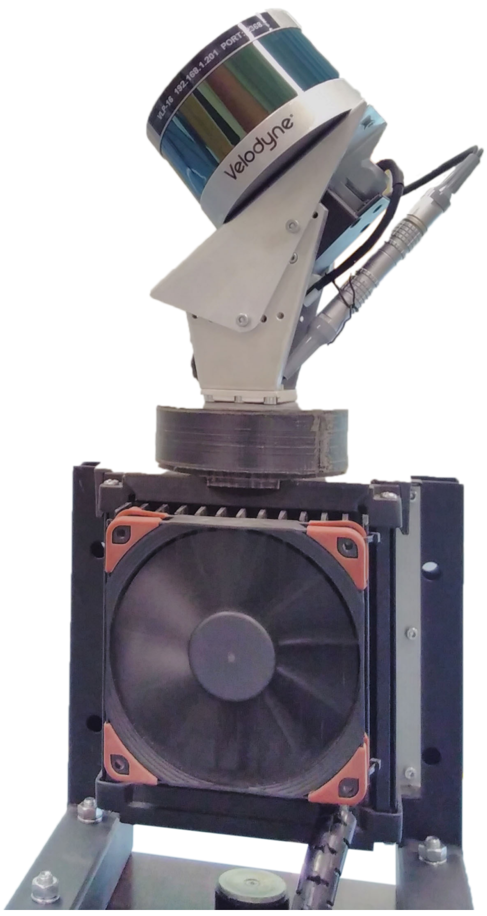


Figure 3.2. The RMBL device used to actuate the Velodyne scanners.

3.2 Custom RMBL system

The setup used here is the same setup initially presented by Hyyti et al. (2019) and is pictured in Figure 3.2. Hyyti et al. (2019) provide a detailed description of the components of the scanner, so the particulars will be omitted here. Generally, the system consists of a Velodyne VLP-16 Puck LITE multi-beam lidar sensor (Velodyne Lidar, 2018a), an IMU sensor (integrated inside the bottom box), a brushless gimbal motor, a magnetic encoder system, and a computer. The sensor is placed on a tilted platform (the angle of which can be adjusted but is here set at 40°), which can be rotated by the motor. The sensor can also be switched and replaced, and the measurements for this thesis were conducted using one Puck LITE sensor, 4 regular Puck sensors (Velodyne Lidar, 2015b) and one 32-beam VLP-32C Ultra Puck device (Velodyne Lidar, 2018b). During measurement, the computer records measurements and the data is processed online into xyz-coordinates. However, calibration is performed offline.

Figure 3.3 shows the organization of the beams inside the VLP-16 and the VLP-32C. The VLP-16 Lite is identical to the VLP-16 in this respect. As can be seen,

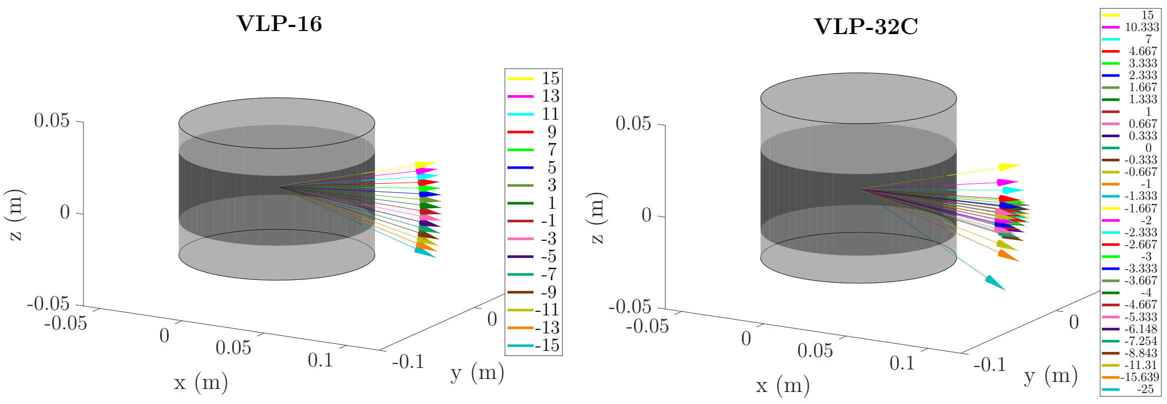


Figure 3.3. Laser beam structure inside the Velodyne VLP-16 and VLP-32C as detailed by the manufacturer.

the lasers in both sensors are presumed to originate in the same location (the origin). In the VLP-16, all have a default azimuth offset of 0°, while in the VLP-32C, there are azimuthal offsets in the construction. The vertical distribution in the VLP-16 is even while there is an obvious concentration in the middle in the VLP-32C.

3.2.1 Kinematic chain

This section derives the specific kinematic chain for the device introduced above. As seen in Section 2.3.1, measurements can be transformed into Cartesian xyz coordinates in the scanner reference frame using the equations:

$$\begin{aligned}\hat{x}_s &= \hat{r}\cos\omega\sin\alpha\\ \hat{y}_s &= \hat{r}\cos\omega\cos\alpha\\ \hat{z}_s &= \hat{r}\sin\omega\end{aligned} \tag{3.1}$$

where α is the azimuth angle, recorded by the lidar, ω is the elevation angle, constant for any individual laser beam and r is the range measurement obtained from the TOF calculation. The angles ω are calibration parameters, in addition to which the other intrinsic calibration parameters can be inserted as:

$$\begin{aligned}\hat{x}_s &= k\hat{r}\cos\omega\sin(\alpha+\beta)+\delta_x\\ \hat{y}_s &= k\hat{r}\cos\omega\cos(\alpha+\beta)+\delta_y\\ \hat{z}_s &= k\hat{r}\sin\omega+\delta_z\end{aligned} \tag{3.2}$$

where k is a proportional correction parameter, the δ's are the intrinsic calibration parameters that define translation offsets and β is the intrinsic calibration parameter that defines the error of the azimuth angle. A point in sensor coordinates will be denoted as $\hat{\mathbf{x}}_s$:

$$\hat{\mathbf{x}}_s = \begin{bmatrix} \hat{x}_s \\ \hat{y}_s \\ \hat{z}_s \end{bmatrix} = k\hat{r} \begin{bmatrix} \cos\omega \sin(\alpha+\beta) \\ \cos\omega \cos(\alpha+\beta) \\ \sin\omega \end{bmatrix} + \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} \tag{3.3}$$

It should be remembered that the above equation presents the calibration parameters for *one* lidar beam, so generalising to a multi-beam device multiplies the number of calibration parameters.

We now have a representation of our observations in the sensor coordinate system. This needs to be further transformed to a global coordinate system. This can be done as a 6-DOF transformation, but the transformation is not constant because the sensor coordinate system is constantly rotating with the sensor, while the global coordinate system should remain stationary.[1]

First, we introduce the rotation matrix $\mathbf{R}$:

$$\mathbf{R} = \begin{bmatrix} \cos\varphi\cos\theta & \cos\varphi\sin\theta\sin\psi - \sin\varphi\cos\psi & \cos\varphi\sin\theta\cos\psi + \sin\varphi\sin\psi \\ \sin\varphi\cos\theta & \sin\varphi\sin\theta\sin\psi + \cos\varphi\cos\psi & \sin\varphi\sin\theta\cos\psi - \cos\varphi\sin\psi \\ -\sin\theta & \cos\theta\sin\psi & \cos\theta\cos\psi \end{bmatrix} \tag{3.4}$$

where φ, θ, ψ are the yaw, pitch and roll angles, respectively, and which (the rotation matrix) is the ZYX realisation of the Tait-Bryan angles. These angles get initial values of 0°, 40° and 0°, respectively, and are extrinsic calibration parameters. We apply the rotation to $\hat{\mathbf{x}}_s$ and add an offset term:

$$\hat{\mathbf{x}}_p = \mathbf{R}\hat{\mathbf{x}}_s + \mathbf{t}_e \tag{3.5}$$

where $\mathbf{t}_e$ is the translation vector composed of x, y, z offsets, which are all calibration parameters, and $\hat{\mathbf{x}}_p$ is a dummy variable to illustrate this part of the transformation. Finally, we consider the rotation of the platform with another

[1]No difference is drawn here between a coordinate system that stays stationary relative to the device (as a whole) and a global coordinate system, as the device is stationary throughout the experiment. In a deployment setting, where the device is used in motion, a further transformation—based on the motion—is required to reach an invariant coordinate system.

rotation matrix $\mathbf{R}_p$ to get global coordinate point $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}} = \mathbf{R}_p(\mathbf{R}\hat{\mathbf{x}}_s + \mathbf{t}_e), \quad \mathbf{R}_p = \begin{bmatrix} \cos\mu & -\sin\mu & 0 \\ \sin\mu & \cos\mu & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

where μ is rotation angle of the platform, recorded by the motor encoder. Thus we reach the final global coordinates with which the calibration algorithms can be computed.

Finally, let us examine the calibration parameters that were defined. The maximum set of intrinsic calibration parameters is:

$$\Theta_{intrinsic} = \bigcup_{i=1}^{b} \Theta_i, \quad \Theta_i = \{\omega_i, \beta_i, \delta_{xi}, \delta_{yi}, \delta_{zi}, k_i\} \tag{3.7}$$

where b is the number of beams in the sensor and Θ is the set of calibration parameters, Θ_i being the beam-specific set. The total number of intrinsic calibration parameters, then, is $6b$. However, the proportional correction parameter k cannot be determined from stationary measurements using a data-based method and so will be disregarded (or rather, assigned a constant value of 1) in this thesis. When using target-based calibration, these parameters could also be determined, but results show that the value is not likely to change significantly.

Extrinsic calibration parameters are similar to those presented in Section 2.4.2. However, the translation in the z direction does not change the geometry of the cloud, only its position relative to the origin, so it cannot be determined based on data, either. Extrinsic calibration parameters thus are:

$$\Theta_{extrinsic} = \{\varphi, \theta, \psi, t_x, t_y\} \tag{3.8}$$

where t_x, t_y are the elements of $\mathbf{t}_e$ along with t_z, which is held constant.

3.3 Environment and setup

This section explains the experimental setup used for the empirical comparison of the performance of the calibration methods presented in Section 2.4 and detailed in Section 4.

3.3.1 Environment

The experimental measurements used in this thesis were conducted in an indoor gym during one afternoon in September 2019. The space was chosen for some key properties: namely, large size for measurements at a wide range, plenty of planar surfaces for feature-based calibration, and few glass surfaces. Large size is important because the sensors have a range of 100 m (Velodyne Lidar, 2015b); though the gym cannot provide such distances, a range of measurements from approximately 1 to 25 metres can be obtained. Large planar surfaces are necessary for the feature-based calibration algorithm. They may also improve results from calibration based on local planarity. Finally, the lack of glass surfaces reduces ambiguity in the point cloud.

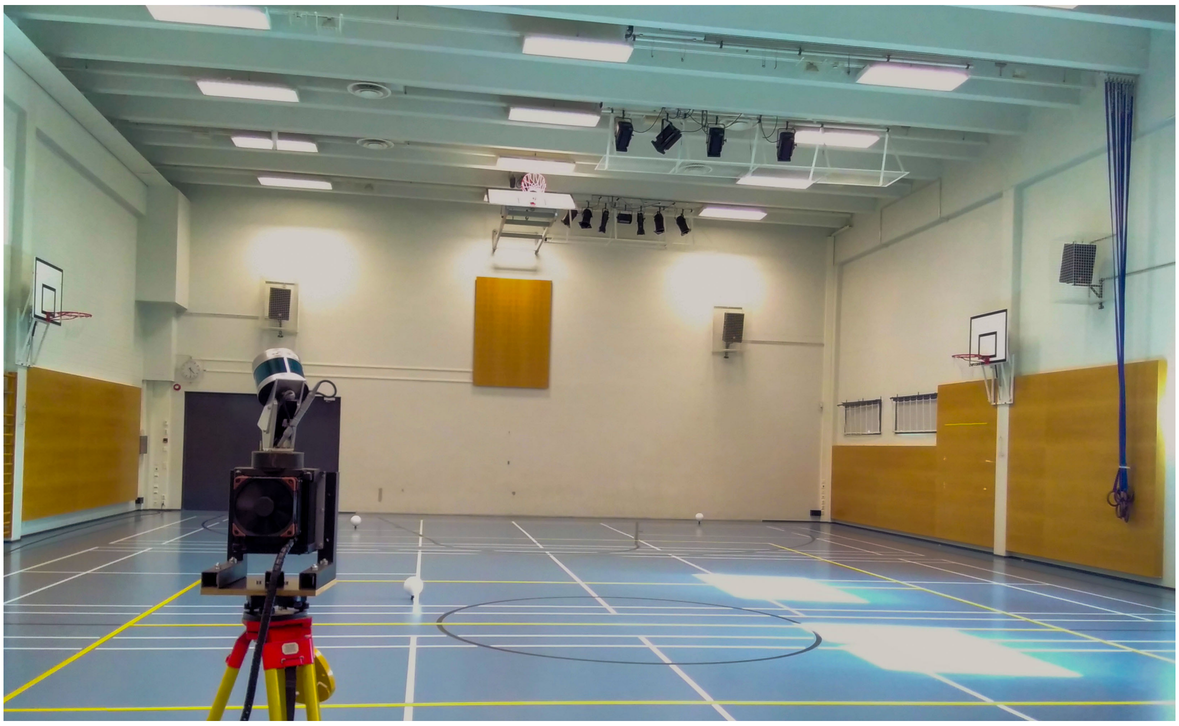


Figure 3.4. The RMBL system on a tripod in the test environment.

Figure 3.4 shows the gym used as the measurement environment. The gym is approximately $30 \times 16 \times 8$ meters in size. One wall is largely covered by gymnastics wall bars, while others are mainly flat except for some individual features. The floor is also flat, but the ceiling contains basketball hoops, lights, beams, and other discrepancies. For the purposes of identifying large planar surfaces, the walls and floor are most useful.

Though it is true that a very specific choice for experimental environment and the use of a single calibration environment undermine the universality of the results, this thesis limits itself to the comparison of these methods in this environment. This limitation allows robust comparisons between reference calibration and experimental algorithms. The applicability of the calibration procedures in other circumstances has been investigated in previous work and is a likely topic for future research in the area. The results of this thesis can perhaps be used as a starting point.

3.3.2 Scanner setup

Figure 3.5 shows the approximate positioning of the laser scanners in the gym. Position 1 describes the location of the RMBL device while positions 2 and 3 were supplementary for TLS scanning. The position was chosen to achieve a large variance in the measured ranges to simulate practical environments where objects are observed at a variety of distances. The sensors are placed on a tripod, which is held at a fixed location. This means that there is theoretically no xy-displacement in the RMBL's and TLS scanner's origins (though the height of their optical center differs), which facilitates comparisons of point clouds from different measurements. The Velodyne sensors have similar diplacements in the z direction as well. In practice, small shifts in position are likely to occur when the sensor is replaced.

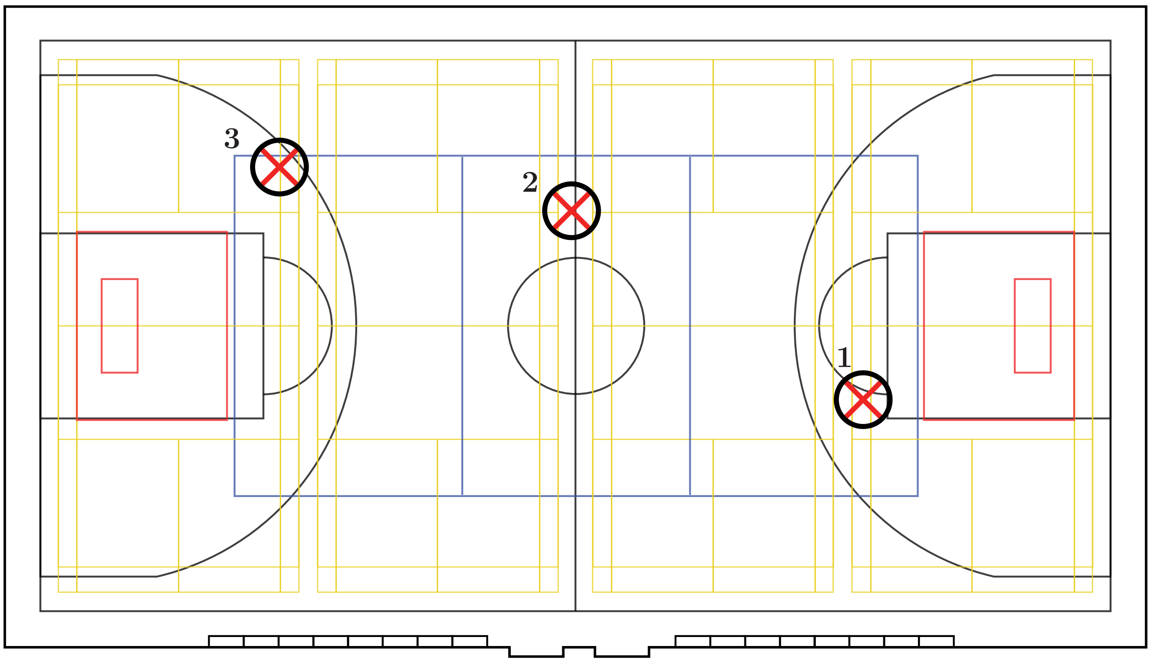


Figure 3.5. Floor plan of the gym and placement of tripod for three scans. RMBL measurements were made at position 1, while TLS measurements were made at positions 1–3. Scale is approximate.

In addition to calibration measurements in one location, reference measurements were made using the TLS scanner described in Section 3.1. These consist of high-accuracy scans in three locations, also shown in Figure 3.5. The scans were combined to produce a high-resolution point cloud of the entire space, which was used as the target in target-based calibration.

3.3.3 TLS measurement processing

For TLS referencing, spherical targets were placed in the scene during TLS scanning. The three scans were matched using these targets to produce a single high-density point cloud. Matching was done with Leica Cyclone software (Leica Geosystems, 2019). For processing efficiency, a sample from the point cloud was extracted for further analysis. The size of the original point cloud was approximately 460 million points, and for target-based calibration this was sampled down to a hundredth by randomly choosing 1% of the points for reasons of efficiency.

3.3.4 Point cloud measurements

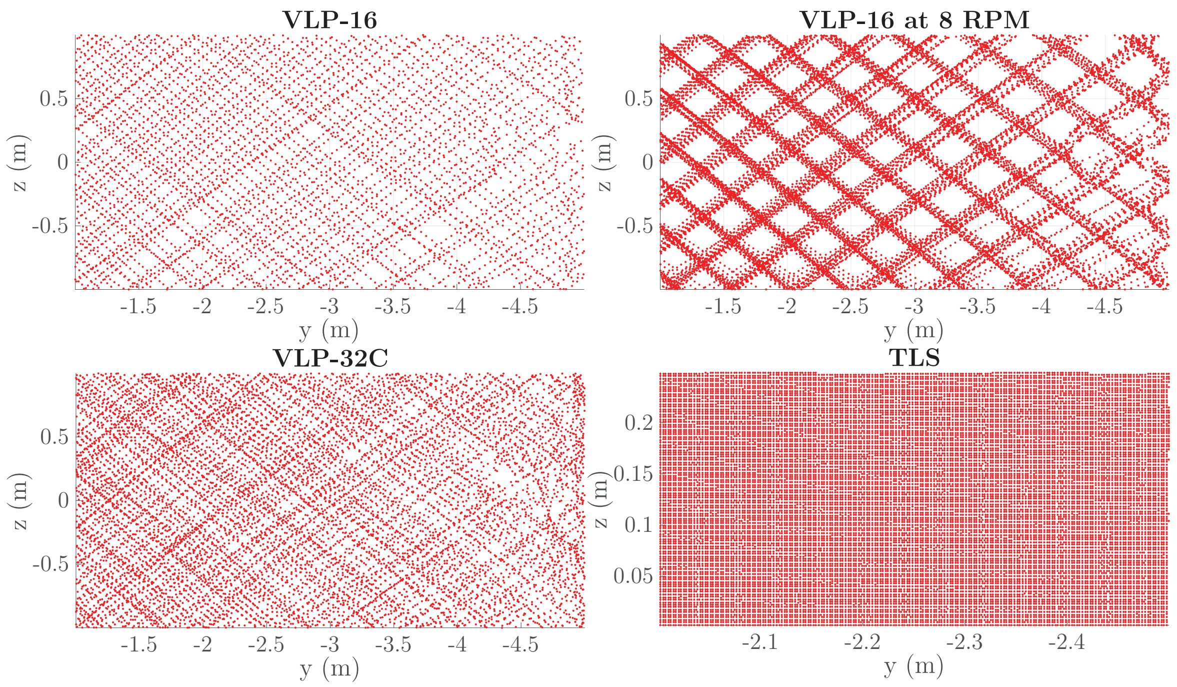


Figure 3.6. Point cloud distributions from different sensors and settings from a perpendicular view of a flat wall.

Figure 3.6 shows the distribution of points in different measurements of the experimental environment. It is from a wall approximately 8 meters from the scanner. The left-hand side clouds depict the sensors and scans used for experimental purposes in this thesis—Velodyne sensors mounted on the RMBL rotating at 29 rounds per minute (RPM). As can be seen, the 32-beam sensor has roughly double the point density. The right-hand side shows more dense clouds. The scan from the RMBL at 8 RPM shows how the layered pattern of the VLP-16 sensor becomes visible at smaller rotation speeds. This is manifested as a grid-like pattern because the cloud is composed from one full rotation of the RMBL which covers two oppositely tilted views of one direction. The TLS point cloud is of a much smaller area than the others, since at the same resolution it would look simply solid. The high resolution of the TLS sensor is apparent in the axes scales and its uniformity in the pattern.

