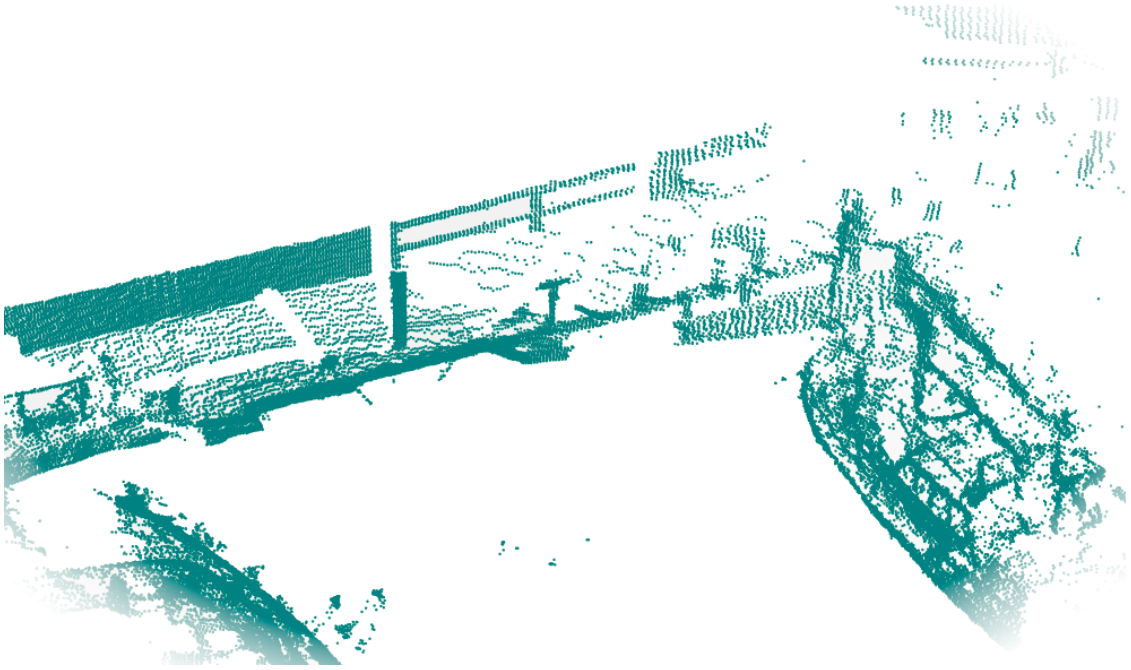

Lidar-based simultaneous localisation and mapping in marine vehicles

Handling the complex motions in a marine setting

Master's thesis in Complex Adaptive Systems

Artur Engström and Domenic Geiseler

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Lidar-based simultaneous localisation and mapping in marine vehicles

Handling the complex motions in a marine setting

ARTUR ENGSTRÖM
DOMENIC GEISELER

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2022

Lidar-based simultaneous localisation and mapping in marine vehicles
Handling the complex motions in a marine setting
ARTUR ENGSTRÖM, DOMENIC GEISELER

Supervisor: Ola Benderius, Department of Mechanics and Maritime Sciences
Examiner: Ola Benderius, Department of Mechanics and Maritime Sciences

Master's Thesis 2022:31
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Figure illustrates a point cloud from the lidar near the shore. The details depicts part of a jetty and two boats.

Typeset in LaTeX
Printed by Chalmers Digitaltryck
Gothenburg, Sweden 2022

Lidar-based simultaneous localisation and mapping in marine vehicles
Handling the complex motions in a marine setting
ARTUR ENGSTRÖM, DOMENIC GEISELER
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

To perform precise and safe manoeuvres in an autonomous vehicle, it is important to be able to extract its position in relation to its surroundings. Furthermore, an accurate depth representation of the environment using a lidar scanner helps to detect distances and possible dangers. Simultaneous localisation and mapping (SLAM) is a well-researched tool to aid autonomous cars in operation and data collection to solve both of these tasks at once. The less explored domain of SLAM on a marine vessel presents more complex kinematic movements and is vulnerable to drift in the vertical plane. This master thesis is part of the Reeds data set project, which aims to provide researchers with sensor data to support projects in marine settings. In this work, it was investigated how to reduce this vertical drift using hardware and software-based approaches and what the minimal conditions for performing SLAM are. The baseline performance was built on the lidar-only NDT and ICP registration algorithms. Data loggings were recorded using a high-performance lidar with additional sensor data from an inertial measurement unit and a global navigation satellite system. The latter sensors were fused in an Unscented Kalman filter to estimate the sensor position and thereby improve mapping. The hardware-based approach used a gimbal stabiliser to constrain movement to rotations on the yaw axis. In the analysis, it was shown that both approaches were able to significantly reduce deviation compared to the lidar-only approach. While the UKF state estimation performed better in situations with very sparse point clouds, the gimbal approach offered computational advantages due to the rotational restrictions. Both approaches can be combined to benefit from each other. The minimal conditions for SLAM were bound to the point-cloud density and velocity, as was shown by correlation. These findings can aid in future data recordings regarding the building of a sensor platform.

Keywords: SLAM, UKF, Gimbal, Mapping, Localisation, Marine systems, Marine datasets, Lidar.

Acknowledgements

We would like to express our sincerest gratitude to our supervisor and examiner, Ola Benderius, for his continued support throughout this project. Ola's enthusiasm and uplifting attitude have been invaluable for us in our work. His coaching during our weekly meetings has helped us maintain a high pace and he also provided great details regarding the peculiarity that goes into the academic process. Also, we thank Christian Berger for the help. He assisted us during the initial phase of this project by providing his expertise in creating the interface to the lidar. We express thanks to Krister Blanch and Ted Sjöblom for helping us during the recording and collection of our data sets. They provided help, both during the installation of the sensor set-up and in driving the vessel. We would like to thank the whole team at Reeds and Revere for letting us be a part of their work when collecting and working with data from their sensors. We are grateful for the weekly meetings; as they provided great insight into the research and challenges that one has to face when working within the maritime industry. We show great appreciation to Fredrik von Corswant for helping us create the mounting for the sensor set-up, it came to great use. We also thank Varun Ganapati Hegde and Liangyu Wang for the time they took to consult and present their previous work, which this project was based on. It accelerated our understanding of the SLAM algorithm and provided a great overview of the challenges ahead. Lastly, we want to thank Ebaa Asaad, Sara Larsson, and Heegyeong Kim for brainstorming ideas during our meetings. These discussions have both been fun and educative.

Artur Engström, Domenic Geiseler, Gothenburg, June 2022

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ADAS	Advanced Driving-Assistance Systems
AV	Autonomous Vehicle
CD	Continuous Deployment
CI	Continuous Integration
CTRV	Constant Turn Rate Velocity
CV	Constant Velocity
DOF	Degrees of Freedom
DNN	Deep Neural Network
EKF	Extended Kalman Filter
GNSS	Global Navigation Satellite System
ICP	Iterative-Closest-Point
IMU	Inertial Measurement Unit
KF	Kalman Filter
lidar	Light Detection and Ranging
NDT	Normal Distribution Transform
PDF	Probability Density Function
PF	Particle Filter
SLAM	Simultaneous Localisation and Mapping
UKF	Uncented Kalman Filter

Contents

List of Figures

List of Tables

1
Introduction

With a surge in hardware and software improvements in recent years, autonomous transportation has become a growing research field [1]. The demand for autonomous vehicles (AV) is expected to continue to expand with a compound annual growth rate of 63.1% between 2021 and 2030 [2]. Rapid expansion creates great opportunities for the AV sector. However, the doubts and challenges that remain to be overcome are still huge, as the implementation of an autonomous driving environment encompasses not only complex automotive technology, but also human behaviour, ethics, traffic management strategies, policies and liabilities [3]. An AV should be able to manoeuvre in various road conditions, as different surfaces could be highly unpredictable and vary for different environments. It should also be able to handle varying weather and traffic conditions, since the presence of, for example, rain or fog should not interfere with the decision-making of AVs.

It is of great interest to create an algorithm that enables an AV to navigate in an unknown environment while considering the naturalistic settings it might encounter. This is a problem often described as *simultaneous localisation and mapping* (SLAM), which is a technique used for AVs to build a map of an unknown environment while simultaneously localising the vehicle position. A successful implementation of the SLAM problem would allow the robot to perform tasks such as path planing and obstacle avoidance, which would be particularly useful when the AV would explore new environments without any prior knowledge. It has even been concluded that a solution to the SLAM problem could be seen as a *holy grail* for the mobile robotics community since it would provide the means to make a robot truly autonomous [4].

Although the SLAM problem has been considered a closed problem in some scenarios, it is still not possible to apply a single robust algorithm that fits all environments [5]. The main issue originates from the fact that the sensor readings are not completely accurate and have been shown to accumulate drift over time [6]. The drifts in localisation are most common in the vertical direction since sensor correction is more common in the horizontal plane due to the sensors' field of view, where the vertical field of view of most sensors is limited in comparison to the horizontal. These drifts of the trajectories are usually bounded when operating within an indoor environment due to correction of the walls, floor and roof, but could theoretically diverge in an outdoor environment [7].

Since most of the available literature presents research that has been focused on road vehicles [8][9], the objective of this paper is to investigate how the SLAM algorithm would perform in a marine environment. This is particularly interesting because a vessel in water will have substantially more complex movement compared

to a vehicle on the road, due to the increased complexity of the pitch, roll, and heave. The marine environment is also intriguing due to the different reflective characteristics of water in comparison to solid ground, since the water cannot give direct reflection when hit with a light beam. This is important since the *light detection and ranging* (lidar) sensors will not be able to detect the ground, which is a distinct difference from how a similar sensor would operate on land. This circumstance of not detecting the ground enables an unbounded vertical region. This leads to the investigation on how to reduce the drift in the vertical direction.

Moreover, software-based state estimation algorithms have been proposed to be used to improve localisation and mapping [10]. These solutions often incorporate a methodology named *sensor fusion*, which is the process of combining multiple external sensors to reduce the resulting uncertainty of the measurement. This combination has been shown to improve the resulting performance of SLAM algorithms, if one can obtain reliable sensor readings. However, this is not always the case in the marine environment. For example, a GPS signal is shown to fail to produce an accurate signal when the GPS antenna is within the vicinity of tall structures, or within indoor environments [11]. Erroneous sensor readings could, for example, occur if the vessel is traversing along a tall cruise ship or passing under a bridge. Another problem is that most sensor setups are stationary mounted on the AV, causing potential drifts in the SLAM algorithm due to offsets of the global frame. Since most algorithms compare two consecutive frames when performing SLAM, the first frame of the recording will usually be considered as the origin. This is a problem if the initial frame is not aligned with the global frame, which furthermore might cause inaccurate trajectory predictions. This creates a demand for producing solutions that can incorporate a simplistic implementation by replacing external sensors, which rely on a software-based fusion, with a hardware-based solution that can compensate for the complex kinematic movement of the vessel and disregard the dependence from the GPS.

Further concerns have previously been expressed about the limited datasets available for benchmarking and comparing the performance of robot perception algorithms [12], this is especially evident for the maritime industry. Therefore, it is of high interest to investigate the minimal conditions that would be needed to perform SLAM in a marine environment since this has not yet been done. It is also of interest to explore how the algorithm behaves over a longer time, as most data sets introduced in the past are relatively small and specialised [13, 14, 15].

To answer the proposed research questions, this master's thesis is in cooperation with the Reeds project [16], which is a collaboration between Chalmers University of Technology, University of Gothenburg, Research Institute of Sweden, and the Swedish Maritime Administration.

1.1 Research question

This project aims to investigate the following research questions:

RQ-1: How to reduce the deviation in the vertical plane when implementing SLAM on a vessel?
RQ-2: How does a software-based solution compare to a hardware-based solution for reducing the deviation in the vertical plan?
RQ-3: What are the minimum conditions for implementing SLAM in a marine environment?

1.2 Limitations

The aim of the project was to perform live processing and restricted its focus to lidar sensors, not to cameras or radar. Data from the *global navigation satellite system* (GNSS) and the *inertial measurement unit* (IMU) were used, however, the authors were not responsible for creating an interface for these sensors since this had already been done by the team at Reeds. Some SLAM implementations require a 9-axis IMU and multiple GNSS antennas [17][18], this project, however, only had access to a 6-axis IMU and a single antenna. Furthermore, SLAM was restricted to only use point clouds from a single lidar sensor.

1.3 Outline

Following this introduction, the remaining part of this report is organised as follows. Sect. 2 provides a brief background on existing data sets for AVs (2.1), continued by a description of common sensors that are used for control and perception algorithms (2.2) together with the description of a sensor stabilizer (2.3). The SLAM problem is then explained (2.4) before concluding with the description of containerised software (2.5). Sect. 3 presents the methodology used in this work, where the implemented SLAM algorithms are detailed (3.1), together with an overview of the software architecture (3.2). The experiments for the proposed research questions are then described (3.3), followed by how they were planned to be evaluated (3.4). Sect. 4 presents the results regarding the vessel experiment (4.1), gimbal experiment (4.2) and SLAM requirements (4.3). Sect. 5 contains the discussion of the results followed by the conclusion in Sect. 6.

2
Background

In the first part of this section, there is a general introduction to the work related to collecting and using various data sets for AV, Sect. 2.1. Then, a description of the most common sensors and previous studies connected to control and perception algorithms are covered in Sect. 2.2. Furthermore, the description of localisation and mapping algorithms are explained in Sect. 2.4. This section concludes with a discussion of the practical implementation of containerised software in Sect. 2.5.

2.1 Data sets for autonomous vehicles

When designing control and perception algorithms, one would prefer to directly use existing data sets collected by other researchers, since data acquisition can be quite time-consuming and cumbersome. Not only is this a matter of convenience, it also helps researchers to use various data sets as benchmarks to evaluate and compare the performance of different algorithms.

2.1.1 Explicit and implicit factors

The development of robust methods and novel metrics depends, in part, on having access to large-scale naturalistic and diverse driving data sets. It is of high importance that the collected data be as natural and diverse as possible in order to fairly represent realistic driving scenes. Naturalistic settings that one needs to consider to improve the driveability of an AV can often be categorised into either *explicit/environmental* factors or *implicit/behavioural* factors [8]. Explicit factors that would affect the AV could, for example, be due to weather, illumination, road geometry, road condition, lane marking, traffic conditions, etc. Implicit factors include vehicle behaviour, pedestrian behaviour, driver behaviour, motorcyclist/bicyclist behaviour, etc. Handling each factor presents its own research challenge, and according to previous research, existing approaches are still far from delivering robust solutions that consider all factors in their data sets [8].

2.1.2 Overview

Previous literature has presented an overview of the available data sets that have been collected on public roads. Two papers have presented a detailed overview of 54 and 27 data sets, respectively. These data sets are supposed to represent the most popular ones, used by researchers when developing control and perception algorithms for AV [8][9]. All data sets are collected from platforms that have been mounted on *road* vehicles. It can be observed that most of these data sets were collected for object detection, object tracking, and semantic segmentation, which are some of the early research fields for autonomous driving. It is also concluded that only a few data sets can be used to analyse implicit factors, which is a research topic that has been studied insufficiently [8].

There are three data sets that clearly outrank the remaining data sets in popularity [9]. These are *Caltech (2009)* [14], *KITTI (2012)* [19], and *Daimler pedestrian (2006-2016)* [20]. While Caltech and Daimler focused on urban traffic, specifically pedestrian detection, KITTI is used in a broader perspective. Specifically, targeting the development of perception algorithms for autonomous road vehicles. KITTI also provides canonical benchmarks for testing and comparison of baseline algorithms that can help the development of tasks like optical flow, visual odometry/ SLAM and 3D object detection. It serves as an open platform on which other researchers can upload evaluation results of their own algorithms to compare their results with others of similar types.

Figure 2.1: Example images taken from the *Caltech data set* (left), *KITTI data set* (middle) and *Daimler data set* (right).

To aid in further research on the topic, new data sets are needed. Since many of the data sets were collected more than a decade ago, there has been concern about outdated hardware [21]. Additionally, most of the available data sets only pertain to road vehicles. There has been interest in expanding data sets for govering maritime vehicles, since this is a research field that only has a few publicly available data sets. However, availability is not the only issue. The data sets within the maritime domain are often not diverse enough, on-board recordings are rare, and are not created with ship autonomy in mind [22, 23, 24]. It is therefore of great interest to investigate further on how control and perception algorithms would perform in these types of settings.

2.2 Sensors

The perception of the environment is one of the major challenges of AV. When designing control and perception algorithms, one has to decide on what type of data is to be used to provide solutions to the desired task.

The following sections present a summary of the most common sensors that are used for AV when gathering data. It is followed by a brief summary of the different benefits and challenges that one might encounter when using each sensor. A detailed description of the control and perception algorithms that have been focussed on in previous research is provided for each sensor.

2.2.1 Camera

Cameras are starting to become the most used device, as they can provide a visual sense of the environment in the same way humans do with vision. The increase in popularity comes from the development of neural networks and its ability to handle multiple camera inputs [25].

Cameras, in comparison to lidar systems, have the benefit of being cheaper and are less affected by weather conditions such as fog, rain, and snow. Here, the latter can be both good and bad, depending on the situation. The limitation of cameras originates from the fact that they only provide raw image data, which means that they usually lack the depth and exact location of different objects. However, researchers have investigated the possibility of estimating a distance using only camera sensors [26].

For most of the available data sets, there is a balanced use of monocular and stereo cameras, where the primarily focus has been on object and road detection [8]. There is also some research that has focused on 3D reconstruction and vision-based SLAM [27][28].

2.2.2 Light detection and ranging

Historically, most AV companies have relied heavily on *light detection and ranging* (lidar) sensors as their primary perception sensor [29]. lidar sensors operate by emitting pulsed light waves into the surrounding environment. By measuring the time it takes for each pulse to return to the sensor after colliding with an object, the sensor obtains a distance. By repeating this process millions of times per second, the lidar is able to create a precise 3D map of its environment.

One of the primary advantages of lidar is accuracy and precision; in contrast to the beams of, for example, radars and sonars, a laser beam can be made highly focused, without sidelobes. [29]. Compared to cameras, lidar is not affected by shadows, bright sunlight, or other illumination. However, since lidar uses visible lasers to measure distance, it cannot work well in bad weather conditions, such as heavy rain, snow, and fog [30]. The expensive cost of the lidar system is also an issue.

In combination with the camera, two major approaches are pursued. These are extractions of more weather-invariant vision features [31][32], and sensor fusion using

both camera and lidar data [33]. Among the available data sets for AV, Velodyne 64 and Sick were the most popular lidar models [9].

Figure 2.2: Velodyne HDL-64E lidar (left) and Sick MRS6000 lidar (right).

2.2.3 Global navigational satellite system

A *Global navigational satellite system* (GNSS) uses satellites to provide autonomous geo spatial positioning. By using time signals that are transmitted by radio, GNSS is able to determine the longitudinal and latitudinal position with high precision. To provide continuous global positioning capability, a constellation of at least four satellites must be simultaneously electronically visible at all times [34]. This means that areas with limited satellite signals, such as caves and tunnels, will interfere with the GNSS from producing valid output.

It has also been shown that in dense urban environments, there are propagation variations of the resulting GNSS signal [11]. This results in multipath reception errors and degrades positioning accuracy, sometimes causing error messages [21]. Multipath errors are generated when a signal arrives by different routes, caused by the antennas' proximity of nearby structures that reflects the signal; see Fig 2.3 for an illustration of the multipath reception error.

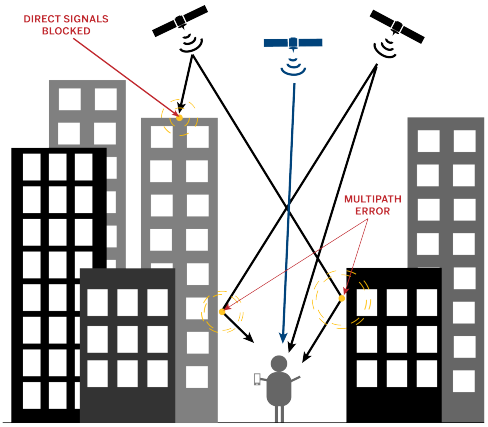

Figure 2.3: Illustration of the multi-path reception error.

GNSS can also be used to verify and aid various state estimation algorithms [21][35]. These estimation algorithms are often needed in cases where the GNSS either loses signal or gives an erroneous output. To provide valid navigation, the GNSS is often complemented by statistical methods and recursive algorithms to handle vehicle localisation problems. Some examples of these state estimation algorithms are Kalman Filter (KF), Extended Kalman Filter (EKF), Uncented Kalman Filter (UKF) and Particle Filter (PF), see Sect 2.4.2.

