Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2022

Real Time Lidar and ICP-Based Odometry in Dynamic Environments

Ludvig H. Granström and Carl Hampus Hedén

LINKÖPING
UNIVERSITY

Master of Science Thesis in Electrical Engineering

Real Time Lidar and ICP-Based Odometry in Dynamic Environments

Ludvig H. Granström and Carl Hampus Hedén

LiTH-ISY-EX--22/5513--SE

Supervisor: **Kristin Nielsen**
ISY, Linköping University

Examiner: **Gustaf Hendeby**
ISY, Linköping University

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

A robust and highly accurate positioning system is required to transition to fully autonomous vehicles in society. This thesis investigates the potential for lidar sensors to be a part of a localization system, adding redundancy in case of an outage in a *global navigation satellite system* (GNSS). Point cloud data is recorded on a busy road to experimentally study lidar odometry with dynamic objects present. By matching point clouds with the well-established *iterative closest point* (ICP) algorithm, odometry estimates in 6 degrees of freedom are obtained. In this thesis, three ICP variants, point-to-point, point-to-plane and plane-to-plane, are evaluated along with preprocessing and data segmentation techniques to improve accuracy and computational speed.

High-end lidar sensors are known to produce a large amount of data. To achieve real-time performance for the odometry, the point clouds are downsampled using a 3D voxel grid filter to reduce the amount of data by 86% on average. Experiments show that downsampling with a properly tuned voxel grid filter reduces the total process time without sacrificing the accuracy of the estimates.

ICP algorithms assume the environment to be static. Therefore dynamic objects can introduce errors in the odometry estimates. Methods to counteract these errors are evaluated. One approach to address this issue, suggested in the literature, is to segment the point cloud into different objects and remove objects smaller than a given threshold. However, experiments on the recorded data set indicate that this method removes too much point cloud data in certain sections, resulting in inaccurate odometry estimates. This problem is especially salient when the environment lacks larger static structures.

However, outlier rejection methods show promising results for suppressing errors caused by dynamic objects. In scan matching, outlier rejection methods can be used to identify and remove individual data point pair associations whose shared distance deviates from the majority in the point clouds. Removing the outliers strengthens the estimates against errors caused by dynamic objects and improves robustness against measurement noise. Experiments in this thesis show that outlier rejection methods can improve translation accuracy with as much as 39% and rotation accuracy with 57% compared to not using any outlier rejection.

To improve the accuracy of the estimates, this thesis proposes an approach to divide the lidar point clouds into two subsets, ground points and non-ground points. The scan matching can then be applied to the two subsets separately, enhancing the most relevant information in each subset. Compared to the traditional way of using the entire point clouds in one estimate, experiments show that using the best performing ICP variant, a linearized point-to-plane, in combination with this proposed method improves translation accuracy by 10%, rotation accuracy by 27%, and computational speed by 23%.

The results in this thesis indicate that a lidar odometry solution can be accurate and computationally efficient enough to strengthen a localization system during shorter GNSS outages.

Sammanfattning

Ett robust och exakt positioneringssystem är ett krav för övergången till helt autonoma fordon i samhället. Detta examensarbete undersöker om lidarsensorer kan användas som en del av ett lokaliseringssystem, för att skapa redundans för avbrott i *globala navigationssatellitsystemet* (GNSS). Punktmolndata spelas in på en trafikerad allmän väg för att experimentellt studera lidarodometri med dynamiska objekt närvarande. Genom att matcha punktmoln med den väletablerade *iterativt närmsta punkt* (ICP) algoritmen, erhålls odometri i 6 frihetsgrader. I denna uppsats utvärderas tre ICP-varianter, punkt-till-punkt, punkt-till-plan och plan-till-plan, tillsammans med förbearbetnings- och segmenteringstekniker för att förbättra noggrannhet och beräkningshastighet.

Avancerade lidarsensorer är kända för att producera en stor mängd data. För att uppnå realtidsprestanda nedsamplas datan med ett 3D-voxel-rutnätsfilter för att minska mängden data med 86% i genomsnitt. Experiment visar att nedsampling av punktmolnen med ett korrekt inställt voxelgridfilter minskar den totala beräkningstiden utan att försämra uppskattningarnas noggrannhet.

Dynamiska objekt kan introducera fel i odometriska uppskattningar då ICP algoritmer antar att miljön är statisk. Därför utvärderas metoder för att motverka dessa fel. Ett tillvägagångssätt som föreslås i litteraturen är att segmentera punktmolnet i olika objekt och ta bort objekt som är mindre än ett givet tröskelvärde. Experimenten på den inspelade datan indikerar att denna metod tar bort en för stor del av punktmolnsdatan i vissa sektioner, vilket resulterar i felaktiga odometriska uppskattningar. Detta problem är särskilt framträdande när miljön saknar större statiska objekt.

Metoder för att avvisa extremvärden visar dock lovande resultat för att dämpa fel orsakade av dynamiska objekt. Vid skanmatchning används extremvärdsavisning för att identifiera och radera individuella punktpar vars transformation avviker från majoriteten i punktmolnen, vilket stärker uppskattningarna mot fel orsakade av dynamiska objekt och förbättrar robustheten mot mätbrus. Experiment visar att avvisning av dessa extremvärden kan förbättra translationsnoggrannheten med så mycket som 39% och rotationsnoggrannheten med 57% jämfört med att inte använda någon extremavvisning.

För att förbättra uppskattningarnas noggrannhet föreslår denna uppsats att dela upp lidarpunktmolnen i två delmängder, icke-markpunkter och markpunkter. Skanmatchningen kan då appliceras på de två delmängderna separat, vilket gör att den mest relevanta informationen i varje delmängd kan användas. Jämfört med det traditionella sättet att använda hela punktmolnen, visar experiment att användning av den bästa ICP-varianten, en linjäriserad punkt-till-plan, i kombination med denna föreslagna metod förbättrar translationsnoggrannheten med 10%, rotationsnoggrannheten med 27% och beräkningshastighet med 23%.

Resultaten i denna uppsats indikerar att en lidarodometrilösning kan utvecklas som uppskattar position tillräckligt noggrant och beräkningseffektivt för att stötta ett lokaliseringssystem under kortare GNSS-avbrott.

Acknowledgments

First of all, we want to thank our supervisor Kristin Nielsen for your admirable positive attitude, your immense know-how in the research area, and the countless hours spent guiding us on this journey.

We would also like to thank our industrial supervisors Bianca, Tristan and Jimmy for your commitment to this thesis. Your help always sparked interesting discussions.

The daily mind-clearing ping-pong matches would not have been as joyful without Alexander Strid, Daniel Liu, and John Liu. Hope our paths cross in the future.

Finally, we would like to thank our examiner Gustaf Hendeby for your meticulous proofreading, valuable inputs, and for being an inspiration in the research area.

Linköping, June 2022
LHG och CHH

Contents

Notation

Nomenclature

Notation	Meaning
d	Distance
sd	Scaled distance
sa	Scaled angle
θ	Azimuth angle
$\mathcal{P}_j$	Source point cloud
$\mathcal{P}_i$	Target point cloud
$\mathbf{R}$	Rotation Matrix
α	Roll rotation
β	Pitch rotation
γ	Yaw rotation
t	Translation vector
p_n^j	Point in $\mathcal{P}_j$ with index n
p_n^i	Point in $\mathcal{P}_i$ with index n
ν	Grid resolution in the voxel grid filter
$\mu_{1/2}$	Median
μ	Mean
σ	Standard deviation
ϵ	Fixed threshold in RMT
e_n	Relative motion error in RMT
ξ	Trim percentage

Abbreviations

Abbreviation	Meaning
SAE	Society of Automotive Engineers
AV	Autonomous vehicle
GNSS	Global navigation satellite system
IMU	Inertial measurement unit
EKF	Extended Kalman filter
SLAM	Simultaneous localisation and mapping
NDT	Normal distribution transform
RANSAC	Random sample consensus
ARRSAC	Adaptive real-time random sample consensus
MLESAC	Maximum likelihood estimation sample consensus
GPINSAC	Gaussian process incremental sample consensus
RBNN	Radially bounded nearest neighbor
KNN	K nearest neighbor
ICP	Iterative closest point
MMW	Millimeter-wave
PCA	Principal component analysis
GICP	Generalized iterative closest point
PCL	Point cloud library
RMSE	Root mean square error
SRMSE	Scaled root mean square error

1 Introduction

This introductory chapter gives an overview of the localization problem for autonomous vehicles and outlines the thesis's purpose, scope and contributions.

1.1 Background and motivation

The automotive industry is heading towards a paradigm shift, where vehicles are progressing rapidly towards a high level of autonomy. The International Society of Automotive Engineers (SAE) have defined a standard (J3016) [12] for *autonomous vehicles* (AVs). The standard formulates six levels (0-5) of autonomy:

- **Level 0 - No Driving Automation**
  No automation. A human handles all driving-related duties.
- **Level 1 - Driver Assistance**
  Under certain conditions, the car controls the vehicle's speed or steering.
- **Level 2 - Partial Driving Automation**
  The vehicle can maneuver in certain circumstances, but the driver must remain engaged and monitor the environment.
- **Level 3 - Conditional Driving Automation**
  The vehicle can manage most aspects of driving in the right conditions and even monitors the environment. The driver will have to intervene when the vehicle encounters a scenario where it can not navigate itself.
- **Level 4 - High Driving Automation**
  No human input or oversight is required except under special conditions, and the pedals and steering wheel remain in the driver seat.

- **Level 5 - Full Driving Automation**
  The vehicle can operate autonomously in any condition a human could.

Automobile manufacturers are producing cars qualifying as level 3 vehicles, and some vehicles are today being evaluated for a level 4 status [23]. Even though legal obstacles are present to make higher-level cars ubiquitous in society, technology is maturing at a rapid pace. A robust and highly accurate positioning system is required to transition to level 5 vehicles. If an AV's localization system were to malfunction, it could endanger people and property near the vehicle. A typical positioning system incorporates a *global navigation satellite system* (GNSS) to measure the absolute positions of the AV. The positioning system is therefore vulnerable to a malfunction in the GNSS and to GNSS-denied environments, such as tunnels or when surrounded by larger buildings in urban areas.

A *laser imaging, detection and ranging* (lidar) sensor scans the environment and compiles measured ranges into a point cloud [51]. Lidar sensors are mounted and used for object detection among other purposes in some AV's [4]. Suppose the data generated by the lidar can be used for relative positioning. In that case, the localization system can get greater robustness against shorter GNSS-outages without adding additional sensors to the AV.

Estimating positional change over time-based on lidar data is called *lidar odometry*. The estimations can be derived by calculating a transformation that matches two point clouds generated by consecutive scans, commonly referred to as *scan matching*. The calculated transformation corresponds to the movement of the lidar between the lidar scans. A well established scan matching framework for point clouds is the *iterative closest point* (ICP) algorithm [18].

The thesis is conducted in cooperation with a company that works with AV's. For confidentiality, the company shall remain anonymous and be referred to as the *partner company* throughout the thesis. The *partner company* is utilizing lidar sensors on their AV's for other purposes than localization, hence the choice of investigating the lidars for odometry rather than other sensors. The *partner company* has provided data from lidars and their current positioning system to be used as ground truth. The lidar odometry solutions will therefore be evaluated experimentally using this data set. Results presented in this thesis are scaled with a confidential factor to not reveal sensitive data concerning the *partner company*.

1.2 Problem Statement

The majority of research relating to lidar odometry is based on data captured near university campuses, as in [40], or indoors, as in [9], or the KITTI data set [19], as in [55]. The ICP algorithm assumes the environment to be static. Therefore dynamic can introduce errors in odometry estimates [54]. Data collected near campuses or indoors does not contain many, if any, larger dynamic objects. The residential part of the KITTI data set has not been captured during rush hour. Therefore, large dynamic objects in the data set are not frequent enough to cause significant performance decreases.

Literature focusing on mitigating errors caused by dynamic objects often utilize deep learning approaches as in [54]. However, training a neural network requires a well-labeled data set which might not be available for all environments. Moreover, deep learning approaches have a *black box* nature [39]. It can therefore be hard to guarantee the reliability of such solutions. Other papers that focus on dynamic objects in lidar odometry fuse the lidar with external sensors that can measure the radial speed of objects, *e.g.* a millimeter-wave (MMW) radar in [13]. Adding more sensors to a vehicle also adds more points of failure for the system and increases the cost. In [40] a lidar-based approach of filtering dynamic objects based on their size is utilized. However, an urban data set is used, and the performance of this method needs further study in a setting with fewer large structures.

Most of the literature conducts experiments with point cloud data recorded using older and cheaper lidar sensors. They capture fewer range measurements per scan than modern high-end lidar sensors. For example, the popular KITTI data set is recorded using a single Velodyne lidar with 64 channels operating at 10 Hz [19]. Others such as [46] and [9] use 16 channel lidars. Since the scan matching algorithms are computationally expensive, point clouds with lower density do not need to be preprocessed to the same extent as the point clouds with higher density to achieve real-time performance.

In scan matching, *outlier rejection* methods identify and discard individual data point pair associations whose transformation deviates from the majority in the point clouds. In [35], outlier rejection techniques are compared for the purpose of enhancing the scan matching estimates. Furthermore, they are evaluated on 2D point cloud data indoors. However, there is a lack of comparisons between various outlier rejection techniques for dense 3D point cloud data in a dynamic outdoor environment.

1.3 Thesis scope

Figure 1.1 gives a system overview for the lidar odometry system to be investigated. The possibility of integrating a lidar odometry solution with a localization system is investigated as it is an important aspect [50]. However, implementing a filter is outside the scope of this thesis.

In [44], three main categories of ICP variants are formulated:

- Point-to-point
- Point-to-plane
- Plane-to-plane

In this thesis, the three ICP variants are compared and evaluated using ground truth data provided by the *partner company*. The best performing ICP variant

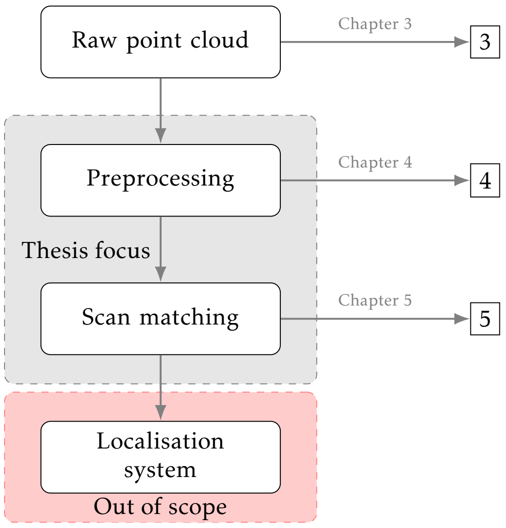


***Figure 1.1:** System overview of the ICP based odometry solution*

is then used for in-depth evaluation of the preprocessing, outlier rejection techniques and dynamic object error mitigation strategies.

1.4 Thesis aim and research questions

The aim of the thesis is first to investigate how a lidar odometry module for a localization system can be constructed with the ICP framework and the methods found in the literature. Second, how well such a module strengthens a localization system during GNSS outages with regard to accuracy and process time.

The aim, together with the challenges and scope stated above, leads to the following five research questions:

- **Which ICP variant performs the best in an industrial suburban environment?**
- **How does preprocessing of dense point cloud data affect the scan matching in regards to process time and accuracy?**
- **What outlier rejection technique offers the most significant performance gain in a real-world setting?**
- **On busy roads, how can dynamic objects be accounted for and errors reduced without using auxiliary sensors or machine learning models?**
- **Is it reasonable to assume that a well-tuned lidar odometry solution can be used as input to a localization system to add redundancy in the event of a GNSS outage?**

1.5 Delimitations

The integration of the odometry solution to a filter-based localization system is outside the scope of this thesis due to time constraints.

As the odometry estimates are evaluated from the perspective of usage in automotive localization, where loop closing is rare for longer routes, mapping is excluded from this thesis. The focus is instead to optimize the odometry estimates between each full scan, *i.e.* a *frame-to-frame* methodology.

The sections relating to dynamic objects will exclusively focus on lidar-based approaches and do not use other sensors for filtering purposes. Due to a lack of labeled data, machine learning methods are not evaluated in this thesis.

1.6 Scientific approach

To answer the research questions, the following approach is taken. First, a thorough review of the literature relating to lidar ICP methods is conducted. From the literature review, three prominent ICP variants are selected that use the geometric information in the lidar data to different degrees. From the literature, preprocessing methods, outlier rejection techniques, and dynamic object error mitigation strategies are also selected for evaluation.

Data is collected using the *partner company*'s test vehicle during rush hour to obtain the sought-after data set, which contains many dynamic objects and sections with and without many large structures.

The three selected ICP methods are compared. The best performing ICP variant is then used for in-depth comparisons of the preprocessing settings, outlier rejection techniques and dynamic object error mitigation strategies. Only using one of the ICP variants for the other evaluations is done to keep the number of experiments reasonable. Finally, the results from the different evaluations are combined, and the results are used to answer the research question.

1.7 Contributions

The thesis contributions are:

- An evaluation of how voxel grid filtering and ground separation affect dense point cloud data, and the following ICP estimates can be found in chapter 4. An alternative way of using the ground plane based on the research in [46] is presented that increases the odometry accuracy.
- An evaluation of the three ICP variants based on accuracy and computational speed on a novel data set can be found in chapter 5. Outlier rejection methods are evaluated similar to [35] but using a data set recorded in a real-world environment and with high-end modern 3D lidars rather than a 2D lidar in an indoor environment.

- This thesis adds to the literature with an evaluation of lidar-based, non-machine learning strategies for minimizing the effects of dynamic objects in ICP estimates. This can be found in chapter 6
- An evaluation of the best performing ICP variant as a localization module can be found in chapter 7. The evaluation is based on accuracy, processing time and distribution of errors.

Overall this thesis adds to the lidar odometry literature with several assessments of important subsystems from a perspective of lidar localization as a module in a more extensive localization system.

1.8 Division of labor

The labor has been divided between the two authors in the following way.

Hedén has developed and had the primary responsibility for the sections relating to the point-to-point ICP and the point-to-plane ICP. He has also led the efforts with the implementation of outlier rejection methods.

Granström has led the efforts with the implementation of the plane-to-plane GICP. He also developed the code relating to segmentation and wrote the chapters relating to kd-trees and range images. Granström has also developed the proposed ground separation approach to estimate the transformations twice and had responsibility for the preprocessing implementation.

There has, however, been a large amount of collaboration during the thesis, where one author has built on existing work from the other. For example, after Granström implemented the segmentation function, Hedén optimized the tuning. Hedén wrote the base code for evaluating the results, and afterward, Granström debugged it. The remaining work not mentioned above has been done in collaboration.

1.9 Thesis overview

This thesis is organized as follows:

- Chapter 1 gives an overview of the thesis problem, defines the purpose and scope, outlines this thesis's content, and provides a system overview for the odometry solution.
- Chapter 2 presents background theory.
- Chapter 3 describes the hardware and software used to experimentally implement and evaluate the algorithms. The data set and the data collection is described as well as a detailed description of the evaluation metric.
- Chapter 4 outlines the sequential preprocessing steps and motivates the tuning parameters.

- Chapter 5 describes the implementation of the ICP variants and compares them. The measures are taken to improve performance, such as outlier rejection.
- Chapter 6 elaborates on the odometry errors introduced by dynamic objects and evaluates strategies to minimize them.
- Chapter 7 presents the performance of the implemented odometry solution as a module of a localization system.
- Chapter 8 summarizes the findings from this thesis and gives suggestions for future work.

2 Theory

This chapter is intended to describe the most common methods and algorithms found in the literature on lidar-based odometry. The most common way to obtain relative localization with lidar sensors is to utilize *point cloud scan matching*. The methodology is to find the transformation that minimizes the difference between two subsequent point clouds.

2.1 Lidars and point clouds

The main function of a lidar sensor is to measure ranges by emitting laser pulses and analyzing the time it takes for the pulses to return. The resulting ranges can be compiled into a *point cloud* where each point represents a distance to the surrounding area from the lidar. A point cloud at time step i is defined as

$$\mathcal{P}_i = \{p_1^i, p_2^i, \dots p_N^i\}, \tag{2.1}$$

where N equals the total number of points in the point cloud. In addition to the ranges, the intensity of the returning light and/or Doppler shift [1] can be measured, which extends the dimensions of a point cloud in 3D to 4D, often referred to as a 4D lidar [2].