To illustrate the difference in precision between the RMBL system and the

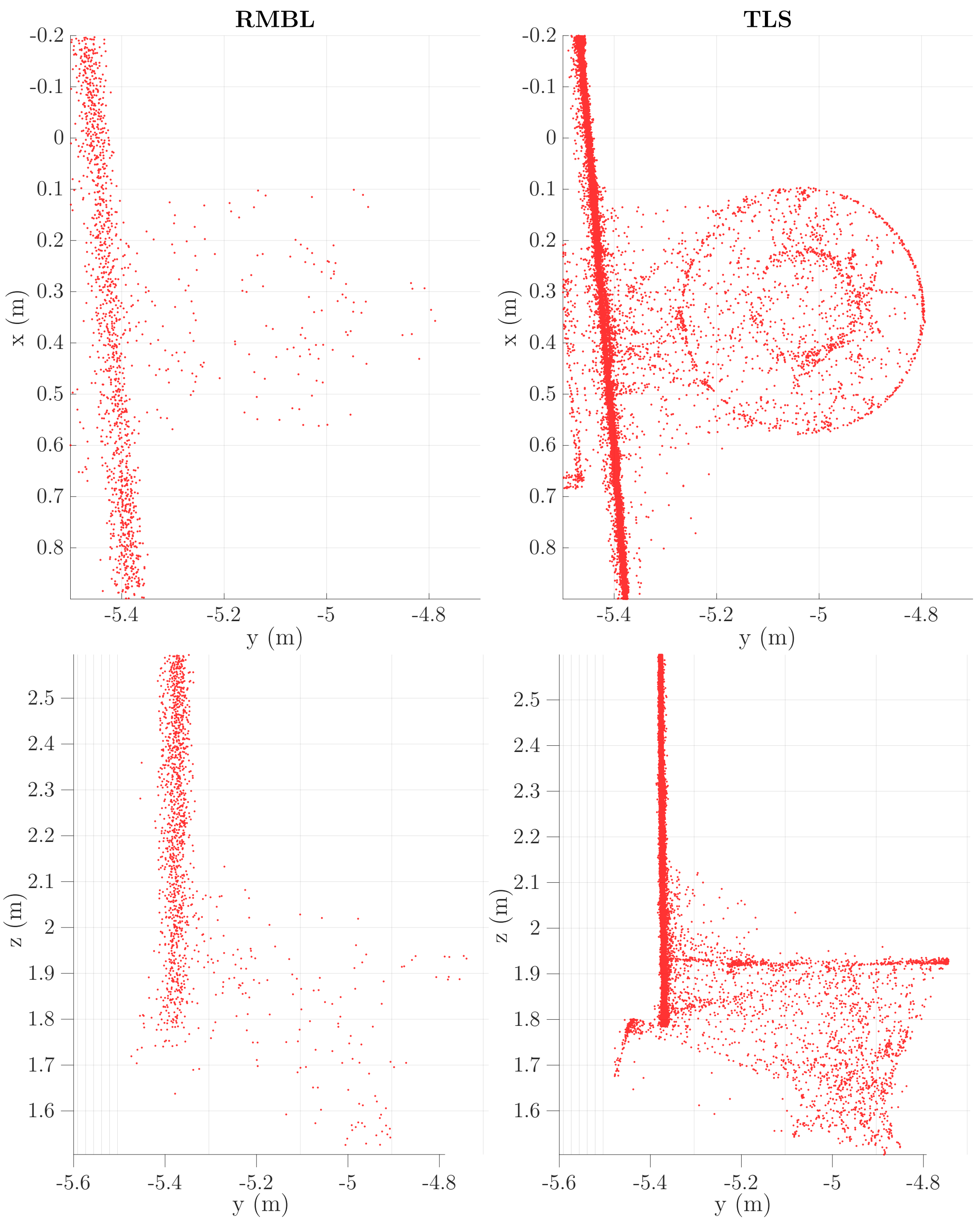


Figure 3.7. A detail from the experimental scene depicting the imprecision and low point density of the RMBL compared to TLS. The TLS data has been decimated to one tenth of original point density.

TLS scanner, Figure 3.7 shows a detail from the experimental environment—a basketball hoop—from two perspectives as measured by the two devices. The RMBL sensor for which the cloud is visualised is a Velodyne VLP-16. The TLS cloud has been randomly decimated to $\frac{1}{10}$ of the points for visualisation purposes, so the actual TLS cloud is even more dense. The RMBL cloud is much more sparse, in addition to which there is a lot more noise in the features, which is especially evident in the backboard of the hoop. At the same time, the scattered TLS points show that the TLS scan, too is subject to imprecisions, especially with the intricate details of hoop (i.e. ropes and small metal bars), which cause multiple and ambiguous echoes.

4. Proposed methodology

This section explains the calibration methods examined in this thesis, providing detailed explanations of the calibration algorithms implemented. Each algorithm is described in words and presented schematically. Pseudocode is available in Appendix 7.

4.1 Optimisation routine

As mentioned before, the values of the cost functions described in the previous sections are minimised using MATLAB's *fminsearch* function. The function uses a default value of $200p$ iterations, where p is the number of optimisation parameters. In the calibrations run for this thesis, there are 85 calibration parameters for a 16-beam Velodyne Puck (and 165 for the 32-beam Ultra Puck), and $200 \times 85 = 17,000$. Most of the functions did not converge in this timeframe, however, so the *fminsearch* function was called 5 times with 10,000 iterations each for the 16-beam sensors and 15,000 iterations each for the 32-beam sensor. The separate function calls were used to reset the simplex so it might possibly find a different minimum. Experimentally, this was found to speed up optimisation and find minimas with lower values, as seen in the convergence plot in Figure 4.1. The peaks in the blue line appear at times when the function is restarted, as the function takes larger steps in mostly higher-value directions. The restart does help find a larger gradient as well, however, as the divergence of the two plots shows. The graph was made using the plane fitting method to illustrate this point and does not reflect general convergence properties. As described in Section 5.4 and displayed in Figure 5.2, different methods behave very differently as well.

The cost functions were implemented in both MATLAB (plane detection and fitting) and in C++ (target-based, local planarity and entropy) as MATLAB MEX functions (*MEX File Functions - MATLAB & Simulink - MathWorks Nordic*, 2019). This allowed reductions in computation time, as the cost function evaluations involve many a lot of looping, which is inefficient in MATLAB. Additionally, the MEX functions utilised the C++ libraries *nanoflann* (Blanco & Rai, 2014) and *Eigen* (Guennebaud, Jacob, & others, 2010). *nanoflann* is a simplified fork of *FLANN* (Muja & Lowe, 2013), used for nearest neighbour searches, while *Eigen* was used for matrix calculations.

4.2 Target-based calibration

Target-based calibration is mathematically simple, but requires an unambiguous definition of the target. In this case, the target was a room, which was modelled using a TLS scanner as described in Section 3. The TLS device was assumed to be well calibrated. The resulting point cloud was then used as a true model of the room, called the reference point cloud. For calibration purposes, this reference was compared against the point cloud gathered by the RMBL device, and calibration

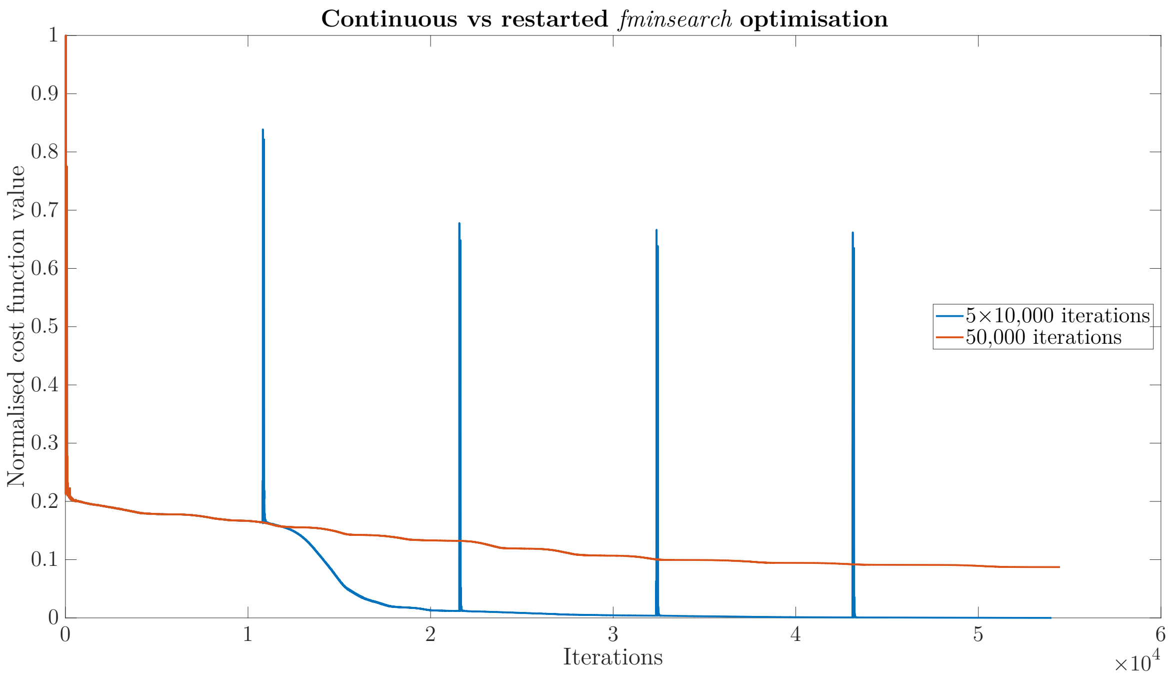


Figure 4.1. Convergence plot for plane segmentation and fitting for VLP-16 scanner (4). Cost function value on the y-axis is normalised to be in range $[0, 1]$.

parameters were optimised by minimising point–point distances between the reference and observed environments. In addition to the intrinsic and extrinsic calibration parameters described in Section 2.4.2, a 6-DOF transformation for matching the calibration and reference point clouds was also computed during this optimisation process, since any mismatch in the origin of the clouds would likely lead to compensation by changing the calibration parameters. This removed the need to make a transformation before optimisation, as a well calibrated point cloud might fit slightly differently to the reference than an uncalibrated one.

The target-based calibration algorithm is described by Figure 4.2. The algorithm is simple: it iterates through all measurements, finds the nearest neighbour in the reference point cloud, and sums the squared distances to these neighbours. The target-based cost function is also used as a universal metric to score other calibration methods by calculating its value for the optimised parameters obtained using the other methods.

4.3 Data-based calibration

4.3.1 Plane detection and fitting

Similarly to target-based calibration, sensors can be calibrated by first using the measurements to model the environment and then using this model as a calibration target. The most straightforward approach is to find planar features in the point cloud and, tuning the calibration parameters, minimise errors in these planes for relevant points. This section describes the plane detection and

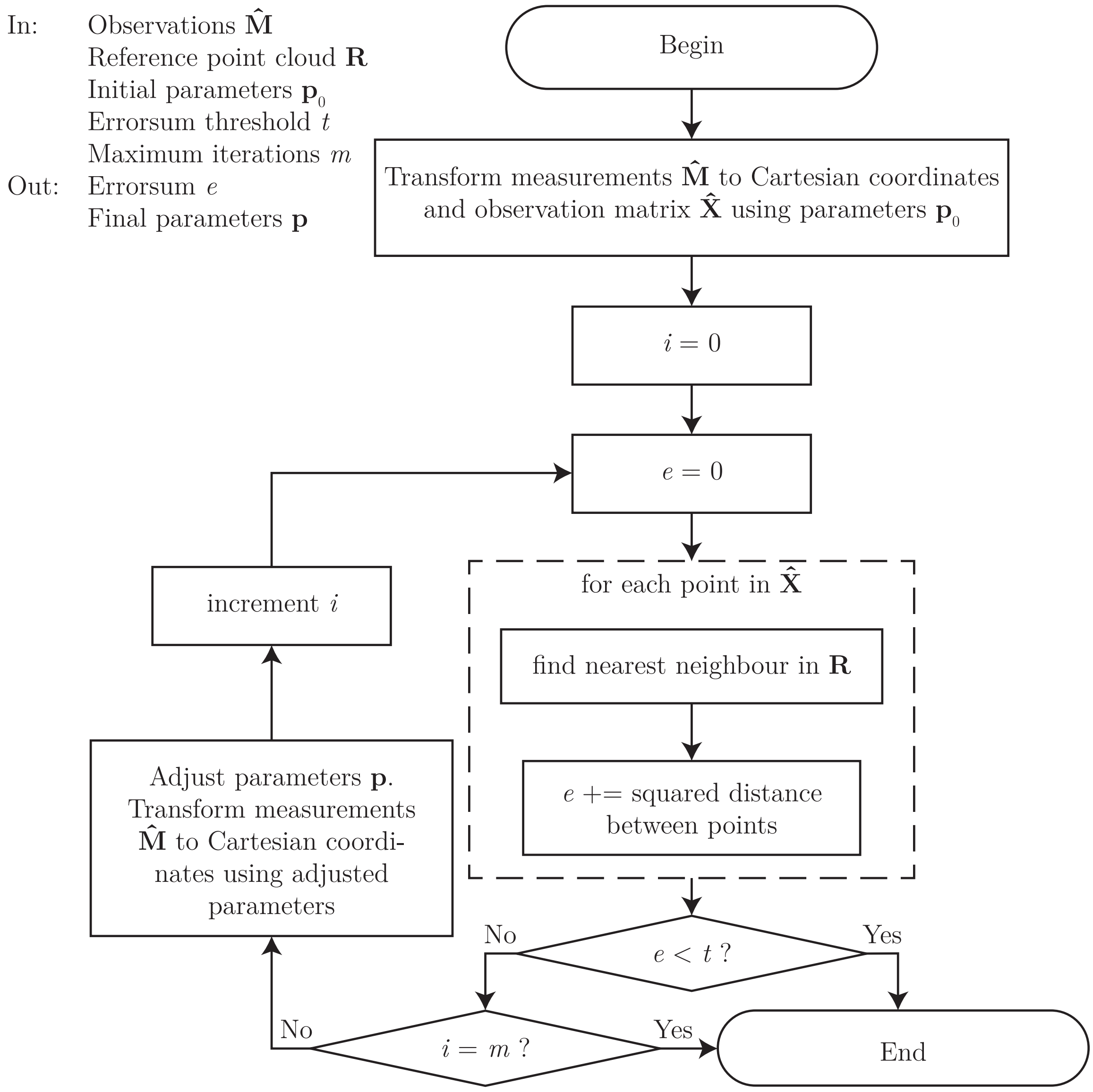

In:
Observations $\hat{\mathbf{M}}$
Reference point cloud $\mathbf{R}$
Initial parameters $\mathbf{p}_0$
Errorsum threshold t
Maximum iterations m
Out:
Errorsum e
Final parameters $\mathbf{p}$
Begin
Transform measurements $\hat{\mathbf{M}}$ to Cartesian coordinates and observation matrix $\hat{\mathbf{X}}$ using parameters $\mathbf{p}_0$
$i = 0$
$e = 0$
increment i
for each point in $\hat{\mathbf{X}}$
find nearest neighbour in $\mathbf{R}$
e += squared distance between points
Adjust parameters $\mathbf{p}$. Transform measurements $\hat{\mathbf{M}}$ to Cartesian coordinates using adjusted parameters
No
$e < t$?
Yes
No
$i = m$?
Yes
End

Figure 4.2. Target-based calibration using a reference point cloud and point–point distances. Available in pseudocode in Algorithm 1 in Appendix B.1.

fitting algorithms implemented for this thesis.

There are several ways to automatically find planes in a point cloud. Most implementations tend to be variations of the random sampling and consensus (RANSAC) algorithm. As its name suggests, RANSAC relies on randomly sampling a set of points, fitting a plane to these points and testing the validity of this plane against the other points in the data. The planes receive 'votes' based on the number of points that fit to them (i.e. are a reasonable distance away). A naïve version would sample 3 points to form the plane while improved versions use, for example, a neighbourhood of n points or a maximum likelihood estimator.

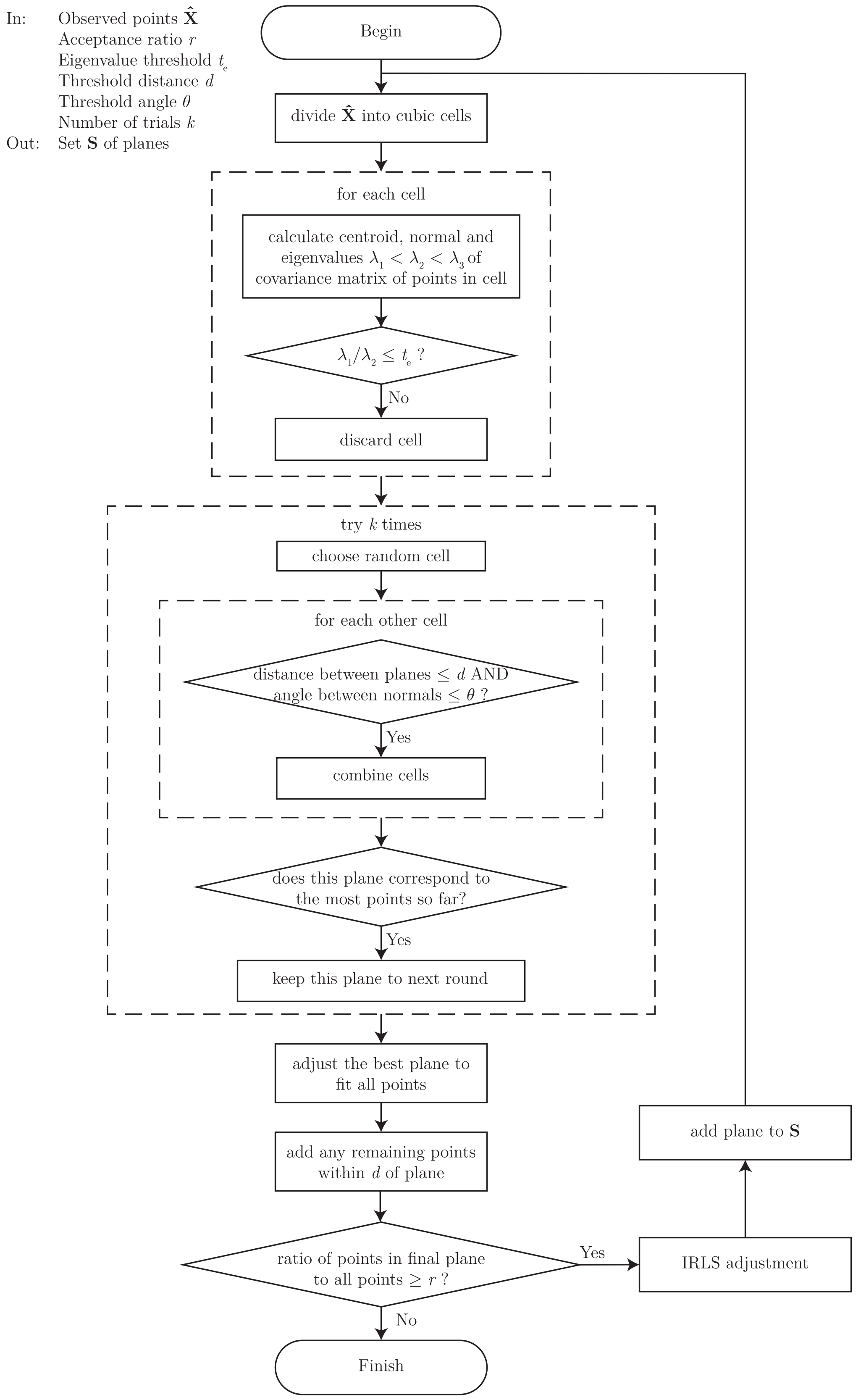

In: Observed points $\hat{\mathbf{X}}$
Acceptance ratio r
Eigenvalue threshold t_e
Threshold distance d
Threshold angle θ
Number of trials k
Out: Set $\mathbf{S}$ of planes
Begin
divide $\hat{\mathbf{X}}$ into cubic cells
for each cell
calculate centroid, normal and eigenvalues $\lambda_1 < \lambda_2 < \lambda_3$ of covariance matrix of points in cell
$\lambda_1/\lambda_2 \leq t_e$?
No
discard cell
try k times
choose random cell
for each other cell
distance between planes $\leq d$ AND angle between normals $\leq \theta$?
Yes
combine cells
does this plane correspond to the most points so far?
Yes
keep this plane to next round
adjust the best plane to fit all points
add any remaining points within d of plane
ratio of points in final plane to all points $\geq r$?
Yes
IRLS adjustment
add plane to $\mathbf{S}$
No
Finish

Figure 4.3. Plane detection using NDT RANSAC as proposed by Li et al. (2017). Available in pseudocode in Algorithms 2 and 3 in Appendix B.2.

The plane detection algorithm implemented here is based on a RANSAC method proposed by Li et al. (2017), who divide the point cloud into cubic cells, use a normal distribution transform (NDT) to find planar cells and use random sampling across these cells. The method then matches the sampled planar cell with cells with similar plane parameters, adds other points on the plane and refines the plane using an iterative reweighted least-squares (IRLS) procedure. The algorithm is described in Figure 4.3. The IRLS adjustment step is described in pseudocode in Algorithm 2 in Appendix 7. Compared to a more naïve RANSAC approach, this method has less emphasis on random sampling and for that reason produces more consistent results on different executions.

In this implementation, $2 \times 2 \times 2$ metre cubic cells were used due to the large sizes of the room and the planar features in it. As suggested by Li et al. (2017), t_e is given a value of 0.04. Threshold distance d was set to 0.04 m and threshold angle θ to 15°. The initial number of trials k was 100, but this was redefined every time a new best plane was found using the equation

$$k \geq \frac{\ln(1 - 0.99)}{\ln(1 - \frac{n_{this}}{n_{total}})} \tag{4.1}$$

where n_{this} and n_{total} are the number of points in the current best plane and in total, respectively. 0.99 represents the confidence level that is desired for the plane. The redefinition of k allows for the flexible determination of the number of trials, so if a very good plane is found, fewer iterations are tried, and if the best plane identified is still poor, more iterations are added. Once the planes have been detected, they can be used for calibration by plane fitting as described in Figure 4.4.

Here, plane fitting refers to the iteration during which parameters are adjusted in order to move points into positions that best fit the planes identified earlier. The actual fitting is performed using standard algebraic methods, namely by calculating the eigenvalues and eigenvectors of the covariance matrix of the coordinates of the points corresponding to a plane. The symmetric positive definite 3×3 covariance matrix describes the distribution of the data in 3 dimensions so that the eigenvalues define magnitude and the eigenvectors define orthogonal directions. Therefore, the smallest eigenvalue describes variance in the direction of the corresponding eigenvector, which is normal to the plane defined by the other 2 eigenvectors, which is the best fitting plane. This eigenvalue can be considered to quantify the noise in the plane, the sum of which over all planes is what constitutes the *errorsum* in the algorithm.

