Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2023

Robust Graph SLAM in Challenging GNSS Environments Using Lidar Odometry

Jesper Sundström and Alfred Åström

LINKÖPING UNIVERSITY

Master of Science Thesis in Electrical Engineering

Robust Graph SLAM in Challenging GNSS Environments Using Lidar Odometry

Jesper Sundström and Alfred Åström

LiTH-ISY-EX--23/5626--SE

Supervisor: **Joel Nilsson**
ISY, Linköpings universitet
Bianca Otake
Company supervisor
Samuel Berndtsson
Company supervisor
Jimmy Hammenstedt
Company supervisor

Examiner: **Gustaf Hendeby**
ISY, Linköpings universitet

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

Abstract

Localization is a fundamental part of achieving fully autonomous vehicles. A localization system needs to constantly provide accurate information about the position of the vehicle and failure could lead to catastrophic consequences. *Global navigation satellite systems* (GNSS) can supply accurate positional measurements but are susceptible to disturbances and outages in environments such as indoors, in tunnels, or nearby tall buildings. A common method called *simultaneous localization and mapping* (SLAM) creates a spatial map and simultaneously determines the position of a robot or vehicle. Utilizing different sensors for localization can increase the accuracy and robustness of such a system if used correctly. This thesis uses a graph-based version of SLAM called graph SLAM which stores previous measurements in a factor graph, making it possible to adjust the trajectory and map as new information is gained. The best position state estimation is gained by optimizing the graph representing the log-likelihood of the data. To treat GNSS outliers in a graph SLAM system, robust optimization techniques can be used, and this thesis investigates two techniques called *realizing, reversing, recovering* (RRR), and *dynamic covariance scaling* (DCS). High-end GNSS and Lidar sensors are used to gather a data set on a suburban public road. Information about the position and orientation of the vehicle are inferred from the data set using graph SLAM together with robust techniques in three different scenarios. The scenarios contain disturbances called multipathing, Gaussian disturbances, and outages. A parameter study examines the free parameters Φ in DCS and the p-value in the RRR method. The localization performance varies less when changing the free parameter in RRR than DCS. The localization performance from RRR is consistent for most values of p. DCS shows greater variation in the localization performance for different values of Φ. In the tested cases, results conclude that Φ should be set to 2.5 for the most consistent localization across all states. RRR performed best with a p-value set to 0.85. A lower value led to too many discarded measurements which decreased performance. DCS outperforms RRR across the tested scenarios but further testing is needed to determine whether RRR is better suited for handling larger errors.

Sammanfattning

Lokalisering är en fundamental del i att uppnå självkörande fordon. Lokaliseringssytemets uppgift är att kontinuerligt förse exakt information om fordonets position och vid fel kan detta leda till katastrofala följder. *Global navigation satellite systems* (GNSS) används ofta i ett lokaliseringssystem för att uppnå exakta positionsmätningar men i vissa miljöer så som parkeringshus, tunnlar eller storstäder kan störningar uppstå. Genom att förlita sig på fler typer av sensorer kan lokaliseringen bli mer noggrann och robust mot störnignar. En vanlig metod som kan skatta ett fordons position och samtidigt skapa en karta över omgivningen är *simultaneous localization and mapping* (SLAM). I detta examensarbete används graph SLAM, en version an SLAM som utnyttjar en faktorgraf för att representera mätvärden och sedan estimera position av fordonet. Robusta metoder kan användas inom SLAM för hantering av felaktiga mätningar i ett grafbaserat SLAM nätverk, och här undersöks två metoder, *realizing*, *reversing*, *recovering* (RRR) och *dynamic covariance scaling* (DCS). Data från GNSS och Lidarsensorer av hög kvalitet samlades in på en offentlig väg i stadsmiljö. I tre olika scenarion beräknas testfordonets position och orientering med graph SLAM tillsammans med de två robusta metoderna som undersöks. Scenarion utgör fall med olika typer av störningar som agerar på GNSS mätningarna. Störningarna är av typerna multipath, Gaussiskt brus, samt avbrott. DCS presterar bättre jämfört med RRR under de tester som utförts. En parameterstudie har utförts som undersöker parametern Φ i DCS och p i RRR. När Φ varieras i DCS ger det en större skillnad på resultatet än när p varieras i RRR. Detta indikerar att det är lättare att hantera och använda RRR optimalt. Trots att DCS presterar bättre än RRR i de testade fallen, krävs vidare undersökning för att besluta om RRR hanterar stora fel bättre än DCS. De bästa inställningarna visades vara 2,5 för Φ i DCS och större än 0,85 för p i RRR.

Acknowledgments

We would like to express our gratitude to the people involved with our work on this thesis. Thank you, Bianca, Samuel, and Jimmy for offering your guidance and help throughout all stages of the thesis process, and for providing innovative ideas for us to expand upon. Your technical knowledge and expertise have also been key to the completion of this thesis.

Also, we would like to thank our university supervisor Joel Nilsson for his engaging discussions and excellent proofreading. And a thanks to our examiner Gustaf Hendeby for valuable input and contributions towards improving our thesis.

Finally, we would like to thank our family and friends for their encouragement and support during our academic studies.

Linköping, 2023
Jesper Sundström and Alfred Åström

Contents

Notation

Acronyms and Abbreviations

Abbreviation	Definition
SLAM	Simultaneous Localization And Mapping
GNSS	Global Navigation Satellite System
Lidar	Light Detection and Ranging
Radar	Radio Detection and Ranging
SC	Switch Constraint
DCS	Dynamic Scaling Covariance
IMU	Internal Measurement Unit
INS	Inertial Navigation System
FoV	Field of View
ToF	Time of Flight
RMSE	Root Mean Square Error
SRMSE	Scaled Root Mean Square Error
PCL	Point Cloud Library
UTM	Universal Transverse Mercator

1 Introduction

This introductory chapter gives a brief background of the localization problem for automated vehicles. Followed by the aim, research questions, scientific approach, scope, and contributions for this thesis.

1.1 Background

As automated vehicles make their way into the transportation sector the requirements for safety are very high. Many systems depend upon accurate information of the vehicle position, or *localization*, for example, perception, path planning, and motion control. Erroneous localization propagates to these systems impairing performance and causing safety risks. It is also of high importance that the vehicle is aware of its surroundings in order to navigate the world. *Simultaneous localization and mapping* (SLAM) is a method that localizes the vehicle and at the same time creates a spatial map using measurements from the surroundings.

In conditions such as indoors, in cities with tall buildings, or in tunnels, *global navigation satellite systems* (GNSS) can fail to consistently provide the needed localization accuracy. In the worst case, reception is lost and GNSS cannot provide any information at all. GNSS signals can bounce on nearby structures before being registered by the receiving sensor, which means the signal has traveled further, giving an incorrect range to the satellite. This effect is called multipath, adding a bias to the position measurement.

The unreliability of GNSS in these conditions necessitates other localization methods of the vehicle, especially in more complex and urban environments. *Light Detection and Ranging* (Lidar) sensors scan the environment and measures the

range from the sensor to visible objects in the surrounding. The result is a high-resolution 3D scan in the form of a point cloud which can be used for odometry estimation through scan matching. Lidar odometry is a promising complement to GNSS in the search for robust localization since it can provide accurate motion estimation based on the environment. Furthermore, lidar is already a widely used sensor for autonomous vehicles for perception and object detection. The drawback of lidar odometry is the accumulation of small estimation errors leading to drift over time if no adjustments are made.

Variations of SLAM exist, with classic probabilistic methods utilizing Kalman filters and particle filters for determining the most likely solution. One SLAM method that has gained increased popularity in the last two decades is graph SLAM which models the problem as a factor graph that is solved using nonlinear least squares methods [1]. Studies have been conducted regarding methods on how to handle outliers in graph SLAM. A method for handling incorrect loop closure detections which is when a previously visited location is detected, called *Switchable Constraints* (SC) adds an additional variable for disabling measurements [2]. Agarwal et al. [3] introduces a method called *Dynamic Covariance Scaling* (DCS) which scales the covariance of measurements that creates large errors in the solution. Another approach for loop closures called *Realizing*, *Reversing*, *Recovering* (RRR) uses a consistency-based verification step to determine if a measurement is correct or faulty [4]. Other papers apply the same methods to GNSS measurements where tightly coupled GNSS pseudorange measurements are turned on or off by adding or removing them in the factor graph [5, 6].

Other papers have sought to compare and evaluate robust methods in the context of loop closures and graph SLAM. Loop closure is an artificial measurement based on place recognition along a traveled path in graph SLAM. Sünderhauf and Protzel [7] compares three different approaches for robust graph SLAM which is SC, max-mixtures (MM), and RRR for loop closure measurements. The conclusion was that none of the methods worked perfectly for every scenario and that more research on robust methods in factor graphs is necessary. Another comparative study Latif et al. [8] further explores robust methods for graph SLAM and includes the method DCS. It concludes that an important aspect of the robust methods is how outliers are treated. MM and RRR make a binary decision by removing measurements deemed as outliers based on statistical tests, while DCS and SC continuously scales the weight of measurement in the final solution. They argue that in the context of loop closures, the binary methods are better fitted if the results from the graph SLAM are to be used for navigation.

RRR and DCS have been shown in previous studies as well performing and popular for reducing the effects of outliers. In this thesis RRR and DCS are studied when applied to GNSS measurements in a graph SLAM system. The methods were chosen as representatives of robust techniques using either binary or non-binary decision-making.

1.2 Thesis Aim and Research Questions

This thesis seeks to compare the methods RRR and DCS when applied on disturbed GNSS measurements when combined with lidar measurements in a multiple lidar sensor setup.

The questions that this thesis aims to answer are:

1. How can GNSS be combined with lidar to provide robust localization in the presence of GNSS disturbances?

2. How does the localization performance of robust techniques in graph SLAM compare in the presence of disturbances?

3. Are binary or non-binary robust techniques more suitable for GNSS measurements in a graph SLAM system?

1.3 Scientific Approach

An open source graph SLAM framework is used in order to estimate the position of a vehicle traveling on a public road in a suburban setting. The vehicle is equipped with several sensors such as lidar, radar, and GNSS receivers and has a dedicated navigation system. The collected data is used in three scenarios in which GNSS struggles to provide accurate localization. The first scenario consists of a section from the data set where multipath disturbances affects one receiver in the altitudinal direction. The other two scenarios consist of a different section of the data set with artificially added disturbances and an outage in order to compare the resilience of robust optimization techniques with different noise characteristics. The two different disturbances are multipath in the longitudinal and lateral directions and Gaussian noise. These disturbances are added before and after the outage to simulate entering and exiting a tunnel.

A performance baseline is established with the graph SLAM framework for all scenarios, and the robust optimization techniques are evaluated in relation to the baseline. The already existing internal navigation system is used as ground truth.

The performance of the robust methods RRR and DCS are assessed on the localization performance when applied in the back end of the graph SLAM system. The evaluated parameters are the position and orientation of the test vehicle.

1.4 Thesis Scope

The goal is to investigate robust optimization techniques using graph SLAM in a realistic setting. A complete graph SLAM system is used in order to examine the performance and robust methods that are applied in the optimization part of graph SLAM.

The two methods chosen for comparison in this thesis are RRR and DCS. Although there are many more methods and approaches to this problem they are disregarded due to time limitations.

Mapping will be disregarded as an evaluation parameter. A map exists in the shape of point cloud data but due to the lack of ground truth data, this is purposely excluded in this investigation.

The front end which is responsible for data processing and scan matching is kept constant while investigating the different robust optimization methods. A better performing front end would improve the overall localization performance but how the front end performance impacts localization is left outside the scope of this report.

1.5 Contributions

The contribution of this thesis is how lidar odometry can be used in conjunction with GNSS in the case of unreliable GNSS measurements. It adds to the literature on robust optimization methodology by providing an evaluation using a complete system using real world data collected on a public road.

The thesis contributes knowledge about how robust methods perform in a larger system utilizing several different sensors, such as the four lidar setup and a dual GNSS receiver. A contribution is also made by comparing robust methods when using GNSS measurements added as position observations, which is less common then the pseduorange observation approach for the graph SLAM context.

1.6 Division of Labor

Throughout the thesis work, both authors have contributed important work to answer the research questions. The work has been divided between the two authors in the following way.

Åström mainly focused on the implementation of the *Realising, Reversing, Recovering* algorithm and the lidar odometry estimation together with point cloud deskewing. In addition, he also did the modeling of the tunnel scenario.

Sundström mainly focused on *Dynamic Covariance Scaling* and the implementation to incorporate GNSS measurements into the SLAM framework. Further the evaluation of the methods performance in the scenario with multipath GNSS measurements.

There was a large amount of work needed to get a properly functioning graph SLAM framework running together with the data set used, this was done through collaboration from both authors.

1.7 Thesis Outline

In Chapter 2 the theory and concepts relevant for understanding the methods employed are presented. First, the general theory for simultaneous localization and mapping is explained and is followed by how the problem is formulated using factor graphs. The theory of robust optimization techniques is explained and is then followed by some theory about sensors.

Chapter 3 describes the experimental setup used for collecting measurements and then how the measurements are used to generate constraints that are used in the optimization problem formulation. This section also describes modeling of GNSS disturbances which are used for evaluation.

The experiments are presented in Chapter 4, first the evaluation metrics are introduced. The experiments are conducted with three scenarios, one scenario with disturbances found in the GNSS measurements and two scenarios with modeled GNSS disturbances. A parameter sweep for the two evaluated methods is conducted followed by experiments in each scenario where the state estimation is compared to a ground truth INS system.

Chapter 5 discusses the results from the experiments in Chapter 4. The discussion covers parameter sensitivity, localization performance and the method used. Conclusions from the thesis are presented Chapter 6 along with suggestions for further work.

2
Theory

This chapter presents the SLAM problem and the theory used for evaluating robust methods within graph SLAM. The first section introduces and formulates the SLAM problem. It is followed by a description of factor graphs and how graph SLAM utilizes factor graphs to solve the SLAM problem. The theory used for optimizing the graph in order to retrieve the best state estimation in 3D is also presented. The robust techniques applied in this paper are described in Section 2.4.

2.1 Simultaneous Localization and Mapping

In order to navigate in an unknown environment, an autonomous vehicle needs to determine its own position and generate a map of the local environment. The vehicle obtains information about its movement and the surrounding world through sensor measurements. The presented problem is called *simultaneous localization and mapping* (SLAM) and this section presents the basic probabilistic definition of the SLAM problem.

The basic SLAM components are pose $\mathbf{x}_i$, odometry $\mathbf{u}_i$, measurement $\mathbf{z}_i$, and map $\mathbf{m}$. Location $\mathbf{x}_i$ consists of both position and orientation which for the 2D case is position (x, y) and yaw θ.

Map $\mathbf{m}$ represents the local environment and a common way to express the map is through landmarks m_k. Landmarks are distinguishable features of particular interest in the surrounding environment. An example could be trees in a park or lighthouses at sea. Depending on the sensors used, what is considered landmarks can vary. If a high-resolution camera or lidar is used, a landmark can be a corner of a building or lamp post. A SLAM system can also disregard landmarks

completely, and create a map from raw measurements.

The odometry $\mathbf{u}_i$ is the change in position and orientation between each time step i. Odometry $\mathbf{u}_i$ could be derived in several ways, such as from the control signal, and wheel encoders or any other sensor able to estimate positional change. If a wheel encoder is used, $\mathbf{u}_i$ is calculated based on the steering angle and the number of wheel turns. The last building block, measurement $\mathbf{z}_i$, carries relative information about the state vector $\mathbf{x}_i$, map $\mathbf{m}$ or both [1].

In a noise-free system, odometry would be enough to decide the complete trajectory, and measurements $\mathbf{z}_i$ would provide information resulting in a correct map. Since odometry $\mathbf{u}_i$ and measurements $\mathbf{z}_i$ are derived from sensors, noise, and bias have to be considered. Due to the noise uncertainty, the SLAM problem is conveniently stated in a probabilistic manner.

With a path consisting of all locations $\mathbf{x}_{0:i} = \mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_i$, map $\mathbf{m} = m_1, m_2, ..., m_k$, corresponding odometry $\mathbf{u}_{1:i} = \mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_i$, and measurements $\mathbf{z}_{1:i} = \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_i$ the full SLAM problem can be defined as finding the posterior probability:

$$p(\mathbf{x}_{0:i}, \mathbf{m} \mid \mathbf{z}_{1:i}, \mathbf{u}_{1:i}) \tag{2.1}$$

Both the sought-after trajectory and map consisting of landmarks are characterized by the probability distribution given the measurements and odometry. SLAM can also be expressed as the online SLAM problem, where only the current location $\mathbf{x}_t$ and map are estimated, given all previous measurements $\mathbf{z}_{1:i}$ and odometry $\mathbf{u}_{1:i}$:

$$p(\mathbf{x}_i, \mathbf{m} \mid \mathbf{z}_{1:i}, \mathbf{u}_{1:i}) \tag{2.2}$$

To solve the SLAM problem a transition model and an observation model are needed. A measurement can be modeled as

$$\mathbf{z}_i = h(\mathbf{x}_i, m_k) + \mathbf{e}_i \tag{2.3}$$

where $h(\cdot)$ is the measurement function. $h(\cdot)$ is usually a nonlinear function, describing the relation between the measurement $\mathbf{z}$, pose $\mathbf{x}$, and the corresponding landmark m_k. Noise is denoted $\mathbf{e}$ and is assumed zero-mean Gaussian with measurement covariance $\mathbf{R}$. The shape of the measurement function is highly dependent on what sensor is used, with some sensors such as GNSS only providing information about the position of the vehicle, and not map or orientation. The probability distribution of the observation model can be expressed as

$$p(\mathbf{z}_i \mid \mathbf{x}_i, m_k) \sim \mathcal{N}(\mathbf{z}_i;\ h(\mathbf{x}_i, \mathbf{m}_k),\ \mathbf{R}) \tag{2.4}$$

In the same way, a transition model from odometry measurements can be modeled as

$$\mathbf{x}_i = g(\mathbf{x}_{i-1}, \mathbf{u}_i) + \mathbf{v}_i \tag{2.5}$$

with Gaussian noise denoted $\mathbf{v}_i$ and covariance $\mathbf{Q}$.

The transition model describes the probability distribution of the current location $\mathbf{x}_i$ given the previous location $\mathbf{x}_{i-1}$ and odometry measurement $\mathbf{u}_i$. Formulated the transition model as a probability distribution results in

$$p(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \mathbf{u}_i) \sim \mathcal{N}(\mathbf{x}_i;\ g(\mathbf{x}_{i-1}, \mathbf{u}_i),\ \mathbf{Q}) \tag{2.6}$$

An important assumption and simplification made in order to make the solving of SLAM feasible is the Markov assumption which asserts that the current state is only dependent on the previous state. This means that given the previous state, the current state is independent of all other past states [9]. For example, $P(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \mathbf{x}_{i-3})$ is reduced to $P(\mathbf{x}_i|\mathbf{x}_{i-1})$ in accordance with the Markov Assumption. Another assumption is made of the map which is considered to be static, with all features of the map having a fixed position.

