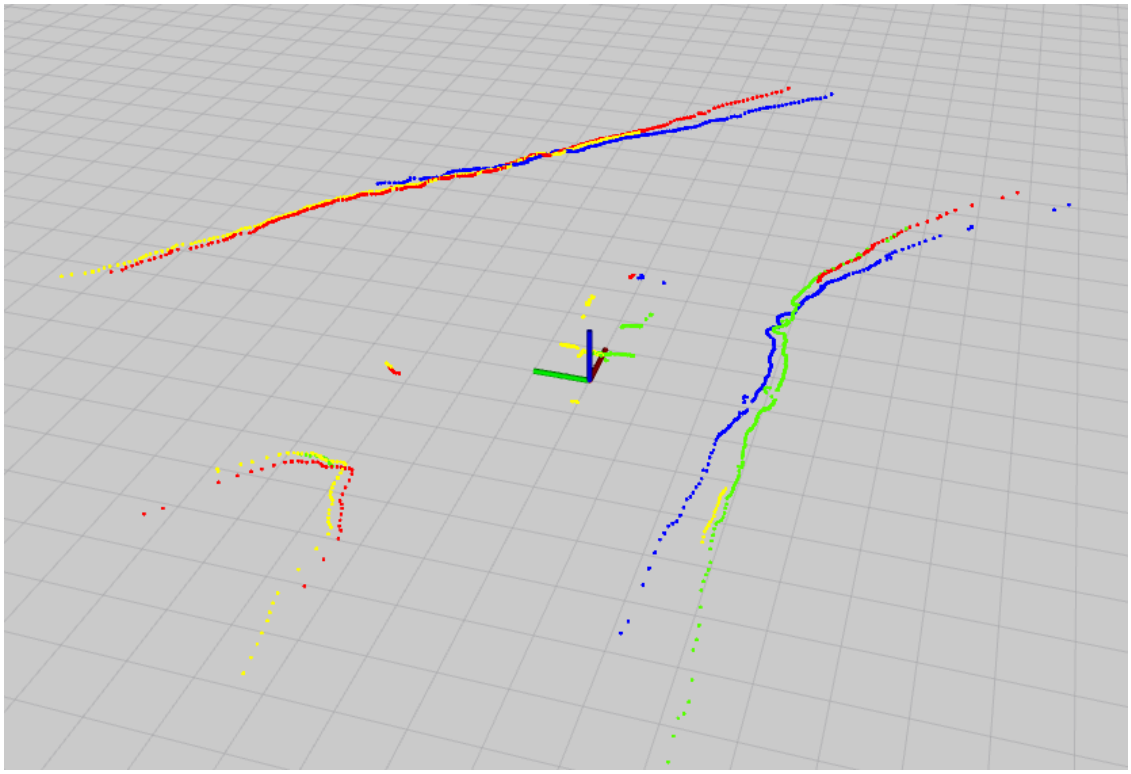
CHALMERS
UNIVERSITY OF TECHNOLOGY



# Extrinsic calibration of multiple 2D Lidars using a Genetic Algorithm

A robust method to estimate the x, y and yaw coordinates of vehicle-mounted 2D Lidars using data acquired when moving

Master's thesis in System Control and Mechatronics

LARS BROWN

GUSTAV LINDBERG

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019:07

# Extrinsic calibration of multiple 2D Lidars using a Genetic Algorithm

A robust method to estimate the x, y and yaw coordinates of vehicle-mounted 2D Lidars using data acquired when moving

LARS BROWN
GUSTAV LINDBERG


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
*Division of Vehicle Engineering and Autonomous Systems*
Chalmers University of Technology
Gothenburg, Sweden 2019

Extrinsic calibration of multiple 2D Lidars using a Genetic Algorithm
A robust method to estimate the x, y and yaw coordinates of vehicle-mounted 2D Lidars using data acquired when moving
LARS BROWN
GUSTAV LINDBERG



Supervisors:
Admas Yeshigeta, CPAC Systems AB
Sina Torabi, Department of Mechanics and Maritime Sciences

Examiner:
Peter Forsberg, Department of Mechanics and Maritime Sciences

Master's Thesis 2019:07
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visualization of four uncalibrated Lidars. The Lidar setup is mounted on a truck were each Lidars measurements are represented with a separate colour.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2019

Extrinsic calibration of multiple 2D Lidars using a Genetic Algorithm
A robust method to estimate the x, y and yaw coordinates of vehicle-mounted 2D Lidars using data acquired when moving
LARS BROWN, GUSTAV LINDBERG
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

# Abstract

To correctly transform and merge measurements from multiple vehicle-mounted Lidar sensors, it is essential to know the correct position of each Lidar. By using a genetic algorithm we have designed an alternative calibration technique which simultaneously estimates the $x$, $y$, and yaw coordinates of four 2D Lidars. A grid-based approach with decreasing cell size is used to evaluate the alignment of point cloud merging. Additionally, the method overcome difficulties of sparse input data by utilizing information from multiple perspectives attained as the vehicle is moving. With this feature the algorithm successfully calibrate the Lidar setup in more than 95% of the test cases. These tests are performed using data from indoor or outdoor environments and show that the technique is very much capable of calibrating such Lidar setups irregardless of the operating environment. Gathering data while the Lidars are moving induce distortions of the Lidar scans, wherefore compensation is also taken into account.

Keywords: 2D Lidar, extrinsic calibration, genetic algorithm, autonomous driving, multiple scenes,

# Acknowledgements

We would like to express our gratitude to both our advisor Admas Yeshigeta, Sina Torabi and our examiner Peter Forsberg. Especially we would like say thanks for showing such enthusiasm and the stimulative discussions we had along the way.

We are also thankful for the opportunity to write our thesis at CPAC Systems AB, where we have felt very welcome and appreciated. Last, and perhaps least, we would like to thank CPAC's exceptional coffee machine for providing us with energy, great conversations and ideas.

Lars Brown & Gustav Lindberg, Gothenburg, May 2019

Thesis Advisors: Admas Yeshigeta, Sina Torabi
Thesis Examiner: Peter Forsberg

# Glossary

## Abbreviations

| | |
|---|---|
| FL | Front Left, Lidar location |
| FR | Front Right, Lidar location |
| GA | Genetic Algorithm |
| ICP | Iterative Closest Point |
| IMU | Inertial Measurement Unit |
| Lidar | Light Detection and Ranging |
| RL | Rear Left, Lidar location |
| RR | Rear Right, Lidar location |
| RWS | Roulette Wheel Selection |

## Technical terms

| Term | Definition |
|---|---|
| Base link | The vehicle's point of rotation. Also, the origin of the coordinate system of the truck. |
| Chromosome | The information contained by an individual. |
| Gene | A part of a chromosome. |
| Individual | A candidate solution of a genetic algorithm. |
| Initial guess | A set of starting values (in the form of a chromosome) which estimates the 12 Lidar coordinates. |
| Point cloud | A set of data points in space. |
| Population | A set of individuals. |
| Premature convergence | The Genetic Algorithm converges to a local minima so the global optimum is not reached. |
| Scan | A 270° sweep of Lidar measurements. |
| Scan skewing distortion | Distortion of Lidar measurements if the Lidar is moving but assumed stationary. |
| Scan timing inconsistency | Lidar scans from different Lidars are unsynchronized in time. |
| Scene | A set of scans, one from each Lidar obtained at approximately the same time instance. |

# Contents

# List of Figures

# List of Tables

# 1
# Introduction

In recent years, autonomous trucks and heavy duty vehicles have been introduced in work yards and mines. In order to navigate such vehicles, it is of great importance to understand the surrounding environment. Various systems and methods have been developed both in academia and industry for navigation of autonomous vehicles. The navigation systems typically relies on clean and preprocessed data from various sensors such as Light detection and ranging (Lidar). In this thesis a truck equipped with four 2D Lidars is considered, as depicted in Figure 1.1. The raw available data from Lidars are, however, not suitable to be used directly. Instead, the individual scans can be merged into a single data set forming a virtual sensor covering a wider horizontal angle. A **scan** is here formed by a 270° sweep of Lidar measurements.

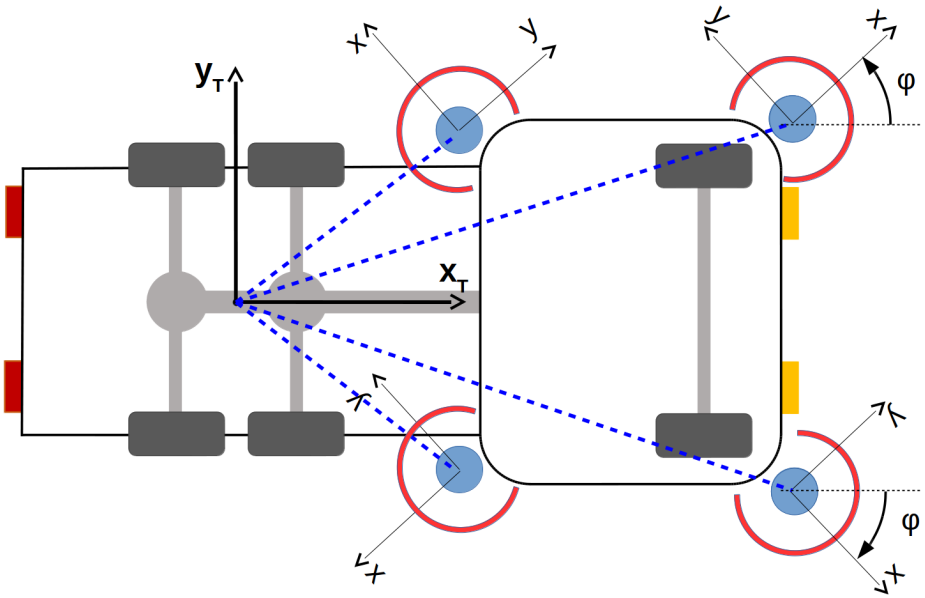



**Figure 1.1:** An illustration of a tractor unit equipped with four 2D Lidars marked as blue circles. The Lidar scans are transformed to the vehicle coordinate frame in order to be merged into a single point cloud.

By transforming all scans to the vehicle's coordinate system, denoted by $x_{\mathrm{T}}$ and $y_{\mathrm{T}}$ in Figure 1.1, the scans can be merged together. The result is a point cloud which resembles the view of all four Lidars, as if one, positioned at the vehicle origin. A **point cloud** is a set of data points in space. In this process, the $x$, $y$ and yaw coordinates of each Lidar are important to correctly transform measurements from each Lidar to form one merged point cloud. Using incorrect Lidar coordinates

results in duplicates of detected objects and consequently, the perception will not be accurate. Therefore, there is a need to accurately calibrate the position and yaw angle of each Lidar. However, to accurately measure each Lidar's coordinates by hand implies tedious, on the egde of impossible, work due to their physical arrangement. Instead one can make use of measured objects in common field of views and match their corresponding observations to estimate the coordinates. To do this, CPAC Systems, a developer of control systems for trucks manufactured by Volvo AB, makes use of a method called Iterative Closest Point (ICP). However, as will be described in Section 1.2.1, the ICP method has several drawbacks, where of one is the need of densely overlapping measurements. Therefore there is a desire for a more robust method which can estimate the Lidar coordinates despite sparse overlap. A further objective, which is a more challenging task, is also to carry out the calibration in cases where the truck is moving. This is the topic of this thesis work, which has been carried out in collaboration with CPAC Systems.

## 1.1 Purpose and limitations

The purpose of this thesis is to investigate how stochastic optimization can be used to calibrate the x, y and yaw coordinates of multiple Lidars. The work is focused on a genetic algorithm used to calibrate a set of four 2D Lidars. Trying to apply such an algorithm to this problem, the most central question of the project arises, *What is a good calibration?*
The project assumes that all Lidars are scanning within the same horizontal plane. However, the Lidars may be mounted such that its z-axis is pointing up or down.

## 1.2 Related work

In this section some calibration techniques and related literature are introduced and compared to the setup used in this work. The work on Lidar calibration can be divided in three general categories; Iterative Closest Point, feature-based calibration and occupancy grids. Hereafter, a summary of each of the methods is given in a separate section.