Lidar sensors can be categorized as solid state or rotating. Solid state lidars create the point cloud from an instant snapshot, while the rotating lidar creates the point cloud by compiling multiple smaller scans taken during a sweep of the lidar diodes [37]. The rotating lidar can get a wider field of view than the solid state lidar but can introduce *motion distortion* to the data. Motion distortion occurs if the lidar or observed objects move during a sweep. Motion distortion results in an offset for every point in the point cloud that does not belong to the first recorded part of every sweep. Since different parts of the point cloud are captured from different locations, the point cloud does not represent the environment accurately [52].

An advantage of using a lidar sensor compared to a regular camera is that the lidar functions well during both day and night and gives three-dimensional information [31]. The lidar sensor normally produces less data than a camera but still enough to cause issues with on-board storage for vehicles, and real-time processing [27]. A modern commercial lidar sensor can capture 2,621,440 points per second [34]. The lidar sensors are also more expensive than a camera.

2.2 Scan matching

There exists several scan matching algorithms, e.g. *normal distribution transform* (NDT) [26]. This thesis will only focus on the ICP framework and its variants [6]. The goal of scan matching is to find the rigid transformation consisting of the rotation matrix **R** and the translation vector t, to match the *source* point cloud $\mathcal{P}_j$ to the *target* point cloud $\mathcal{P}_i$, such that $\mathcal{P}_i = \mathbf{R}\mathcal{P}_j + t$ [30]. This assumes identical point clouds $\mathcal{P}_j$ and $\mathcal{P}_i$ that might be misaligned. **R** can in three dimensions be defined as

$$\begin{aligned}\mathbf{R} &= \mathbf{R}_z(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_x(\gamma)\\ &= \begin{bmatrix}\cos(\alpha) & -\sin(\alpha) & 0\\ \sin(\alpha) & \cos(\alpha) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos(\beta) & 0 & \sin(\beta)\\ 0 & 1 & 0\\ -\sin(\beta) & 0 & \cos(\beta)\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\gamma) & -\sin(\gamma)\\ 0 & \sin(\gamma) & \cos(\gamma)\end{bmatrix}.\end{aligned} \tag{2.2}$$

whose Euler angles are α, β and γ, respectively. The Euler angle describes the rotation of one coordinate system to another one. Commonly a body-fixed coordinate frame to a global coordinate frame [21]. In this thesis, the Euler angles are calculated as the rotation defined from the previous point cloud. If the body-fixed coordinate frame XYZ is initially aligned with another coordinate frame xyz before an intrinsic rotation represented by Euler angles, the successive orientation during this three elemental operation may be denoted as

- x-y-z (initial orientation).
- x'-y'-z' (after the first rotation).
- x''-y''-z'' (after the second rotation).
- X-Y-Z (final orientation, after the third rotation).

There are 12 meaningful ordered sequences of intrinsic Euler angle conventions denoted from the body axes. An example is the ZYX order which is illustrated in figure 2.1 and implies that

- α represents a rotation around the z-axis. Often referred to as yaw.
- β represents a rotation around the y'-axis. Often referred to as pitch.

- γ represents a rotation around the x''-axis. Often referred to as roll.

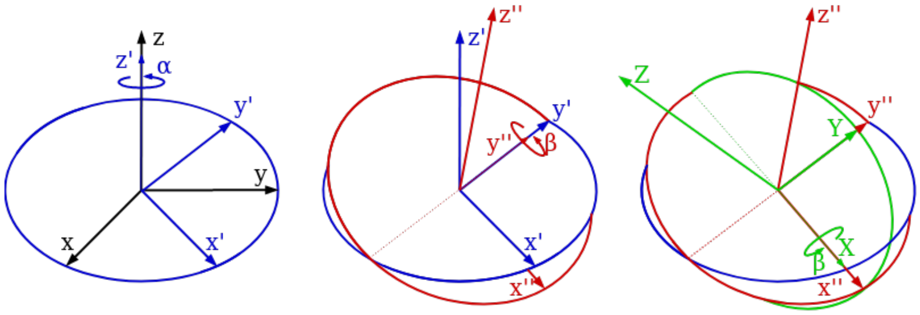


Figure 2.1: *Illustration of sequential intrinsic Euler angle rotations in the ZYX order.*

For point clouds in three dimensions, the translation vector t is defined as

$$t = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T, \tag{2.3}$$

where the translation vector is the minimizing distance after a rotation have been applied.

Most scan matching algorithms consist of the following iterative steps:

1. Initial guess of the transformation.
2. Point association.
3. Remove outliers.
4. Calculate transformation.
5. Check convergence criteria. If not converged, repeat from step 2.

Figure 2.2 shows an overview of the typical scan matching steps.

2.3 Preprocessing of point cloud data

Most scan matching algorithms in the literature are computationally heavy, suggesting that a large amount of data will result in poor real-time performance. Therefore, the point clouds can be reduced to ease processing.

2.3.1 Distance-based removal of points

A trivial way of preprocessing the point cloud is to remove points close to and far away from the lidar [9]. First, the lidar mount and parts of the vehicle can obstruct the vision and therefore be represented as a part of the point cloud at

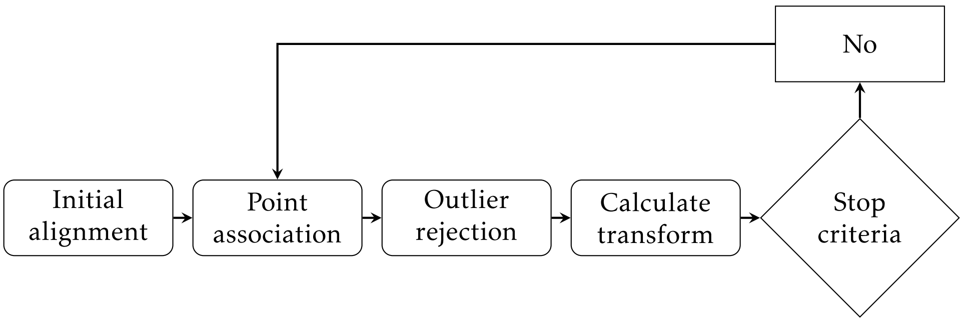


Figure 2.2: *Overview of the scan matching steps.*

every scan. Hence, the points close to the lidar can be removed without losing information relevant for estimating odometry. Second, lidars have a maximum range specified by the manufacturers. Measurements beyond this range are unreliable and can therefore be considered noise. The unreliability is due to the laser beam radius widening with distance which decreases the photon density per area unit. The photon density will be below the photoreceptor's detection limit at a certain range. Atmospheric disturbances will also have an increased effect as the range increases. Removing points far from the lidar ensures a higher average quality point cloud.

2.3.2 Downsampling

A simple method to reduce the data set would be to remove points semi-randomly, *i.e.* remove every n^{th} point in the point cloud. However, a more sophisticated method that retains more information is to apply a *3D voxel grid filter* [22]. The 3D voxel grid filter divides the point cloud into a 3D grid consisting of voxels. The most common grid structure for a 3D space is a cubic grid. However, spherical and cylindrical grids can also be used to better utilize the inherent structure of lidar measurement data [3].

The points in each voxel are reduced to one point, either as the mean of the points enclosed in the voxel or as the geometric center of that voxel [22]. The former is naturally slower than the latter but better represents the distribution of points in that voxel.

An illustration of a 2D voxel grid filter is shown in Figure 2.3. The voxel grid filter introduces the design parameter *grid resolution* ν. A coarse grid resolution (larger ν) removes more points from the point cloud resulting in a lower amount of data to process. In contrast, a finer grid resolution (smaller ν) retains more information from the original point cloud and more data to process. Consequently, choosing the right grid resolution can be difficult and data set specific. In [9], the point cloud data is preprocessed lightly to keep approximately 10,000 points in each cloud on average with a grid resolution of 0.25 m.

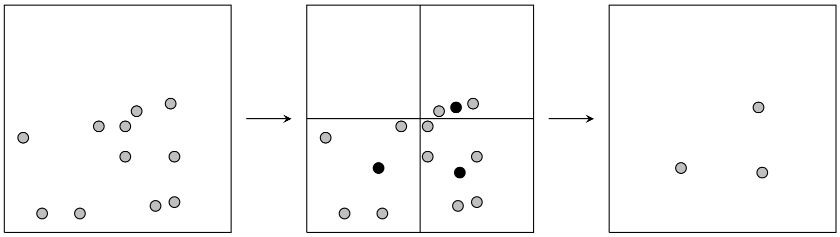


Figure 2.3: *Simplified visualisation of a voxel grid filter in 2D. The square to the left illustrates 11 measured points. The square gets divided into 4 voxels where the mean of the points in every voxel is represented as black dots. The square to the right illustrates the resulting points after applying the 2D voxel grid filter. 11 points have been reduced to 3.*

2.4 Spatial data structures

As elaborated in section 2.5, a nearest neighbor search is performed to assign point pairs in the scan matching. A brute force nearest neighbor search has a $\mathcal{O}(nm)$ time complexity in the size of the point clouds, as all points in the first point cloud of size n, must be compared with all points in the other point cloud of size m. The time complexity can be simplified to $\mathcal{O}(n^2)$ for point clouds of equal size. This order of complexity can be improved by utilizing suitable data structures. This section introduces two popular spatial data structures for point cloud data.

2.4.1 kd-tree

A kd-tree is a binary search tree, sorting points spatially using k-dimensions, where k corresponds to the point dimensions. The root node represents a point in space, preferably as close to the median as possible in all dimensions for a more balanced search tree. First, every node in the tree splits the space in one dimension. Second, the discriminating dimension in which the space is split is altered at each depth level [14]. The time complexity of building a kd-tree is $\mathcal{O}(n\log(n))$ where n is the number of points. Additionally, in [7] they show that a nearest neighbor search using the kd-tree is close to $\mathcal{O}(\log(n))$. Figure 2.4 illustrates a kd-tree for a set of points in 2 dimensions.

2.4.2 Range images

Another method of bringing structure to point clouds is to project the points onto a *range image*. The range image is 2D, where each row corresponds to an elevation angle, each column corresponds to an azimuth angle, and each pixel corresponds to the distance measured by the lidar. Subsequently, transforming the 3D data into a grayscale 2D image. First, making the data easier to visualize in 2D and second, facilitating processing since the point cloud can be traversed

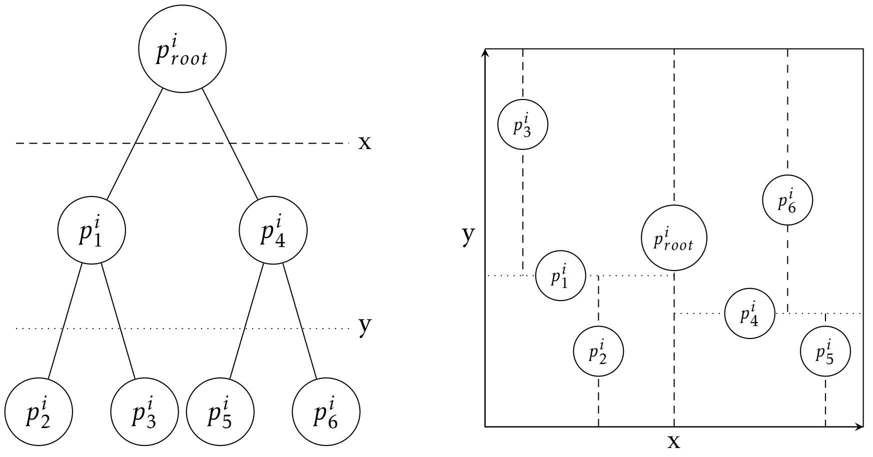


Figure 2.4: *Illustration of a kd-tree with $k = 2$ to the left, sorting the two dimensional points of $\mathcal{P}_i$ in the confined space to the right.*

column-wise and row-wise when processing the data. This method is used by [46], [45], [56], [8] to facilitate segmentation of point clouds. Figure 2.6 shows a range image of the point cloud in Figure 2.5.

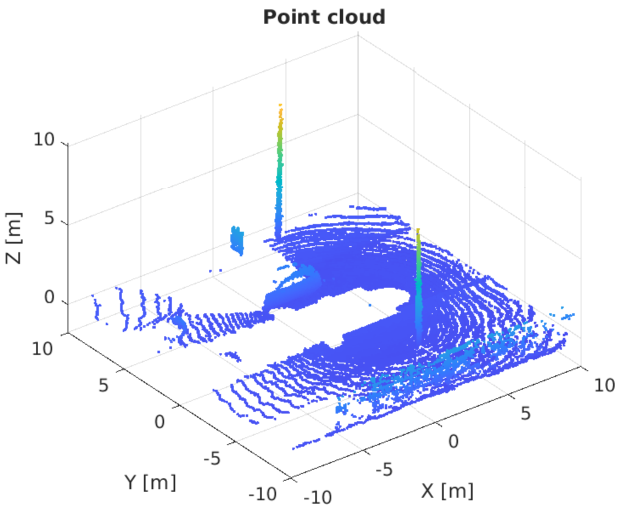


Figure 2.5: *A typical point cloud. The color of the points indicates the height.*

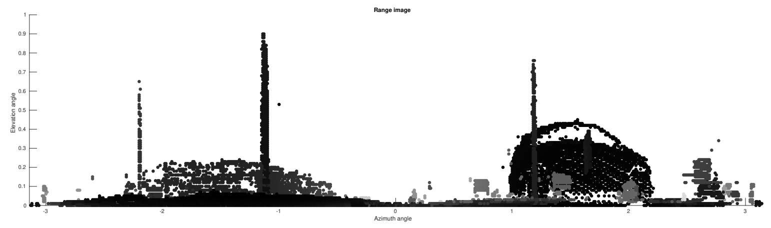


Figure 2.6: *Range image where the points in Figure 2.5 have been projected. The color indicates a range where black is close and white is far away. The perspective of the range image is from the center of the vehicle's wheel axle.*

2.5 Iterative Closest Point (ICP)

The iterative closest point (ICP) algorithm is the most common technique for scan matching. The algorithm aligns the source point cloud $\mathcal{P}_j$ with a target point cloud $\mathcal{P}_i$ iteratively, using two key steps [6]. First, point association is performed to select which pairs of points are to be evaluated in a goal function to be minimized. Second, the rotation matrix $\mathbf{R}$ and translations vector t that minimize that goal function is computed and applied to $\mathcal{P}_j$. The process is repeated until a set of stop critera is met.

Examples of stop criteria are the convergence of estimates with smaller and smaller transformations, the value of the goal function compared to a threshold or if a maximum number of iterations are reached. Furthermore, some minor modifications of the ICP are presented in [41] based on:

- Altering the association method.
- Rejecting associated point pairs, described in detail in Section 2.7.
- Weighing point pairs differently in the goal function.

More significant modifications are also presented, such as:

- Changing the goal function.
- Changing how to minimize the goal function.

There are three main categories of ICP variants based on these modifications, point-to-point, point-to-plane, and plane-to-plane. The biggest differences between the variants are the goal functions that are to be minimized. The goal functions for each variant are presented in sections 2.5.1, 2.5.2, and 2.6.2 respectively. The steps of the ICP algorithm are illustrated in Figure 2.2.

2.5.1 Point-to-Point

For the point-to-point variant, the goal function that is to be minimized each iteration is the sum of the distance between every point in $\mathcal{P}_j$ and its corresponding point in $\mathcal{P}_i$,

$$\mathbf{R}, t = \underset{\mathbf{R},t}{\arg\min} \sum_{n=1}^{N} ||\mathbf{R}p_n^j + t - p_n^i||^2. \tag{2.4}$$

N is the number of points matched in the association step, not the total number of points. The associations are commonly found by a nearest neighbor search in a kd-tree constructed for $\mathcal{P}_i$. $\mathcal{P}_i$ and $\mathcal{P}_j$ are sorted according to this association, i.e., index n for the point p_n^j in $\mathcal{P}_j$ corresponds to its associated point p_n^i in $\mathcal{P}_i$. According to [5], the arguments $\mathbf{R}$ and t in the goal function (2.4) can be found using a nonlinear solver or by applying singular value decomposition of the cross-covariance matrix $\mathbf{C}$ defined by

$$\mathbf{C} = \sum_{n=1}^{N} (p_n^i - \overline{p}^i)^T (p_n^j - \overline{p}^j), \tag{2.5}$$

where $\overline{p}^i$ and $\overline{p}^j$ denotes the center of mass for the point clouds $\mathcal{P}_i$ and $\mathcal{P}_j$. By singular value decomposition, find

$$\text{SVD}(\mathbf{C}) = \mathbf{USV}^T, \tag{2.6}$$

where $\mathbf{U}, \mathbf{V}$ are unitary, and $\mathbf{S}$ is a diagonal matrix consisting of the three singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3$ of C. The optimal solution for the rotation is

$$\mathbf{R} = \mathbf{UV}^T. \tag{2.7}$$

In [28] they prove that if the rank of $\mathbf{C}$ is equal to 3, the translation vector t can be calculated by subtracting the mean of $\mathcal{P}_i$ with the rotated mean of $\mathcal{P}_j$ as

$$t = \overline{p}^i - \mathbf{R}\overline{p}^j \tag{2.8}$$

where the $\overline{p}^i$ denotes the mean of all associated points in $\mathcal{P}_i$ and $\overline{p}^j$ the mean of all associated points in $\mathcal{P}_j$. The point-to-point ICP procedure is described step by step in Algorithm 1.

Algorithm 1 ICP - Point-to-point

```
1: $\mathbf{R} \leftarrow \mathbf{R}_0$, $t \leftarrow t_0$,
2: $\mathcal{P}_j \leftarrow \mathbf{R}\mathcal{P}_j + t$
3: $kdtree \leftarrow$ **createKDTree**$(\mathcal{P}_i)$
4: **while** not $converged$ **do**
5:     $\mathcal{P}'_j, \mathcal{P}'_i \leftarrow$ **association**$(kdtree, \mathcal{P}_j)$
6:     $\mathbf{R}, t \leftarrow$ **calculateTransform**$(\mathcal{P}'_j, \mathcal{P}'_i)$        ▷ Equations (2.5)-(2.8)
7:     $\mathcal{P}_j = \mathbf{R} * \mathcal{P}_j + \mathbf{T}$
8:     $converged \leftarrow$ **checkConvergence**$(t, \mathbf{R})$
9: **end while**
```

2.5.2 Point-to-Plane

The main difference in point-to-plane ICP compared to point-to-point ICP is the adjusted goal function

$$\mathbf{R}, t = \underset{\mathbf{R},t}{\arg\min} \sum_{n=1}^{N} \|(\mathbf{R}p_n^j + t - p_n^i) \cdot w_n^i\|^2, \tag{2.9}$$

where w_n^i denotes the normal vector for the point p_n^i. To estimate the normal vector, a plane can be estimated for a point, and its k nearest neighbors [24]. A well-established methodology is to take advantage of *principal component analysis* (PCA) to analyze the neighborhood covariance and deduce the direction of the smallest variation of points [25]. The covariance matrix for the neighboring points can be calculated by

$$\mathbf{C} = \frac{1}{k} \sum_{i=1}^{k} (p_i - \overline{p})(p_i - \overline{p})^T, \tag{2.10}$$

where k is the number of neighboring points and $\overline{p}$ represents the mean of the neighboring points. It follows that

$$\mathbf{C}v_j = \lambda_j v_j, \quad j \in \{0, 1, 2\}, \tag{2.11}$$

where λ_j is the j-th eigenvalue of the covariance matrix, and v_j the j-th eigenvector. The cross-product of the two eigenvectors with the largest eigenvalue is the estimated normal vector as

$$v_0 \times v_1 = w. \tag{2.12}$$

The goal function (2.9) is a least-squares optimization problem where the rotation matrix **R** is nonlinear. A linear approximation is necessary to take advantage of linear least-square methods [32]. The linearization assumes that the rotation angles are small $\theta \approx 0$ and therefor $\sin(\theta) \approx \theta$ and $\cos(\theta) \approx 1$. Thus the nonlinear rotation matrix **R** in (2.2) is approximated to a linear matrix

$$\hat{\mathbf{R}} = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix}. \tag{2.13}$$

Now substituting $\hat{\mathbf{R}}$ for **R** in (2.9), the problem can be formulated as

$$\hat{\mathbf{R}}, t = \underset{\hat{\mathbf{R}},t}{\arg\min} \sum_{n=1}^{N} ||(\hat{\mathbf{R}} p_n^j + t - p_n^i) \cdot w_n^i||^2, \tag{2.14}$$

which can be identified as

$$x = \underset{x}{\arg\min} ||(\mathbf{A}x - b)||^2. \tag{2.15}$$

The matrix **A** and the vectors b and x are defined as

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} (p^j \times w^i)^T & (w^i)^T \end{bmatrix}, \\ b &= (p^j - p^i) \cdot w^i, \\ x &= \begin{bmatrix} \gamma & \beta & \alpha & t_x & t_y & t_z \end{bmatrix}^T. \end{aligned} \tag{2.16}$$

The optimal x can now be calculated by taking the pseudo-inverse of **A** as [32]

$$x_{opt} = \mathbf{A}^+ b. \tag{2.17}$$

The pseudo-inverse of **A** is defined as the matrix $\mathbf{A}^+ = \mathbf{V}\Sigma^+\mathbf{U}^T$,where Σ^+ is the matrix created by taking the inverse of the non-zero elements of Σ and leaving the zero elements unchanged. An important remark is that $\hat{\mathbf{R}}$ is optimal only for the linearized problem. Therefore, the linear approximated matrix $\hat{\mathbf{R}}$ might not be a valid rigid rotation matrix. A solution is to use the nonlinear rotation matrix **R** with α_{opt}, β_{opt}, and γ_{opt} as defined in $\mathbf{x}_{opt}$ instead of $\hat{\mathbf{R}}$. Algorithm 2 describes the point-to-plane algorithm.