4.3.2 Local planarity

Levinson and Thrun (2014) propose a calibration scheme that is specially designed for multi-beam scenarios, where points observed by a single beam are

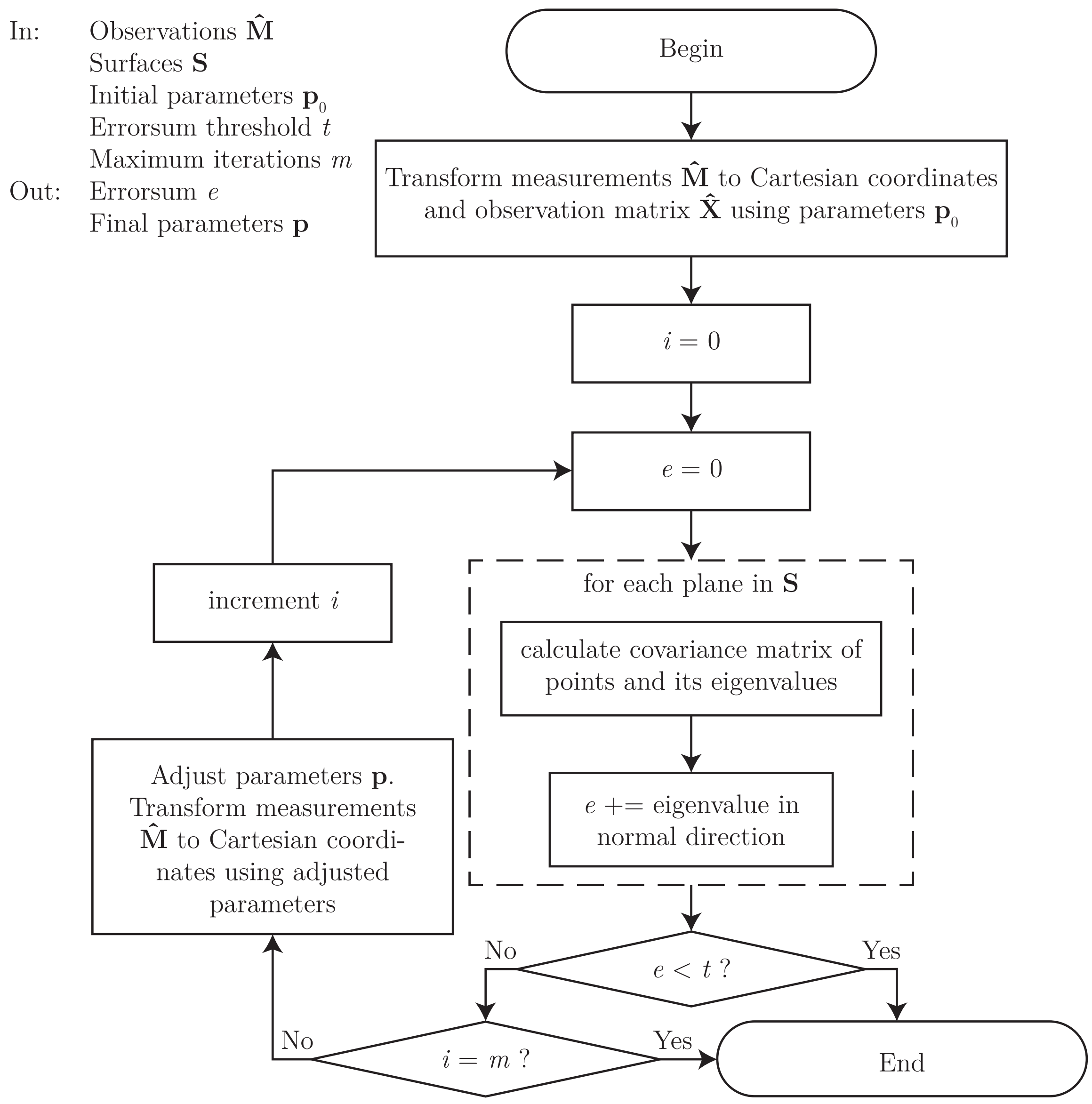

In: Observations $\hat{\mathbf{M}}$
Surfaces $\mathbf{S}$
Initial parameters $\mathbf{p}_0$
Errorsum threshold t
Maximum iterations m
Out: Errorsum e
Final parameters $\mathbf{p}$
Begin
Transform measurements $\hat{\mathbf{M}}$ to Cartesian coordinates and observation matrix $\hat{\mathbf{X}}$ using parameters $\mathbf{p}_0$
$i = 0$
$e = 0$
for each plane in $\mathbf{S}$
calculate covariance matrix of points and its eigenvalues
e += eigenvalue in normal direction
increment i
Adjust parameters $\mathbf{p}$. Transform measurements $\hat{\mathbf{M}}$ to Cartesian coordinates using adjusted parameters
No
$e < t$?
Yes
No
$i = m$?
Yes
End

Figure 4.4. Calibration parameter optimisation using previously defined planes. Available in pseudocode in Algorithm 4 in Appendix B.3.

compared with their neighbours from other beams in an attempt to have the observations from different planes form contiguous surfaces. The suggested method optimises the planarity of all such neighbourhoods, which in practice translates to smoothing all continuous surfaces in the point cloud. This, of course, assumes that the point cloud (or rather, the environment) is somewhat continuous and locally planar.

The method is described in Figure 4.5. It iterates through all beams and for each beam, all points observed by the beam. For each point, it finds the nearest k neighbours observed by adjacent beams and fits a plane to these points. Finally, it sums the squared orthogonal distances from the plane to each of the points in

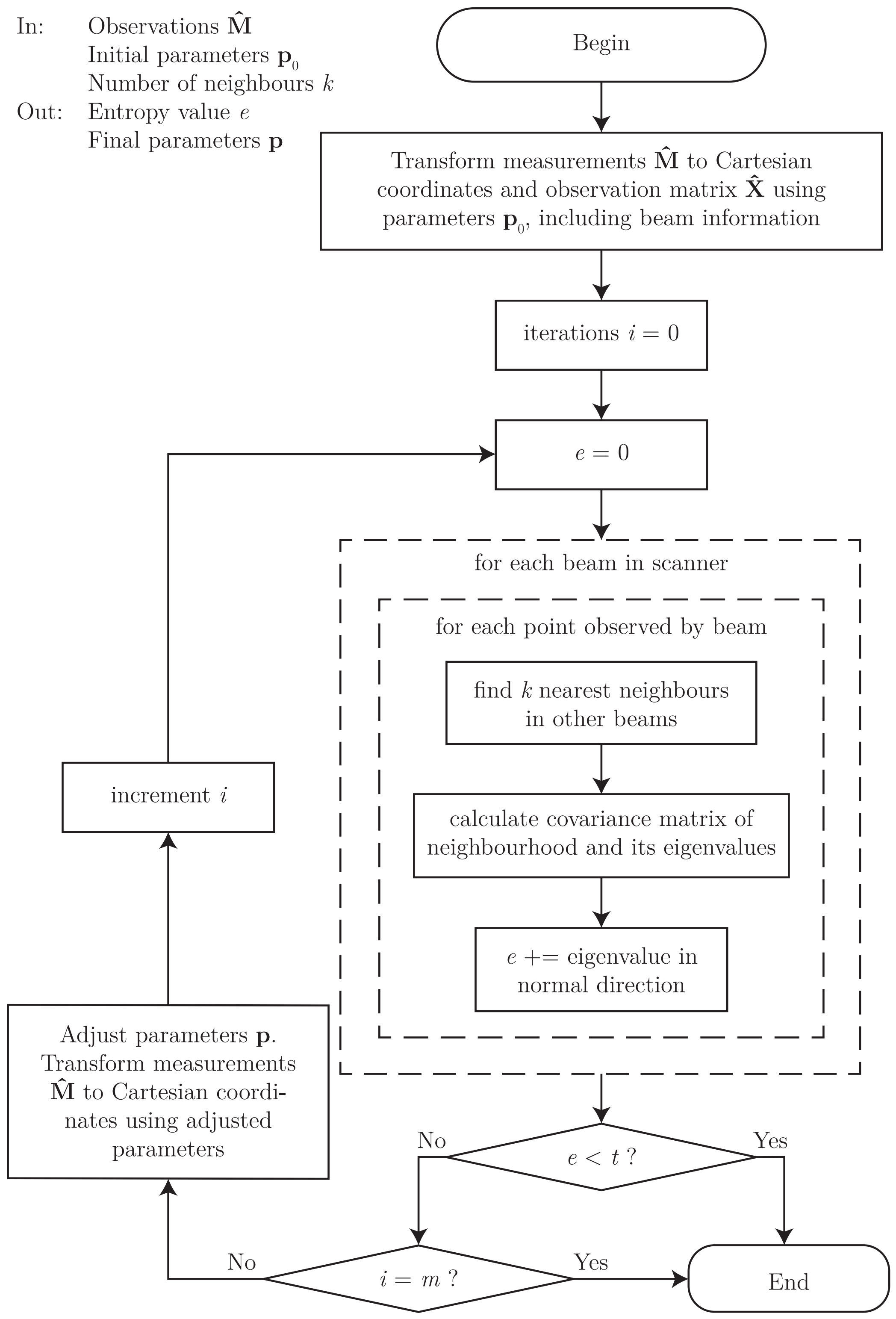

In: Observations $\hat{\mathbf{M}}$
Initial parameters $\mathbf{p}_0$
Number of neighbours k
Out: Entropy value e
Final parameters $\mathbf{p}$
Begin
Transform measurements $\hat{\mathbf{M}}$ to Cartesian coordinates and observation matrix $\hat{\mathbf{X}}$ using parameters $\mathbf{p}_0$, including beam information
iterations $i = 0$
$e = 0$
for each beam in scanner
for each point observed by beam
find k nearest neighbours in other beams
calculate covariance matrix of neighbourhood and its eigenvalues
e += eigenvalue in normal direction
$e < t$?
No
Yes
$i = m$?
No
Yes
End
Adjust parameters $\mathbf{p}$. Transform measurements $\hat{\mathbf{M}}$ to Cartesian coordinates using adjusted parameters
increment i

Figure 4.5. Calibration parameter optimisation as described by Levinson and Thrun (2014), using local planarity across neighbouring beams. Available in pseudocode in Algorithm 5 in Appendix B.4.

the neighbourhood to produce an errorsum, which is further summed over the rest of the iterations.

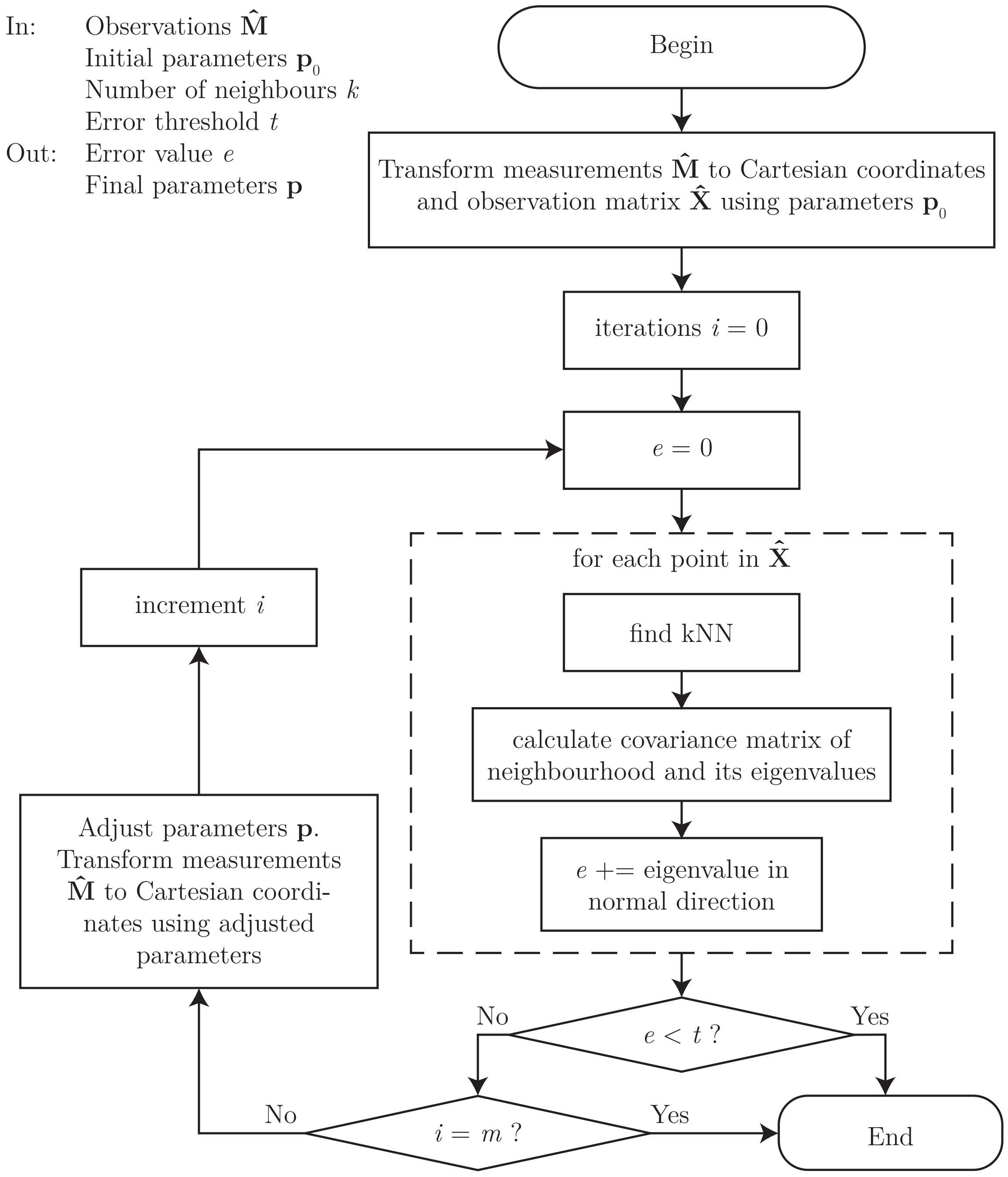
Evidently, the methodology contains much iterating, and when programmed—as can be seen in Figure 4.5—multiple nested for loops. Additionally, there are many nearest neighbour searches over various sets of points (each beam is ignored in turn), which makes the algorithm computationally complex. This is caused not just by the many searches, but also the construction of the data structures that enable them.

A similar optimisation procedure can be implemented without consideration for separate beams, but the relatively sparse vertical distribution of points (that is, the layered nature of the cloud) means that any nearest neighbour search will favour points from the same beam over points from adjacent beams. This results in fitting a plane to what is, analogously, a line. This depends, of course, on the scanner being used and the actuation of the scanner. Levinson and Thrun (2014) use a Velodyne HD-64E with 64 beams that is rigidly attached to a moving vehicle. However, the use of a RMBL in this thesis results in a more evenly distributed point cloud—as seen in Figure 3.6—which disqualifies the beam-wise method and allows the use of this simplified version. The simplified version is presented in Figure 4.6. In this implementation, $k = 40$ was chosen as the number of neighbours to consider, since this produced the best calibration results, i.e. the best score as described in Section 5. The results leading to this choice are presented in Section 5.3.

4.3.3 Entropy

Equation (2.26) described the cost function of entropy optimisation. Algorithmically, for optimisation purposes, this translates to **(A)** in Figure 4.7. Notably, the algorithm contains a nested for loop that iterates through all points within an iteration through all points. This slows the algorithm to running in $O(n^2)$ time, which is typically too much when dealing with hundreds of thousands or millions of points. Complexity can be halved by having the second loop only iterate through points that come *after* it (in the matrix). This is because i and j are interchangeable in terms of the value of $\exp\big(-\frac{1}{\sigma^2}(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)^T(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)\big)$. This has the added benefit of skipping the calculation for a point itself, for which the equation becomes a constant $\exp(0) = 1$. The reduced complexity achieved is hardly adequate, however.

To hasten the computation further, we can choose to only care about the nearest neighbours of each point. In practice, when the distance between $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ grows, $e_{i,j} \to 0$. The pace at which this convergence to 0 happens depends on the parameter σ^2. The value chosen for σ is small (some centimetres, 5 in this implementation, giving $\sigma^2 = 0.05^2\,\mathrm{m}^2 = 0.0025\,\mathrm{m}^2$) relative to the size of the point cloud, which justifies limiting the entropy computation to some nearest neighbours of each point. The limitation can either be enforced using a k Nearest Neighbours



In: Observations $\hat{\mathbf{M}}$
Initial parameters $\mathbf{p}_0$
Number of neighbours k
Error threshold t
Out: Error value e
Final parameters $\mathbf{p}$
Begin
Transform measurements $\hat{\mathbf{M}}$ to Cartesian coordinates and observation matrix $\hat{\mathbf{X}}$ using parameters $\mathbf{p}_0$
iterations $i = 0$
$e = 0$
for each point in $\hat{\mathbf{X}}$
find kNN
calculate covariance matrix of neighbourhood and its eigenvalues
e += eigenvalue in normal direction
$e < t$?
No
Yes
$i = m$?
No
Yes
End
Adjust parameters $\mathbf{p}$. Transform measurements $\hat{\mathbf{M}}$ to Cartesian coordinates using adjusted parameters
increment i

Figure 4.6. Calibration parameter optimisation using general local planarity. Available in pseudocode in Algorithm 6 in Appendix B.4.

(kNN) search or a radius search. Considering that $\lim_{(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j) \to \infty} e = 0$—in practice, once the distance between points is 3σ, for example[1], e can be considered negligibly small—radius search can give us an intuitive cut-off. However, radius search

[1] 3σ is justified by the 3-sigma rule, whereby in a normal distribution, the probability for a point to lie further than three standard deviations from the mean is less than 3 in 1,000 (Upton & Cook, 2008). This does not directly apply to multivariate distributions, but provides a large enough margin for the operation to be reasonable.

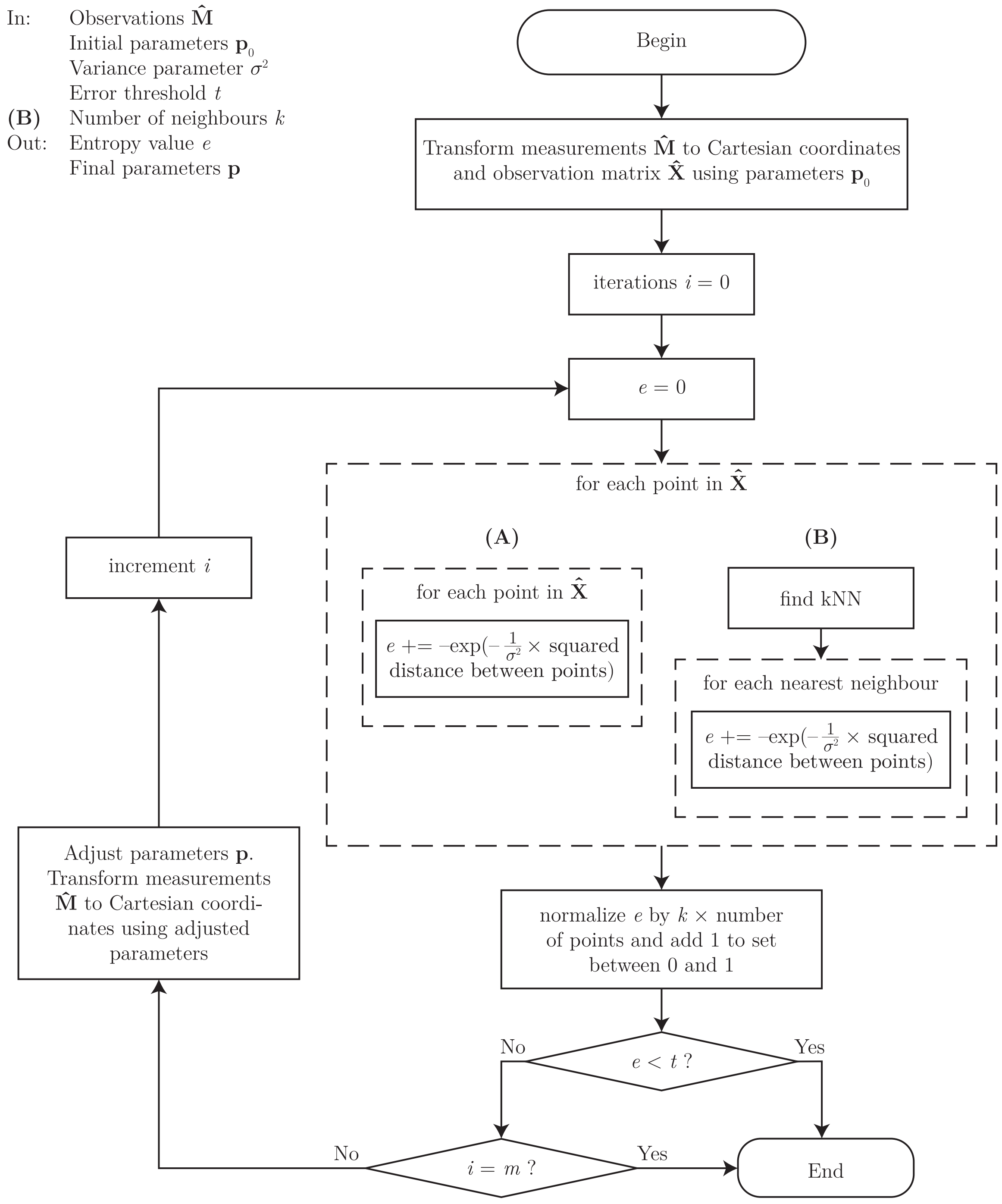

In: Observations $\hat{\mathbf{M}}$
Initial parameters $\mathbf{p}_0$
Variance parameter σ^2
Error threshold t
(B) Number of neighbours k
Out: Entropy value e
Final parameters $\mathbf{p}$
Begin
Transform measurements $\hat{\mathbf{M}}$ to Cartesian coordinates and observation matrix $\hat{\mathbf{X}}$ using parameters $\mathbf{p}_0$
iterations $i = 0$
$e = 0$
for each point in $\hat{\mathbf{X}}$
(A)
(B)
increment i
for each point in $\hat{\mathbf{X}}$
e += −exp($-\frac{1}{\sigma^2}$ × squared distance between points)
find kNN
for each nearest neighbour
e += −exp($-\frac{1}{\sigma^2}$ × squared distance between points)
Adjust parameters $\mathbf{p}$. Transform measurements $\hat{\mathbf{M}}$ to Cartesian coordinates using adjusted parameters
normalize e by k × number of points and add 1 to set between 0 and 1
No
$e < t$?
Yes
No
$i = m$?
Yes
End

*Figure 4.7. Calibration parameter optimisation using entropy. **(A)** describes the entropy cost function without any optimisation while **(B)** describes the algorithm using kNN to decrease computational cost. Available in pseudocode in Algorithms 7 and 8 in Appendix B.5.*

does emphasise areas with high point density. To eliminate this effect, we can normalise the entropy value for a particular neighbourhood by the number of points found within the radius.