In order to solve the full SLAM problem, the full SLAM posterior $p(\mathbf{x}_{0:i}, \mathbf{m}|\mathbf{z}_{1:i}, \mathbf{u}_{1:i})$ is rewritten using Bayes rule

$$\begin{aligned} &p(\mathbf{x}_{0:i}, \mathbf{m} \mid \mathbf{z}_{1:i}, \mathbf{u}_{1:i}) \\ &\quad = \ \eta\ p(\mathbf{z}_i \mid \mathbf{x}_{0:i}, \mathbf{m}, \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i})\ p(\mathbf{x}_{0:i}, \mathbf{m} \mid \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i}) \end{aligned} \tag{2.7}$$

with η being a normalization constant. The last factor $p(\mathbf{x}_{0:i}, \mathbf{m} \mid \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i})$ can be factored further. Doing this over $\mathbf{x}_i$ gives

$$\begin{aligned} &p(\mathbf{x}_{0:i}, \mathbf{m} \mid \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i}) \\ &\quad = \ p(\mathbf{x}_i \mid \mathbf{x}_{0:i-1}, \mathbf{m}, \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i-1})\ p(\mathbf{x}_{0:i-1}, \mathbf{m} \mid \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i-1}) \end{aligned}$$

Given the previous state $\mathbf{x}_{i-1}$, the current state $\mathbf{x}_i$ is conditionally independent from $\mathbf{m}$, $\mathbf{x}_{1:i-2}$, and $\mathbf{u}_{1:i-1}$ under Markov assumption and only depends on $\mathbf{x}_{i-1}$ and odometry $\mathbf{u}_i$. $p(\mathbf{x}_{0:i}, \mathbf{m} \mid \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i})$ is now written as:

$$\begin{aligned} &p(\mathbf{x}_{0:i}, \mathbf{m} \mid \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i}) \\ &\quad = \ p(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \mathbf{u}_i)\ p(\mathbf{x}_{0:i-1}, \mathbf{m} \mid \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i-1}) \end{aligned} \tag{2.8}$$

Similarly to (2.8), the factor $p(\mathbf{z}_i \mid \mathbf{x}_{0:i}, \mathbf{m}, \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i})$ in (2.7) can be simplified. In $p(\mathbf{z}_i \mid \mathbf{x}_{0:i}, \mathbf{m}, \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i})$, the variable $\mathbf{z}_i$ is conditionally independent of $\mathbf{x}_{0:i-1}$, $\mathbf{z}_{1:i-1}$, and $\mathbf{u}_{1:i}$ given $\mathbf{x}_i$ and $\mathbf{m}$. Removing these variables results in the full SLAM problem being expressed as:

$$\begin{aligned} &p(\mathbf{x}_{0:i}, \mathbf{m} \mid \mathbf{z}_{1:i}, \mathbf{u}_{1:i}) \\ &\quad = \ \eta\ p(\mathbf{z}_i \mid \mathbf{x}_i, \mathbf{m}) \\ &\qquad p(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \mathbf{u}_i)\ p(\mathbf{x}_{0:i-1}, \mathbf{m} \mid \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i-1}) \end{aligned} \tag{2.9}$$

Here the densities describing the transition, and observation model can be identified. The factor $p(\mathbf{x}_{0:i-1}, \mathbf{m} \mid \mathbf{z}_{1:i-1}, \mathbf{u}_{1:i-1})$ represents all previous time steps and measurements [10].

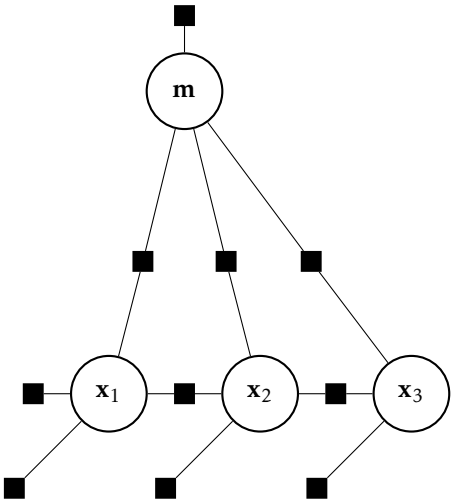


Figure 2.1: *Factor graph model of the SLAM problem. Variable nodes are* $\mathbf{x}$ *and* $\mathbf{m}$ *and the squares represent factors* $\phi_i(\mathbf{x}_i)$.

2.2 Factor Graphs

A factor graph is a graphical model describing the factored function $\phi(\mathbf{x})$. The graph consists of two types of nodes representing variables, and functions respectively. The lines connecting variable nodes with function nodes are called edges. A simple factor graph can be seen in Figure 2.1. The functions are the factors, and each function depends on its connected variables. The overall function $\phi(\mathbf{x})$ can be described as the product of the factors which depend upon all variables in the system, as in (2.10). Through the factored function $\phi(\mathbf{x})$ all sought-after states can be estimated given their relation described by the functions.

The main motivation for using factor graphs lies in the usefulness of the graphical representation but also computational benefits. It is easy to quickly evaluate the product of all factors in the graph that are merely related to the unknown variables $\mathbf{x}_i$, opening up for evaluating and optimizing variables in the graph. Factor graphs also offer increased flexibility when modeling, since it is not limited to only modeling proper density functions [11]. A factor graph shares a lot of similarities with a Bayesian network. The probabilistic definition of the SLAM problem makes a Bayesian network a suitable graphical model for the problem as in Figure 2.2. A Bayesian network is a directed graph that formulates a joint density from the product of several conditional probability densities, much like a factor graph. However, when inferring knowledge about the unknown variables in SLAM while using a factor graph, the measurements $\mathbf{z}$ are not explicitly represented in the graph. The unknown variables are the vehicle pose $\mathbf{x}$ and the map $\mathbf{m}$. The prior probability of $\mathbf{z}$, which is $P(\mathbf{z})$, is disregarded and instead each $\mathbf{z}_i$ is added to the graph as fixed value. This essentially means that priors $P(\mathbf{z})$, which

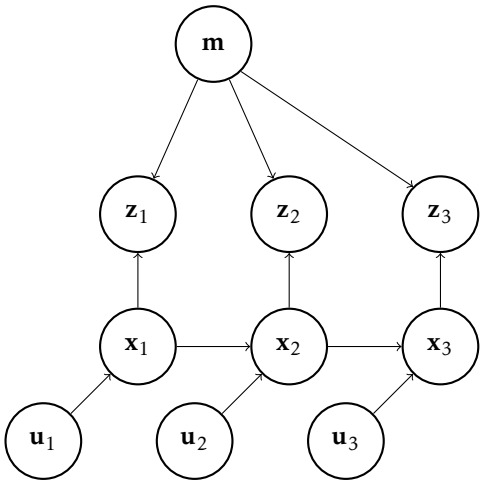


Figure 2.2: *Bayesian model of the SLAM problem*

are part of the normalization factor η in (2.9) are disregarded [12].

A Bayesian network can be converted to a factor graph by splitting each node into a factor node and a variable node. The function node is connected to the variable node and the parent node. Parent nodes are indicated by edges directed toward the current variable node. If a node contains known variables it is removed and instead becomes a parameter in the corresponding factor. Figure 2.2 is converted to a factor graph which can be seen in Figure 2.1. Here odometry measurements $\mathbf{u}$ and landmark measurements $\mathbf{z}$ are converted to factors represented by squares. Variable nodes $\mathbf{x}_1$ and $\mathbf{m}$ have no parent nodes but receive a factor describing the likelihood or density function associated with the respective variable node [13]. The overall function generated from a factor graph can be written as:

$$\phi(\mathbf{x}) = \prod_i \phi_i(\mathbf{x}_i) \tag{2.10}$$

where ϕ_i represents the function nodes, and $\mathbf{x}_i$ denotes the states contained in the variable nodes. The variable nodes are connected to the corresponding function node ϕ_i. $\phi(\mathbf{x})$ becomes the overall function [12].

In the case of SLAM, the factors $\phi_i(\mathbf{x}_i)$ can either represent the measurement distribution $p(\mathbf{z}_i \mid \mathbf{x}_i, \mathbf{m})$ or transition distribution $p(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \mathbf{u}_i)$. The exact shape of these factors is explained further in Section 2.3.2. The function $\phi(\mathbf{x})$ now represents the full SLAM problem in (2.9) as

$$\phi(\mathbf{x}) \propto p(\mathbf{x}_{0:i}, \mathbf{m} \mid \mathbf{z}_{1:i}, \mathbf{u}_{1:i}) \tag{2.11}$$

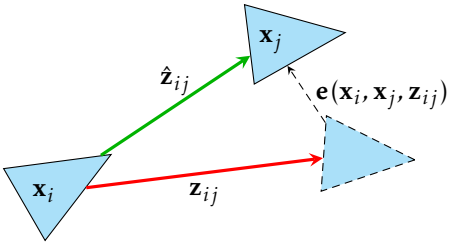


Figure 2.3: *The 2D pose error function $\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$ can be explained in a simplified way as the difference between the observed relative pose $\mathbf{z}_{ij}$ and the estimated relative pose $\hat{\mathbf{z}}_{ij}$.*

2.3 Graph-based SLAM

Graph SLAM utilizes a factor graph to store information from measurements and odometry readings which are used to make the best estimate of the current state vector. The factors $\phi_i(\mathbf{x}_i)$ in (2.10) are represented by error functions $\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$, and the best estimate is found by minimizing the squared sum of all error functions. The error functions measure how well the state variables of two nodes $\mathbf{x}_i$, $\mathbf{x}_j$ satisfy the corresponding measurement $\mathbf{z}_{ij}$. The measurement $\mathbf{z}_{ij}$ describes the relative transformation from $\mathbf{x}_i$ to $\mathbf{x}_j$ and can be derived from various sources such as odometry measurements or by aligning observations acquired from the two locations. Figure 2.3 shows a simple representation of an error function in two dimensions with an odometry measurement.

In the three-dimensional case, the state variables stored in each node consist of a translational part and a rotational part. There are different approaches to representing the rotational component. A common way is by using Roll, Pitch, and Yaw which is a minimal representation of the rotational states. But due to a phenomenon called gimbal lock, where one *degree of freedom* (**dof**) is lost, quaternion representation is used instead. The quaternion components are q_x, q_y, q_z, q_w and the state vector $\mathbf{x}_i$ describing 3D-pose becomes

$$\mathbf{x}_i = \begin{pmatrix} t_x & t_y & t_z & q_x & q_y & q_z & q_w \end{pmatrix}^T \tag{2.12}$$

where t_x, t_y, t_z represents translation [14].

Every consecutive node $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ is connected by an edge with the relative motion measurement $\mathbf{z}_{ij}$ which is derived from an odometry estimate. Every measurement $\mathbf{z}_{ij}$ has a corresponding information matrix $\mathbf{\Omega}_{ij}$ which describes the measurement uncertainty [1]. In addition to the measurements between nodes, other measurements can be included. Figure 2.4 represents a graph where GNSS positions are included as measurements.

The graph SLAM problem can be split into two separate problems: the *front end*, and *back end*. The front end consists of collecting and processing sensor data

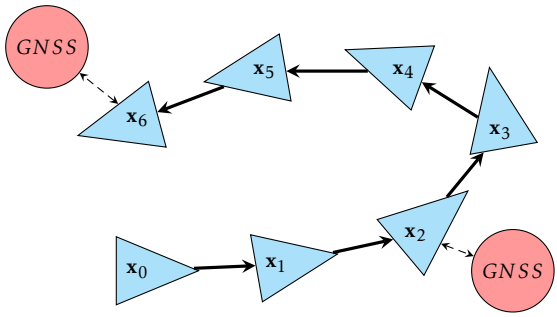


Figure 2.4: *Example of a pose graph with bold arrows representing odometry constraints between poses, dotted arrows representing GNSS edges.*

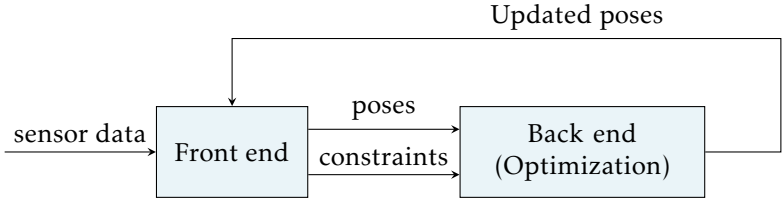


Figure 2.5: *The two parts of graph SLAM visualized. The front end handles sensor data and constructs nodes and edges that are fed to the back end. In the back end, the factor graph is optimized to minimize the total error.*

which is then used to construct nodes and edges in the factor graph. The back end optimizes the graph by finding the best state estimate for each variable node given the edge constraints. This is achieved by minimizing the log-likelihood of the factor graph function $\phi(\mathbf{x})$ which is further explained in Section 2.3.2. Each time the log-likelihood function is minimized a new node configuration is updated to the front end. A schematic block diagram can be seen in Figure 2.5, which shows how the front end interacts with the back end.

2.3.1 Optimization on a Manifold

The optimization methods employed when solving the full SLAM problem using a factor graph, such as Gauss-Newton, and Levenberg-Marquart is designed with the assumption that the space of parameters is Euclidean. If this assumption is not fulfilled, errors are introduced into the solution. The parameters used to describe the state vector $\mathbf{x}_i$ of the 3D SLAM problem in this thesis contains a translation part $\mathbf{t}_i$ to describe the position and a quaternion $\mathbf{q}_i$ which describes the orientation. The translation part forms a Euclidean space while the quaternion component is a non-Euclidean representation of rotation. In addition, the quaternion representation introduces one extra DOF by using four components to describe a three-dimensional orientation. Quaternions are introduced to avoid gimbal lock, a problem occurring when using three states (RPY) describing rota-

tion where a DOF is lost. Using quaternion avoids the problem of gimbal lock, but the extra DOF can still lead to ambiguities when trying to solve the optimization problem. To solve these problems a common method is to perform the optimization on a manifold. A manifold is a space that locally can be seen as a Euclidean space but globally behaves differently, see Figure 2.6 for a visual representation.

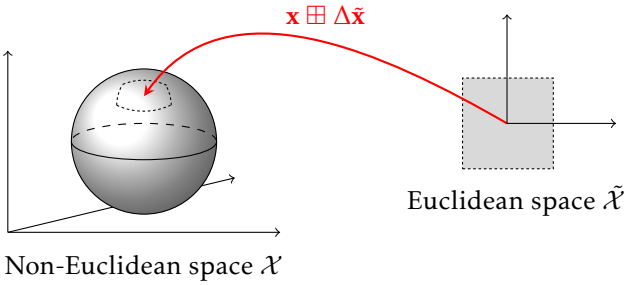


Figure 2.6: *Example of a locally Euclidean space being projected onto a non-Euclidean space using the $\boxplus$ operator defined in (2.16).*

The working principle of methods such as Gauss-Newton, and Levenberg-Marquart is to minimize some squared error function $S(\mathbf{x})$ by iteratively updating a state vector $\mathbf{x}$ with small increments [15]

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x} \tag{2.13}$$

and the $\Delta\mathbf{x}$ is calculated by solving

$$\frac{\partial S(\mathbf{x} + \Delta\mathbf{x})}{\partial \Delta\mathbf{x}}\Big|_{\Delta\mathbf{x}=0} = 0. \tag{2.14}$$

To solve the problem of over-parametrization when optimizing, Grisetti et al. [14] uses a manifold with a 6D parametrization of the state vector in (2.12) $\mathbf{x}$ which only uses the vector part of the unit quaternion. This means that the manifold representation of the state vector becomes

$$\Delta\tilde{\mathbf{x}} = \begin{pmatrix} \Delta\tilde{\mathbf{t}} \\ \Delta\tilde{\mathbf{q}} \end{pmatrix}, \tag{2.15}$$

where $\Delta\tilde{\mathbf{t}} = (\Delta t_x, \Delta t_y, \Delta t_z)^T$ is the translation and $\Delta\tilde{\mathbf{q}} = (\Delta q_x, \Delta q_y, \Delta q_z)^T$.

To move between the Euclidean space and the original space, the idea is to introduce the *box-plus* operator $\boxplus$ which can be seen as the equivalent of the normal addition operator $+$ in Euclidean spaces. This means that incremental update $\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$ becomes $\mathbf{x} \leftarrow \mathbf{x} \boxplus \Delta\tilde{\mathbf{x}}$ when using the manifold. The operator $\boxplus$ maps $\Delta\tilde{\mathbf{x}}$ into the original space and is defined as:

$$\mathbf{x} \boxplus \Delta\tilde{\mathbf{x}} = \mathbf{x} \oplus \begin{pmatrix} \Delta\tilde{\mathbf{t}} \\ \Delta\tilde{\mathbf{q}} \\ \sqrt{1 - \| \Delta\tilde{\mathbf{q}} \|^2} \end{pmatrix}, \tag{2.16}$$

the $\oplus$ operator is defined in Section 2.5.1.

2.3.2 Nonlinear Graph Optimization Using Least-squares

As explained in Section 2.1 and Section 2.2 the full SLAM problem (2.1) can be solved through estimating conditional densities $p(\mathbf{z}_i \mid \mathbf{x}_i, m_k)$ and $p(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \mathbf{u}_i)$ stemming from the nonlinear motion model $g(\cdot)$ and measurement model $h(\cdot)$. This section covers the structure of the optimization problem created from the factor graph. This is done by creating and linearizing error functions based on measurements with normally distributed noise using Taylor expansion.

The negative log posterior or information form is preferred due to computational efficiency [12]. Under Gaussian noise assumption, the full SLAM problem can be written as

$$-\log \phi(\mathbf{x}) = -\log p(\mathbf{x}_{0:i}, \mathbf{m} \mid \mathbf{z}_{1:i}, \mathbf{u}_{1:i}) = const \tag{2.17a}$$

$$-\sum_i [\mathbf{x}_i - g(\mathbf{x}_{i-1}, \mathbf{u}_i)]^T \, \mathbf{Q}_i^{-1} \, [\mathbf{x}_i - g(\mathbf{x}_{i-1}, \mathbf{u}_i)] \tag{2.17b}$$

$$-\sum_i [\mathbf{z}_i - h(\mathbf{x}_i, \mathbf{m})]^T \, \mathbf{R}_i^{-1} \, [\mathbf{z}_i - h(\mathbf{x}_i, \mathbf{m})] \tag{2.17c}$$

where (2.17b) is an odometry reading and (2.17c) is a feature observation. The error function $\mathbf{e}$ becomes

$$\mathbf{e}(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{u}_{i+1}) = [\mathbf{x}_{i+1} - g(\mathbf{x}_i, \mathbf{u}_{i+1})], \tag{2.18}$$

and for simplicity of notation, the measurement and node indices can be encoded in the indices of the error function

$$\mathbf{e}_{i,i+1} = \mathbf{e}(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{u}_{i+1}).$$

The graph SLAM optimization problem is a nonlinear optimization problem that can be solved using least squares and by using standard optimization methods, Grisetti et al. present a way to structure the problem that is based on least-squares error minimization [14].

For the vector of nodes, $\mathbf{x} = (\mathbf{x}_1^T, ..., \mathbf{x}_n^T)^T$ where each $\mathbf{x}_i$ represents the vehicle pose of node i, and with a set $\mathcal{C}$ that contains pairs of indices which has an observation $\mathbf{z}$. The goal is to find the optimal configuration of nodes $\mathbf{x}^*$ that minimizes a negative log-likelihood $\mathbf{F}(\mathrm{x})$ with respect to all observations. Rewriting Equation (2.17c) with indices i and j results in

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j \in \mathcal{C} \rangle} \underbrace{\mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}}_{=\mathbf{F}_{ij}} \tag{2.19}$$

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \, \mathbf{F}(\mathbf{x}). \tag{2.20}$$

The information matrix $\mathbf{\Omega}_{ij}$ is the concatenated inverses of covariance matrices $\mathbf{R}_i$ and $\mathbf{Q}_i$.