### 1.2.1 Iterative Closest Point

Iterative Closest Point (ICP) based methods are often applied to minimize the difference between two point clouds, and has been widely used within robotic positioning [1], [2], [3]. In Lidar calibration, a variant of ICP is used [4]. The basic idea of the algorithm is to align one geometric shape $S$ to another similar shape, as described in the comprehensive review [5]. The two shapes are referred to as *reference* and *reading* and the objective is to find the transformation matrix with which the best alignment between two shapes is achieved. As is usually the case, a shape $S$ in this context is analogous to one Lidar scan. Let $P^{\mathrm{FL}}$ be a *reading* scan from the Front Left (FL) Lidar and $Q^{\mathrm{FR}}$ a *reference* scan from the Front Right (FR) Lidar. Here,

FL and FR represent the coordinate frames of each Lidar. The first step of the algorithm is to estimate the transformation from FL to FR, $T^{\mathrm{FL}\to\mathrm{FR}}(S)$, by minimizing an error function $error(P, Q)$:

$$\hat{T}^{\mathrm{FL}\to\mathrm{FR}} = \underset{T}{\operatorname{argmin}} \Big( error(T(P^{\mathrm{FL}}), Q^{\mathrm{FR}}) \Big) \tag{1.1}$$

The data points of the two scans are associated using a *matching function*. The simplest version of this method is to associate each point in the *reading* to the closest point in the *reference*, i.e. point-to-point. A variant to this known as point-to-line is described in [4]. Let $M = match(P, Q) = \{(p, q) : p \in P, q \in Q\}$ be the set of matches between $P$ and $Q$, and $d(p, q)$ the distance function between two matching points. The error function can then be described as the sum of all matching distances:

$$error(P, Q) = \sum_{(P,Q)\in M} d(p, q) \tag{1.2}$$

To increase the robustness of the method, each distance is scaled using a weight such that its influence in the error function can be altered. With $W = outlier(M) = \{w(p, q) : \forall (p, q) \in M\}$ being the set of weights and the error function (defined in Equation (1.2)) is augmented to:

$$error(P, Q) = \sum_{(P,Q)\in M} w(p, q)d(p, q) \tag{1.3}$$

With ideal matching, this error would yield the best transformation estimate $T^{\mathrm{FL}\to\mathrm{FR}}$, but, in practice, it is unlikely that the *matching function* provides the best associations at once. However, the idea of ICP is that also an imperfect *matching function* can provide a better estimate of the transformation which in turn can bring better matching. This motivate an iterative process where intermediate transformations ${}^{i+1}T$ are found using:

$${}^{i+1}T \leftarrow \underset{T}{\operatorname{argmin}} \Big( error(T({}^{i}P), Q) \Big) \tag{1.4}$$

where ${}^{i}P$ is a scan gradually transformed using all previous estimates. Finally, the complete transformation is the combination of all intermediary transforms together with the initial guess provided at the start. Nevertheless, the initial transformation is of great importance because if the associations found during the first iteration is not sufficiently good the algorithm may get stuck at local minima. In practice this means that careful measurements has to be made between adjacent Lidars such that the initial transformation can be sufficiently determined.
In order to calibrate the four Lidars in practice, one Lidar is measured by hand such that its coordinates are known in relation to the truck. A scan from this Lidar is then used to generate the *reference* scan to which another Lidar scan can be fit. It is a pairwise procedure where one Lidar, i.e. *reading*, is calibrated to the reference Lidar. Given the physical arrangement, only adjacent Lidars have overlapping field of view and, therefore, the reading Lidar has to become the reference Lidar (after its own calibration) in order to calibrate the remaining ones. This implies that errors

from one calibration can propagate to Lidars later in the calibration sequence, as the position of the reference Lidar is also estimated through calibration. In conclusion, using the ICP method in practice results in multiple drawbacks:

- One Lidar has to be accurately measured by hand and then used as a reference, after which the other Lidars coordinates can be optimized for. If the reference Lidar is badly estimated, the rest of them will also be inaccurate.
- The algorithm need very good initial positions in order to converge.
- The algorithm is prone to converge to local minima.
- The calibration is done using a set of scans, containing only one from each Lidar, and thus it is essential to measure objects in all the common field of views

### 1.2.2 Feature-based calibration

In feature-based calibration methods, certain features are extracted and used as common target points for several sensors. A feature can be a natural or artificial landmark extracted from the Lidar data, usually objects that are distinguishable from the surrounding and perceived similarly from different perspectives. Natural landmarks are typically trees, walls and intersections between walls and floors. Artificial landmarks are often simple geometrical objects added to the environment such as spheres, cylinders, poles or boxes. More elementary shapes are easier to detect and segmented with simpler algorithms using less memory. However, such shapes have to be added to the environment using artificial landmarks.

Plenty of feature-based methods have been used to calibrate various Lidar setups, for example, using boxes [6] and planar boards [7] in combination with cameras. Moreover, in [8] and [9], natural landmarks are used to calibrate 3D Lidars and in [10] poles with reflective tape are used to find translational and rotational offsets of a single 3D Lidar. Additionally in [11] multiple 2D Lidars are calibrated without specific targets but by collecting data while moving in a pre-scanned environment. Feature-based methods are mostly used with sensor setups containing at least one 3D Lidars and/or camera. Very few works have been done with featured-based methods for 2D Lidars, due to the difficulties in searching for, and extracting, features in 2D data. When calibrating 2D Lidars, it is popular to use either supplementary sensors or additional landmarks to bridge the gap between different 2D representations.

In [12], calibration of two 3D Lidars and one 2D Lidar is performed with a moving sphere. A ball is used as target in the mutual field of view to estimate the Lidar positions. Multiple scans are gathered in cases where the ball is moved to different spots. One scan sequence of a moving ball (measured at three locations) is sufficient for determining the relative pose. As 2D Lidars only provide measurements in one plane, finding the center of the ball is slightly different from when using a 3D Lidar. The 2D method can be divided into three steps: 1. segmentation of the laser scan

(finding circular arcs, taking advantage of the fact that any planar section of the ball is a circle), 2. detection of circle and calculation of its center coordinates and radius and 3. calculation of the center of the ball from the known diameter of the ball. Due to the symmetry of the ball it is not possible to know if the center is under or above the scan, which has to be known a priori. The 3D method is using the 2D algorithm repeatedly for every scan layer giving as many estimated centers as there are layers with hits of the ball. The midpoint of the ball is approximated as the mean of all estimations.

Instead of using a sphere, [13] finds the relative pose of two 2D Lidars using two orthogonal planes. The goal is to calibrate all degrees of freedom, i.e. translation as well as rotation, using the following steps: 1. extract two vectors in each target planes (one from each Lidar) from the scans with a least mean square method, 2. compute the normal vector to each plane by cross product of the two vectors and 3. assuming the planes are perpendicular, the dot product between the normal vectors is zero. This way, the Lidar's coordinates can be determined in two dimensions: those which are normal to the target planes. To solve for the remaining degree of freedom the planes are re-positioned and the calibration is performed once more. In the special case when the two Lidar scans are in parallel planes, as in this project, the normal to the target planes cannot be uniquely determined (step 2). This special case was considered in [13] and solved using an alternative least square method. The general method was further extended to handle planes close to, but not perfectly perpendicular by applying non-linear optimization.

The described methods of using a sphere or orthogonal planes has the advantage that it requires no initial guess, i.e. measurement of the relative pose between the Lidars. However, using the method on four Lidars does not allow the calibration of all to be carried out simultaneously. The reason being that this algorithm calibrates the Lidars in a pairwise manner, similar to the ICP method, which results in a sequence of calibrations. Thus, estimation errors will be propagated as one previously calibrated Lidar is used as reference.

### 1.2.3 Occupancy grids and map merging

The fusion of 2D Lidar scans is similar to multi-robot exploration and indoor mapping. Multiple robots can be used to simultaneously explore an environment and the observations of each robot can then be used to construct a more detailed map. Occupancy grids can then be used to pairwise merge maps from several robots [14], [15], [16]. In short, occupancy grids can be described as an image where each pixel indicate whether a part of the map is considered as free space, occupied or unknown. In these articles occupancy grids are visualized as images where each class (free, occupied, unknown) is assigned with a separate colour. Using this representation, maps are merged by moving one map over another until identical parts are aligned. In [14] and [15] a similarity metric $\psi$ is used to find identical regions in the two maps. However, solely using $\psi$ as a heuristic function causes more overlaps of the maps than intended, thus, an additional measure is added to the heuristic

in order to prevent over merging. To solve for the best transformation parameters ($\{x, y, \theta\}$) stochastic optimization techniques are used. In [14] an adaptive random walk is applied where the algorithm manages to merge maps from several robots, as long as there are overlapping observations, but struggles if the overlap is sparse. Instead of a random walk, [15] proposed to use a genetic algorithm to prevent getting stuck in a local optima and to improve the convergence speed. In their work, the crossover and mutation probabilities are changed as the average fitness increases, which also helps to avoid premature or slow convergence.

The scans of 2D Lidars could be considered as separate maps and hence the technique could be a viable alternative to feature-based or ICP calibration methods. However, a single scan of a 2D Lidar generally contains less information than a map and thus such an algorithm is less likely to succeed given a single scan. Furthermore, the heuristic function $\psi$ is designed for quite detailed maps, it is therefore believed to be superfluous when the data is more sparse.

Nevertheless, recent research [9] suggests how occupancy grids can be used to calibrate the yaw angles of multiple 3D Lidars. Four 3D Lidars are mounted on each side of a vehicle such that every Lidar has overlapping field of view with both its neighbors. To calibrate the yaw angle of each Lidar, a scan from each Lidar is fused to form a single 3D point cloud. The 3D point cloud is projected onto the x-y plane and the resulting map is used to form an occupancy grid. The grid is handled as a gray scale image where each pixel is represented by an 8-bit value. 0 represents the highest certainty of an object, 255 is considered as free space and intermediate values indicate less certain assumptions. A genetic algorithm is then used to solve for all four yaw angles. However, the heuristic used is much simpler than in [14] and [15]. By counting the number of pixels whose values are below a threshold, i.e. pixels considered as objects, each fuse can be evaluated. If the yaw angle of each Lidar is more accurate, the fused image concise less pixels considered as objects and thus receives a lower heuristic value. Large overlapping regions of Lidar data guide the search which results in a successful calibration. But, similar to multi-robot map merging, it becomes more difficult when the overlapping of data is less present.

### 1.2.4 Conclusion of literature

In general there are typically three different approaches to calibration of Lidars: ICP, feature-based and occupancy grid representation. Using 2D or 3D Lidars influences the technique in use since the amount of common information varies. Feature-based techniques are more commonly applied when calibrating 3D Lidars as the scanning view and data set is significantly larger, so more intricate objects can be identified. For instance, using a sphere as a calibration target can be very suitable when calibrating 3D Lidars as the object is perceived round from all perspectives. However, using spheres when working with 2D Lidars is not as straight forward as the case with the 3D Lidars, due to the ambiguity in the absolute position and requirement of prior information. In order to make use of features from solely 2D scans simpler shapes can be used such as orthogonal walls. However, if the Lidars are scanning

in parallel planes, to the authors' knowledge, no feature-based methods have been developed.

The ICP methods have been commonly used and tried out in different applications, such as in robot positioning. It benefits from extensive research and documentation but is sensitive to the initial transformation, i.e. measurement between Lidars. With inaccurate measurements, the algorithm may get stuck at local minima and thus not reach an optimal solution. Therefore, the ICP solution is not suitable and development of an alternative method is required.

Occupancy grids based methods originate from a map merging perspective but has recently been used within calibration of Lidars. The work carried out in [9] is found highly relevant and shows promising results applying occupancy grids together with a Genetic Algorithm to calibrate the yaw angles of four 3D Lidars. With one Lidar mounted at each side of a vehicle, the setup and the field of view are closely related to the problem formulated in this thesis. However, the x, y and z coordinates of each Lidar is known in advance and also 3D Lidars rather than 2D are used. 3D Lidars provide point clouds in three dimensions rather than two, and with more information a projection to the x-y plane will result in a more comprehensive 2D representation. The amount of overlapping data between Lidars is thus greater which is beneficial when aligning the grids.

Stochastic optimization techniques are commonly used together with occupancy grids with various heuristic functions. The simple function used in [9] should also be applicable to 2D Lidars. However, the need of elaborate occupancy grids becomes less important when simply counting the number of pixels with greater value than a constant threshold. In contrast to the ICP, this technique allows for simultaneous calibration of multiple sensors and could possibly be extended to include x and y translations. Calibrating four Lidars, with three parameters each, results in a 12-dimensional search space. Therefore, using stochastic optimization algorithms (e.g. a genetic algorithm) is well motivated considering their capability of handling large and complex search spaces.
Regardless of the method used, the quality of the data plays an important role. Laser scans which are too sparse may not provide enough information to achieve convergence and/or may lead to inaccurate solutions.

# 2
# Lidar properties

The Lidars used on the truck are of type *SICK LMS 111*. These build upon an IR sensor and a rotating mirror to transmit light beams which are reflected on surrounding objects. By measuring the time of flight, i.e. the time from the ray being transmitted to the return, the range to surrounding objects is estimated. The range estimation together with the corresponding angle and intensity of reflection make up one measurement. With measurements comprising angle and range the surrounding environment is represented in polar coordinates and intensities may be used to distinguish objects from noise. The measurements cover a sector of 270°. All Lidars used in this project spin with 50 Hz and the angular resolution between the measurements is 0.5°. Chapter 2 includes some theory regarding radial distance and complications which emerge while a Lidar is simultaneously moving and scanning.

## 2.1 Effects of radial distance

The physical distance between two consecutive measurements increase with the range, and this distance is of importance when evaluating a calibration, see Section 3.1.2 below. Assuming that the light beams have an infinitesimal diameter, and measure with a 0.5° angular resolution, the arc length between two consecutive measurements follows from Equation (2.1).

$$L_{arc} = 2\pi r \cdot \frac{0.5}{360} \tag{2.1}$$

By approximating the distance between measurements using the arc length, Figure 2.1 display how the gap between two consecutive measurements are affected by the range. The measurements are marked with red dots and the distance between these two increase with the range. The maximum range of the Lidars are 20 m and, thus, the largest arc length is 0.1754 m.

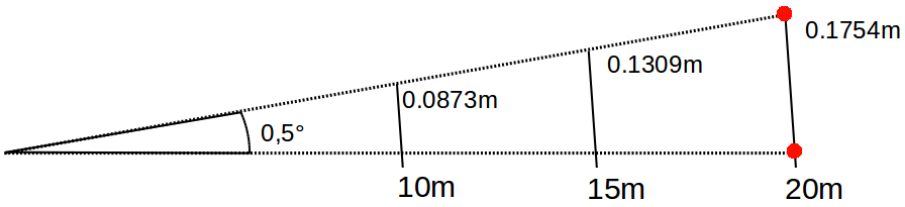



**Figure 2.1:** The distance between Lidar measurements are estimated using the arc length of a circle sector with central angle 0.5°. Thus, the distance between two consecutive measurements varies dependant of the range.