In practice, as explained in Section 6.1.3, radius search does not work, so a kNN approach is implemented. **(B)** in Figure 4.7 describes the optimisation algorithm using kNN search. This brings computational cost down to $O(n(k + \log n))$ when using kd-tree data structures for searching. This algorithm is the one used for calibration in this thesis.

4.4 Testing

The main contribution of this thesis is the quantitative testing of the calibration methods presented above. The calibration results are evaluated by scoring a set of observations from each sensor using the resulting calibration parameters of each method. The scoring function is the same as the cost function used in target-based calibration—that is, a sum of square distances to the nearest points in the reference point cloud. A lower score signifies smaller error and thus a better calibration according to this metric. The scoring method and the individuality of the scanners and the data samples mean that the scores are not comparable between different scanners. For each device, a data sample is selected, and the same sample transformed into a point cloud using calibrated parameters for each method. This sample is different from the one used for the calibrations. To avoid any displacement between the origins of the calibration and reference clouds, a 6-DOF transformation is found for the reference cloud so that the score is minimal. This is a similar minimisation process as the target-based calibration algorithm, except that the parameters being optimised are the six transformation parameters and the calibration parameters are kept constant. The score is then computed by summing the distances to nearest neighbours in the reference point cloud. Some examples of calibration parameters before and after calibration are also presented, and all parameters are available in Appendix A.

The tests provide material for further examination of the methods as well. First, the choice of $k = 40$ for the local planarity method is justified with experimental results (for the entropy method k was set at 30). Second, convergence patterns for the various algorithms are examined by recording the evolution of calibration parameters over the iterations and scoring the parameters at different stages of the optimisation process. Finally, the computational complexities of the methods and the variation between them are discussed.

5. Results

5.1 Evaluation of methods

The results of calibrations were evaluated by scoring a set of observations from each sensor using the resulting calibration parameters of each method as explained above in Section 4.4.

	Default	**Target-based**	**Plane fitting**	**Local planarity**	**Entropy**
VLP-16 Lite	4704.42	1765.26	1876.32	3090.02	1791.15
VLP-16 (1)	3159.07	1642.21	2826.27	3447.01	2851.45
VLP-16 (2)	3158.97	1440.51	2925.03	3588.36	2951.92
VLP-16 (3)	6310.82	1760.01	4184.16	2974.59	2740.63
VLP-16 (4)	4716.38	1729.12	1749.19	2641.04	1752.44
VLP-32C	4860.14	3036.42	3175.96	3581.12	3011.05
Sum	26 910	11 374	16 737	19 322	15 099

Table 5.1. Scores obtained by different calibration methods for different sensors. The score values are not comparable between sensors.

Table 5.1 summarises the results of the calibration method comparison. Evidently, the target-based method fares best, as its cost function is equal to the scoring function. It is also the only absolute method in that it has a specifically defined target. Of the three data-based methods, the plane fitting and entropy methods achieve the best scores, and the local planarity method the worst. Notably, the scores achieved by plane fitting vary significantly, indicating that the detected planes in different cases were of uneven quality. This indicates unreliability even in circumstances where the environment was chosen to favour plane segmentation. At the same time, the entropy method was rather consistent, suggesting robustness, while the local planarity method seems occasionally decent, but in some cases actually returns worse results than default values and is thus of little compared to the others. The methods and their properties are discussed further in Section 6.1.

5.2 Calibration results

All calibration parameters and their values for all sensors before calibration and after calibration by each method are given in Appendix A. In this section, some notable results and observations are presented. Focus is directed to intrinsic calibration parameters, since values acquired for extrinsic calibration parameters are not particularly interesting, as they only reflect the precision with which the RMBL system was constructed. Additionally, the observations are based on parameters calibrated using the target-based method, since this is considered to provide the most accurate results. A visualisation of calibrated intrinsic parameters for each method for VLP-16 (3) can be seen in Figure 5.1. It can be contrasted with the default configuration in Figure 3.3.

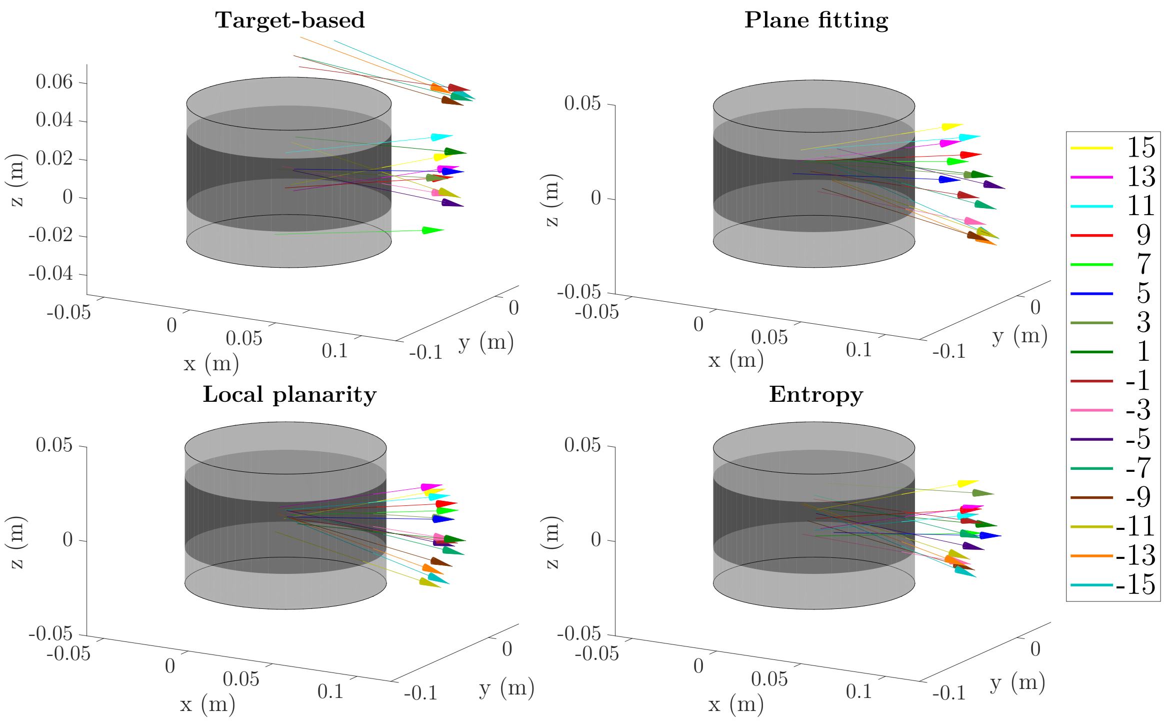


Figure 5.1. Calibrated internal parameters of VLP-16 (3) sensor visualised with short (10 cm) arrows emitted from the optical centre each beam in the directions of the laser beams. The legend indicates the default elevation angle of each beam.

Most intrinsic calibration parameters remain more or less constant through optimisation. Values for x and y offsets are small, as are offsets in azimuth angle. There is more variation in elevation angle, and especially notable is the fact that in the VLP-16 Lite and one of the VLP-16 sensors, the elevation angles seem to be systematically off from the default value by approximately 2°, an observation made also by Hyyti et al. (2019). Significant differences to the default can be seen elsewhere as well. Finally, there is notable variance in z offset values, and though sometimes these parameters get values that are physically impossible (i.e. outside the frame of the sensor), they are suggestive of the possibility that the beam origins are on a vertical axis inside the sensor, slightly displaced from one another. In addition, it is possible that there are non-mechanical offsets in the measurements.

As can be seen in Figure 5.1, the z offset was largest in the target-based calibration method, which is supposed to be the most accurate of the methods. It is indicative of the vertical offset between the RBML and TLS point clouds. Because the transformation between the coordinate frames was found in the same optimisation procedure for target-based calibration—as explained in Section 4.2—this could be a result of the offset between the point cloud origins manifesting in the intrinsic parameters, which is, of course, not a desirable result. If this is the case, the transformation between target and calibration point clouds should be separated from the calibration, even though this, too, was found intuitively problematic, though not tested.

5.3 Local planarity

There was a large dependence on the number of nearest neighbours k used to evaluate local planarity. An experimentation with $k = \{10, 20, 30, 40, 50\}$ determined the use of $k = 40$, since it yielded the best score. Table 5.2 shows the comparative scores. The comparison was made with the data from the Puck LITE sensor.

k **value**	**Score**
10	4633.50
20	4241.98
30	3090.02
40	2696.63
50	2851.39

Table 5.2. Scores for different k values in local planarity method for VLP-16 Lite. 40 nearest neighbours was chosen.

Even when choosing a more optimal number of neighbours for comparison, the local planarity method failed to achieve scores on par with plane fitting or entropy. Nonetheless, it may be noteworthy to note that the optimal value for k likely depends on the environment in which the sensor is being calibrated. The environment used in this thesis contained large planar surfaces, so even with relatively small point density $k = 40$ was justifiable. In environments with less planarity or with more sparse point clouds, the optimal value may be lower, and vice versa. However, in non-planar environments, this method is likely to fare even worse, though it does not seem recommendable even in planar environments.

5.4 Convergence

The different calibration methods behaved differently in optimisation, as could be expected. Figure 5.2 collects some convergence plots to illustrate this. As can be seen, the local planarity method is the fastest to converge. Reasons for this are unclear, as there do not seem to be any particular properties of the method that lead to fast convergence. In fact, the method is in many ways similar to the plane fitting method, only that the number of planes is equal to the number of points. It is also noteworthy in the figure that the entropy method benefits most obviously from the large number of iterations and restarted simplexes, achieving its largest descent only in the fourth optimisation round.

It may be possible to use these convergence properties in future calibration implementations. If one method is quick to find some minimum and improve the

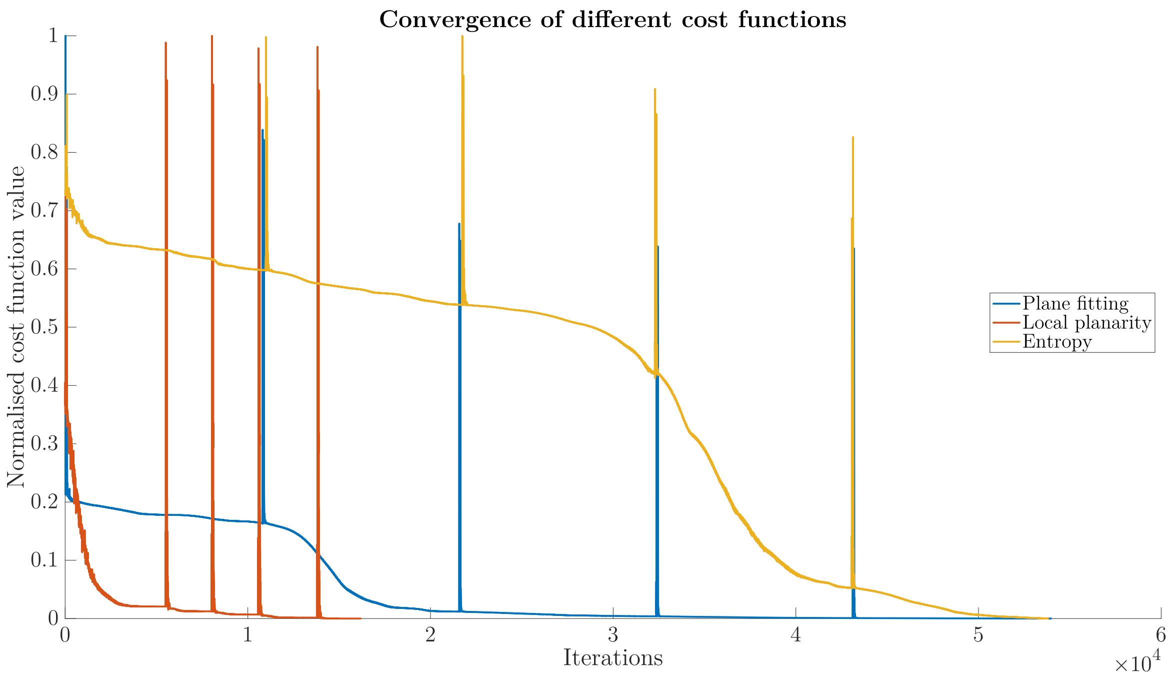


Figure 5.2. Convergence plot of different calibration methods for one scanner, VLP-16 (4). The normalised cost function value on the y-axis is set to be in range [0,1].

calibration parameters to some extent, it can be used to jump-start the calibration process before switching to a more precise method, which may be slower overall but might need less iterations after such a jump-start. Any potential here would need to be investigated further in future research.

Similarly, we can examine the progress of the calibration methods using the scoring function, as can be seen in Figure 5.3. It should be noted that the methods are scored against the reference point cloud fitted to the final iteration of the entropy calibration. The final score values thus do not correspond to those in Table 5.1 for the plane fitting and local planarity methods. The figure largely validates the use of both the plane fitting and entropy methods while invalidating the local planarity method, which increases the score value instead of decreasing it. At the same time, the local planarity method does achieve better scores than default values, which seems to contradict this observation. The reasons for this remain unclear, and the optimisation behaviour would need further investigation for full understanding.

5.5 Computational complexity

As is often the case with calculations involving large point clouds, it is necessary to consider the computational costs of calibration algorithms. When trying to find a calibration method that can perform calibration independently of any targets and preferably as a routine procedure e.g. before measurements, speed is a key attribute of the method. This section provides some insight into the

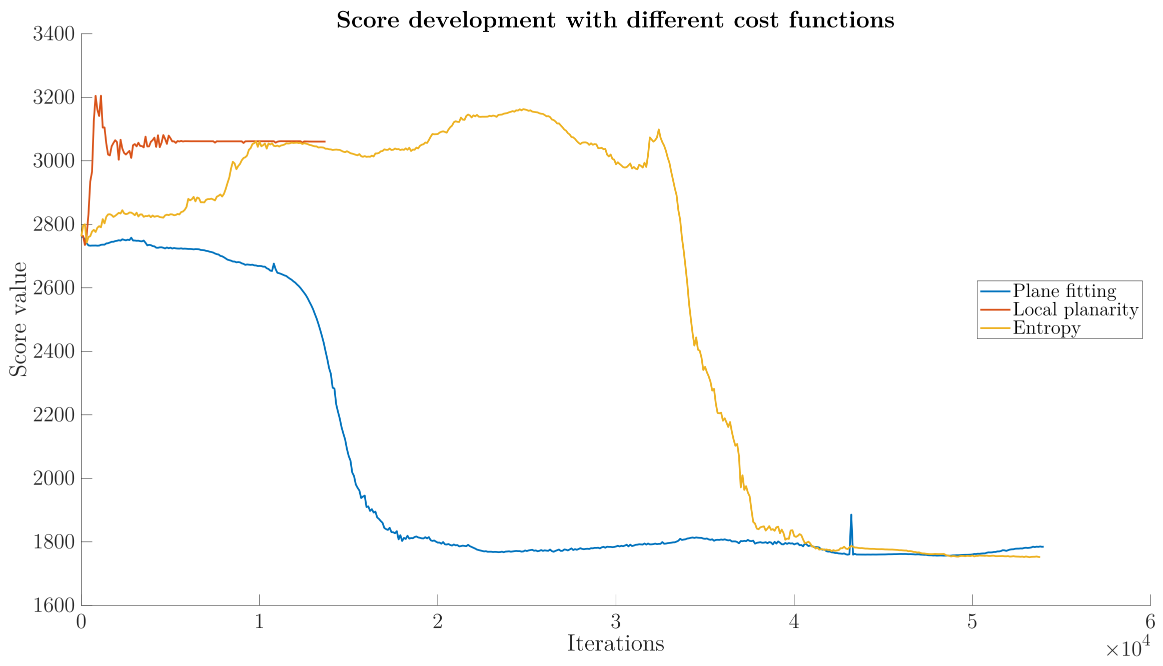


Figure 5.3. Plot showing the development of the score through the iteration of different cost functions. Evaluated for scanner VLP-16 (4).

computational costs of the proposed methodologies, with relevant details about the implementations to shed light on the reasons behind the results. In all cases it is the iterative optimisation that consumes the vast majority of computation time due to its iterative nature and the large number of parameters being optimised. This thesis used five consecutive MATLAB *fminsearch* optimisation routines with 10,000 iterations each, a total of 50,000 iterations for 16-beam sensors and 15,000 iterations each (75,000 total) for the 32-beam sensor. These were separate because of the way the Nelder-Mead algorithm works, with a reinitialisation providing the possibility of finding a smaller minimum. This was, in practice, found to result in relatively stable minima, though most of the functions did not converge to a final value even after these iterations.

Method	**Complexity**
Target-based	$O(n \log n_T)$
Plane fitting	$O(N_P(3^2 n_P))$
Local planarity	$O(n \log n + n(\log n + 3^2 k))$
Entropy	$O(n \log n + n(k + \log n))$

Table 5.3. Approximate computational complexity of cost functions. N_P is the number of planes being fitted, n is the number of points in the cloud, n_P signifying points in a plane and n_T points in the target cloud, and k is the number of nearest neighbours considered in the algorithm.

Table 5.3 presents a comparison of complexities of the cost functions in a straightforward manner. However, the actual time taken for the computations can not be directly deduced from these. In any case, plane fitting is by far the fastest of the calibration methods. Target-based calibration is next, since the search structure for nearest neighbours remains the same in each iteration. Finally, the entropy method is slightly faster than the one based on local planarity. However, the local planarity method tended to converge much faster than the entropy method in practice.

Many of the calibration algorithms involve matrix manipulations, the complexities of which will be briefly introduced here. The multiplication of two matrices of size $n \times m$ and $m \times p$, when performed in the naïve way has a complexity of $O(nmp)$. Le Gall (2014) shows that the complexity of square matrix multiplication can be reduced from the brute force $O(n^3)$ for 2 $n \times n$ matrices to some $O(n^{2.373})$ and later Le Gall and Urrutia (2018) reduce the complexity of multiplying rectangular matrices ($n \times n^k$ by $n^k \times n$) to $O(n^{\omega(k)+\epsilon})$, where $\epsilon > 0$. Exploring what these mean in the context of point clouds is not in the scope of this work. In any case, the software and libraries that were used—MATLAB (MATLAB, 2019) and Eigen (Guennebaud et al., 2010)—optimise these calculations. However, in the case of MATLAB, information on the way in which this is done is not available.

In addition, the eigenvalue decomposition is also often used and has a complexity of approximately $O(n^3)$ (see e.g. Pan & Chen, 1999, for a more comprehensive analysis). In this thesis, all eigenvalue decompositions are performed on 3×3 covariance matrices, which means that eigenvalue decompositions run in constant time are thus omitted from complexity expressions. When computing these covariance matrices a scaling operation (division by the number of points) is performed, and this has a complexity of $O(nm)$ for a matrix of size $n \times m$. Again, this is in practice a constant $O(3^2)$, which is likewise omitted.

Finally, complexities relating to kd-trees are based on the Wikipedia article on kd-trees (*k-d tree*, 2019).

5.5.1 Target-based calibration

Target-based calibration is a relatively simple procedure, taking some $O(n \log n_T)$ time as it iterates through all points and finds a nearest neighbour for each in the target point cloud. Additionally, some time ($O(n \log n_T)$ is required to build the kd-tree out of the target point cloud, but this can be done only once, while the cost function is evaluated thousands of times. For efficiency reasons, the number of points in the reference cloud was reduced from the original 450 million to approximately 4.5 million, which was still considered plenty, as it was significantly larger than the number of points in the calibration cloud. This decimation by random sampling should not affect the calibration results, since, for each calibration point, a point can be found rather nearby, so there is no—or at least very little—matching between points that do not represent the same physical vicinity.

5.5.2 Plane detection and fitting

Table 5.3 does not include time taken for plane detection. The NDT RANSAC method proposed by Li et al. (2017) has many steps, many of which involve multiplications and decompositions of matrices of varying size. Furthermore, random sampling and iterative nature of the algorithm mean that an exact expression of complexity is not possible to achieve. The environment also makes a large difference. However, plane detection is carried out only once, so it is not a significant factor compared to the optimisation. In practice, it was found that detecting 4-9 planes in point clouds of 100,000–2,000,000 points took approximately 1–90 seconds on an Intel(R) Core(TM) i7-7500U 2.70 GHz CPU. Relative to the time taken to calculate thousands of iterations of any of the cost functions, this is negligible.

Plane fitting is computed in $O(N_p)$ time, with N_p signifying the number of planes. However, for each plane, the eigenvalue decomposition of the covariance matrix is calculated, which has a complexity of approximately $O(3^2n)$, where n is the number of points in the plane and 3 the dimensionality of the point cloud. This comes from matrix multiplication, scaling and the eigenvalue decomposition as explained above. In practice, the optimisation using detected planes (4–8 in number) took approximately 1–6 hours when run in MATLAB on the above machine, depending also on the number of points.

5.5.3 Local planarity

The original calibration prodecure proposed by Levinson and Thrun (2014) is computationally more complex than the one used here. The original, with its consideration for points seen be neighbouring beams, would have a complexity of approximately $O(bN\frac{n}{b}(\log(\frac{n}{b}) + 3^2k))$, where b signifies the number of beams, N the number of neighbouring beams considered per beam, k the number of neighbours considered per point and n the number of total points.

The implementation used in this thesis is simpler, though only by the removal of the factor N and the denominator in the logarithm, giving $O(n(\log n + 3^2k))$. This is a consequence of not considering the beam by which each point was observerd. In practice, an optimisation round of $5 \times 10,000$ iterations with 85 calibration parameters and a point cloud of approximately 200,000 points took approximately 6 hours to converge (only 11,000–15,000 iterations were typically necessary) on the aforementioned computer when cost function evaluation was implemented in C++ as a MATLAB MEX file function.