2.2.4 Intertial measurement unit

An *inertial measurement unit* (IMU) is an electronic device that is capable of measuring the specific angular rate, orientation, and acceleration of a body by using gyroscopes, accelerometers, and sometimes magnetometers. IMUs are generally classified according to the amount of sensors that they have built-in. Units consisting of a gyroscope and an accelerometer are called 6-*degrees-of-freedom* (DOF) IMUs. Although the use of the accelerometer allows tracking of relative rotatinal changes, it is not capable of measuring the absolute rotational position in pitch, roll, and yaw. This is only possible by adding a magnetometer. These advanced devices are known as 9-DOF IMUs.

The gyroscope operates by producing a high-update rate attitude solution by integrating the angular rate measurements. The magnetometer is similar to a magnetic compass, thus providing an orientation reference. The accelerometer measures the linear acceleration in each direction, according to the Cartesian coordinate system. However, since the accelerometer also records the gravitational acceleration, it has to be subtracted by the gravitational acceleration to obtain the linear acceleration of the system due to motion in all directions [36]. The subtraction can be made by using estimates of the system's attitude.

It is possible to use an IMU for navigation. The position of the system can be obtained by integrating the velocity, where the velocity in each direction can be obtained by integrating the acceleration measurement. The disadvantage of using

IMUs for navigation is due to the accumulated errors that occur over time when integrating the acceleration. This means that any measurement error, which is constant, will cause a linear error growth in velocity and a quadratic error growth in position. It has also been illustrated that a constant error in the gyroscope attitude rate will result in a growth of both quadratic and cubic error for velocity and position, respectively [37].

2.3 Gimbal

A hardware system used to stabilise sensors (in particular cameras) is known as a *gimbal*, see Fig. 2.4. It consists of a platform with three motors, corresponding to the rotational axes, that can correct any rotational offset to a desired fixed position. It uses an internal 9-DOF IMU to calculate the forces needed to stabilise the device. Gimbals are not commonly used in sensor setups, as most sensors are usually mounted in fixed positions. Since a vessel displays much more movement in pitch and roll compared to onshore vehicles due to currents and waves, a gimbal could act as a stabiliser, possibly compensating for the complex kinematic movements.

Figure 2.4: Ronin-SC gimbal for remote control and 3-axis stabilisation.

2.4 Simultaneous localisation and mapping

Simultaneous localisation and mapping (SLAM) is a technique used for AVs that constructs a map of an unknown environment while simultaneously localising the position of the vehicles. The acronym *SLAM* was first introduced in 1995 in a mobile robotics survey paper and has since been widely researched within different areas of robotics [38]. It has been reported that a solution to the SLAM problem can be

seen as *holy grail* for the mobile robotics community, as it would provide the means to make a robot truly autonomous [4]. A successful implementation of SLAM would allow the AV to perform tasks such as path planning and obstacle avoidance. This would make the robot capable of exploring new environments without any prior knowledge.

The main problem with SLAM is that no sensor reading is perfect. That is, regardless of accuracy, there will always be an error in the estimation of both the vehicle position and the map, which might cause a deviation over time. In addition, the methods used for SLAM depend on both the sensor data and the environment of each AV. This problem is especially evident for boats, in part due to the different characteristics that water provides compared to solid ground, but mainly due to the increased complexity in the direction of movements (pitch, roll, heave[1]). For example, a SLAM algorithm that uses lidar data will not be able to see the surface of the water due to the penetration of light beams. Therefore, it is important to consider the application scenario when performing SLAM. In the following subsections, various methods that are used for SLAM are explained.

2.4.1 Scan registration

Scan registration is the procedure of matching multiple scans of data when the relative pose difference is unknown [39]. It includes finding the spatial transformation that consists of translating and rotating scans such that they can be merged into a globally consistent coordinate frame. A scan is typically defined as raw 3D point cloud data which, for most cases, are obtained from either lidars or depth cameras [40].

The most commonly used registration algorithm is the *iterative closest point* (ICP) algorithm which minimises the point-wise distances of the scans in order to align frames [41]. This algorithm performs well when there is sufficient overlap between scans where it converges to a local minimum [42]. However, the disadvantages are that, since it is a point-based method, it does not consider local features of the surface around each point, where it is also susceptible to statistical outliers. If the initial frames are not sufficiently aligned, ICP may fail to find the global minimum, and hence result in erroneous transformation. As a result of this, other methods are often combined with the ICP to find an initial estimate of the transformation [21][43].

The main issue with scan registration algorithms is that the transformation is usually non-trivial due to the difficulty of obtaining a point-wise correspondence between frames. Other drawbacks include that novel scan matching algorithms are usually limited to an application-specific environment and that the resulting registration is only computed over a small number of scans [44].

[1]Traversal in the vertical direction, even when the vessel is stationary

2.4.2 State estimation

State estimation is the procedure of estimating the internal states of a system. In the case of SLAM, this constitutes of estimating the AVs trajectory in the map based on different measurements from the external sensors.

2.4.2.1 Sensor fusion

Sensor fusion is the process of combining multiple sensor data from external sources such that the resulting information has less uncertainty, compared to the case where the data from each sensor would be used individually. It is common to complement the use of a GNSS with an IMU, which creates a GNSS-aided inertial navigation system (GNSS + INS). This combination has been shown to improve the performance of the position, velocity, and attitude estimates of the AV [45]. Sensor fusion is important since it can help stabilise occasionally noisy data, such as the multipath reception error.

2.4.2.2 Extended Kalman filter

Extended Kalman filter (EKF) is the nonlinear version of the Kalman filter (KF) which uses a series of sensor measurements, including noise and other inaccuracies, to produce estimates of unknown variables. It is a standard technique that is frequently used in various applications when the motion model of the AV is nonlinear [10]. The framework of the EKF can be described by the discrete nonlinear Markov model in Eq. 2.1

$$\begin{aligned} \boldsymbol{x_k} &= F(\boldsymbol{x_{k-1}}, \boldsymbol{u_k}) + \boldsymbol{w_{k-1}} \\ \boldsymbol{y_k} &= G(\boldsymbol{x_k}) + \boldsymbol{v_k} \end{aligned} \tag{2.1}$$

where $\boldsymbol{x_k}$ denotes the state vector and $\boldsymbol{u_k}$ is the control vector that holds the sensor data from the GNSS and IMU. $\boldsymbol{w_k}$ is the n-dimensional process noise vector that adds to the uncertainty of the motion model, $\boldsymbol{v_k}$ is the p-dimensional observational noise vector that adds to the uncertainty of the measurements, both are assumed to have zero mean and Gaussian noises with covariance $\boldsymbol{Q_k}$ and $\boldsymbol{R_k}$ respectively. The nonlinear dynamics of the system is described by F and G, which are assumed to be known.[2] $\boldsymbol{y_k}$ is the noisy output vector of the system.

The objective of the model in Eq. 2.1 is to recursively estimate the state vector $\boldsymbol{x_k}$ from the accumulated collection of noisy measurements $\boldsymbol{y_1}, ..., \boldsymbol{y_k}$. This is carried out by implementing a first-order Taylor expansion of the nonlinear system and recursively performing a *prediction* step and an *update* step. Let the notation $\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}$ represent the estimate of $\boldsymbol{x_k}$ at time k, given $k-1$ noisy estimates, see Eq. 2.2

$$\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} = \mathbb{E}(\boldsymbol{x_k}|\boldsymbol{y_{1:k-1}}) \tag{2.2}$$

[2] If the system model is inaccurate of not well known, then it is better to implement a particle filter, which utilises a Monte Carlo method.

The prediction step is then performed according to Eq. 2.3

$$\begin{aligned} &\text{Predicted state estimate} && \hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} = F(\hat{\boldsymbol{x}}_{\boldsymbol{k-1|k-1}}, \boldsymbol{u_k}) \\ &\text{Predicted covariance estimate} && \boldsymbol{P_{k|k-1}} = \boldsymbol{F_k P_{k-1|k-1} F_k^T} + \boldsymbol{Q_k} \end{aligned} \tag{2.3}$$

and the update step is performed according to Eq. 2.4

$$\begin{aligned} &\text{Measurement residual} && \tilde{\boldsymbol{z}}_{\boldsymbol{k}} = \boldsymbol{y_k} - G(\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}) \\ &\text{Residual covariance} && \boldsymbol{S_k} = \boldsymbol{G_k P_{k|k-1} G_k^T} + \boldsymbol{R_k} \\ &\text{Kalman gain} && \boldsymbol{K_k} = \boldsymbol{P_{k|k-1} G_k^T S_k^{-1}} \\ &\text{Update state estimate} && \hat{\boldsymbol{x}}_{\boldsymbol{k|k}} = \hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} + \boldsymbol{K_k} \tilde{\boldsymbol{z}}_{\boldsymbol{k}} \\ &\text{Update covariance estimate} && \boldsymbol{P_{k|k}} = (\boldsymbol{I} - \boldsymbol{K_k G_k}) \boldsymbol{P_{k|k-1}} \end{aligned} \tag{2.4}$$

where $\boldsymbol{F_k}$ and $\boldsymbol{G_k}$ are the Jacobians, defined according to Eq. 2.5

$$\boldsymbol{F_k} = \frac{\partial F}{\partial \boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}_{\boldsymbol{k-1|k-1}}, \boldsymbol{u_k}} \qquad \boldsymbol{G_k} = \frac{\partial G}{\partial \boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}} \tag{2.5}$$

It has been concluded that the EKF is the most widely used state estimator for non-linear systems [46]. However, since the EKF approximates the state distribution by a random Gaussian variable and propagates it analytically through the first-order linearisation of the non-linear system, it can introduce large errors in the true posterior of the mean and covariance [10]. This may lead to sub-optimal performance and divergence of the EKF. In order to address this, there have been modified versions of the EKF that better capture the non-linearities of the system dynamics.

2.4.2.3 Unscented Kalman filter

Unscented Kalman filter (UKF) is a modified version of the EKF that addresses the problem with the first-order linearisation by introducing a deterministic sampling approach. Instead of propagating a first-order Taylor expansion around the mean through the nonlinear system, the UKF selects a minimal set of carefully chosen sample points, called *sigma points*, which are then propagated instead. The transformation of the sigma points is intended to estimate the posterior distribution with greater precision.

By performing this methodology, it has been proven that the UKF is capable of capturing the posterior mean and covariance to an accuracy of a third-order Taylor expansion, in contrast to the first-order accuracy of the EKF [10]. It should also be noted that no explicit calculation of any Jacobians or Hessians is necessary and that the computational complexity of the UKF is identical to that of the EKF. The only disadvantage of implementing a UKF over an EKF is that, in practise, the latter is slightly more computationally efficient, yielding faster computation; otherwise, the UKF is more robust to nonlinear transformation and should be the desired choice [47]. The implementation of the UKF is covered in Sect. 3.1.3.

2.4.3 Kinematic problems

While SLAM has been considered a closed problem in some scenarios, it is not yet possible to apply a single algorithm that is applicable to all types of environments [5]. The main issue that arises when performing SLAM on a boat is due to the accumulated drift that occurs from the false estimations of various state estimation algorithms [48]. Another way to describe the same phenomenon is that most algorithms fail to converge to an optimal solution, hence converging to a local minimum instead of a global minimum. This is a common problem when the motion model of the AV is described by a non-linear system. It has also been concluded that the convergence of an algorithm strongly depends on having a sufficient number of features and that the initial guess of the estimation must be sufficiently close to the global minimum at the beginning of the algorithm [48].

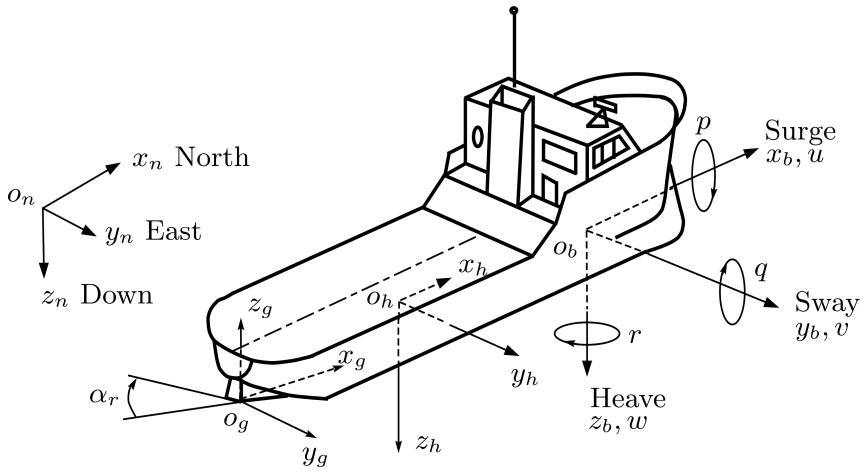

Figure 2.5: Kinematic movement and terminology of a vessel.

Given that most SLAM algorithms use a methodology that compares the relationship between two scans, there have been problems with estimating the initial pose and orientation of the first scan [48]. This has been illustrated to cause drifts in localisation due to an rotational between the coordinate systems of a lidar, and other sensor readings like GPS. The drifts are most common in the vertical direction, as sensor corrections are more common in the horizontal plane [6]. This is a particular issue in maritime outdoor environments, as the robot usually lacks ground-truth features from the ground and the roof. These inaccuracies might cause a theoretical unbounded drift in the vertical plane, while still being able to give reasonable results in the projected horizontal plane [7]. The complex movements of a vessel are illustrated in Fig. 2.5

2.5 Containerised software and microservices

Containerising an application is the concept of packaging a software binary together with its dependencies so that it can run reliably on computers with varying operating systems [49]. The principle of containerisation works well with the microservice software architecture pattern, where each microservice can be containerised individually. The principle of this paradigm is to divide the programme into several microservices, whose objective is to perform a single action with specified inputs and outputs. The conception of microservices can be thought of as an independent function that interacts via messages [50], where each microservice is responsible for performing a specific task. This methodology is a common course of action among developers when working with large projects, as it allows *continuous integration and deployment* (CI/CD), which is the concept of effectively building, testing, and merging programme code.

A common approach for performing the containerisation is through Docker, which is a set of open-source platform services that utilises virtualisation at the OS level to deliver software in packages. Docker has become the leading reference technology for building containers and is widely used within the industry [51].[3] A container can be realised through a Docker image, which is a set of instructions that acts as a blueprint for building the container.

[3]https://www.docker.com/

3
Methods

This section describes the methodology used in this project. First, the used SLAM algorithm is detailed in Sect 3.1, including the preprocessing, registration, and sensor fusion using an UKF. Furthermore, the software architecture is explained in Sect. 3.2. The two experiments concerning a software and hardware solution are described in Sect. 3.3.1 and Sect. 3.3.2. Lastly, Sect. 3.4 explains how the results govering the research questions were evaluated.

3.1 SLAM

The SLAM algorithm is divided into its sequential steps. Sect. 3.1.1 covers the preprocessing of the point cloud. Sect. 3.1.2 covers the normal distribution transform (NDT) and iterative-closest-point (ICP) algorithms, and Sect. 3.1.3 covers the Unscented Kalman filter (UKF) algorithm.

3.1.1 Point cloud preprocessing

The point cloud was pre-processed prior to registration (Sect. 3.1.2) in order to remove redundant points. The purpose was to remove the dynamical movement of the vessel together with noisy data. The filtering methods are described below.

3.1.1.1 Crop box filter

The crop box filter was used to retain points within the region of interest, where the input point cloud was filtered within a given box of defined minimum and maximum boundaries. The purpose of this filter is to exclude points that are either far away or close to the lidar, since these points are much noisier due to environmental interference.

Since the lidar was installed on the bow of the vessel, the points from the rear end of the cabin had to be removed to prevent the dynamical movement of the vessel from being continuously mapped. The removal of unnecessary points can also improve the processing speed of the algorithm and increase the performance of the mapping.

3.1.1.2 Statistical outlier removal

Due to measurement errors, certain data sets present a large number of shadow points, also known as noisy points, which complicate the estimation of local features of the point cloud [52]. The cause of these errors might be due to noisy measurement of the lidar, environmental interference, or object reflection characteristics. By performing a statistical analysis on each point's neighbourhood, outliers can be filtered by removing those points that do not meet certain criteria.

The methodology of this filter is to compute the distribution of each point's distances to its neighbours from the input point cloud. The algorithm iterates through the input point cloud twice, where during the first iteration the average distance is computed of each point to its nearest k neighbours. Assuming that the average distance between all points in the input point cloud and its nearest k neighbouring points meets the Gaussian distribution. During the next iteration, points are considered to be inliers or outliers depending on if their mean neighbour distance is below or above the threshold; see Algorithm. 1.

Algorithm 1 Statistical outlier removal

Input: P_i, k, α
Output: P_f

P_i represents the input point cloud
k represents the number of closest points around point $\mathbf{x}_i$
α represents the standard deviation multiplier
for every $\boldsymbol{x}_i \in P_\text{i}$ **do**
    Find the location of the k nearest neighbours to $\boldsymbol{x}_i$
    Compute the mean distance d_i from $\boldsymbol{x}_i$ to its k nearest neighbours
end for
Compute the mean μ_d of the distances d_i
Compute the standard deviation σ_d of the distances d_i
Compute the threshold $T = \mu_d + \alpha \cdot \sigma_d$
for every $\boldsymbol{x}_i \in P_\text{i}$ **do**
    if $d_i > T$ **then**
        Eliminate $\boldsymbol{x}_i$
    end if
end for
return Filtered point cloud P_f

3.1.2 Point cloud registration

The methods described in this section are based on an approach that only uses point-cloud data, and will denote the baseline method in this project. The registration is performed through a combination of the normal distribution transform (NDT) and iterative-closest-point (ICP) algorithm.

3.1.2.1 Normal distribution transform

Normal distribution transform (NDT) was first proposed by Biber and Straßer in 2003 as a method for 2D scan registration [53]. It is a common registration algorithm that is used in many fields for aligning poses. The NDT algorithm provides a *rough registration* of point clouds by matching the target cloud (a new frame scanned by the lidar) to a reference cloud according to a Gaussian distribution. The Gaussian distribution is generated by splitting up points from the reference cloud into voxels, where a voxel is a three-dimensional equivalent of a pixel. A probability density function (PDF) is computed for each cell, based on the point distribution within the cell. By assuming that the location of the reference scan points were generated by a normal random process, the likelihood of observing a particular point $\boldsymbol{x_i}$ is

$$p(\boldsymbol{x_i}) = \frac{1}{(2\pi)^{\frac{3}{2}}\sqrt{|\boldsymbol{\Sigma}|}} \exp\left(\frac{-(\boldsymbol{x_i}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x_i}-\boldsymbol{\mu})}{2}\right) \tag{3.1}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ denotes the mean vector and covariance matrix of the points within a voxel, see Eq. 3.2 and Eq. 3.3

$$\boldsymbol{\mu} = \frac{1}{|P_{\text{vox}}|}\sum_{i=1}^{|P_{\text{vox}}|} \boldsymbol{y_i} \tag{3.2}$$

$$\boldsymbol{\Sigma} = \frac{1}{|P_{\text{vox}}|-1}\sum_{i=1}^{|P_{\text{vox}}|} (\boldsymbol{y_i}-\boldsymbol{\mu})(\boldsymbol{y_i}-\boldsymbol{\mu})^T \tag{3.3}$$

where $\boldsymbol{y_{i=1\ldots,|P_{\text{vox}}|}}$ denotes the points within a voxel.