Moreover, if multiple Lidars operate in common, and observe the same object, the distance between measurements become less. As depicted in Figure 2.2, if two adjacent Lidars measure the same object, the distance between measurements can be at most halved the distance, compared to using only one Lidar.

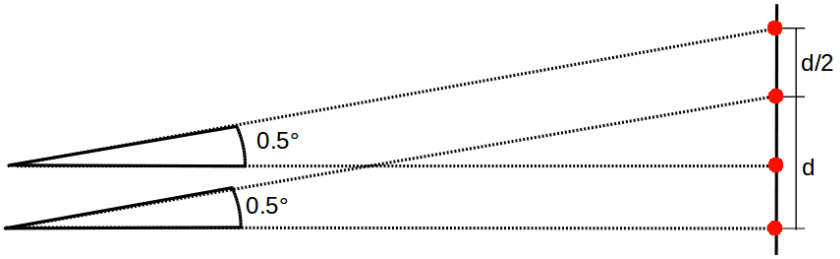



**Figure 2.2:** Distance between measurements for a double Lidar setup. When the same object is observed by two Lidars, measurements overlap and the distance between them is less compared to using just one Lidar.

## 2.2 Moving Lidar complications

Measuring the surrounding environment involves sampling of the continuous time domain. No Lidar scans are performed instantaneously, and therefore subject to distortion if the Lidar is moved during scanning. The speed of a moving truck is relatively large compared to the scanning frequency of the Lidars. Therefore, two types of distortions occur: *timing inconsistencies* caused by a multi Lidar setup and *sweep distortion* of scans due to the spinning motion of each Lidar. Here effects of longitudinal and turning velocities are discussed, but the traversing movements are not considered.

When the truck is turning, the distance to base link affect how far the Lidars are traveling in lateral direction. The **base link** is the vehicle's point of rotation and also the origin of the coordinate system of the truck. The front and rear Lidars are located approximately 6 respectively 3 m ahead of this point. The truck is generally driven at 25 km/h and no more than 80 km/h. A turning rate of 0.1 rad/s is typical for an operating truck while 0.2 rad/s is possible, but requires an aggressive maneuver. These numbers are used to exemplify the magnitude of *timing inconsistencies* and *sweep distortion*.

### 2.2.1 Scan timing inconsistency

As the Lidars operate with a frequency of 50 Hz, one revolution takes 20 ms. In the ideal case all Lidars measure synchronously and deliver their scans at the same time instance, but in reality this is rarely the case. Each Lidar spins continuously, so, theoretically the maximum time difference between all four scans should not exceed 20 ms. However, network latency and CPU priorities could cause longer time windows between scan deliveries so that the complete time difference may increase. These distortions also occur unpredictably and with varying magnitude, so the effect

is irregular.

In order to construct one scene, i.e. one scan from each Lidar, it is generally assumed that all scans represent the surrounding at the same time instance. But, when the truck is moving, scan timing affects how the Lidars perceives the surrounding environment. Figure 2.3 displays a scenario where two Lidars, $L_1$ and $L_2$, observe the same tree but with a time difference of $\Delta t$. $L_1$ first observes the tree at time $t_0$ and later $L_2$ observes the same tree at time $t_0 + \Delta t$. This will cause a problem when scans are about to be merged. Assuming that both Lidars observed the tree at the same time $t_0$, the result will be a fusion indicating two trees instead of one. Neither of the observations are wrong, but assuming both scans are synchronized in time causes the resulting perception to be inaccurate.

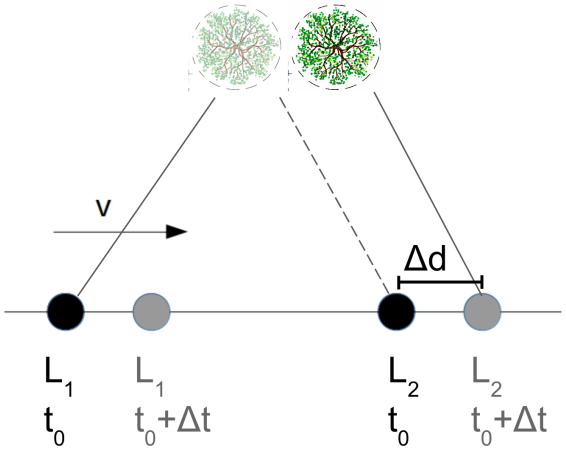



**Figure 2.3:** The moving Lidars $L_1$ and $L_2$ observe the same tree but from two different perspectives, at time $t_0$ and at time $t_0 + \Delta t$. Merging these two observation as if scanned at the same time instance result in a perception including two trees.

This phenomenon is present regardless of which direction the truck is moving, but, the most common movements are forward driving and turning. Therefore, effects of such movements are given in Table 2.1 and 2.2. The synchronization error $\Delta t$ represent possible time differences between the first and last scan in a scene. 20 ms is the theoretical maximum timing error given the Lidar properties, but 50 ms is rather common because of network latency. At slow longitudinal velocities, the difference is considered minor but becomes significant if the truck is driven faster.

**Table 2.1:** Longitudinal distance $\Delta d$ (m) which the Lidars moved due to truck speed and time synchronization error $\Delta t$.

| **Long. dist.** $\Delta d$ (m) | | Long. velocity $v_T$ (km/h) | | |
|---|---|---|---|---|
| | | 5 | 25 | 75 |
| $\Delta t$ (ms) | 20 | 0.0278 | 0.1389 | 0.4167 |
| | 50 | 0.0694 | 0.3472 | 1.0417 |

As the Lidars are located with an offset to the base link, there are also lateral shifts

when turning. Examples of lateral movements, withing one scene, are presented in Table 2.2.

**Table 2.2:** Lateral distance $\Delta d$ (m) which the Lidars moved due to truck turn rate and time synchronization error $\Delta t$. 6 and 3 m is the longitudinal distance from the front and rear lidars to base link.

| **Lateral dist.** $\Delta d$ (m) | | | | Turn rate $\omega_T$ (rad/s) 0.05 | 0.10 | 0.20 |
|---|---|---|---|---|---|---|
| Long. dist. to base link (m) | 3 | $\Delta t$ (ms) | 20 | 0.0030 | 0.0060 | 0.0120 |
| | | | 50 | 0.0075 | 0.0150 | 0.0300 |
| | 6 | $\Delta t$ (ms) | 20 | 0.0060 | 0.0120 | 0.0240 |
| | | | 50 | 0.0150 | 0.0300 | 0.0600 |

The lateral distances, presented in Table 2.2, are not as significant as the longitudinal affects given in Table 2.1. Nonetheless, timing inconsistencies are always present during driving and may inflict misinterpretations despite accurate measurements.

### 2.2.2 Lidar sweep distortion

If the truck moves during the scanning period, the location of the Lidar will change between the individual measurements, thus introducing sweep distortions as described in [17]. If the Lidar is stationary during the scan, the light rays will form a pure circular pattern, as illustrated with blue in Figure 2.4 **(a)**. But if the Lidar is moving, and observed from a stationary point, the Lidar rays will instead form a skewed curve, as illustrated with red in Figure 2.4 **(a)**.


**(a)** **(b)**

**Figure 2.4:** Lidar sweep distortion. Blue curves represent a sweeping light ray from a stationary Lidar. In panel **(a)**, the red curve represent the Lidar light ray trajectory as observed from a stationary point. In panel **(b)**, the red curve represent the sweep as perceived in the coordinate system of the moving Lidar.

After the light ray has rotated 270°, the measurements are collected and grouped into a scan marked with the time of the first measurement, i.e ray angle 0°. Still, all

but the first measurements occurred after this time. Thus, if the Lidar is moving, the perception of the surroundings is incorrect if assuming that the sweep was gathered instantaneously. Figure 2.4 **(b)** demonstrate an example where a Lidar was moving during a sweep. The red curve represent the sweep as perceived in the coordinate system of the moving Lidar. This is also the information delivered by the Lidar, so, if it is assumed stationary, the measurements are inaccurate. The magnitude of the sweep distortion is dependant on the velocity of the truck and angular difference from the first ray. Table 2.3 show the longitudinal movement of a Lidar between the first and last ray. All intermediate rays are thus subject to less distortion.

**Table 2.3:** Longitudinal distance $\Delta d$ (m) a Lidar moved over one complete scan with constant truck velocity. The angular span of the Lidar is 270°, but, all intermediate rays are subject to less distortion than the last ray, as, the time difference is proportional to the angle of each ray.

| **Long. dist.** $\Delta d$ (m) | | Long. velocity $v_T$ (km/h) | | |
|---|---|---|---|---|
| | | 5 | 25 | 75 |
| Ray angle | 270° | 0.0208 | 0.1042 | 0.3125 |

The largest source of sweep distortion is due to the longitudinal velocity, but as the truck is turning, angular turn rate introduces lateral distortions as well. The lateral distance a Lidar moves for different turning rates are given in Table 2.4.

**Table 2.4:** Lateral distance $\Delta d$ (m) which the Lidars moved due to truck turn rate and angle increment of Lidar ray during one sweep. 270° is the Lidar scan range, 6 and 3 m is the longitudinal distance from the front and rear Lidars to base link.

| **Lateral dist.** $\Delta d$ (m) | | | | Turn rate $\omega_T$ (rad/s) | | |
|---|---|---|---|---|---|---|
| | | | | 0.05 | 0.10 | 0.20 |
| Long. dist. to base link (m) | 3 | Ray angle | 270° | 0.0023 | 0.0045 | 0.0090 |
| | 6 | | 270° | 0.0045 | 0.0090 | 0.0180 |

Similar to scan timing inconsistency the Lidar sweep distortions depends on both the longitudinal velocity and turning rate. However, longitudinal distortions, presented in Table 2.3, are much greater than distortions generated if the truck is turning, given in table Table 2.4. A longitudinal velocity of just 5 km/h cause more distortion than a hard turn of 0.20 rad/s. Furthermore, sweep distortion is an effect of the fundamental properties of Lidars and therefore inflict on every individual scan.

# 3
# Methods

In order to produce a clear single point cloud using multiple Lidars, the $x$, $y$ and yaw coordinates of each Lidar must be known. The coordinates are used to transform measurements from each Lidar to be resembled from a common perspective, the base link. To solve for the respective positions, the proposed method does not require any artificial objects or external measurements of the environment. Instead it relies on alignment of mutual observations which does however assume Lidar hits in common field of views. With a set of four Lidars, the goal of the calibration is to estimate a total of twelve parameters. The proposed method simultaneously calibrates all four Lidars, therefore, the problem's solution space is large. Given this, and the nonlinear nature of rotational transformations, a genetic algorithm is a suitable choice for this problem.

## 3.1 Genetic Algorithm

Genetic algorithm (GA) is a type of Evolutionary Algorithm which, as its name indicate, is inspired by Charles Darwin's theory of evolution. Therefore, most terminologies used in GAs stems from biology but the implications are greatly simplified compared to their original counterparts. As described in [18], the basic GA builds upon the six phases *initializing a population*, *fitness evaluation*, *selection*, *crossover*, *mutation* and *elitism*. These reflect the process of natural selection where well suited **individuals**, i.e. a candidate solution, get to produce offspring which form the next generation. The process can also be described as *survival of the fittest* because more fit individuals have a greater chance of having offspring. By iterating such a process better individuals will be found and the concept is used to search for optimal solutions.

In this project, the overall idea of the algorithm is to simultaneously calibrate all four Lidars. Therefore, all twelve parameters are encoded in a single individual using real-number encoding (see [18]). Real-number representation allows for unrestricted ranges as no decoding function is required. Each **gene**, a unit of an individual, therefore directly represents the $x$, $y$ or yaw coordinate of a Lidar. The 12 genes form a **chromosome**, which is the complete information contained by an individual, see Figure 3.1.


**Figure 3.1:** Chromosome layout. The $x$, $y$ and yaw coordinates of all four Lidars are represented using real-number encoding.

The algorithm starts by initializing the first set of individuals called a **population**. The size of the population is kept constant throughout the calibration and set to contain 50 individuals, as proposed by [18]. Each individual's *fitness* is evaluated by determining how good the scans align when transformed according to the chromosome. After evaluation, *selection*, *crossover*, *mutation* and *elitism* are performed to build the next generation of individuals. The process in then repeated until a set number of generations is reached.

### 3.1.1 Initial population

The initial population should include a diverse set of chromosomes such that various solutions are considered and evaluated. It is, however, not necessary to include coordinates and angles far different to an intuitive guess. A measurement accuracy is used to bound the gene values. For each chromosome, every gene is drawn from a uniform distribution in the following way:

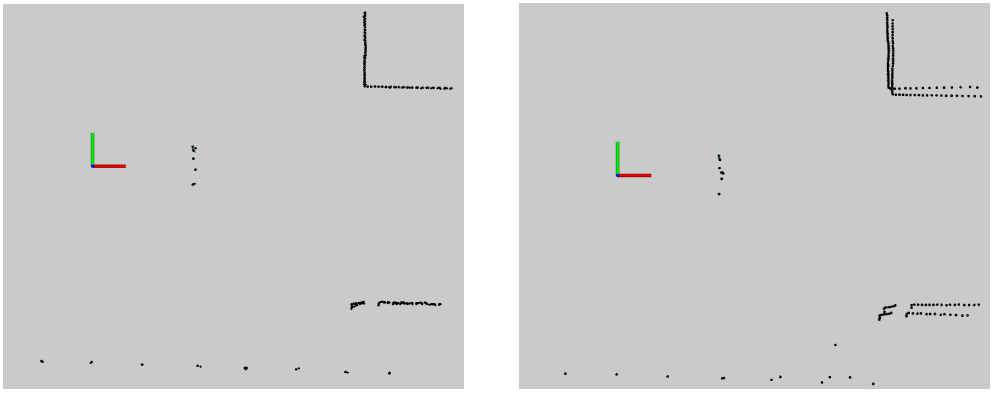
$$\text{Gene}_i = \textbf{Initial guess}_i + \text{Meas Accuracy} \cdot r\,, \quad i = 1{:}12,\ r \in [-1{:}1] \tag{3.1}$$

Where **Initial guess** is a set of starting values (in the form of a chromosome) which estimates the 12 Lidar coordinates. *Meas Accuracy* is the expected error of the coordinate system of each Lidar (typically measured by hand), and $r$ is a random number uniformly drawn between -1 and 1. As the yaw angles and x-y coordinates are of separate ranges the measurement accuracy is set accordingly. This procedure is repeated until a complete population is filled.