Algorithm 2 ICP - Point-to-plane

```
 1: $\mathbf{R} \leftarrow \mathbf{R}_0$, $t \leftarrow t_0$,
 2: $\mathcal{P}_j \leftarrow \mathbf{R}\mathcal{P}_j + t$
 3: $kdtree \leftarrow$ createKDTree($\mathcal{P}_i$)
 4: $W_i \leftarrow$ calculateNormals($\mathcal{P}_i$)                ▷ Equations (2.10)-(2.12)
 5: while not $converged$ do
 6:     $\mathcal{P}'_j, \mathcal{P}'_i, W'_i \leftarrow$ association($kdtree, \mathcal{P}_j$)
 7:     $\mathbf{R}, t \leftarrow$ calculateTransform($\mathcal{P}'_j, \mathcal{P}'_i, W'_i$)   ▷ Equations (2.16) & (2.17)
 8:     $\mathcal{P}_j = \mathbf{R}\mathcal{P}'_j + \mathbf{T}$
 9:     $converged \leftarrow$ checkConvergence($t, \mathbf{R}$)
10: end while
```

2.6 Generalized ICP (GICP)

Generalized iterative closest point (GICP) is a unifying framework introduced in [44]. A probabilistic model is applied to the goal function where the points are considered stochastic variables.

2.6.1 Probabilistic model

The existence of two underlying sets of points is assumed, $\hat{P}_i = \{\hat{p}_n^i\}$ and $\hat{P}_j = \{\hat{p}_n^j\}$, that generates P_i and P_j through $p_n^i \sim \mathcal{N}(\hat{p}_n^i, C_n^i)$ and $p_n^j \sim \mathcal{N}(\hat{p}_n^j, C_n^j)$. The optimal transformation between two point clouds should result in $\hat{p}_n^i = \mathbf{R}\hat{p}_n^j + t$ for every associated point. The distance depends on the transformation, which is defined as

$$d_n^{(\mathbf{R},t)} = \mathbf{R}p_n^j + t - p_n^i. \tag{2.18}$$

The points p_n^i and p_n^j are assumed to be generated by independent Gaussian distributions. Hence,

$$\begin{aligned} d_n^{(\mathbf{R},t)} &\sim \mathcal{N}(\hat{p}_n^i - (\mathbf{R}\hat{p}_n^j + t), \mathbf{C}_n^i + \mathbf{R}\mathbf{C}_n^j\mathbf{R}^T) \\ &= \mathcal{N}(0, \mathbf{C}_n^i + \mathbf{R}\mathbf{C}_n^j\mathbf{R}^T). \end{aligned} \tag{2.19}$$

Maximum likelihood estimation can be applied in each iteration to calculate the translation vector t and rotation matrix $\mathbf{R}$ by letting

$$\mathbf{R}, t = \underset{\mathbf{R},t}{\operatorname{argmax}} \prod_{n=1}^{N} p(d_n^{(\mathbf{R},t)}) = \underset{\mathbf{R},t}{\operatorname{argmax}} \sum_{n=1}^{N} \log(p(d_n^{(\mathbf{R},t))}). \tag{2.20}$$

The maximum likelihood estimation can be simplified to

$$\mathbf{R}, t = \underset{\mathbf{R},t}{\operatorname{argmin}} \sum_{n=1}^{N} (d_n^{(\mathbf{R},t)})^T (\mathbf{C}_n^i + \mathbf{R}\mathbf{C}_n^j \mathbf{R}^T)^{-1} d_n^{(\mathbf{R},t)}. \tag{2.21}$$

This probabilistic model allows for point-to-point and point-to-plane versions of GICP depending on how the covariance matrices are defined. Point-to-point GICP is achieved by letting

$$\begin{aligned} \mathbf{C}_n^i &= \mathcal{I}, \\ \mathbf{C}_n^j &= 0. \end{aligned} \tag{2.22}$$

The point-to-plane GICP can be achieved by letting

$$\begin{aligned} \mathbf{C}_n^i &= \mathbf{P}_n^{-1}, \\ \mathbf{C}_n^j &= 0, \end{aligned} \tag{2.23}$$

where $\mathbf{P}_n$ is defined as the orthogonal projection matrix onto the normal vector for a point p_n^i. Strictly, P_n is not invertible since it is rank deficient. If P_n is approximated with an invertible matrix Q_n, the GICP framework approaches a point-to-plane methodology as Q_n limits towards P_n. The limit can be interpreted as constraining p_n^i to the normal vector plane [44].

2.6.2 Plane-to-plane

Plane-to-plane GICP was proposed by [44] and can be formulated by assuming that the uncertainty of a point's position is lower in the direction of its normal vector. Utilizing this assumption in both point clouds, the planar geometry of both point clouds can be considered, unlike the point-to-plane variation where the planar geometry of only one point cloud can be used. If e_1 is the direction of the surface normal and ϵ denotes a small constant, the covariance matrix for a point on that surface will be

$$\mathbf{C}_0 = \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.24}$$

Using a small constant for ϵ results in a large certainty that the point is located on the plane surface. In order to transform the covariance matrix to the correct coordinate frame, a rotation matrix $\mathbf{R}_{x_i}$ is used to align e_1 with w, where w represents the normal direction for point i. The normal vectors are calculated as described in section 2.5.2. According to [44], if μ denotes a normal vector in point cloud $\mathcal{P}_i$, and ν denotes a normal vector in $\mathcal{P}_j$, the covariance matrices are

$$\begin{aligned} \mathbf{C}_n^i &= \mathbf{R}_{\mu_n}\mathbf{C}_0, \\ \mathbf{C}_n^j &= \mathbf{R}_{\nu_n}\mathbf{C}_0. \end{aligned} \tag{2.25}$$

Algorithm 2 can be used to describe the plane-to-plane algorithm where the covariance matrices are calculated using (2.25) instead of the normal vectors in line 4.

2.7 Outlier rejection in ICP

Given that the source point cloud $\mathcal{P}_j$ and target point cloud $\mathcal{P}_i$ are not identical, every point in $\mathcal{P}_j$ will not necessarily have a correct match in $\mathcal{P}_i$. If such points are matched, the resulting point pair can be regarded as an outlier [53]. Different techniques for outlier rejection are evaluated in [35] and can be categorized as:

- **Fixed**: Setting a fixed maximum distance d_{max} between a point and its nearest neighbor. All associated points with a distance greater than d_{max} are rejected.
- **Gaussian**: Proposed in [17] which sets $d = \mu + \sigma$, where μ and σ are the mean and standard deviation of the associated distances. A Gaussian distribution of distances is assumed, which is a bold assumption in dynamic environments [35].
- **Adaptive Gaussian**: Adapting the threshold d_{max} as

$$d = \begin{cases} \mu + 3\sigma, & \text{if } 0 < \mu < \eta \\ \mu + 2\sigma, & \text{if } \eta < \mu < 3\eta \\ \mu + \sigma, & \text{if } 3\eta < \mu < 6\eta \\ \mu_{1/2}, & \text{otherwise} \end{cases}, \tag{2.26}$$

  where $\mu_{1/2}$ is the median of the distance between paired points, and η is set by the user, μ and σ are the mean and standard deviation of the associated distances. Introduced in [38]
- **Median**: The median of the distances between point associations determines the maximum allowed distance, $d_{max} = 3\mu_{1/2}$, where $\mu_{1/2}$ is the median, according to [35].
- **Trim**: Introduced by [11] the trim method rejects points based on

$$N_{considered} = (1 - \xi)N_{total}, \tag{2.27}$$

  where ξ is set by the user and N_{total} is the total number of associated points. The points are sorted based on their associated distance, with the idea of rejecting a percentage of the pair with the largest associated distance.

- **Relative Motion Threshold (RMT)**: Proposed by [35] and adapts the distance threshold d_t in each iteration based on the previous iterations translation vectors. If the iterative translation of the point clouds is converging, the threshold becomes smaller according to

$$e_t = \begin{cases} e_0, & \text{if } t < 2 \\ \min(e_{t-1}, \lambda e_{t-1}), & \text{otherwise} \end{cases}. \tag{2.28}$$

  The ratio λ determines if the iterative translations are converging by

$$\lambda = \frac{||t_{t-1}||}{||t_{t-2}||}. \tag{2.29}$$

  The distance threshold is then calculated as $d_{max} = e_t + \epsilon$ where ϵ can be a fixed threshold or based on the statistical variance of the associated distances and e_{init} is set by the user.

The techniques were evaluated in [35] and results show that RMT outperformed the other techniques, while the Gaussian, median, and trim approaches beat the fixed and adaptive Gaussian approach.

2.8 Segmentation

Segmentation is used to classify points as different objects. The objects can then be used to streamline the matching phase by removing points belonging to objects that do not meet specified criteria such as position or size. The segmentation algorithms are divided into *ground segmentation* and *non-ground segmentation*. Ground segmentation is defined as classifying points to belong to the ground surface or not. Non-ground segmentation in this thesis refers to methods that classify points belonging to different non-ground objects such as houses, cars, lamp posts, et cetera.

2.8.1 Ground segmentation

A popular preprocessing method is classifying points as belonging to the ground surface or not and removing the ground points. Ground points make up a large portion of a typical point cloud and are information sparse with regards to yaw rotation and translations along the x-and y-axis. If the yaw rotation and translations along the x-and y-axis are the main interest, the ground points can be removed without sacrificing accuracy [10]. The ground points do, however, contain valuable information for determining pitch and roll rotations and translation along the z-axis [46] which means that removing the ground plane can reduce the accuracy for said estimates.

Segmentation methods are presented in [16], where the authors argue for a voxel grid based segmentation when dealing with dense point clouds. They suggest that ground points can be found by clustering adjacent voxels based on vertical means and variances, thereby isolating points that are part of the ground plane. The reason is its simplicity, ease of representation, and scalability by adjusting the resolution.

The *random sample consensus* (RANSAC) algorithm can be used for ground segmentation [36]. The RANSAC algorithm is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers. In the case of point clouds in a 3D environment, a plane model

$$ax + by + cz + d = 0, \tag{2.30}$$

with the normal vector

$$w = \begin{bmatrix} a & b & c \end{bmatrix}^T, \tag{2.31}$$

can be estimated where ground points are considered inliers, and non-ground points are treated as outliers. In [36], a new framework for real-time robust estimation of a planar surface called *adaptive real-time random sample consensus* (ARRSAC) is presented. In [51] it is noted that there is spacial priori knowledge of ground points. Utilizing the priori knowledge that ground points are located in the lower part of the point cloud, they propose a faster converging algorithm that eliminates some random sampling in RANSAC. They also argue that a single plane approach to ground segmentation is insufficient since the ground might exhibit elevation changes. Their solution is to apply the algorithm in segments along the local axis of travel. In [49], a generalisation of RANSAC is formalised as MLESAC. The method adopts the same random sampling as RANSAC to generate putative solutions but estimates a solution to maximize the likelihood rather than just the number of inliers.

When handling sparse point clouds, [16] introduced *Gaussian process incremental sample consensus* (GPINSAC) to estimate a ground surface where non-ground objects are present.

2.8.2 Non-ground segmentation

In non-ground segmentation, the point cloud is segmented into different clusters. The goal is for each cluster to represent a different object. Ground segmentation is often performed before non-ground segmentation to ease the differentiation of objects and different clusters.

A point cloud can be segmented into clusters by considering the Euclidean distance between points and their neighbors, facilitated with a kd-tree. A minimum distance threshold is set to distinguish between different clusters [42].

A faster algorithm than Euclidean clustering algorithms is presented in [51] where points are segmented using a two-scan labeling algorithm of a range image. First, the algorithm traverses the range image row by row and groups points with labels by comparing its neighbors. Second, the algorithm traverses the labeled groups to merge them based on overlap in the range image. This segmentation method is also used in [46] where the range image is divided into equally large portions, and features are extracted from the clusters. The space is evenly divided before extracting features to utilize information in every direction. In [56], a segmentation algorithm is proposed that ensures real-time performance. The segmentation uses a range image inspired by [8], in combination with an *angle threshold* method to avoid under-and-over segmentation.

Graph-based segmentation methods consider the point clouds as a graph. An example of a simple model is where each point corresponds to a node, and the edges connect them to pairs of neighboring points [33]. A graph-based segmentation strategy is suggested in [29] based on *radially bounded nearest neighbor* (RBNN) graph, in contrast to the more common *k nearest neighbor search* (KNN) graph used by [20]. In the RBNN graph, nodes are connected to other nodes within a predefined radius. Nodes are connected to k neighbors regardless of distance in the KNN graph. The authors of [47] argue that graph-based segmentation algorithms are robust to noise but lack real-time properties due to high time complexity. A survey of segmentation methods [33] further strengthens this statement. Machine learning is a popular tool for object detection in RGB images but has also grown in popularity in recent years when processing point cloud data. Machine learning is used for object detection in [54] and segmentation in [47].

2.9 Differentiation of dynamic and static points

Most literature relating to scan matching assumes a static environment. Dynamic objects can therefore introduce errors in odometry estimates if not considered and managed. In [13], the research is based on a method proposed in [46], and extended to account for dynamic environments. MMW radar is fused with the lidar point clouds to detect dynamic objects. The radar can use the Doppler shift to measure objects' radial speed and ease non-static objects' filtering. The same technique could be applied if the lidar sensor measures the Doppler shift for each point. In [15], a lower drift is achieved in the odometry estimation by segmenting the point cloud and removing objects smaller than a given threshold. However, this method assumes that dynamic objects are small compared to the static objects present in the environment. A fully-Convolutional Neural Network (FCNN) is used in [54] to detect and segment dynamic objects such as cars, pedestrians, and cyclists.

2.10 Kalman filter

The information in this section is gathered from [21]. Filtering can be contextualized as an extension of estimation to a non-stationary state vector x_k governed by a dynamic model $x_{k+1} = f(x_k, u_k, v_k)$ where u_k is a known input and v_k model errors. The uncertainty of the estimates is expressed as a covariance matrix $\mathbf{P}_k$. The Kalman filter computes the posterior distribution exactly for linear Gaussian systems [21]. The Kalman filters are prominent in odometry estimations since it finds the best possible unbiased linear filter and can be extended to nonlinear models.

3 Experimental Setup

The scan matching algorithms are evaluated experimentally with data recorded from a car driving in an industrial suburban area. This chapter describes the hardware used to capture and process the data and the software used to evaluate preprocessing and ICP algorithms. A detailed description of the data and the route used to capture it is also included in this chapter, and a clarification of the evaluation metric is presented.

3.1 Hardware

The hardware description is divided into the vehicle equipped with sensors used to capture the data and the hardware used to evaluate the algorithms.

3.1.1 Vehicle

The data is recorded using a modified Lincoln MKZ, equipped with lidars, IMU, and GNSS receivers. The sensors of importance for this thesis are the two identical, revolving 3D lidars mounted on the vehicle's roof. The lidars have 128 channels and 270° horizontal field of view. The measurements from the remaining 90° of the horizontal field of view are aimed towards the car and are discarded during the recording. The vertical field of view is 90°(±45°). The specifications can be found in more detail in Table 3.1, while their positioning and orientation on the vehicle are illustrated in Figure 3.1 and Figure 3.2. The coordinate frame for the vehicle is defined as a right-hand Cartesian coordinate system with the x-axis in the vehicle direction and z-axis opposite to the gravity vector, pointing upwards. The origin of the coordinate system is placed on the rear wheel axle, from now on referred to as *base*. The lidars are positioned facing backward in respect to the vehicle coordinate frame, resulting in a yaw rotation of 180°. Moreover, Table 3.2 describes the transform variables to align the lidar coordinate frames to the vehicle coordinate frame, with the base in its origin.

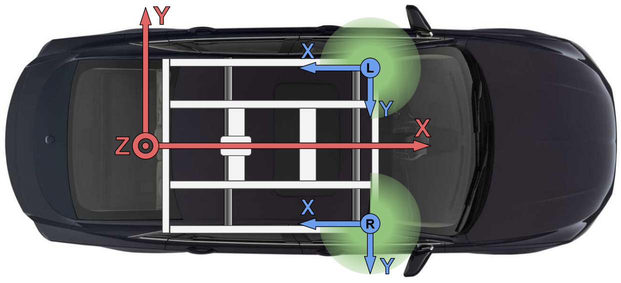


Figure 3.1: *Aerial view of the vehicle used to record data. Red arrows illustrate the coordinate frame for the test vehicle. Blue circles represent lidars, and blue arrows illustrate lidar coordinate frames. Green gradient semicircles indicate the horizontal field of view*

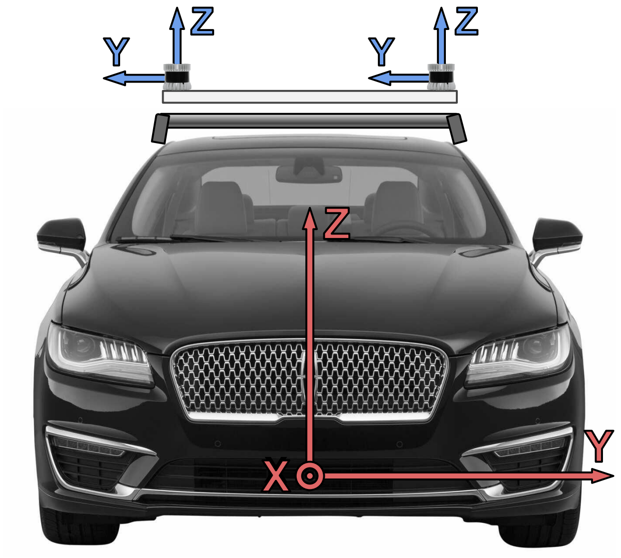


Figure 3.2: *Front view of the vehicle used to record data. Red arrows illustrate the coordinate frame for the test vehicle. Blue arrows illustrate Lidar coordinate frames.*

Table 3.1: *Specification for the lidar sensor used.*

Lidar specifications	
Vertical resolution	128 channels
Horizontal Resolution	2048
Maximum Range	50 [m]
Vertical Field of View	90°(±45°)
Precision	± 1.5 - 5 [cm]
Rotation Rate	20 [Hz]

Table 3.2: *Transforms variables for lidar coordinate frame to vehicle coordinate frame.*

Sensor	x [m]	y [m]	z [m]	roll [rad]	pitch [rad]	yaw [rad]
Left lidar	1.552	0.518	1.619	0	0	π
Right lidar	1.553	-0.513	1.618	0	0	π

Calibration

A sensor configuration of this type is sensitive to calibration errors. The lidar position and orientation calibration is performed by the *partner company* and regarded as adequate. The calibration is done with the assistants of a camera system. If the calibration is inaccurate, it will introduce errors in the odometry estimation delivered in the base frame. If the transform to the base has an incorrect calibration in orientation, it could cause a bias in both the translation and rotation estimation. Additionally, in the case of multiple lidars, an incorrect calibration could cause objects to be duplicates of points in the point cloud.