The first step in solving (2.19) is using the $\boxplus$ operator defined in Section 2.3.1 with the initial guess $\check{\mathbf{x}}$ and local variation $\Delta\tilde{\mathbf{x}}$. The error function becomes

$$\mathbf{e}_{ij}(\check{\mathbf{x}}_i \boxplus \Delta\tilde{\mathbf{x}}_i,\, \check{\mathbf{x}}_j \boxplus \Delta\tilde{\mathbf{x}}_j) = \mathbf{e}_{ij}(\check{\mathbf{x}} \boxplus \Delta\tilde{\mathbf{x}}). \tag{2.21}$$

and can be approximated by its first order Taylor expansion

$$\mathbf{e}_{ij} + \mathbf{J}_{ij}\Delta\tilde{\mathbf{x}} \tag{2.22}$$

where the Jacobian $\mathbf{J}_{ij}$ is

$$\frac{\partial \mathbf{e}_{ij}(\check{\mathbf{x}} \boxplus \Delta\tilde{\mathbf{x}})}{\partial \Delta\tilde{\mathbf{x}}}\bigg|_{\Delta\tilde{\mathbf{x}}=0}. \tag{2.23}$$

Since (2.21) is only dependent on $\Delta\tilde{\mathbf{x}}_i$ and $\Delta\tilde{\mathbf{x}}_j$ the Jacobian gets a sparse structure with non-zero elements at positions i and j. Using the rule for partial derivatives and evaluating the Jacobian in $\Delta\tilde{\mathbf{x}} = 0$ we get the Jacobian matrix

$$\mathbf{J}_{ij} = \begin{pmatrix} 0 \cdots 0 & \underbrace{\frac{\partial \mathbf{e}_{ij}(\check{\mathbf{x}})}{\partial \check{\mathbf{x}}_i}}_{\mathbf{A}_{ij}} \cdot \underbrace{\frac{\check{\mathbf{x}}_i \boxplus \Delta\tilde{\mathbf{x}}_i}{\partial \Delta\tilde{\mathbf{x}}_i}\Big|_{\Delta\tilde{\mathbf{x}}=0}}_{\mathbf{M}_i} & 0 \cdots 0 & \underbrace{\frac{\partial \mathbf{e}_{ij}(\check{\mathbf{x}})}{\partial \check{\mathbf{x}}_j}}_{\mathbf{B}_{ij}} \cdot \underbrace{\frac{\check{\mathbf{x}} \boxplus \Delta\tilde{\mathbf{x}}_j}{\partial \Delta\tilde{\mathbf{x}}_j}\Big|_{\Delta\tilde{\mathbf{x}}=0}}_{\mathbf{M}_j} & 0 \cdots 0 \end{pmatrix} \tag{2.24}$$

Here, $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$ are the derivative of the error function $\mathbf{e}_{ij}(\check{\mathbf{x}})$ with respect to $\check{\mathbf{x}}$. $\mathbf{M}_i$ and $\mathbf{M}_j$ are the derivatives of the local variation $\Delta\tilde{\mathbf{x}}$ projected on the initial guess $\check{\mathbf{x}}$ using the $\boxplus$ operator with respect to $\Delta\tilde{\mathbf{x}}$.

By substituting the error terms $\mathbf{e}_{ij}$ in (2.19) with the Taylor expansion (2.22)

$$\begin{aligned} \mathbf{F}_{ij}(\check{\mathbf{x}} \boxplus \Delta\tilde{\mathbf{x}}) &= (\mathbf{e}_{ij} + \mathbf{J}_{ij}\Delta\tilde{\mathbf{x}})^T \mathbf{\Omega}_{ij} (\mathbf{e}_{ij} + \mathbf{J}_{ij}\Delta\tilde{\mathbf{x}}) \\ &= \underbrace{\mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}}_{c_{ij}} + 2\underbrace{\mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{J}_{ij}}_{\mathbf{b}_{ij}} \Delta\tilde{\mathbf{x}} + \Delta\tilde{\mathbf{x}}^T \underbrace{\mathbf{J}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{J}_{ij}}_{\mathbf{H}_{ij}} \Delta\tilde{\mathbf{x}} \\ &= c_{ij} + 2\mathbf{b}_{ij}\Delta\tilde{\mathbf{x}} + \Delta\tilde{\mathbf{x}}^T \mathbf{H}_{ij} \Delta\tilde{\mathbf{x}} \end{aligned} \tag{2.25}$$

is obtained. The sparse structure of the Jacobian $\mathbf{J}_{ij}$ transfers into the information matrix $\mathbf{H}_{ij}$ and $\mathbf{b}_{ij}$ which also becomes sparse with non-zero elements only at the

i and j positions

$$\mathbf{H}_{ij} = \begin{pmatrix} 0 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 0 & & & & & & & & \\ & & & \mathbf{A}_{ij}^T\mathbf{\Omega}_{ij}\mathbf{A}_{ij} & 0 & \cdots & 0 & \mathbf{A}_{ij}^T\mathbf{\Omega}_{ij}\mathbf{B}_{ij} & & & \\ & & & 0 & 0 & & & 0 & & & \\ & & & \vdots & & \ddots & & \vdots & & & \\ & & & 0 & & & 0 & 0 & & & \\ & & & \mathbf{B}_{ij}^T\mathbf{\Omega}_{ij}\mathbf{A}_{ij} & 0 & \cdots & 0 & \mathbf{B}_{ij}^T\mathbf{\Omega}_{ij}\mathbf{B}_{ij} & & & \\ & & & & & & & & 0 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 0 \end{pmatrix}, \mathbf{b}_{ij} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mathbf{A}_{ij}^T\mathbf{\Omega}_{ij}e_{ij} \\ 0 \\ \vdots \\ 0 \\ \mathbf{B}_{ij}^T\mathbf{\Omega}_{ij}e_{ij} \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Due to the known structure of $\mathbf{H}_{ij}$, $\mathbf{b}_{ij}$ and factor graph, $\mathbf{F}(\mathbf{x})$ can be rewritten on quadratic form by using (2.25) and simply taking the sum of every $\mathbf{H}_{ij}$, $\mathbf{b}_{ij}$ and c_{ij} for each observation in $\mathcal{C}$

$$\begin{aligned} \mathbf{F}(\check{\mathbf{x}} \boxplus \Delta\tilde{\mathbf{x}}) &= \sum_{\langle i,j \in \mathcal{C} \rangle} \mathbf{F}_{ij}(\check{\mathbf{x}} \boxplus \Delta\tilde{\mathbf{x}}) \\ &\simeq \sum_{\langle i,j \in \mathcal{C} \rangle} (c_{ij} + 2\mathbf{b}_{ij}\Delta\tilde{\mathbf{x}} + \Delta\tilde{\mathbf{x}}^T\mathbf{H}_{ij}\Delta\tilde{\mathbf{x}}) \\ &= \underbrace{\sum_{\langle i,j \in \mathcal{C} \rangle} c_{ij}}_{=c} + 2 \underbrace{\sum_{\langle i,j \in \mathcal{C} \rangle} \mathbf{b}_{ij}}_{=\mathbf{b}} \Delta\tilde{\mathbf{x}} + \Delta\tilde{\mathbf{x}}^T \underbrace{\sum_{\langle i,j \in \mathcal{C} \rangle} \mathbf{H}_{ij}}_{=\mathbf{H}} \Delta\tilde{\mathbf{x}}. \\ &= c + 2\mathbf{b}^T\Delta\tilde{\mathbf{x}} + \Delta\tilde{\mathbf{x}}^T\mathbf{H}\Delta\tilde{\mathbf{x}}. \end{aligned} \tag{2.26}$$

The optimal configuration of nodes $\mathbf{x}^*$ can now be computed by solving linear system

$$\mathbf{H}\Delta\tilde{\mathbf{x}}^* = -\mathbf{b}. \tag{2.27}$$

This can be done by using sparse Cholesky factorization, the solution $\Delta\tilde{\mathbf{x}}^*$ is at this stage in the local Euclidean space surrounding the initial guess $\check{\mathbf{x}}$ and needs to be remapped to the original space with the $\boxplus$ operator [14]

$$\check{\mathbf{x}}^* \leftarrow \check{\mathbf{x}} \boxplus \Delta\tilde{\mathbf{x}}^*. \tag{2.28}$$

2.4 Robust optimization techniques

When the SLAM problem is formulated on a quadratic form that is solved by minimizing the sum of squared errors two important assumptions are made. The

assumptions of the least squares method are independent Gaussian noise and that the data is free from systematic errors [16]. In graph SLAM, outlier data is usually detected and handled by the front end. In real world operation and in uncontrollable environments the risk of ambiguities and outliers is higher compared to controlled conditions. The problem with outliers is that they influence the solution more than inliers because the error is squared, therefore the back end optimizer is heavily dependent on the front end generating a topologically correct factor graph representation [2]. To reduce the effect of outliers affecting the final solution, robust optimization techniques are used in the back-end.

In least squares minimization problems, M-estimators are a standard method for increasing robustness against outliers. M-estimators achieve robustness by using a loss function that weighs down large errors. There are different loss functions that have their own behavior, such as Huber, Tukey, and Cauchy[17].

Switchable constraints (SC) were first introduced within the graph SLAM back end as a robust technique for handling loop closure outliers. A loop closure is a constraint between two nodes i and j based on place recognition. The constraint is added to the graph when the robot revisits a previously visited area and acknowledges this through a measurement. Sünderhauf and Protzel [2] introduces an additional free variable to the negative log-likelihood function which controls a switch function and disables individual constraints. The negative log-likelihood function consists of odometry constraints, switchable loop closure constraints, and switch prior constraints.

2.4.1 Dynamic Covariance Scaling

Dynamic covariance scaling builds upon switch constraints by providing a solution to the switch factor s_{ij} analytically. This removes the need for an extra parameter in the optimization problem while still increasing the covariance of unfit loop closure constraints.

The odometry error function is defined as:

$$\mathbf{e}^{\text{odom}}(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{z}_{i,i+1}) = f(\mathbf{x}_i, \mathbf{x}_{i+1}) - \mathbf{z}_{i,i+1} \tag{2.29}$$

Here, the error function describes a constraint between two consecutive nodes i and $i+1$. The value of the function is the difference between the predicted motion $g(\mathbf{x}_i, \mathbf{x}_{i+1})$ and odometry reading $\mathbf{z}_{i,i+1}$.

The loop closure error function $\mathbf{e}^{\text{LC}}$ is defined similarly to the odometry error function. The value of the function is the difference between the predicted distance between node i and j, $g(\mathbf{x}_i, \mathbf{x}_j)$, and the measured distance $\mathbf{z}_{ij}$. The loop closure error function $\mathbf{e}^{\text{LC}}$ is formulated as

$$\mathbf{e}^{\text{LC}}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) = g(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_{ij} \tag{2.30}$$

For simplicity of notation, the measurement and node indices are encoded in the indices of the error functions. The function to minimize in order to estimate the

most likely states becomes:

$$\mathbf{F}(\mathbf{x}) = \arg\min_{\mathbf{x}} \sum_{i} (\mathbf{e}_{i,i+1}^{\text{odom}})^T \mathbf{\Omega}_i \mathbf{e}_{i,i+1}^{\text{odom}} + \sum_{ij} s_{ij}^2 \underbrace{(\mathbf{e}_{ij}^{\text{LC}})^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}^{\text{LC}}}_{=\chi_{l_{ij}}^2} \tag{2.31}$$

where $\mathbf{e}_{i,i+1}^{odom}$ and $\mathbf{e}_{ij}^{LC}$ are error functions for odometry and loop closure constraints defined in (2.29) and (2.30). And the two information matrices $\mathbf{\Omega}_i$ and $\mathbf{\Omega}_{ij}$ containing the inverse covariance of the measurements respectively.

The scaling factor s_{ij} is calculated by

$$s_{ij} = \min(1, \frac{2\Phi}{\Phi + \chi_{l_{ij}}^2}). \tag{2.32}$$

and is dependent on $\chi_{l_{ij}}^2$, which is the error term for each loop closure described in (2.31). This term scales the information matrix individually for each constraint. Φ is a free parameter which is usually set to 1 [3, 18].

2.4.2 Realizing, Reversing, Recovering

Latif et al. [4] proposes an algorithm called *Realizing*, *Reversing*, *Recovering* (RRR) that performs outlier rejection to remove faulty loop closure constraints in graph SLAM. The algorithm is consistency-based and uses a series of χ^2 tests to identify constraints that agree with the vehicle odometry locally and globally. Before the algorithm can be utilized, the constraints are divided into two sets: the odometry set which contains sequential constraints on the vehicle movement, and the second set called *edges* which contains other constraints such as loop closures or as in our case position constraints from GNSS measurements.

The first step involves further dividing the second set into smaller subsets called *clusters* by grouping constraints together based on a simple time threshold t_g. Consequently, a new cluster is initialized every t_g seconds and constraints are added to it. Each cluster is then checked independently in Algorithm 1 named *intra-cluster consistency*, to find if they are internally consistent with the odometry constraints by doing a χ^2 test

$$D_G^2 = \sum_{i} \mathbf{e}_{i,i+1}^{\text{odom}\,T} \mathbf{\Omega}_{i,i+1} \mathbf{e}_{i,i+1}^{\text{odom}} + \sum_{ij} \mathbf{e}_{ij}^{\text{edge}\,T} \mathbf{\Omega}_{ij} \mathbf{e}_{ij}^{\text{edge}} < \chi_{\alpha,d_G}^2. \tag{2.33}$$

If the cluster is consistent, each edge within the cluster is considered good if it

satisfies the χ^2 test

$$D^2_{\text{edge}} = \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij} < \chi^2_{\alpha, d_{\text{edge}}}, \tag{2.34}$$

where d_G and d_{edge} are the degrees of freedom for the full graph and each edge respectively. Clusters and edges that do not satisfy the χ^2 tests are removed from the set.

Algorithm 1 Intra Cluster Consistency

Input: Graph, Odometry edges, cluster edges
Output: Cluster edges

```
active_edges ← {odometry edges, cluster edges}
Graph->optimize(active_edges)
if D_G^2 < χ^2_{α,d_G} then
    for each edge ∈ cluster do
        if D^2_edge < χ^2_{α,d_edge} then
            Accept edge
        else
            Reject edge
        end if
    end for
else
    Reject cluster
end if
```

Once Algorithm 1 is finished, all clusters are internally consistent and considered candidates. The outputs from Algorithm 1 is used as input to Algorithm 2 which aims to identify all candidate clusters that are jointly consistent. A cluster can either be considered good or rejected, so two empty sets called good clusters and rejected clusters are defined. The graph is then optimized assuming that all clusters are consistent by incorporating all candidates in the optimization problem. Once the graph is optimized the algorithm tests every edge to determine if it satisfies the χ^2 test in (2.34). If all edges pass the test, Algorithm 2 considers all clusters as good and stops executing. However, if an edge fails the test, the cluster to which the edge belongs is removed from the optimization and added to the rejected clusters set.

Algorithm 2 RRR

Input: odometry_edges, candidate_clusters, Graph
Output: good_clusters

```
1: good_clusters ← {}
2: rejected_clusters ← {}
3: loop
4:     active_edges ← {odometry_edges}
5:     for each edge ∈ candidate_clusters do
6:         active_edges ← {active_edges, edge}
7:     end for
8:     graph->optimize(active_edges)
9:     for each cluster ∈ candidate_clusters do
10:        for each edge ∈ cluster do
11:            if D²_edge > χ²_{α,d_edge} then
12:                add cluster to rejected_clusters
13:                remove cluster from candidate_clusters
14:                break
15:            end if
16:        end for
17:    end for
18:    if isempty(candidate_clusters) then
19:        break
20:    else
21:        s = candidate_clusters.size
22:        (good_clusters, r_clusters)
23:        ← Inter_Cluster_Consistency(good_clusters, candidate_clusters)
24:        if good_clusters.size > s then
25:            rejected_clusters ← {}
26:        else
27:            rejected_clusters ← {rejected_clusters, r_clusters}
28:        end if
29:    end if
30: end loop
```

Then Algorithm 3, *inter cluster consistency* runs until all candidate clusters have been classified as either good or rejected. The graph is optimized with both good and candidate clusters together, checking if they satisfy both the χ^2 test in (2.33) and

$$D_C^2 = \sum_{ij} \mathbf{e}_{ij}^{\text{edge}\,T} \mathbf{\Omega}_{ij} \mathbf{e}_{ij}^{\text{edge}} < \chi^2_{\alpha, d_C}. \tag{2.35}$$

If they pass these tests, the candidates are added to the good set. If not, a consis-

tency index

$$CI = \frac{D_C^2}{\chi^2_{\alpha,d_C}} \tag{2.36}$$

is calculated for each cluster to remove the cluster which introduces the largest error to the graph. Then *inter cluster consistency* is called recursively until the candidate set is empty.

Algorithm 3 Inter Cluster Consistency

Input: odometry_edges, good_clusters, candidate_clusters, Graph
Output: good_clusters, rejected_clusters

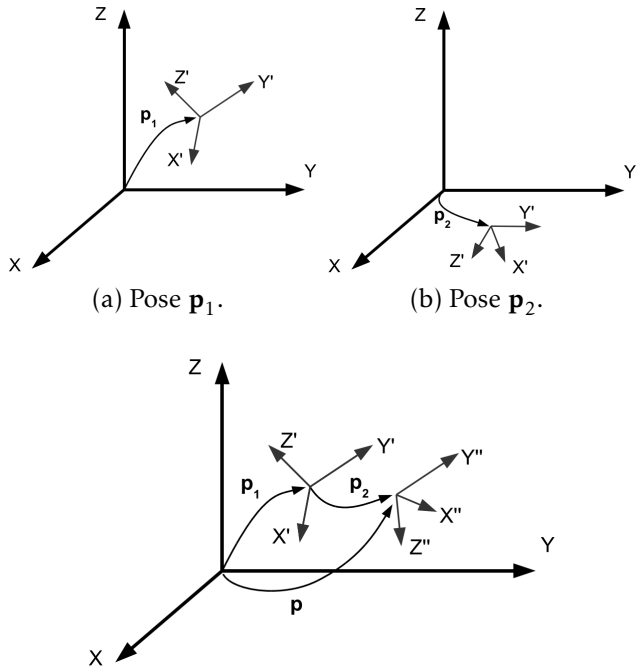
```
active_edges ← odometry edges
for each edge ∈ (good_clusters ∪ candidate_cluster) do
    active_edges ← {active_edges, edge}
end for
graph->optimize(active_edges)
if D_C^2 < χ^2_{α,d_C} ∧ D_G^2 < χ^2_{α,d_G} then
    good_clusters ← {good_clusters, candidate_cluster}
else
    Find cluster_i ∈ good_clusters with largest Consistency Index (D_C^2/χ^2_{α,d_C})
    Remove cluster_i from good_clusters
    rejected_clusters ← cluster_i
    if ¬isempty(candidate_clusters) then
        (good_clusters, r_clusters)
        ← inter_cluster_consistency(good_clusters, candidate_clusters)
        rejected_clusters ← {rejected_clusters, r_clusters}
    end if
end if
```

Once RRR has classified all clusters as either good or rejected, all edges in the good clusters set are added to the graph.

To summarize the algorithm, a series of χ^2 tests are used to identify constraints consistent with the odometry estimate both locally inside time-segmented clusters and globally.