### 3.1.2 Fitness evaluation

In order to compare the individuals of a population one use a fitness measure. If an individual is considered to be favourable it receive a high fitness score and vice versa. The fitness score determines the probability for an individual to produce offspring for the next generation. However, the fitness function varies depending on the problem and often there are multiple viable alternatives. It is a very central part of the algorithm and should be chosen carefully.

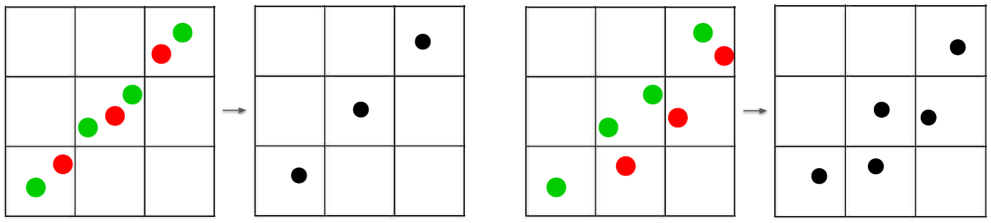
In this project, the genes of a chromosome are used to transform a scan from each Lidar's local coordinate system to a system using the base link as its origin, as in Figure 1.1. The resulting four point clouds are then concatenated into one single cloud used to evaluate the chromosome. In this way, the fitness evaluation can measure how good the scans align after transformation. Figure 3.2 display the concatenation of the FL and FR Lidar with two different chromosomes. In panel **(a)** a good chromosome is used and in panel **(b)** a bad one.



**(a)** **(b)**

**Figure 3.2:** Concatenation of point clouds measured by the FL and the FR Lidars. The quality of the chromosome determines how well the scans align. In panel **(a)** a good chromosome is used and in panel **(b)** a bad one.

The fitness measure is based on the idea that hits of the same objects coincide if the Lidars are well calibrated, as in Figure 3.2 **(a)**. To do this, the fused point cloud is downsampled by removing points in regions where the data is dense. Thus, good alignment results in a greater reduction of points and the chromosome is rewarded with a higher fitness score. Downsampling in the algorithm is performed by applying a filter which replace closely placed points by their centroid. Figure 3.3 display a conceptional example where two scans are downsampled using a square grid. Points from two different scans are coloured with red and green. After filtering the result is presented with black points. How well the scans align determine how many points are removed by the filter.



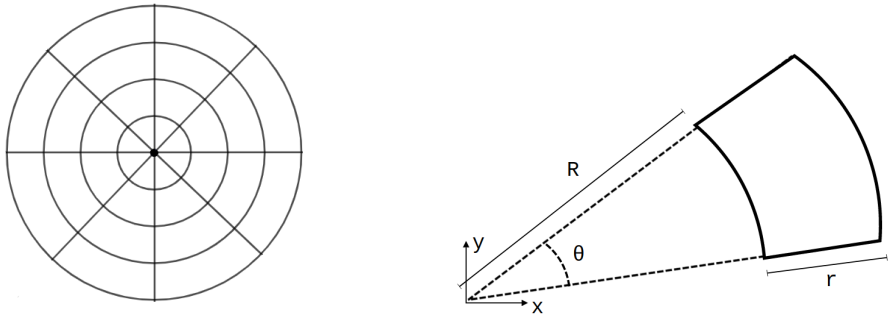
**(a)** **(b)**

**Figure 3.3:** A conceptual fitness evaluation example. A square grid filter is used to replace multiple points in a cell by their centroid and the fitness is equal to the number of removed points. In panel **(a)**, two scans are well aligning and the fitness is equal to 4. In panel **(b)** the scans align badly and the fitness is two.

The example of Figure 3.3 is used to display the general idea behind the fitness evaluation. The fitness score is simply equal to the number of removed points. However, in this project, a circular filter grid is used to downsample the merge of scans.

#### 3.1.2.1 Circular filter grid

The filter grid used to evaluate fitness is defined in polar coordinates by radial width r, angle $\theta$ and radius R as in Figure 3.4. Each cell is a segment of a circle sector and, therefore, the cell area increase with the radius R. As described in Section 2.1, more distant Lidar measurements are inherently further apart than close ones. A circular grid allow fitness evaluation to be geometrically suitable for points both close to and far from the truck as the cell area varies proportionally to the distance from the truck.




**(a)** **(b)**

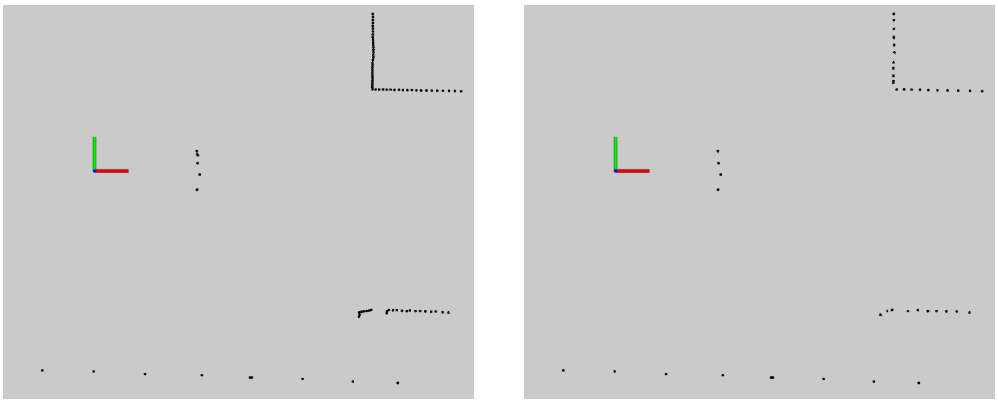
**Figure 3.4:** A circular grid filter is used to evaluate each chromosome, displayed in panel **(a)**. Each cell size is defined using an angle $\theta$ and width r, displayed in panel **(b)**. To evaluate each chromosome, points within each cell are replaced by their centroid and the fitness score is equal to the total number of removed points.

The grid origo is set to coincide with the centroid of the four Lidar positions. This is not ideal for any Lidar because the origo of the filter do not coincide with any Lidar, but, in a geometric perspective it is equal for all of them.

#### 3.1.2.2 Sparsify input data

The result of the filtering depends on the size of the observed objects. Big objects such as walls, see top right corner panel **(a)** in Figure 3.5, result in scans with dense clusters of points. When evaluating such a scan, close measurements inherently yield a higher fitness even if there are no matching hits from another Lidar. Reversely, small objects such as scattered poles, bottom part of panel **(a)** in Figure 3.5, result in sparse or even single hits and is therefore only removed if aligned with points from another Lidar. Thus, the maximum number of points that can possibly be reduced when filtering small objects only depend on how many Lidars observed the object.

Before concatenating the point clouds from each Lidar, a preparatory filter is applied to reduce the number of points in dense clusters. This filter is similar to the one used during evaluation. By downsampling each Lidar's separate scans, hits of smaller object are unaffected given that the filter cells are small enough. In Figure 3.5, a scan from the FR Lidar is displayed. The bottom hits of both panels are unaffected by filtering, but, the measurements of walls are sparsified.



**(a)** **(b)**

**Figure 3.5:** The scans from each Lidar are sparsified to avoid increased fitness due to dense measurements. Panel **(a)** includes a raw scan from the FR Lidar and panel **(b)** show the same scan, but sparsified. Sparse objects are unaffected and thus maintain the same potential fitness.

The sparsified point clouds, one from each Lidar, are then concatenated before evaluation. This ensures that big objects which would have had higher fitness mainly due to denser cluster, are instead rewarded with higher fitness only if they are aligned with another Lidar point cloud. In order for this to work in symbiosis with the evaluation, the size of the cells follow the same grid size used for evaluation. In other words, the evaluation procedure can be summarized in four step:

1. Transform each Lidar's scan to the coordinate system of the vehicle.
2. Sparsify each transformed scan.
3. Merge all sparsified scans.
4. Evaluate the fitness using the merge of scans.

#### 3.1.2.3 Grid size reduction

When evaluating a chromosome using a grid filter, see section 3.1.2, the result is highly dependant on the size of the cells. Large cells are likely to encompass many points resulting in a large reduction of points, and vice versa using a tighter grid. Moreover, big cells allow for an exploratory search as only major changes of a chromosome alter the fitness score. On the contrary, smaller cells can reveal if small changes are beneficial and, thus, useful to achieve better precision. This calls for a decrease in the grid size during optimization to allow both exploration and precision.

The grid size is reduced linearly with number of generations. It is a straightforward but perhaps reckless approach as the performance of the population is not taken into consideration. If the grid size is decreased too quickly, the algorithm is prone to get stuck with incorrect transformations and the final result is incorrect. However, it is intuitive and easy to alter the rate of reduction by simply changing the number

of generations. With a varying grid size, additional tuning parameters emerge and make it harder to use. Even so, the initial and maximum grid size is rather easy to tune since the only requirement is to allow higher chance of exploring the search space in the initial phase of the optimization. However, one must be careful in setting the final and simultaneously minimum grid size.
The smallest effective grid size is closely connected to the fundamental properties of Lidars. As described in Section 2.1, the distance between consecutive measurements for multiple Lidars are at most half the distance compared to a single scan. The Lidars measure with an angular precision of 0.5°, thus the smallest angular distance between points is 0.25° which is also used as the smallest angle of the grid cells. The radial measurement accuracy of the Lidars are ±3 cm, and the radial minimum cell width is set to 0.025 m.

### 3.1.3 Selection

In the selection phase it is determined which individuals get to transfer their genes to the next generation. There exist different selection methods, but in this thesis roulette wheel selection (RWS) is used. In RWS, each individual is selected with a probability proportional to its fitness. This way, the absolute fitness scores are accounted for. Each individual retrieve a probability of being selected proportional to its fitness. Since the assigned probabilities are proportional to an individual's fitness, the chance of an individual being selected increases if it has a higher fitness (i.e. better alignment). The selection procedure is repated until a new generation is filled under the constraint that the size of the population is kept constant.

A detailed description of RWS is given in [18] and here follow an explanatory example using four individuals A, B, C and D with fitness $f_A > f_B > f_C > f_D$. The selection probability may therefore be as in Figure 3.6.

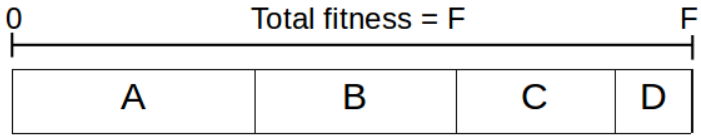



**Figure 3.6:** Proportionate selection example using roulette wheel selection. The fitness distribution is $f_A > f_B > f_C > f_D$.

The probability of selecting an individual thus follow the formula:

$$P_{ind} = \frac{f_{ind}}{\sum_{j=1}^{N} f_j} \quad ind = \{A, B, C, D\},\ N = \text{number of individuals} \tag{3.2}$$

Where $f$ is the fitness of an individual and $P_{ind}$ is the probability of selecting an individual.

### 3.1.4 Crossover

The idea with crossover is to mix the genes of two individuals causing new offspring, as in biological mating. Crossover of two selected individuals (Section 3.1.3), is

performed by randomly selecting a crossover point, allowing the two chromosomes to be divided and then assembled as in Figure 3.7.

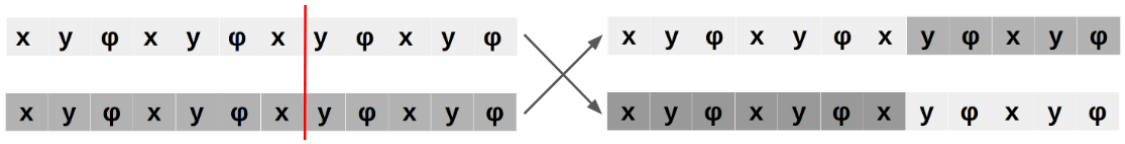



**Figure 3.7:** Example of one point crossover. A randomly selected crossover point is used to cross two individuals resulting in new chromosomes.

Crossover is a powerful tool to generate new solutions and, thus, provide efficient exploration of the search space. However, if the diversity of the population is low it is unlikely that the crossover further help the optimization progress. Mating of two similar individuals results in closely related offsprings. To control its influence, the crossover is performed with a probability of $P_{cross}$. More advanced alternatives exists such as two-point crossover or non length preserving crossovers, but they are not considered in this work.

### 3.1.5 Mutation

As a new offspring is produced, crossed or not, all of its genes are subject to mutation. Mutation of a gene is performed with a probability of $P_{mutate}$. Mutation adds diversity to the next generation but the mutation rate $P_{mutate}$ should be kept low because otherwise the search becomes more random than evolutionary.