5.5.4 Entropy-based calibration

In theory, the entropy-based method is by far the most expensive computationally, taking $O(n^2)$ time. However, it considers all the points that it is fed and can be considered more robust. Computation times can also be vastly reduced with various optimisation procedures, such as limiting the calculation of entropy for each point to only the nearest points. The rationale for this is that points that are further away have an exponentially smaller effect on the outcome of the cost function than points near to the current point. This does introduce an additional tuning parameter k for kNN search, which can be considered problematic. Even so, this reduces computational cost to $O(n(k + \log n))$ time, where k is the number of neighbours found, when optimising the search with a kd-tree structure. In the case of entropy, there is no need for matrix multiplication or decompositions.

In practice, the time taken by a single evaluation of the cost function was approximately the same as with the local planarity method, but the entropy method never fully converged, so $5 \times 10,000$ iterations took about 24 hours to complete. The entropy method was also implemented as a MATLAB MEX function.

6. Discussion

6.1 Strengths and weaknesses of approaches

As seen in Section 5, the entropy method obtained the best scores with plane fitting rather close behind while local planarity was found, in some cases, to worsen the calibration. Alongside these quantitative results, some other properties of the various calibration methods were observed. Computational complexity is discussed below.

6.1.1 Feature-based calibration

Feature-detection-based calibration was found to be sensitive to the feature detection process. With normal RANSAC plane segmentation, the number of iterations and chance can have a large effect on the quality of the detected features. However, the NDT RANSAC algorithm used here produced rather consistent results. However, the algorithm may misidentify features as planes or rather include outliers or disclude inliers, depending on the parameters and the error in the point cloud. In some cases, the uncalibrated point clouds were so noisy that the algorithm did not recognise the floor as a single surface. Figure 6.1 shows the profiles of two such point cloud floors. In (A), the points representing the floor can be seen to curve, though this is not a real property of the floor (in fact, a hill-like shape is formed where the top is under the origin). A plane fitted to these points would not be parallel to a plane of the floor. In (B), there are two unparallel planes composing the floor, though this, too, in three dimensions is composed of the same shape as in (A) and its reflection. This is likely to be a result of the rotating motion of the RMBL, where each area is scanned twice during one rotation (the points in both clouds are from a single rotation). In fact, calibration results show that in some of the VLP-16 sensors—namely, the VLP-16 Lite and VLP-16 (3)—the elevation angle values are systematically approximately 2° off.

On a more general level, feature-based calibration is intuitive and can be monitored reliably. It is possible to examine the detected features and the corresponding point sets to determine whether the matching is satisfactory. It is also possible to tune plane detection parameters for different scans and even for different areas of the same scan. The features can also be manipulated manually.

A crucial factor in the success of feature-based calibration is the detection of planes that are not adjacent. In the case of scanning a large gym as in this thesis, it is imperative that the algorithm detects at least one wall and the floor. If the algorithm only identifies the ceiling and floor, for example, the optimisation procedure will attempt to flatten all measurements, for example by reducing the elevation angles of different lasers to the same angle. As it is, the algorithm only evaluates the planes with regard to the points in the plane, not the original plane that was identified. This is a feature of the way the cost function is calculated, only considering the smallest eigenvalue of the covariance matrix of the points in the plane. To keep the location of the planes constant (and only minimise the error to this stationary plane), we could change the cost function to, for example, calculate square distance from the plane to each point assigned to the plane. However, using the initial planes as references is also problematic, because it assumes that the definitions of the features do not improve, or in practice, that the normal does not shift at all. As Figure 6.1 shows, the original plane (and its normal) can be far from ideal.

In addition to errors such as this, plane detection results vary in quality. The plane detection algorithms implemented here favours large planes and includes as many points in a plane as possible without consideration for adjacency. At the same time, the large amount of noise (± 3 cm) in the Velodyne sensors means that a rather low threshold must be maintained for including points in a detected plane. This, in turn, means that a number of points included in a plane are almost certainly not, in reality, located on that plane, returning instead from a different wall on the same corner, for example. Surfaces with nonplanar details, such as a wall covered in a tight pattern of small rectangular columns, is likely to classified as planar since the variation of the pattern in the normal direction is small. For feature detection purposes, such misclassifications may be rather inconsequential, but an accurate feature-based calibration requires high confidence in features. As it is, the plane fitting method does provide improved calibration, but does not perform as consistently as alternative approaches.

However, the results also suggest that detected planes do not need to be perfectly planar. For example, one of the walls of the gym was partially covered in the aforementioned pattern of rectangular columns (see Figure 6.2). This happened consistently, as the pattern only varies by a couple centimetres, while the error in Velodyne sensors is ± 3 cm. Other similar details were also present in the environment. Even so, the plane fitting method was able to find good calibration

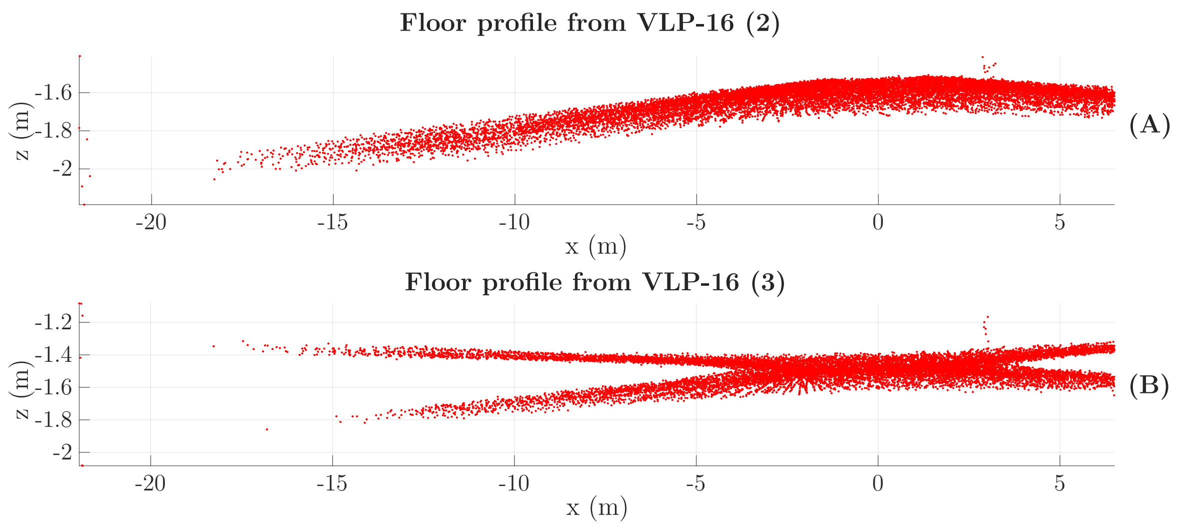


Figure 6.1. Profiles of gym floor from two non-calibrated scans. The scanner is in the origin.

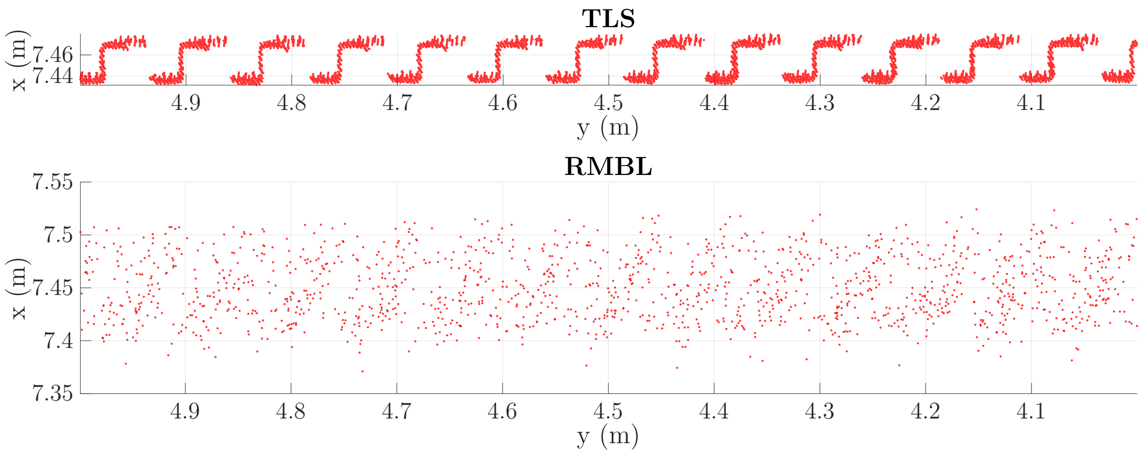


Figure 6.2. Views from above of wall detail with rectangular columns from. The corresponding area in RMBL clouds was identified as planar by the plane detection algorithm.

parameters in most cases. This is likely because even if an identified plane is not a uniform and single plane, it is mostly planar in that it has much larger variances in the two principal directions than in the normal direction. Since reducing errors in the cloud is likely to reduce noise in all directions, the variance in the normal direction should also decrease with improved calibration. Therefore, even though the features do not truly represent the environment, optimising them represents a realistic minimum in the optimisation space.

6.1.2 Local planarity

As was mentioned in Section 2.4.5, the nature of the RMBL setup used in this thesis is not suitable for a beam-wise calibration approach. The continuous rotation of the platform and the sensor allows the collection of evenly distributed point clouds, where observations from different lasers are mixed. This is in itself a good reason to use such a rotation mechanism for actuation, since it improves the coverage of the measurements. This fact was used to generalise the method presented by Levinson and Thrun (2014) to a universal measure of local planarity. However, as seen in Section 5, the method was not particularly successful in our tests.

In particular, it is interesting that the local planarity method converges quickly, but does not find the correct parameters. This suggests that there is something fundamentally wrong with the cost function as an evaluation of point cloud quality—a fact that makes sense considering the nonplanar nature of most environments. However, the environment used in this thesis contains a large amount of planar surfaces, and reasons for the failure of this method can be found perhaps in the large amount of error in the measurements. In cases such as the divergent floors of Figure 6.1, this method may not encourage the unification of these areas as k nearest neighbours only returns points on the same plane, and moving towards the merging of the floors would increase the value of the cost function in the short term before decreasing it. In other words, distances between local minima in the optimisation space might be too large. Calculating the local planarity cost function value for a well-calibrated version of the same point cloud indeed yields a lower score—for example, for VLP-16 (4), the cost function value of the scoring data set using local planarity parameters is 1519.5, while using entropy parameters it is 1223.9, which is significantly lower. The local planarity cost function thus seems to have converged on a local minimum.

6.1.3 Entropy

With entropy-based calculation, Section 4.3.3 proposed limiting computational complexity by narrowing the search down to k nearest neighbours or a certain radius r. However, experimental calibrations show that using radius search for entropy calculation does not work. This is because with radius search, the optimisation algorithm finds a minimum when as few as possible points fall within the given radius. Thus the optimisation routine chooses the calibration parameters so that they disperse the point cloud as much as possible, yielding completely useless results. kNN search avoids this problem because it always considers the same number of points. Maddern et al. (2012) propose a further optimisation for choosing the number of neighbours to consider for each point, but this was not implemented in this thesis.

Entropy was found to yield the best and most consistent results out of the data-based methods considered here. In addition, as the brief comparison above in Section 6.1.2 suggested, the entropy cost function seems to be rather robust even when faced with large errors. However, the method is slow, both in calculating the cost function and to converge. These factors should be considered when planning any possible implementations of the method.

6.2 Limitations

This thesis has looked at a selection of calibration methods for lidar systems and compared their performance in a test environment. The results can not be generalised to any environment, any sensor or any measurement, but they provide a reference point for consideration when implementing a data-based calibration approach on a lidar system. In addition, the integrity of the target-based calibration method can be questioned, as there are obviously errors in TLS data as well. The TLS data was ultimately deemed to represent such a significant improvement in precision that it was considered a sufficient model of the space. However, the dismissal of the target-based method would dismiss also the scoring system, which would mean that the data-based calibration methods would have to be evaluated in another manner, perhaps visually. As the main contribution of this thesis is a quantitative evaluation, a purely visual, qualitative approach was deemed inadequate.

The results of this thesis only show calibration results for a single data set for each of the sensors. Statistical reliability through repetition would provide more robust results, but multiple data points were, in this case, instead achieved by multiple sensors. This was for the practical reasons that more sensors could be calibrated and less computational time exhausted.

6.3 Future research

The results presented in Section 5 indicate that there are real differences in chosen data-based calibration method. Future research might investigate further how to make the choice of calibration method in a given situation by comparing results using different methods in different environments. In addition, attention could be paid to the way different cost functions converge, and efficiency could be improved by first employing a fast method and then switching to a more accurate one. Efficiency and complexity are major concerns, since the number of calibration parameters is large and computation slow. Further work could also be done to determine how to choose the calibration parameters—as noted in Section 5.2, azimuth angles are almost constant, for example, and could be discarded—that most improve the results of measurement. More generally, the impacts of including or excluding different parameters could be studied. As the physical dimensions in Figure 5.1 show, the obtained calibration parameters are not always physically possible. Could a more exclusive (or inclusive) calibration produce more realistic results?

The methods themselves could most likely also be improved upon. The computational complexity of the entropy method could be addressed, for example, with an implementation on fewer points or by optimising the code. Though the local planarity method seems to be invalid, it may be interesting to experiment with an indicator function that limits the assessment of planarity to areas that are already somewhat planar. The plane detection method could be further improved to adjust plane detection parameters based on the measurements—if very planar features are available, the criteria may be tighter, and vice versa.

The work done for this thesis is meant for practical application. It would thus be worthwhile to consider the practical applicability of RMBL calibration—when a sensor system is deployed, is on-site recalibration useful? Do calibration parameters remain constant over time and in use, or should calibration be periodic? The incorporation of platform motion into calibration should be investigated, as the current versions of the methods do not consider it. Finally, if calibration parameters are not constant, a real-time methodology could be created for constantly tuning calibration parameters while measuring. Maye (2014), for example, studies online calibration for robotic systems and proposes a calibration pipeline using properties similar to entropy as discussed in this thesis.

7. Conclusion

This thesis set out to investigate geometric calibration methods for rotating multi-beam lidar (RMBL) systems, to describe their properties and evaluate their performance. Three data-based methods were investigated: plane segmentation and fitting, local planarity and entropy, in addition to which a target-based calibration using a reference point cloud was conducted for reference. The quantitative results of this thesis indicate that the data-based calibration methods can markedly improve calibration in sensor systems. Especially the plane fitting and entropy methods were found to be useful, with plane fitting leading the way with short computation times and entropy providing robustness and reliability.

As discussed, the supremacy of one method is not absolute. Different circumstances and considerations can provide grounds for choosing another method or combining methods. The properties and behaviour of different cost functions in different environments require further research and any implementation should consider this and other points raised earlier. This thesis has presented a variety of factors for consideration in practical implementations, and though the uniqueness of the test sensor and environment may hinder direct application in universal settings, a limited amount of customisation should yield useful results. The algorithms presented in Section 4 and Appendix B bring together research results from across the field and provide broad replicability for practitioners.

As lidar systems become increasingly common in practical applications, further attention might be afforded to the behaviour of calibration procedures in different environments and different systems. As discussed, the point cloud distribution determines, to an extent, which algorithms can or should be considered, and the distributions produced by the system examined here only represent a limited section of the realm of possibilities. At the same time, the stability of calibration parameters has not been addressed by this thesis, and the possibility of online calibration is promising especially concerning fully automatic mobile systems that run for extended periods of time. The methods discussed address a static offline calibration, which is only valid in future measurements if calibration parameters are sufficiently stable. Addressing these and more considerations provides grounds for much research.

Indeed, much of this research has been done already, and this thesis is largely based on a selection of previous work that proposed self-calibration methods for lidar sensors and systems. However, previously proposed calibration methods have not previously been evaluated and compared insofar as the author is aware. The calibration problem reduces to identifying a metric for point cloud quality, and as such, the fundamental question is what metric best measures this quality. This thesis proposed a scoring method for determining the quality of such metrics and used this method to evaluate calibration approaches. All methods can improve calibration, which suggests both that initial calibrations are often unsatisfactory and that any calibration that can be conducted typically pays off. However, the local planarity method does not do so consistently and, according to these results, provides inadequate parameters by getting stuck in local minima. Moreover, large differences in performance emphasise that the choice between plane fitting and entropy is not arbitrary, either. A quick calibration in an appropriate setting can be conducted with plane detection and fitting, while the entropy method seems to be the most accurate and universally applicable.

References

Alismail, H., & Browning, B. (2015, August). Automatic Calibration of Spinning Actuated Lidar Internal Parameters: Automatic Calibration of Spinning Actuated Lidar. *Journal of Field Robotics*, *32*(5), 723–747. doi: 10.1002/rob.21543

An, S.-Y., Lee, L.-K., & Oh, S.-Y. (2015, October). Line segment-based fast 3D plane extraction using nodding 2D laser rangefinder. *Robotica*, *33*(8), 1751–1774. doi: 10.1017/S0263574714000927

Atanacio-Jiménez, G., González-Barbosa, J.-J., Hurtado-Ramos, J. B., Ornelas-Rodríguez, F. J., Jiménez-Hernández, H., García-Ramirez, T., & González-Barbosa, R. (2011, November). LIDAR Velodyne HDL-64E Calibration Using Pattern Planes. *International Journal of Advanced Robotic Systems*, *8*(5), 59. doi: 10.5772/50900

Batavia, P., Roth, S., & Singh, S. (2002, September). Autonomous coverage operations in semi-structured outdoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (Vol. 1, pp. 743–749 vol.1). doi: 10.1109/IRDS.2002.1041479

Blanco, J. L., & Rai, P. K. (2014). *nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees.* Retrieved from `https://github.com/jlblancoc/nanoflann`

Borrmann, D., Elseberg, J., Lingemann, K., & Nüchter, A. (2011, November). The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, *2*(2), 3. doi: 10.1007/3DRes.02(2011)3

Bosse, M., & Zlot, R. (2009, May). Continuous 3D scan-matching with a spinning 2D laser. In *2009 IEEE International Conference on Robotics and Automation* (pp. 4312–4319). Kobe: IEEE. doi: 10.1109/ROBOT.2009.5152851

Bromiley, P. A. (2013, November). Products and Convolutions of Gaussian Probability Density Functions. *Tina-Vision Memo*, *3*(4), 1–5. doi: 10.1.1.583.3007

Chan, T. O., & Lichti, D. D. (2013). Feature-based self-calibration of Velodyne HDL-32E lidar for terrestrial mobile mapping applications. In *The 8th International Symposium on Mobile Mapping Technology* (pp. 1–3). Tainan, Taiwan.

Chen, C., Zou, X., Tian, M., Li, J., Wu, W., Song, Y., ... Yang, B. (2017, November). Low cost multi-sensor robot laser scanning system and its accuracy investigations for indoor mapping application. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-2/W8*, 83–85. doi: 10.5194/isprs-archives-XLII-2-W8-83-2017

Chen, C.-Y., & Chien, H.-J. (2012, October). On-Site Sensor Recalibration of a Spinning Multi-Beam LiDAR System Using Automatically-Detected Planar Targets. *Sensors*, *12*(10), 13736–13752. doi: 10.3390/s121013736

Collis, R. T. H. (1965, August). Lidar Observation of Cloud. *Science*, *149*(3687), 978–981. doi: 10.1126/science.149.3687.978

Dias, P., Matos, M., & Santos, V. (2006). 3D Reconstruction of Real World

Scenes Using a Low-Cost 3D Range Scanner. *Computer-Aided Civil and Infrastructure Engineering*, *21*(7), 486–497. doi: 10.1111/j.1467-8667.2006.00453.x

Droeschel, D., Schwarz, M., & Behnke, S. (2017, February). Continuous mapping and localization for autonomous navigation in rough terrain using a 3D laser scanner. *Robotics and Autonomous Systems*, *88*, 104–115. doi: 10.1016/j.robot.2016.10.017

Ellis, C. G. (2019, March). *Self-driving vehicles safety system.* U.S. Patent No. 10235877. Minneapolis, MN. Retrieved 2019-08-14, from `http://www.freepatentsonline.com/10235877.html`

Gallo, O., Manduchi, R., & Rafii, A. (2011, February). CC-RANSAC: Fitting planes in the presence of multiple surfaces in range data. *Pattern Recognition Letters*, *32*(3), 403–410. doi: 10.1016/j.patrec.2010.10.009

Gong, Z., Wen, C., Wang, C., & Li, J. (2018, January). A Target-Free Automatic Self-Calibration Approach for Multibeam Laser Scanners. *IEEE Transactions on Instrumentation and Measurement*, *67*(1), 238–240. doi: 10.1109/TIM.2017.2757148

Guennebaud, G., Jacob, B., & others. (2010). *Eigen v3*. Retrieved from `http://eigen.tuxfamily.org`

Hecht, J. (2018, January). Lidar for Self-Driving Cars. *Optics and Photonics News*, *29*(1), 26–33.

Hulik, R., Spanel, M., Smrz, P., & Materna, Z. (2014, January). Continuous plane detection in point-cloud data based on 3D Hough Transform. *Journal of Visual Communication and Image Representation*, *25*(1), 86–97. doi: 10.1016/j.jvcir.2013.04.001

Hyyti, H., Mäkelä, J., Kukko, A., & Kaartinen, H. (2019). An integrated positioning and mapping sensor for forest machinery. In *Open Engineering Automation in Finland 2019 Special Issue.* Oulu: Suomen Automaatioseura ry.

James Ring. (1963, July). The laser in astronomy. *New Scientist*, *18*(344), 672–673.