2.5 Mathematical Operations on 3D Poses

This section defines two important mathematical operators that are used with the state representation used in (2.12). The state representation is referred to as a 3D pose and the mathematical operators are used to solve the error functions used in the 3D SLAM problem. The composition operator $\oplus$ and the inverse of a pose are defined [15].



(a) Pose $\mathbf{p}_1$. (b) Pose $\mathbf{p}_2$.

(c) The composition $\mathbf{p}_1 \oplus \mathbf{p}_2$ leads to the new pose $\mathbf{p}_3$.

Figure 2.7: *The composition of two poses creates* $\mathbf{p}_3$*. Source: [15].*

2.5.1 Composition between two poses

The composition between two poses can be thought of as calculating the sum of two poses $\mathbf{p}_i$ and $\mathbf{p}_j$ using the *o-plus* operator $\oplus$. The translation part represents the displacement of $\mathbf{p}_j$ added to $\mathbf{p}_i$ point of view with respect to its rotation and position. The rotational part is simply the quaternion product. The composition operator $\oplus$ is defined as:

$$\mathbf{p}_i \oplus \mathbf{p}_j = \begin{pmatrix} t_{i,x} + t_{j,x} + 2\left[-(q_{i,y}^2 + q_{i,z}^2)t_{j,x} + (q_{i,x}q_{i,y} - q_{i,w}q_{i,z})t_{j,y} + (q_{i,w}q_{i,y} + q_{i,x}q_{i,z})t_{j,z}\right] \\ t_{i,y} + t_{j,y} + 2\left[(q_{i,w}q_{i,z} + q_{i,x}q_{i,y})t_{j,x} - (q_{i,x}^2 + q_{i,z}^2)t_{j,y} + (q_{i,y}q_z - q_{i,w}q_{i,x})t_{j,z}\right] \\ t_{i,z} + t_{j,z} + 2\left[(q_{i,x}q_{i,z} - q_{i,w}q_{i,y})t_{j,x} + (q_{i,w}q_{i,x} + q_{i,y}q_{i,z})t_{j,y} - (q_{i,x}^2 + q_{i,y}^2)t_{j,z}\right] \\ \mathbf{q}_i \cdot \mathbf{q}_j \end{pmatrix}. \tag{2.37}$$

2.5.2 Inverse of a pose

Taking the inverse of pose $\mathbf{p}$, can be seen as negating a vector in Euclidean space.

The inverse of a pose can be divided into two parts, the translational part, and the rotational part. The translation part can be interpreted as the relative position of

the origin $[0,0,0]^T$ as seen from the pose $\mathbf{p}$. The rotational part is simply the conjugate quaternion. The inverse of a pose is defined as:

$$\mathbf{p}^{-1} = \begin{pmatrix} -t_x + 2\left[-(q_y^2+q_z^2)(-t_x) + (q_x q_y + q_r q_z)(-t_y) + (-q_r q_y + q_x q_z)(-t_z)\right] \\ -t_y + 2\left[(-q_r q_z + q_x q_y)(-t_x) - (q_x^2+q_z^2)(-t_y) + (q_y q_z + q_r q_x)(-t_z)\right] \\ -t_z + 2\left[(q_x q_z + q_r q_y)(-t_x) + (-q_r q_x + q_y q_z)(-t_y) - (q_x^2+q_y^2)(-t_z)\right] \\ -q_x \\ -q_y \\ -q_z \\ q_w \end{pmatrix}. \tag{2.38}$$

2.6 Lidar

A lidar sensor works by omitting several light beams. If the light beam reflects on a nearby object and returns to the sensor a time of flight (ToF) measurement can be recorded, generating a point in space. Combining multiple measurements from one scan results in a point cloud showing a 2D or 3D view of the sensor surroundings. Most lidar sensors are also capable of returning an intensity measurement indicating the reflectivity of the object, which can give insights about the material of the object and can be used for object classification or tracking.

There are two main categories of lidar sensors, rotating and solid-state. Rotating lidar sensors are the most mature and use a rotor-based mechanism that provides a 360° horizontal *field of view* (FoV) by rotating the scanning sensor. The vertical FoV varies and is determined by the number of laser emitter and receiver pairs, also called channels. Solid-state lidar sensors have no rotating parts and generally have a smaller horizontal FoV [19].

One drawback of the rotating lidar is that the motion of the sensor or observed objects during a scan introduces distortion to the point cloud since each point is received at different times. The distortion introduced by sensor movement can often be removed if the scanning frequency and information about the motion during the scan are known. With additional sensors such as an IMU and a sensor to measure velocity, the motion can be estimated. Each point in the point cloud is then moved a linear distance based on when during the sweep the point was recorded [20].

2.7 Scan matching

Scan matching is a method used to align two point clouds and is sometimes also referred to as *registration*. The use of range measurements such as 2D or 3D point clouds has grown in popularity for robotic and automotive use in recent years. One of the large applications is related to sensing the environment and localization. The goal is to find the correct alignment of a target scan X and

source scan P by determining the correct rotation and translation between them. The target scan is fixed in a coordinate system, while the source scan is the new measurement to be aligned.

Iterative closest point (ICP) is a common technique used for scan matching point clouds. The concept of the algorithm can be simplified and explained in two steps:

1. Compute the correspondence between each point in scan X and P.

2. Compute the translation and rotation which minimizes the distance between each corresponding pair of points.

These steps are iterated until the maximum number of iterations is reached or until a convergence criterion is reached. The resulting transformation can be used as a six-degree of freedom motion estimate between the two scans [21]. Throughout the years, multiple variations of the ICP algorithm have been published, such as point-to-plane ICP and GICP (plane-to-plane) ICP [22].

Another common technique used for scan matching is called the *Normal Distributions Transform* or NDT. Similarly to an occupancy grid, the (NDT) subdivides the 2D or 3D space into cells. Each cell is then assigned a normal distribution which locally models the probability of a point being present. One of the main advantages is that no explicit correspondence between scans has to be computed, instead, the sum of all the normal distributions can be maximized resulting in a direct transform [23].

2.8 GNSS

Today there are multiple systems used for satellite navigation, together they are defined as *Global Navigation Satellite Systems* (GNSS). With GNSS it is possible to achieve accurate, continuous, and worldwide positioning and velocity information. The concept used for localization is *time of arrival* (TOA) ranging and when multiple TOA measurements from different satellites are used it is possible to calculate a position in three dimensions [24].

An inaccurate TOA measurement can greatly worsen the localization performance. This is common in so-called urban canyons, inside cities with tall buildings surrounding the road but can also happen in less urban areas. GNSS signals can bounce on nearby structures before being registered by the receiving sensor, which means the signal has traveled further, giving an incorrect range to the satellite. This effect is called multipath and can also take the form of multiple registrations of the same signal.

2.9 IMU

An IMU is a combined sensor, incorporating an accelerometer, gyroscope, and sometimes a magnetometer. Acceleration is measured by the accelerometer, angular rate is measured by the gyroscope and orientation can be drawn from the magnetometer. It is a widely used sensor in navigation and can be found in products such as smartphones. Often, the measured values are further processed. Integrating acceleration measurements result in a velocity estimate and through double integration a position estimate can be acquired. Likewise, the angular rate can be integrated resulting in an orientation estimate. The process of estimating higher order states from lower order measurements is called dead reckoning. The main disadvantage of using dead reckoning is the accumulation of errors over time.

3
Method

As described in Section 2.3 the front end in graph SLAM processes sensor data and outputs measurement constraints to the back end. The two types of measurement constraints used in this thesis are odometry constraints that describe the motion of the vehicle, and position constraints with robust methods that tie a node to a global position using GNSS measurements.

This section goes over the experimental setup, the front and back end, and the modeling of two types of disturbances. The front end implementation builds upon the open source *hdl_graph_slam* framework and it is modified to handle multiple Lidar sensors, point cloud deskewing, multiple GNSS receivers, offset GNSS receivers, GNSS covariance, and GNSS noise.

3.1 Experimental Setup

This section presents the experimental setup which consists of the vehicle sensor setup from which the data is recorded and the graph SLAM framework used to process the data and estimate the vehicle states.

3.1.1 Vehicle setup

The sensor data is recorded using a passenger vehicle with four lidar sensors, two IMUs, and two GNSS receivers. Detailed sensor specifications such as resolution and update frequency are not relevant to this thesis science the evaluated methods are independent of specific sensor models.

The lidar sensors are automotive grade with 360-degree horizontal field-of-view

Table 3.1: *Transforms between sensors and the vehicle coordinate frame.*

Sensor	x [m]	y [m]	z [m]	roll [rad]	pitch [rad]	yaw [rad]
Ground Truth INS	-0.22	-0.022	0.466	-π	0	0
IMU	0.7	0	0	0	0	0
Front left lidar	1.552	0.518	1.619	0	0	-π
Front right lidar	1.552	-0.513	1.618	0	0	-π
Rear left lidar	0.442	0.518	1.619	0	0	-$\pi/2$
Front right lidar	0.443	-0.513	1.618	0	0	$\pi/2$
Left GNSS receiver	1.240	0.560	1.670	0	0	0
Right GNSS receiver	1.240	-0.560	1.670	0	0	0

placed in an array on a roof rack. However, due to the placement, the field of view is obstructed by the other sensors placed on the same rack. Therefore, the point clouds do not overlap perfectly, making scan matching between sensors challenging.

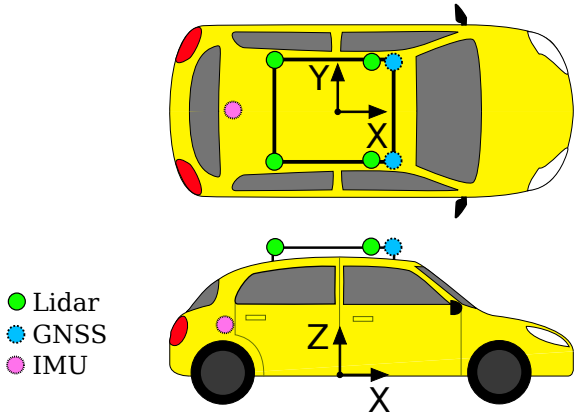


Figure 3.1: *Top and side view of sensor placements on the test vehicle together with a coordinate frame for the vehicle. Four Lidar sensors and two GNSS receivers are located on a roof rack. An IMU is placed inside the vehicle.*

3.1.2 Software

To evaluate the robust optimization techniques in this thesis, open-source software and frameworks have been used as a foundation. The front end is based on a framework by Koide et al. [25], known as *hdl_graph_slam*. Modifications have been made to the framework to make it compatible with the specific data set used.

For solving the optimization problem represented as a factor-graph a general-purpose framework for optimizing graph-based nonlinear error functions called g^{2}o is used [26]. *Point Cloud Library* (PCL) is used for registration and filtering of lidar data [27]. All of the above mentioned software are used together in the open-source framework called *Robot Operating System* (ROS) [28].

3.2 Lidar Odometry Estimation

This section covers how the raw data from four lidar sensors are used to create an odometry measurement. The raw data is in the form of point clouds, is firstly preprocessed by removing irrelevant points, then deskewed with respect to vehicle velocity. An outlier removal filter is applied together with a voxel filter in order to improve processing speed. The preprocessed point clouds are used for scan matching which outputs an odometry estimation.

3.2.1 Point cloud pre-processing

Due to the large amounts of data generated from lidar sensors, some pre-processing steps are used to reduce the point cloud size. This is done to improve the speed and accuracy of the scan matching algorithm.

The first processing step taken is the removal of points that do not contain any relevant information for calculating the motion between scans. Therefore all points in a radius of 5.5 meters around the lidar sensor are removed as they mostly contain points reflected from the vehicle and the road below, also points outside the maximum specified sensor range of 100 meters are removed.

Because rotating lidar sensors are used, a simple deskewing algorithm was implemented that corrects for the sensor rotation and ego motion using IMU and velocity measurements.

Then two filters in PCL are used. Radius outlier removal is used to remove points in the point cloud which have less than three neighboring points inside a one-meter radius. Then a voxel grid filter is applied with a voxel size of 0.3 meters which improved processing speed for the scan matching algorithm while still getting good odometry estimates.

3.2.2 Scan Matching

To calculate the relative pose between scans the choice of scan matching algorithm was determined through experiments. The experiments showed that GICP worked best together with the sensor setup used.

The initial guess for scan matching is calculated with the assumption that the velocity and heading remain constant between two scans. The estimated distance traveled between scans is calculated as the velocity multiplied by the elapsed

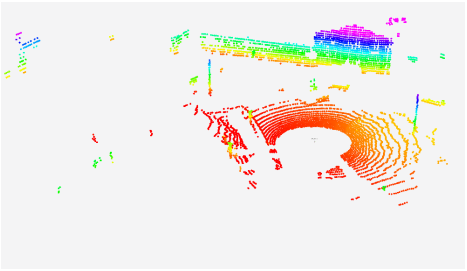


Figure 3.2: *Pre-processed point cloud from the rear left lidar.*

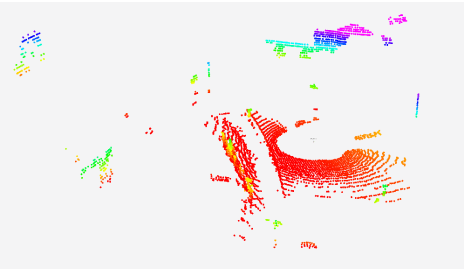


Figure 3.3: *Pre-processed point cloud from the front right lidar.*

Table 3.2: *Parameters used with GICP*

Parameter	Value
Convergence threshold	0.001 [m]
Maximum Iterations	64 [-]
Maximum Correspondence distance	1.0 [m]

time.

During the experiments, the scan matching algorithm had difficulties finding correct corresponding points between lidar sensors placed diagonally due to very few overlapping points. Figure 3.2 and Figure 3.3 together showcase the small overlap between two point clouds captured at the same time. In order to decrease the effects of incorrect correspondences, all four point clouds are concatenated into one larger point cloud that is used as the scan matching target, see Figure 3.4.

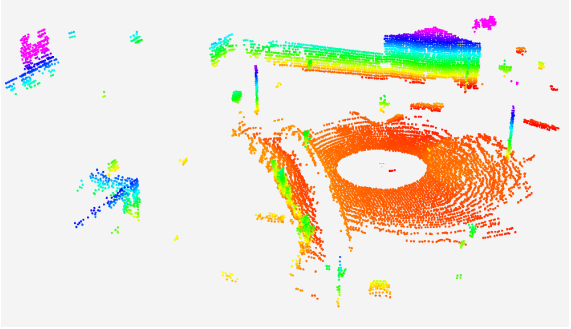


Figure 3.4: *The resulting point cloud from all four lidar sensors is used as the scan-matching target in the odometry estimation. Each point in the figure is enlarged for better visibility.*

With the transforms specified in Table 3.1 the relative position of each lidar to the vehicle coordinate frame is known. A measurement $\mathbf{z}_{ij}$ describes the transformation between two consecutive states $\mathbf{x}_i$ and $\mathbf{x}_j$ are expressed in the vehicle coordinate frame of $\mathbf{x}_i$ and consists of the translation and a quaternion representation of orientation. The resulting measurement $\mathbf{z}_{ij}$ from GICP is on the form:

$$\mathbf{z}_{ij} = \begin{pmatrix} t_x & t_y & t_z & q_x & q_y & q_z & q_w \end{pmatrix}^T. \tag{3.1}$$

3.3 GNSS

In the data set used, GNSS measurements are attained from the two GNSS receivers placed on the top-left and top-right side of the vehicle, according to Figure 3.1. The receivers provide altitude and latitude-longitude coordinates which are transformed into the local *universal transverse mercator* (UTM) coordinate frame along with an estimated covariance matrix describing the quality of the measurement. The measurement $\mathbf{z}_i$ describes the UTM position of the GNSS receiver.

Using the transforms in Table 3.1 the offset between the GNSS receiver and the vehicle coordinate frame is calculated in the UTM coordinate frame and accounting for the vehicle orientation. Using quaternion rotation the offset $\mathbf{P}'_i$ between the GNSS receiver and vehicle coordinate frame is

$$\mathbf{P}'_i = \mathbf{q}_i \cdot \mathbf{P}_i \cdot \mathbf{q}_i^* \tag{3.2}$$

where $\mathbf{q}_i$ is the current orientation of the vehicle, its conjugate $\mathbf{q}_i^*$, and $\mathbf{P}_i$ is the translation offset from receiver to the vehicle frame expressed as a quaternion with scalar part $q_w = 0$. The operation $\cdot$ is quaternion multiplication. The offset $\mathbf{P}'_i$ is added to the measurement $\mathbf{z}_i$ resulting in

$$\mathbf{z}'_i = \mathbf{z}_i + \mathbf{P}'_{i[1:3]}, \tag{3.3}$$

where [1:3] selects the first three elements of the quaternion and discards the quaternion component q_w. The measurement $\mathbf{z}'_i$ describes the position of the vehicle frame in the UTM coordinate frame.

Each GNSS measurement is then associated with a node based on finding the measurement $\mathbf{z}'_i$ that minimizes the difference in timestamp to $\mathbf{x}_i$. To make calculations simpler, the first acquired GNSS measurement is used for zero correction of all GNSS measurements in order to transform the measurement to local map coordinates.

3.4 Formulation of Optimization Problem

When formulating the SLAM problem as a factor graph optimization problem, information from a wide range of sensors can be incorporated into the solution. In

Table 3.3: *Parameters that decide when to initialize a new node.*

Parameter	Value
Translation threshold	1.0 [m]
Rotation threshold	0.03 [rad]

the formulation of the optimization problem for this thesis, the lidar odometry estimation is used to formulate relative pose constraints between each consecutive node, and GNSS measurements are used as position constraints to associate each node with a UTM coordinate.

We try to find the values in the state vector of each node $\mathbf{x}^*$ that minimizes the cost function

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \mathbf{F}^{odom}(\mathbf{x}) + \mathbf{F}^{GNSS}(\mathbf{x}), \tag{3.4}$$

which includes odometry and GNSS measurements. This section explains how $\mathbf{F}^{odom}(\mathbf{x})$ and $\mathbf{F}^{GNSS}(\mathbf{x})$ are calculated.

3.4.1 Relative Pose Constraints

The front end scan matching continuously estimates the vehicle odometry relative to the most recent node $\mathbf{x}_i$. Whenever the change in translation or rotation exceeds a threshold, an event is triggered. This event initializes a new node $\mathbf{x}_{i+1} = \mathbf{x}_j$ and creates a measurement vector $\mathbf{z}_{ij}$. The scan matching algorithm starts over and begins estimating the odometry relative to the new node. The state vector of $\mathbf{x}_j$ is initialized by composing the relative pose $\mathbf{z}_{ij}$ to the estimate of the previous node $\mathbf{x}_i$.

To formulate the error function the mathematical pose operators defined in Section 2.5 are used. The idea is to calculate the difference between the measured relative pose and the current estimate of the relative pose. The error function is formulated

$$\mathbf{e}_{ij}^{odom}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) = (\mathbf{z}_{ij}^{-1} \oplus (\mathbf{x}_i^{-1} \oplus \mathbf{x}_j))_{[1:6]}, \tag{3.5}$$

where $(\cdot)_{[1:6]}$ keeps the first six elements and discards the quaternion component q_w, which is done as a part of the manifold optimization as explained in Section 2.3.1.