Using real-number encoding, the mutation procedure is performed using creep mutation. If the gene is subject to mutation, the new value is drawn from a uniform distribution centered around the original gene value. The interval of the mutation span is set using a creep rate. By randomizing a number $r$, between -1 and 1, the new value of the gene is formed using:

$$\text{Gene}_{\text{mutated}} = \text{Gene}_{\text{original}} + \text{CreepRate} \cdot r\,, \quad r \in [-1:1] \tag{3.3}$$

Mutation introduces new genetic material into the population which is important to maintain the diversity of the population for better exploration of the search space. Nevertheless, the creep rate influences the potential fitness variation. With a large evaluation grid the fitness can only be stimulated using a big mutation creep. To refine the solution, and achieve good accuracy, a more subtle creep is better. Therefore, the creep rate (the range in which the new gene values are defined) is reduced linearly with increasing generations analogous to the grid size reduction. This, however, does not affect the probability of a gene to be mutated. By coupling the creep rate to generational count tuning parameters are kept to a minimum. Nonetheless, starting and final mutation creep must be determined. In this work, the starting creep rate is set to a fraction of the measurement accuracy in order to avoid excessive boundary hits and the final creep rate is determined through trial-and-error.

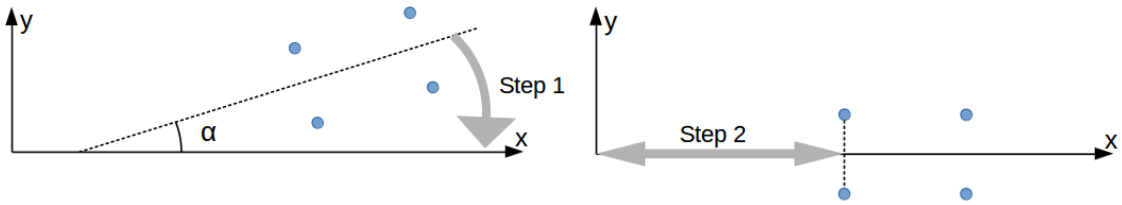
Moreover, using a real-number representation, together with creep mutation, the genes may ultimately take any value. The solution could drift from the base link without changing the relative positions as the evaluation only relies on alignment of points. In order to avoid drifting, the chromosomes are kept within a reasonable range by restricting the gene values. If a gene is mutated outside the range specified in Section 3.1.1, it is instead set to the border value. Using such a restriction the search space is limited and each Lidar's coordinates are kept within a virtual box.

### 3.1.6 Elitism

After a new population has been formed using *selection*, *crossover* and *mutation* the best individual of the previous generation may be lost. It is likely to have been selected, but if so possibly crossed or mutated. In order to preserve the best individual, it is stored away and inserted into the new generation, a procedure called *elitism*.

## 3.2 Absolute positioning relative to truck

Even with a high quality calibration the absolute positions relative to the truck is not uniquely determined. With the proposed method only the relative poses between the Lidars are estimated, leaving three undetermined degrees of freedom (x, y, $\varphi$) of the complete Lidar setup. Both front and rear Lidars are assumed to be mounted symmetrically with respect to the longitudinal center of the truck. Adjusting the Lidar setup to the coordinate system of the truck is done in two steps, 1) rotation and 2) longitudinal translation. Figure 3.8 display the point of rotation, angle of adjustment and the translation adjustment of a skewed Lidar setup.




**(a)** Positioning adjustment step 1. **(b)** Positioning adjustment step 2.

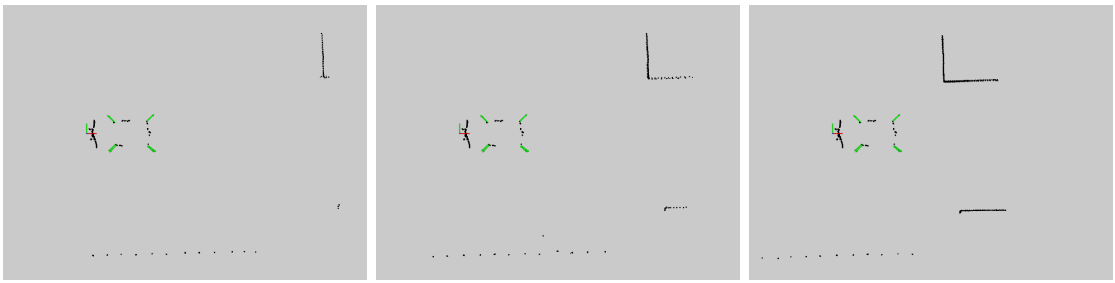
**Figure 3.8:** Absolute positioning adjustment of four Lidars displayed as blue points. Step 1: Rotate the Lidar setup such that the center line of the Lidars coincide with the longitudinal center of the truck. Step 2: The Lidar setup is adjusted in longitudinal direction to match a measured distance.

When rotating the Lidar setup (step 1), the yaw angle ($\varphi$) of each Lidar must be adjusted to avoid introducing angular distortions. If the complete setup is rotated clockwise with the angle $\alpha$, each Lidar's yaw angle becomes $\varphi - \alpha$. Shifting the complete setup in this way ensures that the relative poses are preserved in all coordinates x, y and $\varphi$. This procedure is performed to all chromosomes in each generation so that the setup does deviate in position relative to the truck.

## 3.3 Multiple scene evaluation

Calibrating the Lidars using one scan from each Lidar, i.e. a scene, may not provide sufficient information for the calibration to converge. If the data is sparse, such as large regions without hits, not even multiple Lidars may provide any mutual information, which in turn makes the calibration very difficult. To accommodate for cases with less data, multiple scenes from various time instances can be included such that it is more likely to have hits from multiple Lidars in the common field of views. By gathering a set of scenes, all/a subset of them can be used during evaluation. Thus, the fitness evaluation needs to be modified to account for multiple scenes, see below.

The evaluation step, see Section 3.1.2, is extended to include a set of scenes where each of them provide a fitness score. The fitness from all evaluated scenes are then summed such that the new fitness score represent how well a chromosome accommodate to all of the applied scenes. In Figure 3.9, an example of three different scenes are displayed. The truck is moving forward and, thus, the object visible to the front Lidars in Figure 3.9a, will eventually end up in the common field of view of either of the rear Lidars. One should note that the execution time to evaluate each chromosome is highly dependant on the number of evaluated scenes.



**(a)** Scene 1. **(b)** Scene 2. **(c)** Scene 3.

**Figure 3.9:** A set of Lidar scenes are used to evaluate each chromosome to increase robustness. With multiple scenes it is more likely that all Lidars provide measurements in mutual field of views so that points can align.

## 3.4 Velocity compensation

As mentioned in Section 2.2, two types of complications occur when dealing with moving Lidars: scan timing inconsistency and sweep distortion. Two approaches have been used to compensate for such effects. Scan timing compensation is performed as a Lidar position modification, where the coordinates of each Lidar are altered to better represent the true position of the Lidar at the time the scan is performed. Unlike scan timing compensation, sweep skew compensation is a point cloud modification, where individual points in the scanned point cloud are moved to represent the surroundings more accurately. In order to simplify the equations, each Lidar is referred to by their location on the truck as Front Left (FL), Front

Right (FR), Rear Left (RL) and Rear Right (RR). The subscript $L$ represent any of the Lidars.

$$L = \{FL, FR, RL, RR\}$$

### 3.4.1 Scan timing compensation

It is difficult to assure that all Lidar spin synchronously and deliver their scans at the same time instant in real-time systems. As demonstrated in Section 2.2.1 there may be major deviations of the Lidars positions if the truck is moving while gathering scans to construct a scene. The longitudinal velocity $v_T$ and turn rate $\omega_T$ of the truck is obtained using external IMU measurements. Each scan also holds the time-stamp of the first ray, making it possible to adjust the scans in such a way that they were obtained at the same time instant. There are two adjustments to account for: the longitudinal speed of the truck and the movement generated if the truck is turning.

Each chromosome holds the coordinates of each Lidar used to transform the scans as if measured from the base link. Section 3.1.2 describes the evaluation procedure with the assumption that the relativity of all coordinates are static which is incorrect once the truck is moving and the scans are obtained at slightly different times. Therefore, each Lidar coordinates are adjusted (before evaluation) by including a timing compensation in the transformation from each Lidar to the base link. Thus, the chromosomes remain unmodified but will be evaluated in the correct way despite timing inconsistencies. Here, all scans are adjusted to the time of the FL Lidar scan. The time difference $\Delta t_L$ for each Lidar is the difference between the FL Lidar scan time $t_{FL}$ and the respective Lidar scan time $t_L$.

$$\Delta t_L = t_{FL} - t_L \tag{3.4}$$

#### 3.4.1.1 Longitudinal movement compensation

The coordinate system of the truck is set to have its x-axis in the longitudinal direction. Here, only such movements are considered and thus only the x-coordinate needs to be compensated. The new coordinate $x'_L$ is the actual position $x_L$ minus the distance traveled since the FL Lidar delivered its scan.

$$x'_L = x_L - \Delta t_L \cdot v_T \tag{3.5}$$

#### 3.4.1.2 Turning compensation

All Lidars are located with an offset to the point of rotation, i.e. the base link. So when the truck is turning, each Lidars movement can be represented as a rotation around the base link. Using the angular velocity of the truck $\omega_T$ the Lidar coordinates, $x_L$, $y_L$ and $\varphi_L$, can be compensated accordingly:

$$\theta_L = \omega_T \cdot \Delta t_L \tag{3.6}$$

$$\begin{bmatrix} x'_L \\ y'_L \end{bmatrix} = \begin{bmatrix} cos\,\theta_L & -sin\,\theta_L \\ sin\,\theta_L & cos\,\theta_L \end{bmatrix} \cdot \begin{bmatrix} x_L \\ y_L \end{bmatrix} \tag{3.7}$$

$$\varphi'_L = \varphi_L - \theta_L \tag{3.8}$$

where $\theta_L$ is the angular change of each Lidar with base link as the point of rotation.

### 3.4.2 Sweep skew compensation

Measuring the surroundings with Lidars involve sampling in the continuous time domain. A complete scan of 270° takes 15 ms and if the Lidar is moving during this time the perception of the surrounding environment is skewed. As discussed in Section 2.2.2 both longitudinal and lateral movements cause skewing of scans. Table 2.4 displays the lateral movement of a Lidar between the first and last ray and there is generally no more than 0.018 m of lateral movement. Therefore, distortions generated during turning are considered as minor and, consequently, there is no compensation for lateral movements. Longitudinal movements are, however, taken into consideration and compensated for using the following method.

In order to observe 360° of the surrounding environment, all Lidars are mounted to point in different directions. Figure 3.10 is a schematic representation of a truck showing how the coordinate systems of each Lidar is pointing in relation to the truck. As depicted, none of the Lidars coordinate system align with the coordinate system of the truck. So if the truck is moving forwards, each Lidar is traveling in both x and y direction relative to its own frame of reference. Therefore, sweep skew compensation is a point cloud modification, where individual points in the point cloud is moved in longitudinal direction to better represent the scanned surroundings.

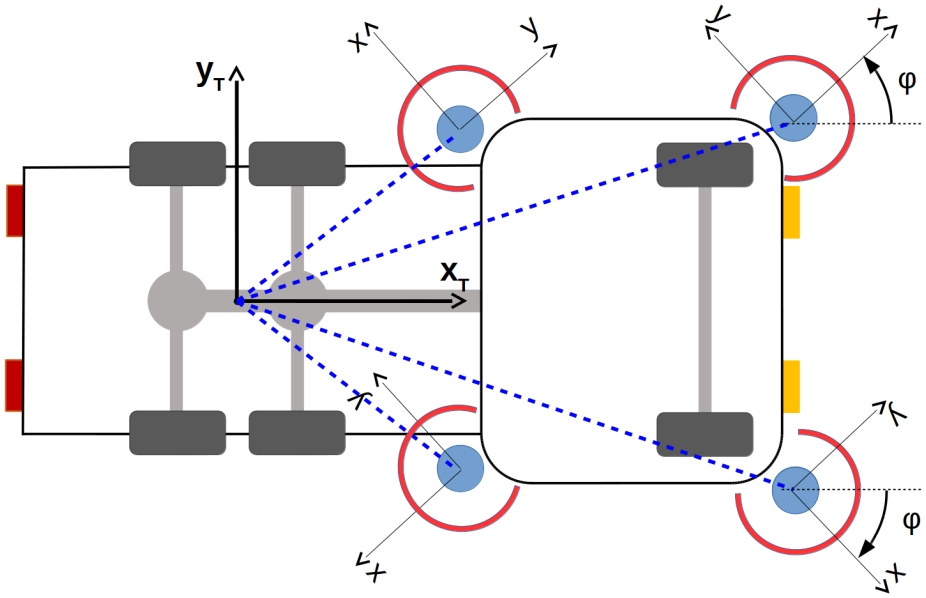



**Figure 3.10:** An illustration of a tractor unit equipped with four 2D Lidars. All Lidars point in different directions and thus longitudinal movements of the truck result in both x and y movements if observed from either of the Lidars coordinate systems.