Jaynes, E. (1963). Information Theory and Statistical Mechanics. In *Statistical Physics* (pp. 181–218). New York, Amsterdam: W.A. Benjamin, Inc. Retrieved 2019-07-11, from `http://bayes.wustl.edu/etj/articles/brandeis.pdf`

Jesús Morales, Victoria Plaza-Leiva, Anthony Mandow, Jose Gomez-Ruiz, Javier Serón, & Alfonso García-Cerezo. (2018, January). Analysis of 3D Scan Measurement Distribution with Application to a Multi-Beam Lidar on a Rotating Platform. *Sensors*, *18*(2), 395. doi: 10.3390/s18020395

Kang, J., & Doh, N. L. (2016, October). Full-DOF Calibration of a Rotating 2-D LIDAR With a Simple Plane Measurement. *IEEE Transactions on Robotics*, *32*(5), 1245–1263. doi: 10.1109/TRO.2016.2596769

Klamt, T., & Behnke, S. (2017, September). Anytime hybrid driving-stepping locomotion planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4444–4451). (ISSN: 2153-0866) doi: 10.1109/IROS.2017.8206310

Le Gall, F. (2014). Powers of Tensors and Fast Matrix Multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation* (pp. 296–303). New York, NY, USA: ACM. (event-place: Kobe, Japan) doi: 10.1145/2608628.2608664

Le Gall, F., & Urrutia, F. (2018, January). Improved Rectangular Matrix Multiplication using Powers of the Coppersmith-Winograd Tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 1029–1046). Society for Industrial and Applied Mathematics. doi: 10.1137/1.9781611975031.67

Leica Geosystems. (2018). *Leica ScanStation P50/P40/P30 User Manual.* Leica Geosystems. Retrieved 2019-10-10, from `surveyequipment.com/assets/index/download/id/457/`

Leica Geosystems. (2019). *Leica Pegasus: Backpack Wearable Mobile Mapping Solution.* Retrieved 2019-10-23, from `https://leica-geosystems.com/products/mobile-sensor-platforms/capture-platforms/leica-pegasus-backpack`

Leingartner, M., Maurer, J., Ferrein, A., & Steinbauer, G. (2016). Evaluation of Sensors and Mapping Approaches for Disasters in Tunnels. *Journal of Field Robotics*, *33*(8), 1037–1057. doi: 10.1002/rob.21611

Levinson, J., & Thrun, S. (2014). Unsupervised Calibration for Multi-beam Lasers. In O. Khatib, V. Kumar, & G. Sukhatme (Eds.), *Experimental Robotics* (Vol. 79, pp. 179–193). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-28572-1_13

Li, L., Yang, F., Zhu, H., Li, D., Li, Y., & Tang, L. (2017, May). An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells. *Remote Sensing*, *9*(5), 433. doi: 10.3390/rs9050433

Lichti, D. D., Stewart, M. P., Tsakiri, M., & Snow, A. J. (2000). Calibration and testing of a terrestrial laser scanner. *International Archives of Photogrammetry and Remote Sensing*, *33, Part B*, 485–492.

lidar, n. (2019, August). Oxford University Press. Retrieved 2019-08-14, from `http://www.oed.com/view/Entry/108026`

Maddern, W., Harrison, A., & Newman, P. (2012, May). Lost in translation (and rotation): Rapid extrinsic calibration for 2D and 3D LIDARs. In *2012 IEEE International Conference on Robotics and Automation* (pp. 3096–3102). St Paul, MN, USA: IEEE. doi: 10.1109/ICRA.2012.6224607

Marquardt, D. (1963, June). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, *11*(2), 431–441. doi: 10.1137/0111030

Martínez, J. L., Morales, J., Reina, A. J., Mandow, A., Pequeño-Boter, A., & García-Cerezo, A. (2015, March). Construction and calibration of a low-cost 3D laser scanner with 360 field of view for mobile robots. In *2015 IEEE International Conference on Industrial Technology (ICIT)* (pp. 149–154). doi: 10.1109/ICIT.2015.7125091

MATLAB. (2019). *Release 2019a.* Natick, Massachusetts: The MathWorks Inc.

Maye, J. (2014). *Online self-calibration for robotic systems* (Doctoral dissertation, ETH Zurich, Switzerland). (Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 21912, 2014.) doi: 10.3929/ethz-a-010213941

McDaniel, M. W., Nishihata, T., Brooks, C. A., & Iagnemma, K. (2010, May). Ground plane identification using LIDAR in forested environments. In *2010 IEEE International Conference on Robotics and Automation* (pp. 3831–3836). Anchorage, AK: IEEE. doi: 10.1109/ROBOT.2010.5509963

MEX File Functions - MATLAB & Simulink - MathWorks Nordic. (2019). Retrieved 2019-12-12, from `https://se.mathworks.com/help/matlab/call-mex-file-functions.html`

Molebny, V., Kamerman, G., & Steinvall, O. (2010, October). Laser radar: from early history to new trends. In G. W. Kamerman et al. (Eds.), *Electro-Optical Remote Sensing, Photonic Technologies, and Applications IV* (Vol. 7835, pp. 9–38). Toulouse, France: SPIE. doi: 10.1117/12.867906

Moon, Y.-G., Go, S.-J., Yu, K.-H., & Lee, M.-C. (2015, July). Development of 3D laser range finder system for object recognition. In *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)* (pp. 1402–1405). (ISSN: 2159-6247, 2159-6255) doi: 10.1109/AIM.2015.7222736

Morales, J., Martínez, J. L., Mandow, A., Pequeño-Boter, A., & García-Cerezo, A. (2011, April). Design and development of a fast and precise low-cost 3D laser rangefinder. In *2011 IEEE International Conference on Mechatronics* (pp. 621–626). doi: 10.1109/ICMECH.2011.5971190

Morales, J., Martínez, J. L., Mandow, A., Reina, A. J., Pequeño-Boter, A., & García-Cerezo, A. (2014, November). Boresight Calibration of Construction Misalignments for 3D Scanners Built with a 2D Laser Rangefinder Rotating on Its Optical Center. *Sensors*, *14*(11), 20025–20040. doi: 10.3390/s141120025

Muhammad, N., & Lacroix, S. (2010, October). Calibration of a rotating multi-beam lidar. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 5648–5653). Taipei: IEEE. doi: 10.1109/IROS.2010.5651382

Muja, M., & Lowe, D. (2013, January). *FLANN - Fast Library for Approximate Nearest Neighbors.* Retrieved from `https://www.cs.ubc.ca/research/flann/`

Nelder, J. A., & Mead, R. (1965, January). A Simplex Method for Function Minimization. *The Computer Journal*, *7*(4), 308–313. doi: 10.1093/comjnl/7.4.308

Nelder–Mead method. (2019, September). Retrieved 2019-10-24, from `https://en.wikipedia.org/w/index.php?title=Nelder%E2%80%93Mead_method&oldid=915181932` (Page Version ID: 915181932)

Neumann, T., Dülberg, E., Schiffer, S., & Ferrein, A. (2016). A Rotating Platform for Swift Acquisition of Dense 3D Point Clouds. In N. Kubota, K. Kiguchi, H. Liu, & T. Obo (Eds.), *Intelligent Robotics and Applications* (Vol. 9834, pp. 257–268). Cham: Springer International Publishing. doi: 10.1007/978-3-319-43506-0_22

Neumann, T., Ferrein, A., Kallweit, S., & Scholl, I. (2014). Towards a Mobile Mapping Robot for Underground Mines. In *Proceedings of the 2014 PRASA, RobMech and AfLaT International Joint Symposium* (pp. 279–284). Cape

Town, South Africa.

Nouira, H., Deschaud, J. E., & Goulette, F. (2015, August). Target-free extrinsic calibration of a mobile multi-beam lidar system. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *II-3/W5*, 97–104. doi: 10.5194/isprsannals-II-3-W5-97-2015

Nouira, H., Deschaud, J.-E., & Goulette, F. (2016, July). Point cloud refinement with a target-free intrinsic calibration of a mobile multi-beam lidar system. In *ISPRS congress 2016 International Society for Photogrammetry and Remote Sensing* (p. 9). Prague, Czech Republic.

Oberlander, J., Pfotzer, L., Roennau, A., & Dillmann, R. (2015, September). Fast calibration of rotating and swivelling 3-D laser scanners exploiting measurement redundancies. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 3038–3044). Hamburg, Germany: IEEE. doi: 10.1109/IROS.2015.7353796

Pan, V. Y., & Chen, Z. Q. (1999). The Complexity of the Matrix Eigenproblem. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing* (pp. 507–516). New York, NY, USA: ACM. (event-place: Atlanta, Georgia, USA) doi: 10.1145/301250.301389

Pandey, G., McBride, J. R., Savarese, S., & Eustice, R. M. (2015, August). Automatic Extrinsic Calibration of Vision and Lidar by Maximizing Mutual Information. *Journal of Field Robotics*, *32*(5), 696–722. doi: 10.1002/rob.21542

Paracosm. (2019). *PX-80 Overview.* Retrieved 2019-10-23, from `https://labs.paracosm.io/px-80-overview`

Parzen, E. (1962, September). On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, *33*(3), 1065–1076. doi: 10.1214/aoms/1177704472

Pfrunder, A., Borges, P. V. K., Romero, A. R., Catt, G., & Elfes, A. (2017, September). Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3D LiDAR. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2601–2608). (ISSN: 2153-0866) doi: 10.1109/IROS.2017.8206083

Principe, J. C., & Dongxin Xu. (1999, October). Learning from examples with Renyi's information criterion. In *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers (Cat. No.CH37020)* (Vol. 2, pp. 966–970 vol.2). doi: 10.1109/ACSSC.1999.831853

Rosenblatt, M. (1956, September). Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, *27*(3), 832–837. doi: 10.1214/aoms/1177728190

Rényi, A. (1960). On measures of entropy and information. In *Proceedings of the fourth Berkeley Symposium on Mathematics* (pp. 547–561).

Schubert, S., Neubert, P., & Protzel, P. (2016). How to Build and Customize a High-Resolution 3D Laserscanner Using Off-the-shelf Components. In L. Alboul, D. Damian, & J. M. Aitken (Eds.), *Towards Autonomous Robotic Systems* (pp. 314–326). Springer International Publishing.

Schulz, T. (2008). *Calibration of a terrestrial laser scanner for engineering geodesy* (Doctoral Thesis, ETH Zurich). doi: 10.3929/ethz-a-005368245

Shannon, C. E. (1948, July). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379–423. doi: 10.1002/j.1538-7305.1948.tb01338.x

Shaukat, A., Blacker, P. C., Spiteri, C., & Gao, Y. (2016, November). Towards Camera-LIDAR Fusion-Based Terrain Modelling for Planetary Surfaces: Review and Analysis. *Sensors*, *16*(11), 1952. doi: 10.3390/s16111952

Sheehan, M., Harrison, A., & Newman, P. (2012, April). Self-calibration for a 3D laser. *The International Journal of Robotics Research*, *31*(5), 675–687. doi: 10.1177/0278364911429475

Sheehan, M., Harrison, A., & Newman, P. (2014). Automatic Self-calibration of a Full Field-of-View 3D n-Laser Scanner. In O. Khatib, V. Kumar, & G. Sukhatme (Eds.), *Experimental Robotics* (Vol. 79, pp. 165–178). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-28572-1_12

Sheh, R., Jamali, N., Kadous, M. W., & Sammut, C. (2006, December). A Low-Cost, Compact, Lightweight 3D Range Sensor. In *Australian conference on robotics and automation* (p. 8).

Tarsha-Kurdi, F., Landes, T., & Grussenmeyer, P. (2007). Hough-Transform and Extended RANSAC Algorithms for Automatic Detection of 3D Building Roof Planes from Lidar Data. In *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007* (Vol. 36, pp. 407–412).

k-d tree. (2019, November). Retrieved 2019-12-05, from `https://en.wikipedia.org/w/index.php?title=K-d_tree&oldid=928055279` (Page Version ID: 928055279)

The MathWorks, Inc. (2019). *Optimizing Nonlinear Functions - MATLAB & Simulink - MathWorks Nordic.* Retrieved 2019-09-25, from `https://se.mathworks.com/help/matlab/math/optimizing-nonlinear-functions.html`

Torr, P. H. S., & Zisserman, A. (2000, April). MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, *78*(1), 138–156. doi: 10.1006/cviu.1999.0832

Ueda, T., Kawata, H., Tomizawa, T., Ohya, A., & Yuta, S. (2006). Mobile SOKUIKI Sensor System: Accurate Range Data Mapping System with Sensor Motion. In *Proceedings of the International Conference on Autonomous Robots and Agents* (pp. 304–309). Palmerston North, New Zealand.

Upton, G., & Cook, I. (2008, January). three-sigma rule. In *A Dictionary of Statistics.* Oxford University Press. Retrieved 2019-08-26, from `https://www.oxfordreference.com/view/10.1093/acref/9780199541454.001.0001/acref-9780199541454-e-1639`

Velodyne Lidar. (2008). *HDL-64E: User's manual.* Velodyne Acoustics, Inc.

Velodyne Lidar. (2015a). *HDL-32E: User's manual and programming guide.* Velodyne LiDAR, Inc.

Velodyne Lidar. (2015b). *Velodyne Lidar Puck Datasheet.* Velodyne LiDAR, Inc.

Velodyne Lidar. (2018a). *Puck LITE: light weight real-time 3D LiDAR sensor.* Velodyne LiDAR, Inc.

Velodyne Lidar. (2018b). *VLP-32C User Manual.*

Venter, G. (2010, December). Review of Optimization Techniques.. doi: 10.1002/9780470686652.eae495

Weingarten, J., & Siegwart, R. (2006, October). 3D SLAM using planar segments. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3062–3067). (ISSN: 2153-0858, 2153-0866) doi: 10.1109/IROS.2006.282245

Wulf, O., & Wagner, B. (2003, July). Fast 3D scanning methods for laser measurement systems. In *International conference on control systems and computer science (CSCS14)* (pp. 2–5).

Xiao, J., Zhang, J., Adler, B., Zhang, H., & Zhang, J. (2013, December). Three-dimensional point cloud plane segmentation in both structured and unstructured environments. *Robotics and Autonomous Systems*, *61*(12), 1641–1652. doi: 10.1016/j.robot.2013.07.001

Xie, D., Xu, Y., & Wang, R. (2019, March). Obstacle detection and tracking method for autonomous vehicle based on three-dimensional LiDAR. *International Journal of Advanced Robotic Systems*, *16*(2), 1729881419831587. doi: 10.1177/1729881419831587

Yoshida, T., Irie, K., Koyanagi, E., & Tomono, M. (2010, October). A sensor platform for outdoor navigation using gyro-assisted odometry and roundly-swinging 3D laser scanner. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1414–1420). (ISSN: 2153-0866, 2153-0858, 2153-0858) doi: 10.1109/IROS.2010.5652172

Özbay, B., Kuzucu, E., Gül, M., Öztürk, D., Taşcı, M., Arısoy, A. M., ... Uyanık, I. (2015, July). A high frequency 3D LiDAR with enhanced measurement density via Papoulis-Gerchberg. In *2015 International Conference on Advanced Robotics (ICAR)* (pp. 543–548). doi: 10.1109/ICAR.2015.7251509

A. Calibration parameters

Calibrated parameter values for each sensor and each method are available in the following tables. Angles are given in degrees and distances in millimetres. The first five parameters for each sensor are extrinsic and the rest intrinsic, and these are separated by a line.

A.1 VLP-16 Lite

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
t_x (mm)	0	-5	9	-0	10
t_y (mm)	0	-0	4	0	9
φ (°)	0	-0.10	0.07	0.00	0.13
θ (°)	40	39.75	40.16	38.73	39.57
ψ (°)	0	-0.73	-0.51	-0.73	-0.64
ω_1 (°)	-15	-13.26	-14.41	-12.15	-12.88
ω_2 (°)	1	2.80	1.81	2.57	2.63
ω_3 (°)	-13	-10.97	-12.16	-9.72	-11.15
ω_4 (°)	3	4.80	3.78	5.13	4.18
ω_5 (°)	-11	-9.40	-10.17	-7.96	-9.47
ω_6 (°)	5	6.92	5.83	7.04	6.98
ω_7 (°)	-9	-7.25	-8.12	-4.81	-7.41
ω_8 (°)	7	8.85	7.76	9.52	8.40
ω_9 (°)	-7	-5.08	-6.15	-4.82	-5.16
ω_{10} (°)	9	10.80	9.82	9.92	10.35
ω_{11} (°)	-5	-3.29	-4.18	-2.25	-3.18
ω_{12} (°)	11	13.02	11.91	12.54	12.61
ω_{13} (°)	-3	-1.15	-2.08	-0.39	-1.83
ω_{14} (°)	13	14.78	13.87	14.21	14.48
ω_{15} (°)	-1	0.92	-0.12	2.13	0.63
ω_{16} (°)	15	16.79	16.00	16.90	16.57
β_1 (°)	0	-0.01	-0.02	0.00	0.04
β_2 (°)	0	0.01	-0.02	-0.00	0.12
β_3 (°)	0	-0.01	0.02	-0.00	0.01
β_4 (°)	0	0.01	-0.00	-0.01	0.02
β_5 (°)	0	-0.00	-0.00	0.00	-0.01
β_6 (°)	0	0.00	-0.06	-0.01	0.00
β_7 (°)	0	0.01	0.06	0.00	0.16
β_8 (°)	0	0.05	0.01	-0.00	0.01
β_9 (°)	0	0.00	0.07	-0.08	0.00
β_{10} (°)	0	0.03	-0.03	-0.04	-0.02
β_{11} (°)	0	0.01	0.00	-0.04	0.08
β_{12} (°)	0	-0.00	-0.01	-0.00	0.04
β_{13} (°)	0	-0.00	0.01	-0.01	-0.00
β_{14} (°)	0	0.04	0.00	-0.00	-0.01
β_{15} (°)	0	0.00	-0.00	0.00	0.02

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
β_{16} (°)	0	0.00	-0.00	-0.00	-0.02
δ_{x1} (mm)	0	14	11	3	-72
δ_{x2} (mm)	0	-9	4	-72	5
δ_{x3} (mm)	0	-3	4	2	6
δ_{x4} (mm)	0	-3	-3	0	21
δ_{x5} (mm)	0	1	12	-0	12
δ_{x6} (mm)	0	2	-2	2	0
δ_{x7} (mm)	0	2	5	-2	-0
δ_{x8} (mm)	0	-0	-0	1	22
δ_{x9} (mm)	0	-1	8	-3	9
δ_{x10} (mm)	0	-4	-7	0	30
δ_{x11} (mm)	0	5	10	-0	9
δ_{x12} (mm)	0	-10	-8	0	-1
δ_{x13} (mm)	0	-3	1	-12	-0
δ_{x14} (mm)	0	-4	-5	42	10
δ_{x15} (mm)	0	-0	-2	-9	-3
δ_{x16} (mm)	0	-6	-7	-1	24
δ_{y1} (mm)	0	2	-7	-38	-3
δ_{y2} (mm)	0	1	-6	5	-9
δ_{y3} (mm)	0	-11	-9	4	-13
δ_{y4} (mm)	0	-0	-5	-0	-9
δ_{y5} (mm)	0	-6	-1	-7	-2
δ_{y6} (mm)	0	-5	-3	8	7
δ_{y7} (mm)	0	-4	-2	13	-0
δ_{y8} (mm)	0	-1	-9	0	-19
δ_{y9} (mm)	0	-1	-0	1	13
δ_{y10} (mm)	0	-4	-9	2	-16
δ_{y11} (mm)	0	-5	-1	-26	-23
δ_{y12} (mm)	0	1	-9	14	1
δ_{y13} (mm)	0	-4	-6	1	-4
δ_{y14} (mm)	0	-2	-3	0	-13
δ_{y15} (mm)	0	-1	-5	-10	6
δ_{y16} (mm)	0	3	-4	-1	-7
δ_{z1} (mm)	0	61	17	-0	67
δ_{z2} (mm)	0	8	-4	-3	-9
δ_{z3} (mm)	0	-0	-15	-3	-14
δ_{z4} (mm)	0	3	-4	3	59
δ_{z5} (mm)	0	81	-3	26	53
δ_{z6} (mm)	0	-14	-5	-0	-58
δ_{z7} (mm)	0	39	-15	-106	22
δ_{z8} (mm)	0	-8	10	1	35
δ_{z9} (mm)	0	0	-10	0	-12
δ_{z10} (mm)	0	7	3	-0	40
δ_{z11} (mm)	0	37	-4	-6	-17

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
δ_{z12} (mm)	0	-35	-11	2	-7
δ_{z13} (mm)	0	5	-19	-2	74
δ_{z14} (mm)	0	1	2	64	-3
δ_{z15} (mm)	0	-3	-13	-1	5
δ_{z16} (mm)	0	0	-15	-15	0

Table 1.1. Calibration parameter values for VLP-16 Lite.