When using NDT for point cloud registration, the goal is to find the transformation matrix $T(\vec{p}, \boldsymbol{x_i})$ that maximises the likelihood of the target point cloud compared to the reference point cloud, see Fig. 3.1, where

$$\vec{p} = [t_x, t_y, t_z, \phi_x, \phi_y, \phi_z] \tag{3.4}$$

is the set of parameters that describes the translation and rotation, and

$$T(\vec{p}, \boldsymbol{x_i}) = \begin{bmatrix} c_y c_z & -c_y s_z & s_y \\ c_x s_z + s_x s_y c_z & c_x c_z - s_x s_y s_z & -s_x c_y \\ s_x s_z - c_x s_y c_z & c_x s_y s_z + s_x c_z & c_x c_y \end{bmatrix} \boldsymbol{x_i} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{3.5}$$

is the transformation matrix, where $c_i = \cos\phi_i$ and $s_i = \sin\phi_i$. The best transformation should therefore be the one that maximises the following likelihood function

$$\Psi = \prod_{k=1}^{|P_t|} p(T(\vec{p}, \boldsymbol{x_k})) \tag{3.6}$$

or equivalently, minimises the negative log-likelihood of Eq. 3.6

$$s(\vec{p}) = -\log \Psi = -\sum_{k=1}^{|P_t|} \log \Big(p(T(\vec{p}, \boldsymbol{x_k}) \Big) \tag{3.7}$$

where $p()$ is the PDF defined in Eq. 3.1 and $T()$ is the transformation matrix defined in Eq. 3.5. The size of the target point cloud is denoted as $|P_t|$. The optimal value of Eq. 3.7 can be found by using Newton-Raphsons method to find the parameters of Eq. 3.4. This is done by iteratively solving

$$\begin{aligned} \boldsymbol{H} \Delta \vec{p} &= -\vec{g} \\ \vec{p} &= \vec{p} + \Delta \vec{p} \end{aligned} \tag{3.8}$$

where $\boldsymbol{H}$ and $\vec{g}$ is the Hessian and gradient vector of Eq. 3.7, which are defined as

$$H_{ij} = \frac{\partial^2 s}{\partial p_i \partial p_j} \tag{3.9}$$

$$g_i = \frac{\partial s}{\partial p_i} \tag{3.10}$$

The advantage of using the NDT algorithm is that it provides a computationally fast estimate compared to other methods [54]. Due to NDTs averaging effect, it is also robust to noisy data. However, the performance of the algorithm is sensitive to the size selection of the voxels, meaning that the voxel size needs to be tuned depending on both the sensor and environmental conditions. For data collected in this work, a voxel size of $4.5 \times 4.5 \times 4.5$ [m^3] was used.

In previous work, NDT has been concluded to provide accurate results when faced with scan data that has little overlap and weak geometric features [55]. It is therefore used as an initial transformation before continuing further with more precise alignments, see Algorithm 2.

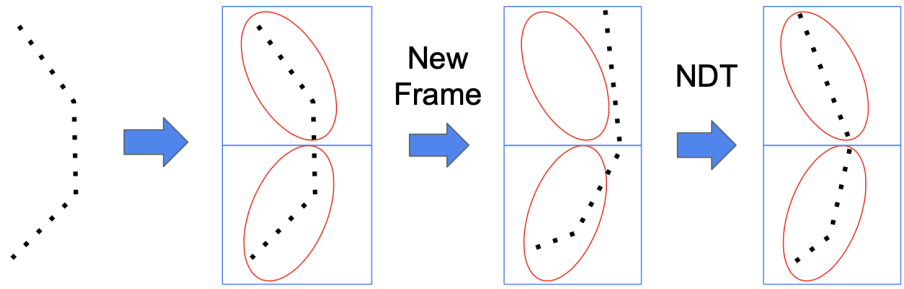

Figure 3.1: Two-dimensional illustration of the NDT algorithm. Points from the reference frame are divided into multiple voxels. The red ellipses illustrates the Gaussian distribution in each voxel generated by the points from the reference frame. When the new target frame is generated, it is then matched with the Gaussian distribution from the reference frame. A transformation is then performed between the target and reference frame, maximising the overlapping score.

Algorithm 2 Normal distribution transform (NDT)

Input: P_{r}, P_{t}

P_{r} represents the previous point cloud.
P_{t} represents the latest point cloud.
Divide P_{r} into voxels b_i of a specified size
for every voxel **do**
    Compute the mean $\boldsymbol{\mu}$ (see Eq. 3.2)
    Compute the covariance matrix $\boldsymbol{\Sigma}$ (see Eq. 3.3)
end for
while not converged **do**
    $score \leftarrow 0$
    $\vec{g} \leftarrow 0$
    $\boldsymbol{H} \leftarrow 0$
    for every $\boldsymbol{x}_k \in P_{\mathrm{t}}$ **do**
        Find the voxel b_i that contains $T(\vec{p}, \boldsymbol{x_k})$
        $score \leftarrow score + p(T(\vec{p}, \boldsymbol{x_k}))$
        Update $\vec{g}$ (see Eq. 3.10)
        Update $\boldsymbol{H}$ (see Eq. 3.9)
    end for
    Update parameters (see Eq. 3.8)
end while

3.1.2.2 Iterative-closest-point

Iterative-closest-point (ICP) is a *refine registration* pointwise algorithm that operates by matching different frames. The methodology of ICP is to compute the refine transformation between the reference and target point cloud by minimising the error of point-wise distances [42]. This is performed by minimising the sum of squared differences between the coordinates of the matched pairs through iteration, where the algorithm searches through a combination of rotations and translations; see Eq. 3.11

$$\vec{q_R}, \vec{q_T} = \underset{\vec{q_R}, \vec{q_T}}{\arg\min} \frac{1}{|P_\mathrm{r}|} \sum_i \left|\left| P_\mathrm{t}^i - \left(\boldsymbol{R}(\vec{q_R}) \cdot P_\mathrm{r}^i + \vec{q_T} \right) \right|\right|^2 \tag{3.11}$$

where, $P_\mathrm{t}, P_\mathrm{r}$ denotes the target (previous) and reference (latest) frame, and $\vec{q_R}, \vec{q_T}$ are vectors that describe rotation and translation. The first step of the algorithm is to define the pointwise correspondence of the target scan with the reference scan. This is done by pairing each point in the target scan with the closest neighbour in the reference scan, according to the Euclidean distance. After computing the pointwise correspondance, the mean point for both point sets is computed according to Eq. 3.12

$$\boldsymbol{\mu_r} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x_i} \qquad \text{and} \qquad \boldsymbol{\mu_t} = \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{\tilde{x}_i} \tag{3.12}$$

where $N = |P_r|$ and $M \in [1, N]$ is the number of points matched between the reference frame and the target frame,[1] the values given in Eq. 3.12 represents the centre of mass for each frame. The optimal rotation and translation of Eq. 3.11 is then found using a quaternion-based method [56]. This includes first computing the cross-covariance matrix Σ_{rt} of the reference and target set according to Eq. 3.13

$$\Sigma_{rt} = \frac{1}{N} \sum_{i=1}^{N} \left[(\boldsymbol{\tilde{x}_i} - \boldsymbol{\mu_t})(\boldsymbol{x_i} - \boldsymbol{\mu_r})^t \right] \tag{3.13}$$

By then using the cyclic components of the antisymmetric matrix $A_{ij} = (\Sigma_{rt} - \Sigma_{rt}^T)_{ij}$ to form the column vector $\Delta = [A_{23}, A_{31}, A_{12}]^T$, the following symmetric 4×4 matrix is constructed

$$Q(\Sigma_{rt}) = \begin{bmatrix} \mathrm{Tr}(\Sigma_{rt}) & \Delta^T \\ \Delta & \Sigma_{rt} + \Sigma_{rt}^T - \mathrm{Tr}(\Sigma_{rt}) \boldsymbol{I_3} \end{bmatrix} \tag{3.14}$$

where Tr denotes the trace of a matrix and $\boldsymbol{I_3}$ is the 3×3 identity matrix. The optimal rotation can now be obtained by computing the unit quaternion eigenvector $\vec{q_R} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^t$ corresponding to the maximum eigenvalue of Eq. 3.14, where $q_0 \geq 0$ and $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$. The optimal translation vector $\vec{q_T}$ is then given by

$$\vec{q_T} = \boldsymbol{\mu_r} - \boldsymbol{R}(\vec{q_R}) \boldsymbol{\mu_t} \tag{3.15}$$

where the rotation matrix generated by a unit quaternion is defined as in Eq. 3.16

$$\boldsymbol{R}(\vec{q_R}) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^3 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix} \tag{3.16}$$

[1] In the example given in Fig. 3.2, $N = 7$ and $M = 3$.

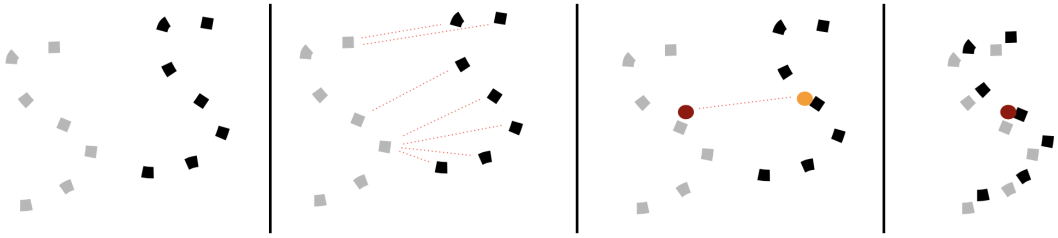

Figure 3.2: First panel displays the reference frame (black) and the target frame (gray). Second panel illustrates how the ICP algorithm finds a pointwise correspondance between the reference frame and the target frame , where the dotted red lines is the shortest Euclidean distances between points. Third panel highlights the center of mass for both set of points, according to Eq. 3.12. Forth panel shows the resulting translation and rotation after one iteration where the center of mass for both set of points have overlapped.

By iteratively updating and refining the relative poses according to Eq. 3.11, the ICP algorithm is able to provide a more precise fit compared to NDT. However, the drawback is that it is computationally heavy, especially when working with dense clouds. Due to the non-convexity of the ICP optimisation, the performance of the algorithm is strongly dependent on the initial estimate between frames [57], which is a limitation with this approach. The nearest-neighbour point generally does not correspond to the same point on the scanned frame, which means that if the poses in the ICP algorithm are too far apart, the algorithm may fail to converge. See Fig. 3.2 for an illustration of the first iteration.

To run efficiently, the alignment of two frames must have a sufficient overlap, where it has been concluded that the algorithm always converges to a local minimum [42]. Therefore, NDT was performed prior to ICP as an initial estimation prior to further refinement. It is noted that ICP does not require having equally sized frames since it automatically takes care of point-wise correspondence due to using the closest pair, see Algorithm 3.

Algorithm 3 Iterative-closest-point (ICP)

Input: P_{r}, P_{t}

P_{r} represents the previous point cloud.
P_{t} represents the latest point cloud.
while not converged **do**
    Find a pointwise correspondence between P_{r} and P_{t}
    Compute the mean points μ_r, μ_t of both sets (see Eq. 3.12)
    Compute the cross-covariance matrix Σ_{rt} (see Eq. 3.13)
    Compute the rotation $\vec{q_R}$ and translation $\vec{q_T}$ (see Eq.3.14,3.15,3.16)
end while

3.1.3 UKF

The *Unscented Kalman Filter* (UKF) was implemented to calculate a state estimate based on the sensor data recorded. The difference in position between two frames is then used to calculate an initial transformation for the ICP algorithm, where the implementation of the UKF will denote the *software-based approach* in this project. The UKF uses data from the IMU and GNSS to predict the future pose of the vessel. It operates by recursively propagating a random variable $\boldsymbol{x}$, which describes the states, through a non-linear function $\boldsymbol{y} = g(\boldsymbol{x})$, which describes the dynamics of the model.

3.1.3.1 Initialisation

The UKF starts by initialising its current pose $(X, Y, v, \theta, \psi, \varphi)$ based on the measurements of the GNSS and IMU, where X, Y denotes the transformed longitudinal and latitudinal coordinates from the GNSS, v denotes the velocity of the vessel in the plane, and θ, ψ, φ denotes the pitch, roll, and yaw measurements from the IMU. The other states $(Z, \dot{\theta}, \dot{\psi}, \dot{\varphi}, \eta_{\dot{v}}, \eta_{\dot{Z}}, \eta_{\dot{\theta}}, \eta_{\dot{\psi}}, \eta_{\dot{\varphi}})$ are initialised as zero, where Z denotes the altitudinal position of the vessel, $\dot{\theta}, \dot{\psi}, \dot{\varphi}$ denotes the pitch, roll, yaw rate, $\eta_{\dot{v}}, \eta_{\dot{Z}}$ denotes the noise of the acceleration in both the vessels and the altitudinal direction, $\eta_{\dot{\theta}}, \eta_{\dot{\psi}}, \eta_{\dot{\varphi}}$ denotes the noise of the angular rate of pitch, roll, and yaw. The total number of states is shown in Eq. 3.17.

$$\boldsymbol{x} = (X, Y, Z, v, \theta, \psi, \varphi, \dot{\theta}, \dot{\psi}, \dot{\varphi}, \eta_{\dot{v}}, \eta_{\dot{Z}}, \eta_{\dot{\theta}}, \eta_{\dot{\psi}}, \eta_{\dot{\varphi}}) \tag{3.17}$$

The state covariance matrix $\boldsymbol{P}$ which is of size 15×15, that describes the uncertainty of the states and the performance of the filter, is initialised to a large value.

3.1.3.2 Prediction step

The nonlinear pose transformation f of the vessel was described by a combination of a *constant velocity* (CV) model [58] and a *constant turn rate velocity* (CTRV) model [59]. The CTRV model is chosen to describe the dynamics when the yaw rate is above a certain threshold; otherwise, the CV model is chosen. This particular methodology has previously been shown to successfully predict the movement of road vehicles with six degrees of freedom and was therefore replicated here [21].

The prediction step starts with the generation of $2N + 1$ sigma points $\mathcal{X}_i$ with their corresponding weights W_i, where $N = 15$ is the number of states in the model. This is done according to Eq. 3.18

$$\begin{aligned}
\mathcal{X}_0 &= \bar{\boldsymbol{x}}_{k-1} && && (3.18)\\
\mathcal{X}_i &= \bar{\boldsymbol{x}}_{k-1} + \Big(\sqrt{(N+\lambda)\boldsymbol{P}_{k-1}}\Big)_i && i = 1, ..., N\\
\mathcal{X}_i &= \bar{\boldsymbol{x}}_{k-1} - \Big(\sqrt{(N+\lambda)\boldsymbol{P}_{k-1}}\Big)_{i-N} && i = N+1, ..., 2N\\
W_0^{(m)} &= \lambda/(N+\lambda) && && (3.19)\\
W_0^{(c)} &= \lambda/(N+\lambda) + (1-\alpha^2+\beta)\\
W_i^{(m)} &= 1/\Big(2(N+\lambda)\Big) && i = N+1, ..., 2N\\
W_i^{(c)} &= 1/\Big(2(N+\lambda)\Big) && i = N+1, ..., 2N
\end{aligned}$$

where $\bar{\boldsymbol{x}}_k$ denotes the mean point of the states at timestep k, $\lambda = \alpha^2(N + \kappa) - N$ is a scaling parameter where $\kappa \geq 0$ and $\alpha \in [0, 1)$ are parameters that influence how far the sigma points are away from the mean. β is used to describe the prior distribution of states, where it has been concluded that $\beta = 2$ is optimal for Gaussian distributions [10]. $(\sqrt{(N+\lambda)\boldsymbol{P}_{k-1}})_i$ is the i:th row of the square root of the matrix, where $A = \sqrt{D}$ is a square root of the matrix if $D = AA^T$ holds. The superscripts m and c of the weights in Eq. 3.19 denote the abbreviation for *sigma points* and *covariance*.

The sigma points are then propagated through the non-linear transformation function f according to Eq. 3.20

$$\tilde{\mathcal{X}}_i = f(\mathcal{X}_i) \qquad\qquad i = 0, ..., 2N \tag{3.20}$$

The propagated sigma points $\tilde{\mathcal{X}}_i$ are then weighted to produce the mean and covariance of the posterior, see Eq. 3.21

$$\begin{aligned}
\hat{\mathcal{X}} &= \sum_{i=0}^{2N} W_i^{(m)} \tilde{\mathcal{X}}_i && (3.21)\\
\hat{\boldsymbol{P}}_{\boldsymbol{x}} &= \sum_{i=0}^{2N} W_i^{(c)} (\tilde{\mathcal{X}}_i - \hat{\mathcal{X}})(\tilde{\mathcal{X}}_i - \hat{\mathcal{X}})^T + \boldsymbol{Q}
\end{aligned}$$

where $\boldsymbol{Q}$ is the covariance of the noise from the motion model. Given the predictions in Eq. 3.21, a new set of sigma points $\mathcal{S}_i$ and weights W is calculated. These new points are then transformed through the measurement function h according to Eq. 3.22

$$\tilde{\mathcal{Y}}_i = h(\mathcal{S}_i) \qquad\qquad i = 0, ..., 2N \tag{3.22}$$

where h is the function that updates the states depending on the obtained sensor measurement.

3.1.3.3 Update step

The mean and covariance of the transformed points from Eq. 3.22 are then calculated according to Eq. 3.23

$$\begin{aligned}\hat{\mathcal{Y}} &= \sum_{i=0}^{2N} W_i^{(m)} \tilde{\mathcal{Y}}_i \\ \hat{\boldsymbol{P}}_{\boldsymbol{yy}} &= \sum_{i=0}^{2N} W_i^{(c)} (\tilde{\mathcal{Y}}_i - \hat{\mathcal{Y}})(\tilde{\mathcal{Y}}_i - \hat{\mathcal{Y}})^T + \boldsymbol{R} \\ \hat{\boldsymbol{P}}_{\boldsymbol{xy}} &= \sum_{i=0}^{2N} W_i^{(c)} (\tilde{\mathcal{S}}_i - \hat{\mathcal{X}})(\tilde{\mathcal{Y}}_i - \hat{\mathcal{Y}})^T\end{aligned} \tag{3.23}$$

where $\boldsymbol{R}$ is the covariance matrix of the measurement noise. To update the final mean and covariance estimates, the Kalman gain K is first calculated according to Eq. 3.24

$$\mathcal{K} = \hat{\boldsymbol{P}}_{\boldsymbol{xy}} \hat{\boldsymbol{P}}_{\boldsymbol{yy}}^{-1} \tag{3.24}$$

The final update is then computed according to Eq. 3.25

$$\begin{aligned}\acute{\mathcal{X}} &= \hat{\mathcal{X}} + \mathcal{K}(\mathcal{S} - \hat{\mathcal{Y}}) \\ \boldsymbol{P}_k &= \hat{\boldsymbol{P}}_{\boldsymbol{x}} - \mathcal{K} \hat{\boldsymbol{P}}_{\boldsymbol{yy}} \mathcal{K}^T\end{aligned} \tag{3.25}$$

When the new states of the vessel have been updated, the scan of the point cloud is transformed by translating every point to the new predicted UKF position. The rotation of the cloud is then obtained by first performing an *Euler to quaternion* conversion of the roll ψ, pitch θ and yaw φ angle according to Eq. 3.26 [60]

$$\vec{q_R} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\varphi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\varphi}{2} \\ \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\varphi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\varphi}{2} \\ \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\varphi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\varphi}{2} \\ \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\varphi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\varphi}{2} \end{bmatrix} \tag{3.26}$$

The final rotation is then generated by an quaternion rotation, identical to that described in Eq. 3.16. Finally, the steps of the UKF prediction and update are summarised in Algorithm 4–5.

Algorithm 4 UKF prediction

1: Generate sigma points
2: $\mathcal{X} = [\bar{\boldsymbol{x}}_{k-1}, \bar{\boldsymbol{x}}_{k-1} \pm \sqrt{(N+\lambda)\boldsymbol{P}_{k-1}}]$
3: Predict future state
4: $\tilde{\mathcal{X}} = f(\mathcal{X})$
5: $\hat{\mathcal{X}} = \sum_i W_i^{(m)} \tilde{\mathcal{X}}_i$
6: Predict error covariance
7: $\hat{\boldsymbol{P}}_{\boldsymbol{x}} = \sum_i W_i^{(c)} (\tilde{\mathcal{X}}_i - \hat{\mathcal{X}})(\tilde{\mathcal{X}}_i - \hat{\mathcal{X}})^T + \boldsymbol{Q}$
8: Recalculate sigma points
9: $\mathcal{S} = [\bar{\hat{\boldsymbol{x}}}_k, \bar{\hat{\boldsymbol{x}}}_k \pm \sqrt{(N+\lambda)\boldsymbol{P}_x}]$
10: Continue with update step

Algorithm 5 UKF update

1: Predict future state
2: $\tilde{\mathcal{Y}} = f(\mathcal{S})$
3: Compute Kalman gain
4: $\hat{\mathcal{Y}} = \sum_i W_i^{(m)} \tilde{\mathcal{Y}}_i$
5: $\hat{\boldsymbol{P}}_{\boldsymbol{yy}} = \sum_i W_i^{(c)} (\tilde{\mathcal{Y}}_i - \hat{\mathcal{Y}})(\tilde{\mathcal{Y}}_i - \hat{\mathcal{Y}})^T + \boldsymbol{R}$
6: $\hat{\boldsymbol{P}}_{\boldsymbol{xy}} = \sum_i W_i^{(c)} (\tilde{\mathcal{S}}_i - \hat{\mathcal{X}})(\tilde{\mathcal{Y}}_i - \hat{\mathcal{Y}})^T$
7: Update states and covariance
8: $\acute{\mathcal{X}} = \hat{\mathcal{X}} + \mathcal{K}(\mathcal{S} - \hat{\mathcal{Y}})$
9: $\boldsymbol{P}_k = \hat{\boldsymbol{P}}_{\boldsymbol{x}} - \mathcal{K} \hat{\boldsymbol{P}}_{\boldsymbol{yy}} \mathcal{K}^T$
10: Return to prediction step

3.1.3.4 Heading approximation

Since a 6-axis IMU can only deliver a relative direction, the absolute direction of the vessel was computed according to Eq. 3.27

$$\varphi = \arctan(X, Y) \tag{3.27}$$

where the positional coordinates were obtained from the GNSS. The longitude and latitude reading of the GNSS was transformed into Cartesian coordinates (X, Y) according to the Mercator projection [61], where (X, Y) denotes the current position of the vessel and the origin is the position two seconds prior. The delay of two seconds was introduced to acquire a smoother variation; see Fig. 3.3.