In order to compensate for longitudinal movements, the longitudinal velocity of the

truck $v_T$ is decomposed into each Lidars coordinate system as $v_{x,L}$ and $v_{y,L}$, see Equation (3.9). The velocity $v_T$ is obtained using external IMU measurements and considered constant over one sweep. $\varphi_L$ is the yaw angle of each Lidar. However, the yaw angles are unknown but estimated during calibration. Therefore, the decomposition makes use of the current best estimates of $\varphi_L$ once calibration is run.

$$\begin{aligned} v_{x,L} &= v_T \cdot cos(\varphi_L) \\ v_{y,L} &= v_T \cdot sin(\varphi_L) \end{aligned} \tag{3.9}$$

The point cloud used for compensation is defined in Cartesian coordinates using each Lidars coordinate system. Each point in a scan corresponds to a ray angle $\alpha$ which represents the angle from the first measurement. The ray angle $\alpha$ of each point ($p_{x,L}$, $p_{y,L}$) is calculated using the equation

$$\alpha = atan2(p_{y,L},\, p_{x,L}) - \alpha_0, \tag{3.10}$$

where the function $atan2(y, x)$ returns the principal value of $arctan(y/x)$ taking the signs of $x$ and $y$ into consideration in order to determine the quadrant. The constant $\alpha_0$ shift the ray angels from [-135°, 135°] to be in range [0°, 270°]. The Lidar ray rotates with a constant speed meaning that the time stamp of each point ($p_{x,L}$, $p_{y,L}$) is proportional to the ray angle $\alpha$. It takes $T$ seconds for the ray to complete one revolution so the time of each point $t_\alpha$ is calculated as a fraction of one revolution, as shown in Equation (3.11).

$$t_\alpha = \frac{\alpha}{2\pi} \cdot T \tag{3.11}$$

Longitudinal compensation of each point ($p_{x,L}$, $p_{y,L}$) is performed using Equation (3.12). The compensation is proportional to the velocity in x and y directions in the frame of the Lidar, and the time from first scan to the current ray, $t_\alpha$. The updated point ($p'_{x,L}$, $p'_{y,L}$) is compensated for the longitudinal sweep distortion.

$$\begin{aligned} p'_{x,L} &= p_{x,L} + v_{x,L} \cdot t_\alpha \\ p'_{y,L} &= p_{y,L} + v_{y,L} \cdot t_\alpha \end{aligned} \tag{3.12}$$

The compensation is applied to all points in each point cloud delivered by the Lidars. The procedure is shown in Algorithm 1.

**Algorithm 1** Longitudinal sweep compensation

$v_T$ = velocity of truck
**for** L = {FL, FR, RL, RR} **do**
  $\varphi_L$ = yaw angle of Lidar
  $v_{x,L} = v_T \cdot cos(\varphi_L)$
  $v_{y,L} = v_T \cdot sin(\varphi_L)$
  **for** $p_i$ = {all points in Point Cloud$_L$} **do**
    $p_{x,L}$ = x-coordinate of point $p_i$
    $p_{y,L}$ = y-coordinate of point $p_i$
    $\alpha = atan2(p_{y,L}, p_{x,L}) - \alpha_0$
    $t_\alpha = \frac{\alpha}{2\pi} \cdot T$
    $p_{x,L} = p_{x,L} + v_{x,L} \cdot t_\alpha$
    $p_{y,L} = p_{y,L} + v_{y,L} \cdot t_\alpha$
  **end for**
**end for**

# 4

# Results

In this chapter a statistical robustness test of the genetic algorithm (GA) and the ICP is presented and also the results of applying the GA to a real truck. Lastly, the effects of velocity compensation is considered and examined visually.

## 4.1 Robustness of ICP and GA calibration

In order to evaluate the robustness of the GA and ICP algorithm, 200 initial guesses were tried on both algorithms and the results were analyzed. **Initial guess** is a set of starting values (in the form of a chromosome) which estimates the 12 Lidar coordinates. Each output, i.e. x, y and yaw coordinates for each Lidar, was checked by visually examining the merged Lidar scans to determine the correctness of the calibration. The tests were performed on three different cases with no or minor truck velocity to see how the algorithms performed on different sets of data without distortions due to movement. Both algorithms were run with settings selected to maximize their performances. Whenever possible, the GA calibration was performed using multiple scenes but due to its architecture the ICP is always restricted to one. The initial x, y and yaw coordinates were drawn from uniform distributions as in Equation (4.1).

$$Coordinate_i = mean_i + width \cdot r\,, \quad i = 1{:}12,\ r \in [-1{:}1] \tag{4.1}$$

The *width* of the distribution was first set to 0.1 m for the x and y coordinates and 7.5° for all yaw angles. A second initialization was also used with *width* set to 0.05 m for x and y coordinates and 3.75° for all yaw angles. The *mean* values were set to the corresponding coordinates of a previously calibrated truck. This way, two sets, namely $S_{wide}$ and $S_{narrow}$ including 100 coordinate collections each, were created to replicate a wide range of feasible guesses as if a new truck would be calibrated with the knowledge of an existing one. The ground truth is considered close to the *mean*, though no absolute ground truth exist as the existing calibration is also an estimation. Figure 4.1 displays 12 histograms of the coordinate sets generated with $S_{wide} = \{width_{xy} = 0.1$ m, $width_{\varphi} = 7.5°\}$ and $S_{narrow} = \{width_{xy} = 0.05$ m, $width_{\varphi} = 3.75°\}$ and it is notable how the data follow uniform distributions.

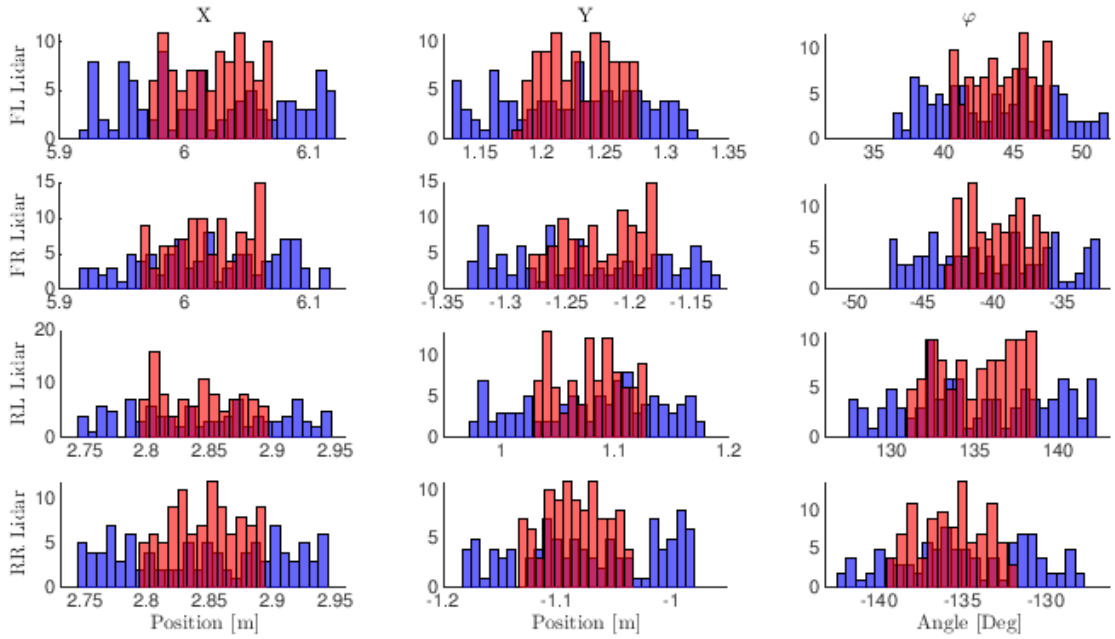
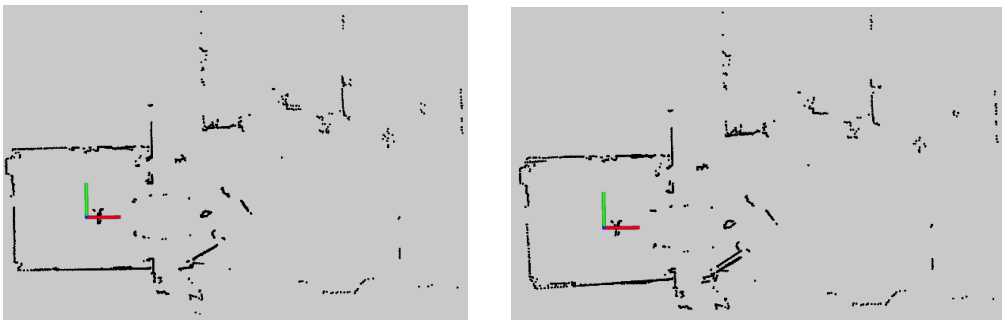



**Figure 4.1:** Input data. 100 initial guesses, coordinates of each variable was drawn from two uniform distributions with mean of an existing calibration. Blue histograms are variables drawn using $width_{xy} = 0.1$ m and $width_{\varphi} = 7.5°$. Red histogram display a set using half the spread: $width_{xy} = 0.05$ m and $width_{\varphi} = 3.75°$.

Both coordinate sets $S_{wide}$ and $S_{narrow}$ are run on both the ICP and the GA for three different cases. The resulting calibrations were visually examined and judged similar to the examples of panel **(a)** and **(b)** in Figure 4.2.

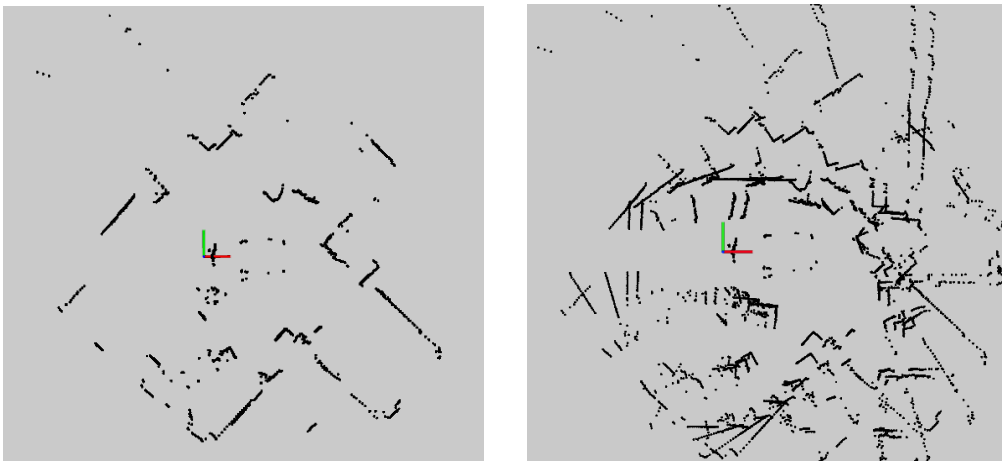


**(a)** Case 1. Good calibration. **(b)** Case 1. Bad calibration.

**Figure 4.2:** Case 1. The truck is parked in a garage facing rightwards with its rear section surrounded by walls. Panel **(a)** display a good calibration where points from all Lidars align which is not the case in panel **(b)**.

Case 1 includes a single scan from each Lidar where the truck is parked inside a garage. This indoor environment provides plenty of data points with overlap in all common field of views. Figure 4.2 displays the measurements from all four Lidars. The truck is parked facing right with its rear section surrounded by walls. The

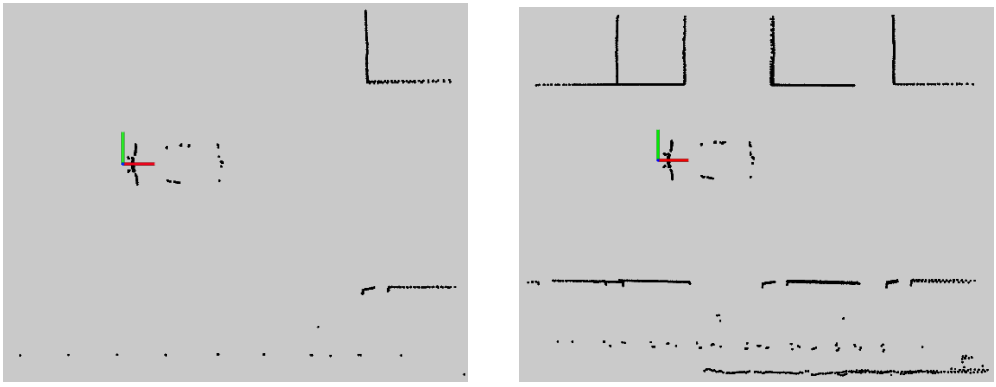
intersection of the red and green axes mark the base link and origin of the vehicle coordinate system. Panel **(a)** of Figure 4.2 show a good calibration and panel **(b)** a calibration which is unsuccessful. The difference could be thought as minor but gives an indication of the accepted tolerance. The resulting rate of success is displayed in Table 4.1.



**(a)** Case 2. Inside garage, one scene. **(b)** Case 2. Inside garage, five scenes.

**Figure 4.3:** Case 2. The truck is slowly reversed inside a garage and multiple scenes are gathered over one minute. Panel **(a)** display one scene used when applying the ICP algorithm and panel **(b)** include four extra scenes also used by the GA.

Case 2 is also recorded inside a garage, but includes multiple scenes gathered over a time span of one minute in a case where the truck was slowly reversing. Thus this data set allow for multiple scene evaluation as described in Section 3.3. The ICP make use of data from one scene, see panel **(a)** of Figure 4.3, but the GA utilize five scenes as displayed in panel **(b)**, Figure 4.3. Note that the GA evaluates each scene separate, but here, all scenes are shown in one picture. These scans do, however, provide less points in the rear common field of views compared to Case 1. The rate of success from both algorithms provided with both input sets are displayed in Table 4.1. The GA manages to calibrate almost all of the inputs but the ICP does not ever succeed. The major reason for such large rejection of the ICP is because of bad alignment of the RL Lidar. This could potentially be explained by the lack of sufficient overlap of data, i.e. the RL Lidar has less points in common with its reference Lidar, FL Lidar, compared to the rest.