A.2 VLP-16 (1)

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
t_x (mm)	0	1	-3	0	-5
t_y (mm)	0	-6	1	-0	-0
φ (°)	0	0.00	0.05	0.00	0.01
θ (°)	40	39.69	39.73	39.37	39.65
ψ (°)	0	0.40	0.42	-0.00	0.40
ω_1 (°)	-15	-15.12	-15.01	-14.39	-14.94
ω_2 (°)	1	1.02	1.00	0.84	1.05
ω_3 (°)	-13	-13.04	-12.90	-13.15	-12.97
ω_4 (°)	3	3.09	3.02	3.14	2.89
ω_5 (°)	-11	-11.07	-10.98	-11.22	-11.03
ω_6 (°)	5	5.15	5.06	5.40	4.88
ω_7 (°)	-9	-9.13	-9.06	-9.09	-9.03
ω_8 (°)	7	-635.67	-114.36	5.99	-10.57
ω_9 (°)	-7	-6.89	-7.11	-6.76	-7.19
ω_{10} (°)	9	9.18	8.98	8.36	8.87
ω_{11} (°)	-5	-5.08	-5.00	-4.42	-5.03
ω_{12} (°)	11	11.32	11.04	10.70	11.01
ω_{13} (°)	-3	-3.06	-3.03	-2.08	-3.05
ω_{14} (°)	13	13.22	13.04	12.89	12.93
ω_{15} (°)	-1	-1.10	-1.11	-1.63	-1.22
ω_{16} (°)	15	15.38	15.05	15.23	15.01
β_1 (°)	0	-0.00	0.06	0.00	0.07
β_2 (°)	0	0.00	0.02	-0.00	-0.00
β_3 (°)	0	-0.01	0.04	-0.00	-0.00
β_4 (°)	0	-0.03	0.00	0.00	-0.00
β_5 (°)	0	-0.03	0.05	-0.00	0.00
β_6 (°)	0	-0.00	0.00	0.00	0.00
β_7 (°)	0	-0.03	-0.00	0.00	-0.00
β_8 (°)	0	-0.34	-0.96	-0.00	0.01
β_9 (°)	0	-0.07	0.02	-0.00	-0.01
β_{10} (°)	0	0.00	-0.00	-0.00	0.00
β_{11} (°)	0	0.00	-0.00	0.00	-0.01
β_{12} (°)	0	-0.01	0.00	0.00	0.00

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
β_{13} (°)	0	-0.06	-0.01	0.00	0.00
β_{14} (°)	0	0.00	0.02	-0.00	0.00
β_{15} (°)	0	-0.00	0.01	0.00	0.01
β_{16} (°)	0	-0.05	-0.00	0.00	-0.00
δ_{x1} (mm)	0	-3	-4	0	0
δ_{x2} (mm)	0	-5	-4	-1	-7
δ_{x3} (mm)	0	-4	-10	0	-8
δ_{x4} (mm)	0	-3	-2	-0	3
δ_{x5} (mm)	0	0	-5	-0	-0
δ_{x6} (mm)	0	7	-0	0	21
δ_{x7} (mm)	0	5	-4	0	-3
δ_{x8} (mm)	0	-5	-1954	-0	44
δ_{x9} (mm)	0	6	2	-0	12
δ_{x10} (mm)	0	-4	1	-0	-0
δ_{x11} (mm)	0	5	-5	0	-3
δ_{x12} (mm)	0	0	-3	1	9
δ_{x13} (mm)	0	-0	-3	-0	-0
δ_{x14} (mm)	0	2	-3	-0	9
δ_{x15} (mm)	0	14	6	1	18
δ_{x16} (mm)	0	-15	-2	0	7
δ_{y1} (mm)	0	8	-0	1	-1
δ_{y2} (mm)	0	-0	-2	1	-1
δ_{y3} (mm)	0	8	1	0	3
δ_{y4} (mm)	0	0	-0	1	-2
δ_{y5} (mm)	0	-1	0	0	4
δ_{y6} (mm)	0	1	1	0	-4
δ_{y7} (mm)	0	4	2	-1	0
δ_{y8} (mm)	0	146	-10	-1	-3
δ_{y9} (mm)	0	0	-1	1	-0
δ_{y10} (mm)	0	1	-2	-0	-2
δ_{y11} (mm)	0	3	0	1	0
δ_{y12} (mm)	0	2	-0	1	-0
δ_{y13} (mm)	0	2	-0	-0	-2
δ_{y14} (mm)	0	8	1	0	-1
δ_{y15} (mm)	0	1	-0	-0	-1
δ_{y16} (mm)	0	4	-0	0	-1
δ_{z1} (mm)	0	55	6	-1	-0
δ_{z2} (mm)	0	17	-5	-1	-10
δ_{z3} (mm)	0	39	-13	-0	-1
δ_{z4} (mm)	0	1	-8	-1	10
δ_{z5} (mm)	0	62	2	-0	9
δ_{z6} (mm)	0	-7	-1	-1	24
δ_{z7} (mm)	0	48	0	-1	-4
δ_{z8} (mm)	0	1195	186	-0	-103

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
δ_{z9} (mm)	0	6	24	0	30
δ_{z10} (mm)	0	-17	-0	0	15
δ_{z11} (mm)	0	38	-5	-0	-4
δ_{z12} (mm)	0	-48	-10	-1	-9
δ_{z13} (mm)	0	31	1	-0	0
δ_{z14} (mm)	0	-31	-13	-1	-0
δ_{z15} (mm)	0	45	26	1	37
δ_{z16} (mm)	0	-61	-11	-1	-11

Table 1.2. Calibration parameter values for VLP-16 (1).

A.3 VLP-16 (2)

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
t_x (mm)	0	-9	9	0	4
t_y (mm)	0	-5	-1	-0	-1
φ (°)	0	0.12	0.14	-0.00	0.05
θ (°)	40	39.77	39.78	39.48	39.74
ψ (°)	0	0.31	0.29	-0.00	0.28
ω_1 (°)	-15	-15.19	-15.12	-15.00	-15.14
ω_2 (°)	1	1.12	1.01	1.10	1.04
ω_3 (°)	-13	-13.06	-13.06	-12.53	-13.06
ω_4 (°)	3	3.10	2.90	2.99	2.94
ω_5 (°)	-11	-11.11	-11.04	-10.58	-11.12
ω_6 (°)	5	5.15	4.95	4.80	4.91
ω_7 (°)	-9	-9.09	-9.04	-8.38	-9.05
ω_8 (°)	7	7.30	6.97	6.95	6.93
ω_9 (°)	-7	-7.02	-7.06	-6.57	-7.03
ω_{10} (°)	9	9.29	9.02	8.73	9.04
ω_{11} (°)	-5	-5.18	-5.23	-4.66	-5.26
ω_{12} (°)	11	11.18	10.89	10.60	10.66
ω_{13} (°)	-3	-3.03	-3.02	-2.86	-3.09
ω_{14} (°)	13	13.20	12.68	12.49	12.65
ω_{15} (°)	-1	-1.12	-1.13	-0.70	-1.31
ω_{16} (°)	15	15.49	15.09	14.51	15.16
β_1 (°)	0	0.10	0.04	0.00	0.03
β_2 (°)	0	0.03	-0.02	0.00	-0.04
β_3 (°)	0	0.05	-0.00	0.00	-0.00
β_4 (°)	0	-0.01	0.01	0.00	0.01
β_5 (°)	0	-0.04	0.01	-0.00	0.00
β_6 (°)	0	-0.04	-0.00	-0.00	-0.04
β_7 (°)	0	0.01	0.03	0.00	-0.00
β_8 (°)	0	0.00	-0.02	0.00	-0.05
β_9 (°)	0	0.00	-0.04	0.00	-0.05
β_{10} (°)	0	0.05	-0.00	0.00	-0.03

Appendix A (6/16)

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
β_{11} (°)	0	0.02	0.02	0.00	0.00
β_{12} (°)	0	-0.01	-0.02	-0.00	-0.05
β_{13} (°)	0	-0.02	-0.01	0.00	-0.02
β_{14} (°)	0	0.06	-0.01	0.00	-0.04
β_{15} (°)	0	-0.00	-0.00	0.00	-0.02
β_{16} (°)	0	0.04	0.04	-0.00	0.00
δ_{x1} (mm)	0	4	2	-0	13
δ_{x2} (mm)	0	-14	-12	0	-10
δ_{x3} (mm)	0	-1	-9	0	1
δ_{x4} (mm)	0	-5	-1	0	7
δ_{x5} (mm)	0	3	-8	-0	0
δ_{x6} (mm)	0	-7	-2	0	9
δ_{x7} (mm)	0	-5	-11	0	-1
δ_{x8} (mm)	0	-15	-2	1	8
δ_{x9} (mm)	0	2	-9	0	-1
δ_{x10} (mm)	0	-13	-6	0	9
δ_{x11} (mm)	0	-0	8	-0	20
δ_{x12} (mm)	0	-26	7	0	6
δ_{x13} (mm)	0	0	-8	1	3
δ_{x14} (mm)	0	-17	-1	-1	7
δ_{x15} (mm)	0	2	0	0	25
δ_{x16} (mm)	0	-28	-13	0	1
δ_{y1} (mm)	0	1	1	-0	1
δ_{y2} (mm)	0	2	-1	0	-2
δ_{y3} (mm)	0	-1	-0	-0	-2
δ_{y4} (mm)	0	-2	-1	0	-2
δ_{y5} (mm)	0	-2	0	0	0
δ_{y6} (mm)	0	4	0	-0	-1
δ_{y7} (mm)	0	-6	-1	0	0
δ_{y8} (mm)	0	4	-1	0	0
δ_{y9} (mm)	0	-3	-0	-0	-0
δ_{y10} (mm)	0	-2	-0	0	0
δ_{y11} (mm)	0	-1	-1	-0	-2
δ_{y12} (mm)	0	3	-0	0	-0
δ_{y13} (mm)	0	3	-1	0	-2
δ_{y14} (mm)	0	-0	-2	-0	-2
δ_{y15} (mm)	0	2	-1	0	-1
δ_{y16} (mm)	0	1	-0	-0	-2
δ_{z1} (mm)	0	102	33	0	31
δ_{z2} (mm)	0	26	-11	0	-22
δ_{z3} (mm)	0	72	14	0	7
δ_{z4} (mm)	0	26	13	-1	-4
δ_{z5} (mm)	0	85	11	0	16
δ_{z6} (mm)	0	16	10	0	3

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
δ_{z7} (mm)	0	71	1	-2	-4
δ_{z8} (mm)	0	-9	6	-0	0
δ_{z9} (mm)	0	58	0	-0	-15
δ_{z10} (mm)	0	-14	-6	-0	-26
δ_{z11} (mm)	0	94	49	0	46
δ_{z12} (mm)	0	6	12	-0	33
δ_{z13} (mm)	0	59	-0	-0	-1
δ_{z14} (mm)	0	-5	43	0	34
δ_{z15} (mm)	0	75	31	0	48
δ_{z16} (mm)	0	-53	-22	1	-49

Table 1.3. Calibration parameter values for VLP-16 (2).

A.4 VLP-16 (3)

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
t_x (mm)	0	-9	5	11	9
t_y (mm)	0	-3	-5	-2	1
φ (°)	0	-0.00	0.00	-0.00	0.01
θ (°)	40	39.75	40.00	39.54	39.72
ψ (°)	0	-0.14	-0.10	0.00	-0.16
ω_1 (°)	-15	-13.35	-14.44	-12.91	-13.28
ω_2 (°)	1	2.78	1.57	2.50	2.53
ω_3 (°)	-13	-11.31	-12.40	-10.92	-11.39
ω_4 (°)	3	4.77	3.56	4.44	4.34
ω_5 (°)	-11	-9.10	-10.42	-9.53	-9.41
ω_6 (°)	5	6.83	5.61	6.87	6.58
ω_7 (°)	-9	-7.36	-8.40	-7.56	-7.42
ω_8 (°)	7	8.96	7.52	9.04	8.52
ω_9 (°)	-7	-5.41	-6.48	-5.27	-5.52
ω_{10} (°)	9	10.87	9.61	10.16	10.51
ω_{11} (°)	-5	-3.11	-4.48	-3.22	-3.42
ω_{12} (°)	11	12.74	11.56	12.09	12.55
ω_{13} (°)	-3	-1.16	-2.41	-0.75	-1.44
ω_{14} (°)	13	14.91	13.58	14.41	14.58
ω_{15} (°)	-1	0.58	-0.43	-0.04	0.50
ω_{16} (°)	15	16.76	15.49	16.57	16.38
β_1 (°)	0	0.06	0.00	0.00	0.06
β_2 (°)	0	-0.00	-0.01	-0.00	-0.02
β_3 (°)	0	-0.00	-0.00	-0.01	0.08
β_4 (°)	0	-0.00	-0.02	0.00	0.00
β_5 (°)	0	-0.01	0.04	-0.00	0.04
β_6 (°)	0	0.00	0.00	-0.00	-0.00
β_7 (°)	0	-0.05	0.01	-0.00	0.01
β_8 (°)	0	0.02	-0.00	0.00	0.00

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
β_9 (°)	0	-0.01	-0.00	0.00	-0.00
β_{10} (°)	0	0.04	-0.03	0.00	-0.00
β_{11} (°)	0	-0.01	-0.03	-0.00	-0.01
β_{12} (°)	0	-0.01	-0.02	-0.00	0.01
β_{13} (°)	0	-0.03	-0.04	0.01	-0.01
β_{14} (°)	0	0.03	-0.02	0.00	-0.00
β_{15} (°)	0	0.01	-0.03	0.00	0.00
β_{16} (°)	0	0.00	-0.01	0.00	-0.01
δ_{x1} (mm)	0	10	9	1	-1
δ_{x2} (mm)	0	2	6	9	8
δ_{x3} (mm)	0	-3	8	-4	-6
δ_{x4} (mm)	0	-6	1	1	7
δ_{x5} (mm)	0	1	8	0	-7
δ_{x6} (mm)	0	4	-12	2	12
δ_{x7} (mm)	0	-0	5	-1	-5
δ_{x8} (mm)	0	-8	-4	2	-0
δ_{x9} (mm)	0	7	9	0	-0
δ_{x10} (mm)	0	-2	-0	3	1
δ_{x11} (mm)	0	2	12	1	1
δ_{x12} (mm)	0	-0	-0	3	-0
δ_{x13} (mm)	0	-5	2	2	-7
δ_{x14} (mm)	0	0	-10	-3	3
δ_{x15} (mm)	0	7	-1	2	-0
δ_{x16} (mm)	0	-1	-6	-1	2
δ_{y1} (mm)	0	3	6	-3	-0
δ_{y2} (mm)	0	2	1	-4	1
δ_{y3} (mm)	0	0	4	-1	-4
δ_{y4} (mm)	0	-2	1	-1	-0
δ_{y5} (mm)	0	1	5	-11	1
δ_{y6} (mm)	0	-3	-1	-3	-0
δ_{y7} (mm)	0	5	-1	2	2
δ_{y8} (mm)	0	0	-6	3	1
δ_{y9} (mm)	0	1	-1	10	0
δ_{y10} (mm)	0	-0	1	1	0
δ_{y11} (mm)	0	-0	2	-1	1
δ_{y12} (mm)	0	-3	1	-6	-0
δ_{y13} (mm)	0	1	-0	-3	-1
δ_{y14} (mm)	0	4	0	-1	1
δ_{y15} (mm)	0	-2	-2	-1	-0
δ_{y16} (mm)	0	3	-3	0	-0
δ_{z1} (mm)	0	74	2	-1	2
δ_{z2} (mm)	0	17	8	-4	3
δ_{z3} (mm)	0	72	-5	-1	5
δ_{z4} (mm)	0	-0	4	3	17

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
δ_{z5} (mm)	0	14	-4	-7	4
δ_{z6} (mm)	0	2	-3	-1	-9
δ_{z7} (mm)	0	58	-8	-3	-5
δ_{z8} (mm)	0	-35	6	-1	-13
δ_{z9} (mm)	0	59	6	-2	9
δ_{z10} (mm)	0	-10	5	1	-3
δ_{z11} (mm)	0	-0	13	1	-0
δ_{z12} (mm)	0	9	11	3	-9
δ_{z13} (mm)	0	1	-11	1	-12
δ_{z14} (mm)	0	-12	4	3	-8
δ_{z15} (mm)	0	55	0	-2	7
δ_{z16} (mm)	0	-9	11	-3	2

Table 1.4. Calibration parameter values for VLP-16 (3).

A.5 VLP-16 (4)

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
t_x (mm)	0	16	16	2	9
t_y (mm)	0	6	-0	1	-1
φ (°)	0	-0.03	-0.13	0.00	-0.08
θ (°)	40	39.78	39.82	39.32	39.75
ψ (°)	0	-1.05	-0.84	-0.00	-1.07
ω_1 (°)	-15	-15.02	-15.14	-14.83	-15.19
ω_2 (°)	1	0.96	0.83	1.28	0.78
ω_3 (°)	-13	-13.36	-13.12	-13.32	-13.24
ω_4 (°)	3	3.03	2.77	3.12	2.57
ω_5 (°)	-11	-11.28	-11.23	-11.42	-11.36
ω_6 (°)	5	4.79	4.77	5.12	4.70
ω_7 (°)	-9	-9.25	-9.24	-9.45	-9.33
ω_8 (°)	7	6.85	6.83	7.22	6.58
ω_9 (°)	-7	-7.52	-7.23	-7.34	-7.27
ω_{10} (°)	9	9.35	8.80	8.76	8.68
ω_{11} (°)	-5	-5.06	-5.25	-4.41	-5.14
ω_{12} (°)	11	11.32	10.67	10.64	10.56
ω_{13} (°)	-3	-3.01	-3.20	-2.38	-3.26
ω_{14} (°)	13	12.81	12.74	12.58	12.73
ω_{15} (°)	-1	-1.09	-1.17	-0.28	-1.07
ω_{16} (°)	15	14.84	14.85	14.66	14.77
β_1 (°)	0	0.17	-0.00	-0.00	0.00
β_2 (°)	0	0.01	-0.02	-0.00	0.02
β_3 (°)	0	-0.06	-0.04	-0.00	0.01
β_4 (°)	0	-0.00	-0.02	0.00	0.05
β_5 (°)	0	-0.00	-0.03	-0.00	-0.02
β_6 (°)	0	0.07	-0.00	-0.00	-0.02

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
β_7 (°)	0	0.03	-0.01	-0.00	-0.00
β_8 (°)	0	0.02	0.01	0.00	-0.00
β_9 (°)	0	0.06	-0.04	0.00	-0.00
β_{10} (°)	0	0.14	0.00	0.00	0.03
β_{11} (°)	0	-0.03	-0.06	-0.00	0.03
β_{12} (°)	0	0.00	0.03	-0.00	0.02
β_{13} (°)	0	-0.00	-0.03	0.00	0.04
β_{14} (°)	0	0.07	0.00	0.00	0.07
β_{15} (°)	0	0.02	-0.03	-0.00	-0.01
β_{16} (°)	0	0.02	0.01	-0.00	0.05
δ_{x1} (mm)	0	7	-6	-1	-4
δ_{x2} (mm)	0	15	-0	-2	-0
δ_{x3} (mm)	0	-16	-7	-1	-8
δ_{x4} (mm)	0	-10	-0	-1	-2
δ_{x5} (mm)	0	19	-2	0	1
δ_{x6} (mm)	0	25	-2	-4	-3
δ_{x7} (mm)	0	16	-1	0	1
δ_{x8} (mm)	0	11	-4	1	7
δ_{x9} (mm)	0	7	-4	3	3
δ_{x10} (mm)	0	-18	-2	-1	-0
δ_{x11} (mm)	0	4	-2	0	12
δ_{x12} (mm)	0	-18	8	1	-0
δ_{x13} (mm)	0	25	-4	-1	0
δ_{x14} (mm)	0	14	4	0	-0
δ_{x15} (mm)	0	22	0	2	2
δ_{x16} (mm)	0	8	-5	0	0
δ_{y1} (mm)	0	-28	-2	-0	4
δ_{y2} (mm)	0	0	-1	1	0
δ_{y3} (mm)	0	-10	-1	2	-0
δ_{y4} (mm)	0	13	-2	1	-1
δ_{y5} (mm)	0	-14	-3	2	3
δ_{y6} (mm)	0	-11	0	-0	5
δ_{y7} (mm)	0	-14	-3	-1	2
δ_{y8} (mm)	0	-8	-2	-4	3
δ_{y9} (mm)	0	-16	-0	0	5
δ_{y10} (mm)	0	-8	-2	0	-2
δ_{y11} (mm)	0	-9	0	0	-3
δ_{y12} (mm)	0	-3	0	1	2
δ_{y13} (mm)	0	-5	-0	2	-1
δ_{y14} (mm)	0	-2	-2	1	2
δ_{y15} (mm)	0	-2	-1	-1	-0
δ_{y16} (mm)	0	-16	-1	-2	3
δ_{z1} (mm)	0	0	-3	1	9
δ_{z2} (mm)	0	-6	1	0	12

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
δ_{z3} (mm)	0	62	-6	0	7
δ_{z4} (mm)	0	-41	-10	-1	21
δ_{z5} (mm)	0	51	1	1	21
δ_{z6} (mm)	0	16	-2	-1	7
δ_{z7} (mm)	0	40	4	-1	14
δ_{z8} (mm)	0	-1	-13	1	20
δ_{z9} (mm)	0	84	2	1	-4
δ_{z10} (mm)	0	-104	-11	0	-1
δ_{z11} (mm)	0	3	14	0	-3
δ_{z12} (mm)	0	-88	8	0	33
δ_{z13} (mm)	0	-1	7	-2	23
δ_{z14} (mm)	0	1	-1	1	-1
δ_{z15} (mm)	0	15	-1	1	-4
δ_{z16} (mm)	0	-1	-20	1	-14

Table 1.5. Calibration parameter values for VLP-16 (4).