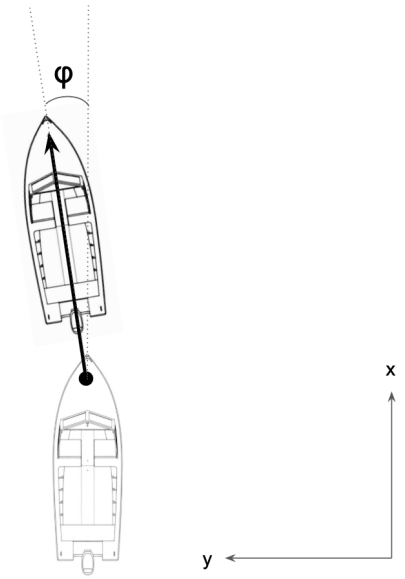

Figure 3.3: Illustration of how the heading φ in Eq. 3.27 is computed. The current position of the vessel (black) is compared with the position two seconds earlier (grey). The coordinate system is rotated according to the setup of the lidar, see Fig. 3.5.

3.1.3.5 Velocity approximation

Since it was not possible to obtain a velocity reading from the sensors, the vessel velocity in frame t was computed according to Eq. 3.28

$$v(t) = \frac{1}{50} \sum_{i=0}^{49} s(t-i) \cdot dt \tag{3.28}$$

where dt is the reciprocal of the GNSS frequency and $s(t)$ is the difference in GNSS distance between frames t and $t-1$. The latter is computed according to Eq. 3.29

$$s(t) = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_t - \varphi_{t-1}}{2}\right) + \cos\varphi_{t-1} \cdot \cos\varphi_t \cdot \sin^2\left(\frac{\lambda_t - \lambda_{t-1}}{2}\right)}\right) \tag{3.29}$$

where φ_{t-1}, φ_t is the latitude and λ_{t-1}, λ_t is the longitude, in radians, of the position of the vessels in frame $t-1$ and t. The radius of the Earth is denoted by R. To

reduce the noise of the GNSS data in Eq. 3.28, the instantaneous velocity was computed as a mean over 50 frames (5 s).

3.2 Software architecture

Since the team at Reeds has previously emphasised the positive effects of using containerised design principles when creating algorithms for AVs [62], it was decided to implement a containerised software framework in order to realise the SLAM algorithm. The framework was divided into two different microservices, where one of the services was responsible for the NDT and ICP algorithm, and the other for the UKF algorithm. Data capture, communication and hardware interfacing were achieved using the OpenDLV software framework [63], powered by libcluon.[2] Each microservice was built and run as a docker container. Both services were started by using a *docker-compose file*, which ensures that all services are run in parallel.

3.2.1 Algorithms

The algorithms used for SLAM in this paper are summarised in Fig. 3.4. Algorithm 1 is only utilising lidar data while performing NDT and ICP. Algorithm 2 is utilising lidar data together with IMU and GNSS data, where the UKF acts as an initial transformation prior to the ICP if the predicted yaw deviation is below a threshold; otherwise, the registration will continue with the NDT.[3]

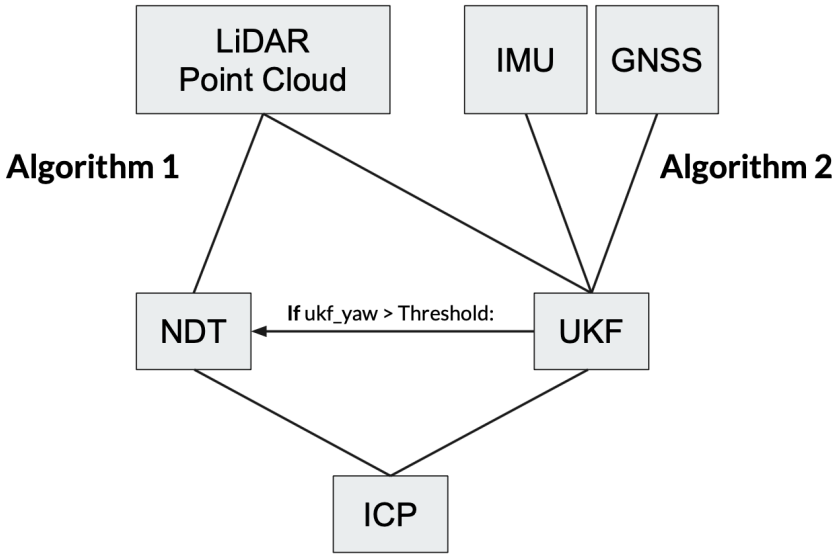

Figure 3.4: Overview of this papers SLAM algorithms. Algorithm 1 is implementing a *only lidar*-approach while Algorithm 2 is implementing *sensor fusion*.

[2] `https://github.com/chrberger/libcluon`

[3] In the continuation of this report, Algorithm 1 and Algorithm 2 will be referred to as *only lidar* and *UKF*, respectively.

3.2.2 Interfacing the Ouster OS2-128 sensor

In order to utilise the point cloud data for the algorithms, an interface was created for the lidar sensor. The conversion of sensor data to point cloud data was performed according to Eq. 3.30.

$$
\begin{aligned}
r &= range\_mm \\
n &= LiDAR\_to\_beam\_orgin\_mm \\
\theta_{encoder} &= 2\pi \cdot \left(1 - \frac{encoder\_count}{90112}\right) \\
\theta_{azimuth} &= -2\pi \frac{beam\_azimuth\_angles[i]}{360} \\
\phi &= 2\pi \frac{beam\_altitude\_angles[i]}{360}
\end{aligned}
\tag{3.30}
$$

$$
\begin{aligned}
x &= (r-n)\cos(\theta_{encoder} + \theta_{azimuth})\cos(\phi) + n\cos(\theta_{encoder}) \\
y &= (r-n)\sin(\theta_{encoder} + \theta_{azimuth})\cos(\phi) + n\sin(\theta_{encoder}) \\
z &= (r-n)\sin(\phi)
\end{aligned}
$$

where the specific details of the lidar sensor are illustrated in Fig. 3.5 [64].

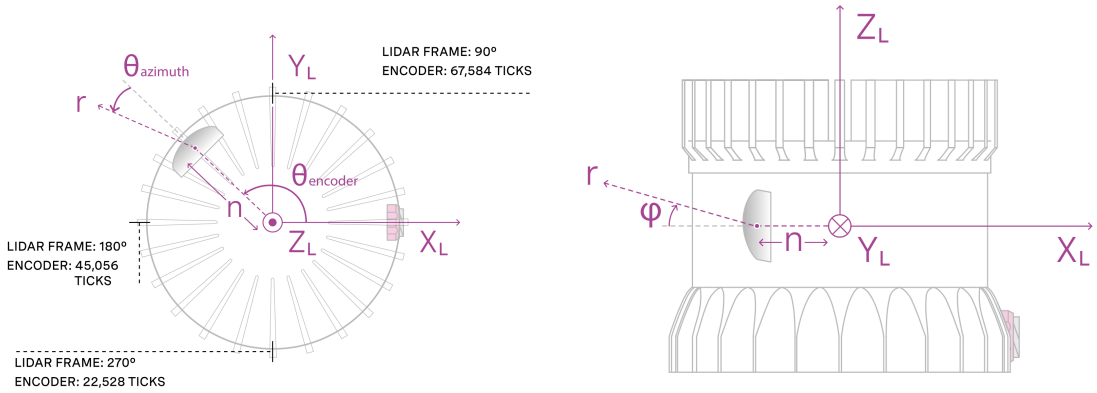

Figure 3.5: The left figure displays a top-down view of the lidar Coordinate Frame. The figure on the right displays a side view of the lidar coordinate frame.

3.3 Experiments

The experiments are divided into a vessel recording, which focuses on a software-based approach with additional sensors in order to perform a UKF state estimation, as well as laboratory recordings with a gimbal-stabilised setup. Both runs are also referred to as software-based (UKF) and hardware-based (gimbal).

3.3.1 Vessel experiment

The initial data logging was an essential part of evaluating the problems that arise when trying to perform SLAM in a complex kinematic environment. As a vessel on any body of water has significantly more movement in most degrees of freedom compared to automotive vehicles, this step was important to identify the shortcomings of a lidar-based SLAM.

It was decided to perform two data logging sessions along the coastline of Gothenburg, specifically starting from Långedrag, Västra Frölunda. These two sessions are referred to as *first run* and *second run*. When recording different runs, it was essential to capture multiple different conditions and situations. Therefore, these varied in velocity, amount of observable features, dynamic objects, and special cases like loop-closing scenarios, docking, and static behaviour. The raw data from the runs are summarised in Sect 4.1. The aim of constructing these runs was to focus on how the algorithms would perform in the relatively calm bay area.

The vessel used in this project can be seen in Fig. 3.6, it was a 13 m long boat that is specially designed for rescue missions. The vessel was equipped with supporting sensors on a compact tripod to evaluate the odometry, the sensor setup is illustrated in Fig. 3.8 and is fully described in Sect. 3.3.1.1.

Figure 3.6: SSRS-1100, the vessel that was used to collect data.

The paths of the evaluated runs are illustrated in Fig. 3.7, where the first run included multiple loops near the pier and the second run included higher speeds while traversing along the coast.

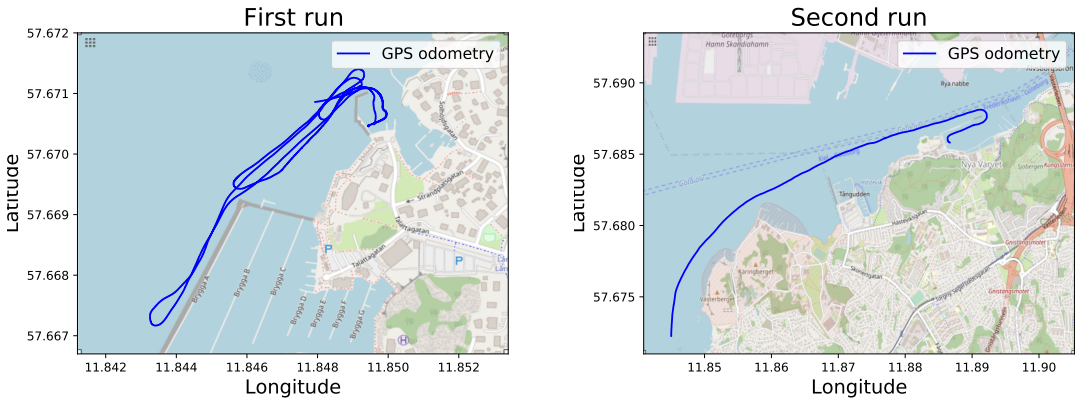

Figure 3.7: Path followed in first and second run.

3.3.1.1 Hardware setup - Vessel

The hardware in this experiment included an Ouster OS2-128 lidar, an external 6-axis IMU and a Trimble SPSx61 modular GPS receiver together with a single GA530 antenna. See Fig. 3.8. The lidar obtained a resolution of 128 beams in the vertical axis and 1024 increments on the horizontal plane at a frequency of 10 Hz. The field of view was 22.5° in the vertical direction and 360° in the horizontal direction. The IMU composed of a 3-axis gyro with a 3-axis accelerometer and operated at a frequency of 100 Hz. The GNSS obtained latitude and longitude readings while operating at a frequency of 20 Hz.

Figure 3.8: Sensor platform setup for the vessel run, containing a Ouster-OS2-128 lidar, an IMU and a GNSS antenna.

3.3.2 Gimbal experiment

As the vessel used for the previous experiments broke down due to an engine failure, further experiments were performed in the Revere laboratory. The goal was to evaluate how a gimbal-stabilised sensor platform would compare to an unstabilised one. In order to simulate the conditions on a vessel, the gimbal was programmed to rotate the lidar on the x- and y-axis in a sinus-wave pattern during the unstabilised run. This results in a larger change of the perceived vertical point cloud window. The main concern of this experiment was to observe whether any significant deviations in the vertical plane would occur with the use of a gimbal and how they compare to the simulated vessel movement. Using a gimbal, the rotational position is restricted to movement in the yaw direction. Any changes in pitch and roll should be continuously adjusted by the motors and can be assumed to be equal to zero. Due to this, the alignment between two frames is reduced from six to four dimensions, hence expecting to result in more accurate tracking of the sensor odometry.

For the recording session, the sensor platform was placed on a cart and driven through the laboratory in the shape of an eight. Due to the environment, it was not possible to record GNSS data and therefore it was not possible to evaluate the deviation in the horizontal position.

3.3.2.1 Hardware setup - Laboratory

Once again, the Ouster OS2-128 lidar was used. The internal IMU of the lidar data was recorded, but it was disregarded, as it differs in quality from the IMU used in the other recording sessions. With a mounting plate, the lidar was attached to a DJI Ronin-SC. This cost-efficient gimbal is usually used for mirrorless cameras, but due to the low weight of the Ouster lidar, it was able to stabilise it without much effort. Some careful planning with the cable setup was required as it can shift the centre of gravity, possibly resulting in force overload for the gimbal motors.

Figure 3.9: Sensor platform setup for the laboratory run, containing a Ouster-OS2-128 mounted on a DJI Ronin-SC gimbal.

	First run	Second run	Third run	Forth run
Frames	18635	14161	560	570
Algorithm Time (s)	1863.5	1461.1	56.0	57.0
Speed (m/s)	0.0-3.8	0-4.9	-	-
Distance (m)	3086	4923	40	40
Lidar Freq. (Hz)	10	10	10	10
Lidar Mode	1024	1024	2048	2048
In water	Yes	Yes	No	No
GNSS	Yes	Yes	No	No
Min Cropbox	$(\pm5, \pm5, \pm3)$	$(\pm5, \pm5, \pm3)$	$(\pm2, \pm2, \pm2)$	$(\pm2, \pm2, \pm2)$
Max Cropbox	-	-	$(\pm40, \pm40, \pm8)$	$(\pm40, \pm40, \pm8)$

Table 3.1: Logging data from the four recordings.

3.4 Evaluation of results

In order to evaluate the proposed algorithms, it was important to evaluate multiple metrics. As a SLAM method always consists of mapping and localisation, both parts must perform well.

Evaluating the performance of the localisation is done mainly by analysing the computed trajectory of the lidar. As a 6-DOF IMU was used in the experiments, the starting positions of the rotational angles are unknown. Therefore, the coordinate system of the lidar does not correlate to the global frame. If the tracked movement of the lidar exerts a constant decrease in the z-axis, this does not necessarily correspond to a bad localisation. It just means that due to the rotational offset between the coordinate systems, that it moves in the vertical axis of the global frame. Therefore, a constant velocity in the z-axis is an indicator for a well-performing localisation. To evaluate the performance of the vertical drift, the acceleration on the z-axis was computed to track any orthogonal offsets that would occur. That is the key indicator used to assess the deviations. Note that the main reason for choosing to evaluate the acceleration, and not the velocity, was because the prior is easier to compare between runs. Furthermore, the evaluation of the movement was split into the horizontal and vertical plane. The movement in the horizontal plane was compared with the ground truth of the GNSS data, while the movement in the vertical plane was compared with the ground truth of zero, based on the assumption that the logging area was flat. The reason for splitting up the evaluation of the movement was because previous studies had claimed that sensor corrections were common in the horizontal plane, while still deviating in the vertical plane [6].

The performance of the mapping was evaluated through a comparison with chart data taken from OpenSeaMap [65]. When the alignment of two frames results in failed registration, the accumulated offsets cause smearing effects on the stitched point cloud. These offsets will either result in erroneous rotation or stacking of the points, where the performance of the latter was evaluated through observation.

In order to examine the performance of the SLAM algorithms in different conditions, the error metrics from the localisation was compared to the average

velocity of the sensor and to the density of the point cloud. The goal was to investigate whether there existed any correlation in the SLAM performance between the environmental factors.

The evaluation of how the software- versus hardware-based solution performed was carried out by comparing the error metrics between the experiments described in Sect. 3.3.1 and Sect. 3.3.2, and how they affected a proposed sensor platform.

4
Results

This chapter is divided into the parts for each experiment and are benchmarked with the lidar-only based approach described in Sect. 3.2. Sect. 4.1 is concered with the vessel experiments, and its evaluation of the mapping and localisation. Sect. 4.2 details the results of the gimbal stabilised run. Finally the findings about minimal SLAM conditions are presented in Sect. 4.3.

4.1 Vessel experiment

The Vessel experiment included two data logging sessions along the coastline of Gothenburg. The total distances from the first run and the second run were 3.086 m and 4.923 m, and each run took thirty and twenty-five minutes respectively. The velocities of the vessel in the first and second run are illustrated in Fig. 4.1, and the features of each run is illustrated in Fig. 4.2. Note that the vessel was static during the first 4,250 frames of the first run and last 1,000 frames of the second run, the static portions of each run were not evaluated.

As data sets like KiTTI are usually split into multiple smaller runs, the data logging sections for the localisation presented are divided into sections of 1000 frames each, which equate to 100 s of recording. This is longer than the average length in such data sets in order to investiage the behaviour under extended periods of time.

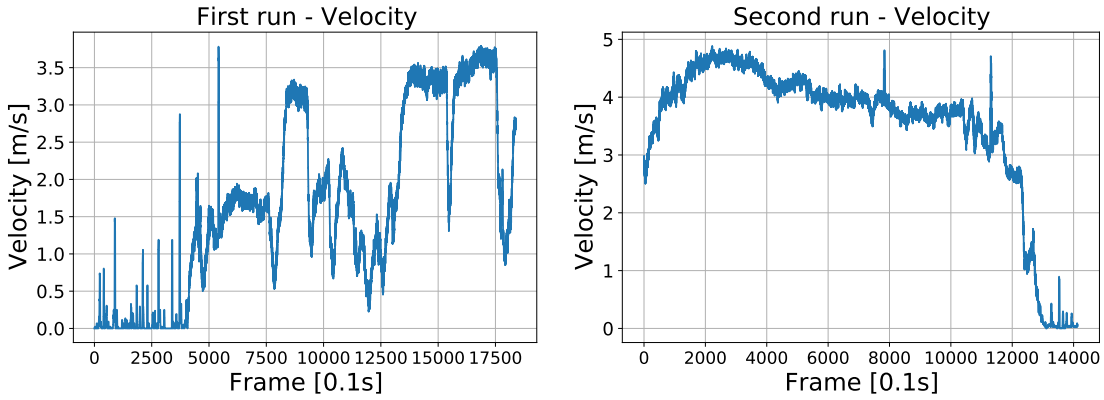

Figure 4.1: Velocity of the vessel in the first and second run. The velocity was computed from the GPS data, according to Eq. 3.28.

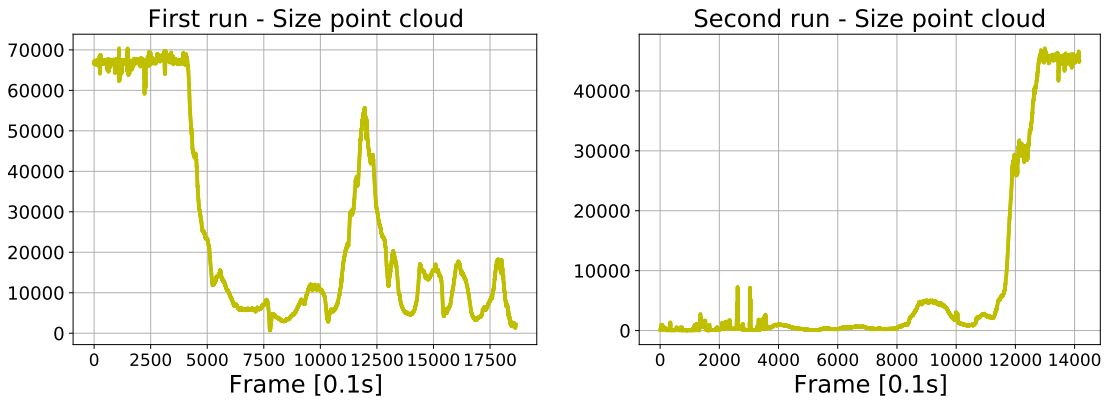

Figure 4.2: Size of the point clouds over different frames for the first and second run.