3.1.2 Computer hardware

The computer used on which the algorithms are evaluated is a *Dell Inc. XPS 15 9510* equipped with *11th Gen Intel® Core™ i7-11800H* CPU with a base clock frequency of 2.30 GHz and 8 cores (16 threads thanks to hyper-threading). Moreover, the GPU is a *Mesa Intel® UHD Graphics* (TGL GT1). However, all calculations are carried out using the CPU.

3.2 Software

MATLAB 2021b is used throughout to implement and evaluate the algorithms. The data from the lidars is published using *Robotic operating system* (ROS) to be conveniently saved as rosbags [48]. The rosbags are imported into the MATLAB environment. For the plane-to-plane version using the GICP framework, the scan matching is evaluated using the Point cloud library (PCL) in C++ [43].

The decision to use PCL with C++ is based on the ease of implementation. Consequently, the processing time between plane-to-plane with C++ and the other variants using MATLAB can not be compared fairly. The accuracy can, however, be compared more fairly.

3.3 Data collection

This section aims to elaborate on the collected data and the route used. In order to fill the literature gap of lidar evaluation in a suburban environment with a high degree of dynamic objects, the data has been recorded during the daytime when the route road is busy with traffic.

3.3.1 Logging route

The data is recorded at relatively low vehicle speeds, typical for suburban areas. The route is a 1 102 meter-long loop consisting of three roundabouts with mostly straights connecting them. The route is illustrated in Figure 3.3, and the entirety of the route is a public road. The recording is 4 minutes and 48 seconds long. It starts and ends at approximately the same geographical place. A total of 5777 lidar scans are collected. The environment can be classified as *industrial suburban* where part of the road is surrounded by buildings and larger structures. However, some parts of the route have no buildings within the range limit of the lidar. The data were recorded during the day with a natural traffic flow. The data set is captured during rush hour. Hence, many moving cars, trucks and pedestrians are present in parts of the data set.

A data set was also recorded standing still at the beginning of the route. A small portion of the road is visible for the lidar from the start location, and cars pass by. However, they pass far away and makeup only a small fraction of the point cloud. Although the data set contains passing cars and trees affected by the wind, the environment can be classified as highly static.

3.3.2 Point clouds

The data is saved as timestamped ROS messages, where a message contains all points from an entire lidar sweep. From each message, an $N \times 3$ matrix can be extracted with the Cartesian coordinate of every point in the point cloud, where N equals the number of points. Figure 2.5 shows a typical point cloud.

The timestamp of every message corresponds to the start time of that sweep. However, there is no information about the exact timestamp for individual points, and therefore, all points are assumed to have been captured at the start time of every sweep. Consequently, motion distortion (described in Section 2.1) could be present and perturbing the data. However, the lidar frequency of 20 Hz is relatively high compared to the vehicle's maximum speed. Hence, motion distortion is neglected.

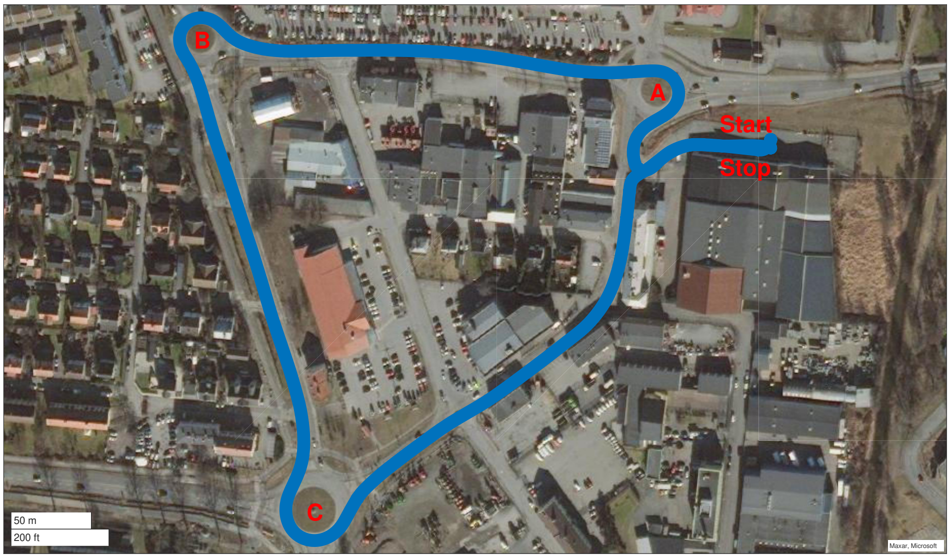


Figure 3.3: *Illustration of the route the point cloud data has been recorded. The vehicle traversed the route counterclockwise. The start and stop position is marked. The three roundabouts are named A, B, and C.*

As previously mentioned in Section 3.1.1, the vehicle is equipped with two rotating lidar sensors, and the point clouds collected from both lidars are merged, to form one coherent point cloud at each time step. One of the lidars is regarded as the reference when merging the point clouds based on timestamps. Since the data recording starts at the same time and the sweep frequency of the lidars are identical, point clouds can be merged without a time offset.

3.3.3 Ground truth

The *partner company*'s existing experimental state estimation is used for ground truth. The state estimation consists of a Kalman filter variant with measurements gathered from GNSS-RTK receivers and an IMU. The values for ground truth are linearly interpolated for every time step in the evaluation. The state estimation frequency is much greater than the lidar sweep frequency. Therefore, linearly interpolating the values for ground truth should not introduce noticeable errors.

3.4 Evaluation metrics

All absolute results are scaled to protect the *partner company*'s confidential data. The evaluation metric and plots will therefore lack units.

3.4.1 Scaled root mean square error (SRMSE)

A standard metric to evaluate accuracy is the *root mean square error* (RMSE). The error is calculated as $e = \hat{x} - x$, where $\hat{x}$ is the estimate and x the ground truth. The RMSE is computed as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{n=1}^{N}(e_n)^2}, \tag{3.1}$$

where N is the total number of estimated frames.

The existing localization system used by the *partner company* has an estimated accuracy for the translation. This accuracy is confidential and is used to scale all the results presented in this thesis. Hence, the SRMSE is calculated as

$$\text{SRMSE} = \frac{\text{RMSE}}{s}, \tag{3.2}$$

where s is the scaling factor. When using the SRMSE for evaluations, a scaled value of less than one is within the confidence of the ground truth. The *partner company* has not provided any estimated accuracy regarding the vehicle's orientation. Therefore, the RMSE for rotational estimates is scaled with the same factor as for the translation estimates. The lack of precise rotation accuracy for the ground truth complicates the evaluation of the rotation estimates but is, as stated, necessary to not reveal sensitive data.

3.4.2 Scaled units

All plots presented in this thesis will also be scaled to protect the accuracy of the provided ground truth. Therefore the unit *scaled distance* (*sd*) and *scaled angle* (*sa*) is introduced to clarify when results are scaled.

4 Data preprocessing

This chapter describes the sequential preprocessing steps and presents results regarding the voxel grid filter and the ground separation methodology. The preprocessing steps are applied as Figure 4.1 illustrates. First, points captured on the vehicle and noisy measurements far away disrupt the scan matching accuracy. The points close and far away from the lidars are therefore removed.

Second, reducing the point cloud data naturally leads to a faster scan matching process but how the voxel grid filter affects the accuracy of the scan matching is not as intuitive. To evaluate the effects of the voxel grid filter, transformations are estimated via scan matching for the entirety of the data set, using varying settings for the voxel grid filter's grid resolution and plotting the SRMSE against the average processing time. The plots help in analyzing the relationship between processing time and accuracy for different grid resolutions and are presented in section 4.2.

Third, points in the point clouds contain different geometrical information depending on where they are located in the environment. *E.g.* points that belong to the ground plane are not useful when estimating the translation along the x- and y-axis or yaw rotation. They can, however, be effectively used to estimate the roll and pitch rotations and z-translation. The same experiment as the voxel grid filter is conducted to evaluate how the ground plane affects the scan matching. This time the scan matching is performed using the entirety of the point clouds, only the non-ground points of the point cloud, and a proposed method of using the ground points and non-ground points separately.

The examples demonstrating the effect of the preprocessing steps are conducted using a typical point cloud from the data set. The introduced design parameters and their default values can be found in Table 4.1. The motivation of the default values is elaborated on in corresponding sections.

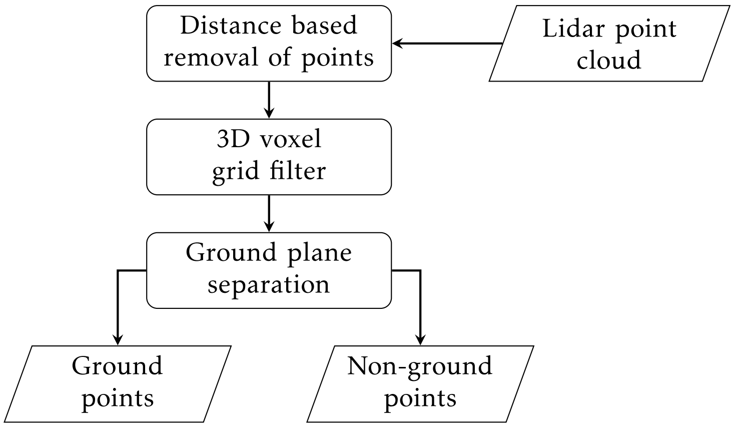


Figure 4.1: *Overview of the preprocessing steps*

Table 4.1: *Preprocessing design parameters*

Parameter	**Default value**
Threshold far away [m]	$\|\|x\|\| < 50, \|\|y\|\| < 50, -5 < z < 20$
Threshold close [m]	$-1.5 < x < 4, \|\|y\|\| < 2, \|\|z\|\| < 2$ [m]
Ground separation	True [-]
Ground removal (angular threshold)	5 [deg]
Ground removal (inlier threshold)	0.2 [m]
Voxel grid resolution	0.3 [m]

4.1 Removal based on distance

First, points with a high probability of being of low quality are removed. This includes points that are further away than the specified maximum range of the lidar, 50 meters in the x-and y-direction are removed. Points 5 meters below and 20 meters above the *base* are removed as the location of *base* is known, and these points can be assumed to be noise with high confidence. Figure 4.2 shows a raw point cloud sweep where the lidars register points well outside their reliable range of 50 meters and even well below the ground surface.

Second, the points close to the lidar are removed. The removal is due to the lidar registering many points on the vehicle itself, which are non-informative for the odometry calculations since they move with the vehicle at every frame. The test vehicle is assumed to be a $5.5 \times 4 \times 2$ meter cuboid. Furthermore, if no visible objects are present in a given lidar elevation and azimuth angle, the lidar will not register any range measurement for that angle. These no-return measurements are registered as zero range and located in origo. Hence, removing points close to the lidars also removes the no-return measurements. Figure 4.3 shows the same point cloud as in Figure 4.2 with the described preprocessing based on the distance applied. The number of points is reduced from 37 854 in the raw point cloud to 35 645, a reduction of 5.83%.

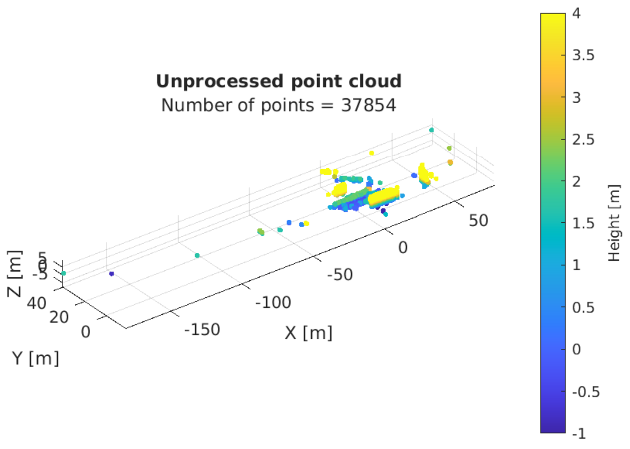


Figure 4.2: *Unprocessed point cloud.*

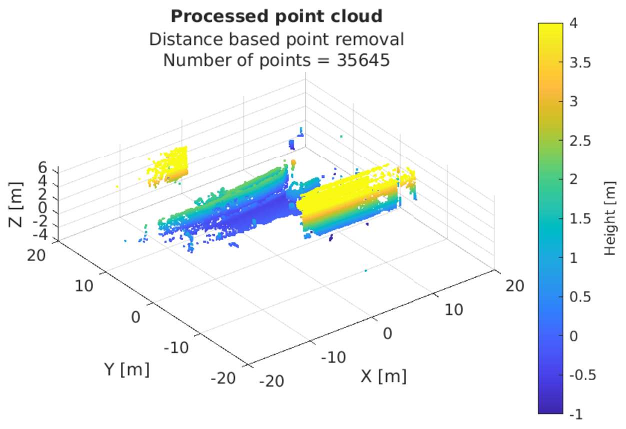


Figure 4.3: *The point cloud in figure 4.2 after removing points far away as well as the points close to the lidars, as outlined in section 4.1. Note that the point cloud is much more zoomed in due to the removed outlier points.*

4.2 3D Voxel grid filter

A 3D voxel grid filter is applied to reduce the number of points while allowing the remaining points to represent the environment adequately. Figure 4.4 shows the voxelized point cloud with the grid resolution ν set to 0.3 m. The number of points is now reduced to 5 339 from the original 37 854, a reduction of 85.90% from the raw point cloud. This is a large reduction, but since it is done systematically, points can be removed without sacrificing too much information about the environment. When comparing Figure 4.3 to Figure 4.4, the overall look of the point cloud is similar, and consequently, the filtered point cloud retains information about the environment well.

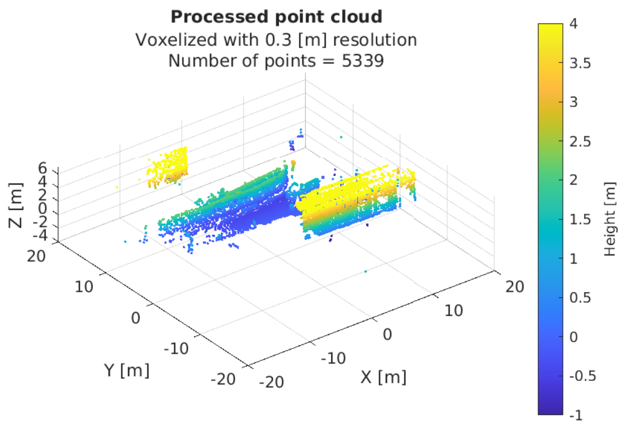


Figure 4.4: *The figure illustrates the point cloud from Figure 4.3 with a voxel filter added. The grid resolution is set to 0.3 [m]*

Figure 4.5 visualizes the effect of grid resolutions for the point-to-plane ICP. The figure shows that the total accuracy of scan matching increases with a smaller ν until around 0.3 m for the translation and 0.4 m for the rotation. Hence, the default value for the grid resolution is set to 0.3 m. Figure 4.5 only displays the point-to-plane variant but the same behavior was present for all evaluated ICP variants. The decrease in accuracy for finer grid resolutions might be due to the average movement per frame in the recorded data set. As elaborated in section 2.2, the ICP algorithm associates point pairs based on the nearest neighbor of a source point in the target points cloud. If the grid resolution is too small compared to movement per frame, the points in the voxel grid filtered points clouds will overlap more, increasing the probability that the ICP algorithm finds a local minimum rather than a global optimum for the goal function. A smaller ν also results in a longer processing time. Moreover, the gain in accuracy between 0.3 m resolution and 0.4 m resolution is small compared to the processing time gained.

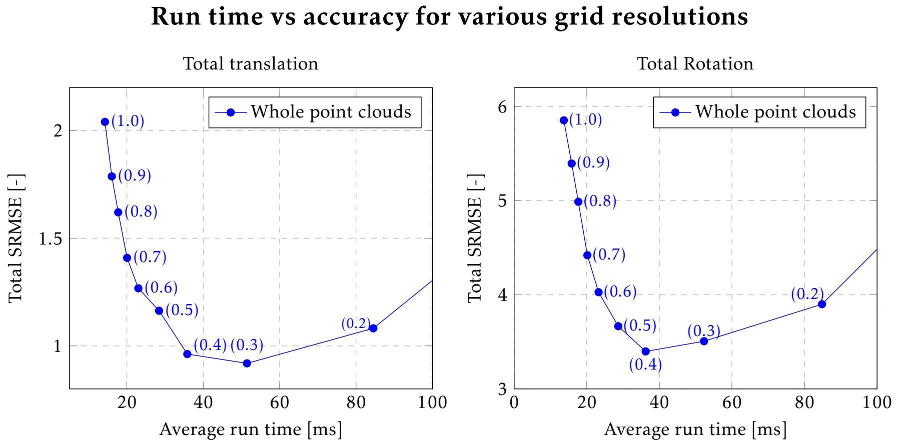


Figure 4.5: Plots showing the average run time and scaled RMSE for the recorded data set using the point-to-plane ICP with different grid resolutions. The ideal solution is located in the bottom left corner. The grid resolutions are indicated in parentheses. The RMSE is scaled using the estimated translation accuracy of the ground truth. A scaled RMSE below 1 is within the ground truth accuracy for translation estimates. Since the RMSE is scaled with the same value for rotation estimates, the quality of the rotation estimates can not be stated.

4.3 Ground separation

An MLESAC algorithm is used to find a plane of points representing the physical ground surface. The algorithm is described in Section 2.8.1 and is implemented as a plane fitting function in the MATLAB standard library called `pcfitplane()`. A priori knowledge of the ground plane's normal vector is used where the normal vector (defined in (2.31)) must be parallel with the gravity vector within a threshold of 5 degrees. An inlier threshold of 0.2 m is then used to classify points as ground or not, depending on the point's distance to the estimated plane model. When separating the ground plane, the angular and inlier threshold is based on reasonable assumptions and proved robust for the data set. However, no granular experiments were conducted to assure optimal values since no major performance increases were expected.

The point cloud with a removed ground plane can be seen in Figure 4.6. The number of points is now reduced to 3 353, a total reduction of 91.14% from the raw point cloud. When comparing Figure 4.4 with Figure 4.6 the point clouds now look quite different. However, since the ground plane is removed, not much useful information for the relative position along the x-and y-axis is lost.

Figure 4.8 shows that performing scan matching on the entirety of the point clouds gives better overall translation accuracy than with the ground points re-

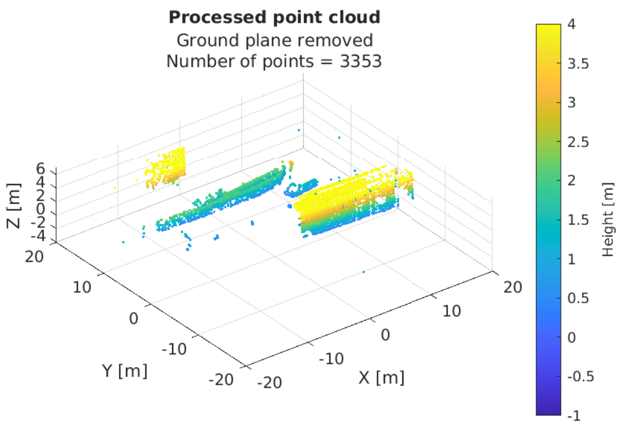


Figure 4.6: The figure illustrates the point cloud from Figure 4.4 with the ground plane removed from the point cloud as outlined in section 4.3

moved. The better accuracy is due to the better performance in z-translation while translation along the x-axis suffers. The same can be concluded in Figure 4.9 for the rotation estimates, where the scan matching on the whole point clouds results in better pitch and roll rotations and worse yaw rotation.