**(a)** Case 3. Outside, one scene. **(b)** Case 3. Outside, four scenes.

**Figure 4.4:** Case 3. The truck is slowly driving rightwards such that a corner appear at multiple relative locations. Panel **(a)** display one scene used when applying the ICP algorithm and panel **(b)** include three extra scenes also used by the GA.

Case 3 was recorded outside while the truck was slowly driving forward (rightwards in figure) such that a corner appeared at multiple relative locations. It includes less data and more sparse objects such as poles. As similar to Case 2, this data set allows for multiple scene evaluation, so the GA is set to make use of four scenes. Panel **(a)** in Figure 4.4 and Panel **(b)** show the data used for the ICP and the GA respectively. As presented in Table 4.1, the GA once again manages to calibrate regardless of the initial spread while the ICP never succeeds. Using only one scene, there are very few points which makes it a very difficult scene regardless of the algorithm used. However, with the ability to use multiple scenes the GA manages to successfully calibrate in almost all of these 400 tries.

**Table 4.1:** Rate of success. Each algorithm was run on three different cases initialized with coordinates drawn from two separate distributions. The result of 200 different coordinate combinations, 100 from each distribution, was evaluated visually and judged as successful or not.

| | Distribution width | | | |
|---|---|---|---|---|
| | $S_{\text{wide}}$ | | $S_{\text{narrow}}$ | |
| | 0.1 m, 7.5° | | 0.05 m, 3.75° | |
| **Rate of success** | GA | ICP | GA | ICP |
| Test case 1 | 56% | 25% | 42% | 46% |
| Test case 2 | 100% | 0% | 99% | 0% |
| Test case 3 | 98% | 0% | 96% | 0% |

Table 4.1 shows the results as rate of success for both algorithms using the two initial sets. The GA manages to calibrate in majority of the tries but the ICP fails if the overlapping data between Lidars is insufficient. The ability to make use of multiple scenes becomes a key factor. Using multiple scenes thus enables the GA to process more data and from various perspectives. However, the ICP only make use of one scene and instead fails to calibrate one or multiple Lidars.

## 4.2 Calibration of Lidars on truck

The GA was applied to find the coordinates of four new Lidars just mounted by the supplier. 19 scenes were gathered when the truck was driven, reversing and turning, inside a garage to capture various scenes from different perspectives. Figure 4.5 shows one of the scenes used during calibration. The truck was always moving slowly and thus no speed compensation was used. The lateral distance of both rear Lidars was measured and inserted as part of the initial guess. The mean value of the two measurements are kept throughout the calibration as part of the absolute positioning described in Section 3.2 and should, therefore, be measured with care.

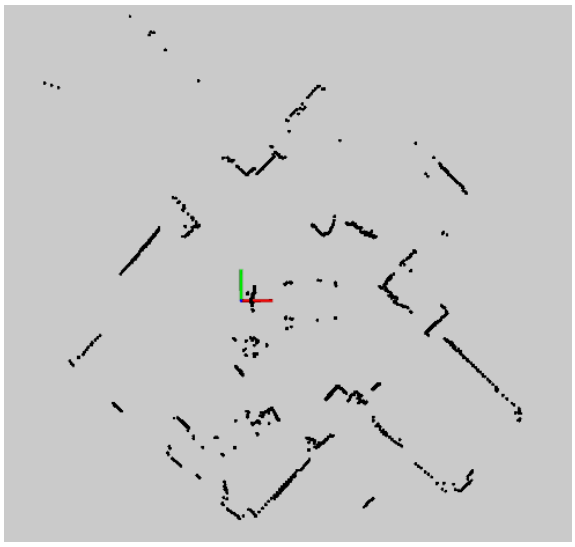


**Figure 4.5:** A truck with four newly mounted Lidars was calibrated inside a garage. The resulting coordinates were applied for real usage of the truck.

After calibration, the coordinates were used to run the Lidar setup in live performance. To analyze the results, traffic cones, cardboard boxes and white board screens were moved around the truck to examine the measurements of the objects aligned. Each Lidar is mounted using a rigid housing, so trying to physically measure the coordinates is considered inaccurate. However, the resulting coordinates were all considered feasible given the placement of each housing. Finally, the inspection indicated a set of well calibrated coordinates, so the result was used in the Lidar-based navigation system of the truck.

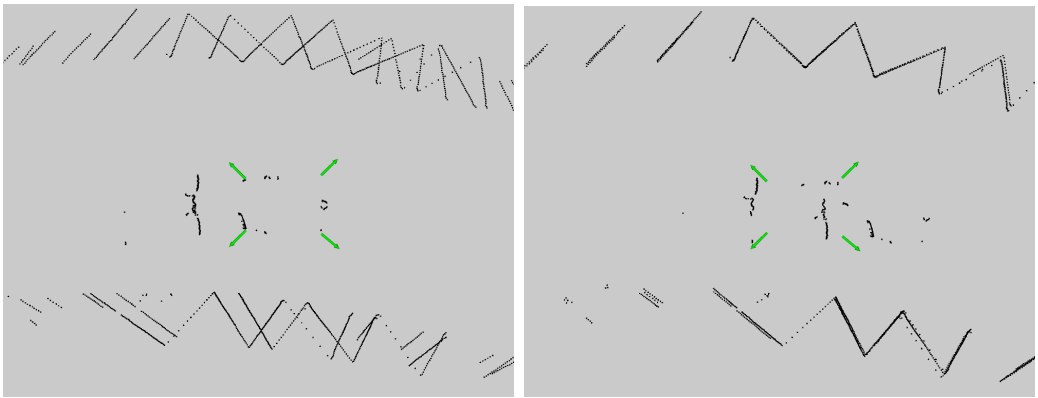
## 4.3 Velocity compensation

As described in Section 2.2, scan timing inconsistency and Lidar scan skewing are two effects of moving Lidars. In Section 3.4, two methods were introduced to compensate for timing and scan skewing distortions. The results of applying these methods on the overall performance of the calibration are presented and discussed in the following two sections.

### 4.3.1 Compensation of scan timing inconsistency

Assuming that the coordinates of all Lidars are known, scan timing inconsistency causes misalignment of point clouds once merging them into a common perspective. However, if the GA is run with scenes containing timing inconsistency without considering any compensation, the calibration is prone to result in inaccurate estimation of the Lidar coordinates. The GA solves for coordinates which yields a good alignment of points, so if the scenes are corrupted with timing inconsistency the GA will not compensate for this. The resulting visualization of such a calibration may be clear, but the coordinates are inherently wrong because the input is corrupted. Therefore, the evaluation of this compensation was performed using a truck calibrated without velocity distortions.

Compensation of timing inconsistency, described in Section 3.4.1, was evaluated by recording Lidar scans from the truck moving at $v_T = 20$ km/h and turning at $\omega_T = 0.1$ rad/s. From the recordings, a scene was tailor made such that it had abnormal time differences between the scans. Figure 4.6 displays two versions of the scene corrupted with timing inconsistency of $\Delta t = 800$ ms between the first and last scan. By visually examining the scene before and after the compensation, it is clear how all the scans are moved to all be representative at the same time instance. Panel **(a)** in Figure 4.6 shows the merged scene using a good set of Lidar coordinates, before compensation. The result is a merge of scans which do not align. Panel **(b)** in Figure 4.6 shows the results after applying timing inconsistency compensation to the same scene. The scans align but note how points in the center of the figure are also moved. These points are Lidar hits of the truck which align badly. Due to the relative velocity between the truck and the Lidars is zero, compensation of these points is incorrect.



**(a)** Before timing compensation **(b)** After timing compensation

**Figure 4.6:** Evaluation of scan timing compensation on a scene with time difference $\Delta t = 800$ ms, velocity of $v_T = 20$ km/h and turn rate $\omega_T = 0.1$ rad/s. Please note that this is a scene with extra large time differences produced specifically for evaluation. In panel **(a)**, distortion is present and scans do not align. In panel **(b)**, the timing error is compensated and scans align. Lidar hits of the truck however align badly.

A timing difference of 800 ms is an extreme case, but, by looking at Figure 4.6 the example demonstrate how well the scans align despite such a large distortion. The outcome is similar with less timing inconsistency but not as evident. However, despite the compensation, the resulting merge is not perfectly aligned. Effects from sweep distortion, varying speed, and turn rate may also inflict and restrain the accuracy. With less timing error, however, the result is less exposed to varying velocities.

### 4.3.2 Compensation of Lidar scan skewing

Scan skewing is a phenomenon that affects point clouds generated by each Lidar and moves around the points in the horizontal plane. The compensation method to remove such distortions, described in Section 3.4.2, shifts the points in longitudinal direction since the lateral distortions are rather insignificant. Points in each Lidar's cloud are compensated to a different amount depending on the angular distance from the first ray. The first ray is set as the starting point with the longitudinal shift intensifying with increasing angle of the ray. Applying this to the multiple Lidar setup, results in a varying amount of compensation around the truck. Figure 4.7 displays a merge of point clouds both before and after compensation. The longitudinal velocity is $v_T = 35$ km/h and with a turning rate of $\omega_T = 0.17$ rad/s. White points represent the raw input point clouds as perceived by the Lidars while black points are comprised by compensated clouds. Note, the longitudinal velocity $v_T$ is utilized during compensation but the turn rate is disregarded as described in Section 3.4.2.

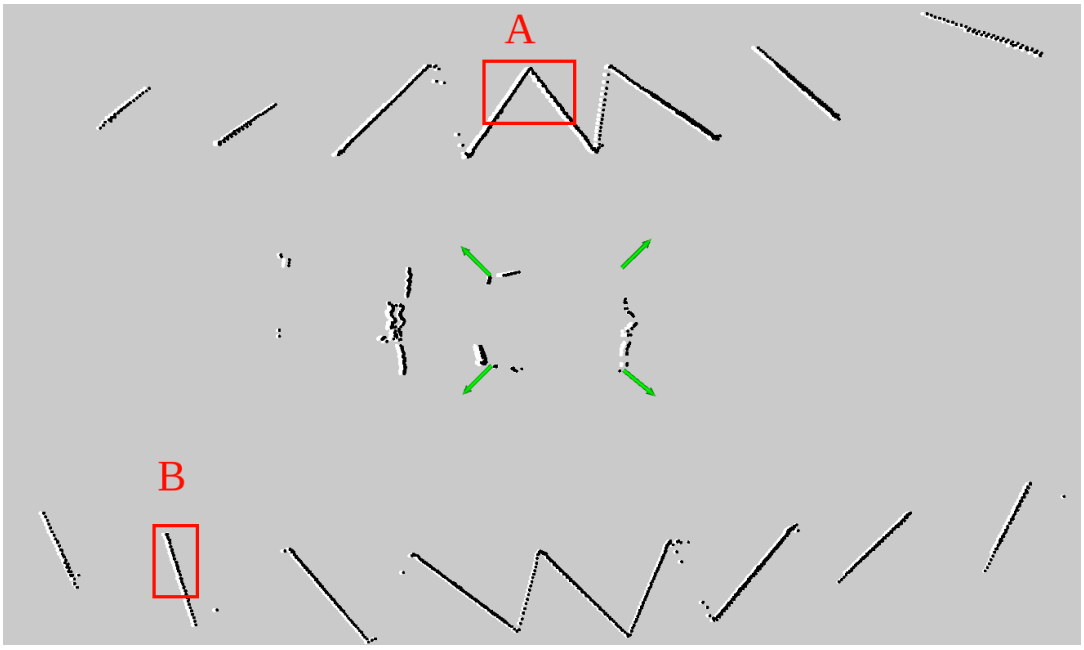



**Figure 4.7:** Scan sweep distortion. White and black points represents the scans before respectively after sweep compensation. The skewing can be seen varying in magnitude over the area, similar to Figure 4.8. Detailed views of area **A** and **B** are displayed in Figure 4.9.

The difference between before and after compensation varies based on the objects' location on the scene. For example, in the middle top part of Figure 4.7 denoted by **A**, there is a distinguishable difference between the black and white points, but in the bottom middle section it is not as clear. Figure 4.8 illustrates a theoretical approach to this result. The blue curves represent the Lidar scans when the truck was stationary and the red ones when the truck was moving in longitudinal direction. The difference between these two represents the compensation. The region marked with an **A** thus experience big compensation of both the FL and RL Lidars but region **B** on the other hand has an intermediate compensation of the RR Lidar and almost none of the FR and RL Lidars.