4.1.1 UKF performance

Fig. 4.3 depict the trajectory obtained from the UKF localisation of the first and second run, compared to the ground truth from the GNSS data. The sensor fusion was using data from the IMU and GNSS. It is illustrated that the filtered trajectory has a sub-meter accuracy during the straight segments while having some deviation when the vessel is turning. It is also shown that the UKF localisation is able to compensate during the cases when the GNSS data is faulty.

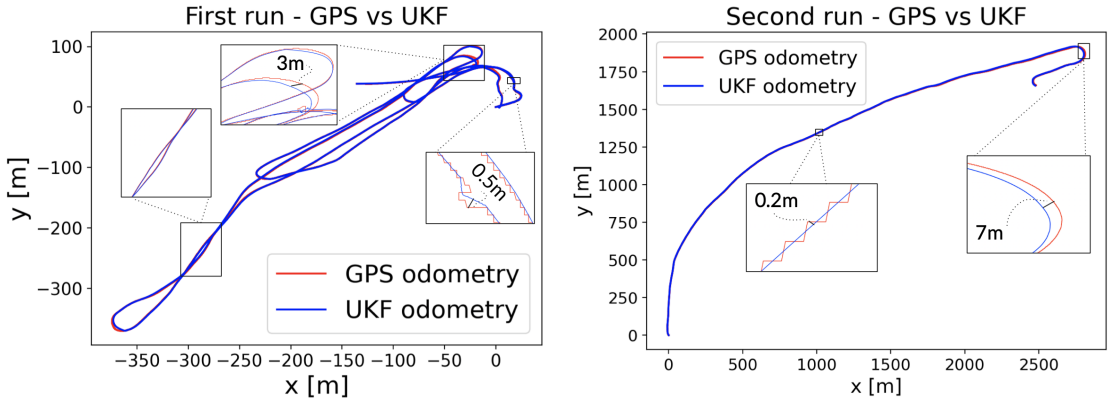

Figure 4.3: Illustration of the UKF performance from the first and second run. The zoomed in segments highlight when the vessel was heading straight ahead and in turns. It is also illustrated that the filtered trajectory from the UKF delivers a smoother path compared to the zigzag characteristics of the GPS.

The performance of the UKF, measured by the root mean square error between GNSS and the filtered state, was computed for each section of the first run in Table. 4.1, and for the second run in Table. 4.2. The vessel's average velocity was also included in both tables.

Frame	RMSE [m]	Velocity [$\mathrm{m\,s^{-1}}$]
4,250–5,250	0.38	1.19
5,250–6,250	0.88	1.60
6,250–7,250	0.43	1.72
7,250–8,250	2.56	1.38
8,250–9,250	4.98	3.08
9,250–10,250	1.66	1.83
10,250–11,250	4.39	1.65
11,250–12,250	0.60	0.92
12,250–13,250	1.82	1.31
13,250–14,250	4.91	3.11
14,250–15,250	3.62	3.30
15,250–16,250	3.83	3.00
16,250–17,250	4.80	3.60
17,250–18,250	3.94	2.19

Table 4.1: RMSE between filter and GNSS, average velocity for sections of the first run.

Frame	RMSE [m]	Velocity [$\mathrm{m\,s^{-1}}$]
0–1,000	0.28	3.60
1,000–2,000	2.26	4.38
2,000–3,000	7.05	4.67
3,000–4,000‡	-	4.53
4,000–5,000	14.04	4.20
5,000–6,000	13.95	4.11
6,000–7,000	14.72	3.96
7,000–8,000	16.66	3.93
8,000–9,000	17.33	3.70
9,000–10,000	16.57	3.70
10,000–11,000	14.30	3.62
11,000–12,000	11.47	3.18
12,000–13,000	5.58	1.51

‡ This sequence failed to register data for the UKF algorithm.

Table 4.2: RMSE between filter and GNSS, average velocity for sections of the second run.

4.1.2 Mapping

While performing SLAM during the experiment, it was possible to accurately map out the environment. When following a lidar-based approach, the mapping was vulnerable to accumulated deviations in the vertical plane, as well as to pitch and roll offsets. Such an occurrence can be seen in Fig. 4.4.

Furthermore, performing a longer run enabled the mapping of the coastline of the first experiment, which is displayed in a top-down view in Fig. 4.5. This particular mapping was obtained from the lidar-only based approach over a 332.5 s recording, evaluated between frames 4,250–7,575.

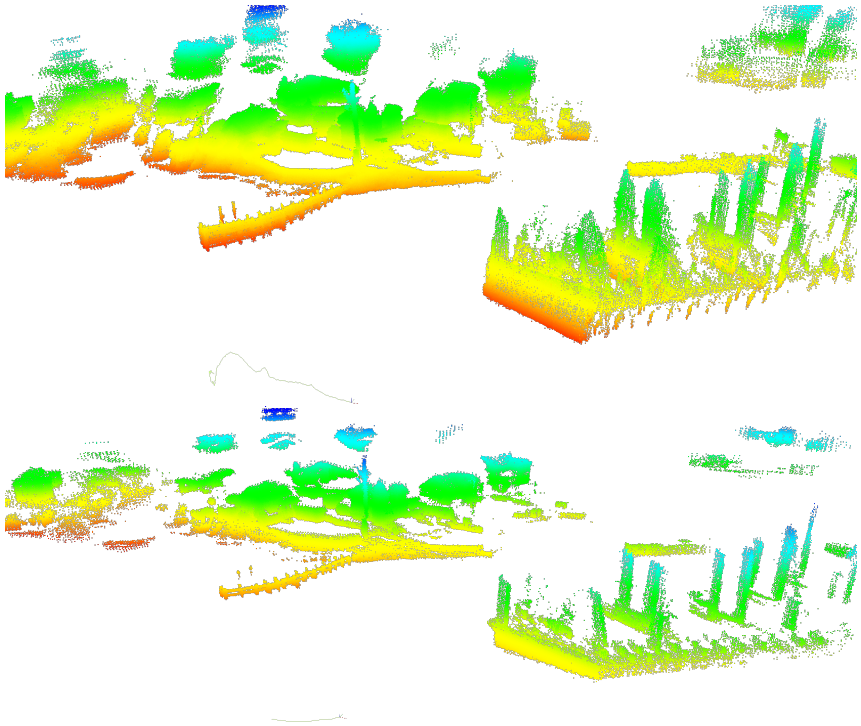

Figure 4.4: Upper figure: Pointcloud registration with NDT as basis.
Lower figure: Pointcloud registration with UKF as basis. The point clouds are viewed diagonally from the side, hence illustrating the stacking effect of points in the upper figure.

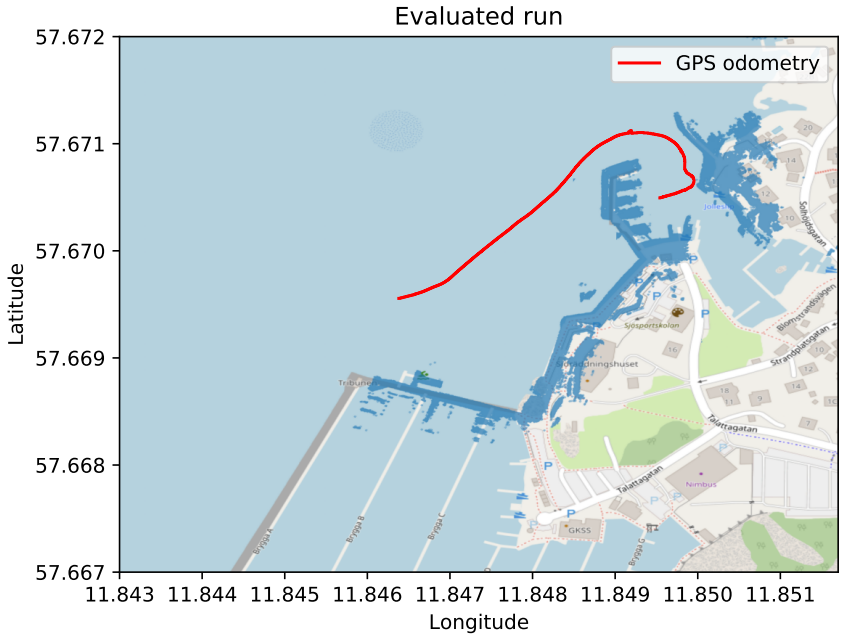

Figure 4.5: Top view of the registered point cloud aligned with the corresponding coastline. The red curve shows the path that was traversed. The mapping was performed with the lidar-only based approach.

4.1.3 Localisation

This section presents the results of the localisation during the vessel experiment. Every plot contains data from runs following only a lidar-based SLAM, as well as with state estimation through UKF data. An important metric is the z-deviation, which displays the lidar odometry on the z-axis. Even more significant is the z-acceleration a_z =($\mathrm{m\,s^{-2}}$). It shows the changes in the z-drift. Complementary, the point size is provided for every run, which correlates with the available in each frame. While all runs can be found in the appendix, the most relevant are highlighted in this section.

4.1.3.1 First run

For all sections of the fist run, the z-acceleration proved to be significantly lower for the UKF based method, compared to a lidar-only approach. The segment from frames 4,250–5,250 has the second largest point clouds sizes of the run, and both approaches proved to be stable in terms of horizontal localisation.

During the first run, the boat turned around, starting from frame 5,600. While performing this manoeuvre, only faint features in the distance were visible to the lidar. Due to this, a dip in point cloud features can be observed in Fig. 4.7. While the NDT method loses the target, it was possible to track the curve with the UKF approach while maintaining a significantly lower z-deviation.

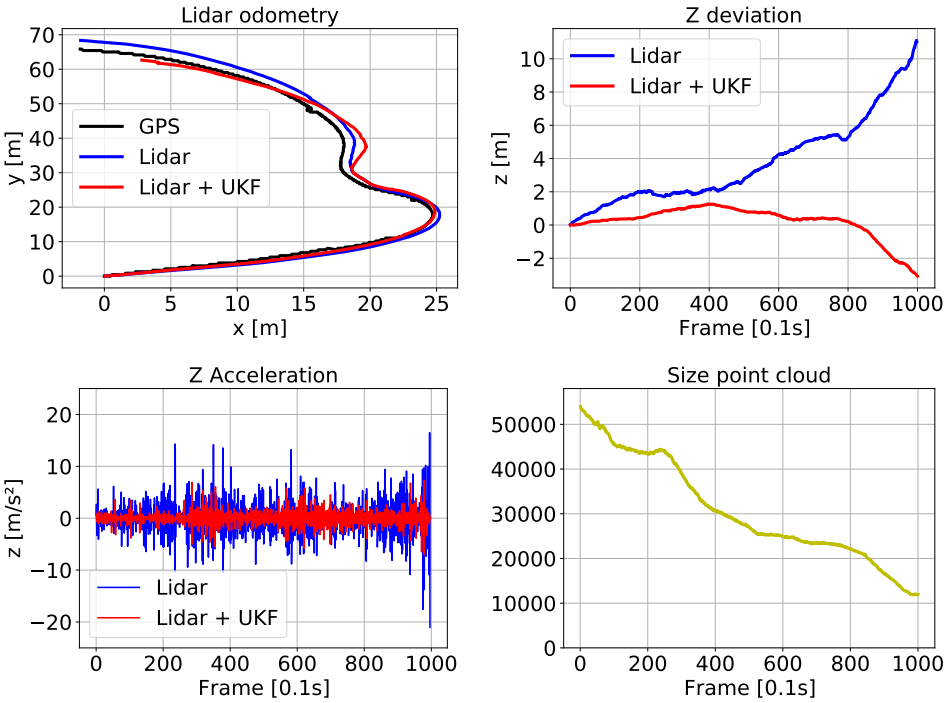

Figure 4.6: First run: frame 4,250–5,250.

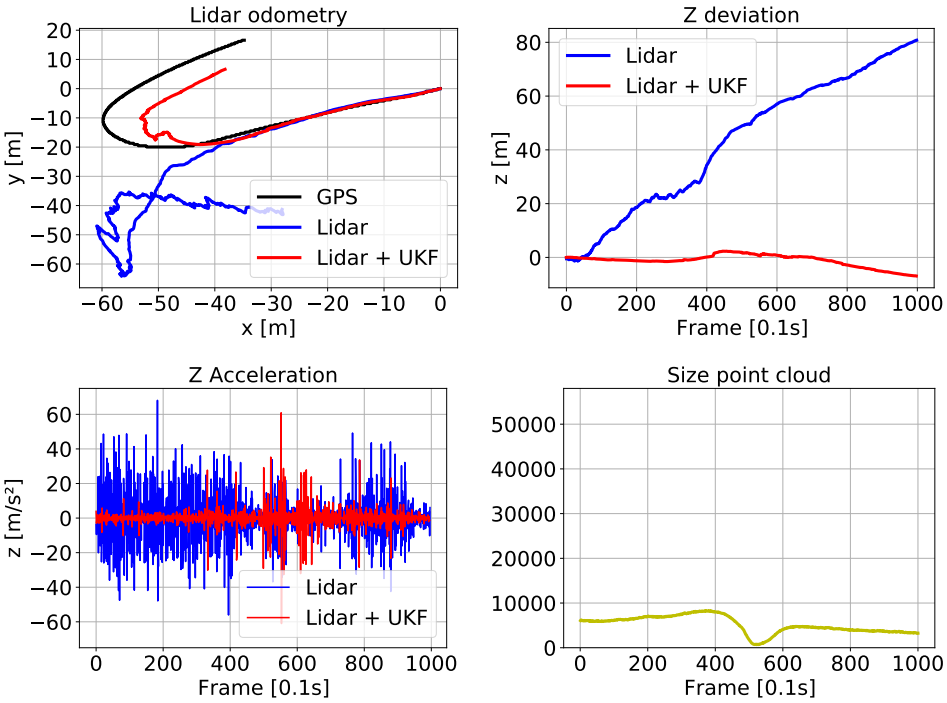

Figure 4.7: First run: frame 7,250–8,250. Note the y-axis in the bottom left plot, the spike between frame 506–507 had a range of 5,000 units.

The following two segments, displaying frames 11,250–12,250 and 15,250–16,250 illustrates scenarios where the UKF approach proved to be much more stable in

regards to the vertical deviation, but faced issues accurately tracking the horizontal trajectory.

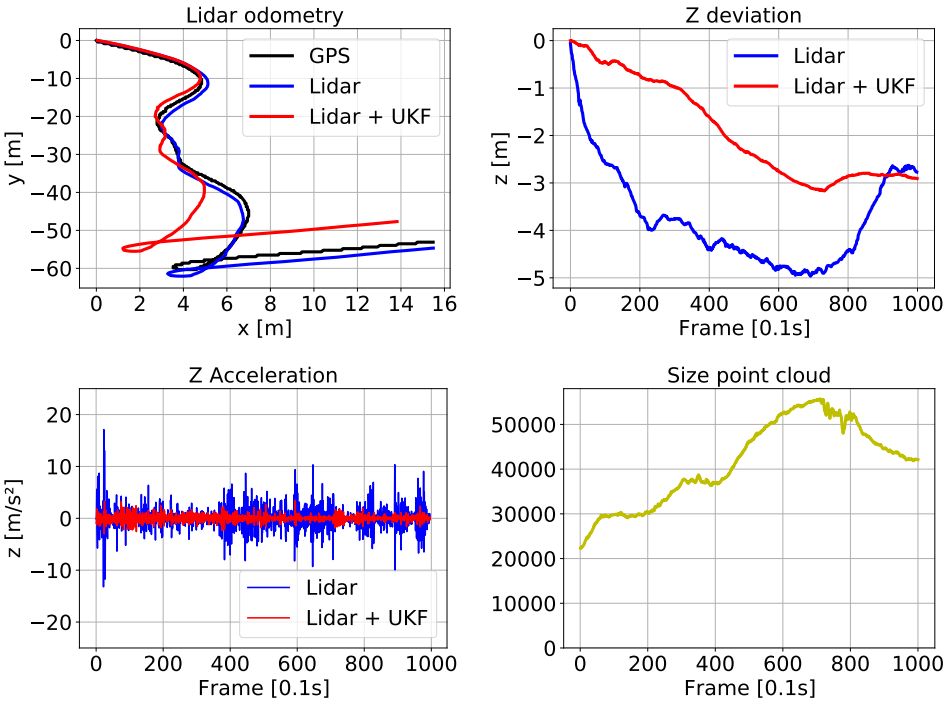

Figure 4.8: First run: frame 11,250–12,250. Note that, in the bottom left plot, the two spikes between frame 700–800 had a range of 300 units, respectively.

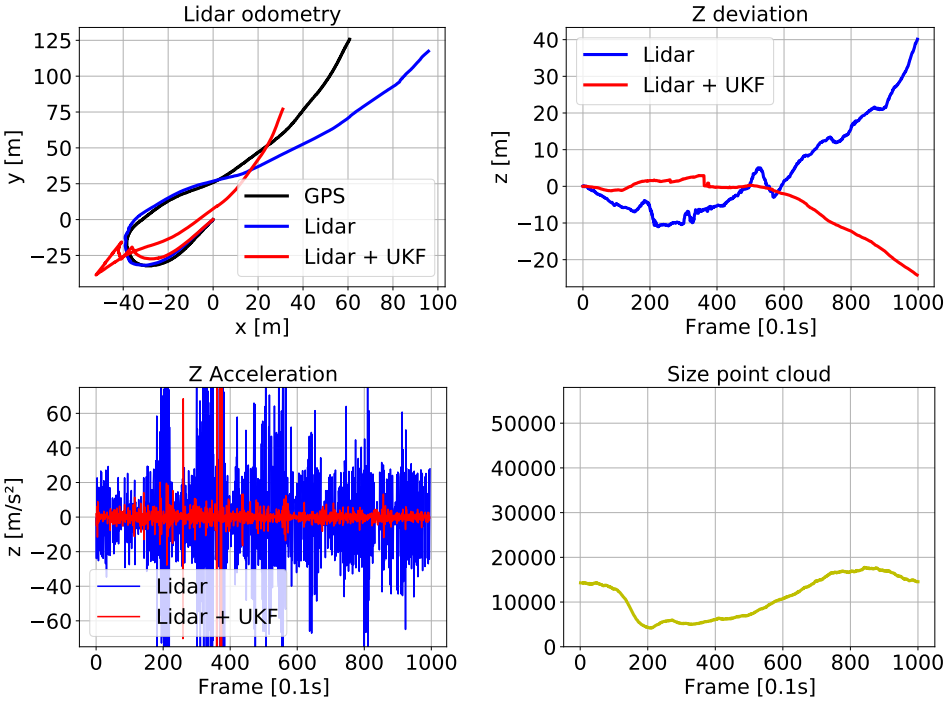

Figure 4.9: First run: frame 15,250–16,250.

For further analysis of each methods performance, both methods are compared by the RMSE between lidar-odometry and GNSS, range of z-deviation, variance of the z-acceleration and point cloud size. Tables 4.3 and 4.4 list the metrics with bold values corresponding to better performance. In general, the UKF approach delivered a much more stable z-acceleration and z-range. With regards to the horizontal localisation, the lidar-only based approach usually performed better.

Frame	RMSE Lidar vs GPS	Range [m] z-deviation	Variance z-acceleration	Average PCL size
4,250–5,250	**1.34**	11.09	12.3	30,646
5,250–6,250	**8.76**	20.47	187.4	11,476
6,250–7,250	**28.67**	15.60	530.1	5,987
7,250–8,250	34.00	82.16	228.9	5,157
8,250–9,250	**34.91**	**36.36**	2,786.3	4,909
9,250–10,250	**9.39**	43.65	260.0	10,350
10,250–11,250	**24.62**	49.56	3,470.5	10,439
11,250–12,250	**1.48**	4.90	6.4	41,517
12,250–13,250†	1.99.	11.71	15.4	25,279
13,250–14,250	**29.44**	19.21	423.3	8,023
14,250–15,250	**6.67**	**77.75**	194.0	14,542
15,250–16,250	**16.37**	51.15	1,243.1	11,023
16,250–17,250	19.21	**41.85**	615.7	7,283
17,250–18,250	**11.02**	68.30	1,167.3	11,435

† This sequence failed to register data for the UKF algorithm.

Table 4.3: Lidar-only based registration performance for the first run.