To improve the accuracy of the estimates, this thesis proposes an approach inspired by [46] to divide the lidar point clouds into two subsets, ground points and non-ground points. The scan matching can then be applied to the two subsets separately, enhancing the most relevant information in each subset. First, the non-ground points are used to estimate yaw rotation and x-and y-translation. The ground points are then aligned using the derived transformation and then used to estimate z-translation and pitch and roll rotations. A flow chart of the method can be seen in Figure 4.7. This method's processing time is longer than only using the non-ground points but faster than using the whole of the point cloud. The accuracy for translation estimates is improved by 8%, and the accuracy for rotation estimates is improved by 22%, compared to using the entirety of the point cloud.

If the odometry is intended for 2D localization and only the estimates of yaw rotation and x-and y-translation are of interest, only the non-ground points are of use since the ground points do not improve x,y, or yaw estimations.

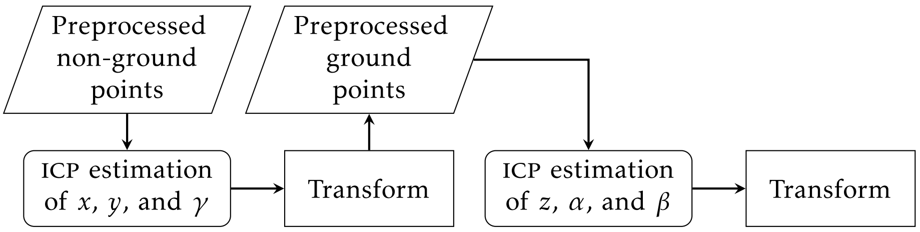

Preprocessed non-ground points
Preprocessed ground points
ICP estimation of x, y, and γ
Transform
ICP estimation of z, α, and β
Transform

Figure 4.7: *Overview of the proposed method where the transformations are estimated in two steps.*

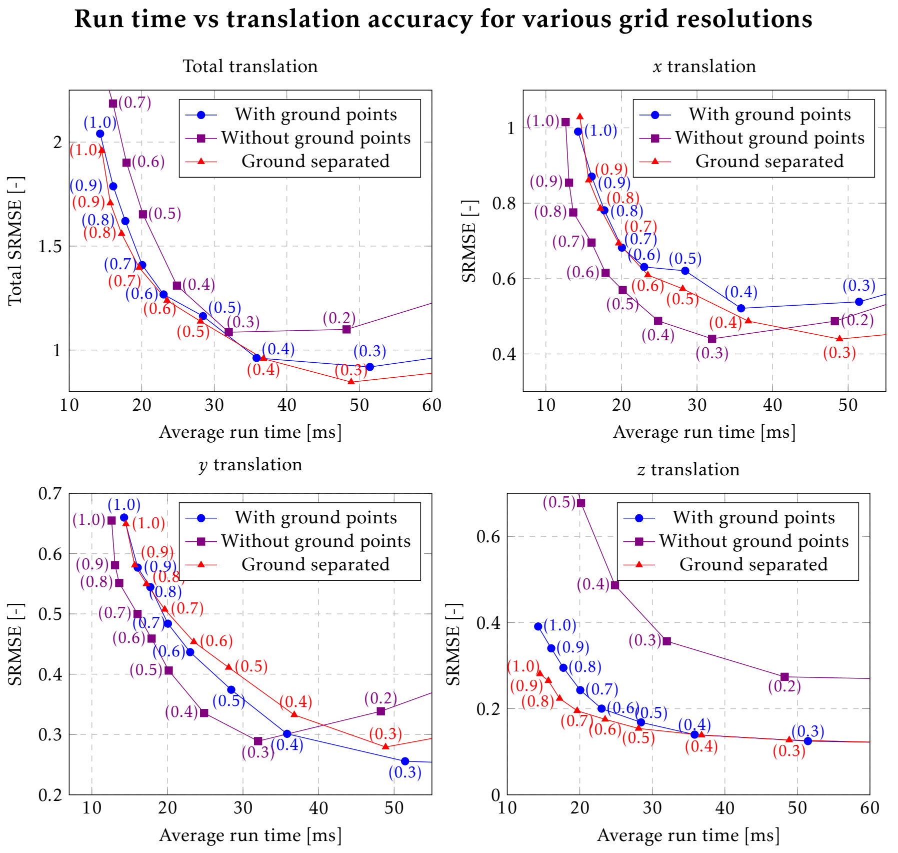


Figure 4.8: *Plots showing the average run time and translation accuracy for the point-to-plane ICP with different grid resolutions. The ideal solution is located in the bottom left corner. The grid resolution is indicated in parenthesis. Grid resolutions with longer process time and worse accuracy are not of interest and not shown in the plots. The RMSE values are scaled as described in 3.4.1. The Euler angles are defined from the previous point clouds coordinate frame*

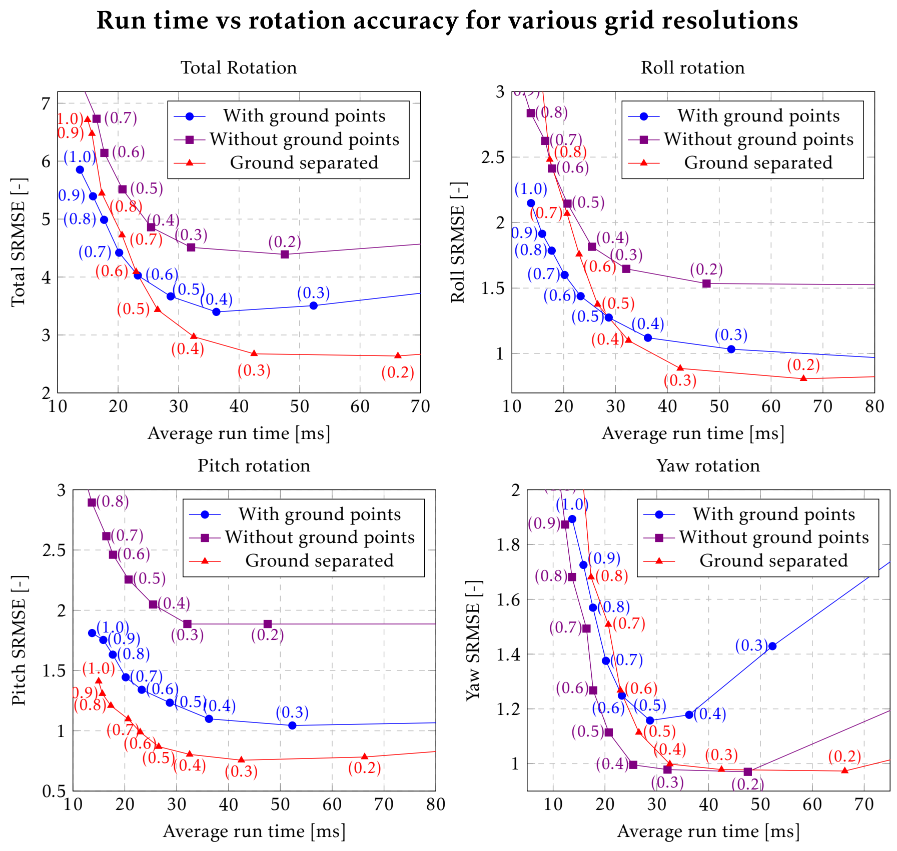


Figure 4.9: Plot showing the average run time and rotation accuracy for the point-to-plane ICP with different grid resolutions. The ideal solution is located in the bottom left corner. The grid resolution is indicated in parenthesis. Grid resolutions with longer process time and worse accuracy are not of interest and not shown in the plots. The RMSE values are scaled as described in 3.4.1.The Euler angles are defined from the previous point clouds coordinate frame

5 Scan matching

This chapter describes the implementation of the ICP scan matching, evaluates tuning parameters and compares the ICP variants. First, the sensitivity of the ICP to the initial alignments of the point clouds is investigated. Second, outlier rejection techniques for improving the ICP estimates are evaluated and discussed. Finally, the ICP variants are compared. Table 5.1 show the tuning parameters and their default values. Motivation for each parameter value is found in the corresponding sections. More detailed theoretical aspects of the algorithms are found in section 2.5.1, 2.5.2 and 2.6.2.

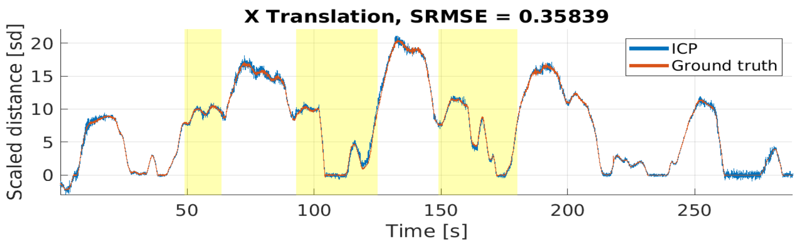
Table 5.1: *Parameters of the ICP algorithm and their default values.*

Parameter	**Default value**
Convergence translation threshold	1 [mm]
Maximum number of iterations	50 [-]
Number of neighbors used to calculate normal vectors	10 [-]

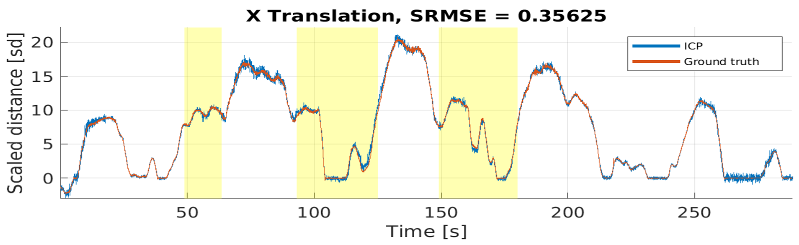
5.1 Initial guess of the transform

Scan matching algorithms and the ICP framework, in particular, are characterized as sensitive to the initial alignment of the point clouds to be matched. Since the algorithm lacks optimality, the algorithm can converge to a local minima. Therefore, an initial guess of the transform can be applied to the source point cloud before the scan matching is initiated. This initial guess could take the form of the previously estimated transform since they should be roughly equivalent, given a high frequency of the estimations. However, to make the scan matching algorithms robust and independent of previous estimates, an initial guess of no movement between the scans is provided as default.

To investigate the sensitivity to the initial alignment, a near-perfect initial guess based on the corresponding ground truth transform is provided for every frame and compared to the default case with no movement between the frames as an initial guess. The SRMSE for the x-translation when no movement is assumed between the scans is 0.35839, and the SRMSE is marginally better at 0.35625 when the ground truth is used, an increase of 0.59%. Consequently, since ground truth can be considered a near-perfect initial guess, the point-to-plane ICP algorithm would not benefit substantially by providing initial guesses via other motion sensors or previous estimates. Figure 5.1 shows the frame-to-frame estimations for x-translations. Only the estimations in x-translations are shown, but the same effect is true for the other five degrees of freedom.



(a) ICP estimations for the recorded data with no movement as an initial guess.



(b) ICP estimations for the recorded data with the ground truth as the initial guess.

Figure 5.1: *Comparison of the frame-to-frame estimations with a) no movement as the initial guess and b) ground truth as the initial guess. Yellow sections indicate the three roundabouts. The outlier rejection uses 15% trim.*

5.2 Point pair association

Point pair association is facilitated by constructing a kd-tree (described in section 2.4.1) of the target point cloud. The kd-tree is then used to find the nearest neighbor in the target point cloud for each point in the source point cloud. Multiple points from the source point cloud may be associated with the same point in the target point cloud. Nonetheless, only the pairs with the shortest shared distance are considered, and the rest are disregarded.

5.3 Outlier rejection

If the point pairs in the ICP framework are weighted equally, noisy points and points captured on dynamic objects can perturb the estimates. Hence, point pairs with an abnormal distance can be considered outliers. Outlier rejection is one way to discard the point pair outliers and improve the accuracy of ICP estimates. The procedure in this section differs from [35] in that the evaluations are done using 3D point cloud data collected in a dynamic outdoor environment instead of 2D data in an indoor environment. As presented in Table 5.2, all tested outlier rejection techniques can improve performance when tuned. The whole data set described in section 3.3.1 is used for the evaluation.

Table 5.2: *The best performing versions of the outliers rejection methods from Table 5.3, 5.4, and 5.5 gathered in one table.The first column indicates the method, the second column displays the setting, and the third to fifth column shows the performance change as a percentage. The percentages are calculated as* $\% = \frac{Outlier}{NoOutlier}$. *The best values in each column are highlighted with bold text.*

Method	Setting		SRMSE t_{tot}	SRMSE R_{tot}	Time
Median	d_{max}	$2\,\mu_{1/2}$	62.82%	43.03%	**96.71%**
RMT	e_1 / ϵ	1 m / 3σ	**60.98%**	**42.73%**	106.6%
Trim	ξ	15 %	62.70%	43.76%	97.02%

5.3.1 Fixed

The *fixed distance threshold* outlier removal method is the easiest to implement since the only required information is the distance between each point pair, which is calculated in the nearest neighbor search per default. It uses one tuning variable, the maximum allowed distance (d_{max}) between two points in a pair. The fixed distance method performs the worst of all the evaluated methods, as seen in Table 5.3.

At moderate high values of the fixed distance threshold of $d_{max} \in [3, 12]$ m (for this data set), the fixed method is relatively insensitive, giving a modest one to three percent performance change per meter. If d_{max} is set too low compared to the grid resolution of the voxel grid filter, most point pairs are initially too far away and are rejected. For this data set, $d_{max} \leq 0.1$ m rejects too many points.

Setting the threshold d_{max} to a higher value is more robust and increases performance compared to no outlier rejection since the most extreme outliers are rejected.

Table 5.3: *The table contains results for different settings for the fixed outlier rejection method The first column is the variant used. The second column is the settings. Columns three to five shows performance change as a percentage relative to not using any outlier rejection technique. The results are for total translation SRMSE, total rotation SRMSE, and average process time per frame. Percentages are calculated as* $\% = \frac{\text{Outlier}}{\text{No outlier}}$.

Method	Setting	SRMSE		Time
	d_{max}	T_{tot}	R_{tot}	
Fixed	5 m	87.80%	70.34%	96.19%
	1 m	78.70%	52.78%	**91.91%**
	0.4 m	74.48%	47.65%	93.02%
	0.2 m	81.95%	43.34%	99.5%
	0.1 m	279.34%	99.79%	131.14%
Median	1.5 $\mu_{1/2}$	69.44%	44.38%	105.41%
	2 $\mu_{1/2}$	**62.82%**	**43.03%**	96.71%
	3 $\mu_{1/2}$	71.2%	44.79%	94.56%

5.3.2 Median

A more adaptive way to set the threshold is to use the median method outlined in section 2.7. This approach considers that the distances between point pairs change every iteration of the ICP. According to [35], d_{max} should be three times the median, $\mu_{1/2}$. However, as seen in Table 5.3, the results are improved for this data set by setting d_{max} to two times the median. The difference could be because the original paper used a 2D lidar for indoor use, while the data set used in this thesis is collected with a high-end 3D lidar that generates more data. With more data, the outlier rejection method can be more restrictive and still have enough points to generate good transformation estimates. When tuned well, the median method performs comparably with the best-performing methods, as seen in Table 5.2

5.3.3 Relative motion threshold (RMT)

The RMT method is an extension of the fixed method. According to the literature presented in section 2.7 the RMT is one of the best performing outlier rejections methods. The experimental results indicate major improvements in the accuracy, as seen in Table 5.4. However, the RMT has a drawback it is the only outlier rejection method that consistently increases processing time when properly tuned. This is because the relative motion error e_n is iteratively refined and increases the number of iterations the ICP requires for convergence. The RMT with $e_1 = 1$ and $\epsilon = 2\sigma$ needs on average 11.03 iterations to converge compared to the trim method with $\xi = 15\%$ which requires 9.17 iterations on average. The RMT also has two tuning parameters, which complicates tuning.

The RMT method shares similarities with the fixed method in regards to the tuning of the maximum distance between point pairs d_{max}. Tuning ϵ excessively large will reduce the effectiveness of the RMT as it might let through more outliers. However, it will never wholly prevent the ICP from converging on a solution. If the maximum distance is too small, too many point pairs are rejected for the ICP to function accurately. Tuning e_1 to a high value does not impact the accuracy in any meaningful way but can increase the processing time. This is because it is refined each iteration. Thus having e_1 as a higher value simply increases the number of iterations needed to find the optimal d_{max}. Setting e_1 to a low value leads to similar performance degradation as having a too low value of d_{max} in the fixed method.

Table 5.4: *The table contains results for different settings for the RMT outlier rejection method. The first column is the method name. The second column is the settings. Columns three to five shows performance change as a percentage relative to not using any outlier rejection technique. The results are for total translation SRMSE, total rotation SRMSE, and average process time per frame. Percentages are calculated as* $\% = \frac{\text{Outlier}}{\text{No outlier}}$

	Setting	SRMSE		Time
	e_1 / ϵ [m]	T_{tot}	R_{tot}	
RMT	0.8 / 3σ	61.05%	43.10%	110.69%
	1 / σ	72.89%	61.10%	141.85%
	1 / 2σ	62.88%	47.19%	115.12%
	1 / 3σ	60.98%	**42.73%**	**106.60%**
	1 / 4σ	65.07%	43.05%	106.78%
	5 / 3σ	**60.95%**	43.10%	110.69%

According to [35], the minimum allowed distance ϵ should be tuned based on the sensor noise. They propose to use one standard deviation off the point pair distances in a static environment. However, one standard deviation is too restrictive for this data set and using three standard deviations delivers better results. The main advantage of using RMT should, in theory, be ϵ being sensor dependant and not dependent on the environment. Since the results in this thesis are only based on data from a single environment, this hypothesis cannot be tested.

5.3.4 Trim

The trim method, outlined in section 2.7 removes a percentage of the pairs with the greatest shared distance and has only one variable to tune, the trim percentage ξ. The method is easy to implement, increases performance and is robust when using smaller values of ξ. Setting ξ too large results in poor performance as too many points are discarded. A large ξ can also enhance the influence of dynamic objects in certain cases. If an object moves in the same direction and at the same speed as the vehicle, the points recorded on the dynamic object will represent the pairs with the shortest shared distance. Thus non-dynamic points will instead be removed, and the dynamic points become a more significant share of the retained point cloud.

Table 5.5: *The table contains results for different settings for the trim outlier rejection method. The first column is the variant used. The second column is the settings. Columns three to five shows performance change as a percentage relative to not using any outlier rejection technique. The results are for total translation SRMSE, total rotation SRMSE, and average process time per frame. Percentages are calculated as* $\% = \frac{Outlier}{No\ outlier}$

Method	Setting	SRMSE		Time
	ξ	T_{tot}	R_{tot}	
Trim	20%	66.08%	45.03%	100.43%
	15%	62.70%	**43.76%**	97.02%
	10%	65.51%	43.98%	**94.85%**
2Step Trim	ξ_1 / ξ_n	T_{tot}	R_{tot}	Time
	10% /20%	**62.07%**	44.88%	97.29%
	20% /30%	66.72%	46.25%	102.82%
	10% / 30%	62.39%	45.90%	99.25%
	10% / 40%	62.97%	47.46%	101.13%

Two-step trim

In order to minimize the shortcomings of the trim method, a *two-step trim* method is proposed. By rejecting a set percentage of the pairs with the greatest and smallest shared distance in the first iteration, the source point cloud is transformed to better match the majority of the points in the points cloud. The point cloud can then be trimmed with a more aggressive ξ_n without allowing dynamic objects to represent a larger percentage of the point pairs. However, the two-step trim method has the drawback of introducing an extra parameter to tune in ξ_1.

The two-step trim achieves the best accuracy for translation, and the basic trim method achieves the best accuracy for rotation. Results can be seen in Table 5.5 where the two methods give very similar performance. However, a bigger performance difference is exhibited when studying a smaller section of the data set with more dynamic objects. An example is visualized in Table 5.5 in the roundabout at 155-175s. The performance of the two-step trim with $\xi_1 = 10\%, \xi_n = 30\%$ performs 14% better than a standard trim with $\xi = 15\%$. This section of the data set was chosen as dynamic objects clearly affect the ICP's estimates during this roundabout.

5.3.5 Summary of outlier rejection

All of the outlier rejection methods can perform similarly well when adequately tuned. According to [35], the RMT should only be sensor dependent, which would make it a more general approach. However, since this data set only uses one type of sensor, this statement cannot be tested. The RMT is, however, the slowest, so if the processing time is a top priority, using the median method would be the recommendation.