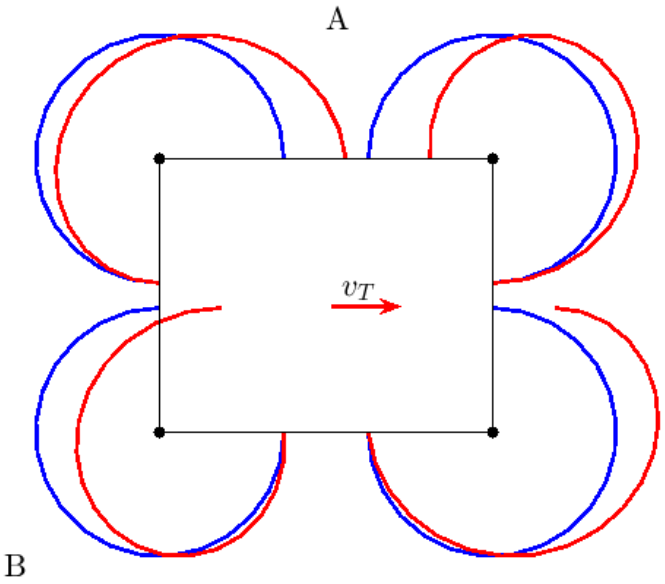
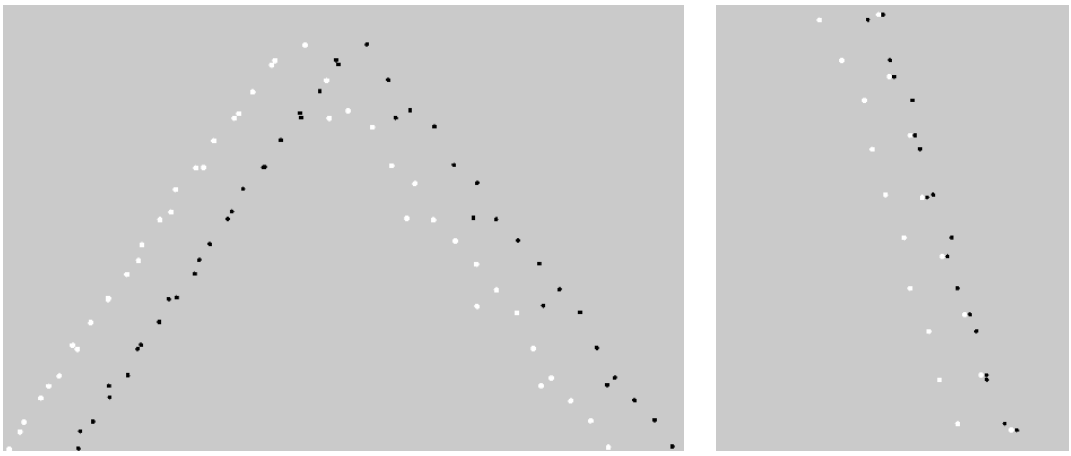



**Figure 4.8:** Scan sweep compensation of multiple Lidars setup. The blue curves represent the Lidar scans when the truck was stationary and the red ones when the truck was moving in longitudinal direction. The difference between these two represents the compensation.

This theoretical approach, as depicted in Figure 4.8, is also evident in the result. Figure 4.9 shows the details of region **A** and region **B**. In **A**, all points are shifted by about the same distance so the density is preserved. On the other hand, in region **B**, points from the RR Lidar should be compensated longer in longitudinal direction than points from the FR and RL Lidar. The resulting black points therefore end up to be more condensed which is reasonable assuming that the Lidars scanned the same object.



**(a)** Points in region **A** **(b)** Points in region **B**

**Figure 4.9:** Detailed views of Figure 4.7. White and black points represent the points before and after the sweep compensation, respectively. In region **A**, the magnitude of sweep compensation is large, but similar for all points regardless of the sourcing Lidars. In region **B**, the compensation is varying in magnitude for points from different Lidars.

The examples of region **A** and **B** show how the sweep compensation varies around the vehicle, thus, making the compensation unique in every direction and location around the vehicle. In conclusion, with such minor adjustments, it is difficult to verify the correctness. The ground truth is unknown and thus no absolute comparison can be made. However, the results do correlate with the theoretical description of the phenomenon but there is no guarantee that the correction works perfectly. Again, here the compensation only concerns the longitudinal distortions.

# 5
# Discussion

The purpose of this project was to explore the possibility of applying stochastic optimization algorithms to the problem of Lidar calibration. The initiative grew from a desire of a more convenient tool which did not require accurate measurements in order to succeed. Running the algorithm, without velocity distortions as in Section 4.2 and 4.1, shows very promising results. However, in order to reach such results, two different methods were used to answer the most central question of the thesis, *What is a good calibration?*. Firstly, the method of *fitness evaluation*, see Section 3.1.2, quantify the alignment and thus determine if a set of coordinates yields a good calibration or not. But, the constructed method only provides a relative measure which does not state whether the end result is useful. Secondly, to tell if the output coordinates are good or bad, a more subjective evaluation was used. By visually inspecting the composition of transformed point clouds the calibrations has been judged as successful or not. The following sections discuss these two methods in relation to the results and lastly also the results of velocity compensation.

## 5.1 Fitness evaluation

The fitness measure, described in Section 3.1.2, is a very central part of the GA wherefore much time and effort was focused on this. A more intuitive and straight forward method is to use a square filter grid to down sample and count the number of removed points. Square voxel grids are a common functionality in point cloud libraries and hence quite simple to apply. However, square grid cells are constant in size which does not entirely comply with point clouds generated by Lidars. Instead, a circular grid has an increasing cell size with the distance from the origin, similar to Lidar measurements. Intuitively this yields a more correct comparison regardless of the measurements being close or more distant form the truck. By using a circular grid instead of a square grid, the algorithm became more robust and converge faster. This was an impressive improvement and it is noteworthy that small changes of the evaluation procedure may have major impact on the performance of the algorithm.

Moreover, sparsification of the input data described in Section 3.1.2.2 also contributed to the increase in the robustness of the algorithm. This ensures that big objects, which provide high fitness mainly due to dense cluster, are instead rewarded with increased fitness only if aligned with another Lidars point cloud. By starting with a large grid, and thus potentially removing many points, the evaluation rely on more salient structures in the scans. It tends to guide the algorithm by easing

the gap between good and bad alignment which otherwise can be quite significant. It should, therefore, also relax the fitness gained if a local minimum is reached.

With roulette wheel selection, it is possible to increase the risk of premature converge but with the preparatory sparsification it is not an apparent problem. Additionally, introducing the ability to evaluate on multiple scenes, makes the calibration less sensitive to premature convergence. By examining the results in Table 4.1, it is clearly advantageous to make use of more than one scene. The unsuccessful cases are often stuck in a local minimum and just slightly misaligned. Thus, the risk of premature convergence is reduced by using several scenes. However, the carefully prepared evaluation procedure is considered to be the underlying cause of such success. The combination of applying preparatory sparsification (Section 3.1.2.2), a circular grid evaluation (Section 3.1.2.1) and a gentle decrease of the filter grid size (Section 3.1.2.3) is recognized as the fundamental component for the algorithm to succeed in such a high number of cases.

## 5.2 Robustness, GA and ICP comparison

The performance of the algorithms are evaluated by visually inspecting the merged scans. Because the ground truth is not available, manual and visual inspection are the final assessment tools to judge the performance. Visual inspection is advantageous because it is simple and accurate. A trained eye can easily reject incorrect solutions, but because it is manual it is also tedious. To be able to evaluate the robustness, the tests were restricted to 100 runs. This may not bring a thorough statistical evidence, but it provides enough information for significant indication of the performance.

As displayed in Table 4.1, the ICP method tends to converge more often with coordinates drawn from the narrow distribution. It is a reasonable result because the *matching function* should provide better matches given the guess is more accurate. However, the GA tends to perform slightly better for all three cases once the coordinates are drawn using the wider distribution. This is quite unexpected because more accurate initial guesses should, intuitively, result in a more robust calibration. Further more, the visual inspection is a subjective judgment and therefore not totally accurate. Lastly, an alternative idea is that with a faulty guess the GA is maintaining a higher degree of diversity and, therefore, more often finds the global optimum. However, even though the results of the robustness tests are slightly surprising, it shows significant indications of a well functioning algorithm. The GA outperforms the ICP in most of the cases, especially when multiple scenes are available.

## 5.3 Velocity compensation

The compensation of scan timing inconsistency is a constant bias of the Lidar coordinates because the distortion is solely apparent in the coordinate system of the truck. As mentioned in Section 3.4.1, the magnitude of correction can be calculated

before starting the GA because it only relies on the input scans and their time of arrival rather than the Lidar coordinates. Furthermore, as shown in Section 4.3.1 it is rather simple to visually examine if the compensation is viable. The evaluation make use of a scene with exaggerated timing errors but despite this the compensation produce a good, though not perfect, alignment. Since no ground truth is available the correctness is difficult to fully verify, but with correct Lidar coordinates, the alignment of the merge indicate an accurate compensation. Consequently the scan timing compensation can handle timing errors and is considered adequate to prevent the calibration from adapting to time distorted scenes. Compensation of scan skewing on the other hand is not as straightforward.

As mentioned in Section 4.3.2, the merging of scans, including skewing compensation, is more persistent as points turn out to be more dense and closer to reality. So, with correct Lidar coordinates this could improve the accuracy of the merged scans. However, in difference to timing inconsistency the magnitude of scan skewing is dependant on the location and angle of each Lidar. So if the estimated Lidar coordinates are bad, the compensation will inherently be so as well. Consequently, applying the compensation while running the GA could introduce errors. Incorrect compensation may alter the search space and introduce new, and inaccurate, optima which were otherwise not existent. The population would indirectly adapt to this and the GA would likely succeed to produce a well aligned merge of the scans, but the resulting coordinates would be incorrect. The alternation of the points is rather minor so errors of such cases are difficult to detect. It is therefore very complex to determine the effects on calibration if continuously altering the input scans.

# 6
# Conclusion

The purpose of this project was to come up with an alternative calibration tool to estimate the x, y and yaw coordinates of four 2D Lidars mounted on a truck. We have used a genetic algorithm to estimate the 12 Lidar coordinates and the algorithm successfully do so in more than 95% of the test cases. The alignment of Lidar point clouds are evaluated using a grid-based approach and with this method, no artificial landmarks are required as long as there are sufficient Lidar measurements in the overlapping fields of views. Nevertheless, making use of multiple scenes, i.e. snapshots including a scan from each Lidar, has shown to overcome the issue of insufficient Lidar measurements.

We have also studied distortions emerging once the truck and Lidars are moving. The ultimate goal was to calibrate a new Lidar installation using data from a moving truck. Two methods to compensate for distortions caused by motion have been developed and shown promising results of improving the accuracy of merged scenes. However, the effects of scan skewing compensation was found difficult to verify and it has not been proved sufficiently reliable to be used during calibration. Therefore calibration should be performed on data recorded by a slowly moving truck. This way only minor distortions are present but it is possible to acquire various perspectives such that the advantages of multiple scene evaluation can be utilized.

A further development of the work may include modification of the algorithm to be run online as a background task once the truck is operating. As the intended environment is both rough and includes navigation in tight regions, the Lidar positions may change due to collisions. It would therefore be suitable to develop an algorithm to monitor the Lidar setup and possibly adjust or warn if incorrect calibration is detected. However, the proposed algorithm is computationally demanding and unsuitable to be run on the truck with limited computational resources. Nevertheless, the method of fitness evaluation is found successful and could possibly be transferred to create an online version. By making use of a less computationally demanding technique, such as particle swarm optimization, it could search for disruption in the point cloud alignment. Lastly, the work could be extended to include 3D Lidars, and with such field of view, the roll and pitch angles and z-coordinates could be considered.

# Bibliography


[1] Feng Lu and Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 935–938, June 1994.

[2] S. T. Pfister, K. L. Kriechbaum, S. I. Roumeliotis, and J. W. Burdick. Weighted range sensor matching algorithms for mobile robot displacement estimation. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 2, pages 1667–1674 vol.2, May 2002.

[3] B. Jensen and R. Siegwart. Scan alignment with probabilistic distance metric. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2191–2196 vol.3, Sep. 2004.

[4] Andrea Censi. An icp variant using a point-to-line metric. *2008 IEEE International Conference on Robotics and Automation*, pages 19–25, 2008.

[5] François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4:1–104, 05 2015.

[6] Z. Pusztai and L. Hajder. Accurate calibration of lidar-camera systems using ordinary boxes. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 394–402, Oct 2017.

[7] Yoonsu Park, Seokmin Yun, Chee Sun Won, Kyungeun Cho, Kyhyun Um, and Sungdae Sim. Calibration between color camera and 3d lidar instruments with a polygonal planar board. *Sensors*, 14(3):5333–5353, 2014.

[8] José E. Guivant, Eduardo Mario Nebot, and Hugh Durrant Whyte. Simultaneous localization and map building using natural features in outdoor environments.

[9] J. M. Maroli, Ü. Özgüner, K. Redmill, and A. Kurt. Automated rotational calibration of multiple 3d lidar units for intelligent vehicles. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, Oct 2017.

[10] James Underwood, Andrew Hill, and Steven Scheding. Calibration of range sensor pose on mobile platforms. pages 3866 – 3871, 10 2007.

[11] Mengwen He, Huijing Zhao, Jinshi Cui, and Hongbin Zha. Calibration method for multiple 2d lidars system. 05 2014.

[12] Marcelo Pereira, Vitor Santos, and Paulo Dias. Automatic calibration of multiple lidar sensors using a moving sphere as target. In Luís Paulo Reis, António Paulo Moreira, Pedro U. Lima, Luis Montano, and Victor Muñoz-

Martinez, editors, *Robot 2015: Second Iberian Robotics Conference*, pages 477–489, Cham, 2016. Springer International Publishing.

[13] Dong-Geol Choi, Yunsu Bok, Jun-Sik Kim, and In So Kweon. Extrinsic calibration of 2-d lidars using two orthogonal planes. *IEEE Transactions on Robotics*, (1):83, 2016.

[14] A. Birk and S. Carpin. Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE*, 94(7):1384–1397, July 2006.

[15] Xin Ma, Rui Guo, Yibin Li, and Weidong Chen. Adaptive genetic algorithm for occupancy grid maps merging. In *2008 7th World Congress on Intelligent Control and Automation*, pages 5716–5720, June 2008.

[16] H. Li, M. Tsukada, F. Nashashibi, and M. Parent. Multivehicle cooperative local mapping: A methodology based on occupancy grid map merging. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2089–2100, Oct 2014.

[17] Pierre Merriaux, Yohan Dupuis, Rémi Boutteau, Pascal Vasseur, and Xavier Savatier. Lidar point clouds correction acquired from a moving car based on can-bus data. *CoRR*, abs/1706.05886, 2017.

[18] M Wahde. *Biologically inspired optimization methods: an introduction*. WIT Press, Ashurst, 2008.