Frame	RMSE Lidar vs GPS	Range [m] z-deviation	Variance z-acceleration	Average PCD size
4,250–5,250	2.45	**4.35**	**2.3**	30,646
5,250–6,250	20.58	**10.39**	**7.4**	11,476
6,250–7,250	39.77	**3.13**	**11.4**	5,987
7,250–8,250	**6.83**	**9.23**	**40.8**	5,157
8,250–9,250	70.43	40.83	**31.3**	4,909
9,250–10,250	10.41	**6.49**	**13.5**	10,350
10,250–11,250	32.98	**11.96**	**55.6**	10,439
11,250–12,250	4.12	**3.17**	**0.6**	41,517
13,250–14,250	48.99	**19.85**	**11.2**	8,023
14,250–15,250	34.43	82.58	**25.2**	14,542
15,250–16,250	38.35	**27.12**	**212.8**	11,023
16,250–17,250	49.40	84.70	**112.7**	7,283
17,250–18,250	21.28	**20.25**	**24.1**	11,435

Table 4.4: UKF based registration performance for the first run.

4.1.3.2 Second run

This section shows the results for the second run. As most data parts of the run contained much sparser point clouds than the first run, only two sections are further highlighted here. This is because the lidar-only approach fails entirely to follow the actual trajectory in case of very sparse point clouds.

The part from frames 6,000–7,000 contained almost no features, as the boat was quite far away from the coastline, see Fig. 4.10. It can be seen that the lidar-only based approach had issues correlating these and failed to follow any trajectory, while the UKF method was able to follow the path for larger parts of the run. It is evident that the vertical drift was clearly worse for the lidar-only based approach.

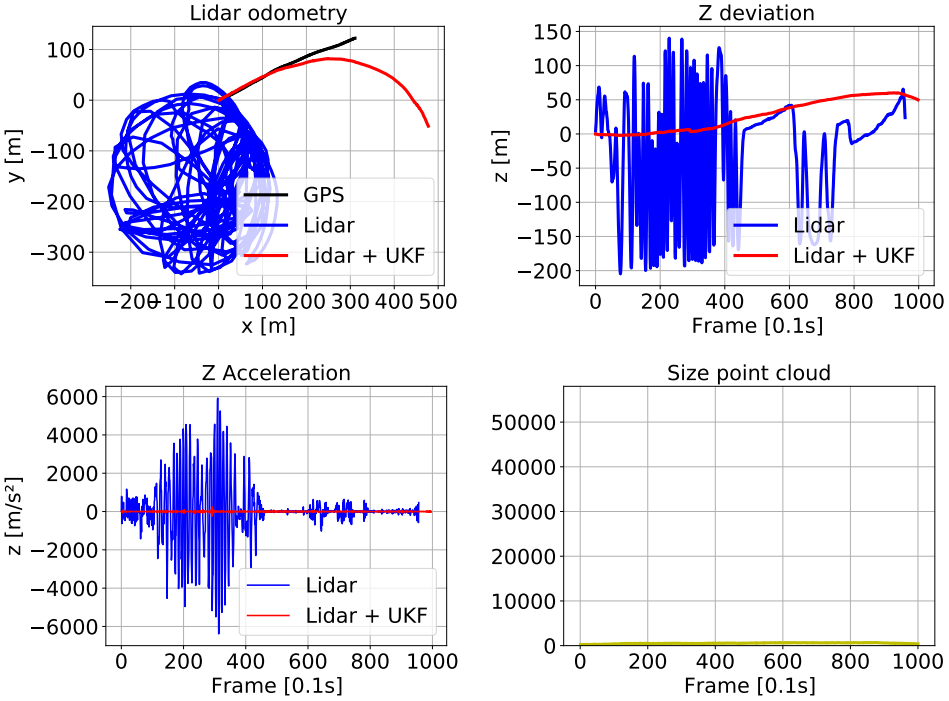

Figure 4.10: Second run: frame 6,000–7,000. Note the plot in the bottom right, the point cloud size is immensely less compared to the other figures.

From frames 12000–13000, the boat was performing a prolonged left turn. While the localisation using the UKF data was worse, it can be seen that the deviation correlates with the x-axis, implicating a global frame offset.

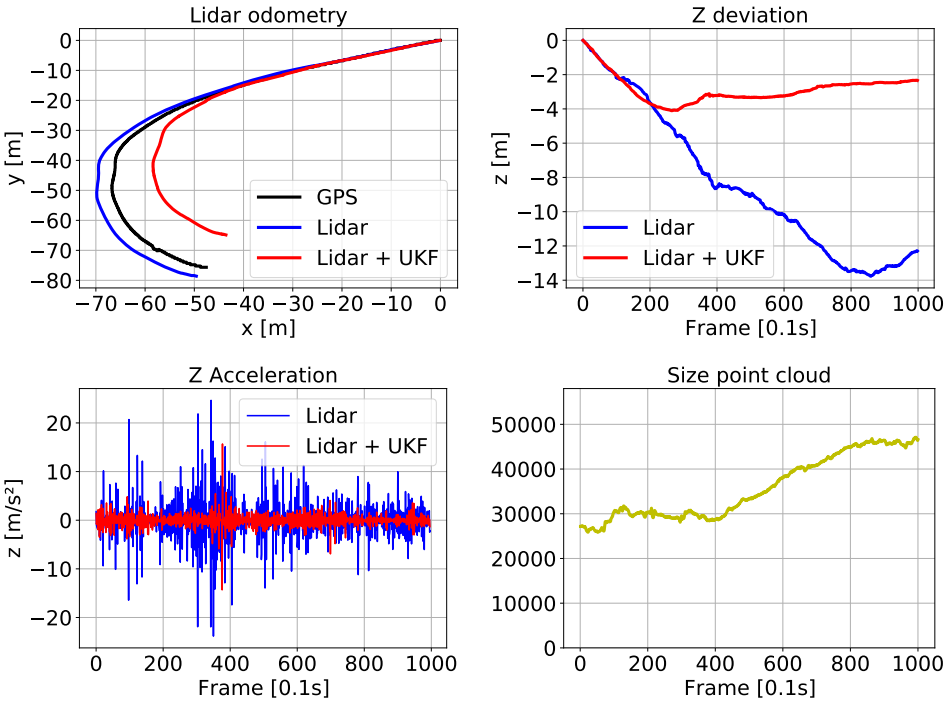

Figure 4.11: Second run: frame 12,000–13,000.

Analogue to the first run, the metrics of both methods are listed in Table 4.5 and 4.6. As most of the runs had very few features, the RMSE is higher. It can be seen that the UKF approach performed better on almost all metrics.

Frame	RMSE Lidar vs GPS	Range [m] z-deviation	Variance z-acceleration	Average PCL size
0–1,000‡	-	-	-	149
1,000–2,000‡	-	-	-	380
2,000–3,000	240.39	502.98	$2.82 \cdot 10^6$	601
3,000–4,000‡	-	-	-	708
4,000–5,000	390.71	369.28	$6.39 \cdot 10^4$	535
5,000–6,000‡	-	-	-	213
6,000–7,000	302.52	344.99	$2.00 \cdot 10^6$	536
7,000–8,000	281.63	683.57	$3.41 \cdot 10^6$	307
8,000–9,000	155.13	284.00	7,112.0	2,576
9,000–10,000	**121.23**	152.85	$2.02 \cdot 10^4$	3,902
10,000–11,000	130.92	96.77	1,472.7	1,567
11,000–12,000	**66.65**	29.32	445.62	9,334
12,000–13,000	**3.38**	13.75	16.7	35,753

‡ These sequences failed to register data for both the lidar-only and UKF algorithm.

Table 4.5: Lidar-only based based registration performance for the second run.

Frame	RMSE Lidar vs GPS	Range [m] z-deviation	Variance z-drift	Average PCL size
2,000–3,000	**206.36**	**174.91**	**1,320.6**	601
4,000–5,000	**234.04**	**191.7**	**471.0**	535
6,000–7,000	**116.45**	**62.38**	**483.3**	536
7,000–8,000	**114.79**	**241.36**	**3,838.0**	307
8,000–9,000	**99.67**	**94.00**	**583.2**	2,576
9,000–10,000	182.69	**40.32**	**205.9**	3,902
10,000–11,000	**58.29**	**35.38**	**131.6**	1,567
11,000–12,000	191.34	**28.31**	**39.6**	9,334
12,000–13,000	10.80	**4.10**	**2.0**	35,753

Table 4.6: UKF based based registration performance for the second run.

4.2 Gimbal experiment

The experiment in the laboratory environment, utilising a gimbal setup was analysed in the same manner as prior. It is important to note that the runs had slightly different lengths. The recording using pitch and roll correction is referred to as the third run, while the logging using artificially introduced roll and pitch movement is referred to as the fourth run. As can be seen, the range of z-movement, as well as the z-acceleration are significantly lower for the stabilised run. As this run did not contain any GNSS data, there is no comparison to the ground truth. The localisation will rather be evaluated on the basis of the z acceleration.

4.2.1 Mapping

Although both approaches were able to map out the environment, the stabilised approach resulted in a preciser registration compared to the simulated vessel movement run. As can be seen in Fig. 4.12, the unstabilised run had many occurrences of smearing. A difference in some captured elements of the laboratory is present between the two figures as the unstabilised run observed a broader view of the environment due to the rotational movement.

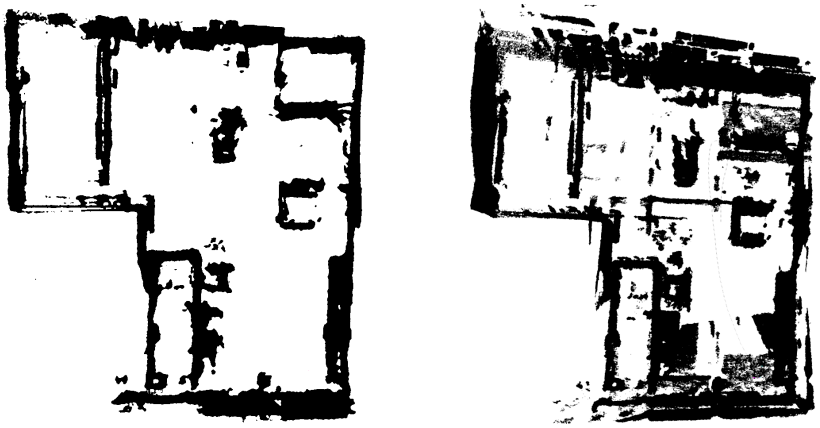

Figure 4.12: Left: Top-view mapping of the stabilised run.
Right: Top-view mapping of the unstabilised run.

4.2.2 Localisation

The localisation of the gimbal proved to be quite stable as expected. With the z-acceleration orders of magnitudes lower and a range of below 10 centimeters, it clearly outperformed the unstabilised run.

Run	Range [m] z-deviation	Variance z-acceleration	Average PCL size
Stabilised	0.09	$5.7 \cdot 10^{-5}$	193,687
Unstabilised	1.39	2.0	200,084

Table 4.7: Registration performance for stabilised and unstabilised runs

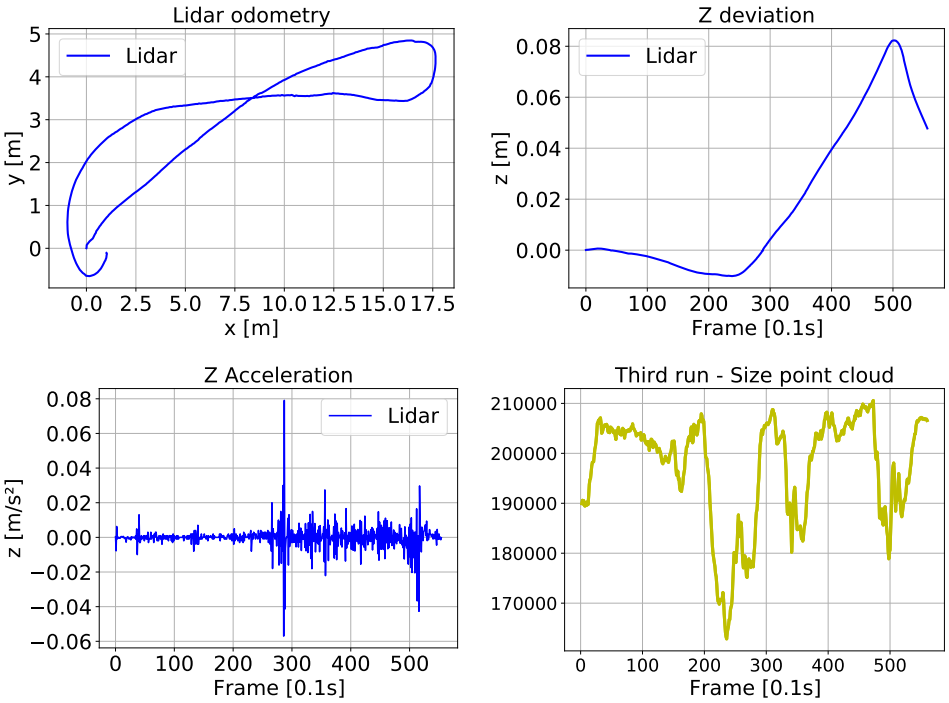

Figure 4.13: Third run: frame 0-559.

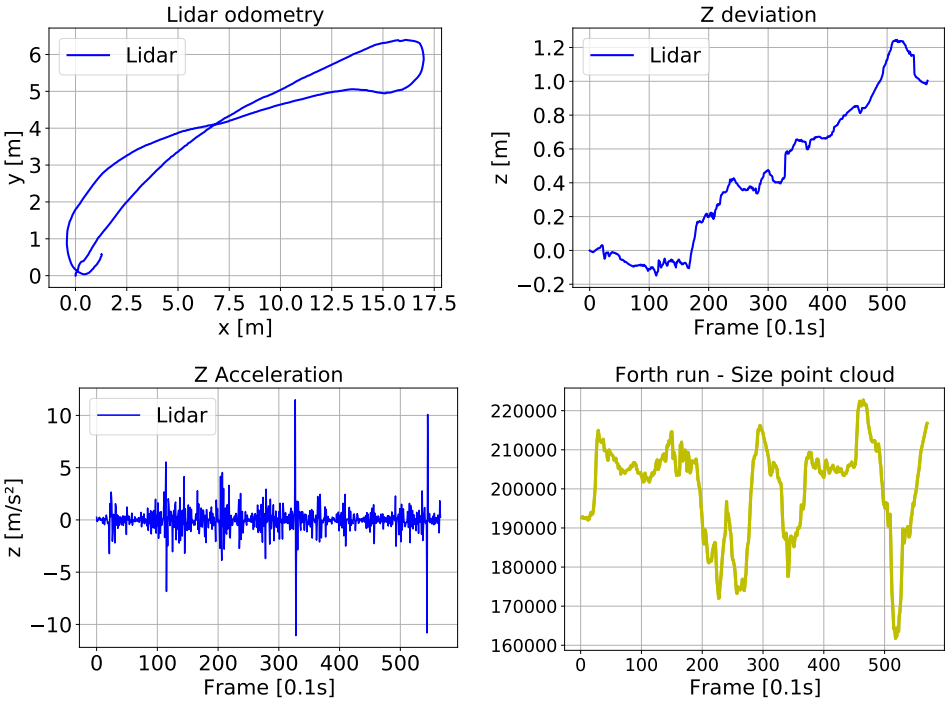

Figure 4.14: Forth run: frame 0-569. Note the y-axis for the top right and bottom left plot compared to Fig. 4.13.

4.3 SLAM requirements

The following figures illustrate how the different metrics from Sect. 4.1 are correlated to the point cloud size and velocity.

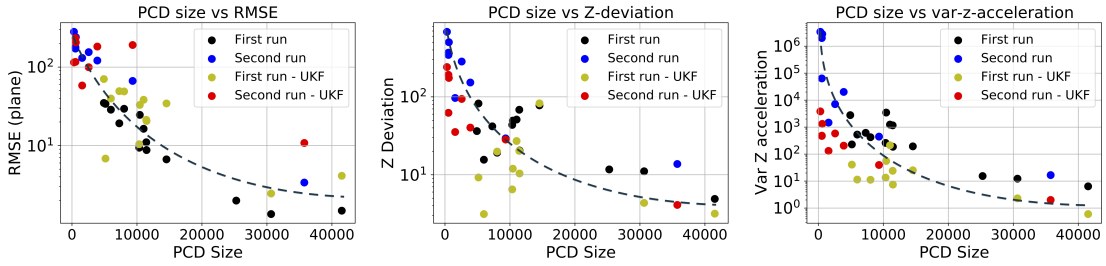

Figure 4.15: The left panel illustrates how the RMSE changes with different point cloud sizes. Middle panel illustrates how the Z-deviation changes with different point cloud sizes. The right panel illustrates how the Variance-of-Z-acceleration changes with different point cloud sizes. The y-axis is set in log-scale. The dotted line display the fitted exponential function for each dataset.

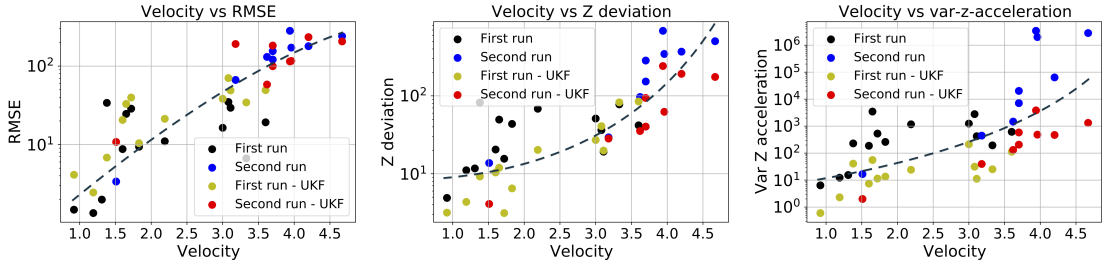

Figure 4.16: The left panel illustrates how the RMSE changes with different point cloud sizes. Middle panel illustrates how the Z-deviation changes with different point cloud sizes. The right panel illustrates how the Variance-of-Z-acceleration changes with different point cloud sizes. The y-axis is set in log-scale. The dotted line display the fitted exponential function for each dataset.

It was observed from Fig. 4.15 that the RMSE, z-deviation and the variance of the z-acceleration decreased as the point cloud size increased. The algorithm that only utilised lidar data seemed to yield lower RMSE compared to the algorithm that included the UKF, while the z-deviation and variance of the z-acceleration were higher.

It was observed from Fig. 4.16 that the RMSE, z-deviation and the variance of the z-acceleration increased as the velocity of the vessel increased. There was no evident advantage of any of the algorithms with regards to the RMSE. However, the UKF-based algorithm seemed to yield lower z-deviation and variance of the z-acceleration in comparison to the lidar only.

5
Discussion

The discussion regarding research questions 1 and 2 about deviations in the vertical plane is detailed in Sect. 5.1 and 5.2. Furthermore, it is divided into mapping, localisation, and general performance of the sensor fusion. The findings of which requirements are needed for SLAM, regarding research question 3 are discussed in Sect. 5.4. The rest of the discussion is focused on the social and ethical impacts in Sect. 5.5 and possible future work on the project in Sect. 5.6.

5.1 UKF performance

The trajectory resulting from the sensor fusion to track the kinematic state of the vessel performed well for most sections of the logging, as can be seen in Table. 4.1 and Fig. 4.3. Occasionally, the offset is slightly larger due to the discrete nature of the GNSS data. As the data is coming in at 20 Hz and does not have the same fidelity as the UKF, there will always be a slight difference between the two trajectories. As the logging sessions used a minimalistic sensor setup, with only an IMU and a GNSS with one antenna, there was no data available for tracking the z-deviation like an altimeter. Therefore, the predicted z-position was always close to zero. Accurate tracking of our kinematic state is important for evaluating the performance of the registration in the next step.

As the platform used a 6-DOF IMU, the initial global coordinate system is unknown. This is the case as the rotational axes can only be measured using a magnetometer. Therefore, the discussion and evaluation of the registration are focused on whether the mapping and localisation follow a constant drift in the vertical plane, which equals to a constant rotational offset between the coordinate systems of the registration and the coordinate system of the UKF, which assumes no pitch or roll offset.

5.2 SLAM

To evaluate the effectiveness of the SLAM algorithm, it was important to take into account both the localisation and mapping parts. One could follow the UKF trajectory by heart and make a case for a good localisation. This will however result in worse point cloud registration, as the sensor-fusion on its own is neither sufficient nor reliable at all times. Nevertheless, it was occasionally possible to receive a good registration for a short time, albeit with terrible localisation. A

good visual indicator for this was the combined point cloud registration and the three-dimensional trajectory, as it combines the two aspects.

5.2.1 Mapping

To establish a baseline, the lidar-only method was used for registration. Although this method performs well for a short period of time, it does accumulate deviations, especially in pitch and vertical movement. The effect of this can be seen in Fig. 4.4a. This smearing effect in the vertical plane has negative impacts on the details and accuracy of the resulting point cloud and affects any further usage of the data. As the lidar-only based approach does not use any knowledge about the kinematic state of the platform, it establishes its guess purely on the comparison of the current and previous frame. This can be vulnerable to changes in features, as elements in the point cloud enter or leave the frame. Aligning point clouds with different elements will result in small errors that accumulate over time.