5.4 Calculate Transformation

Depending on the ICP variant, the goal functions differ and are also solved differently. The point-to-point algorithm uses the singular value decomposition of the cross-covariance matrix to estimate the rotation, as described in section 2.5.1. The point-to-plane algorithm solves a linearized goal function through linear regression, as described in section 2.5.2. Neither of these algorithms uses explicit solvers as their optimization problems can be solved using basic linear algebra. The normal vectors in point-to-plane are calculated by using MATLAB's built-in function *pcnormals()* with 10 considered neighbors, based on PCA.

The plane-to-plane algorithm is based on the GICP framework outlined in section 2.6.2, and the minimization of the goal function requires a nonlinear solver. This is done using PCL in combination with MATLAB. First, all point clouds are preprocessed in MATLAB and saved as point cloud data files (.pcd). A C++ file then estimates all transformations using PCL's *GeneralizedIterativeClosestPoint()* class. The transformations are then saved as a CSV file and exported to MATLAB to uphold visual coherency when displaying results.

5.5 Stop criteria

Two stopping criteria are used in the point-to-point and point-to-plane ICP variants. First, the convergence criteria state that if t does not change more than a threshold for a consecutive number of iterations, the algorithm is assumed to have converged. The default threshold is set to less than 1 millimeter in change for three consecutive iterations, which is well within the confidence of the ground truth. Second, a maximum number of allowed iterations before the estimation is halted. The maximum number of allowed iterations is set to 50 as default, same as [44].

The plane-to-plane variant only requires one iteration below the threshold for convergence. This difference is due to the use of pre-built programs from PCL.

The stop criteria will affect the processing time and accuracy. The convergence criteria and the maximum number of iterations are set to strike a good balance between computational speed and accuracy in experiments.

5.6 Comparison of scan matching algorithms

To establish which variant of the ICP framework performs the best, the three major categories of ICP, point-to-point, point-to-plane and plane-to-plane are compared in Table 5.6. The point-to-point version is slightly faster than the point-to-plane. However, this comes at the cost of lower accuracy. It is clear that the point-to-plane version performs better than both plane-to-plane and point-to-point in terms of accuracy represented by the SRMSE.

The plane-to-plane variant is slightly worse than the point-to-plane in terms of accuracy. This is a surprising result as some of the literature, e.g., [44] would suggest that the plane-to-plane GICP version is the most accurate algorithm. This difference could be due to the data set in use being less urban than the data set used by others. The plane-to-plane variant gives more weight to the environmental geometry, and surface normals are easier to derive accurately for large planar surfaces such as facades.

The plane-to-plane variant lacks real-time properties in its current implementation, with an average process time of 1 141 ms, making it unsuitable for AV localization. When comparing the point-to-point version with the point-to-plane version, the point-to-plane variant is the superior algorithm overall, considering the small trade-off in process time and the significant gain in accuracy. Hence the point-to-plane variant is the proposed variant of choice.

Table 5.6: *The table shows the performance for the three ICP variants. The grid resolution v is set to 0.3 m and the outlier rejection is set to a fixed distance of 1 m for all methods. The RMT method allows for better performance, but the fixed method is used for a more fair comparison to the GICP plane-to-plane variant. A SRMSE value below 1 for the total translation is within the confidence of the ground truth. The Euler angle rotations are defined from the previous point clouds coordinate frame.*

SRMSE for the ICP variants			
Parameter	Point-to-point	Point-to-plane	Plane-to-plane
x	4.5777	**0.5042**	0.6895
y	1.0105	**0.3221**	0.3334
z	1.2413	**0.1124**	0.1674
Total translation	5.8320	**0.9387**	1.1903
γ (Roll)	1.3250	**0.9457**	1.4707
β (Pitch)	1.6095	**0.9721**	1.5410
α (Yaw)	3.9751	**1.1668**	1.3300
Total rotation	6.9095	**3.0846**	4.3417
Avg time [ms]	**35.64**	55.83	1141.8

6 Dynamic objects in the environment

This chapter highlights the errors caused by dynamic objects and strategies to minimize them. The scaled RMSE (SRMSE) is used to evaluate the performance. The scaling is for the purpose of confidentiality. The data used for the analysis in this chapter is a subset of the data set described in section 3.3.1. It starts at the 150-second mark and ends at the 210-second mark. This section is chosen as there are many errors caused by dynamic objects, which makes it easier to study the mitigation strategies. The data set contains the last roundabout (C) and a straight road which is important for the general applicability of the results. The complete route can be seen in Figure 3.3. When analyzing the effects of the mitigation strategies, a base case setting of the default point-to-plane ICP with a fixed outlier rejection where $d_{max} = 5\,\text{m}$ is used, visualized in Figure 6.1, and results are presented as a change of performance from the baseline.

The performance change is presented as a percentage which is calculated as $\% = \frac{\text{Dynamic Mitigation RMSE}}{\text{Baseline RMSE}}$. Thus lower percentages are desired.

6.1 Display of errors

The first question that needs to be addressed is whether the estimation errors are due to dynamic objects or something else. In Figure 6.1 during the 185-195 second interval, there is a clear bias toward over-estimations. During this section, several trucks drive in the oncoming lane towards the vehicle, which is illustrated in figure 6.3a. When the trucks are manually removed from the point clouds by deleting all points on the road, the translation accuracy improves, and the results can be seen in Figure 6.2. This improvement is because the trucks cause the average relative velocity of the point cloud to be higher than the vehicle speed, which

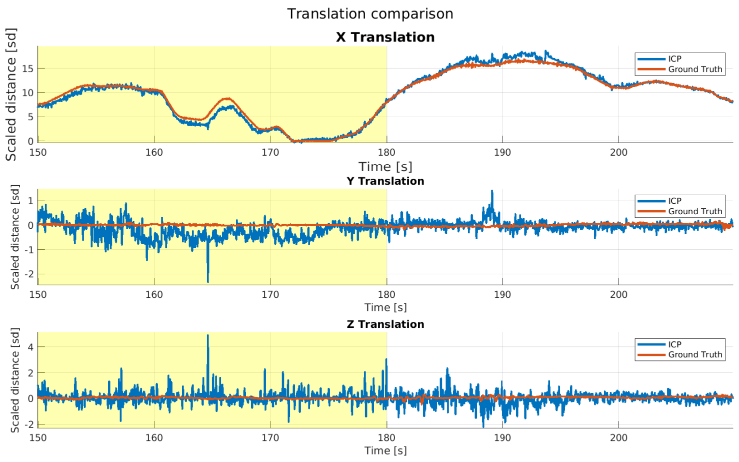


Figure 6.1: *Full point-to-plane ICP translation results with no reduction strategies for dynamic objects. Groundplane is removed, and the v is set to 0.3m. Outlier rejection is set to a fixed distance of 5 m. The yellow section indicates a roundabout being in close proximity.*

causes the ICP to estimate that the vehicle has moved forward more than it has, resulting in a higher value for translation estimates than the ground truth. Thus it can be concluded that dynamic objects indeed introduce errors in estimations.

Another part of the run with an apparent deviation from ground truth is in the roundabout around the 160-170s interval where cars are driving along with the test vehicle and hence introducing a bias towards lower value in the x-translation estimates. An example frame from this can section be seen in Figure 6.3b.

6.2 Road removal

A trivial method to reduce the errors caused by dynamic objects is to remove all non-ground points that are geometrically positioned on the road based on x- and y-coordinates since there is a high likelihood of them being dynamic. Vehicles such as trucks can have a relatively large surface area and therefore make up a large portion of the point cloud and have a greater influence on the estimates. The most common dynamic objects outside the road are pedestrians, which have a much smaller surface area than a vehicle and move much slower. Pedestrians are therefore not expected to cause as large errors in the estimates.

Distinguishing points positioned on the road is difficult without using a map or other sensory data. An elementary strategy is to assume that the road is straight along the vehicle and remains constant in its width. This heuristic, of course, does not work in all environments or data sets as it will remove non-road points

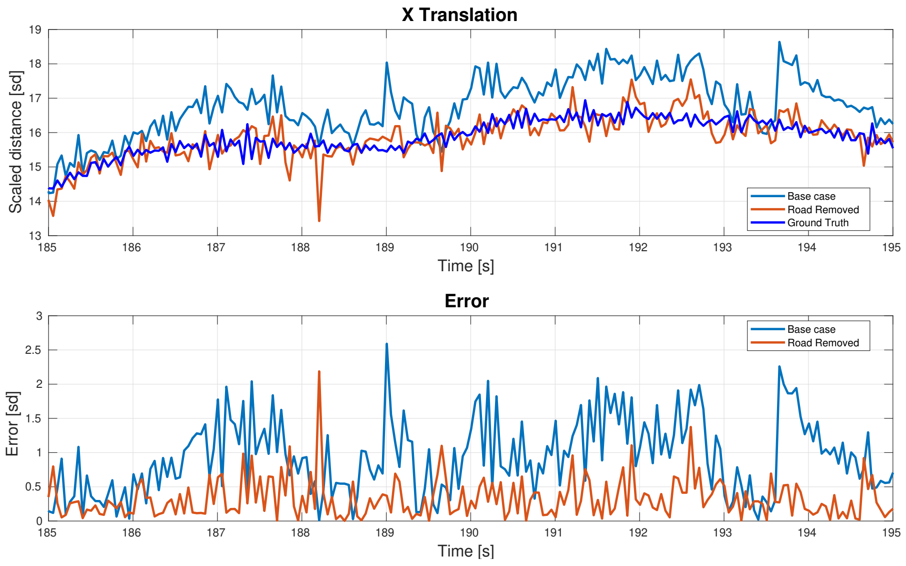
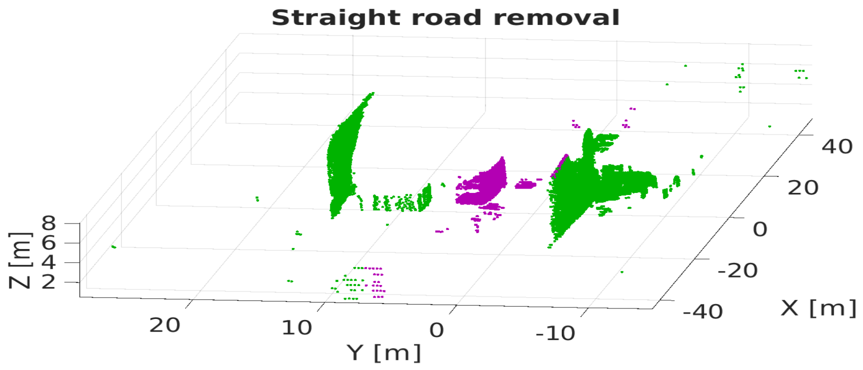


Figure 6.2: *Full point-to-plane* ICP *translation results with no reduction strategies for dynamic objects. Groundplane is removed, and the v is set to 0.3m. Outlier rejection is set to a fixed distance of 5 m.*

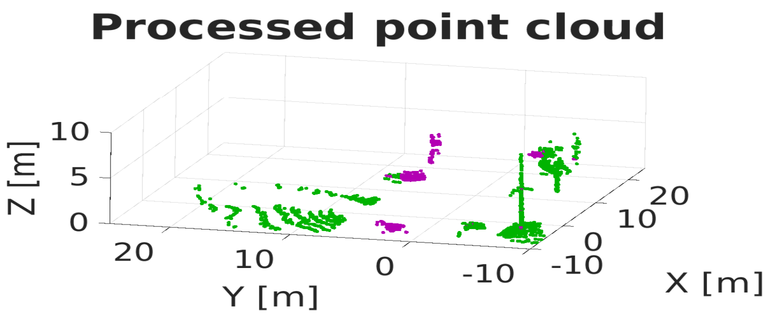
when the road curves, e.g., in a roundabout. When implementing this heuristic, it is assumed that the road is an infinite straight line with a width of seven m to the left of the car base and three m to the right of the car's base. This assumption is incorrect, but for certain environments, the information loss due to incorrect removal is compensated by filtering dynamic objects. The results for this strategy on our data set can be seen in Figure 6.4 and compared to the base case in Figure 6.1, notable gains in the x-translation estimates can be seen after the roundabout at the 180 s mark and comparable within the roundabout. By removing the road, the SRMSE for x-translation estimates improves (decreases) by 33.50% and the total SRMSE of 9.95%, which corresponds to a total translation SRMSE of 1.48. However, the rotation estimates worsen, increasing the SRMSE by 14.96%. The degradation in rotation estimates is primarily due to a degradation in the pitch rotation β where estimates get 32.23% worse. The trucks removed around the 190 s mark introduce errors in x-translation. However, since the trucks are depicted as large vertical planes, they add relevant information for the roll and pitch rotations estimations.

6.3 Outlier Rejection and Dynamic Objects

As mentioned in section 5.3, having an effective outlier rejection strategy can drastically improve the performance of ICP estimations. When examining the translation graphs, *e.g.* Figure 6.5, it is visible that biases identified as being caused by dynamic objects in section 6.1, such as the cars in the 163-168s interval



*(**a**) Illustration of a truck being removed at time 193 s.*



*(**b**) Illustration of removing a straight line from the test vehicle in the roundabout at time 166 s. The two purple clusters closest to origo are cars, and the third cluster is a fence.*

Figure 6.3: *The figure illustrates the effect on a point cloud when removing a straight corridor along the test vehicle. Purple illustrates the removed points. Green are the remaining points.*

and the trucks at the 185-195s interval, are significantly reduced. Hence, outlier rejection can be a potent strategy for reducing errors caused by dynamic objects.

Dynamic objects cause errors because the distance of the point pairs corresponding to dynamic objects is different compared to the static points, affecting the average and, conversely, the transformation. If the outlier rejection is tuned to discard point pairs with a shared distance that diverges from the bulk of the sam-

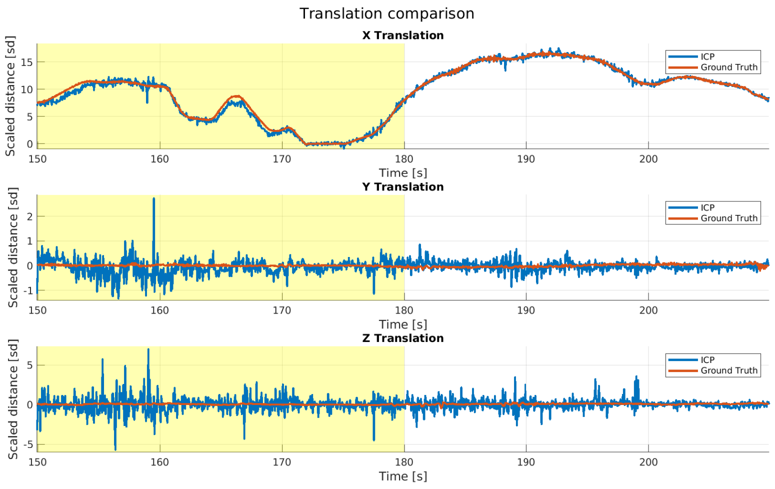


Figure 6.4: *Translation performance when removing all points in a straight line with 7 meters to the left and 3 meters to the right of the car base. Yellow sections indicate the car being in or close to a roundabout.*

ple, the point pairs corresponding to dynamic objects are rejected. A drawback of this approach is that it requires a significant portion of the point cloud to be static. If this is not the case, for example, if the vehicle is on a rural highway, the outlier rejection could instead enhance the errors.

Some outlier rejection methods, such as the trim and the fixed methods, will require different tuning for different environments depending on the vehicle velocity and how large a percentage of the point cloud is dynamic. Therefore, using the RMT, where the tuning parameter is sensor-based, is more robust for real-world applications.

A benefit of using outlier rejection to combat errors from dynamic objects is that it reduces errors caused by measurement noise and is relatively easy to implement. Using an RMT with $e_1 = 1\ m$ and $\epsilon = 3\sigma$ gives an SRMSE of 0.88, a 46.57% performance increase compared to the base case, visualized in Figure 6.5.

6.4 Geometric filtering

In attempts to account for dynamic environments in the odometry estimation, the geometric filtering methodology outlined by [15] and described in section 2.9 is evaluated. The method is evaluated because of its simplicity, ease of implementation, and prevalence in the literature.

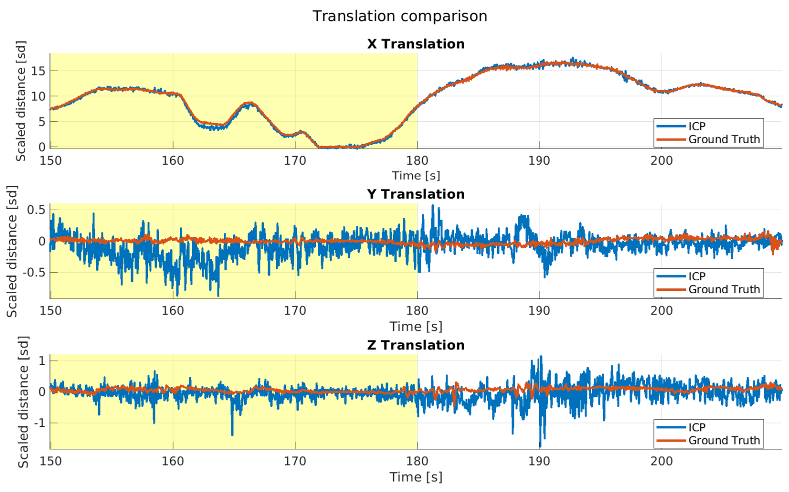


Figure 6.5: *Translation estimates for a plane-to-plane* ICP *with default settings. Outlier rejection method* RMT *is used with $e_1 = 1$ m and $\epsilon = 3\sigma$ m. The x-axis illustrates the time, and the y-axis illustrates the translation values. The blue line is the* ICP *estimates, and the ground truth is the red line. The yellow section indicates roundabout C.*

6.4.1 Parameter tuning and implementation

The procedure is to segment the point cloud into clusters and remove clusters that are geometrically smaller than a threshold. The clustering is performed as an additional step of the preprocessing. In [40], a geometric threshold of $10 \times 10 \times 4$ m is proposed, and this has been found to work the best for the data set used in this thesis.

The clusters are distinguished using MATLAB's *pcsegdist()*. The function is based on a principle of minimum Euclidean distance, which is elaborated in section 2.8.2. The method segments a point cloud based on a *minimum distance between clusters* and the threshold for the *minimum number of points* a cluster must consist of. Through experimentation visualized in Figure 6.6, the optimal settings have been experimentally found to be 1.8 m as the *minimum distance between clusters* and a *minimum number of points* of 35 for this data set.

Figure 6.7 shows the effect on a point cloud when removing clusters smaller than $10 \times 10 \times 4$ m. Note that the cars *i.e.*, dynamic objects are removed. The removal of smaller objects heuristically can, however, lead to unstable ICP estimations in environments where no large buildings or structures are present. A visualization of a problematic environment is seen in Figure 6.8 where only a light post remains after removing small objects.

A quality check and a sanity check have been implemented to counteract these issues. First, if the point cloud only has one cluster after filtering, the point cloud will be restored to the original non-segmented point cloud. This reduces the risk of objects being incorrectly matched, e.g., two different light posts being matched between frames. Second, an extra transformation estimation is performed using the original point cloud. If the translation from the segmented and filtered estimation differs more than 50% of the max speed from the estimates derived using the original point cloud, the transformation from the original point cloud is used. 50% of the max speed was deemed large enough to only reject unreasonable estimates.