Let $\mathcal{C}$ be a set with indices of consecutive pairs i, j with an odometry edge. We get the cost function

$$\mathbf{F}^{odom}(\mathbf{x}) = \sum_{\langle i,j \in \mathcal{C} \rangle} (\mathbf{e}_{ij}^{odom})^T \mathbf{\Omega}_{ij} (\mathbf{e}_{ij}^{odom}) \tag{3.6}$$

The information matrix $\mathbf{\Omega}_{ij}$ is considered constant for every odometry measurement and has been tuned to approximately the same size as the information matrix for the position constraints in Section 3.4.2.

3.4.2 Position Constraints

Position constraints are used together with GNSS measurements so that the localization estimate gets a global coordinate. For a set $\mathcal{G}$ with indices of every node $\mathbf{x}_i$ which has a corresponding GNSS measurement $\mathbf{z}_i$ a position constraint is created. The constraints are added as unary edges to the node $\mathbf{x}_i$ and the error function is

$$\mathbf{e}_i^{GNSS}(\mathbf{z}_i, \mathbf{x}_i) = \begin{pmatrix} t_x - z_x \\ t_y - z_y \\ t_z - z_z \end{pmatrix}. \tag{3.7}$$

The sum of all constraints creates the cost function

$$\mathbf{F}^{GNSS}(\mathbf{x}) = \sum_{i \in \mathcal{G}} (\mathbf{e}_i^{GNSS})^T \mathbf{\Omega}_i (\mathbf{e}_i^{GNSS}). \tag{3.8}$$

The information matrix $\mathbf{\Omega}_i$ is calculated as the inverse of the estimated covariance matrix reported by the GNSS receiver.

In Section 2.4 two different methods for robust optimization are introduced, *Dynamic Covariance Scaling* and *Realizing*, *Reversing*, *Recovering*. These methods are applied to modify the cost function (3.8) in different ways to reduce the effect of outliers such as GNSS multipath measurements.

DCS modifies the cost function (3.8) by adding the scaling factor s_i^2 to the individual constraints

$$\mathbf{F}^{GNSS}(\mathbf{x}) = \sum_{i \in \mathcal{G}} s_i^2 (\mathbf{e}_i^{GNSS})^T \mathbf{\Omega}_i (\mathbf{e}_i^{GNSS}). \tag{3.9}$$

RRR, on the other hand, modifies the set $\mathcal{G}$ and removes measurements entirely if they are considered outliers.

3.5 Modeling of GNSS Disturbances

Common scenarios where disturbances in GNSS occur are tunnels or urban canyons where the line-of-sight to the GNSS satellites is obscured. Figure 3.5 shows how structures can disturb measurements, in these scenarios, it is difficult to obtain a reliable ground truth measurement as the INS relies on good localization from GNSS measurements. Therefore, to evaluate DCS and RRR two types of disturbances are modeled and added to the measurements.

The two types of disturbances being modeled are Gaussian and biased. Both scenarios are modeled at the same position of the data set and are meant to resemble a vehicle entering and exiting a tunnel or urban canyon. First disturbances are added to the measurement, followed by an outage, followed by disturbances. In both scenarios, the measurement covariance is not modified.

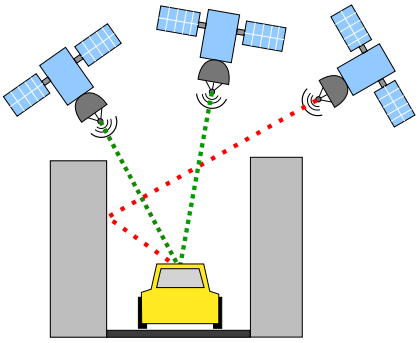


Figure 3.5: *Illustration of the scenario being modeled and visualized as an urban canyon where one of the GNSS satellite line of sight is obscured by surrounding structures. A faulty measurement reflected off the surface results in an incorrect range measurement between the satellite and receiver.*

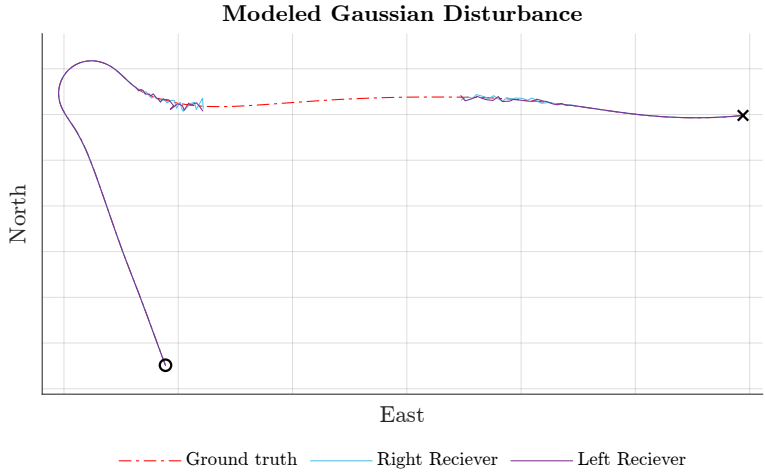
In order to compare the different methods for robustness against outliers repeatability in the added disturbance is needed. To ensure repeatability the deterministic pseudo-random generator *Mersenne Twister 19937* is used. The repeatability ensures that the same factor graph configuration is used when evaluating DCS and RRR. In the same way, the repeatability enables a fair comparison between different tuning.

3.5.1 Gaussian Disturbance

Modeling the Gaussian disturbance, GNSS is available for the first 15 seconds of the data set. Next, the vehicle enters the tunnel, and increasing Gaussian noise is added during a five-second period before the measurements are cut completely. After 20 additional seconds, the GNSS signal with added Gaussian noise is regained. The added noise decreases until a fully stable signal is reached when exiting the tunnel after 5 additional seconds.

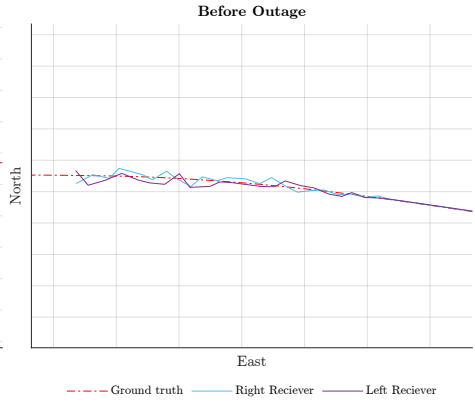
The added noise $\mathcal{N}(0, \sigma^2)$ is normally distributed with zero mean and standard deviation σ. It is added to the latitude, longitude, and altitude independently and is linearly scaled up to $\sigma_{max} = 7.15$ [1] during the five-second period. At the end of the outage, noise is added again and scaled down for five seconds. Starting with a standard deviation of σ_{max}, the noise is scaled down to $\sigma = 0$ reaching a stable signal at the end of the modeled tunnel. The GNSS measurements from the left and right receivers with the added disturbances can be seen in Figure 3.6. The timestamp of the GNSS measurement is used as the seed to the random generator resulting in the same disturbance being added every time the simulation is run.

[1] The standard deviation σ_{max} is shown with scaled units.



(a) Overview of the outage and the Gaussian disturbance added. The start is marked with X and the end is marked with O.

(b) Enlargement of the noise after the outage.



(c) Enlargement of the noise before the outage.

Figure 3.6: *The resulting measurements being used as position constraints from the modeled Gaussian disturbance, shown from above.*

3.5.2 Multipath Disturbance

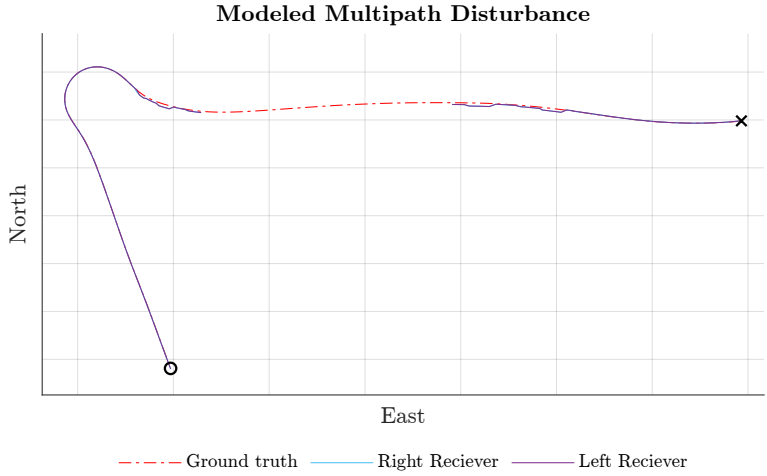
Multipath disturbances occur when the range measurement between the satellite and GNSS receiver is incorrect. This can occur if the signal is reflected off surrounding structures, see Figure 3.5. Because the range measurement during multipath disturbances does not correspond to the true distance between the receiver and the satellite a shift in the position estimate occurs.

Localization in the lateral and longitudinal directions is more important than the

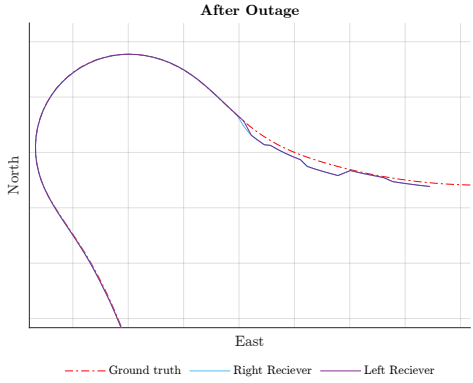
altitudinal direction for autonomous applications. High accuracy in these directions is required when driving on public roads, due to narrow lane width and a high forward velocity. Obstacles and other road users also exist in this plane, which requires precise localization for other tasks such as path planning and control. In order to test the effect of lateral and longitudinal multipath disturbances, a second multipath scenario is modeled using a different section of the collected data set.

The modeled multipath scenario consists of 15 seconds without any added disturbances followed by five seconds of multipath disturbances and then a 20 seconds outage. After the outage disturbances are added for five seconds followed by no added disturbances for the rest of the data set. A shift in position is added to the longitude and latitude of the GNSS measurement every n seconds to emulate the signals bouncing off different structures. The shift size is randomly generated between 0 and 3.75 [2], and the resulting disturbance can be seen in Figure 3.7.

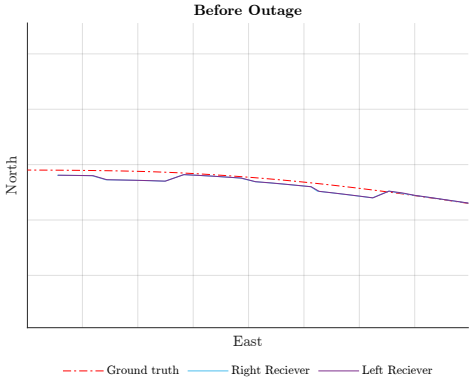
[2] In scaled units.



(a) Overview of the section with outage and multipath disturbance added. The start is marked with an X and the end is marked with O.



(b) Enlargement of the noise after the outage.



(c) Enlargement of the noise before outage.

Figure 3.7: *The resulting measurements being used as position constraints from the modeled Gaussian disturbance.*

4
Experiments

Three scenarios are investigated in this chapter. Different input data to the graph SLAM system is what separates the scenarios. The first scenario is the initial segment of the provided data set in which multipath disturbances are present on the GNSS measurements from the left receiver. The second and third scenarios utilize a different segment of the data set consisting of a straight drive section, a roundabout, and a second straight segment. GNSS measurements have been altered with Gaussian disturbances and biased disturbances respectively. Additionally, measurements have been removed for an extended period to replicate an outage from driving through a tunnel or urban canyon.

The chapter begins by describing how the performance of the system is evaluated. The scaling of results is explained in Section 4.1.2. A sensitivity analysis of parameter selection for DCS in Section 4.2 and for RRR in Section 4.3. Followed by detailed results from the three scenarios using RRR, and DCS, in relation to a baseline that is without any robust method. The results are discussed and analyzed shortly in the respective section.

4.1 Evaluation

In order to evaluate the effects of using the robust methods when GNSS disturbances are present, a baseline for performance is needed. The baseline is established by estimating the vehicle pose using the proposed graph SLAM system and using Gauss-Newton to find a solution to (3.4) without the use of DCS or RRR. Then, the best estimations of the poses are also found using the robust methods before solving (3.4) using Gauss-Newton. The estimated poses are compared to

the position and orientation of ground truth in order to evaluate and compare the different solutions.

The robust methods are also evaluated on how sensitive they are in terms of parameter choices. The parameters affect how much the robust methods alter the solution. DCS has the free parameter Φ which affects if and by how much the covariance of a deviating measurement is scaled. The free parameter of RRR is the p-value used in the χ^2 tests which affects whether a measurement or cluster of measurements are considered outliers and should be removed.

4.1.1 Ground Truth

To evaluate the position and orientation accuracy, a precise GNSS-aided *inertial navigation system* (INS) is utilized as a ground truth reference. The INS makes use of a Kalman filter that fuses measurements from the dual antenna RTK GNSS system and an IMU, providing reliable position, velocity, and orientation. In order to analyze the accuracy of the graph SLAM solution, the ground truth data is linearly interpolated to the same time series as the estimated data points.

In every time step, error residuals are calculated relative to the ground truth. First, translation errors are calculated. The translation error in the x-direction relative to the ground truth gives the longitudinal error, the y-direction gives the lateral error and the z-direction gives the error in altitude. Similarly, the orientation errors are calculated as the difference in quaternion rotation for every given time step. The orientation error is converted to roll, pitch, and yaw for easier evaluation.

4.1.2 Scaled Results

To protect confidential data, all data in figures and results use scaled evaluation metrics and units. *Root mean square error* or RMSE is commonly used to evaluate performance and accuracy. To compute the RMSE first a residual $\mathbf{e}_i = \hat{\mathbf{x}}_i - \mathbf{x}_i$ is calculated at each data point i, where $\hat{\mathbf{x}}_i$ is the estimate and $\mathbf{x}_i$ the ground truth. The RMSE is calculated as the square root of the mean residual error where n is the total number of data points

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathbf{e}_i)^2}. \tag{4.1}$$

To protect the accuracy of translation and rotation estimates the RMSE is scaled with a scaling factor S

$$\text{SRMSE} = \frac{\text{RMSE}}{S}. \tag{4.2}$$

The scale factor S is chosen according to the benchmark performance. For each scenario, the system's performance without using robust methods is considered the baseline and determines the scaling factor. A SRMSE less than one indicates better performance while a SRMSE larger than one indicates less precision.

4.2 DCS Free Parameter Φ

A small value of the free parameter Φ results in an aggressive scaling for smaller deviations, or errors, in the measurements while a large value of Φ only scales the covariance of larger errors and with a smaller factor. The impact of Φ while using DCS is further explained in Section 2.4.1 and the impact of Φ on the scaling factor s can be seen in Figure 4.1.

The SRMSE for translation and orientation states for different scenarios and values of Φ can be seen in Table 4.1. The value of Φ is varied between 0.1 and 10. This is done for the multipath scenario, tunnel scenario with Gaussian disturbance, and tunnel scenario with multipath disturbances. The state estimation of translation states x and y and orientation state yaw are presented separately as the estimation of these states is more important for the localization from a safety perspective. Therefore, when evaluating the value of Φ, states x, y, and yaw are considered more significant.

The state estimation is improved or similar to the baseline for all values of Φ for all scenarios except in the tunnel scenario with Gaussian noise in which the roll and pitch have a SRMSE larger than one for $\Phi = 0.5$ and $\Phi = 1.0$. The state estimation SRMSE of z is also worse than the baseline in the tunnel multipath scenario when Φ is set to a value lower than 2.5. These results in Table 4.1 indicate that DCS improves the state estimation of Φ if the value is set between 0.1 and 10.

In the case of multipath disturbances, DCS improves state estimation for the z state across the different values of Φ while performing similarly to the baseline for all other states. The results from the test can be seen in Table 4.1. This is expected due to the disturbances present in the multipath scenario in the z-direction. The opposite is true for the tunnel multipath scenario where multipath disturbances are added in the x and y direction. For the tunnel multipath scenario state estimation SRMSE for z is greater than the baseline if Φ is set to less than 2.5. As Φ is set to a lower value more correction is made and while this improves state estimation for the xy state in tunnel multipath scenario, the state estimation for z suffers due to less trust in the more accurate z measurements. When measurements are disturbed in all directions x, y, and z, as in the Tunnel Gaussian scenario, improved state estimation increases as the Φ is set to a lower value. For example, the SRMSE of xy in the tunnel Gaussian scenario in Table 4.1 for $\Phi = 10$ is 0.74 and when $\Phi = 0.1$ the SRMSE is 0.64. The SRMSE of state estimation of yaw goes from 0.68 to 0.57 when Φ is set to 10 and 0.1 respectively.

The tunnel scenario with added Gaussian noise shows a more significant impact of DCS and the value of Φ compared to the multipath scenarios. The resulting SRMSE for this scenario can be seen in Table 4.1. This is due to the larger size of added noise in the tunnel scenario with added Gaussian noise. The discrepancy between the accuracy of the noise and the set covariance of the measurements is also greater, implying a correction of the covariance estimation can improve results.

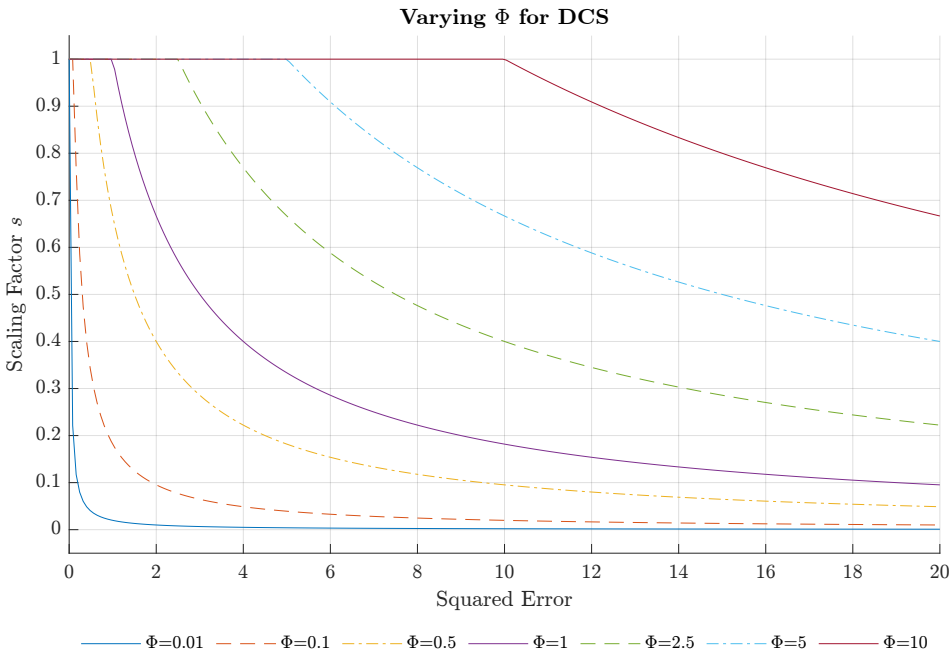


Figure 4.1: *The scaling factor s used in DCS and how it is affected by varying values of the parameter Φ. The scaling factor is dependent on the size of the squared error for a measurement.*

Parameter selection of $\Phi = 0.5$ and $\Phi = 0.1$ performs better across the different scenarios when considering the most important states x, y, and yaw. When a larger value of Φ is used the covariance is not scaled sufficiently in relation to the faulty measurements. If a strict improvement of each state is sought after Φ should be set to 2.5 since the results presented in Table 4.1 show the same performance or improved results across all scenarios and states. If multipath disturbances in the x, y are sought to be minimized, Φ should be set to 0.5 or 0.1.