A.6 VLP-32C

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
t_x (mm)	0	19	-0	1	2
t_y (mm)	0	1	1	1	0
φ (°)	0	0.00	-0.00	0.00	0.00
θ (°)	40	39.74	39.84	39.46	39.69
ψ (°)	0	-0.03	0.00	-0.00	0.00
ω_1 (°)	-25.00	-24.77	-24.79	-24.39	-24.69
ω_2 (°)	-1.00	-0.98	-0.95	-0.59	-1.10
ω_3 (°)	-1.67	-1.41	-1.56	-0.66	-1.23
ω_4 (°)	-15.64	-15.54	-15.56	-15.27	-15.42
ω_5 (°)	-11.31	-11.06	-11.11	-11.02	-11.02
ω_6 (°)	0.00	0.00	0.00	-0.00	-0.00
ω_7 (°)	-0.67	-0.61	-0.56	-0.46	-0.49
ω_8 (°)	-8.84	-8.75	-8.75	-8.16	-8.71
ω_9 (°)	-7.25	-7.03	-7.04	-7.00	-6.98
ω_{10} (°)	0.33	0.04	0.39	0.15	0.44
ω_{11} (°)	-0.33	-0.05	-0.19	-0.32	-0.09
ω_{12} (°)	-6.15	-6.14	-6.10	-5.79	-6.04
ω_{13} (°)	-5.33	-5.25	-5.24	-5.47	-5.19
ω_{14} (°)	1.33	1.39	1.48	1.56	1.52
ω_{15} (°)	0.67	0.13	0.70	1.41	0.63
ω_{16} (°)	-4.00	-3.98	-3.94	-3.95	-3.77
ω_{17} (°)	-4.67	-4.47	-4.52	-4.04	-4.46
ω_{18} (°)	1.67	1.67	1.72	1.63	1.57
ω_{19} (°)	1.00	1.16	1.21	1.50	1.42
ω_{20} (°)	-3.67	-3.54	-3.62	-3.88	-3.64

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
ω_{21} (°)	-3.33	-3.22	-3.14	-2.56	-3.21
ω_{22} (°)	3.33	3.31	3.36	3.41	3.40
ω_{23} (°)	2.33	2.51	2.52	3.16	2.57
ω_{24} (°)	-2.67	-2.77	-2.60	-2.46	-2.55
ω_{25} (°)	-3.00	-2.88	-2.85	-2.49	-2.64
ω_{26} (°)	7.00	6.99	7.11	7.00	7.09
ω_{27} (°)	4.67	4.82	4.86	4.96	4.87
ω_{28} (°)	-2.33	-2.35	-2.30	-2.41	-2.42
ω_{29} (°)	-2.00	-1.94	-1.89	-0.71	-1.91
ω_{30} (°)	15.00	15.10	15.11	15.02	15.12
ω_{31} (°)	10.33	10.44	10.51	10.72	10.56
ω_{32} (°)	-1.33	-1.28	-1.26	-0.65	-1.20
β_{1} (°)	1.40	1.73	1.73	1.74	1.77
β_{2} (°)	-4.20	-3.79	-3.76	-3.75	-3.69
β_{3} (°)	-3.00	1.82	0.41	1.84	1.85
β_{4} (°)	-1.40	-1.18	-1.14	-1.22	-1.15
β_{5} (°)	1.40	1.71	1.79	1.80	1.83
β_{6} (°)	-1.40	-1.04	-1.01	-0.94	-0.96
β_{7} (°)	4.20	4.74	4.65	4.68	4.72
β_{8} (°)	-1.40	-1.09	-1.05	-1.01	-1.01
β_{9} (°)	1.40	1.72	1.80	1.79	1.83
β_{10} (°)	-4.20	-3.73	-3.73	-3.69	-3.65
β_{11} (°)	1.40	1.83	1.82	1.85	1.89
β_{12} (°)	-1.40	-1.11	-1.05	-1.05	-1.05
β_{13} (°)	4.20	4.62	4.62	4.69	4.69
β_{14} (°)	-1.40	-0.98	-1.01	-0.98	-0.95
β_{15} (°)	4.20	4.64	4.65	4.72	4.71
β_{16} (°)	-1.40	-1.00	-0.99	-0.95	-0.95
β_{17} (°)	1.40	1.80	1.81	1.87	1.84
β_{18} (°)	-4.20	-3.67	-3.69	-3.66	-3.60
β_{19} (°)	1.40	1.79	1.80	1.88	1.82
β_{20} (°)	-4.20	-3.71	-3.71	-3.61	-3.61
β_{21} (°)	4.20	4.61	4.63	4.71	4.69
β_{22} (°)	-1.40	-0.94	-0.98	-0.90	-0.91
β_{23} (°)	1.40	1.75	1.78	1.82	1.82
β_{24} (°)	-1.40	-1.08	-1.02	-1.04	-0.99
β_{25} (°)	1.40	1.74	1.77	1.80	1.83
β_{26} (°)	-1.40	-1.03	-1.01	-0.97	-0.97
β_{27} (°)	1.40	1.89	1.85	1.95	1.90
β_{28} (°)	-4.20	-3.78	-3.75	-3.69	-3.66
β_{29} (°)	4.20	4.63	4.64	4.59	4.71
β_{30} (°)	-1.40	-0.98	-0.99	-0.92	-0.92
β_{31} (°)	1.40	2.00	1.91	1.99	2.02
β_{32} (°)	-1.40	-1.09	-1.04	-1.11	-0.99

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
δ_{x1} (mm)	0	3	-1	4	-2
δ_{x2} (mm)	0	0	-4	-2	0
δ_{x3} (mm)	0	0	8	3	-0
δ_{x4} (mm)	0	-2	-8	1	0
δ_{x5} (mm)	0	7	-6	-0	-0
δ_{x6} (mm)	0	-3	-0	-2	1
δ_{x7} (mm)	0	-1	0	-0	-0
δ_{x8} (mm)	0	-0	1	-3	1
δ_{x9} (mm)	0	-4	1	0	1
δ_{x10} (mm)	0	0	0	-4	-1
δ_{x11} (mm)	0	-3	-1	-6	1
δ_{x12} (mm)	0	-2	0	8	1
δ_{x13} (mm)	0	-8	-5	-2	0
δ_{x14} (mm)	0	-4	-1	-1	7
δ_{x15} (mm)	0	-1	-0	-0	-0
δ_{x16} (mm)	0	2	-0	1	-0
δ_{x17} (mm)	0	-0	-1	4	-2
δ_{x18} (mm)	0	-1	-0	4	-1
δ_{x19} (mm)	0	-1	-0	1	1
δ_{x20} (mm)	0	-0	0	2	0
δ_{x21} (mm)	0	-9	-3	2	-0
δ_{x22} (mm)	0	-0	0	-2	-2
δ_{x23} (mm)	0	-1	0	8	1
δ_{x24} (mm)	0	2	3	2	-2
δ_{x25} (mm)	0	-1	1	2	0
δ_{x26} (mm)	0	-3	-1	0	-0
δ_{x27} (mm)	0	-1	0	3	6
δ_{x28} (mm)	0	-4	-3	-3	1
δ_{x29} (mm)	0	-3	0	-1	2
δ_{x30} (mm)	0	2	-0	1	2
δ_{x31} (mm)	0	-1	-11	1	1
δ_{x32} (mm)	0	-4	-1	1	1
δ_{y1} (mm)	0	0	0	-1	0
δ_{y2} (mm)	0	-1	1	1	1
δ_{y3} (mm)	0	1	0	-2	0
δ_{y4} (mm)	0	-1	0	-6	1
δ_{y5} (mm)	0	3	-0	9	0
δ_{y6} (mm)	0	1	1	-4	-0
δ_{y7} (mm)	0	-18	-0	1	-1
δ_{y8} (mm)	0	1	0	-1	-0
δ_{y9} (mm)	0	-0	-0	-1	0
δ_{y10} (mm)	0	5	-0	-1	1
δ_{y11} (mm)	0	3	-0	1	0
δ_{y12} (mm)	0	0	-2	1	1

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
δ_{y13} (mm)	0	5	0	-1	-3
δ_{y14} (mm)	0	-5	0	6	0
δ_{y15} (mm)	0	-1	0	-5	0
δ_{y16} (mm)	0	1	-1	-8	1
δ_{y17} (mm)	0	2	-3	-0	1
δ_{y18} (mm)	0	-1	-1	3	-1
δ_{y19} (mm)	0	-0	-0	-12	5
δ_{y20} (mm)	0	0	-1	-0	-2
δ_{y21} (mm)	0	4	1	4	0
δ_{y22} (mm)	0	-1	0	-0	0
δ_{y23} (mm)	0	-1	0	3	1
δ_{y24} (mm)	0	0	0	3	-1
δ_{y25} (mm)	0	-1	0	-3	-1
δ_{y26} (mm)	0	-1	-0	-2	1
δ_{y27} (mm)	0	1	-0	-1	0
δ_{y28} (mm)	0	-1	-1	-2	0
δ_{y29} (mm)	0	1	1	4	-0
δ_{y30} (mm)	0	2	-2	-1	-0
δ_{y31} (mm)	0	-8	0	-1	-0
δ_{y32} (mm)	0	1	0	-0	-0
δ_{z1} (mm)	0	-0	0	-9	2
δ_{z2} (mm)	0	0	-0	1	0
δ_{z3} (mm)	0	0	-0	-0	-3
δ_{z4} (mm)	0	0	1	-3	1
δ_{z5} (mm)	0	-1	-0	2	-1
δ_{z6} (mm)	0	0	3	-1	0
δ_{z7} (mm)	0	0	-1	-3	0
δ_{z8} (mm)	0	0	0	-1	-0
δ_{z9} (mm)	0	1	-0	1	1
δ_{z10} (mm)	0	5	-0	4	-0
δ_{z11} (mm)	0	-1	-0	-0	2
δ_{z12} (mm)	0	1	1	0	0
δ_{z13} (mm)	0	2	1	-0	-2
δ_{z14} (mm)	0	2	-0	0	-3
δ_{z15} (mm)	0	5	-0	1	-0
δ_{z16} (mm)	0	9	0	4	0
δ_{z17} (mm)	0	-5	-1	-6	0
δ_{z18} (mm)	0	-0	0	-1	1
δ_{z19} (mm)	0	-0	-0	2	-1
δ_{z20} (mm)	0	-5	0	3	1
δ_{z21} (mm)	0	0	-8	-1	0
δ_{z22} (mm)	0	-0	-2	-3	1
δ_{z23} (mm)	0	0	0	4	-0
δ_{z24} (mm)	0	24	0	3	0

Parameter	Default	Target-based	Plane fitting	Planarity	Entropy
δ_{z25} (mm)	0	-0	-2	-1	0
δ_{z26} (mm)	0	1	-10	-2	-2
δ_{z27} (mm)	0	-1	-5	2	1
δ_{z28} (mm)	0	5	3	2	1
δ_{z29} (mm)	0	-0	-1	1	-0
δ_{z30} (mm)	0	1	4	-1	-1
δ_{z31} (mm)	0	2	1	-4	-2
δ_{z32} (mm)	0	4	1	1	1

Table 1.6. Calibration parameter values for VLP-32C.

Appendix A (16/16)

B. Algorithms in pseudocode

B.1 Target-based calibration algorithm

Algorithm 1 describes a target-based calibration procedure where the target is a reference point cloud.

Algorithm 1: Calibration algorithm using target-based point–point distance as cost function.

```
input  : Measurements $\hat{\mathrm{M}}$
input  : Reference point cloud $\mathrm{R}$
input  : Initial parameters $\mathrm{p}_0$
input  : Maximum iterations $m$
output : Errorsum $e$
output : Final parameters $\mathrm{p}_f$

1  begin Calibration
2      $i \leftarrow 0$
3      while not converged and $i < m$ do
4          $\hat{\mathrm{X}} \leftarrow$ transformToCartesian($\hat{\mathrm{M}}$,$\mathrm{p}_i$)
5          $e \leftarrow 0$
6          for $j \leftarrow 1$ to number of points in $\hat{\mathrm{X}}$ do
7              $\mathbf{r} \leftarrow$ nearestNeighbour($\hat{\mathbf{x}}_j, \mathrm{R}$)
8              $e \leftarrow e + \|\hat{\mathbf{x}}_j - \mathbf{r}\|$
9          end
10         $i \leftarrow i + 1$
11         $p_i \leftarrow$ modifyParameters($p_{i-1}$)
12     end
13 end
```

B.2 Plane detection algorithm

Algorithm 3, adapted from Li et al. (2017) describes plane detection based on NDT cells and RANSAC. Algorithm 2 is a description of the iterative reweighted least-squares (IRLS) plane refinement step that is part of Algorithm 3.

Algorithm 2: Iterative reweighted least-squares plane refinement.

```
input  : Plane $P_0$ with $n$ points $\hat{\mathbf{X}}$, centroid $\mu$, normal $\mathbf{n}_0$
input  : Maximum iterations $k_{max}$
input  : Termination threshold $\gamma$
output : Refined plane $P_f$ with $n$ points $\hat{\mathbf{X}}$, centroid $\mu$, normal $\mathbf{n}_f$

1  for $k \leftarrow 1$ to $k_{max}$ do
2      $\mathbf{r} \leftarrow \|(\hat{\mathbf{X}} - \mu) \cdot \mathbf{n}\|$ // row-wise dot product
3      $\mathbf{w} \leftarrow \exp\left(-\frac{\mathbf{r}^{\odot 2}}{2.985^2}\right)$
4      $\mathbf{X}_k = \frac{\mathbf{w}(\hat{\mathbf{X}} - \mu - \mathbf{X}_{k-1})}{\sum \mathbf{w}}$
5      $\mathbf{C} \leftarrow \left(\mathbf{w} \odot (\hat{\mathbf{X}} - \mu - \mathbf{X}_{k-1})\right)^T (\hat{\mathbf{X}} - \mu - \mathbf{X}_{k-1})$
6      $\lambda_{1-3}, \mathbf{e}_{1-3} \leftarrow$ eigenvalueDecomposition(C)
7      $\mathbf{n}_k \leftarrow \mathbf{e}_1$
8      $\gamma_k \leftarrow \max(\text{abs}((\mathbf{n}_{k-1} - \mathbf{n}_k) \oslash \mathbf{n}_{k-1}))$
9      if $\gamma_k < \gamma$ then
10         return
11     end
12 end
```

Algorithm 3: Plane detection using RANSAC and normal distribution transform (NDT) cells, based on Li et al. (2017).

```
input  : Observations $\hat{\mathbf{X}}$
input  : NDT cell size $s$
input  : Planarity threshold $t_e$
input  : Acceptance ratio $r$ for planes ($0 < r < 1$)
input  : Maximum iterations $k_{max}$
input  : Distance threshold $d$
input  : Parallellity threshold $\theta$
output : A set $P$ of planes
while last plane large enough do
    $A \leftarrow \emptyset$ // NDT cells
    $Q \leftarrow \emptyset$ // planar cells
    $\hat{\mathbf{X}}' \leftarrow \emptyset$ // remaining points
    Divide space spanned by $\hat{\mathbf{X}}$ into grid $A$ of cubes size $s$
    for each cell $i$ in $A$ do
        $\hat{\mathbf{X}}_i \leftarrow$ points in $a_i$
        $n_i \leftarrow$ number of points in $\hat{\mathbf{X}}_i$
        $\mathbf{C} \leftarrow 1/n_i \hat{\mathbf{X}}_i^T \hat{\mathbf{X}}_i$ // covariance matrix
        $\lambda_{1-3}, \mathbf{e}_{1-3} \leftarrow$ eigenvalueDecomposition($\mathbf{C}$) // $\lambda_1 \leq \lambda_2 \leq \lambda_3$
        if $\frac{\lambda_1}{\lambda_2} \leq t_e$ then
            $\mathbf{n}_i \leftarrow \mathbf{e_1}$ // normal
            $\mu_i \leftarrow$ mean($\hat{\mathbf{X}}_i$) // centroid
            $Q \leftarrow Q \cup a_i$
        else
            $\hat{\mathbf{X}}' \leftarrow \hat{\mathbf{X}}' \cup A_i$
        end
    end
    $k \leftarrow 0$
    $B \leftarrow \emptyset$ // support set for current best plane
    while $k < k_{max}$ do
        $c \leftarrow$ random cell in $Q$
        for each other cell $i$ in $Q$ do
            $d_i \leftarrow \|(\mu_i - \mu_c) \cdot \mathbf{n}_c\|$
            $\theta_i \leftarrow \mathbf{n}_i \cdot \mathbf{n}_c$
            if $d_i < d$ and $\theta_i < \theta$ then
                $I_k \leftarrow I_k \cup q_i$
            end
        end
        if $|B_k| > |B|$ then // compare number of points
            $B \leftarrow B_k$
            $k_{max} \leftarrow \lceil \frac{\ln(1-0.99)}{\ln(1-\frac{|B_k|}{|\hat{\mathbf{X}}|})} \rceil$
        end
        $k \leftarrow k + 1$
    end
    for each point $i$ in $\hat{\mathbf{X}}'$ do
        $d_i \leftarrow \|(\hat{\mathbf{x}}'_i - \mu_b) \cdot \mathbf{n}_b\|$
        if $d_i < d$ then
            $B \leftarrow B \cup \hat{\mathbf{x}}'_i$
        end
    end
    if $\frac{|B|}{|\hat{\mathbf{X}}|} \geq r$ then // plane covers enough points
        $B \leftarrow$ IRLS($B$) // improve plane fitting with IRLS algorithm
        $P \leftarrow P \cup B$
    else
        return
    end
end
```

B.3 Plane fitting calibration algorithm

Algorithm 4 describes the calibration procedure using plane fitting.

Algorithm 4: Calibration parameter optimisation with plane fitting.

```
input  : Parameter vector $\mathrm{p}_0$ with initial values
input  : Planes $P$ found in 3
input  : Scanner range and angle measurements $\hat{\mathrm{M}}$
output : Final parameters $\mathrm{p}_f$
output : Errorsum $e$ corresponding to final parameters

1  begin calibration
2      $i \leftarrow 0$
3      while not converged and $i < m$ do
4          $e \leftarrow 0$
5          $X \leftarrow$ transformToCartesian($M, \mathbf{p}_i$)
           // transformToCartesian converts range and angle measurements to Cartesian
              coordinates using current iteration of $parameters$
6          for $j \leftarrow 1$ to $|P|$ do
7              $C \leftarrow (\frac{1}{n_j})\hat{\mathbf{X}}_j^T\hat{\mathbf{X}}_j$ // covariance matrix
8              $\lambda_{1-3} \leftarrow$ eigenvalueDecomposition($C$) // $\lambda_1 \leq \lambda_2 \leq \lambda_3$
9              $e \leftarrow e + \lambda_1$
10         end
11         $i \leftarrow i + 1$
12         $p_i \leftarrow$ modifyParameters($p_{i-1}$)
13     end
14 end
```

B.4 Calibration algorithms based on local planarity

Algorithms 5 and 6 describe algorithms for calibration based on local planarity using neighbouring beams and a general local neighbourhood, respectively.

Algorithm 5: Calibration using local planarity across neighbouring beams, based on Levinson and Thrun (2014).

```
input  : Parameter vector $\mathbf{p}_0$ with initial values
input  : Scanner range and angle measurements $M$
input  : Number of adjacent scanners to consider $N$
input  : Number of neighbours to consider $k$
input  : Threshold distance between neighbours $d$
input  : Maximum iterations $m$
output : Parameter vector $\mathbf{p}_f$ determined through optimisation
output : Errorsum $e$ corresponding to final parameters
begin calibration
    $i \leftarrow 0$
    while not converged and $i < m$ do
        $e \leftarrow 0$
        $\hat{\mathrm{X}} \leftarrow$ transformToCartesian($M, \mathbf{p}_i$)
        // transformToCartesian converts range and angle measurements to Cartesian
           coordinates using current iteration of $\mathbf{p}$
        for $j \leftarrow 1$ to number of beams in scanner do
            for $l \leftarrow j - N$ to $j + N$ do
                for $p \leftarrow 1$ to $n_l$ do
                    $\hat{\mathbf{x}}_l \leftarrow$ nearestNeighbour($\hat{\mathbf{x}}_p$, $\hat{\mathrm{X}}_j$)
                    if $\|\hat{\mathbf{x}}_l - \hat{\mathbf{x}}_p\| > d$ then
                        continue
                    end
                    $\hat{\mathrm{L}} \leftarrow$ kNearestNeighbours($\hat{\mathbf{x}}_l$, $\hat{\mathrm{X}}$, $k$) // neighbourhood at $\hat{\mathbf{x}}_l$
                    $\mathrm{C} \leftarrow (\frac{1}{k})\hat{\mathbf{L}}^T\hat{\mathbf{L}}$ // covariance matrix
                    $\lambda_{1-3}, \mathbf{e}_{1-3} \leftarrow$ eigenvalueDecomposition(C) // $\lambda_1 \le \lambda_2 \le \lambda_3$
                    $e \leftarrow e + \|\mathbf{e}_1 \cdot (\hat{\mathbf{x}}_l - \hat{\mathbf{x}}_p)\|^2$
                end
            end
        end
        $i \leftarrow i + 1$
        $\mathbf{p}_i \leftarrow$ modifyParameters($p_{i-1}$)
    end
end
```

Algorithm 6: Calibration using general local planarity.

```
input  : Parameter vector $\mathbf{p}_0$ with initial values
input  : Scanner range and angle measurements $M$
input  : Number of neighbours to consider $k$
input  : Maximum iterations $m$
output : Parameter vector $\mathbf{p}_f$ determined through optimisation
output : Errorsum $e$ corresponding to final parameters

1  begin calibration
2      $i \leftarrow 0$
3      while not converged and $i < m$ do
4          $e \leftarrow 0$
5          $\hat{\mathbf{X}} \leftarrow$ transformToCartesian($M, \mathbf{p}_i$)
           // transformToCartesian converts range and angle measurements to Cartesian
              coordinates using current iteration of $\mathbf{p}$
6          for $j \leftarrow 1$ to $|\hat{\mathrm{X}}|$ do
7              $\hat{\mathrm{N}} \leftarrow$ kNearestNeighbours($\hat{\mathbf{x}}_j$, $\hat{\mathrm{X}}$, $k$) // neighbourhood at $\hat{\mathbf{x}}_j$
8              $\mathrm{C} \leftarrow (\frac{1}{k})\hat{\mathbf{N}}^T\hat{\mathbf{N}}$ // covariance matrix
9              $\lambda_{1-3} \leftarrow$ eigenvalueDecomposition(C) // $\lambda_1 \leq \lambda_2 \leq \lambda_3$
10             $e \leftarrow e + \lambda_1$
11         end
12         $i \leftarrow i + 1$
13         $\mathbf{p}_i \leftarrow$ modifyParameters($p_{i-1}$)
14     end
15 end
```

B.5 Entropy-based calibration algorithms

Algorithm 7 describes the entropy-based calibration algorithm without and Algorithm 8 with kNN.

Algorithm 7: Iterative parameter optimisation using entropy.

```
input  : Parameter vector $\mathbf{p}_0$ with initial values
input  : Scanner range and angle measurements $M$
input  : Free 'standard deviation' parameter $\sigma$
output : Parameter vector determined through optimisation
output : Entropy value e corresponding to final parameters

1 begin Iterative search for minimum $e$
2     $e \leftarrow 0$
3     $\hat{\mathbf{X}} \leftarrow$ transformToCartesian($M(i), \mathbf{p}_n$)
      // transformToCartesian converts range and angle measurements to Cartesian
         coordinates using current iteration of $p$
4     for $i \leftarrow 1$ to number of measurements do
5         for $j \leftarrow i+1$ to number of measurements do
6             $e \leftarrow e - \exp\big(-\frac{1}{\sigma^2}(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)^T(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)\big)$
7         end
8     end
9 end
```

Algorithm 8: Iterative parameter optimisation using entropy and kNN search.

```
input  : Parameter vector $\mathbf{p_0}$ with initial values
input  : Scanner range and angle measurements $M$
input  : Free 'standard deviation' parameter $\sigma$
input  : Number of neighbours $k$
output : Parameter vector determined through optimisation
output : Entropy value $e$ corresponding to final parameters
1  begin Iterative search for minimum $e$ by modifying $\mathbf{p}$
2      $e \leftarrow 0$
3      $\hat{\mathrm{X}} \leftarrow$ transformToCartesian($M(i), \mathbf{p}_n$)
       // transformToCartesian converts range and angle measurements to Cartesian
          coordinates using current iteration of $p$
4      $treeX \leftarrow$ buildKDTree($\hat{\mathrm{X}}$)
5      for $i \leftarrow 1$ to number of measurements do
6          $\hat{\mathrm{N}}_i \leftarrow$ kNNSearch($treeX, \hat{\mathbf{x}}_i, k$); // radiusSearch finds points in $\hat{\mathrm{X}}$ within $r$
              distance from $\hat{\mathbf{x}}_i$
7          $e_i \leftarrow 0$
8          for $j \leftarrow 0$ to number $k$ of neighbors found do
9              $e_i \leftarrow e_i - \exp\big(-\frac{1}{\sigma^2}(\hat{\mathbf{x}}_i - \hat{\mathbf{n}}_j)^T(\hat{\mathbf{x}}_i - \hat{\mathbf{n}}_j)\big)$
10         end
11         $e \leftarrow e + \frac{e_i}{k}$
12     end
13 end
```