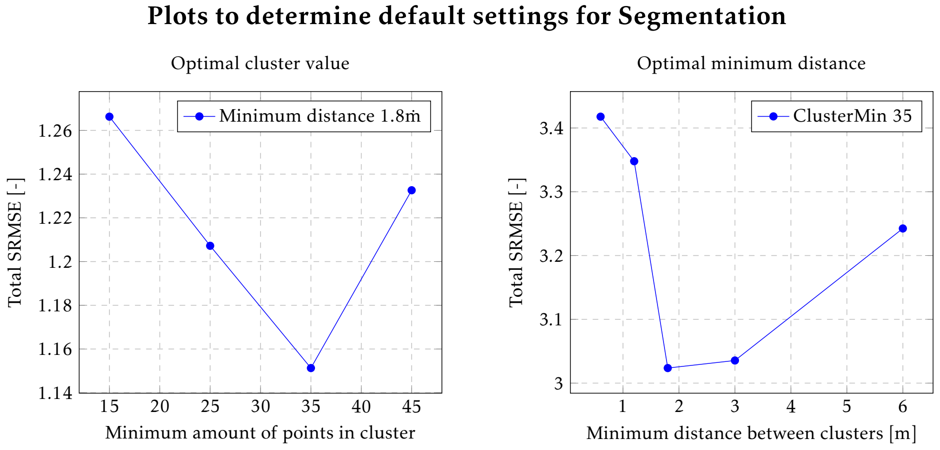
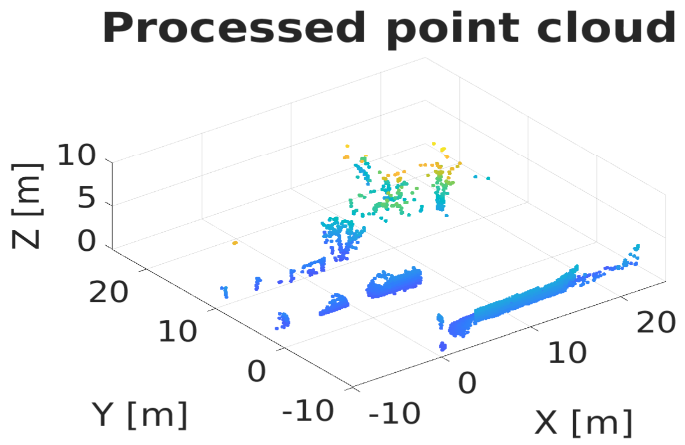


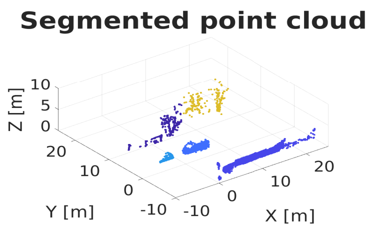
Figure 6.6: Plots showing experimental results deriving the default values for the segmentation settings. The y-value is the total SRMSE. The x-axis is the parameter values. The left plot has a fixed value for the minimum distance between two clusters set to 1.8 m and gives a optimal value for the minimum points in a cluster of 35. The right plot has a fixed minimum amount of point in a cluster set to to 35 and gives an optimal value for the distance between two clusters of 1.8 m.

6.4.2 Geometric filtering performance

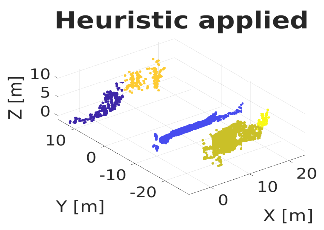
Geometric filtering with a size heuristic works best in urban environments where there are larger buildings or structures present at all times. Results for our data set can be seen in Figure 6.9 where it is visible that the segmentation handles some of the issues caused by dynamic objects by removing the bias seen in the base case at the 163-168 interval as well as some of the bias in the 185-195 interval. However, the frame-to-frame estimations with the geometric filtering method are more volatile than those without geometric filtering. This volatility is due to the remaining number of points in the segmented point cloud. When the majority of the point cloud is removed, the accuracy deteriorates. This correlation can be seen in Figure 6.10 where significant errors are present when there are fewer points in the point clouds. The reason is twofold: first, a lower number



(a) Point cloud after being preprocessed with a 0.3 m voxel filter and having ground plane removed.



(b) Point cloud in Figure 6.7a segmented with using Euclidean segmentation and default settings from section 6.4.1

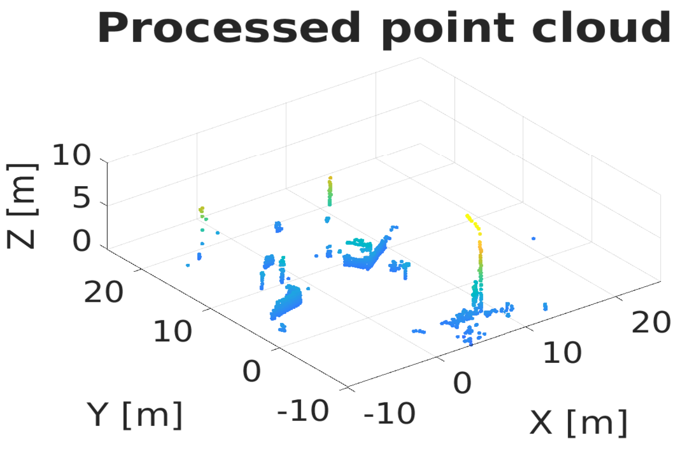


(c) Point cloud in Figure 6.7b processed with heuristic described in section 6.4.1.

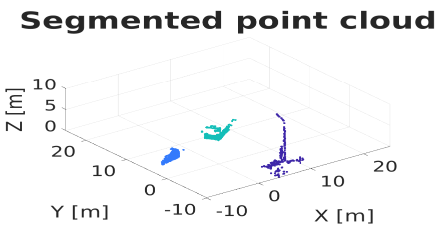
Figure 6.7: *A step-by-step illustration of the geometric filtering of a point cloud. Euclidean segmentation is applied in b), and smaller objects are removed in c).*

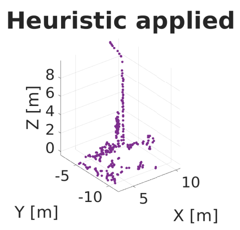
of points equals a smaller sample. Second, the loss of spatial information. The scan matching has better accuracy when using points from all directions. The accuracy will suffer if the remaining large clusters in the point cloud lie within a small angular portion of the space. The reason this approach works well for [40] is most likely because the surroundings where their data is collected contain many large structures, which are visible in their paper. The accuracy degradation due to the volatility in our data set is evident in the total translation SRMSE of 2.72, 65.57% worse than the baseline case.

When the sanity check of comparing the estimate from the segmented point cloud to the normally processed point cloud, described in section 6.4.1 is disregarded, the accuracy of some frames decreases drastically. This is visible in Figure 6.10 where significant errors are present, primarily throughout the roundabout, when the environment and, conversely, the point cloud lack larger structures. T



(a) Point cloud after being preprocessed with a 0.3 m voxel filter and having ground plane removed.





(b) Figure 6.8a segmented with using euclidean segmentation and default settings from section 6.4.1

(c) Figure 6.8b processed with heuristic described in section 6.4.1.

Figure 6.8: *A step-by-step illustration of the geometric filtering of a point cloud where only one large object is within the threshold. Euclidean segmentation is applied in b), and smaller objects are removed in c).*

Since the sanity check is required in every frame, the estimates must be calculated twice, once using the geometrically segmented point cloud and once with the original point cloud to filter the segmented ICP when the estimations become unreliable. The processing time is, therefore, greater than the other strategies accounting for dynamic objects presented in this thesis. The base case has an average process time of 43.1 ms per frame. The segmentation algorithm has an average process time of 91.9 ms per frame, an increase of 111.16%.

Geometric filtering does not perform as well in this data set as the outlier rejection method. However, it does not make the same assumption that the majority of the points are static. Therefore, the segmentation method could be the better performing strategy if an environment has a majority of dynamic points and some large static structures.

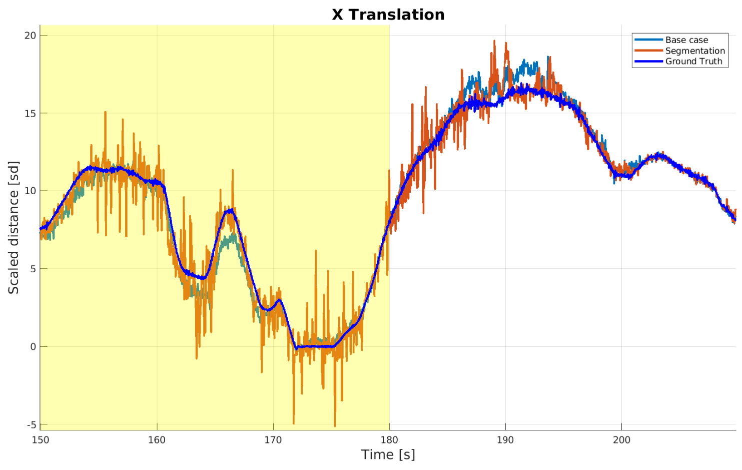


Figure 6.9: *Results of using segmentation to filter dynamic objects with a sanity check compared to the baseline case and the ground truth. The yellow section indicates the proximity of roundabout C.*

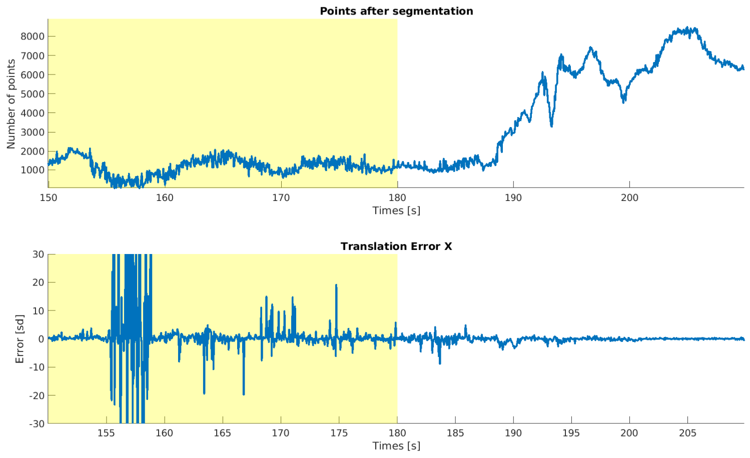


Figure 6.10: *The top plot shows the number of points in the point cloud after the segmentation has been applied with no sanity check. The bottom plot shows the error of each estimate. The yellow section indicates roundabout C.*

6.5 Combinations

As seen in Table 6.1, using a good outlier rejection method is a great way to deal with dynamic objects in an industrial suburban environment. It is, however, not flawless. Adding the road removal method increases the translation SRMSE slightly at the cost of rotation accuracy. Adding a RMT to the segmentation strategy improves its performance but not the other way around.

As mentioned in section 6.4, segmentation increases processing time since one extra scan matching is needed for the sanity check. When comparing the RMT method, visualized in Figure 6.5, with RMT and road removal combined, visualized in Figure 6.11, the results are similar with a slight improvement in the x-translation at around the 160 s mark. In general, combining these different strategies is not beneficial compared to one well-tuned outlier rejection method since robustness is lost. Therefore, the RMT alone is the proposed method to inhibit the influence of dynamic objects.

Table 6.1: *The table shows the performance change of different mitigation strategies for errors introduced by dynamic objects. X indicates the use of a method. The percentages indicate the performance change compared to the baseline case and are calculated as* $\% = \frac{\text{Dynamic Mitigation RMSE}}{\text{Baseline RMSE}}$.

RMT	Road Removal	Segmentation	T_{Tot}	R_{Tot}	Time Improvement
X			53.43 %	**62.49 %**	100.39 %
	X		90.05 %	114.96 %	91.11 %
		X	165.57 %	224.86 %	211.16 %
X	X		**50.97 %**	66.33 %	**83.42 %**
X		X	112.93 %	160.78 %	200.19 %
	X	X	185.09 %	236.78 %	196.75 %
X	X	X	132.17 %	177.87 %	175.95 %

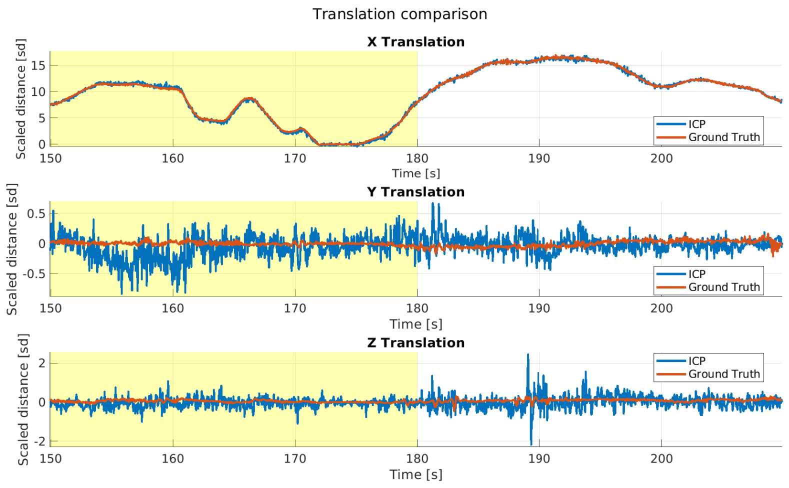


Figure 6.11: *The results for the combination of the road removal strategy with outlier rejection*

7 ICP-based odometry results

This chapter presents and discusses the performance of the best odometry solution. The chapter summarizes the findings in previous chapters and evaluates the solution when the best tuning parameters and methods are used. All presented results lack units in the error metric. This is by virtue of a scaling factor to keep absolute results confidential, as elaborated in section 3.4.1. The setup to achieve the solution is presented in chapter 3, 4, and 5.

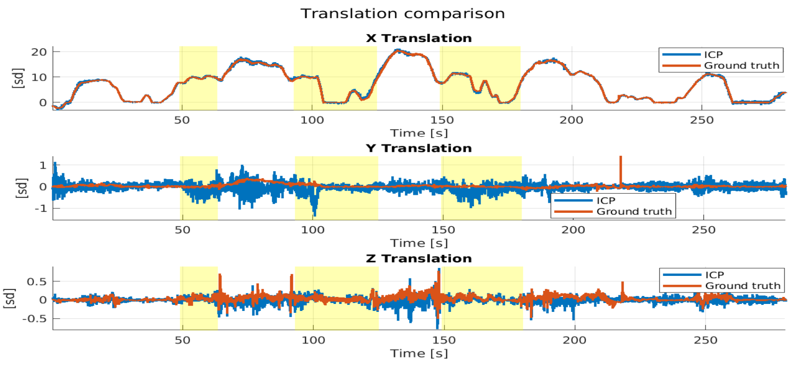
7.1 Performance of the ICP-based odometry

The solution uses a point-to-plane ICP variant to match the point clouds. Ground points and non-ground points are estimated separately. The grid resolution for the 3D voxel grid filter is set to 0.3 m. An RMT outlier rejection is used with $e_1 = 1$ and ϵ set to three standard deviations of the distance between point pairs in the static data set. Table 7.1 displays the SRMSE for all degrees of freedom.

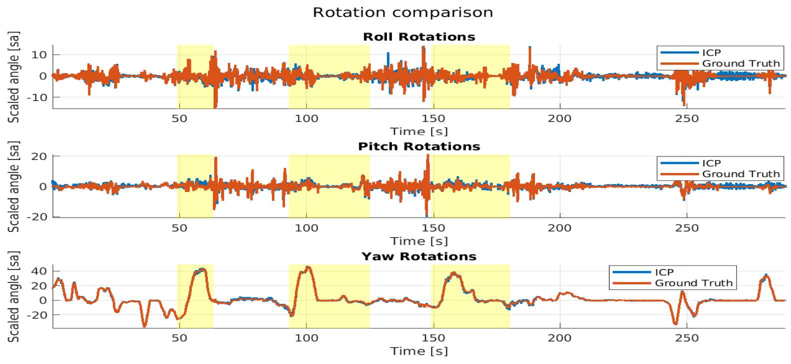
Figure 7.1 visualizes the frame-to-frame estimations compared to the ground truth. The ground truth exhibits some strange behaviors in short periods but can be considered reliable overall. An example of strange behavior is the single significant deviation in y-translation at around 220 seconds. The single large translation along the y-axis is an improbable event, given the vehicle used to record the data and probably an error in the recorded ground truth introduced by faulty GNSS measurements. However, since it is a single value out of 5 777 frames, the calculated accuracy of the solution should not suffer greatly. The most significant errors caused by dynamic objects can be seen in the y-translation at the beginning of roundabout B, where the estimates undershoot the ground truth for about 5 seconds.

Table 7.1: *The table shows the* SRMSE *for all degrees of freedom and the total translation and rotation for the proposed solution. The average time per frame is* 60.42 ms. *A value below 1 is within the accuracy of the ground truth for the translation estimates. The Euler angle rotations are defined from the previous point clouds coordinate frame*

SRMSE for the frame-to-frame estimates.			
Translation		Rotation	
X	0.3558	γ (Roll)	0.9552
Y	0.2451	β (Pitch)	1.0246
Z	0.1066	α (Yaw)	0.7840
Total	0.7075	Total	2.7638



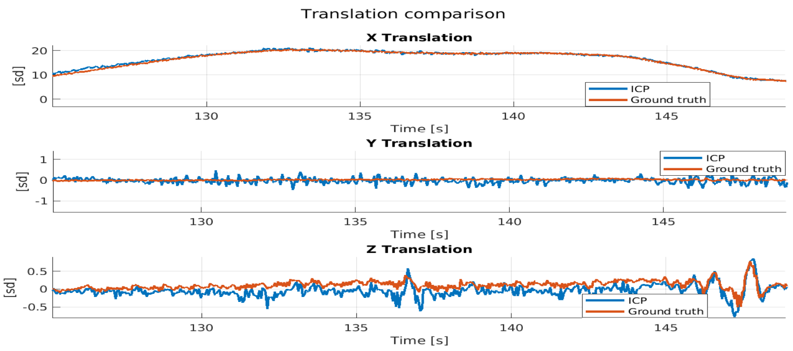
(a) *Translation estimates. The y-axis show the scaled distance.*



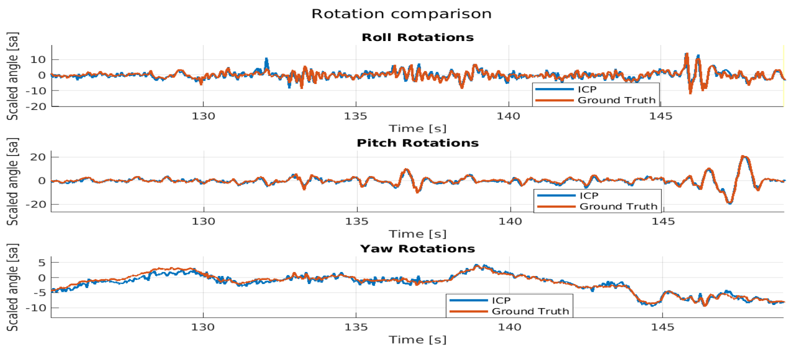
(b) *Rotation estimates. The y-axis show the scaled angle.*

Figure 7.1: *Comparison of the frame-to-frame estimates to the ground truth. The grid resolution of the 3D voxel grid filter is set to* 0.3 m. *The* RMT *outlier rejection is applied with $e_1 = 1$ m and ϵ is set to three standard deviations of the point pair distances for the static data set. Yellow sections indicate roundabouts. The values are scaled as described in section 3.4.1.*

Figure 7.2 shows the estimates compared to ground truth for the section between roundabout B and C. A section of the data is shown to visualize the characteristics of the ICP estimates more easily. The section between roundabout B and C is chosen because it is representative of the whole data set. The estimate undershoots the ground truth in z-translation for the majority of the section. The rest of the estimations follow the ground truth well. The undershooting for z-translation seems to correlate somewhat with the sections of higher speeds. Primarily the sections after the roundabouts. A hypothesis is that the vehicle dynamics at higher speeds tilt the vehicle, so the ICP estimates are altered.



(a) *Translation estimates for a portion of the data set. The y-axis shows the scaled distance.*



(b) *Rotation estimates for a portion of the data set. The y-axis shows the scaled angle.*

Figure 7.2: *Comparison of the frame-to-frame estimates to the ground truth. The grid resolution of the 3D voxel grid filter is set to 0.3 m. The RMT outlier rejection is applied with $e_1 = 1$ m and ϵ is set to three standard deviations of the point pair distances for the static data set. The plots show the section between roundabouts B and C. The values are scaled as described in section 3.4.1.*

Figure 7.3 shows a 2D projection of the global estimated path in the x,y-plane when adding all the frame-to-frame estimates compared to the ground truth. A significant drift is present. Consequently, the relative positioning solution can not be used independently but would contribute to a complete positioning system with adequate estimates to increase robustness. The drift is primarily due to errors in the rotation estimates. When adding the relative odometry estimates, a small error in a rotation estimate will skew the orientation going forward, leading to a larger drift.