Varying Φ between scenarios would improve performance but for further comparisons with RRR $\Phi = 0.5$ will be used for simplicity.

Table 4.1: *SRMSE for estimated translation states (x,y,z) and orientation states roll, pitch, and yaw (R,P,Y) when using DCS and varying Φ for different scenarios. Best result for each state is bolded.*

Scenario	States	$\Phi = 0.1$	$\Phi = 0.5$	$\Phi = 1.0$	$\Phi = 2.5$	$\Phi = 5$	$\Phi = 10$
Multipath	xy	**0.99**	1.00	**0.99**	1.00	1.00	1.01
	z	0.87	**0.79**	0.80	0.91	0.96	0.96
	RP	1.03	1.03	1.02	**1.01**	**1.01**	**1.01**
	Y	**0.97**	**0.97**	1.01	**0.97**	**0.97**	**0.97**
Tunnel Multipath	xy	**0.37**	0.43	0.41	0.83	0.96	0.94
	z	1.94	1.98	1.92	**0.85**	0.90	0.96
	RP	0.89	0.94	0.89	0.88	**0.87**	0.93
	Y	**0.38**	0.41	0.39	0.74	0.73	0.64
Tunnel Gaussian	xy	0.65	**0.47**	0.72	0.83	0.85	0.74
	z	0.69	0.74	0.93	0.51	0.54	**0.46**
	RP	0.93	1.24	1.14	0.99	1.08	**0.90**
	Y	**0.57**	0.77	0.73	0.69	0.79	0.68

4.3 RRR P-Value

RRR is based on consistency checks where measurements are compared within clusters, and clusters are compared with other clusters. The consistency check is a χ^2 hypothesis test evaluated with a p-value and is presented in Section 2.4.2. The value p is usually chosen to 0.95 with RRR in order to have high confidence in the result. Varying the value of p in for the multipath scenario, tunnel scenario with Gaussian disturbance, and tunnel scenario with multipath disturbances gives the resulting SRMSE shown in Table 4.2. A lower p-value will result in removing measurements with smaller residuals from the set of measurements. This introduces the risk of removing good quality measurements and worsening performance.

When comparing the performance of state estimation using RRR states x, y, and yaw are considered more significant. This is due to having to avoid other vehicles and obstacles that exist on the same xy plane as well as the importance of knowing the direction of travel. The roll, pitch, and z are still important but have less

of a direct impact if the estimation is incorrect. The results for the multipath scenario show similar results compared to the baseline for all values of p except $p = 0.80$ which shows worse or similar performance for all state estimation. This is the result of too many measurements being removed.

It is noticeable that the SRMSE for states x, y, roll, pitch, and yaw show a more similar performance over different values of p compared to z. This can be explained by the higher reliance on lidar odometry as measurements are removed. Lidar odometry performs significantly worse in altitudinal state estimation which will result in a larger difference in the z state estimation. RRR shows an improved performance in state estimation when the Gaussian and multipath disturbances are present along with a GNSS outage when the value of p is set between 0.80 and 0.99.

Table 4.2: *SRMSE for estimated translation states (x,y,z) and orientation states roll, pitch, and yaw (R,P,Y) when using RRR and varying p-value for different scenarios. The best results for each scenario are bolded.*

Scenario	States	$p = 0.80$	$p = 0.85$	$p = 0.90$	$p = 0.95$	$p = 0.99$
Multipath	xy	**1.01**	1.02	1.02	1.03	1.02
	z	1.34	**0.96**	0.98	0.97	0.99
	RP	1.11	**1.01**	**1.01**	**1.01**	**1.01**
	Y	1.20	**0.99**	**0.99**	**0.99**	**0.99**
Tunnel Multipath	xy	0.41	**0.40**	0.42	0.41	0.51
	z	2.58	2.49	2.47	2.63	**1.86**
	RP	1.00	1.01	1.00	**0.99**	1.02
	Y	0.46	0.44	**0.42**	0.48	0.45
Tunnel Gaussian	xy	0.59	**0.57**	0.73	0.59	0.66
	z	**1.05**	1.18	1.17	1.21	1.18
	RP	**0.96**	0.97	1.35	0.97	1.15
	Y	0.55	0.54	**0.52**	0.55	0.67

4.4 Multipath Scenario

During the first 38 seconds of the collected data set, disturbances were observed in the measurements of the left GNSS receiver. The disturbance is a varying shift in the measured altitude, seen in Figure 4.2. Due to the characteristics of the disturbance, an assumption is made that it is a multipath disturbance causing the faulty position measurement. There are no observed disturbances in longitude or latitude. Both GNSS receivers are high end automotive grade and report a position uncertainty with good accuracy. In Figure 4.2 the estimated altitude and standard deviation of the estimate can be seen for the ground truth and both receivers. Even though the GNSS receivers are high end the left receiver underestimates the uncertainty during the first 13 seconds.

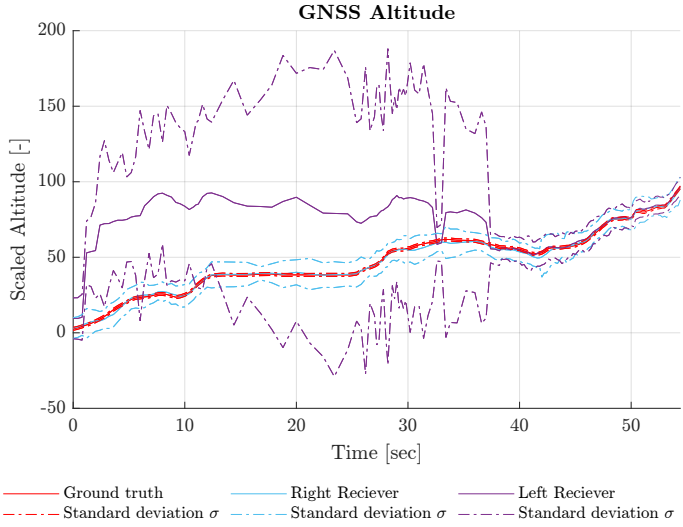


Figure 4.2: *Measured GNSS receiver altitude transformed into the vehicle coordinate frame. The dotted line is the estimated standard deviation from the receiver and ground truth. Disturbances acting on the left receiver altitude can be seen, the right receiver follows the ground truth without any major disturbances. Note that the standard deviation of the left receiver is outside the ground truth during the first 13 seconds.*

The estimated position using Gauss-Newton without any robust method and using Gauss-Newton together with DCS, and RRR is compared with the ground truth INS. Translation errors are shown in Figure 4.3. Longitudinal and Lateral errors are almost identical with and without any robust method being used. Longitudinally there is a bias in the error of 0.6 scaled distance units. Laterally the error is outside the 3σ covariance during a large portion of the data set.

In altitudinal error, the difference between the two robust methods is more visible. Because the standard deviation from the GNSS receiver is used to calculate the information matrix $\mathbf{\Omega}$ used in the cost function (3.8) the effect of the faulty altitude measurements is minimized. However, it makes it more challenging for the robust methods to find measurements that are incorrect because the error they introduce becomes small. Additionally, together with the right receiver which does not have any disturbances the standard Gauss-Newton solution estimates an altitude very close to the estimate produced by RRR. It becomes challenging for RRR to identify incorrect measurements when the residual in the χ^2 tests becomes small. Selecting a smaller p-value improves the result but with the risk of discarding too many measurements, an example of this can be seen in Table 4.2 the SRMSE decreases for the multipath scenario when choosing a p-value of 0.85 and when choosing $p = 0.80$ too many measurements are removed and the SRMSE is worsened.

When using DCS the altitudinal error is smaller without negatively impacting the other translational states. This indicates that the correct measurements are scaled while keeping good measurements untouched.

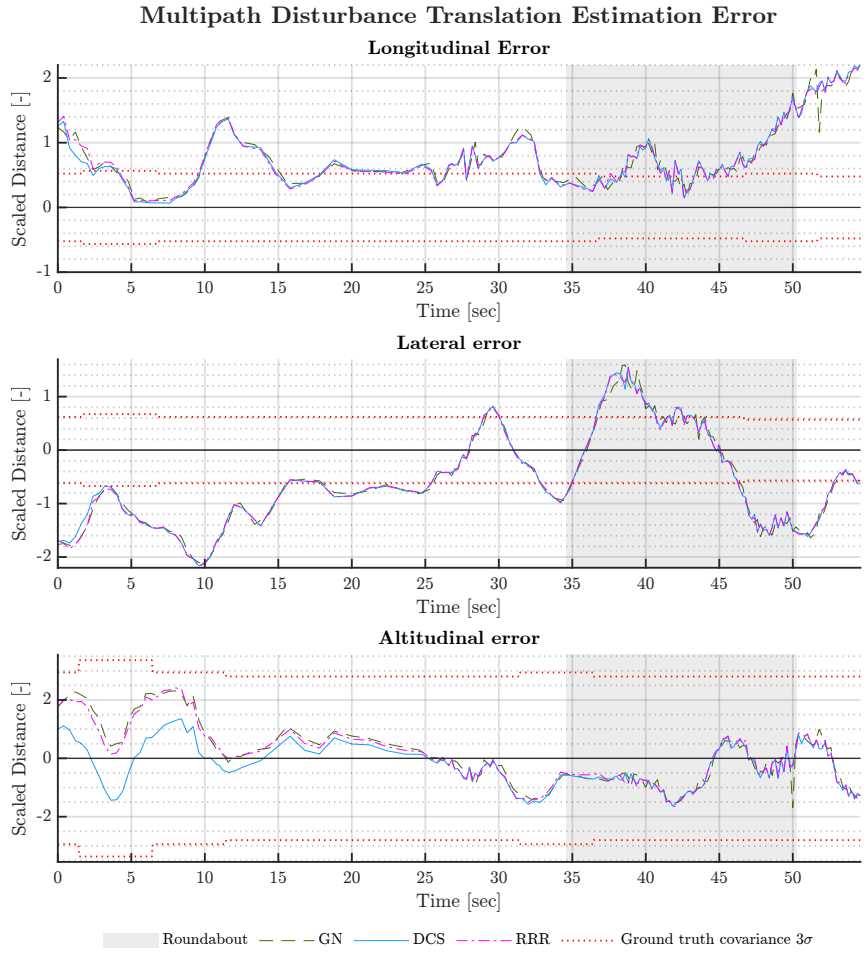


Figure 4.3: *Translation errors relative to the ground truth. Gauss-Newton is compared to DCS with* $\Phi = 0.5$ *and RRR with* $p = 0.90$*. The longitudinal and lateral errors are similar for all three methods. During the first 25 seconds DCS improves the altitude error and RRR makes a minor improvement.*

The errors in orientation estimates in Figure 4.4 are similar for all methods. In Figure 4.5 the evaluated trajectory is shown and a sharp turn is followed by a roundabout. This is reflected mostly in the yaw error, the first turn occurs between the second 12 and 27, and the roundabout is highlighted in the figure. During these periods the yaw error is elevated as a result of difficulties estimating the odometry during high angular velocity. The estimated roll for all methods

has a bias of approximately -1 scaled units and the pitch error is slightly degraded during the roundabout.

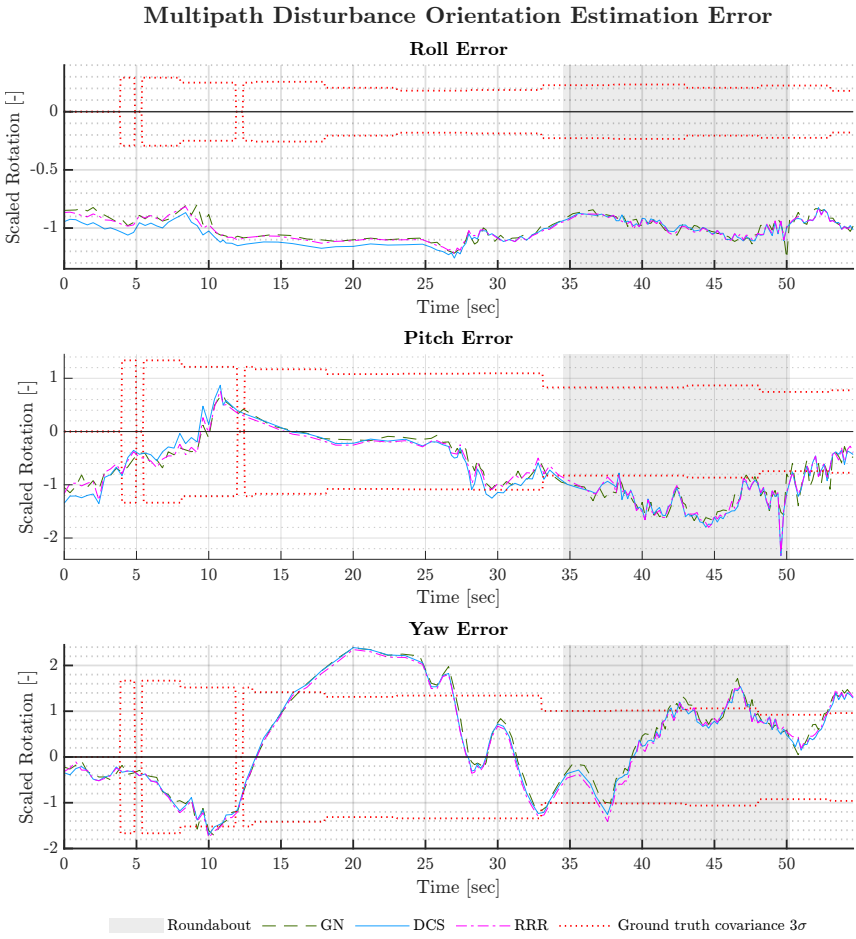


Figure 4.4: *Orientation errors compared to ground truth. Gauss-Newton is compared to DCS with* $\Phi = 0.5$ *and RRR with* $p = 0.90$*. A large variation in yaw can be seen during the roundabout and during 12-27 seconds.*

The SRMSE from the multipath scenario for all six states can be seen in Table 4.3. The results show that the localization performance is similar for the baseline, DCS, and RRR. Except for the altitudinal SRMSE where DCS is slightly better which also is seen in Figure 4.3. The SRMSE of only using lidar odometry shows the largest SRMSE for the altitude estimate which can be associated with the large SRMSE for pitch angle which leads to a large deviation from the true altitude over longer periods of time.

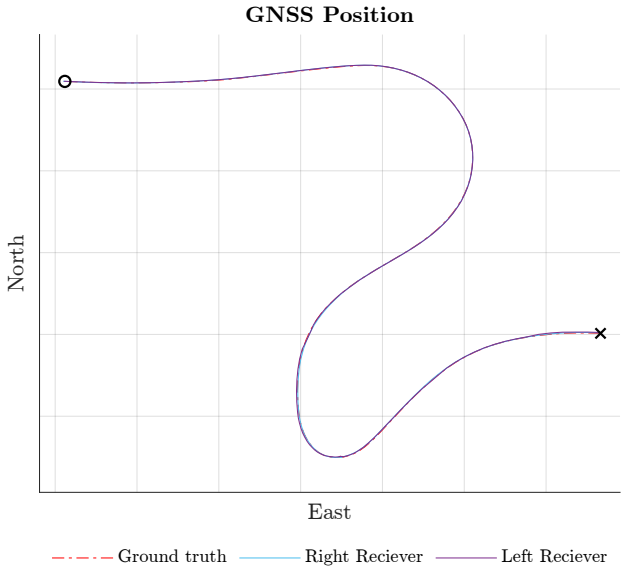


Figure 4.5: *The measured GNSS position shows the driven trajectory during the first 55 seconds of the data set used. The test vehicle starts at ×, and drives from a parking space followed by a sharp right turn onto a public road. The right turn is almost directly followed by a roundabout in which the vehicle exits and the evaluated data ends at ○.*

Table 4.3: *Multipath scenario SRMSE for pose. The estimated state is shown in the first column. The second column shows the SRMSE for the baseline using only Gauss-Newton to estimate the best state configuration. The third column shows SRMSE when using only lidar odometry. In the fourth column, SRMSE is shown for state estimation using DCS. The last column presents the SRMSE for state estimation using RRR. The lidar odometry is localization without any GNSS constraints.*

	Gauss-Newton	Lidar Odometry	DCS	RRR
Longitudinal	1	2.43	0.999	0.997
Lateral	1	1.87	1.037	1.037
Altitudinal	1	35.44	0.786	0.980
Roll	1	1.00	1.024	1.008
Pitch	1	2.44	1.038	1.006
Yaw	1	1.58	0.970	0.987

4.5 Modeled Scenarios

The two modeled scenarios try to resemble a tunnel or urban canyon. The scenarios consist of a GNSS outage with modeled noise added five seconds before and five seconds after the outage. Results are presented for two noise characteristics, Gaussian noise, and a bias. The bias represents the noise phenomenon called multipath in which GNSS signals bounce off nearby buildings, resulting in distorted measurements.

4.5.1 Gaussian Disturbance

The noise added in the Gaussian tunnel scenario is zero mean with a standard deviation of 7.15 scaled distance units and an overview of the data can be seen in Figure 3.6. The scaled error of the estimation of translational states compared to the ground truth in the tunnel scenario using the baseline and using robust methods DCS, and RRR can be seen in Figure 4.6. For the translational states, the baseline has large error spikes in the regions with added noise. The spikes occur both as positive and negative which is expected since the shape of the noise is Gaussian with a zero mean. When the robust methods are used, the spikes are mitigated, and DCS manages to stay close to the ground truth covariance 3σ during the outage for the longitudinal and lateral states. Both RRR and DCS has a similar state estimation with RRR having a larger error.

The longitudinal state error is larger in the second half and after the roundabout when using the robust methods compared to the baseline. This could be due to the higher reliance on lidar odometry which has a worse performance in the longitudinal direction compared to the baseline. The roundabout also poses a challenge for lidar odometry due to large changes in orientation.

There is a small constant error at the beginning of a bias in the estimation of longitudinal and lateral states. Since the error is constant from the beginning of the run and exists for the baseline and when using robust methods, this could be because of a faulty initial guess. The same spikes seen in the translational and lateral state estimation for baseline in Figure 4.6 can be seen in the yaw estimation error in Figure 4.7. When robust methods are utilized, the spike stemming from the disturbances before the outage, and the spike during the outage are mitigated. The yaw error spike from the second disturbance interval is suppressed slightly. Overall, the yaw error is close to or within the 3σ throughout the run when using the robust methods, and RRR performs better when estimating the yaw state than DCS.