The software-based approach delivered less stacked point cloud data, as can be seen in Fig. 4.4b. Provided that the sensor data was accurate and sufficient, the first transformation step in the registration yielded a better approximation of the current frame. This is especially evident with respect to shifts in the vertical plane. As the UKF algorithm provides a more accurate estimate of the translation and rotational movement, the ICP algorithm converges to transformations that produce less stacking of features in the final point cloud. Occasionally, the UKF method accumulated some offsets in the horizontal plane. This usually occurred in cases where the filtering under-performed and was bound to the limited sensor setup. It is assumed that this will improve with a more advanced setup, namely with an attitude sensor and a GNSS system with a second antenna. This will improve the estimation of the heading and z-coordinate and allows for a preciser registration during the ICP step.

Looking at the data from the laboratory platform, the use of a stabilisation resulted in a much more accurate point cloud compared to the run with introduced kinematic movement. This ties together with the assumption that the NDT algorithm will fail in situations with changing available features. As more objects in the lab are exposed during the course of the run, offsets in the trajectory resulted in a worse overall mapping. This does not occur in the gimbal-stabilised run, as the rotational movements are estimated to be zero, thus resulting in preciser transformation preceding the ICP part.

5.2.2 Localisation

Taking the NDT method as a baseline, the algorithm was able to follow the horizontal plane fairly well. As the method depends on the number of available points, it failed to follow the actual trajectory in case of declining point cloud sizes and changes in the point cloud. This results in huge deviations in all dimensions. This is especially evident in the section from frames 7,250–8,250 in the first run as can be seen in Fig. 4.7. As the point cloud size dips to less than 1,000 points, the lidar odometry starts drifting at the wrong direction and cannot keep track of the

curve of the trajectory.

Using the UKF data, on the other hand, resulted in a better lidar-odometry in situations where the features alone were not enough to track it otherwise. The current heading was able to provide much better input for the ICP algorithm in these cases, as can be seen in Fig. 4.7.

Observing the vertical acceleration a_z, it can be concluded that the UKF approach was much more stable in terms of vertical deviation. While a constant drift can still occur due to offsets in the coordinate systems, the usual stable acceleration proved that the lidar-odometry was moving mostly in a single plane and had rarely actual offsets in the orthogonal movement. This correlation to the coordinate offset is especially evident in Fig. 4.8, since the vertical positions of frame 620 and 800 are similar as the vessel returns to its previous position at (x, y) = (3 m, -52 m).

For the gimbal setup, the lidar-odometry could not be compared to the ground truth trajectory as it was not possible to record with the full sensor platform. However, comparing the vertical movement clearly shows that the complex kinematic movement is handled with high accuracy, as the range of below 10 cm and the acceleration is orders of magnitude lower compared to the unstabilised approach. The introduced artificial movement in pitch and roll for the other run resulted in many changes regarding the feature and produced much more deviations, and thus performed much worse.

5.3 Hardware versus software solutions

Comparing hardware and software solutions is not trivial, as they were performed under slightly different conditions. During the recording sessions on the vessel, there were additional sensors present that provided ground truth to the movement in the horizontal plane. As the laboratory logs did not contain GNSS, this was not possible. It is nevertheless possible to analyse the performance of the registration and vertical movement.

The biggest advantage with the gimbal setup is that the movement in roll and pitch are negligible due to the correction from the motors. This results in the reduction of the kinematic equation from six to four dimensions. Looking at the ICP algorithm, a lower dimensionality can improve performance and computation speed in live situations.

A filtered state from the UKF has likewise been proven to improve lidar odometry compared to an NDT approach. This is due to the shortcomings of the NDT method. It was also able to follow paths better with few points in the available point cloud. Although this was not tested with the gimbal, it is assumed to perform worse as it is still using the NDT algorithm.

It can be noted that a combination of both hardware and software solutions can be beneficial in performing SLAM on a vessel as they can deal with different situations. While the gimbal approach can improve computation time and eliminates offsets in pitch and roll, the UKF method can deal with smaller point clouds and keep track of the trajectory with better accuracy.

5.4 SLAM requirements

When analysing the deviations in the vertical and horizontal planes, it was evident that there was some correlation between the number of available points and the error of the registration trajectory. As Fig. 4.15 illustrates, there is a clear connection between these metrics. As explained in previous sections, this stems mainly from the shortcomings of the NDT method. Furthermore, it can be concluded that data-logging sessions are vulnerable to the recording environments. Especially in situations where the vessel needs to turn around and is mostly bound to face water for an extended amount of time, neither of the proposed algorithms can track the odometry accurately. There is room for further improvements, where the trajectory is set entirely to the UKF movement in case of a point cloud size below a certain threshold.

Another correlated metric was the vessel velocity. As a higher speed results in larger differences in the point cloud, it also affects the performance of the registration. In Fig. 4.16, it can be observed that a lower average velocity generally results in lower z-accelerations. This is especially evident at speeds above $3.5\,\mathrm{m\,s^{-1}}$.

With regard to lidar settings, this presents a trade-off between the lidar frequency and the step increment. As the Ouster OS-2 comes with the option for operation in either 10 or 20 Hz, as well as capturing from 512 to 2,048 horizontal beams, it could be beneficial to adapt these settings according to the recording situation. In the case of many features in the environment, it is generally advised to use a higher lidar frequency, compared to situations with less information. Using a smaller horizontal increment might improve the registration for this case.

5.5 Social and ethical impact

While SLAM is only a part of a wide array of systems in a self-driven vehicle, it forms an important part in enabling safer automatic manoeuvres and estimating distances. Even though the first fully autonomous vehicles seem to be available in the near future, in the case of vessels, there is still lots of work left.

As with all emerging technology, there are certain risks and possibilities associated with autonomous vessels. On the upside, it can reduce emissions with steering algorithms that learn to adapt to weather and currents. Furthermore, it can increase the safety of crew members, with monitoring systems and optimised behaviour in complex situations. As crew members face high-risk rates at their job, this can both increase operational safety and prevent accidents with massive economic and environmental implications. A core issue associated with the emergence of autonomous vessels is the threat of a cyber-attack. As more of the control and steering is relegated to autonomous systems, these can be vulnerable to hacks and attacks on information systems. This could result in the hijacking of the ship or even an intentional accident. As more responsibility is given to automation, the issue of cyber-security becomes equally as important.

5.6 Future work

Possible future work in regards to this project can be related to the combination of a hardware and software solution. As both proposed methods had different benefits, it might be interesting to investigate the combined effect of both systems being used simultaneously. As a gimbal can be easily installed on a tripod, this should not pose any complex setup and can easily be integrated into the previously used sensor platform.

In addition, to further improve performance, the platform can be enhanced with additional sensors. As mentioned above, the use of an altimeter can provide a reliable height measurement that can be used for the state estimation. Another improvement would be the use of a second antenna for the GNSS system to enhance stability and provide accurate heading measurement.

With these improvements in place, further research can for example be performed on loop closing or handling dynamic objects. As small errors accumulate over time and the vessel returns to a previous position, the point cloud cannot be easily aligned. This opens a domain of research for handling these situations. Dynamic objects are for example other boats or animals that move relative to the environment. Handling those situations is important for mapping, as these objects will clog up the resulting point cloud because their position changes for every frame and they will be added to every point cloud frame.

6
Conclusion

This project investigated the performance of two different SLAM algorithms in a marine environment, while using point cloud data from a lidar scanner. The first algorithm only used lidar data while performing NDT and ICP, while the second algorithm used lidar data together with IMU and GNSS data through the implementation of a UKF. Two experiments were performed, each investigating a software and hardware-based solution compared to the lidar-only based approach. It can be concluded that both of the proposed methods that implemented an UKF algorithm and a gimbal, have shown to reduce deviations in the vertical plane compared to a lidar-only based approach. Although the baseline method, using only a lidar, performed well during shorter time periods, the differences were evident when looking at longer runs, as was done in this research. The software-based approach, using a UKF state estimation, has proven to perform better in the key metric, the z-acceleration. It also resulted in less smearing of the point clouds. Including a gimbal stabilisation indicated to reduce the z-acceleration significantly, as the point cloud registration performs better due to more precise alignment of consecutive frames. When comparing the two methods, it was concluded that both have different benefits. The UKF state estimation was able to follow the actual sensor trajectory to some degree in cases where the point clouds were quite sparse. This was not possible with the baseline NDT algorithm, which is also used for the gimbal approach. The stabilised solution, on the other hand, offered the benefit of reducing the transformation problem between frames from six to four dimensions, as the pitch and roll movements are constrained. Furthermore, the gimbal stabilisation can easily be part of a tripod setup and eliminates the need for an external IMU. Both approaches can also be combined to benefit from both solutions.

While investigating the minimal conditions for performing a reliable localisation and mapping, it was clear that the biggest indicator was the point cloud size. A clear correlation was shown between the z-acceleration and the RMSE, between the calculated trajectory and the ground truth. Another impacting factor was the velocity, as it correlates with the difference in point cloud frames between registration. To perform SLAM with the proposed algorithms, it is therefore important to capture as many points as possible with the lidar. This should be kept in mind when thinking about any visual constraints on the sensor platform, route planning, and sensor frequency. The performance is also assumed to improve when using additional sensors like an altimeter for vertical position logging, a second antenna for preciser heading data, and a 9-DOF IMU for absolute rotational positions. Future work on the issue could be centred around

how to deal with situations with very sparse point clouds and implementing loop-closing.

Bibliography

[1] Chaiwoo Lee et al. "Consumer Knowledge and Acceptance of Driving Automation: Changes over Time and across Age Groups". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 65. 1. SAGE Publications Sage CA: Los Angeles, CA. 2021, pp. 1395–1399.

[2] Grand View Research. *Autonomous Vehicle Market Size, Share & Trends Analysis Report By Application (Transportation, Defense), By Region (North America, Europe, Asia Pacific, South America, MEA), And Segment Forecasts, 2021 - 2030*. 2020. URL: `https://www.grandviewresearch.com/industry-analysis/autonomous-vehicles-market`.

[3] Margarita Martınez-Dıaz and Francesc Soriguera. "Autonomous vehicles: theoretical and practical challenges". In: *Transportation Research Procedia* 33 (2018), pp. 275–282.

[4] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous localization and mapping: part I". In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110.

[5] Udo Frese. "Interview: Is slam solved?" In: *KI-Künstliche Intelligenz* 24.3 (2010), pp. 255–257.

[6] Sammy Omari et al. "Dense visual-inertial navigation system for mobile robots". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 2634–2640.

[7] Brian Clipp et al. "Parallel, real-time visual SLAM". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 3961–3968.

[8] Junyao Guo, Unmesh Kurup, and Mohak Shah. "Is it safe to drive? an overview of factors, metrics, and datasets for driveability assessment in autonomous driving". In: *IEEE Transactions on Intelligent Transportation Systems* 21.8 (2019), pp. 3135–3151.

[9] Hang Yin and Christian Berger. "When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–8.

[10] Eric A Wan and Rudolph Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee. 2000, pp. 153–158.

[11] N. Viandier et al. “Gnss Performance Enhancement in Urban Environment Based on Pseudo-range Error Model”. In: *2008 IEEE/ION Position, Location and Navigation Symposium*. 2008, pp. 377–382. DOI: 10.1109/PLANS.2008.4570093.

[12] Ola Benderius, Christian Berger, and Krister Blanch. “Are we ready for beyond-application high-volume data? The Reeds robot perception benchmark dataset”. In: *arXiv preprint arXiv:2109.08250* (2021).

[13] Zhifeng Liu. “Performance evaluation of stereo and motion analysis on rectified image sequences”. In: (2007).

[14] Piotr Dollár et al. “Pedestrian detection: A benchmark”. In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE. 2009, pp. 304–311.

[15] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. “Semantic object classes in video: A high-definition ground truth database”. In: *Pattern Recognition Letters* 30.2 (2009), pp. 88–97.

[16] *Reeds Project*. https://reeds.opendata.chalmers.se. Accessed: 2022-02-23.

[17] Tixiao Shan et al. “LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 5135–5142.

[18] Tixiao Shan and Brendan Englot. “LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4758–4765.

[19] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for autonomous driving? the kitti vision benchmark suite”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361.

[20] *Daimler pedestrian benchmark dataset*. http://www.gavrila.net/Datasets/Daimler_Pedestrian_Benchmark_D/daimler_pedestrian_benchmark_d.html. Accessed: 2022-03-02.

[21] Liangyu Wang and Varun Ganapati Hegde. “Mapping and 3D reconstruction based on lidar”. In: (2021).

[22] Bogdan Iancu et al. “Aboships-an inshore and offshore maritime vessel detection dataset with precise annotations”. In: *Remote Sensing* 13.5 (2021), p. 988.

[23] Dilip K Prasad et al. “Video processing from electro-optical sensors for object detection and tracking in a maritime environment: a survey”. In: *IEEE Transactions on Intelligent Transportation Systems* 18.8 (2017), pp. 1993–2016.

[24] Zhenfeng Shao et al. “Seaships: A large-scale precisely annotated dataset for ship detection”. In: *IEEE transactions on multimedia* 20.10 (2018), pp. 2593–2604.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).

[26] Apoorva Joglekar et al. “Depth estimation using monocular camera”. In: *International journal of computer science and information technologies* 2.4 (2011), pp. 1758–1763.

[27] Shahram Izadi et al. “Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera”. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 2011, pp. 559–568.

[28] Thomas Lemaire et al. “Vision-based slam: Stereo and monocular approaches”. In: *International Journal of Computer Vision* 74.3 (2007), pp. 343–364.

[29] *Lidar vs Cameras*. `https://www.autopilotreview.com/lidar-vs-cameras-self-driving-cars/`. Accessed: 2022-03-08.

[30] *Why LiDAR is Doomed*. `https://www.voltequity.com/article/why-lidar-is-doomed`. Accessed: 2022-03-09.

[31] Axel Gern, Rainer Moebus, and Uwe Franke. “Vision-based lane recognition under adverse weather conditions using optical flow”. In: *Intelligent Vehicle Symposium, 2002. IEEE*. Vol. 2. IEEE. 2002, pp. 652–657.

[32] Zheyuan Wang, Guo Cheng, and Jiang Yu Zheng. “All weather road edge identification based on driving video mining”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–6.

[33] Unghui Lee et al. “Development of a self-driving car that can handle the adverse weather”. In: *International journal of automotive technology* 19.1 (2018), pp. 191–197.

[34] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and Elmar Wasle. *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2007.

[35] Federico Grasso Toro et al. “Particle Filter technique for position estimation in GNSS-based localisation systems”. In: *2015 International Association of Institutes of Navigation World Congress (IAIN)*. 2015, pp. 1–8. DOI: `10.1109/IAIN.2015.7352236`.

[36] *GNSS-AIDED INERTIAL NAVIGATION SYSTEM (GNSS/INS)*. `https://www.vectornav.com/resources/inertial-navigation-primer/theory-of-operation/theory-gpsins`. Accessed: 2022-03-16.

[37] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer handbook of robotics*. Vol. 200. Springer, 2008.

[38] Hugh Durrant-Whyte, David Rye, and Eduardo Nebot. “Localization of autonomous guided vehicles”. In: *Robotics Research* (1996), pp. 613–625.

[39] Martin Magnusson. “The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection”. PhD thesis. Örebro universitet, 2009.

[40] Ankit Dhall et al. “LiDAR-camera calibration using 3D-3D point correspondences”. In: *arXiv preprint arXiv:1705.09785* (2017).

[41] Yanghai Tsin and Takeo Kanade. “A correlation-based approach to robust point set registration”. In: *European conference on computer vision*. Springer. 2004, pp. 558–569.

[42] Paul J Besl and Neil D McKay. “Method for registration of 3-D shapes”. In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. Spie. 1992, pp. 586–606.

[43] Xiuying Shi et al. “The iterative closest point registration algorithm based on the normal distribution transformation”. In: *Procedia Computer Science* 147 (2019), pp. 181–190.

[44] Kaustubh Pathak et al. “Evaluation of the robustness of planar-patches based 3D-registration using marker-based ground-truth in an outdoor urban scenario”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 5725–5730.

[45] Siavash Hosseinyalamdary. “Deep Kalman filter: Simultaneous multi-sensor integration and modelling; A GNSS/IMU case study”. In: *Sensors* 18.5 (2018), p. 1316.

[46] Freek Baalbergen, Madeleine Gibescu, and Lou van der Sluis. “Modern state estimation methods in power systems”. In: *2009 IEEE/PES Power Systems Conference and Exposition*. 2009, pp. 1–6. DOI: 10.1109/PSCE.2009.4840003.

[47] Matthew Rhudy et al. “Sensitivity analysis of extended and unscented Kalman filters for attitude estimation”. In: *Journal of Aerospace Information Systems* 10.3 (2013), pp. 131–143.

[48] Shoudong Huang and Gamini Dissanayake. “A critique of current developments in simultaneous localization and mapping”. In: *International Journal of Advanced Robotic Systems* 13.5 (2016), p. 1729881416669482.

[49] Anna Petersson. “Vision-based state estimation of autonomous boats”. In: (2021).

[50] Nicola Dragoni et al. “Microservices: yesterday, today, and tomorrow”. In: *Present and ulterior software engineering* (2017), pp. 195–216.

[51] Theo Combe, Antony Martin, and Roberto Di Pietro. “To docker or not to docker: A security perspective”. In: *IEEE Cloud Computing* 3.5 (2016), pp. 54–62.

[52] *Point Cloud Library Module filters*. 2022. URL: https://pointclouds.org/documentation/group__filters.html (visited on 05/04/2022).

[53] Peter Biber and Wolfgang Straßer. “The normal distributions transform: A new approach to laser scan matching”. In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*. Vol. 3. IEEE. 2003, pp. 2743–2748.

[54] Feng Huang et al. “Point wise or Feature wise? Benchmark Comparison of Public Available LiDAR Odometry Algorithms in Urban Canyons”. In: *arXiv preprint arXiv:2104.05203* (2021).

[55] Martin Magnusson et al. “Beyond points: Evaluating recent 3D scan-matching algorithms”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 3631–3637.

[56] Berthold KP Horn. “Closed-form solution of absolute orientation using unit quaternions”. In: *Josa a* 4.4 (1987), pp. 629–642.

[57] Haggai Maron et al. “Point registration via efficient convex relaxation”. In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), pp. 1–12.

[58] Yuhong Yang, Junchuan Zhou, and Otmar Loffeld. "Quaternion-based Kalman filtering on INS/GPS". In: *2012 15th International Conference on Information Fusion*. IEEE. 2012, pp. 511–518.

[59] Michael Roth, Gustaf Hendeby, and Fredrik Gustafsson. "EKF/UKF maneuvering target tracking using coordinated turn models with polar/Cartesian velocity". In: *17th International Conference on Information Fusion (FUSION)*. IEEE. 2014, pp. 1–8.

[60] Logah Perumal. "Quaternion and its application in rotation using sets of regions". In: *International Journal of Engineering and Technology Innovation* 1.1 (2011), p. 35.

[61] E Grafarend. "The optimal universal transverse Mercator projection". In: *Geodetic Theory Today*. Springer. 1995, pp. 51–51.

[62] Christian Berger, Björnborg Nguyen, and Ola Benderius. "Containerized development and microservices for self-driving vehicles: Experiences & best practices". In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE. 2017, pp. 7–12.

[63] Ola Benderius and Christian Berger. *OpenDLV*. `https://opendlv.org/`. 2022.

[64] *Ouster Datasheet*. 2022. URL: `https://ouster.com/downloads/` (visited on 05/04/2022).

[65] OpenStreetMap contributors. *Planet dump retrieved from https://planet.osm.org*. `https://www.openstreetmap.org`. 2017.

[66] Paul Newman and Kin Ho. "SLAM-loop closing with visually salient features". In: *proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE. 2005, pp. 635–642.

[67] Mengshi Zhang et al. "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems". In: *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2018, pp. 132–142.

[68] V.C. Chen et al. "Micro-Doppler effect in radar: phenomenon, model, and simulation study". In: *IEEE Transactions on Aerospace and Electronic Systems* 42.1 (2006), pp. 2–21. DOI: 10.1109/TAES.2006.1603402.

[69] Arnold Pompos. *Why can radio waves pass through a wall but light cannot? Kernel Description*. 2022. URL: `https://www.fnal.gov/pub/science/inquiring/questions/mikep.html` (visited on 03/31/2022).