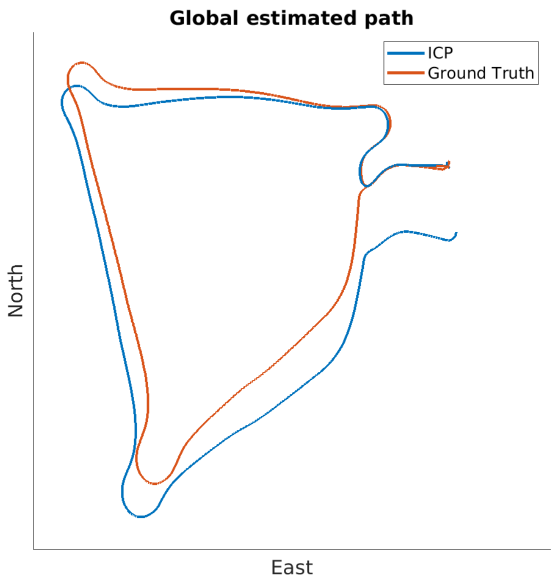


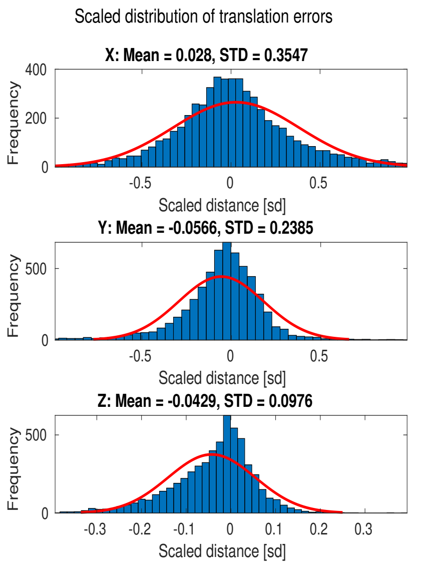
Figure 7.3: *Global estimated path for all the frame-to-frame estimations compared to the ground truth.*

7.2 Distribution of errors

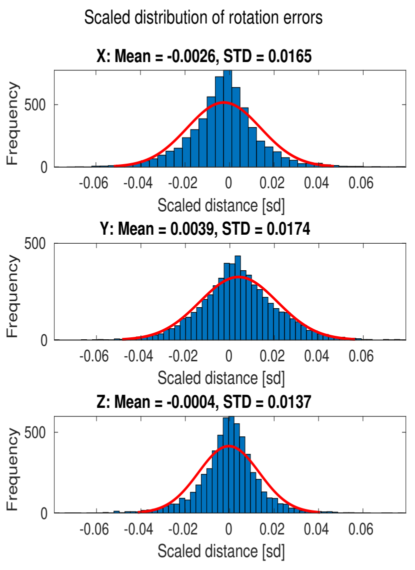
In Figure 7.4, it is shown that the frame-to-frame translation errors have a non-zero mean. This means that the proposed solution will drift if it is used independently with no absolute references. However, the frame-to-frame transformation estimates are intended as input to a more sophisticated localization system, using a Kalman filter variant in conjunction with measurements from an IMU. Fusing the measurements and adopting a motion model to the odometry will counteract the drift to a certain extent. However, if a GNSS outage period is too long, the localization system will still experience significant drift. During shorter outages, the added odometry estimates will provide adequate localization.

When using a Kalman filter, the expected covariance of errors must be provided, as the uncertainty of the measurements will dictate its influence on the final estimate. Figures 7.4c and 7.4d shows the distribution of errors when applying the ICP algorithm to lidar data where the vehicle is standing still. The distributions have Gaussian characteristics, which is necessary when implementing the estimates in a Kalman filter. The static data set is relatively small, and the biases could be due to the small sample size. It is to be noted that the units are scaled, effectively meaning that the mean and standard deviation do not have meters or degrees as units.

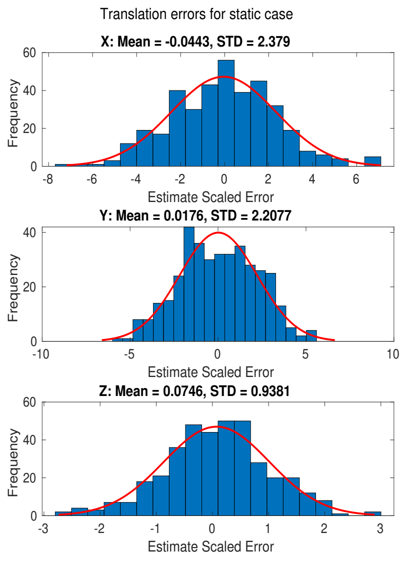
The distribution of errors when the vehicle stands still can be compared to the distributions of errors when driving the route in Figures 7.4a and 7.4b. The distributions when driving are similar in their bell-curved shape but exhibit more heavy-tailed characteristics than a Gaussian distribution. When driving, the distributions of errors for z-translation have the least bell curve shape as it is skewed and often undershoots the estimations. The translations along the x-and y-axis have a bell curve shape, but the distributions are more heavy-tailed due to larger errors, most likely introduced by dynamic objects. If the dynamic object were to be accounted for better, the distributions would likely be more Gaussian.



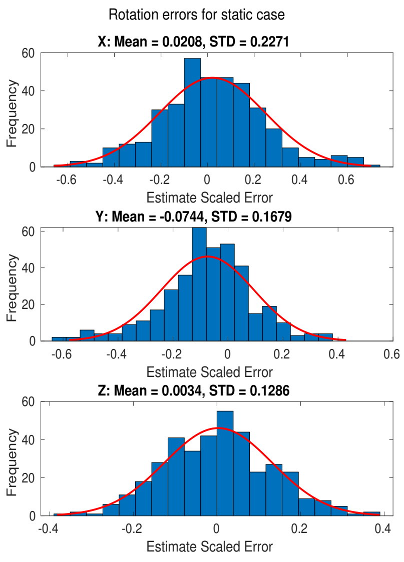
(a) Histogram of translation errors for the recorded data set when driving.



(b) Histogram of rotation errors for the recorded data set when driving.



(c) Histogram of translation errors for the static data set.



(d) Histogram of rotation errors for the static data set.

Figure 7.4: *Histogram of errors for the recorded data set when driving and standing still. The red line illustrates a Gaussian fit. The values are scaled as described in section 3.4.1.*

8 Concluding remarks

To make a positioning system more robust against GNSS-denied environments, ICP-based odometry solutions in 6 *DoF*, using lidar point cloud data and scan matching, have been evaluated. Dynamic objects can introduce errors in the estimates, and mitigation techniques have therefore been evaluated. This chapter concludes the thesis by answering the five stated research questions. Possible areas for future work are also highlighted. The RMSE values for the accuracy are scaled to create the SRMSE metric due to restrictions regarding provided ground truth data and its confidentiality.

8.1 Conclusions

A linearized point-to-plane variant was the best performing ICP algorithm when using the recorded industrial suburban data set. Compared to the most trivial point-to-point ICP variant, when using the fixed outlier rejection method, the best version decreased the root mean square error (RMSE) of the translation estimates by 83.90% and rotation estimates by 55.36%.

Using a tuned 3D voxel grid filter increased the accuracy of the scan matching and drastically improved the computational speed. The voxel grid filter proved essential to achieving real-time properties when using high-end lidars. For this data set, a grid resolution of 0.3 m allowed for the best overall accuracy with a reasonable process time. If a faster solution is desired, the grid resolution could be increased, with a decrease in accuracy as the trade-off.

Removing the ground plane from the point clouds increases the speed of the scan matching and improves estimates of the x-translation and yaw rotation. This improvement comes at the cost of the accuracy in z-translation and the accuracy of pitch and roll rotations. However, by dividing the lidar point clouds into two subsets, ground points and non-ground points, the scan matching can be applied to the two subsets separately, enhancing the most relevant information. The translations along the x-and y-axis and yaw rotation were first estimated using the non-ground points. The ground points were then aligned using this transformation, aiding the second estimate for the z-translation and pitch and roll rotations. This approach provided the best overall accuracy but increased the computational time compared to only using non-ground points.

For the industrial suburban data set, all outlier rejection techniques improved the accuracy of the estimates from the ICP algorithm. Partly due to low-quality point pair associations being removed and partly because the outlier rejection inhibits the influence of dynamic objects in the estimates. The outlier technique called relative motion threshold (RMT) allowed for the best overall accuracy. Compared to not using any outlier rejection technique, the RMT method improved the translation accuracy by 39.02% and the rotation accuracy by 57.27%. The computational speed was, however, slightly decreased by 6.6%. Outlier rejection is thus assessed as one of the most important aspects of ICP-based scan matching for accurate results.

Dynamic objects were shown to introduce errors in the scan matching estimates. Using an RMT reduced the errors from dynamic objects significantly. However, the RMT method requires the majority of the vehicle's surroundings to be static. The translation estimates from the test data were improved by combining an RMT with heuristically removing road-bound objects. This suggests that all errors caused by dynamic objects could not be resolved by an RMT alone. Filtering objects based on their geometrical size proved to remove many dynamic objects and thus reduce their influence on the estimates. For the recorded data set where some sections lack larger static structures, the method removed too much of the data, causing the method to be unstable. A RMT can reduce errors from dynamic objects if most of the environment is static. If the environment is urban, geometric segmentation can remove biases caused by dynamic objects. If the majority of the environment is dynamic and lacks large static structures, no suitable strategy has been found to reduce errors from dynamic objects.

The distribution of errors for the best-performing ICP algorithm is close to Gaussian. Therefore, it could be used as input to a localization system utilizing a Kalman filter variant. The SRMSE for the best odometry solution was 0.7075 for the total translation and 2.7638 for the total rotation. An SRMSE value for translation estimates below 1 is within the accuracy of the ground truth. Since the odometry solution is relative and the drift accumulates over time, it will only be able to fill an auxiliary role to a localization system in shorter GNSS outages.

8.2 Future work

In this thesis, we evaluated three different approaches to minimize the errors caused by dynamic objects. However, there are other approaches that were outside this thesis's scope. Mainly deep learning solutions and integration of other sensors such as radar which can measure the radial speed of objects. A detailed study comparing RMT with these techniques would be of high interest.

Only the spacial information of the points has been used in this thesis. However, many modern high-end lidars can also measure the intensity of the returning light. This information could potentially be integrated with the solution to filter out dynamic objects and thus improve the odometry estimates further.

As mentioned in section 7.2, the Gaussian characteristics of the errors indicate that it, in theory, should be possible to integrate our proposed solution with a filter-based localization system. However, as the saying goes, "in theory, theory and practice are the same. In practice, they are not". Hence, it would be beneficial to conduct further experiments on the validity of this hypothesis.

Bibliography

[1] A complete guide to lidar: Light detection and ranging, Oct 2021. URL https://gisgeography.com/lidar-light-detection-and-ranging/.

[2] Aeva. 4d lidar, May 2022. URL https://www.aeva.com/.

[3] Martin Alsfasser, Jan Siegemund, Jittu Kurian, and Anton Kummert. Exploiting polar grid structure and object shadows for fast object detection in point clouds. In Wolfgang Osten and Dmitry P. Nikolaev, editors, *Twelfth International Conference on Machine Vision (ICMV 2019)*, volume 11433, pages 111 – 118. International Society for Optics and Photonics, SPIE, 2020. doi: 10.1117/12.2557043. URL https://doi.org/10.1117/12.2557043.

[4] Eduardo Arnold, Omar Y. Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795, 2019. doi: 10.1109/TITS.2019.2892405.

[5] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987. doi: 10.1109/TPAMI.1987.4767965.

[6] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi: 10.1109/34.121791.

[7] Sergey Bochkanov. Nearest neighbor search with kd-trees - alglib, May 2022. URL https://www.alglib.net/other/nearestneighbors.php.

[8] Igor Bogoslavskyi and Cyrill Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 163–169, 2016. doi: 10.1109/IROS.2016.7759050.

[9] Kenny Chen, Brett T. Lopez, Ali-akbar Agha-mohammadi, and Ankur Mehta. Direct lidar odometry: Fast localization with dense point clouds. *IEEE Robotics and Automation Letters*, 7(2):2000–2007, 2022. doi: 10.1109/LRA.2022.3142739.

[10] Jie Cheng, Dong He, and Changhee Lee. A simple ground segmentation method for lidar 3d point clouds. In *2020 2nd International Conference on Advances in Computer Technology, Information Science and Communications (CTISC)*, pages 171–175, 2020. doi: 10.1109/CTISC49998.2020.00034.

[11] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The trimmed iterative closest point algorithm. In *2002 International Conference on Pattern Recognition*, volume 3, pages 545–548 vol.3, 2002. doi: 10.1109/ICPR.2002.1047997.

[12] On-Road Automated Driving (ORAD) committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, apr 2021. URL `https://doi.org/10.4271/J3016_202104`.

[13] Xiangwei Dang, Xingdong Liang, Yanlei Li, and Zheng Rong. Moving objects elimination towards enhanced dynamic slam fusing lidar and mmw-radar. In *2020 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4, 2020. doi: 10.1109/ICMIM48759.2020.9298986.

[14] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computation Geometry: Algorithms and Applications, 3rd Edition*. Springer, 2008.

[15] Jean-Emmanuel Deschaud. IMLS-SLAM: Scan-to-model matching based on 3d data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2480–2485, 2018. doi: 10.1109/ICRA.2018.8460653.

[16] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the segmentation of 3d lidar point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805, 2011. doi: 10.1109/ICRA.2011.5979818.

[17] Sébastien Druon, M.J. Aldon, and André Crosnier. Color constrained ICP for registration of large unstructured 3d color data sets. *International Conference on Information Acquisition*, 0:249–255, 08 2006. doi: 10.1109/ICIA.2006.306004.

[18] Mahdi Elhousni and Xinming Huang. A survey on 3d lidar localization for autonomous vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1879–1884, 2020. doi: 10.1109/IV47402.2020.9304812.

[19] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[20] Aleksey Golovinskiy and Thomas Funkhouser. Min-cut based segmentation of point clouds. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 39–46, 2009. doi: 10.1109/ICCVW.2009.5457721.

[21] Fredrik Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur, 2010.

[22] Xian-Feng Han, Jesse S Jin, Ming-Jie Wang, Wei Jiang, Lei Gao, and Liping Xiao. A review of algorithms for filtering the 3d point cloud. *Signal Processing: Image Communication*, 57:103–112, 2017.

[23] Andrew J. Hawkins. Cruise is now testing fully driverless cars in San Francisco. *The Verge*, Dec 2020. URL `https://www.theverge.com/2020/12/9/22165597/cruise-driverless-test-san-francisco-self-driving-level-4`.

[24] Richard Hoffman and Anil K. Jain. Segmentation and classification of range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):608–620, 1987. doi: 10.1109/TPAMI.1987.4767955.

[25] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, page 71–78, New York, NY, USA, 1992. Association for Computing Machinery. ISBN 0897914791. doi: 10.1145/133994.134011. URL `https://doi.org/10.1145/133994.134011`.

[26] Feng Huang, Weisong Wen, Jiachen Zhang, and Li-Ta Hsu. Point wise or feature wise? a benchmark comparison of publicly available lidar odometry algorithms in urban canyons. *IEEE Intelligent Transportation Systems Magazine*, pages 2–20, 2022. doi: 10.1109/MITS.2021.3092731.

[27] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1310–1320, 2020. doi: 10.1109/CVPR42600.2020.00139.

[28] K. Kanatani. Analysis of 3-d rotation fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):543–549, 1994. doi: 10.1109/34.291441.

[29] Klaas Klasing, Dirk Wollherr, and Martin Buss. A clustering method for efficient segmentation of 3d laser data. In *2008 IEEE International Conference on Robotics and Automation*, pages 4043–4048, 2008. doi: 10.1109/ROBOT.2008.4543832.

[30] Huikai Liu, Yue Zhang, Linjian Lei, Hui Xie, Yan Li, and Shengli Sun. Hierarchical optimization of 3d point cloud registration. *Sensors*, 20(23), 2020. ISSN 1424-8220. doi: 10.3390/s20236999. URL `https://www.mdpi.com/1424-8220/20/23/6999`.

[31] Ze Liu, Yingfeng Cai, Hai Wang, Long Chen, Hongbo Gao, Yunyi Jia, and Yicheng Li. Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2021. doi: 10.1109/TITS.2021.3059674.

[32] Kok-Lim Low. Linear least-squares optimization for point-to-plane ICP surface registration. Technical Report TR04-004, Department of Computer Science, Chapel Hill, University of North Carolina, 2004.

[33] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. In *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 225–230, 2013. doi: 10.1109/RAM.2013.6758588.

[34] Ouster. Os0 ultra wide lidar sensor, May 2022. URL `https://ouster.com/products/scanning-lidar/os0-sensor`.

[35] François Pomerleau, Francis Colas, François Ferland, and François Michaud. Relative motion threshold for rejection in ICP registration. In Andrew Howard, Karl Iagnemma, and Alonzo Kelly, editors, *Field and Service Robotics*, pages 229–238, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-13408-1.

[36] Rahul Raguram, Jan-Michael Frahm, and Marc Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *European Conference on Computer Vision*, pages 500–513. Springer, 2008.

[37] Thinal Raj, Fazida Hanim Hashim, Aqilah Baseri Huddin, Mohd Faisal Ibrahim, and Aini Hussain. A survey on lidar scanning mechanisms. *Electronics*, 9(5), 2020. ISSN 2079-9292. doi: 10.3390/electronics9050741. URL `https://www.mdpi.com/2079-9292/9/5/741`.

[38] E Recherche, Et Automatique, Sophia Antipolis, and Zhengyou Zhang. Iterative point matching for registration of free-form curves. *Int. J. Comput. Vision*, 13, 07 1992.

[39] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206, 05 2019.

[40] Nivedita Rufus, Unni Krishnan R. Nair, A. V. S. Sai Bhargav Kumar, Vashist Madiraju, and K. Madhava Krishna. SROM: Simple real-time odometry and mapping using lidar data for autonomous vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1867–1872, 2020. doi: 10.1109/IV47402.2020.9304577.

[41] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001. doi: 10.1109/IM.2001.924423.

[42] Radu Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI - Künstliche Intelligenz*, 24, 11 2010. doi: 10.1007/s13218-010-0059-6.

[43] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. IEEE.

[44] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[45] Jacopo Serafin, Edwin Olson, and Giorgio Grisetti. Fast and robust 3d feature extraction from sparse point clouds. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4105–4112, 2016. doi: 10.1109/IROS.2016.7759604.

[46] Tixiao Shan and Brendan Englot. LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018. doi: 10.1109/IROS.2018.8594299.

[47] Zhihao Shen, Huawei Liang, Linglong Lin, Zhiling Wang, Weixin Huang, and Jie Yu. Fast ground segmentation for 3d lidar point cloud based on jump-convolution-process. *Remote Sensing*, 13(16), 2021. ISSN 2072-4292. doi: 10.3390/rs13163239. URL `https://www.mdpi.com/2072-4292/13/16/3239`.

[48] Gabriel Staples. rosbag. URL `http://wiki.ros.org/rosbag`.

[49] P. H. S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:2000, 2000.

[50] Wei Xu and Fu Zhang. FAST-LIO: A fast, robust lidar-inertial odometry package by tightly-coupled iterated Kalman filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, 2021. doi: 10.1109/LRA.2021.3064227.

[51] Dimitris Zermas, Izzat Izzat, and Papanikolopoulos Nikolaos. Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5067–5073. IEEE, 2017.

[52] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181, 2015. doi: 10.1109/ICRA.2015.7139486.

[53] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.

[54] Shibo Zhao, Zheng Fang, HaoLai Li, and Sebastian Scherer. A robust laser-inertial odometry and mapping method for large-scale highway environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1285–1292, 2019. doi: 10.1109/IROS40897.2019.8967880.

[55] Xin Zheng and Jianke Zhu. Efficient lidar odometry for autonomous driving. *IEEE Robotics and Automation Letters*, 6(4):8458–8465, 2021. doi: 10.1109/LRA.2021.3110372.

[56] Bing Zhou and Ran Huang. Segmentation algorithm for 3d lidar point cloud based on region clustering. In *2020 7th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*, pages 52–57, 2020. doi: 10.1109/ICCSS52145.2020.9336862.