During the outage, orientation estimates oscillate which indicates a discrepancy in the lidar odometry. Since four lidars are used, the oscillations could stem from a faulty sensor transform, or that the concatenation of the four lidar scans result in different biases in the performance depending on the time step of the calculations. The SRMSE for all estimated states for the tunnel scenario with Gaussian noise and a GNSS outage is presented in Table 4.4. The results show an improved

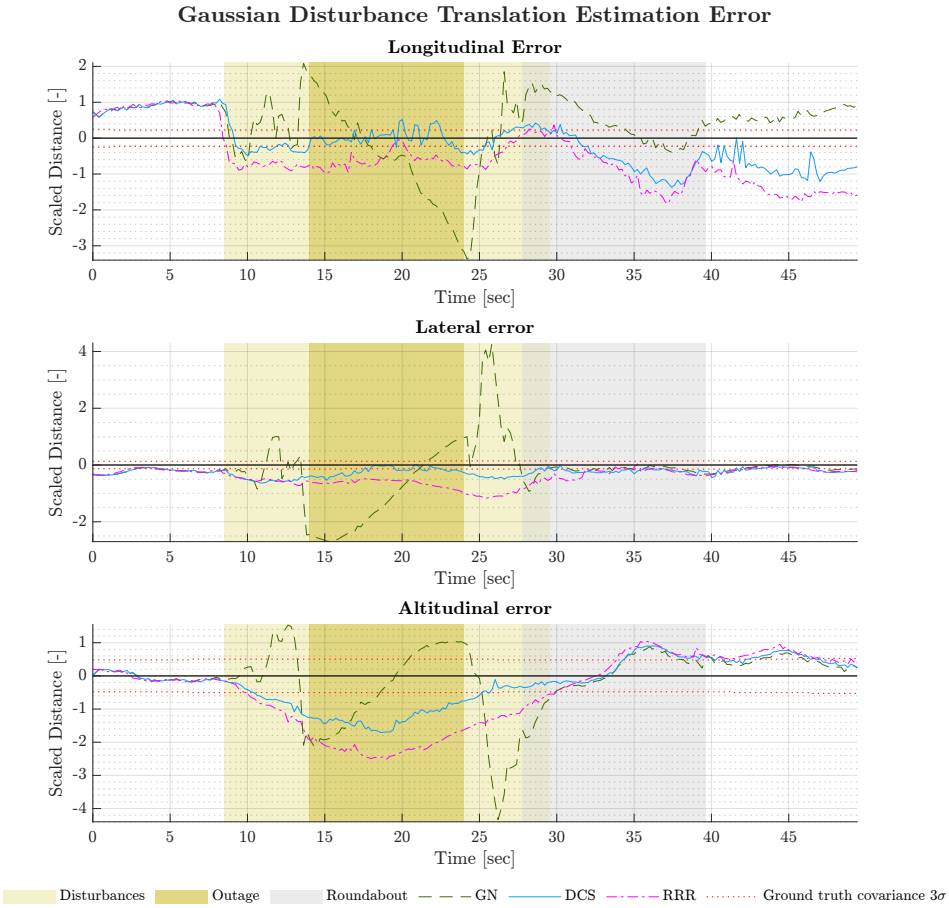


Figure 4.6: *Translation errors for the modeled Gaussian disturbance using the parameter selection* $\Phi = 0.5$ *for* DCS *and* $p = 0.90$ *for* RRR.

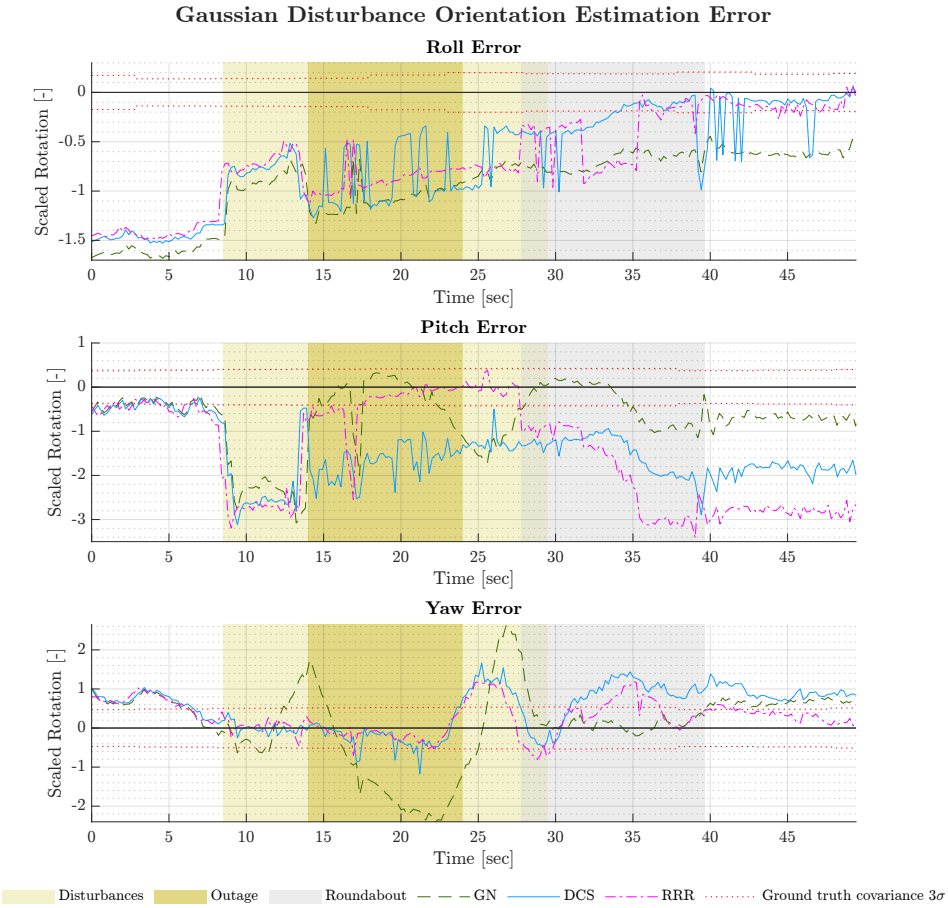


Figure 4.7: *Orientation errors for the modeled Gaussian disturbance using the parameter selection* $\Phi = 0.5$ *for* DCS *and* $p = 0.90$ *for* RRR.

or similar performance using DCS compared to RRR except for the state yaw.

Table 4.4: *Gaussian disturbance SRMSE for pose. The estimated state is shown in the first column. The second column shows the SRMSE for the baseline using only Gauss-Newton to estimate the best state configuration. The third column shows SRMSE when using only lidar odometry. In the fourth column, SRMSE is shown for state estimation using DCS with Φ set to 0.5. The last column presents the RMSE for state estimation using RRR with a p-value of 0.90.*

	Gauss-Newton	Lidar Odometry	DCS	RRR
Longitudinal	1	7.10	0.666	0.968
Lateral	1	3.06	0.288	0.499
Altitudinal	1	25.33	0.825	1.175
Roll	1	0.52	0.825	0.819
Pitch	1	3.73	1.649	1.886
Yaw	1	0.91	0.766	0.518

4.5.2 Multipath Disturbance

The disturbance in the multipath scenario is only added to the latitude and longitude. It is biased and shifts in size and an overview of the data can be seen in Figure 3.7. The biased measurements affect the translation estimate shown in Figure 4.8. For the baseline, the longitude error is shifted negatively and the lateral error is shifted positively. As there is no added disturbance to the altitude, the error becomes small during the disturbance and outage region.

When using DCS and RRR the longitudinal and lateral errors are similar and smaller compared to the baseline. Longitudinally DCS converges into a slightly better solution with a smaller SRMSE and laterally RRR converges into a slightly better solution, as seen partly in Table 4.5.

The altitude error becomes larger for both robust methods which is a result of measurements containing correct altitude information being removed or scaled because the measurements introduce large errors to the other states. Instead, lidar odometry is used as the estimate for the altitude. In relation to the ground truth 3σ the altitude error magnitude becomes smaller compared to the lateral and longitudinal errors. Because DCS scales measurements instead of removing them, some information about the altitude is kept and a better estimate is found.

Orientation errors can be seen in Figure 4.9 and all methods are similar in roll and pitch. In close proximity to the roundabout, there are some spikes in the estimates which are thought to be the problems in estimating the odometry, similar to what was seen in Figure 4.7 for the Gaussian noise. The errors before the

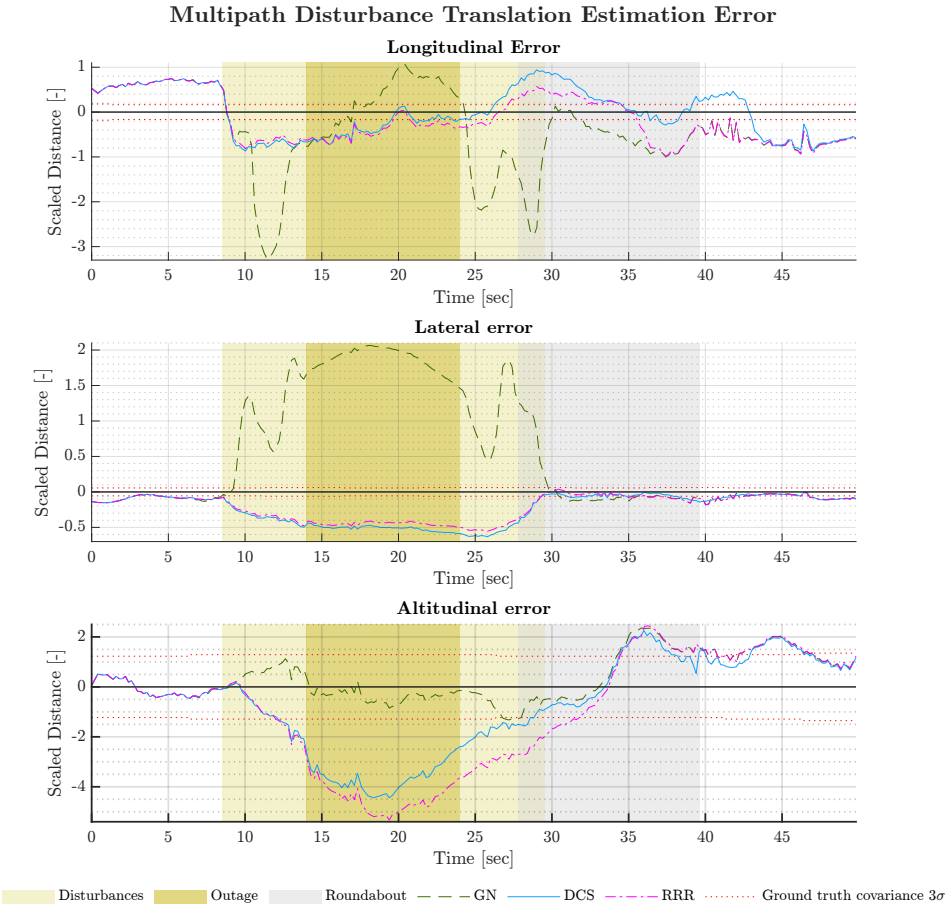


Figure 4.8: *Translation errors for the modeled multipath disturbance using the parameter selection $\Phi = 0.5$ for DCS and $p = 0.90$ for RRR.*

spikes show that both methods find a similar solution. The yaw error is smaller when using DCS or RRR compared to the baseline and the results are comparable even though DCS is slightly better.

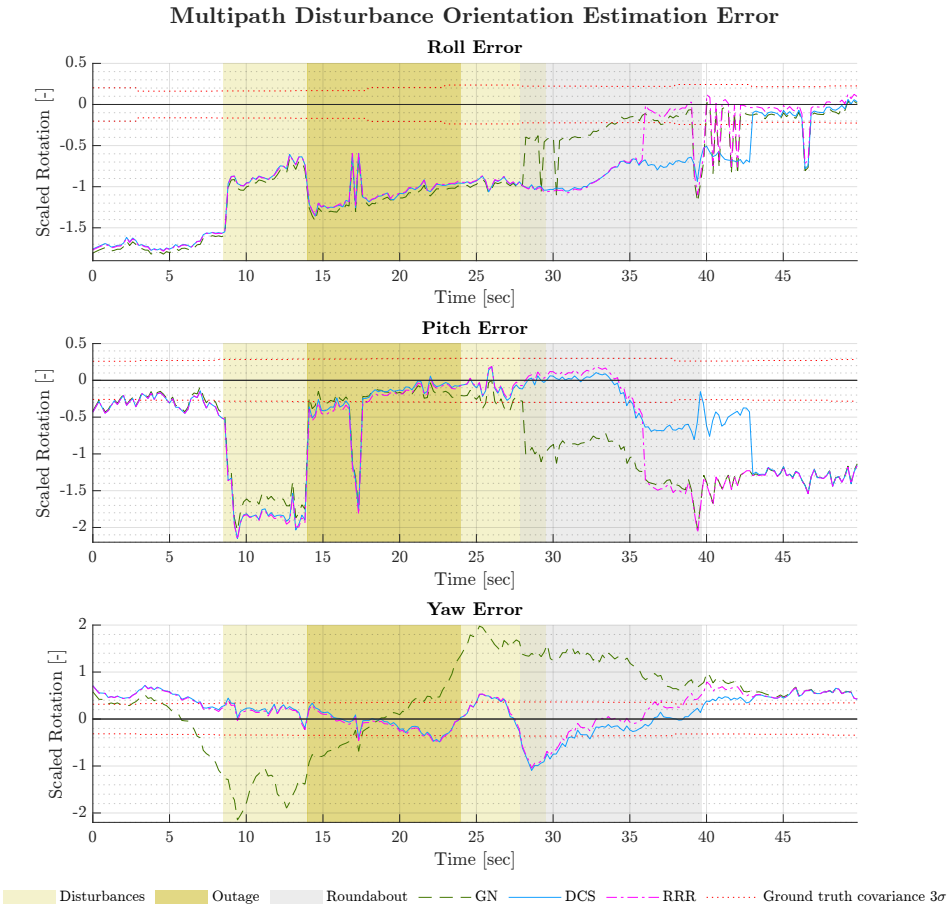


Figure 4.9: *Orientation errors for the modeled multipath disturbance using the parameter selection* $\Phi = 0.5$ *for* DCS *and* $p = 0.90$ *for* RRR.

In Table 4.5 the SRMSE for all six states from the multipath tunnel scenario are presented. The results show an improvement in all states except for altitude and roll when using robust methods. One possible cause for why the altitude SRMSE is worse when using DCS or RRR is GNSS measurements with correct altitude information being removed. The lidar odometry shows the worst performance in the altitude SRMSE which is used instead of GNSS measurements when they are removed.

Table 4.5: *Modeled Multipath disturbance with GNSS outage SRMSE for pose. The estimated state is shown in the first column. The second column shows the SRMSE for the baseline using only Gauss-Newton to estimate the best state configuration. The third column shows SRMSE when using only lidar odometry. In the fourth column, SRMSE is shown for state estimation using DCS with $\Phi = 0.5$. The last column presents the RMSE for state estimation using RRR with a p-value of 0.90.*

	Gauss-Newton	Lidar Odometry	DCS	RRR
Longitudinal	1	4.56	0.537	0.551
Lateral	1	1.19	0.313	0.279
Altitudinal	1	63.53	1.977	2.472
Roll	1	0.59	1.041	1.019
Pitch	1	3.09	0.835	0.981
Yaw	1	0.53	0.409	0.423

5
Discussion

5.1 Results

In this section the results from the experiments in Chapter 4 are discussed. The discussed subjects are parameter sensitivity, localization, ground truth, and the used method.

5.1.1 Parameter Sensitivity

The experiments show that the RRR solution varies less when the p-value is changed compared to DCS. A similar SRMSE can be seen in Table 4.2 for a p set between 0.80 and 0.95 in both modeled scenarios, that is the tunnel multipath and tunnel Gaussian scenarios. This is likely due to the size of the noise, meaning that RRR identifies similar measurements as outliers regardless of the p-value. We see that the SRMSE is worsened when $p = 0.80$ compared to other values in the first scenario where only small disturbances are present. This shows that setting $p = 0.80$ is not appropriate since it might worsen results during segments where no or only small disturbances are present compared to the baseline. The reason is that measurements are more likely to be considered outliers and removed when p is set to a lower value. Measurements that slightly deviate in the multipath scenario are removed as a measure to improve results but in this scenario, more information is lost than gained in the final solution. DCS does not see the same characteristics since the weight of each measurement is scaled based on the error size of the measurements instead of removing it completely even when Φ is set to 0.1, see Table 4.1. When Φ is varied, state estimation is mostly improved or similar for all values of Φ when compared to the baseline, with exceptions in state z for the tunnel multipath scenario and RP in tunnel Gaussian scenario.

The resulting SRMSE for different states in the experiments investigating the impact of parameter selection Φ in DCS and p-value in RRR can be seen Table 4.1 and 4.2. Similar performance to the baseline is sought when small or no disturbances are present, which is the case in the first scenario. Regarding improved state estimation of x, y, and yaw, the p-value should be set between 0.85 and 0.99 to not remove too many measurements in a situation with minor measurement errors. When setting $p = 0.80$ state estimation is less accurate compared to the baseline in the first scenario with multipath which can be seen in Table 4.2. Based on these results, RRR risks affecting the results negatively if not tuned correctly. When the p-value is set to a low value the RRR algorithm removes measurements aggressively meaning measurements with a small deviation are completely removed. This is not the case for DCS for the values tested. However, DCS shows a greater difference in the resulting SRMSE when varying Φ which shows in both modeled scenarios.

In the modeled tunnel multipath scenario, the SRMSE for state z is greater as both DCS and RRR are tuned for stricter scaling and removal of outlier measurements. When $p = 0.80$ the SRMSE for z is 2.58, and when $p = 0.99$ the SRMSE is 1.86. Since disturbances in the modeled multipath scenario are in the xy directions this is expected since removing accurate measurements of the z state will lower performance compared to baseline. However, as the p-value is set lower and more measurements are rejected the xy state estimation is only improved from 0.51 to 0.41 SRMSE. The Yaw estimation goes from 0.45 to 0.46 SRMSE in the modeled multipath scenario with p-value set to 0.99 and 0.80 respectively. In the modeled multipath scenario, the performance of RRR is very similar with a p-value set between 0.80 and 0.95. This is due to all disturbed measurements being identified and removed and setting p to a value lower than 0.95 does not lead to more values being removed. In the same scenario, if p is set to 0.99 only some of the measurements are removed and therefore a different result is seen. This shows that RRR is less sensitive to tuning when larger errors are present. The same can be seen in the tunnel Gaussian scenario in which similar performance can be seen across all values of p.

DCS shows a different tendency, improved performance is seen for lower values of Φ which can be seen in Table 4.1. States xy and yaw has the lowest SRMSE when Φ is set to 0.1 and 0.5. From this information, it can be concluded that RRR is less dependent on tuning in the case of larger errors but with smaller errors RRR might reduce performance when the p-value is set to a low value such as 0.80. DCS is more dependent on tuning in order to maximize performance, but a wide range of the tuning parameter Φ can be chosen, and state estimation of xy and yaw is improved for the tested scenarios. When Φ is set in the upper range between 2.5 and 10 the performance was improved or similar to the baseline for all states when disturbances are present.

It is also worth noting that RRR has at least one other tuning parameter. For example, one parameter is the time segment deciding the size of each cluster. It is

not likely that the cluster size would affect our results since there are no natural clusters in our scenarios that greatly benefit from the clustering component of RRR. It would also be possible to set p to a different value when testing for cluster consistency and individual values. These values are not studied in this thesis but adds complexity to the tuning of RRR.

5.1.2 Localization

The localization performance of the first multipath scenario was improved mainly in the altitude estimate during the first 13 seconds when using DCS. There is no significant difference when using RRR in this scenario compared to the baseline which can be seen in Table 4.3 where the SRMSE is close to one for all states. This is also seen in Figure 4.3 and Figure 4.4 where the errors in translation and orientation for RRR and the baseline are very similar. The most interesting state to discuss in more detail is the altitude where the disturbance was present. The observed noise and estimated covariance from the GNSS receiver in the data set can be seen in Figure 4.2 which shows that the estimated covariance from the receiver increases where the disturbance is found. During the first 13 seconds, it can be seen that the covariance is slightly underestimated. This makes it difficult for the χ^2 tests in RRR to trigger because the error depends on the covariance matrix. On the other hand, DCS has the advantage here being able to make smaller adjustments to the covariance which results in a smaller error during the first 13 seconds in the altitude state.

During both modeled scenarios which contain outages RRR successfully removes noisy measurements and DCS scales the covariance of these measurements. This allows for a smooth transition to using lidar odometry measurements for localization during the outage which improves performance, see Figure 4.6 and Figure 4.8 where the large spikes introduced by the disturbances are removed. In the modeled scenarios DCS performs slightly better than RRR since more information is kept as the covariance is scaled instead of removing the measurements entirely. In the modeled scenario with Gaussian disturbance, it can be seen in Table 4.4 that DCS has a better SRMSE for all estimated states except for Yaw. Compare this to the modeled multipath scenario where Table 4.5 shows that the SRMSE is very similar for both methods except for the altitude state where DCS performs better. One thing both modeled scenarios have in common is that the disturbance is added to the same measurements which makes the disturbance characteristic the differentiating factor. There seems to be no clear reason as to why the methods perform differently with the two disturbance types but one explanation could be that for the Gaussian scenario DCS gets an advantage due to the assumption of Gaussian noise made in the GNSS constraints. The scaling works better in the tunnel scenario with Gaussian noise since disturbances are present in all translational states. This means there is no trade off such as in the tunnel multipath scenario in which the altitudinal state is worsened while longitudinal and lateral states improved.

The cluster removal step of the RRR algorithm is interesting to discuss for use together with GNSS position constraints. The algorithm was originally designed for loop closure constraints which have the characteristics of appearing in smaller clusters. Loop closures are detected using some place recognition algorithm and are found in an area previously visited. The cluster removal step is a part of the algorithm to prevent faulty association between visually similar places such as road crossings which would make a generated map topologically incorrect. The same problem does not apply to GNSS measurements but some similarities can be seen if the GNSS disturbances are of the multipath characteristic of a shift in position. An example of a cluster being removed can be seen in the first scenario with small multipath disturbances. RRR manages to find a similar solution for all p-values except for $p = 0.80$ where all GNSS measurements in the first cluster are removed leading to worse localization. Similarly, in the two scenarios where an outage is combined with disturbances, the RRR removed entire clusters where the majority of measurements had disturbances which leads to correct measurements also being removed. The removal of large clusters can be seen as both positive and negative. The positive aspect is that we can be more confident that no outliers are used, together with a good odometry estimation the localization should keep a consistent quality. On the other hand, the cluster removal step removes correct measurements that could make the performance even better in applications where high precision in localization is needed. For DCS which does not cluster the measurements, it can be seen that the localization errors in both Section 4.4 and Section 4.5 are similar to the errors for RRR. However if the altitude error in Figure 4.6 and Figure 4.8 is looked at closer it can be seen that DCS error moves towards the ground truth covariance faster compared to RRR during the disturbed section between 24 seconds and 29 seconds. This could be explained by the removal of the entire cluster surrounding the disturbances which also contains correct measurements. This could indicate that cluster removal is not needed together with GNSS measurements.

As mentioned in Chapter 1 there have been comparative studies conducted earlier that focus on the performance of different robust methods combined with loop closure constraints. One important aspect brought up by the authors in [8] is whether the robust methods make binary decisions or not. The authors bring up the use of the output from the SLAM solution to higher level tasks such as navigation and planning. In the context of invalid loop closure constraints, they argue that non-binary methods do not completely remove inconsistent paths in the graph which could lead to failures when doing navigation tasks. The reason is that even if DCS correctly scales a faulty loop closure constraint the two nodes associated would still be connected in the graph which in turn can be used as a path in planning. This same problem does not apply if the same graph is used for localization. This is because a localization algorithm would only try to find which node in the graph is most similar to the current position based on a place recognition algorithm, the relationship between nodes in a graph is not relevant. Additionally for use together with GNSS constraints using our methodology the same problem does not occur as the GNSS constraint independently tells the rela-

tionship of a node to a global position and not the relationship to another node in the graph. Therefore robust methods for GNSS constraints that are non-binary would not introduce the same safety concern for a navigation task. The experiments conducted in this thesis show that both a binary method such as RRR and a non-binary method such as DCS are capable of identifying and reducing the effect of outlier GNSS measurements. In terms of localization, DCS performed better across the tested cases.

5.1.3 Ground Truth

The data used as ground truth is based on estimations from a high end INS. The INS also estimates the certainty of estimated states in the shape of a covariance matrix. If our estimate is within the estimated covariance of the ground truth less can be said about the performance since the resolution of the ground truth is limited by this factor. It could also be the case that the INS provides faulty estimations, but this is deemed as having a low impact on the results.

5.1.4 Lidar Odometry

The quality of the lidar odometry estimate greatly affects the localization performance when using robust methods. The lidar odometry estimate method used in this thesis shows signs of overestimating the distance traveled, likely due to the corridor-like shape of a road and the relatively larger longitudinal velocity compared to lateral or altitudinal velocity. This can be seen in the results of the tunnel scenarios in Table 4.3 and 4.4 where the lidar odometry longitudinal SRMSE is less than the lateral SRMSE. When the robust methods are applied, a smaller improvement is also seen in the longitudinal direction compared to the lateral direction which is caused by the quality of the lidar odometry estimate. However, since the presented data is the SRMSE compared to the baseline, the effect of the robust methods is isolated.

One of the important assumptions of the lidar SLAM problem is that the world is static. The dataset used in this thesis was recorded during the daytime with a moderate traffic flow, which introduces dynamic objects to the lidar scans. The dynamic objects make up a small part of the point clouds making the effect on localization small. The GNSS measurements are independent of the time of day.

5.1.5 Map quality

The map quality is closely tied to the performance of localization. Although this thesis does not specifically look into the map quality, during the testing phase it was clear that the map quality was affected by the localization performance. This is due to the system design, where the map points are generated based on the localization estimate. If the localization estimate was poor the map showed several imperfections such as a blurry map, curved buildings, and doublets or triplets of features such as lamp-posts.

5.2 Method

The robust methods chosen for evaluation in this thesis can be categorized into two types of robust methods used in the back end of graph SLAM. The categorization is based on what kind of decision is made once an outlier measurement is found. RRR makes a binary decision and completely removes the measurement while DCS makes a non-binary decision by applying scaling to the measurement. Based on previous research [7, 8] this trait had a great impact on results when the robust methods were applied to loop closure constraints when the final graph SLAM solution is to be used for navigation. Overall, no method was definitively superior to the others, and the data set affected which robust method performed the best. Only two methods were investigated in this thesis due to time constraints, but RRR and DCS were chosen as popular and capable methods based on previous studies that represent both the binary and non-binary methods.

Earlier studies on robustness in graph SLAM with GNSS focus on applying methods with pseudorange observations. This approach is interesting for research purposes but for use in industrial applications, simpler solutions might be opted for instead for efficiency and cost saving. The method used for integrating GNSS in this thesis uses the position estimated by the GNSS receiver as constraints instead of each individual pseduorange observation. The position measurement is also available when using cheaper equipment which makes our research relevant for systems using the same measurements. Watson and Gross [6] conducted a comparative study of robust methods with pseduorange measurements and found that the Switchable Constraints method outperformed Dynamic Covariance Scaling but no further discussion explaining why. One possible explanation could be the characteristics of multipath disturbances where the pseduorange measurement is reflected off surrounding structures. Either it is reflected off a structure or it has not been reflected off a surrounding structure, there is not so much an in-between. This could be an advantage to a binary method such as switchable constraints as it matches the characteristics of the disturbance. In the case of our formulation of the GNSS measurements, the same multipath disturbance would be incorporated into the position measurement, resulting in a slightly disturbed position measurement. This means that a mixture of both good and bad measurements are fused into one position, that remains moderately accurate. Following the logic of selecting an appropriate method depending on the noise characteristics, one could argue that a non-binary method would be suitable for this type of formulation.

Modeling of GNSS noise characteristics in the tunnel scenario was mostly done based on experiences from the partner company. It is hard to evaluate how accurate the modeling is because it was hard to find academic papers that try to model GNSS disturbances when driving inside tunnels. Probably because it is difficult to get an accurate ground truth inside tunnels. Using lidar for odometry estimation with scan matching is challenging due to the low number of features found, tunnels being very monotonous with mostly flat walls, ground, and ceiling. And

for obvious reasons GNSS cannot be used as the ground truth. In order to cover a wider range of scenarios two different types of disturbance were modeled. The first modeled scenario makes the assumption of Gaussian noise. The idea was that the uncertainty grows proportionally to the distance into a tunnel before cutting off completely. This model of a tunnel does not represent every tunnel that exists but roughly represents the problem when driving through tunnels or in urban canyons. This was established in discussions with the partner company. The shape of the noise might be of a different distribution in a real scenario. For example, it might be a Gaussian noise with an offset. Still, the robust methods will have to distinguish outliers at the beginning of the tunnel in order for odometry to correctly estimate position. The second modeled scenario tries to resemble the real disturbances that were observed in the left GNSS receiver in our data set. Even if the modeled disturbances do not resemble GNSS disturbances perfectly conclusions about the performance of robust methods can still be drawn from the experiments that test the robust methods on noise commonly found using GNSS receivers.

Covariance estimation also makes a large difference when trying to find the correct solution. In both modeled scenarios, a decision was made to not update the covariance of the measurement to see how the robust methods would handle the problem. The idea is to emulate the worst case scenario where the GNSS receiver has a hard time estimating the covariance. In a real scenario, this could be the case in a real time application where the covariance estimation is updated less frequently due to limitations in computation speed or a limitation on data transfer capacity between the GNSS receiver and the localization system. Updating the covariance estimate proportional to the size of added disturbance would most likely result in the Gauss-Newton solution improving, and decreasing the impact of the robust methods. The idea was that by not giving the robust methods the best conditions their performance would be tested. Together with the first scenario where the covariance is updated from the receiver, the experiments cover both cases.

6

Conclusions and further work

A summary of the results, analysis, and discussion are presented in this chapter. Proposals for further work are also presented. Each research question is answered in the first section.

6.1 Conclusions

This thesis explored the effect of robust methods for GNSS measurements in a system with lidar for localization. Two methods, DCS and RRR were used as representatives for the non-binary and binary categories of robust methods. The study found that DCS, with its non-binary decision-making approach, aligns better with the GNSS constraint formulation used compared to the binary approach of RRR. However, both methods are capable of reducing the effect of GNSS outliers and improving localization.

Combination of GNSS and lidar

In situations where GNSS fails to provide accurate localization lidar odometry can be relied on as an alternative localization method if coupled with a robust method for the GNSS measurements. Regarding the first research question, graph SLAM can be used with GNSS measurements as position constraints alongside lidar data to estimate the odometry. It is also possible to incorporate the measurements from a dual GNSS receiver as two independent constraints. To reduce the effects of outlier GNSS measurements and GNSS outage on localization, robust methods such as DCS and RRR can be used. Both methods were able to reduce the effects of GNSS outage, increasing Gaussian noise, and multipathing.

DCS and RRR rely on having consistent odometry estimation to identify outlier GNSS measurements. For a system where the lidar odometry is of poor quality, or in environments where scan matching is challenging, the constraint formulation used in this thesis is not recommended.

Localization with GNSS disturbances

In the case of disturbances combined with a GNSS outage, such as in the tunnel scenarios, RRR and DCS successfully identify the erroneous measurements as well as reduce the effect and improve localization. DCS outperforms RRR reducing the effects of disturbance and combined GNSS outage by scaling the information matrix by the size of the errors. DCS also improves state estimation in the first multipath scenario with smaller disturbances. The cluster removal step of RRR is not suitable together with GNSS measurements as it removes too many measurements that still carry information. Although the measurements are inaccurate, the complete removal of clusters is not justified when accurate localization is the objective in a lidar and GNSS system.

The use of a robust method for handling GNSS outliers can be a good idea for some applications where there are high requirements for localization and where disturbances are expected to occur. Provided there is a good and reliable odometry estimate that can be used in conjunction with GNSS measurements. In one way, the robust methods support the covariance estimation done by the GNSS sensors. The robust methods might not improve the localization performance if the covariance estimation is sophisticated enough. However, an extra layer of robustness is useful in both real time applications such as navigation, as well as offline such as mapping.

Binary and non-binary decision-making

When concluding which robust technique is most suitable for GNSS constraints the formulation of the constraints is an important factor. Together with the formulation used in this thesis, it is more appropriate to make a non-binary decision compared to a binary decision to deal with outliers. DCS makes a non-binary decision that better aligns with the formulation used.

The impact of the p-value in the RRR algorithm suggests RRR is less suitable as a robust method to handle GNSS disturbances and outages compared to DCS. If the value of p is set too low it can worsen the performance of state estimation, particularly in scenarios when small disturbances are present which could be seen in the altitude error of the multipath scenario. DCS consistently manages to improve state estimation for x, y, and yaw states for different tuning. However, RRR shows less variance in performance for different values of p compared to varying Φ in DCS. This could be an advantage when applying RRR in certain systems. The usefulness of a robust method greatly depends on the required tuning. If additional parameters are required, the benefits from improved covariance estimation are offset by the increased complexity introduced to the system.

Both DCS and RRR are viable methods for handling outliers in GNSS measurements. However, considering the overall performance, tuning properties, and the ability to make non-binary decisions, the conclusion is that DCS is the most suitable method.

6.2 Further Work

In this thesis, two different methods for increasing robustness against outliers in graph SLAM were evaluated. However, the scenarios assessed provide limited knowledge of how different noise characteristics affect the evaluated methods. In order to further support the extent of these findings, a more quantitative study could be performed which simulated multiple runs with slight noise variations of the presented scenarios. If more real-world data were tested, a better picture of the impact of RRR and DCS could be gained. Samples from a real tunnel would provide a reference to the artificial scenarios in this thesis.

A more refined version of RRR where removing measurements is punished in order to keep as much information as possible has been published and would be interesting to compare.

An interesting extension of DCS would be to scale the covariance of states independently of each other. This would be beneficial for multipath disturbances for example a bias in the z direction should not affect the confidence of lateral and longitudinal information. If the scaling factor s is calculated as a vector where each individual error state is used. This would also add the possibility to tune Φ for each state.

The robust methods DCS and RRR can be applied to different constraints set in graph SLAM in order to improve the performance of localization. For example, a case for this would be the application to lidar odometry constraints. A topic of further work could be in the challenging scenario where both GNSS and lidar data can exhibit outliers. How should a decision be made to correctly identify which source of measurements is true in a graph SLAM context?

Bibliography

[1] Sebastian Thrun and John J. Leonard. *Simultaneous Localization and Mapping*, pages 871–889. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-30301-5. doi: 10.1007/978-3-540-30301-5_38. URL https://doi.org/10.1007/978-3-540-30301-5_38.

[2] Niko Sünderhauf and Peter Protzel. Switchable constraints for robust pose graph SLAM. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1879–1884, 2012. doi: 10.1109/IROS.2012.6385590.

[3] Pratik Agarwal, Gian Diego Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard. Robust map optimization using dynamic covariance scaling. In *2013 IEEE International Conference on Robotics and Automation*, pages 62–69, 2013. doi: 10.1109/ICRA.2013.6630557.

[4] Yasir Latif, Cesar Cadena, and Jose Neira. Robust Loop Closing Over Time. 2012. doi: 10.15607/RSS.2012.VIII.030.

[5] Niko Sünderhauf, Marcus Obst, Gerd Wanielik, and Peter Protzel. Multipath mitigation in GNSS-based localization using robust optimization. In *2012 IEEE Intelligent Vehicles Symposium*, pages 784–789, 2012. doi: 10.1109/IVS.2012.6232299.

[6] Ryan M Watson and Jason N Gross. Robust navigation in GNSS degraded environment using graph optimization. In *Proceedings of the 30th international technical meeting of the satellite division of the institute of navigation (ION GNSS + 2017)*, pages 2906–2918, 2017.

[7] Niko Sünderhauf and Peter Protzel. Switchable constraints vs. max-mixture models vs. RRR - A comparison of three approaches to robust pose graph SLAM. In *2013 IEEE International Conference on Robotics and Automation*, pages 5198–5203, 2013. doi: 10.1109/ICRA.2013.6631320.

[8] Yasir Latif, César Cadena, and José Neira. Robust graph SLAM back-ends: A comparative analysis. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2683–2690, 2014. doi: 10.1109/IROS.2014.6942929.

[9] Josh Bongard, Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic Robotics. *Artificial Life*, 14(2):227–229, 2008. doi: 10.1162/artl.2008.14.2.227.

[10] Sebastian Thrun and Michael Montemerlo. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006. doi: 10.1177/0278364906065387. URL https://doi.org/10.1177/0278364906065387.

[11] H.-A. Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, 2004. doi: 10.1109/MSP.2004.1267047.

[12] Frank Dellaert and Michael Kaess. Factor Graphs for Robot Perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017. ISSN 1935-8253. doi: 10.1561/2300000043. URL http://dx.doi.org/10.1561/2300000043.

[13] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001. doi: 10.1109/18.910572.

[14] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A Tutorial on Graph-Based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010. doi: 10.1109/MITS.2010.939925.

[15] José Luis Blanco-Claraco. A tutorial on **SE**(3) transformation parameterizations and on-manifold optimization. 2022. doi: https://doi.org/10.48550/arXiv.2103.15980.

[16] Michael L. Johnson and Susan G. Frasier. [16] nonlinear least-squares analysis. In *Enzyme Structure Part J*, volume 117 of *Methods in Enzymology*, pages 301–342. Academic Press, 1985. doi: https://doi.org/10.1016/S0076-6879(85)17018-7. URL https://www.sciencedirect.com/science/article/pii/S0076687985170187.

[17] G. Agamennoni, P. Furgale, and R. Siegwart. Self-tuning M-estimators. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, 2015. doi: 10.1109/ICRA.2015.7139840.

[18] Pratik Agarwal, Giorgio Grisetti, Gian Diego Tipaldi, Luciano Spinello, Wolfram Burgard, and Cyrill Stachniss. Experimental analysis of dynamic covariance scaling for robust map optimization under bad initial estimates. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3626–3631, 2014. doi: 10.1109/ICRA.2014.6907383.

[19] Ricardo Roriz, Jorge Cabral, and Tiago Gomes. Automotive LiDAR Technology: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 23 (7):6282–6297, 2022. doi: 10.1109/TITS.2021.3086804.

[20] Ji Zhang and Sanjiv Singh. LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, pages 1–9. Berkeley, CA, 2014.

[21] 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.

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

[23] Peter Biber and Wolfgang Straßer. The Normal Distributions Transform: A New Approach to Laser Scan Matching. volume 3, pages 2743 – 2748, 2003. ISBN 0-7803-7860-1. doi: 10.1109/IROS.2003.1249285.

[24] Elliott D. Kaplan and Christopher J. Hegarty. *Understanding GPS/GNSS: Principles and Applications, Third Edition.* Artech House, 2017. ISBN 9781630810580.

[25] Kenji Koide, Jun Miura, and Emanuele Menegatti. A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement. *International Journal of Advanced Robotic Systems*, 16, 2019. doi: 10.1177/1729881419841532.

[26] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G^2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. doi: 10.1109/ICRA.2011.5979949.

[27] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, 2011. doi: 10.1109/ICRA.2011.5980567.

[28] Stanford Artificial Intelligence Laboratory et al. Robotic Operating System. URL `https://www.ros.org`.