[70] Taohua Zhou et al. "MMW radar-based technologies in autonomous driving: A review". In: *Sensors* 20.24 (2020), p. 7283.

[71] Murray Rosenblatt. "Remarks on a multivariate transformation". In: *The annals of mathematical statistics* 23.3 (1952), pp. 470–472.

[72] K Somani Arun, Thomas S Huang, and Steven D Blostein. "Least-squares fitting of two 3-D point sets". In: *IEEE Transactions on pattern analysis and machine intelligence* 5 (1987), pp. 698–700.

[73] Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. "Closed-form solution of absolute orientation using orthonormal matrices". In: *JOSA A* 5.7 (1988), pp. 1127–1135.

[74] Michael W Walker, Lejun Shao, and Richard A Volz. “Estimating 3-D location parameters using dual number quaternions”. In: *CVGIP: image understanding* 54.3 (1991), pp. 358–367.

A

Appendix 1

A.1 First run

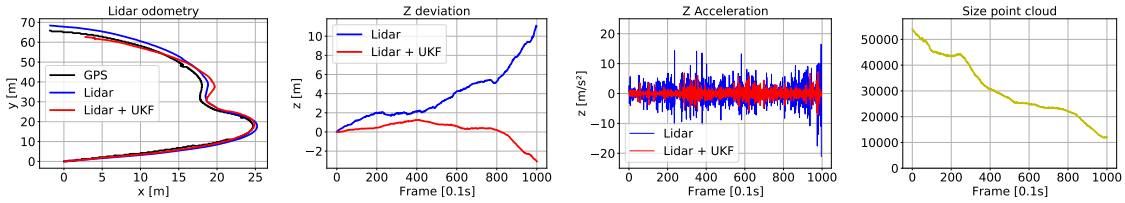

Figure A.1: First run: frame 4,250–5,250.

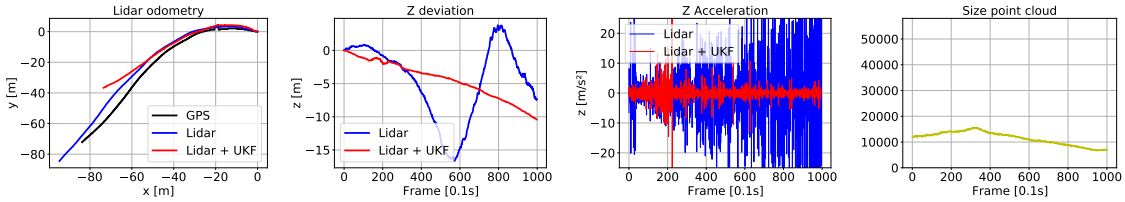

Figure A.2: First run: frame 5,250–6,250.

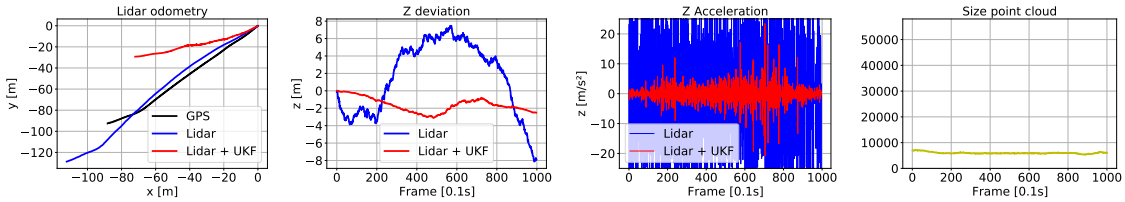

Figure A.3: First run: frame 6,250–7,250.

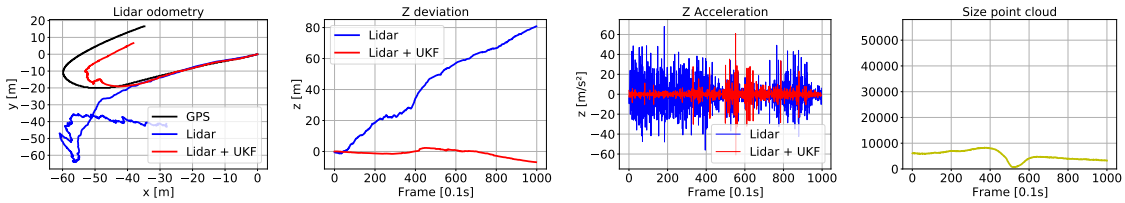

Figure A.4: First run: frame 7,250–8,250.

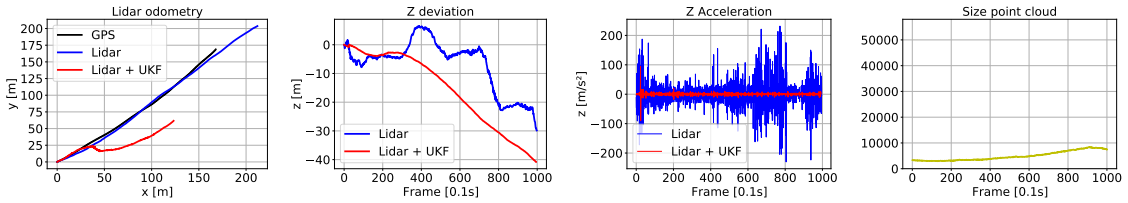

Figure A.5: First run: frame 8,250–9,250.

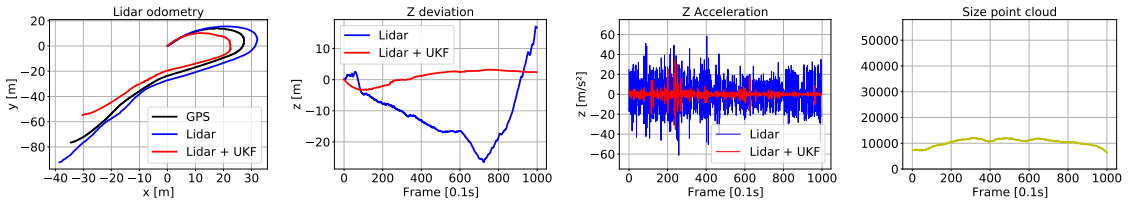

Figure A.6: First run: frame 9,250–10,250.

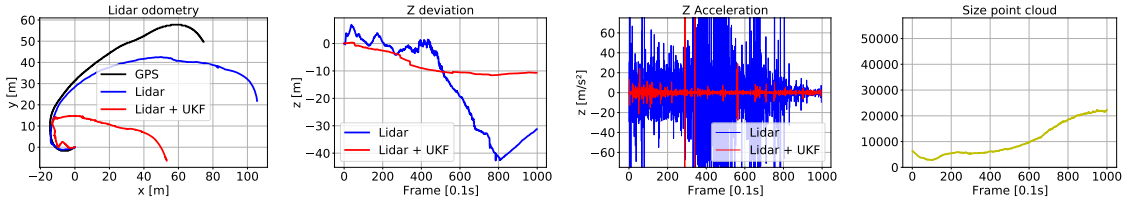

Figure A.7: First run: frame 10,250–11,250.

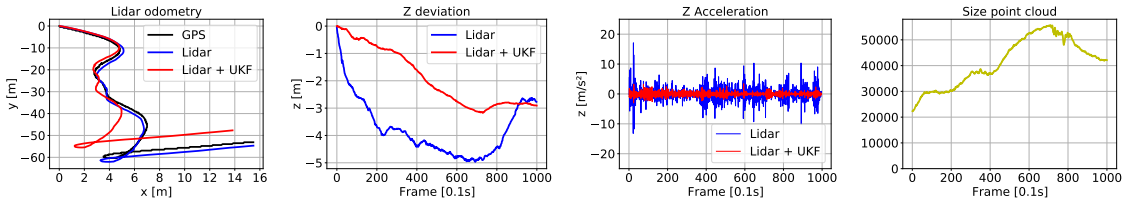

Figure A.8: First run: frame 11,250–12,250.

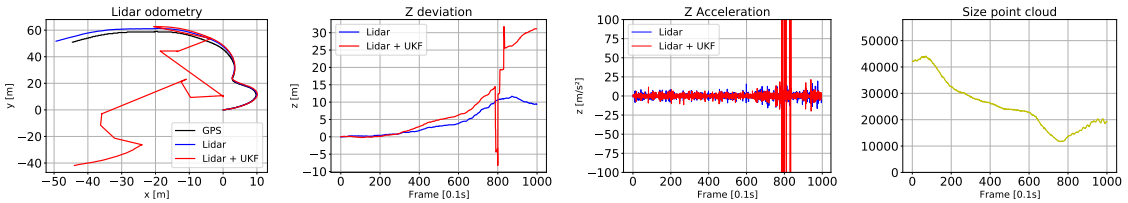

Figure A.9: First run: frame 12,250–13,250.

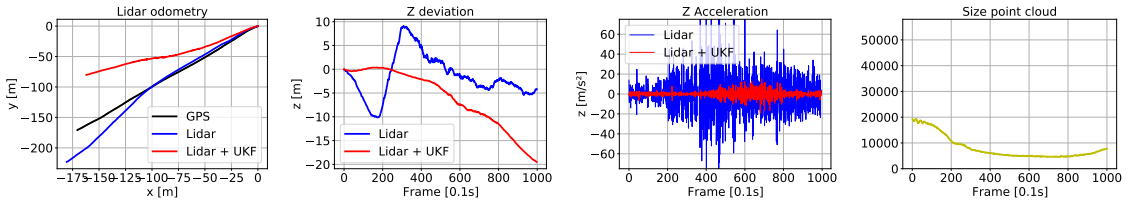

Figure A.10: First run: frame 13,250–14,250.

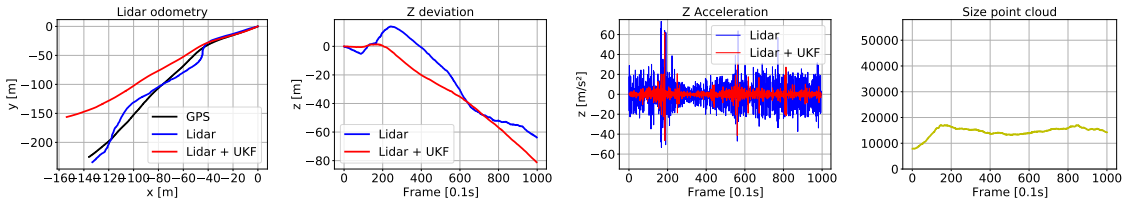

Figure A.11: First run: frame 14,250–15,250.

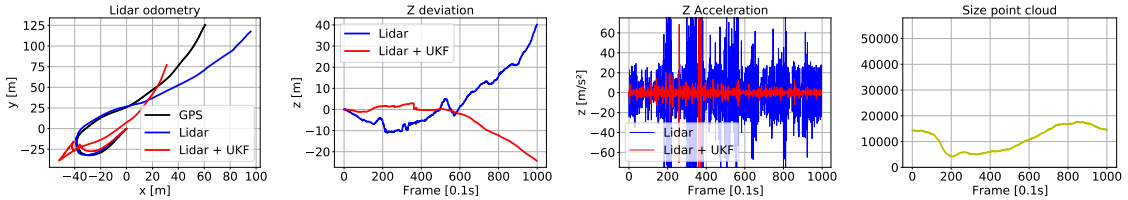

Figure A.12: First run: frame 15,250–16,250.

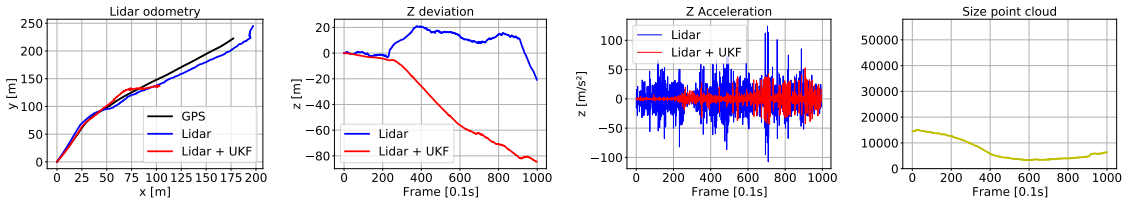

Figure A.13: First run: frame 16,250–17,250.

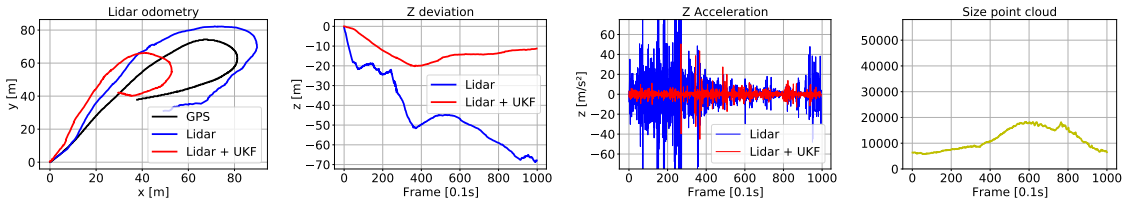

Figure A.14: First run: frame 17,250–18,250.

A.2 Second run

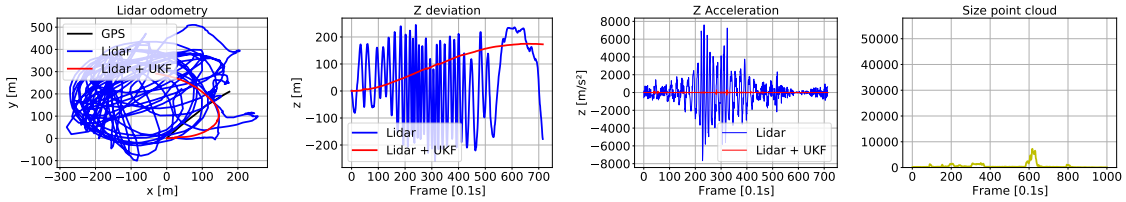

Figure A.15: Second run: frame 2,000–3,000.

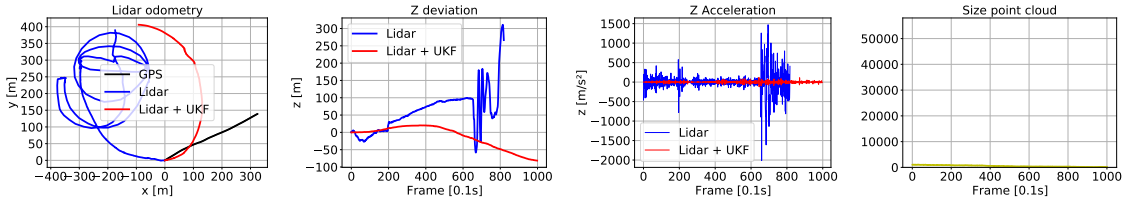

Figure A.16: Second run: frame 4,000–5,000.

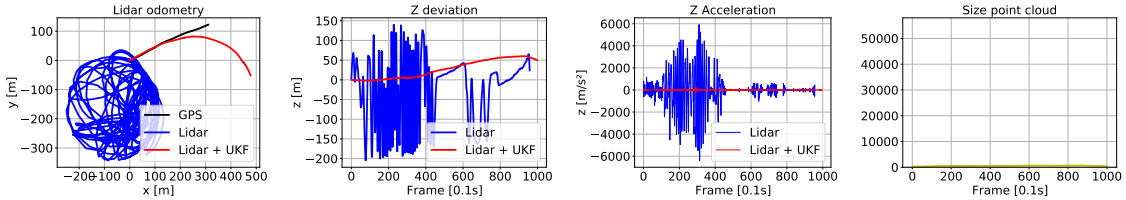

Figure A.17: Second run: frame 6,000–7,000.

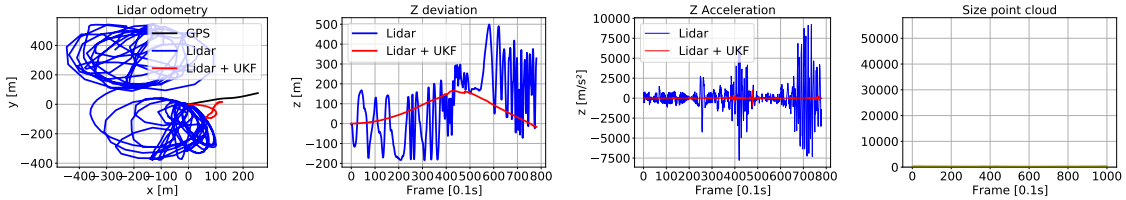

Figure A.18: Second run: frame 7,000–8,000.

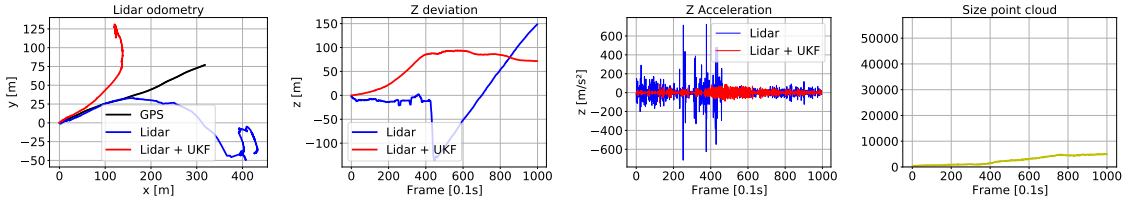

Figure A.19: Second run: frame 8,000–9,000.

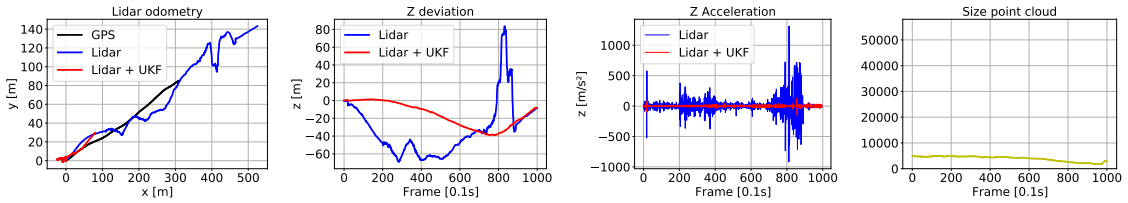

Figure A.20: Second run: frame 9,000–10,000.

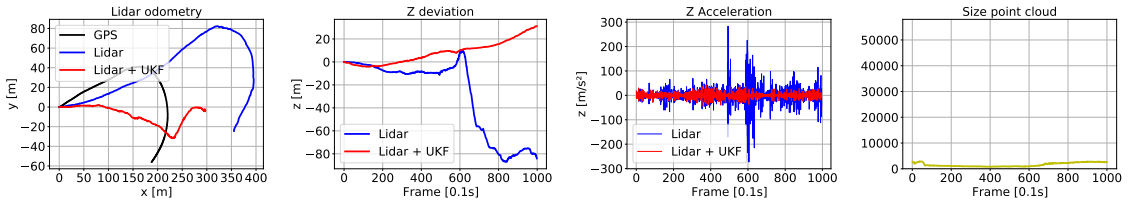

Figure A.21: Second run: frame 10,000–11,000.

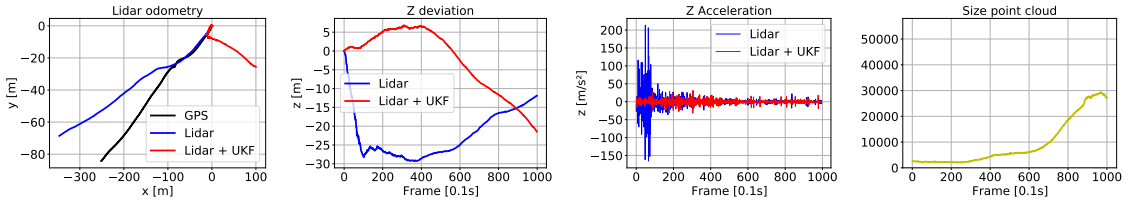

Figure A.22: Second run: frame 11,000–12,000.

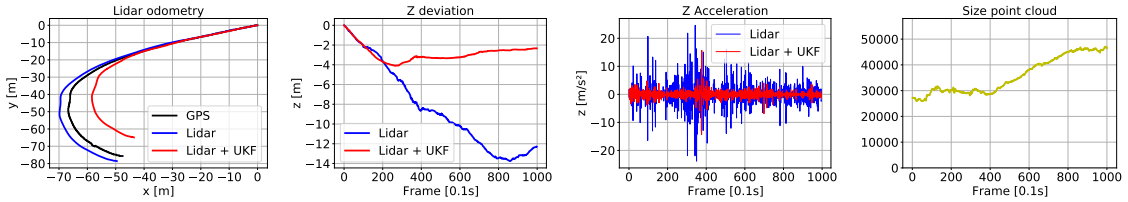

Figure A.23: Second run: frame 12,000–13,000.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY