Aerial Imagery Based Position and Heading Estimation

For maritime autonomous vessels in GNSS-challenged environments

Master's thesis in Master Programme Systems, Control and Mechatronics

ALGOT BERGMAN
TEODOR LINDELL

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Aerial Imagery Based Position and Heading Estimation

For maritime autonomous vessels in GPS-challenged environments

ALGOT BERGMAN
TEODOR LINDELL

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2023

Aerial Imagery Based Position and Heading Estimation
For maritime autonomous vessels in GPS-challenged environments
ALGOT BERGMAN
TEODOR LINDELL

Supervisor: Erik Lund, CPAC Systems AB
Supervisor: Valter Strömberg, CPAC Systems AB
Examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

Master's Thesis 2023
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: Matlab visualization of LiDAR data projected on an aerial image in a maritime environment.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Aerial Imagery Based Position and Heading Estimation
For maritime autonomous vessels in GPS-challenged environments
ALGOT BERGMAN
TEODOR LINDELL
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

One of the most important parts of an autonomous vehicle is the estimation of its position and orientation, which often comes from a GNSS sensor. However, the performance is degraded when the GNSS signals are obstructed. Nowadays, a popular implementation for estimation with accurate precision is to fuse the information from IMU, GNSS and LiDARs. The method of how a position and orientation are calculated from the LiDAR measurements can be done in different ways. This thesis presents a method of how position and orientation can be effectively estimated by matching LiDAR measurements to an aerial image. The method consists of projecting the LiDAR measurements onto the aerial image, filtering the measurements and the image, and then calculating the cross-correlation. The orientation is estimated via stochastic optimization which also finds the maximum correlation to update the position. This is fused in an EKF with information from IMU and a GNSS sensor to get a more precise estimation. The conclusion of the thesis is that the method works well in estimating the position and heading in a GNSS-challenged environment.

Keywords: Extended Kalman filter, LiDAR, aerial imagery, particle swarm optimization, computer vision, point cloud, localization, cross-correlation.

Acknowledgements

We would like to thank all the people who have contributed to the completion of this thesis. Firstly, we would like to thank our examiner Peter Forsberg for all the valuable suggestions and constructive criticism. The weekly meetings we had made sure to keep us on the right path and ensure the successful completion of this thesis. We would also like to extend special thanks to our supervisors, Erik Lund and Valter Strömberg. Their guidance, support, and expertise have been very valuable. Lastly, we would like to thank CPAC Systems AB for the opportunity to do this thesis and for providing us with the equipment to collect real-world data.

Algot Bergman & Teodor Lindell, Gothenburg, June 2023

Thesis advisor: Erik Lund, CPAC Systems AB
Thesis advisor: Valter Strömberg, CPAC Systems AB
Thesis examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

List of Acronyms

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

3D	Three dimensions
2D	Two dimensions
EKF	Extended Kalman Filter
FFT	Fast Fourier Transformed
FoV	Field of View
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
ICP	Iterative Closest Point
IMU	Inertial Measurements Unit
LiDAR	Light Detection and Ranging
MEMS	Micro-Electro-Mechanical-Systems
NDT	Normal Distributions Transform
PSO	Particle Swarm Optimization
RANSAC	Random Sample Consensus
RMSE	Root Mean Square Error
SBAS	Satellite Based Augmentation Systems

Nomenclature

Below is the nomenclature of indices, parameters and variables that have been used throughout this thesis.

Indices

i,j	Indices
m,n	Indices for image width and height
k	Index for discrete time step

Parameters

f	Focal length
s	Skew ratio
p_x	Translations parameters in x direction
p_y	Translations parameters in y direction
γ	Aspect ratio
M, P	Image pixel width
N, Q	Image pixel height
N_{est}	Number of estimations
δ	Computation frequency
g	Gravity vector
δ_{IMU}	IMUs sampling frequency
α_1, α_2, α_3	Tuning parameters
N_m	Maximum number of LiDAR points

Variables

$\mathbf{T}$	Transformation matrix
$\mathbf{R}_{rot}$	Rotation matrix
$\mathbf{R}_{rot,x}$	Rotation matrix around x-axis
$\mathbf{R}_{rot,y}$	Rotation matrix around y-axis
$\mathbf{R}_{rot,z}$	Rotation matrix around z-axis
$\mathbf{t}$	Translation vector
$\mathbf{P}_{proj}$	Projection matrix
$\mathbf{X}$	Set of 3D coordinates
$\tilde{\mathbf{X}}$	Set of 3D homogeneous coordinates
$\tilde{\mathbf{X}}'$	Set of 2D homogeneous coordinates
$\mathbf{K}_{Img}$	Intrinsic camera matrix
$f(t)$, $g(t)$	Time dependent functions
$\mathcal{F}$	Fourier transform
$\mathbf{C}, \mathbf{D}$	2D Images
$\mathbf{B_w}$	Binary Image
$\mathbf{x}_k$	State at the time step k
$\mathbf{x}_0$	Initial state
$\mathbf{P}_0$	Initial covariance
$\mathbf{u}_k$	Control vector at time step k
$\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k)$	Motion model at time step k
$\mathbf{q}_{k-1}$	Gaussian process noise at time step $k-1$
$\mathbf{y}_k$	Sensor measurements at time step k
$\mathbf{r}_k$	Gaussian measurements noise at time step k
$\mathbf{h}(\mathbf{x}_k)$	Measurements model at time step k
$\mathbf{H}_k$	Jacobian of measurements model at time step k
$\mathbf{F}_k$	Jacobian of motion model at time step k
$\hat{\mathbf{x}}_{k\|k-1}$	Posterior mean at time step k
$\mathbf{P}_{k\|k-1}$	Posterior covariance at time step k
$\mathbf{Q}_k$	Covariance of the process noise at time step k
$\mathbf{P}_{k\|k}$	Covariance at time step k
$\hat{\mathbf{x}}_{k\|k}$	Mean at time step k
$\mathbf{R}_{GNSS,k}$	GNSS measurement Covariance matrix at time step k
$\mathbf{R}_{LiDAR,k}$	LiDAR measurement Covariance matrix at time step k
$\mathbf{R}_{Sensor,k}$	Measurement Covariance matrix at time step k

x_{est}	Estimated position of x in 2D space at estimation i
y_{est}	Estimated position of y in 2D space at estimation i
x_{true}	True position of x in 2D space at estimation i
y_{true}	True position of y in 2D space at estimation i
σ_{Gyro}	IMUs gyroscopes standard deviation
σ_{Acc}	IMUs accelerometers standard deviation
$e_{GNSS,k}$	RMSE of the GNSS measurements at time step k
$\sigma_{\text{GNSS Latitude},k}$	Standard deviation of GNSS latitude measurements at time step k
$\sigma_{\text{GNSS Longitude},k}$	Standard deviation of GNSS longitude measurements at time step k
$\sigma_{\text{GNSS Heading},k}$	Standard deviation of GNSS heading measurements at time step k
$\mathbf{X}_{L_1,k}$ - $\mathbf{X}_{L_5,k}$	Individual LiDARs point cloud at time k
$\mathbf{X}_k$	Collective LiDAR point cloud at time k
$\tilde{\mathbf{X}}_k$	Homogeneous collective LiDAR point cloud at time step k
$\mathbf{P}_{proj,k}$	Projection matrix at time step k
ψ_k	Heading at time step k
x_k	Horizontal position at time step k
y_k	Vertical position at time step k
v_{sway}	Velocity in the vessels orthogonal direction at time step k
a_{surge}	Acceleration in the vessels direction at time step k
a_{sway}	Acceleration in the vessels orthogonal direction at time step k
$\dot{\psi}_{surge}$	Vessels yaw rate at time step k
a_x	IMU acceleration in its x-axis at time step k
a_y	IMU acceleration in its y-axis at time step k
ω_z	Angular velocity around the IMUs z-axis at time step k
$\mathbf{Q_a}$	State transition matrix
$\mathbf{H}$	Jacobian of measurements model
N_k	Number of LiDAR points at time step k

Contents

List of Figures

List of Tables

1

Introduction

An autonomous vehicle needs an accurate and reliable position and heading estimation for navigation, regardless if it is operating on land or at sea. The most common way to achieve a position estimation is to use a Global Navigation Satellite System (GNSS) with e.g. Global Positioning System (GPS). The worldwide position accuracy of GNSS is high and can almost pinpoint the exact position in certain applications. However, to achieve this precision, a clear view of the sky and preferably no large reflecting surfaces nearby are required. Quite often surrounding objects are reflecting the satellite signals, making the estimated pose erroneous and sporadic. One solution would be to use a multitude of other positioning methods and fuse these into a more reliable and robust pose.

One positioning method is the use of Inertial Measurements Units (IMUs), which consist of accelerometers, gyroscopes and sometimes magnetometers. The device combines these sensors to measure and provide information about the acceleration and angular velocity of a body. However, IMUs do not provide an absolute position in space, as they rely on dead reckoning, which accumulates errors over time. Another method that is commonly used in positioning and mapping applications is Light Detection and Ranging (LiDAR). LiDAR is a type of remote-sensing technology that uses light to measure distances to objects and produce high-resolution maps of the environment, alternatively referred to as point clouds.

Map-based vehicle localization techniques have grown in connection with the development of autonomous vehicles and the need for precise positioning. Map-based refers to an already existing map of the environment that is used together with different sensor measurements to determine the precise location of a vehicle. This pre-existing map is commonly made up of a high-definition 3D point cloud. However, when it comes to the marine environment, the availability of satellite and aerial images with detailed features is abundant and easily accessible. For example, a dock in the harbor is often clearly visible from an aerial image and could be used for mapping. The development in camera technology, computer processing power, and Unmanned Aerial Systems (UAS) have led to increased capabilities of airborne imagery. In situations when the GNSS signals become unavailable or weak, data from LiDARs and IMU can be fused with information from aerial imagery to achieve a more reliable and robust pose estimation.

1.1 Aim

The aim of this thesis is to develop a reliable and robust position and heading estimation system for a vessel in GNSS-challenged marine environments. The estimation should utilize LiDAR, GNSS and IMU data together with aerial imagery.

1.2 Related work

In this section, the related work for estimating the position and heading for vehicles in marine and terrestrial environments will be presented. Different approaches on how to use the sensor measurements will be covered.

1.2.1 Pose estimation in known environment

Multi-sensor fusion is a common approach to obtain a more accurate and reliable estimate of the vehicle's position and heading. Most often, a combination of the sensors LiDAR, GNSS, IMU, and cameras are used. These are the sensors used in [1], together with a 3D map of the area of interest. The LiDAR and camera are used together to identify points and use them to triangulate the position of the vehicle. This information is then fused with Inertial Navigation System (INS) and GNSS in an Ensemble Kalman Filter (EnKF) to estimate the position of the vehicle. The system showed promising results with a mean translational error of about 0.4m.

A different approach for fusing sensor readings is presented in [2], which does not use cameras but achieves a more accurate pose estimation with a mean error of 5-10cm. The filtering is performed by IMU-based state equations in an error-state Kalman filter with LiDAR and GNSS as the update step. LiDAR measures the position and heading angle of the vehicle by adaptively combining intensity and altitude cues and matching measurements to a pre-built LiDAR map, while GNSS measures only the position.

Pose localization in a marine environment is performed in [3]. The sensors used in this work consisted of LiDAR, IMU and GNSS, which where proven to give low RMSE values. The state estimation uses boat modeling with a constant velocity model together with the velocity readings from the IMU. GNSS, LiDAR and a pre-built LiDAR map were used for the state measurements. For fusing the sensor outputs, the study compares three different approaches; Generic Particle Filter (PF), PF with Unscented Kalman Filter (UKF) and PF with Iterative Closest Point (ICP). The Generic PF proved to be the fastest while the PF with ICP was better suited for precision.

1.2.2 Pose estimation in unknown environment

One problem with the methods above is that they are dependent on a pre-built LiDAR map to match the LiDAR measurements, which will not work in unknown environments. Simultaneous localization and mapping (SLAM) [4] can be used to

map an unknown environment. SLAM can be suitable for online applications if the physical systems can handle the computational load. An implementation of SLAM, in a marine environment, was done in [5]. It aims to reduce vertical drift using hardware and software approaches while identifying the minimum conditions required for SLAM. The baseline performance was built on the LiDAR-only normal distributions transform (NDT) and ICP registration algorithms. The study uses LiDAR, IMU, and GNSS data to fuse 15 states in an Unscented Kalman Filter (UKF), which estimates the sensor position and thereby improves mapping. Both approaches significantly reduce deviation compared to LiDAR-only methods.

An alternative method involves the utilization of Visual Odometry (VO), which does not incorporate the use of a map. VO is a technique used to estimate the motion of a vehicle by tracking features in sequential images captured by the camera. The fallback of VO is that the error grows exponentially if no measurement updates are given. In [6], VO together with GNSS and INS is fused in an error state Extended Kalman Filter (EKF). The results show that the system provides more accurate and continuous positioning results than GNSS alone in GNSS-challenged environment.

A more suitable and reliable localization method for unknown environments should strive to find the absolute position. This would require the use of a map. However, pre-built LiDAR maps will not be available for unknown environments and SLAM approaches can become too computing-heavy for online implementations. Ideally, a localization method that matches measurements to one arbitrary map, working for all situations, would be desired.

1.2.3 Feature matching to aerial view

Aerial imagery can act as a map for localization. A lot of information can be extracted from an aerial image, making it usable in many different scenarios. In [7], vehicle-mounted cameras and an aerial view is used to match visual features to each other using ICP. The proposed algorithm uses lane markings as the visual features. The features from the camera were detected with a Canny edge detector, and with a neural network in the aerial view. The evaluation included both position and heading estimation where the method produces results within 0.25m and 1° heading from the reference position. In [8] a method is proposed for finding road intersections in real-time by the use of LiDAR and aerial image matching. The LiDAR scans are used in an intersection detection algorithm. From the aerial image, an intersection template is extracted and used as the prior for the intersection detection algorithm. The method shows that intersections can effectively be detected at a low false positive rate.

1.2.4 Localization with 2D vector maps

A 2D vector map is a digital representation of geographic information as a collection of points, lines and polygon shapes in two-dimensional space. It provides a compact and structured representation of the environment. In [9], 2D footprints

of the buildings are matched with LiDAR data to achieve an estimated pose. The proposed method uses Normal Distributions Transform (NDT) to match a LiDAR scan with a 2D vector map. A fast filtering method is used to extract aligned points and filter out outliers. The performance shows promising results, with an obtained localization accuracy of 0.21m.

In [10] and [11] a new map structure called multilayer 2D vector map is used for localization. The results show that the new map provides more features for map matching and has less uncertainty, leading to improved accuracy.

However, a 2D vector map provides a simplified representation of objects. For example, the outline of a large building can be represented with multiple straight lines that could be used to match the LiDAR data. But usually smaller objects like a dock in a harbor may be represented with only a straight line. Taking away a lot of useful information about the dock, like the width, length and shape, which makes localization with 2D vector maps less suitable for this thesis.

1.3 Research questions

The following research questions have been identified for further investigation:

- *How does the integration of aerial imagery improve the robustness of a system in case of characteristic GNSS signal disturbance?*
- *Can LiDAR data be effectively matched with aerial imagery to estimate the position and heading of a marine vessel?*
- *How do objects that are only present in the aerial imagery affect the system?*
- *How can uncertainty and errors be handled and propagated for more robust and reliable results?*

1.4 Limitations

The implementation will be done in the Matlab programming language on field data, collected by a vessel in the port of Krossholmen, Volvo Penta's marine leisure testing facility in Gothenburg, Sweden. Therefore, the evaluation of an online implementation will be done by analyzing the computational time required by the offline implementation.

2
Theory

In this chapter, the relevant theory that is essential to understand the methods used in this thesis is presented. First, the different sensors used in this work and their properties are introduced. Followed by how the sensor data is transformed and the algorithms used to yield an estimate of the position and heading. Finally, the filtering process and evaluation process are presented.

2.1 LiDAR properties

A LiDAR is a distance measuring sensor [12]. The sensor calculates the distance by emitting light signals and measuring the time it takes for the light signal to return. The distance is measured in a range of vertical and horizontal directions. Depending on what type of LiDAR is used, the maximum and minimum measurable distance differs. Generally, measured distances are between a few meters to around 200 meters. By emitting several rays of light between a span of vertical and horizontal directions, referred to as a Field of View (FoV), a cluster of points is generated. This cluster of points is often referred to as a point cloud. Point clouds are generated at the sample rate of the LiDAR which usually is between 10Hz - 20Hz.

LiDARs can be categorized into two main types, mechanical and solid state. The main difference between the types is that the mechanical LiDAR has moving parts within and has a wider horizontal and vertical FoV. The solid-state LiDAR has a higher resolution and does not have any moving parts within which makes it less likely to break because of vibrations. However, the solid-state has lower FoV coverage.

2.2 IMU properties

In three axes perpendicular to each other, the IMU gyroscope measures angular velocity, the accelerometer measures linear acceleration and the magnetometer measures magnetic fields [13]. There exist several different types of gyroscopes which measure angular velocity [14]. One type of gyroscope is the Micro-Electro-Mechanical-Systems (MEMS) gyroscope. The MEMS gyroscope generally consists of springs, dampers and a resonating mass with a capacitance. As the resonating mass moves, the capacitance inside the gyroscope changes. This change is proportional to the displacement, which in turn is proportional to the angular velocity.

MEMS accelerometers work much like MEMS gyroscopes in the way that the measurement is computed by measuring the change in capacitance [15]. The MEMS accelerometer consists of a movable mass and fixed electrodes. When a force is applied to the accelerometer, the mass is displaced and the capacitance is changed proportional to the displacement. The change in capacitance is used to calculate the acceleration. When an accelerometer is aligned with the gravity vector, -1g will be measured. Some types of IMUs combine the measurement from gyroscopes and accelerometers to yield the orientation of the IMU and filter out the gravity vector. They can also sometimes use the magnetometer for better estimation of the orientation. For best accuracy, accelerometers should be placed in the center of rotation so momentum is not measured as a linear force. In marine applications, this center of rotation is often referred to as the center of buoyancy.

2.3 GNSS properties

The GNSS sensor receives radio signals from satellites and via triangulation, its latitude and longitude are measured [16]-[17]. The sensor measures the time when a satellite signal is received and then compares it with the time when it was transmitted. Since the transmitted signal travels at a constant speed, the distance to the satellite can be computed. When multiple satellites transmit signals to the sensor and then multiple distances are known, the location is computed. The orientation of the sensor can be estimated if the sensor is equipped with two signal-receiving antennas. GNSS sensor also yields velocity by deriving the distance with respect to the time lapse between two position measurements or by using Doppler measurements related to user-satellite motion [18].

In certain areas around the globe, the precision of GNSS measurements is increased by the use of augmentation systems such as Satellite-Based Augmentation Systems (SBAS) [16]. SBAS consists of a network of ground reference stations that monitors and generates data of GNSS signals. The data is processed and a message is constructed. The message is then transmitted to geostationary satellites in the constellation which emits the message that the GNSS sensor then receives. The sensor then processes the message to get information about the signals and then corrects the measurement.

Some GNSS sensors also estimate the standard deviation of the measured heading and position in real-time. The standard deviation is a good indicator of how well the position and heading have been measured.

2.4 Transformation matrices

The material presented in this section is accessed from [19]. Transformation matrices are used to describe a transformation between two coordinate frames. The transformations theory covered will be transformations in 3D space and transformation

from 3D to 2D space which is also known as projection.

2.4.1 3D transformations

A 3D transformation matrix consists of a translation vector $\mathbf{t}$ and a rotation matrix $\mathbf{R}_{rot}$ as

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{rot} & \mathbf{t} \\ \mathbf{0}^\intercal & 1 \end{bmatrix}. \tag{2.1}$$

The rotation matrix for rotating one coordinate frame to another is done by the rotations along one of the coordinate frame's separate axes. A rotation around its own axis with the given angle Θ is defined by

$$\mathbf{R}_{rot,x}(\Theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\Theta & -\sin\Theta \\ 0 & \sin\Theta & \cos\Theta \end{bmatrix}, \tag{2.2}$$

$$\mathbf{R}_{rot,y}(\Theta) = \begin{bmatrix} \cos\Theta & 0 & \sin\Theta \\ 0 & 1 & 0 \\ -\sin\Theta & 0 & \cos\Theta \end{bmatrix}, \tag{2.3}$$

$$\mathbf{R}_{rot,z}(\Theta) = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.4}$$

The rotation matrix is then defined by

$$\mathbf{R}_{rot} = \mathbf{R}_{rot,x}(\Theta_1)\mathbf{R}_{rot,y}(\Theta_2)\mathbf{R}_{rot,z}(\Theta_3), \tag{2.5}$$

where Θ_1, Θ_2 and Θ_3 are the angels of rotation in each direction. The transformation of a set of 3D coordinates, denoted as $\mathbf{X}_s$, to a base frame, begins by converting $\mathbf{X}_s$ to homogeneous coordinates as

$$\tilde{\mathbf{X}}_s = \begin{bmatrix} \mathbf{X}_s \\ 1 \end{bmatrix}. \tag{2.6}$$

Then the transformation is applied as shown in equation (2.7).

$$\tilde{\mathbf{X}}_B = \mathbf{T}_B^s \tilde{\mathbf{X}}_s \tag{2.7}$$

$\mathbf{T}_B^s$ is the transformation from the coordinate frame of $\mathbf{X}_s$ to the base frame and $\tilde{\mathbf{X}}_B$ is the homogeneous transformed coordinates. To convert $\tilde{\mathbf{X}}_B$ to cartesian coordinates $\mathbf{X}_B$, $\tilde{\mathbf{X}}_B$ is divided by its last row and then the last row is discarded.

2.4.2 2D projection

To transform a view from 3D into 2D, the transformation can be done via a projection matrix [20]. Applying a projection onto a set of 3D coordinates such as $\mathbf{X}_s$ will then yield a homogeneous 2D subspace as

$$\tilde{\mathbf{X}}'_s = \mathbf{P}_{proj}\mathbf{X}_s, \tag{2.8}$$

where $\tilde{\mathbf{X}}'_s$ is the set of coordinates projected to 2D and $\mathbf{P}_{proj}$ is the projection matrix. $\mathbf{P}_{proj}$ is defined by the rotation matrix $\mathbf{R}$, translation vector $\mathbf{t}$ and camera matrix $\mathbf{K}_{Img}$ as equation (2.9) shows.

$$\mathbf{P}_{proj} = \mathbf{K}_{Img} \begin{bmatrix} \mathbf{R}_{rot} & \mathbf{t} \end{bmatrix} \tag{2.9}$$

The intrinsic camera matrix $\mathbf{K}_{Img}$ is defined by the specific properties of the second view. It is used when transforming a set of points onto images and is defined as

$$\mathbf{K}_{Img} = \begin{bmatrix} \gamma f & s & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.10}$$

where f is referred to as the focal length, which corresponds to how the view should be scaled. s corresponds to the skew ratio. p_x and p_y are translations parameters and γ is the aspect ratio. The parameters of the intrinsic camera matrix K can be obtained from the resolution and size of the image.

2.5 Canny edge detection

The Canny edge detector is an edge detection method developed by John F. Canny in 1986 [21]. The purpose of edge detection is to reduce the amount of data to be processed while preserving useful structural information about object boundaries. The process of Canny edge detection involves smoothing the image, detecting areas with a gradient change and connecting those areas to form edges. An example of the Canny edge detection on an aerial image can be seen in Figure 2.1.

Figure 2.1: Input image (left) and the result of canny edge detection algorithm (right).

2.6 RANSAC

Random Sample Consensus (RANSAC), first presented in [22], is an iterative algorithm that estimates the parameters of a mathematical model from a set of data

points. The basic idea of RANSAC involves a random selection of a subset of points as a representation of the model, followed by an evaluation. The evaluation is done by selecting a certain predefined threshold and comparing the number of inliers and outliers among the remaining data points. The procedure is reiterated until a good approximation of the model, with sufficiently many data points classified as inliers, is obtained. RANSAC is commonly used to fit lines in two dimensions.

2.7 Cross-correlation

Cross-correlation is commonly used in image processing to detect similarities between two images. The 2D cross-correlation between two matrices $\mathbf{D}_{M\times N}$ and $\mathbf{G}_{P\times Q}$ is a matrix $\mathbf{C}_{(M+P-1)\times(N+Q-1)}$. The cross-correlation $\mathbf{C}$ is given by

$$\mathbf{C}(o,l) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \mathbf{D}(m,n)\bar{\mathbf{G}}(m-o,n-l), \tag{2.11}$$

where

$$\begin{aligned} -(P-1) \leq o \leq M-1, \\ -(Q-1) \leq l \leq N-1, \end{aligned} \tag{2.12}$$

and $\bar{\mathbf{G}}$ denotes complex conjugation [23]. The mathematical expression can be described as the matrix $\mathbf{G}$ being slid across the matrix $\mathbf{D}$ for calculating the correlation. The operation computes the sum of element-wise products between $\mathbf{D}$ and a shifted version of $\mathbf{G}$ at each position of $\mathbf{D}$. This generates a measurement of similarity between the two matrices at each position.

2.8 FFT-based cross-correlation

Calculating the cross-correlation is usually computationally demanding. However, multiplication of Fast Fourier Transformed (FFT) signals are a common and efficient way to solve this problem. First and foremost, the cross-correlation "$\star$" of a function $f(t)$ and $g(t)$ is equivalent to

$$f(t) \star g(t) = \overline{f(-t)} * g(t), \tag{2.13}$$

where "$*$" is the convolution of the functions. Also, it is known that

$$\mathcal{F}\Big\{\overline{f(-t)}\Big\} = \overline{\mathcal{F}\{f(t)\}}, \tag{2.14}$$

where $\mathcal{F}$ is the Fourier transform. Utilizing this, equation (2.13) becomes

$$\mathcal{F}\{f(t) \star g(t)\} = \overline{\mathcal{F}\{f(t)\}} * \mathcal{F}\{g(t)\}. \tag{2.15}$$

To be able to find the peak value of the cross-correlation, the expression needs to be transformed back to state space with the inverse Fourier transform,

$$\mathcal{F}^{-1}\Big\{\mathcal{F}\{f(t) \star g(t)\}\Big\} = \mathcal{F}^{-1}\Big\{\overline{\mathcal{F}\{f(t)\}} * \mathcal{F}\{g(t)\}\Big\}. \tag{2.16}$$

Referring back to Section 2.7, by replacing the functions $f(t)$ and $g(t)$ with $\mathbf{D}$ and $\mathbf{G}$ the cross-correlation $\mathbf{C}$ becomes

$$\mathbf{C}(k,l) = \mathrm{FFT}^{-1}\Big\{\overline{\mathrm{FFT}\{\mathbf{G}\}} * \mathrm{FFT}\{\mathbf{D}\}\Big\}, \tag{2.17}$$

where the equation is written in terms of fast Fourier transforms. Equation (2.17) illustrates the cross-correlation theorem [24].

2.9 Particle swarm optimization

Particle swarm optimization (PSO) is a stochastic optimization algorithm proposed by Kennedy and Eberhart in 1995 [25]. PSO captures the ability of a swarm to collectively and efficiently search a space for a specific goal. The swarm in PSO is built up by an arbitrary number of particles. Each particle is associated both with a position and a velocity, as well as a method for determining the changes in velocity depending the performance of the particle itself and that of other particles. The algorithm is presented below [26].

Algorithm 1 Basic particle swarm optimization algorithm

1. Initialize positions $\mathbf{x}_i$ and velocities $\mathbf{v}_i$ of each particle p_i, $i = 1, \ldots, N$. The second subscript, $j = 1, \ldots, n$ where n is the dimensionality of the problem.
   1.1 $x_{ij} = x_{\min} + r(x_{\max} - x_{\min})$
   1.2 $v_{ij} = \frac{\alpha}{\Delta t}\Big(- \frac{x_{\max} - x_{\min}}{2} + r(x_{\max} - x_{\min})\Big)$
2. Evaluate each particle in the swarm by computing $f(\mathbf{x}_i)$.
3. Update the best position of each particle $\mathbf{x}_i^{\mathrm{pb}}$ and the global best position $\mathbf{x}^{\mathrm{sb}}$.
   3.1 **if** $f(\mathbf{x}_i) < f(\mathbf{x}_i^{\mathrm{pb}})$ **then** $\mathbf{x}_i^{\mathrm{pb}} \leftarrow \mathbf{x}_i$
   3.2 **if** $f(\mathbf{x}_i) < f(\mathbf{x}^{\mathrm{sb}})$ **then** $\mathbf{x}^{\mathrm{sb}} \leftarrow \mathbf{x}_i$
4. Use the current values of $\mathbf{x}_i^{\mathrm{pb}}$ and $\mathbf{x}^{\mathrm{sb}}$ to update the particle velocities and positions.
   4.1 $v_{ij} \leftarrow w v_{ij} + c_1 q\left(\frac{x_{ij}^{\mathrm{pb}} - x_{ij}}{\Delta t}\right) + c_2 r\left(\frac{x_j^{\mathrm{sb}} - x_{ij}}{\Delta t}\right)$
   4.2 Restrict velocities to maintain the coherence of the swarm, $|v_{ij}| < v_{\max}$
   4.3 $x_{ij} \leftarrow x_{ij} + v_{ij}\Delta t$
5. Return to step 2 until termination criterion has been reached.

In Algorithm 1 the constants q and r is uniform random numbers in the range $[0, 1]$. α is a constant in the range $[0, 1]$ and Δt is the time step length. The goal of the algorithm is to minimize the objective function $f(\mathbf{x})$. The parameters c_1 and c_2 are often referred to as the cognitive component and the social component. c_1 is a measurement of how much each particle should move towards their individual best positions using their own velocity vectors. Similarly, c_2 guides particles to explore new areas by adjusting their velocity vectors based on the global best position found by any other particle in the swarm. w is referred to as the inertia weight and is a measurement of how much the particle prioritizes exploration ($w > 1$) or exploitation ($w < 1$).

2.10 Extended Kalman filter

To obtain a more precise estimation for position and heading, the measurements from the sensors can be fused and filtered via Bayesian filtering [27], a recursive algorithm. In Bayesian filtering, the optimal filtering solution for linear systems with Gaussian measurements is yielded by computing the joint posterior distribution given current and previous states, and measurements. The algorithm involves a prediction step followed by an update step. In the prediction step, an estimate is based on the previous state and a motion model, along with their uncertainties. In the update step, the Bayesian filter computes the likelihood of the next measurement given the predicted state. The likelihood is then combined with the predicted state to obtain an estimation. The implementation of optimal filtering is done by the Kalman Filter (KF) for linear problems. For non-linear problems with non-Gaussian measurements, the Extended Kalman Filter (EKF) can be used. The EKF model can be written as

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{q}_{k-1}, \tag{2.18}$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{r}_k, \tag{2.19}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the vessels states, $\mathbf{y}_k \in \mathbb{R}^m$ is the sensor measurements and $\mathbf{u}_k$ is the control signal, $\mathbf{q}_{k-1}$ is the Gaussian process noise, $\mathbf{r}_k$ is the Gaussian measurement noise, $\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k)$ is the motion model function and $\mathbf{h}(\mathbf{x}_k)$ is the measurement model function. The subscript k denotes the current time instance. The idea of the EKF is to utilize Taylor series approximations to form Gaussian approximations of the filtering densities. The Taylor expansion for the motion and measurement model gives the Jacobians

$$\mathbf{F}_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{x}_{k-1|k-1}}, \tag{2.20}$$

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{x}_{k|k-1}}. \tag{2.21}$$

The posterior mean $\hat{\mathbf{x}}_{k|k-1}$ and the posterior covariance $\mathbf{P}_{k|k-1}$ is calculated by the prediction step to of the EKF as

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k), \tag{2.22}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\intercal + \mathbf{Q}_k, \tag{2.23}$$

where $\mathbf{Q}_k$ is the covariance of the process noise. The covariance $\mathbf{P}_{k|k}$ and mean $\hat{\mathbf{x}}_{k|k}$ of the EKF is computed by the update step as

$$\mathbf{v}_k = \mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}, \tag{2.24}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\intercal + \mathbf{R}_{Sensor,k}. \tag{2.25}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\intercal \mathbf{S}_k^{-1}, \tag{2.26}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{v}_k, \tag{2.27}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} + \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\intercal, \tag{2.28}$$

where $\mathbf{R}_{Sensor,k}$ is the covariance matrix of the measurement. If the sensor measurements are registered at asynchronous times, the update step of the Kalman filter can be constructed with a two-layer fusion structure as described in [28]. This means that the update step is performed separately for every set of sensor measurements.

2.11 Evaluation

The position estimation performance will be evaluated based on RMSE and the maximum error. The RMSE is defined as the errors RMS in a 2D space [29].

$$\text{Position RMSE} = \sqrt{\frac{1}{N_{est}} \sum_{i=1}^{N_{est}} ((x_{est,i} - x_{true,i})^2 + (y_{est,i} - y_{true,i})^2)}, \tag{2.29}$$

$x_{est,i}$ and $y_{est,i}$ are the estimated position and $x_{true,i}$ and $y_{true,i}$ are the corresponding true position of the vessel. N_{est} is the total number of estimations. The maximum position error is defined as

$$\text{Max Position Error} = \max_{i=1}^{N_{est}} \sqrt{(x_{est,i} - x_{true,i})^2 + (y_{est,i} - y_{true,i})^2}. \tag{2.30}$$

Heading estimation performance will also be evaluated by RMSE as

$$\text{Heading RMSE} = \sqrt{\frac{1}{N_{est}} \sum_{i=1}^{N_{est}} (\psi_{est,i} - \psi_{true,i})^2}, \tag{2.31}$$

where $\psi_{est,i}$ is the heading estimated and $\psi_{true,i}$ is the true heading of the vessel. For the heading, the max error is defined as

$$\text{Max Heading Error} = \max_{i=1}^{N_{est}} \sqrt{(\psi_{est,i} - \psi_{true,i})^2}. \tag{2.32}$$

3
Methods

In this chapter, the methods used to arrive at the result will be presented. The primary objective of the method was to determine a position and heading of a vessel by using data from IMU, LiDARs, GNSS and aerial images. This section will start by introducing the equipment chosen and the test environment. Then the process of how the position and heading were estimated will be covered.

3.1 Data collection

To evaluate the performance of position and heading algorithm, data was collected from a vessel driving along the coast in the port of Krossholmen. The data was collected with an IMU sensor, five LiDAR sensors and a GNSS sensor.

3.1.1 Vessel

Tiara 48 was chosen as the test vessel since it is well suited for a wide range of testing applications. The test vessel's coordinate frame can be seen in Figure B.1 in Appendix B and is referred to as the base frame.

3.1.2 IMU

The IMU of choice was developed by CPAC systems AB [30]. The IMU consists of a MEMS gyroscope, a MEMS accelerometer and a magnetometer. The gyroscope measures the angular velocities around its three axes with a range up to 200°/s and the accelerometer can measure forces on the three axes up to ±16g. The IMUs maximum sampling time was 100 Hz and mounted on the Tiara 48 as in Figure B.2 in Appendix B. The IMU was placed as close to the GNSS sensor as possible. The noise on the angular velocity measurement of the gyroscope was $0.008°/\text{s}/\sqrt{\delta_{\text{IMU}}}$ and 37 $\text{mg}/\sqrt{\delta_{\text{IMU}}}$ on the accelerometer, where δ_{IMU} is the IMUs sampling frequency. The variance of the IMU sensors measurements σ^2_{Gyro} and σ^2_{Acc} were calculated from

$$\sigma^2_{Gyro} = \left(\frac{0.008}{\sqrt{\delta_{\text{IMU}}}}\right)^2, \tag{3.1}$$

$$\sigma^2_{Acc} = \left(\frac{\frac{37}{1000} \cdot 9.81}{\sqrt{\delta_{\text{IMU}}}}\right)^2. \tag{3.2}$$

3.1.3 GNSS

The GNSS sensor of choice was the V200s Vector™ GNSS Compass [31]. To maximize the accuracy of the V200s, it was mounted on the roof of the vessel with no blocking objects in the vicinity, as shown in Figure B.3 in Appendix B. The transformation matrix to Tiara 48 base, T_B^{GNSS}, can be seen in Appendix A. The V200s sampling rate was 10 Hz and it also supports SBAS. When the V200s is within reach of SBAS messages, the overall error characteristic is 0.3m RMSE for the position accuracy and 1.5° RMSE for the heading accuracy. However, the performance of the V200s will degrade when there is an obscuration against the GNSS signals and the specified error characteristics will then change as well. The V200s will then estimate the position measurements RMSE in real-time, denoted as $e_{GNSS,k}$. This was used to estimate the variance of the V200s measurements from RMSE as described in [32], see equations (3.3)-(3.5).

$$\sigma^2_{\text{GNSS Latitude},k} = \left(\frac{e_{GNSS,k}}{1,414}\right)^2 \tag{3.3}$$

$$\sigma^2_{\text{GNSS Longitude},k} = \left(\frac{e_{GNSS,k}}{1,414}\right)^2 \tag{3.4}$$

$$\sigma^2_{\text{GNSS Heading},k} = \left(\frac{1.5}{1,414}\right)^2 \tag{3.5}$$

In scenarios where the GNSS environment was too challenging for the V200s, so no position measurement could be yielded, the V200s stopped providing information about the expected RMSE. In such instances, it was manually set to an arbitrarily large number.

3.1.4 LiDAR

The vessel was equipped with five LiDARs, mounted as in Figure B.4 in Appendix B. Transformation according to the theory in sections 2.4.1 was applied to the LiDAR measurements to yield a collective point cloud for all LiDARs. The transformation matrices were computed through calibration prior to this work. The calibration was done by scanning an arbitrary environment with several overlapping features within multiple LiDARs FoV. ICP was then used on the different point clouds to yield the transformation between them.

In Figure B.4 in Appendix B, the position and orientation of each LiDAR mounted on the Tiara 48 can be seen via their coordinate systems. The x-axis of the LiDARs coordinate system corresponds to the direction it was facing. L_1, L_2 and L_3 were of type *RS-BPearl* [33]. This was chosen for its wide FoV and placed on the vessel to get a wide coverage of the surrounding environment. Together with L_4 and L_5, which were of the type *RS-Helios* [34], the vessel has almost total coverage of the environment it is located in. The relevant specification for each LiDAR type can be seen in Table 3.1.

Table 3.1: LiDAR specifications.

Name	*RS-BPearl*	*RS-Helios*
Horizontal FoV	360°	360°
Vertical FoV	87.19°	70°
Horizontal resolution	0.2°	0.2°
Range	$100m$	$150m$
Range Accuracy	$\pm 0.03m$ (3σ)	$\pm 0.03m$ (3σ)
Vertical lines	32	32
Type	Mechanical	Mechanical

The transformation matrices used to transform each individual LiDAR point cloud into a collective point cloud can be seen in Appendix A. A visualization of a LiDAR point cloud can be seen in Figure 3.1.

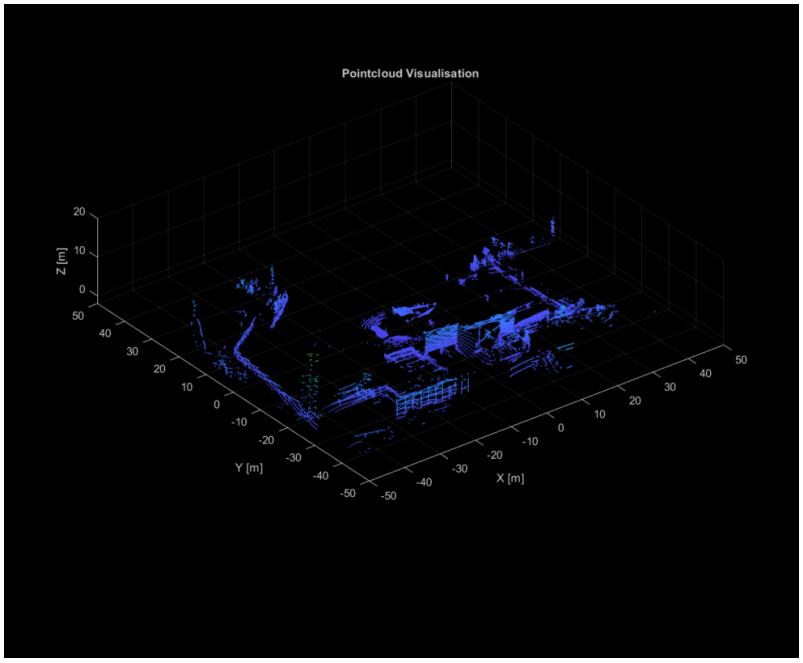

Figure 3.1: Visualisation of LiDAR point cloud.

3.1.5 Test environment

As previously mentioned, the harbor of Krossholmen was selected as the test environment. During testing, the vessel was constrained to be within 60 meters of the land to ensure the LiDARs captured features of the coastline. Three different route scenarios at two different locations were logged and used for testing. In the first session, case 1, the vessel enters the port with the dock in close proximity to its right side, as depicted with the blue trajectory in Figure 3.2a. In the second session, case 2, the vessel enters the port from the opposite direction and makes a direct turn to the left, as depicted with the red trajectory in Figure 3.2a. The third logging session, case 3, takes place at a nearby location in Krossholmen, where the vessel stood still but swayed back and forth, creating a rocking motion. Case 3 is

visualized in Figure 3.2b with a red trajectory. The time duration for the scenarios was around one minute for case 3 and around two minutes for cases 1 and 2.

(a) **(b)**

Figure 3.2: Aerial images of two locations at the port of Krossholmen. The first location is illustrated in (a) together with the trajectory of the two ground truth paths from the GNSS data. The blue trajectory represents case 1 and the red represents case 2. The second location is illustrated in (b) together with the ground truth path/location for case 3.

To assess the algorithm's robustness, additional noise signals were added to the GNSS sensor measurement. The added noise was designed to have the characteristics of noise that occur in a GNSS-challenged environment. According to [35] and [36], GNSS noise can be categorized into two parts, a noisy but unambiguous part and a precise but ambiguous part. This means the GNSS noise can be described as a white-like error with occasional jumps with inconsistent amplitude. The characteristic of the unambiguous part of the noise was made by stationary placing the GNSS receiver under a large metal crane tower for a long period of time. This caused the measured position to gradually deviate over time. To simulate the second part of the GNSS noise, the precis but ambiguous part, a constant value in latitude or longitude was added over a set duration. The GNSS signals without added noise were considered as the vessel's ground truth.

The position and heading estimation was implemented in Matlab using the data collected from the scenarios. All the tests and experiments were done using the same computer with the following specifications

Table 3.2: Specifications of the computer used for testing.

Make	Lenovo
Model	ThinkPad P15 Gen 2
Operating system	Windows 10, 64-bit
Processor	i7-11800H, 2.3 GHz, 8 Core(s)
RAM	32 GB

3.2 General approach

In order to estimate the position and heading, all the different sensor measurements were fused in an EKF, as shown in Figure 3.3. The IMU measured linear acceleration and angular velocities, while GNSS measured position and heading, which were direct inputs to the filter. However, the LiDARs generated a point cloud that needed to be processed together with the aerial image and heading to yield a refined position and heading reading. This was done by a self-constructed positioning algorithm. A detailed framework outlining the specific processing steps is provided in the following sections.

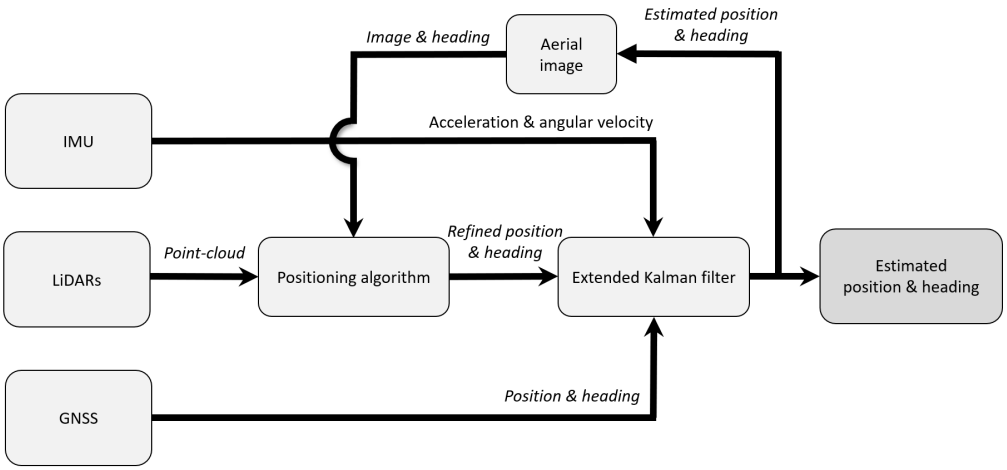

Figure 3.3: A simplified version of the general approach for the position and heading estimation.

3.3 Design and implementation of positioning algorithm

The projection of a point cloud onto an aerial image, captured from a position with a noisy GNSS signal does not align correctly, as illustrated in Figure 3.4. To address this problem, it is necessary to determine the appropriate offset that shifts the point cloud to the correct position. In this case, by adjusting the point cloud a little upwards and to the left, the resulting projection onto the aerial image could be significantly improved with features and edges more accurately aligned with their

corresponding counterparts in the image. The idea behind the positioning algorithm is to find the offset that compensates for the interference caused by the noisy GNSS signal, allowing for more accurate positioning. An overview of how the algorithm works is summarized in Algorithm 2.

Algorithm 2 Positioning algorithm

1. Read aerial image from estimated position
2. Project LiDAR point cloud on aerial image
3. Extract features from aerial image
4. Extract features from LiDAR point cloud
5. Match features from both sources and find transformation between them

Figure 3.4: Aerial image from noisy GNSS signal with projected LiDAR point cloud as blue dots.

3.3.1 Aerial images

Aerial images were provided by a toolbox from Matlab called Mapping Toolbox [37]. The toolbox supports a complete workflow for managing geographic data, making it suitable for obtaining an overhead view of the environment from any location worldwide. It can be used for mapping high-resolution satellite or aerial imagery with a Web Mercator projected coordinate reference system. The Mercator projection preserves angles and is suitable for small regions. To retrieve an aerial image of a specific position from the Mapping Toolbox, the latitude and longitude of the position must be known. The dimensions of the aerial images were set to provide coverage of 150×150 meters around the vessel and the aerial image resolution was

938×940 pixels with no skew ratio. This was used to calculate the parameters as

$$\gamma = \frac{938}{940}, \tag{3.6}$$

$$f = \frac{940}{150}, \tag{3.7}$$

$$p_x = \frac{938}{2}, \tag{3.8}$$

$$p_y = \frac{940}{2}, \tag{3.9}$$

$$s = 0. \tag{3.10}$$

which was used to yield the intrinsic camera matrix $\mathbf{K}_{Img}$ from equation 2.10 as

$$\mathbf{K}_{Img} = \begin{bmatrix} 6.2533 & 0 & 469 \\ 0 & 6.2667 & 470 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.11}$$

A visualization of the aerial images from the mapping toolbox can be seen in Figure 3.4.

3.3.2 Project LiDAR point cloud

The collective LiDAR point cloud $\mathbf{X}_k$ was generated via transformation matrices to the base of the vessel, as described in Section 2.4.1. The transformation matrices can be seen in Appendix A. Equation (2.7) was utilized to convert the point clouds $\mathbf{X}_{L_1,k}$, $\mathbf{X}_{L_2,k}$, $\mathbf{X}_{L_3,k}$, $\mathbf{X}_{L_4,k}$ and $\mathbf{X}_{L_5,k}$ generated by each respective LiDAR, into homogeneous transformed coordinates. The homogeneous collective point cloud $\tilde{\mathbf{X}}_{k,B}$ was obtained by

$$\tilde{\mathbf{X}}_{k,B} = \begin{bmatrix} \mathbf{T}_B^{L_1}\tilde{\mathbf{X}}_{L_1,k} & \mathbf{T}_B^{L_2}\tilde{\mathbf{X}}_{L_2,k} & \mathbf{T}_B^{L_3}\tilde{\mathbf{X}}_{L_3,k} & \mathbf{T}_B^{L_4}\tilde{\mathbf{X}}_{L_4,k} & \mathbf{T}_B^{L_5}\tilde{\mathbf{X}}_{L_5,k} \end{bmatrix}. \tag{3.12}$$

$\mathbf{X}_k$ was then produced by converting the homogeneous coordinates $\tilde{\mathbf{X}}_{k,B}$ to cartesian coordinates. The projection matrix $\mathbf{P}_{proj,k}$ was then applied to project the LiDAR point cloud onto an aerial image, described in Section 2.4.2, as

$$\tilde{\mathbf{X}}'_k = \mathbf{P}_{proj,k}\mathbf{X}_k. \tag{3.13}$$

$\mathbf{P}_{proj,k}$ was defined by the 2D rotation

$$\mathbf{R}_{rot}(\psi_k) = \begin{bmatrix} \cos\psi_k & -\sin\psi_k & 0 \\ \sin\psi_k & \cos\psi_k & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{3.14}$$

the translation vector

$$\mathbf{t} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\intercal, \tag{3.15}$$

and the intrinsic camera matrix $\mathbf{K}_{Img}$ from equation (3.11). ψ_k were the vessels heading at time step k. The last step consists of converting the projected homogeneous coordinates $\tilde{\mathbf{X}}'_k$ to cartesian coordinates $\mathbf{X}'_k$.

3.3.3 Extract features from aerial image

Canny edge detection, introduced in Section 2.5, returns a binary image, also referred to as $\mathbf{B_w}$ (black and white), containing 1's where the function finds edges in the grayscale image and 0's elsewhere [38].

3.3.4 Extract features from LiDAR point cloud

The next part of the positioning algorithm consisted of extracting the important features from the LiDAR point cloud. However, the LiDARs produce a large point cloud, containing a lot of information in each frame. The point cloud therefore needed to be reduced first. A large number of points did not fulfill any important purpose in this situation and were therefore discarded directly. The list below states the criteria for how the points were selected:

- Points outside aerial image.
- Points that have a height value less than 0.3m above the water surface.
- Points within the polygon shape of the vessel.

As presented in Section 2.5, Canny edge detection identifies pixels that have a large gradient, indicating a change in intensity. Because of this, there were likely going to be clear edges identified in the harbor between the blue water and the land. Hence, it was important to keep features like walls and other sharp edges of the docks and piers from the point cloud. When viewing a 3D point cloud projected into 2D, walls and other sharp edges are typically represented as a series of stacked points with high density (same x and y coordinate in 2D). Algorithm 3 selects these points by finding the indices with the most frequently occurring coordinate pairs, so the remaining points could be discarded. A visualization example of the behavior of the algorithm can be seen in Figure 3.5.

Algorithm 3 Select most frequently occurring coordinate pairs

$coords \leftarrow [\text{round}(x), \text{round}(y)]$
$nrPoints \leftarrow 3000$
$uniqueCoords, ic \leftarrow$ Find the unique coordinate pairs in $coords$ and the index mapping of rows in $coords$ to rows in $uniqueCoords$, such that $coords = uniqueCoords(ic)$
$count \leftarrow$ Count the number of times each element in $uniqueCoords$ appears in $coords$
$sortedInd \leftarrow$ Sort $count$ in descending order and return the sorted indices
$topInd \leftarrow$ Select indices of first $nrPoints$ elements of $sortedInd$
$topCoords \leftarrow$ Extract rows in $uniqueCoords$ corresponding to $topInd$
$ind \leftarrow$ Find indices of rows in $coords$ that match rows in $topCoords$

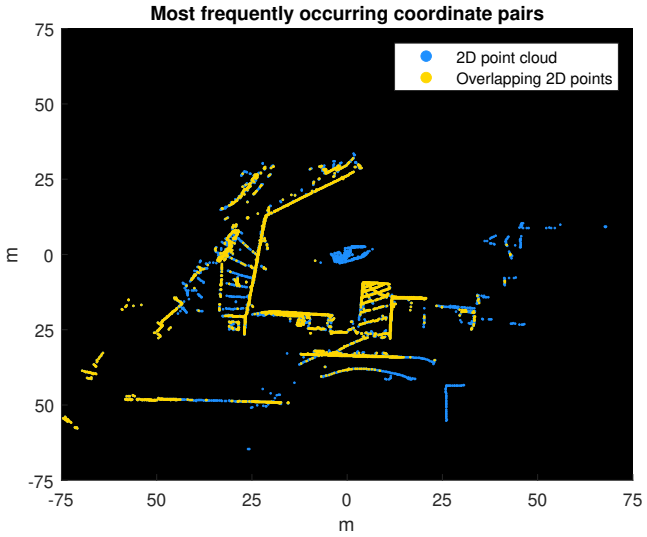

Figure 3.5: The most frequently occurring coordinate pairs (yellow markers) selected with Algorithm 3 applied on all LiDAR points (blue markers).

Since the detected edges in $\mathbf{B_w}$ typically form continuous lines along the boundaries of objects, it should be expected that the LiDAR point cloud would provide similar patterns. As seen in Figure 3.5, the point cloud was not at that stage yet and required further processing. The RANSAC algorithm, presented in Section 2.6, was therefore the next step used to estimate lines from the stacked LiDAR points. The algorithm that was used for this is presented in Algorithm 4. The image was divided into small grids, where the RANSAC algorithm was performed in each grid. This approach proved to be useful since the point cloud contains multiple intersecting and parallel lines. Consequently, the lines were independently estimated in the different regions of the image. An example of how it works is illustrated in Figure 3.6 where the final point cloud, also referred to as $\mathbf{X}_{reduced}$, is shown with red dots.

Algorithm 4 RANSAC-based line fitting for grid segmentation of image

Initilize: Divide binary image into i number of grids, where 1's represents the LiDAR points and 0's represents the absence of LiDAR points in the image
$threshold \leftarrow 15$
for $j \leftarrow 1$ to i **do**
  if number of $points$ in $grid_j > threshold$ **then**
    $line \leftarrow RANSAC(points)$
    $grid_j \leftarrow$ 1's in position of $line$ and 0's elsewhere
  else
    $grid_j \leftarrow$ 0's in all positions
  end if
end for

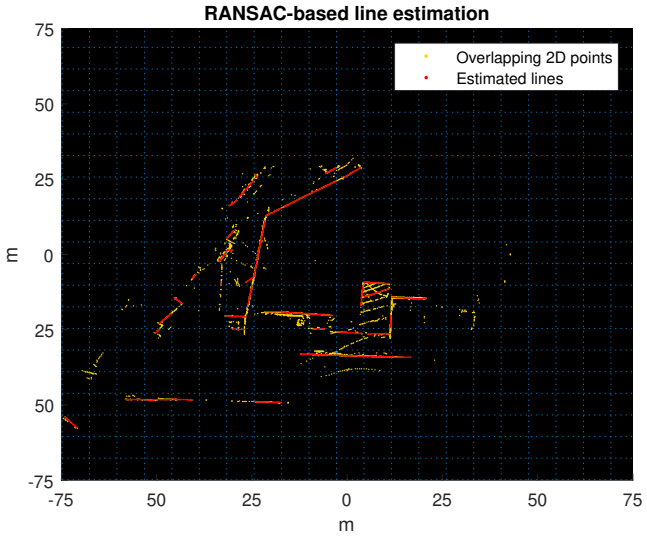

Figure 3.6: LiDAR point cloud before RANSAC (yellow marks) and LiDAR point cloud after RANSAC (red marks). The dashed blue lines show each grid RANSAC is performed in.

3.3.5 Point cloud aligning

The final stage of Algorithm 2 involves finding the transformation between the features present in the reduced point cloud, $\mathbf{X}_{reduced}$, with those extracted from the aerial image, $\mathbf{B_w}$. An example of how the two could look can be seen in Figure 3.7.

Figure 3.7: The grey markers to the left illustrates the features from the aerial image $\mathbf{B_w}$. The red markers to the right illustrates the reduced LiDAR point cloud $\mathbf{X}_{reduced}$.

The sought transformation could be found with an FFT-based cross-correlation, which was described in Section 2.8. Cross-correlation is an effective method for aligning point clouds with high accuracy. However, cross-correlation by itself is not capable of rotating matrices. This limitation means that if the point cloud and

aerial image were not aligned because of a slight inaccuracy in the point cloud's rotation, the computed cross-correlation results would be inaccurate. The relationship between the rotation angle and the cross-correlation is generally nonlinear. As a result, the stochastic optimization method PSO was chosen as a well-suited method for finding the best rotation angle between the matrices. PSO was introduced in Section 2.9.

The search space of interest was set to $[-5°, 5°]$ from the initial pose. Because of the small search space, a good and fast configuration of the PSO was determined to 10 particles and 6 iterations, where each particle represents a rotation angle. The objective function, see equation (3.16), was formulated as the negative cross-correlation between the fixed matrix $\mathbf{B_w}$ and the rotated matrix $\mathbf{X}'_{reduced,i}$ for the i^{th} particle, where $i = 1, \ldots, 10$.

$$f(\mathbf{X}'_{reduced,i}, \mathbf{B_w}) = -\max\Big(\text{FFT}^{-1}\Big\{\overline{\text{FFT}\{\mathbf{X}'_{reduced,i}\}} * \text{FFT}\{\mathbf{B_w}\}\Big\}\Big) \tag{3.16}$$

In conclusion, the PSO finds the optimal rotation angle that maximizes the cross-correlation between $\mathbf{B_w}$ and $\mathbf{X}_{reduced}$, thereby achieving the desired transformation. The remaining parameter values for the PSO was set to $v_{\max} = \alpha = \Delta t = 1$, $c_1 = c_2 = 2$ and $w = 0.4$. A pseudocode for the comprehensive positioning algorithm can be seen in Algorithm 5.

Algorithm 5 Update position and heading measurement from 3D point cloud

Initialize: $x_{est,k}, y_{est,k}, \psi_{est,k}$ as the current estimated position and heading
for 3D point cloud at time k **do**
- $\mathbf{X}_k \leftarrow$ LiDAR measurement ▷ Update point cloud
- $\mathbf{I}_k \leftarrow$ Aerial image of $x_{est,k}$ and $y_{est,k}$ ▷ Load aerial image
- $\mathbf{B}_{\mathbf{w},k} \leftarrow$ Canny-edge($\mathbf{I}_k$) ▷ Detect edges of aerial image
- $\mathbf{P}_{proj,k} \leftarrow \mathbf{K}_{Img}\begin{bmatrix}\mathbf{R}_{rot}(\psi_{est,k}) & \mathbf{t}\end{bmatrix}$ ▷ Update projection matrix
- $\mathbf{X}'_k \leftarrow \mathbf{P}_{proj,k}\mathbf{X}_k$ ▷ Project and translate points
- $\mathbf{X}'_{o,k} \leftarrow \mathbf{X}'_{o,k} \in \mathbf{X}'_k$ ▷ Filter points with Algorithm 3
- $\mathbf{X}_{reduced,k} \leftarrow$ RANSAC($\mathbf{X}'_{o,k}$) ▷ Reduce point cloud with Algorithm 4
- $\psi_{update,k} \leftarrow$ PSO($\mathbf{X}_{reduced,k}, \mathbf{B}_{\mathbf{w},k}$) ▷ Update heading with PSO
- $\mathbf{X}'_{reduced,k} \leftarrow \mathbf{R}_{rot}(\psi_{update,k})\mathbf{X}_{reduced,k}$ ▷ Rotate point cloud with heading
- $x_{update,k}, y_{update,k} \leftarrow$ Cross-correlation($\mathbf{X}'_{reduced,k}, \mathbf{B}_{\mathbf{w},k}$) ▷ Update position
- **Return:** $x_{upd,k}$, $y_{update,k}$ and $\psi_{update,k}$

end for

3.4 Kalman filtering

The Extended Kalman Filter was implemented to yield a more precise estimation of position and heading estimation based on the sensor data. The EKF uses the data from IMU and GNSS as well as the outputs provided from Algorithm 5.

3.4.1 Prediction step

To predict where the vessel was going to be one time step ahead, the motion model was first defined. The nonlinear motion model was defined by the linear forces and momentum applied on the vessel's coordinate system which can be seen in Figure 3.8. The linear forces were applied in the directions of the vessel's base coordinate axis x_b, y_b and z_b which are referred to as surge, sway, and heave respectively. The momentum was applied around the axis and the orientation along the base axis, denoted as the roll, pitch and yaw is also noted as the vessels heading.

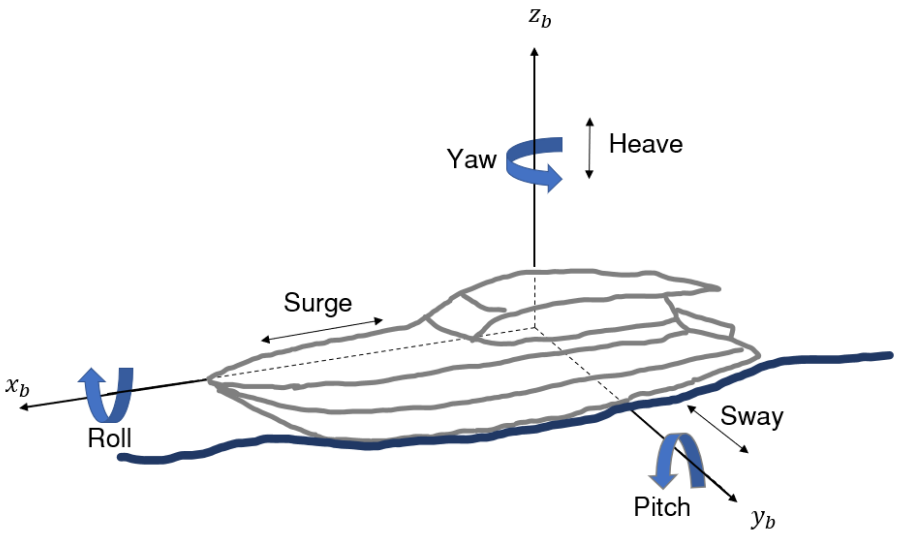

Figure 3.8: Visualisation of the vessel's degrees of freedom and coordinate frame.

Changes in the heave distance, as well as pitch and roll, would yield small changes in the position and heading of the vessel. Therefore, the vessel was assumed to be in a 2D plane where the heave distance as well as the pitch and roll angles were constant. Position and heading were then calculated from the linear acceleration and angular velocity measurements as

$$x_k = x_{k-1} + \delta(\cos(\psi_{k-1})v_{surge,k-1} - \sin(\psi_{k-1})v_{sway,k-1}), \tag{3.17}$$

$$y_k = y_{k-1} + \delta(\sin(\psi_{k-1})v_{surge,k-1} + \cos(\psi_{k-1})v_{sway,k-1}), \tag{3.18}$$

$$v_{sway,k} = v_{sway,k-1} + \delta a_{sway,k-1}, \tag{3.19}$$

$$v_{surge,k} = v_{surge,k-1} + \delta a_{surge,k-1}, \tag{3.20}$$

$$\psi_k = \psi_{k-1} + \delta\dot{\psi}_{k-1}. \tag{3.21}$$

And the state vector was set as

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & v_{surge,k} & v_{sway,k} & \psi_k \end{bmatrix}^\mathsf{T}. \tag{3.22}$$

x_k is the horizontal global position and y_k is the vertical global position. ψ_k is the yaw as well as the heading of the vessel, $v_{sway,k}$ is the vessels velocity in its facing

direction, $v_{surge,k}$ is the velocity sideways (orthogonal to $v_{sway,k}$) and δ is the filters computation frequency which was set to 20 Hz. $\dot{\psi}_{k-1}$ is the yaw rate, $a_{surge,k-1}$ is the acceleration in the vessels facing direction and $a_{sway,k-1}$ is the acceleration sideways. To acquire $\dot{\psi}_{k-1}$, $a_{sway,k-1}$ and $a_{surge,k-1}$, the IMU measurements were transformed to align with the vessel coordinate system

$$\begin{bmatrix} a_{surge,k} \\ a_{sway,k} \\ \dot{\psi}_k \end{bmatrix} = \mathbf{R}_{rot,B}^{IMU} \begin{bmatrix} a_{x,k} \\ a_{y,k} \\ \omega_{z,k} \end{bmatrix}, \tag{3.23}$$

where $\mathbf{R}_{rot,B}^{IMU}$ is the rotation matrix from the IMU to the vessel base, as in Appendix A. $a_{x,k}$ is the linear acceleration measured in the IMUs x-axis, $a_{y,k}$ is the linear acceleration measured in y-axis and $\omega_{z,k}$ is the angular velocity measured around the z-axis. The coordinate system of the IMU can be seen in Appendix B in Figure B.2. The transformed IMU measurements were set as the Kalman filter control vector $\mathbf{u}_k$, see equation (3.24). The IMU measurements are used as a control input because they influence the state prediction step but they are not directly estimated.

$$\mathbf{u}_k = \begin{bmatrix} 0 \\ 0 \\ a_{surge,k} \\ a_{sway,k} \\ \dot{\psi}_k \end{bmatrix} \tag{3.24}$$

From the control vector $\mathbf{u}_k$, state vector $\mathbf{x}_k$ and the state equations (3.17) - (3.20), the motion model $\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k)$ becomes

$$\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) = \begin{bmatrix} 1 & 0 & \delta\cos(\psi_{k-1}) & -\delta\sin(\psi_{k-1}) & 0 \\ 0 & 1 & \delta\sin(\psi_{k-1}) & \delta\cos(\psi_{k-1}) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ v_{surge,k-1} \\ v_{sway,k-1} \\ \psi_{k-1} \end{bmatrix} + \delta\mathbf{u}_k. \tag{3.25}$$

The Jacobian of the motion model $\mathbf{F}_k$ therefore became

$$\mathbf{F}_k = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_{k-1}} = \begin{bmatrix} 1 & 0 & \delta\cos(\psi_{k-1}) & -\delta\sin(\psi_{k-1}) & 0 \\ 0 & 1 & \delta\sin(\psi_{k-1}) & \delta\cos(\psi_{k-1}) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.26}$$

The presence of noise in the control vector was used to model the process noise $\mathbf{q}_{k-1}$ from the IMU variances, introduced in Section 3.1.2. The variances were calculated using equations (3.1) - (3.2) with the IMUs sampling time δ_{IMU} set as equal to the filters computation frequency, 20 Hz.

$$\mathbf{q}_{k-1} = \delta \begin{bmatrix} 0 \\ 0 \\ \mathcal{N}(0, \sigma_{Acc}^2) \\ \mathcal{N}(0, \sigma_{Acc}^2) \\ \mathcal{N}(0, \sigma_{Gyro}^2) \end{bmatrix} \tag{3.27}$$

As described in [39], to derive the process noise matrix $\mathbf{Q}_k$, the state transition matrix $\mathbf{Q_a}$ from equation (3.28) could be used.

$$\mathbf{Q_a} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \delta\sigma^2_{Acc} & 0 & 0 \\ 0 & 0 & 0 & \delta\sigma^2_{Acc} & 0 \\ 0 & 0 & 0 & 0 & \delta\sigma^2_{Gyro} \end{bmatrix} \tag{3.28}$$

The state transition matrix $\mathbf{Q_a}$ and the motion model $\mathbf{F}_k$ yielded the process noise matrix $\mathbf{Q}_k$ as

$$\mathbf{Q}_k = \alpha_1 \mathbf{F}_k \mathbf{Q_a} \mathbf{F}_k^\intercal, \tag{3.29}$$

and can be seen in Appendix C. The tuning constant α_1 addressed uncertainties that arose from assuming the heave distance, pitch, and roll to be constant, as well as accounted for uncertainties related to the placement of the IMU. The process covariance matrix $\mathbf{Q}_k$ together with the motion model $\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k)$ and the motion model Jacobian $\mathbf{F}_k$ was then used to calculate the posterior mean $\hat{\mathbf{x}}_{k|k-1}$ and covariance $\mathbf{P}_{k|k-1}$ as equations (2.22) - (2.23).

3.4.2 Update step

As mentioned in Section 2.10, the update step in the EKF was performed separately for every set of sensor measurements. In this work, it consisted of two different sensor measurements, one from the GNSS sensor and one from the LiDAR-based positioning algorithm. Both of the sensors gave a measurement of the vessel's position and heading, corresponding to the states x_k, y_k and ψ_k. Therefore, the Jacobian of the measurement model became

$$\mathbf{H} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\bigg|_{x_{k-1}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.30}$$

The measurement covariance $\mathbf{R}_{GNSS,k}$ for the GNSS sensor was calculated using the measured standard deviation. During weak GNSS signals, the GNSS sensor did not accurately measure the standard deviation for its measurements. The measurement covariance was therefore multiplied with the tuning parameter α_2 to compensate for the uncertain standard deviations.

$$\mathbf{R}_{GNSS,k} = \alpha_2 \begin{bmatrix} \sigma^2_{\text{GNSS Latitude},k} & 0 & 0 \\ 0 & \sigma^2_{\text{GNSS Longitude},k} & 0 \\ 0 & 0 & \sigma^2_{\text{GNSS Heading},k} \end{bmatrix}. \tag{3.31}$$

The LiDAR-based positioning algorithm, from Section 3.3, is designed to match features. Therefore, the algorithm's performance was expected to improve when there were many features available. The number of features available was assumed to correlate with the number of points in the LiDAR point cloud. The number of points at the time instance k is denoted as N_k and the maximum amount of points the LiDARs can capture is denoted as the constant N_m. As described in section

3.1.4, there were 5 LiDARs on the vessel. Using the LiDARs FoV and resolution together with the number of vertical lines, the maximum number of points could be calculated, see equation (3.32).

$$N_m = 5 \cdot \left(\frac{360}{0.2} \cdot 32 \right) = 288000 \tag{3.32}$$

The measurement covariance of the LiDAR-based update was then set with the tuning constant α_3 as

$$\mathbf{R}_{LiDAR,k} = \alpha_3 \begin{bmatrix} \left(\frac{N_m}{10N_k+1} \right)^2 & 0 & 0 \\ 0 & \left(\frac{N_m}{10N_k+1} \right)^2 & 0 \\ 0 & 0 & 10 \end{bmatrix}. \tag{3.33}$$

With the two measurement covariance matrices and the measurement model defined, the update step was performed via equations (2.24) - (2.28). The update step with LiDAR measurements was set to update with 1 Hz, since the PSO and cross-correlation steps in Algorithm 5 were computationally heavy. To further decrease the computational load of Algorithm 5, the algorithm was adjusted to only load new aerial images when the vessel was estimated to have moved 5 meters away from the position where the previous image was loaded. The update step with GNSS sensor measurements was done with a 10 Hz frequency.

The Kalman filtering process is summarized in Algorithm 6. The if statement in the LiDAR update part of the algorithm was introduced because the stochastic optimization method PSO did not always converge to the global optimum. To overcome this limitation, a maximum allowable distance was implemented to assess the update distance prior to incorporating measurements into the update step of the EKF. The maximum distance was set to 2 meters. This criterion served as an indicator that an unusually large updated distance might indicate that the PSO has found a local optimum, and the measurement should therefore not be used as an update. These measurements are referred to as false measurements. To simulate the vessel entering a GNSS-challenged environment from an environment with accurate GNSS signals, the Kalman filter prior x_0 was set with the true position and heading values during the initialization phase of the filtering,

Algorithm 6 Kalman filtering process

Initialize: $\mathbf{x}_0$ as the prior ▷ Initialize state vector
$\mathbf{P}_0 \leftarrow$ covariance matrix of $\mathbf{x}_0$ ▷ Initialize covariance matrix
$\mathbf{Q}_k \leftarrow$ covariance matrix of IMU noise ▷ Process covariance matrix
$\mathbf{R}_{GNSS,k} \leftarrow$ covariance of GNSS noise ▷ Measurement covariance matrix
$\mathbf{R}_{LiDAR,k} \leftarrow$ covariance of LiDAR noise ▷ Measurement covariance matrix
for $k = 1, 2, 3, \ldots$ **do**

Predict:
$\mathbf{u}_k \leftarrow$ IMU measurement ▷ Update control vector
$\hat{\mathbf{x}}_{k|k-1} \leftarrow \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)$ ▷ Predicted state estimate
$\mathbf{P}_{k|k-1} \leftarrow \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^\top + \mathbf{Q}_k$ ▷ Predicted covariance estimate

Update:
if GNSS measurement available **then** ▷ Kalman Filter Update
$\mathbf{y}_k \leftarrow$ GNSS measurement
$\mathbf{R}_{GNSS,k} \leftarrow$ GNSS measurement
$\mathbf{v}_k \leftarrow \mathbf{y}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}$
$\mathbf{S}_k \leftarrow \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\intercal + \mathbf{R}_{GNSS,k}$
$\mathbf{K}_k \leftarrow \mathbf{P}_{k|k-1}\mathbf{H}^\intercal \mathbf{S}_k^{-1}$
$\hat{\mathbf{x}}_{k|k} \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{v}_k$
$\mathbf{P}_{k|k} \leftarrow \mathbf{P}_{k|k-1} + \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\intercal$
end if
if LiDAR measurement available **then**
$\mathbf{X}_k \leftarrow$ LiDAR measurement ▷ LiDAR update
$\mathbf{y}_k \leftarrow$ Algorithm 5($\mathbf{X}_k$) ▷ Measurement from 3D point cloud
if update distance < max distance **then**
$N_k \leftarrow \#\mathbf{X}_k$ ▷ Update number of LiDAR points
$\mathbf{R}_{LiDAR,k} \leftarrow N_k$ ▷ Update covariance matrix, equation (3.33)
$\mathbf{v}_k \leftarrow \mathbf{y}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}$
$\mathbf{S}_k \leftarrow \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\intercal + \mathbf{R}_{LiDAR,k}$
$\mathbf{K}_k \leftarrow \mathbf{P}_{k|k-1}\mathbf{H}^\intercal \mathbf{S}_k^{-1}$
$\hat{\mathbf{x}}_{k|k} \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{v}_k$
$\mathbf{P}_{k|k} \leftarrow \mathbf{P}_{k|k-1} + \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\intercal$
end if
end if
if No new measurement available **then**
$\hat{\mathbf{x}}_{k|k} \leftarrow \hat{\mathbf{x}}_{k|k-1}$
$\mathbf{P}_{k|k} \leftarrow \mathbf{P}_{k|k-1}$
end if
end for

4
Results

This chapter shows the results of the system for the three different cases presented in Section 3.1.5. There are two different noise scenarios that were tested in each case. Also, some different outcomes of the positioning algorithm are presented.

4.1 Performance

The following tuning parameters were used for all the tests in this section: $\alpha_1 = 10$ $(\mathbf{Q}, \mathbf{k})$, $\alpha_2 = 10$ $(\mathbf{R}_{GNSS,k})$, $\alpha_3 = 0.5$ $(\mathbf{R}_{LiDAR,k})$. The figures below show the true and estimated positions obtained through the measurement signal. The measurement consists of the GNSS data with added noise, leading to signal drift. Two different scenarios were tested in all three cases. The first scenario consists of noise that has gradually drifted over time, resulting in a few meters offset from the true position, see Figures 4.1, 4.3 and 4.5. The position RMSE and max error for the measurement in the first scenario are listed in Table 4.1.

Table 4.1: Position RMSE and max error for measurement signal in the first scenario.

Position [m]	RMSE	Max error
Measurement (Case 1)	1.3567	1.7044
Measurement (Case 2)	1.3658	1.7044
Measurement (Case 3)	1.2847	1.5985

The second scenario consists of a slight reduction of noise but a sudden jump is introduced, representing the GNSS signal becoming temporarily lost, see Figures 4.2, 4.4 and 4.6. During the signal loss period, the tuning parameter for the measurement noise covariance matrix $\mathbf{R}_{GNSS,k}$ was increased to $\alpha_2 = 1000$. The position RMSE and max error for the measurement in the second scenario are listed in Table 4.2.

Table 4.2: Position RMSE and max error for measurement signal in the second scenario.

Position [m]	RMSE	Max error
Measurement (Case 1)	2.4596	14.5443
Measurement (Case 2)	2.4428	14.5115
Measurement (Case 3)	3.6406	9.9748

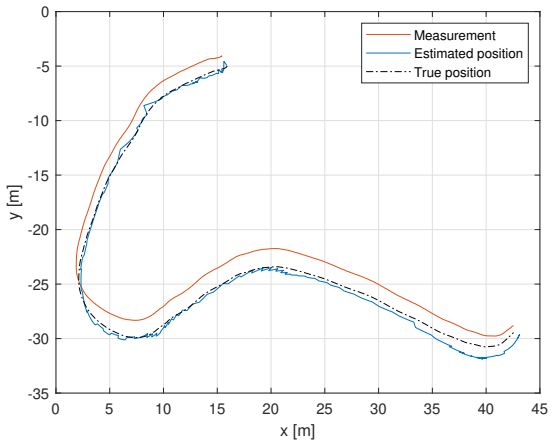

Figure 4.1: True trajectory of the vessel, measurements, and the result of the Kalman estimation for case 1. The GNSS data without noise is used as prior.

Table 4.3: RMSE and max error for estimation from run in Figure 4.1.

State	RMSE	Max error
Position [m]	0.5893	1.6668
Heading [°]	1.7332	4.8879

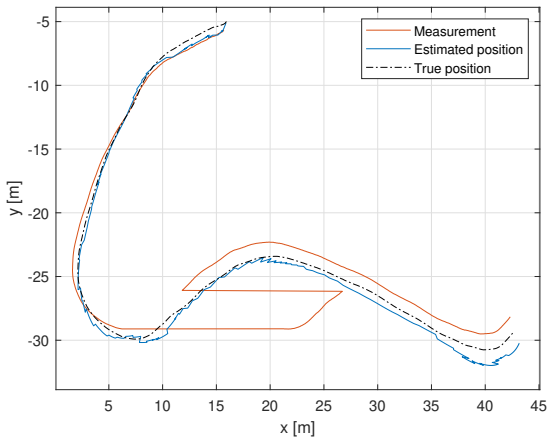

Figure 4.2: True trajectory of the vessel, measurements with a sudden jump, and the result of the Kalman estimation for case 1. The GNSS data without noise is used as prior.

Table 4.4: RMSE and max error for estimation from run in Figure 4.2.

State	RMSE	Max error
Position [m]	0.7241	1.8233
Heading [°]	1.7118	4.9983

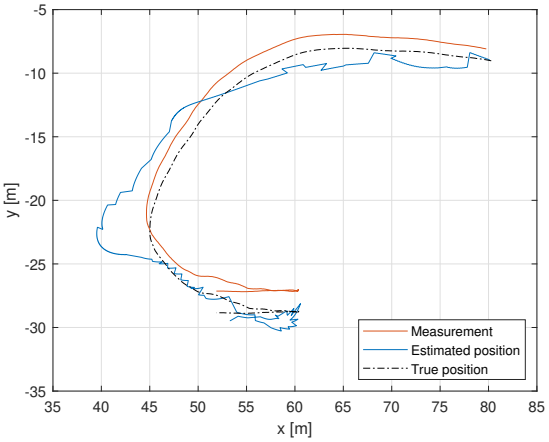

Figure 4.3: True trajectory of the vessel, measurements, and the result of the Kalman estimation for case 2. The GNSS data without noise is used as prior.

Table 4.5: RMSE and max error for estimation from run in Figure 4.3.

State	RMSE	Max error
Position [m]	1.7394	5.5664
Heading [°]	1.6366	3.8959

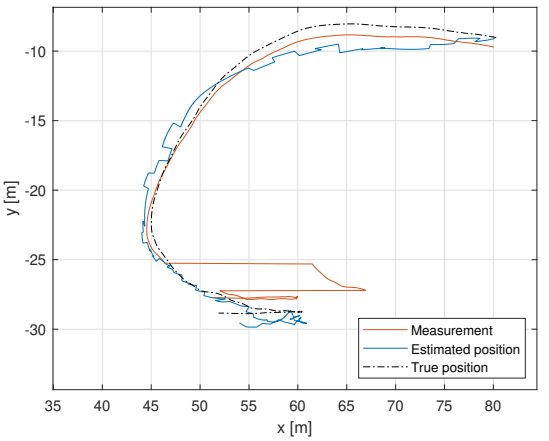

Figure 4.4: True trajectory of the vessel, measurements with a sudden jump, and the result of the Kalman estimation for case 2. The GNSS data without noise is used as prior.

Table 4.6: RMSE and max error for estimation from run in Figure 4.4.

State	RMSE	Max error
Position [m]	1.2505	2.8422
Heading [°]	1.5556	3.9108

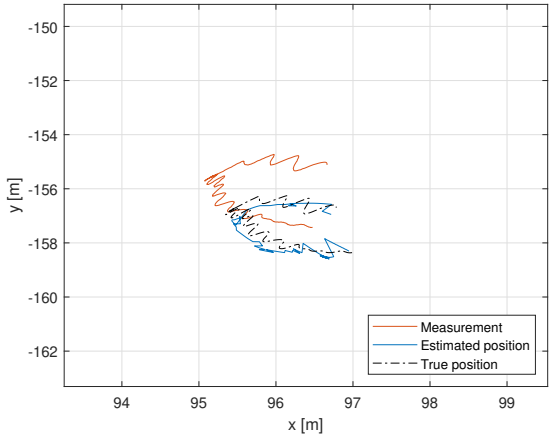

Figure 4.5: True trajectory of the vessel, measurements, and the result of the Kalman estimation for case 3. The GNSS data without noise is used as prior.

Table 4.7: RMSE and max error for estimation from run in Figure 4.5.

State	RMSE	Max error
Position [m]	0.2901	0.9984
Heading [°]	2.3374	5.0670

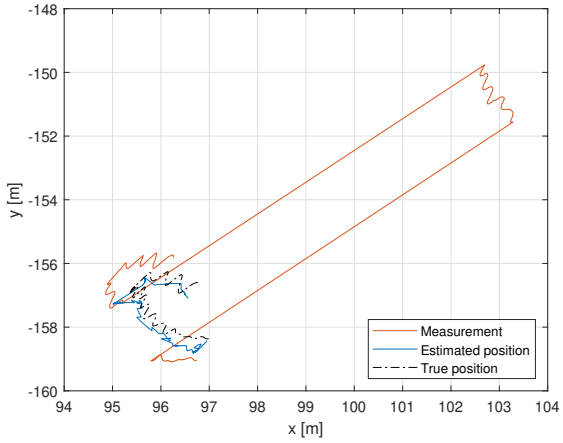

Figure 4.6: True trajectory of the vessel, measurements with a sudden jump, and the result of the Kalman estimation for case 3. The GNSS data without noise is used as prior.

Table 4.8: RMSE and max error for estimation from run in Figure 4.6.

State	RMSE	Max error
Position [m]	0.3500	1.1214
Heading [°]	1.8218	3.2051

4.2 Positioning algorithm

Some different outcomes of the positioning algorithm, Algorithm 5 presented in Section 3.3, can be seen below. The performance of the algorithm was very dependent on the environment and LiDAR data. Figure 4.7 showcases a good performance, while Figures 4.8 and 4.9 showcase a more challenging scenario where it performed worse. Figure 4.10 shows an instance from case 3, which is a relatively uncomplicated environment with good conditions. However, the global best transformation that was found is incorrect. The time requirement of the algorithm was 4.80 sec-

onds when a new aerial image was downloaded and 3.59 seconds when the previously loaded aerial image was reused.

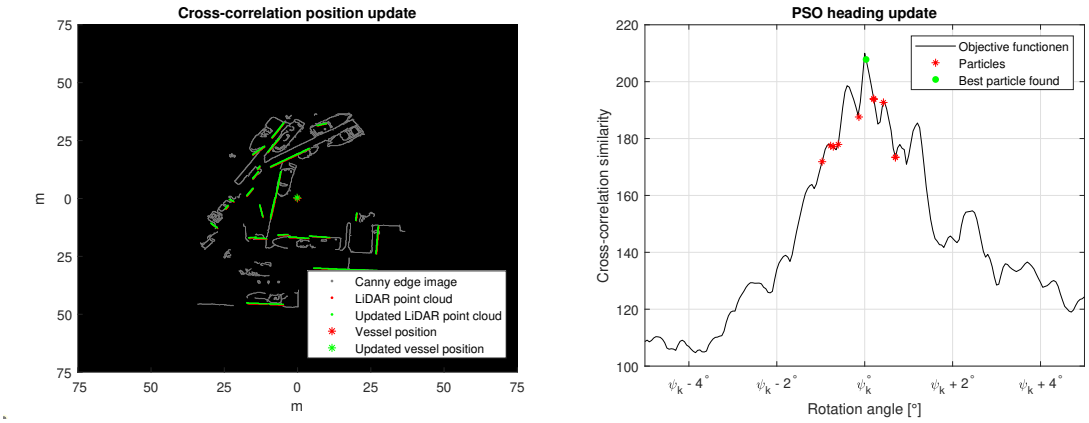

Figure 4.7: An instance from case 1, the first subfigure (left) shows the new alignment of the point cloud (green) after finding the best transformation of the input point cloud (red) with the extracted features from the aerial image (grey). The second subfigure (right) shows the PSO that has found the global best rotation angle with the objective function $f(x)$ as the negative cross-correlation between extracted features and the rotated point cloud. Each particle, for the last iteration, is denoted by red marks, while the green mark indicates the global best position that has been found.

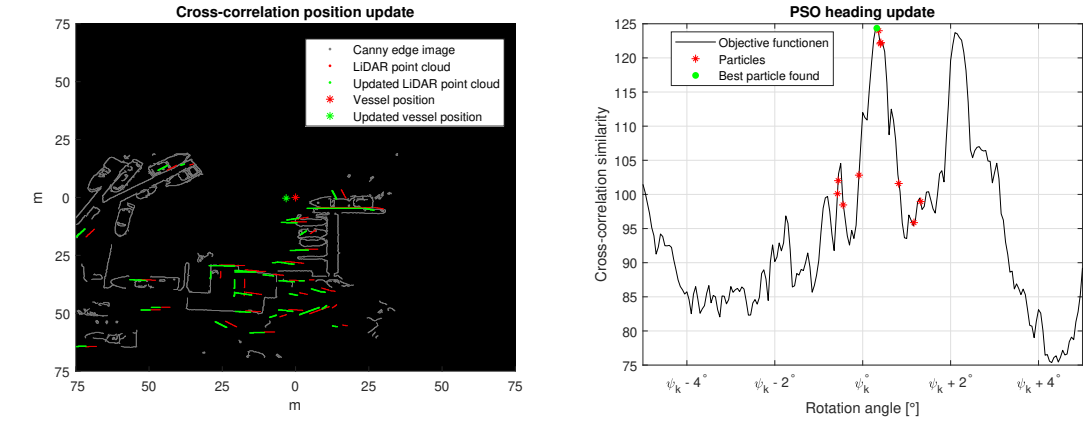

Figure 4.8: An instance from case 2, the first subfigure (left) shows the new alignment of the point cloud (green) after finding a transformation of the input point cloud (red) with the extracted features from the aerial image (grey). The second subfigure (right) shows the PSO that has found the global best rotation angle (green mark) with the objective function $f(x)$ as the negative cross-correlation between extracted features and the rotated point cloud.

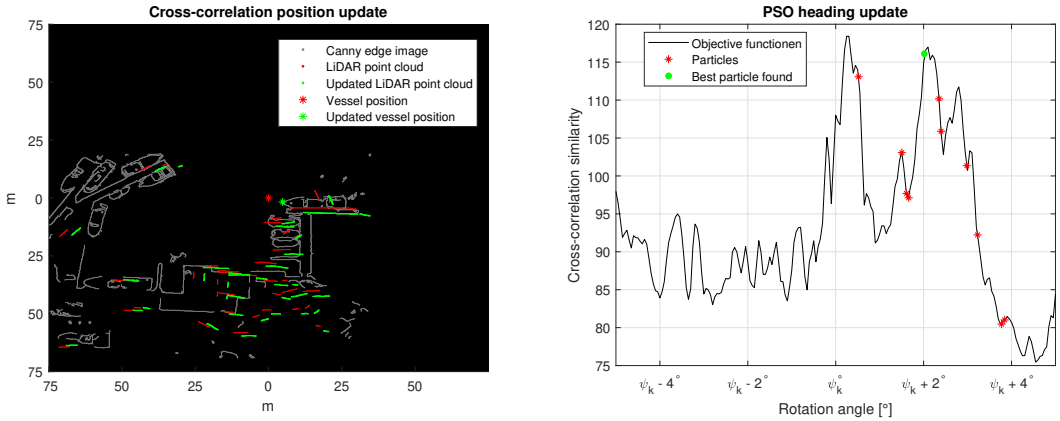

Figure 4.9: Same instance as in Figure 4.8, the first subfigure (left) shows the new alignment of the point cloud (green) after finding another valid transformation of the input point cloud (red) with the extracted features from the aerial image (grey). The second subfigure (right) shows the PSO that has found a local best rotation angle (green mark) with the objective function $f(x)$ as the negative cross-correlation between extracted features and the rotated point cloud.

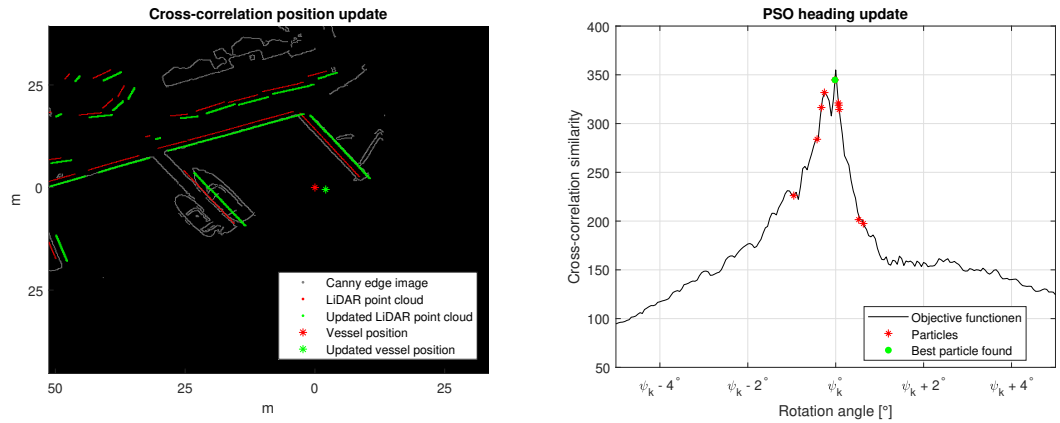

Figure 4.10: An instance from case 3, the first subfigure (left) shows the new alignment of the point cloud (green) after finding the best transformation of the input point cloud (red) with the extracted features from the aerial image (grey). The second subfigure (right) shows the PSO that has found a global best rotation angle (green mark) with the objective function $f(x)$ as the negative cross-correlation between extracted features and the rotated point cloud.

5
Discussion

In this chapter, the results collected are discussed based on the research questions. The methods used to achieve the results presented are also discussed and their performance is evaluated.

5.1 Estimated position and heading performance

As stated in the first research question, this thesis has investigated how integrating aerial imagery improves a system's robustness in case of characteristic GNSS signal disturbance. The presented results show that while the proposed system enhances robustness in certain scenarios, it does not have the same effect in others. To get a better understanding of the system, each case is discussed individually in the subsections below. The discussion also involves how the LiDAR data is matched with the aerial imagery for the estimation of the position and heading, which is the second research question of this thesis.

5.1.1 Case 1

The two scenarios that were tested in case 1 both demonstrated that the estimation was an improvement of the position and heading. It can be seen in Figures 4.1 and 4.2 that the estimation is close to the ground truth. The values in Tables 4.3 and 4.4 also indicate this. The position RMSE goes from 1.36m down to 0.59m for scenario 1, and from 2.46m down to 0.72m for scenario 2. The position max error was also reduced, especially in scenario 2 where the sudden jump was imposed. Figure 4.7 showcases why the system works well for this case. There are many lines, both vertically and horizontally, for the algorithm to use and match with the features from the aerial image. Therefore, a distinct global maximum exists for the rotation angle that produces a high cross-correlation similarity.

5.1.2 Case 2

The performance of the system in a less favorable environment can be seen in Figures 4.3 and 4.4. The results regard test case 2, where the values in Table 4.5 show that the estimation was a deterioration of the position and heading. The position RMSE was increased from 1.37m to 1.74m and the max error from 1.70m to 5.57m in scenario 1. The second scenario, regarding Table 4.6, shows a slight improvement in the estimation but not as good as for case 1. The reason behind the less

accurate estimation in this case is because the system encounters difficulties during the initial one-third of the path. Figures 4.8 and 4.9 show two possible results of the positioning algorithm during the initial phase of the path. In this instance, it is a more challenging scenario where there are no clear vertical lines after the point cloud has been processed, making it difficult to fix the new alignment in the horizontal direction. This is also noticeable in the PSO subfigure, where there are two local maximums. This makes the LiDAR update very sporadic and unreliable. Therefore, it is often classified as a false measurement and is barely used during the initial phase of the path, contributing to the inaccurate estimation.

In Table 3.1, the LiDAR range is specified as 100m. However, upon examining Figures 4.8 and 4.9, it becomes apparent that the LiDARs have not detected any objects on the left side of the image, even though the dock on the left side is only about 50m away from the vessel. This is because the LiDAR points on the left side are classified as below water level, and are subsequently filtered out. In general, when a vessel makes a sharp turn, it tends to lean inward toward the center of the turn. In case 2 the vessel is turning left, making the horizontal plane, that should tangent the water surface, instead point upwards in the direction of the dock. Therefore, the algorithm perceives that the LiDAR points in that area are below the water surface and are removed. To compensate for this and ensure that the LiDAR points are retained, it would have been beneficial to introduce roll as a state in the EKF. In this way, more vertical lines would exist in the LiDAR point cloud and improve the performance of the system for case 2.

5.1.3 Case 3

Case 3 is more unique since the vessel stays at the same location and the recording duration is shorter than for cases 1 and 2. It is the time duration together with the good performance that makes the RMSE values, Tables 4.7 and 4.8, so low compared to the other cases. Figure 4.10 depicts an instance from case 3, where the point cloud consists of a good variety of lines. However, it is important to note that a false measurement is observed in this particular instance, which is discussed in Section 5.2. The incorporation of roll as a state in addition to the variety in lines could have further improved the estimation performance. More information about the motion of the vessel would have let the filter better adapt to the presence of ”waves” and adjust the estimated position and heading.

5.2 Supplementary validation of positioning algorithm

As shown in Section 4.2, and discussed above, the positioning algorithm worked well in some cases and worse in others. Regarding research question three, objects that are only present in the aerial image had a minimal to negligible impact on the system in the majority of situations. This can for example be seen in Figure 4.7, where a few docked vessels are visible as extracted features in the aerial image. However,

in the LiDAR point cloud, only the dock is visible but it still accomplished the task of identifying the correct update. Nonetheless, it does exist specific circumstances when this has an effect on the system. Figure 4.10 showcase a false measurement where the updated LiDAR point cloud aligns with the wrong side of the dock (left side aligns with right side). Usually when this happens, other parts of the LiDAR point cloud would not align with anything, therefore decreasing the cross-correlation. But in this instance, instead of not aligning with anything the other end of the dock happens to align really well with the vessel from the aerial image.

The configuration of the PSO, as described in Section 3.3.5, appears to be a good balance between speed and performance. The chosen search space of $[-5°, 5°]$ also proved to be suitable, given that the maximum error observed during testing was 5.07°.

5.3 Extended Kalman filtering

The fusing of measurements proved to work well for producing a state estimation and the EKF can be used to answer the fourth research question. When the EKF was provided with errors in the GNSS measurements, the measurement covariance $\mathbf{R}_{GNSS,k}$ increased simultaneously with the GNSS sensors measured RMSE. The EKF then performed state estimation with more respect to the IMU and LiDAR measurements, resulting in a trajectory close to the ground truth. The filtering worked well when GNSS measurement errors were handled and propagated but it worked worse for false LiDAR measurements.

The trajectories of the estimated position can be seen taking sudden sharp jumps. These sharp jumps were produced by the EKFs update step with LiDAR measurements which indicates that the LiDAR measurement covariance $\mathbf{R}_{LiDAR,k}$ was modeled with low variances compared to the other sensors measurement. This proved to work well for the estimation in cases 1 and 3 but worse in case 2. When Algorithm 5 provided false measurements as in case 2, the estimation can be seen deviating to the left of the true position in Figure 4.3. The GNSS and IMU measurements addressed this so the estimated trajectory returned to align with the true position.

Since the estimated trajectory deviated when the EKF was provided with false measurements, the modeling of the LiDAR measurement covariance $\mathbf{R}_{LiDAR,k}$ can be assumed to not work as intended. In section 3.4.2, the assumption that "the number of matchable features depends on the size of the point cloud" can still be assumed to be true. However, since it was discovered that false measurements occur independent of the size of the point cloud, the accuracy of the LiDAR measurement should not have been assumed to be only dependent on the number of features. An implementation that could have addressed the issue of inaccurate modeling of LiDAR measurement covariance would be to set variance values dependent on the cross-correlation measured similarity instead, equation (3.16). The values of the cross-correlation measured similarities can be seen in Figure 4.7 - 4.10. In cases 1 and 3, it can be seen that higher values were used for the PSO compared to case 2.

Since the EKFs LiDAR update step performed better in cases 1 and 3, implementing a measured cross-correlation similarity into LiDAR measurement covariance could handle and propagate errors and uncertainties for more reliable results.

Another implementation that could increase the performance of the LiDAR update step would be to model measurement covariance $\mathbf{R}_{LiDAR,k}$ with respect to the number of horizontal and vertical lines. As discussed in Section 5.1.1 - 5.1.2, the performance of the cross-correlation position update and the PSO heading update decreased when there was a shortage of either horizontal or vertical lines as in case 2. This could be used to model the measurement covariance so that the uncertainty should be increased if there was mostly one type of line, as in case 2.

The extended Kalman filtering was restricted to a 2D motion model. This was shown to work for improving the position and heading estimation of the vessel in GNSS-challenged environments. However, a more accurate and correct way of implementing the EKF would have been to model it for 3D. As discussed in Section 5.1.2 - 5.1.3, implementing roll in the state estimation could have improved the performance of the system. If more states were used in the EKF, even better results could have been found, such as in the implementation of [5] which was covered in Section 1.2.

5.4 Data collection

There is a limitation in terms of the amount of data collected for this thesis. There were only three different data sets that have been used for evaluation, and each data set contributed valuable insight into how the system works. The limited data impacted the ability to fully evaluate the system. Where if additional testing could have been done, potentially greater insights could have been obtained.

As presented earlier, the GNSS signal without the added noise has been used as the ground truth. First of all, it would generally be more appropriate to use an additional and independent GNSS sensor when evaluating the system. Secondly, the GNSS measurements by itself have some error characteristics. In Section 3.1.3 it is presented that the position and heading RMSE accuracies are 0.3m and 1.5° respectively for the GNSS sensor used. Therefore, the designed system could output the correct position and heading but still generate a high RMSE, since the ground truth could be incorrect. One solution to this could have been to equip the vessel with an additional GNSS source, that uses real-time kinematic positioning (RTK) for instance. This would ensure that the reference data is obtained from a highly accurate and precise positioning system. Consequently, the obtained results would better reflect the actual performance of the system.

The LiDAR sensors used, RS-Bpearl and RS-Helios worked well in capturing the surrounding environment of the vessel. The IMU also performed well in measuring forces and angular velocities. For better accuracy regarding the IMU measurement, the IMU could have been placed in the vessel's center of rotation.

5.5 Real-time applications

A challenge when implementing a localization system for real-time using IMU, GNSS and LiDAR sensors is the computational load of how the LiDAR data is handled to procure a state measurement. The positioning algorithm, which produces state measurements from the LiDAR data, was set to a computing frequency of 1 Hz. This was slower than the computing time required, therefore the state measurements were not computed fast enough. The time duration for Algorithm 5 can be lowered with the use of high-performing programming languages and code optimization. However, the computer specification in real-time applications would be less powerful.

Filtering and reducing the number of points in the LiDAR data would lower the computational load of the algorithm however it could discard important features. The PSO in the algorithm could be optimized for a lesser computational load which can be achieved by using fewer particles and fewer iterations. This would make the algorithm faster but less accurate since it would be more likely to get stuck at local maximums which had a higher probability of yielding false state estimations. A better approach for making the algorithm more suitable for real-time application would be to keep the PSO optimized for accuracy rather than computation speed and have the EKF update with respect to LiDAR measurements at a lower frequency. Another implementation that could make the PSO yield higher accuracy without increasing PSO particles or iterations would be to use an adaptive search space of interest. As the EKF computes the posterior covariance $\mathbf{P}_{k|k-1}$, this uncertainty could be used to define the search space of the PSO. During low values of posterior covariance, the PSO could be used to fine-tune the heading estimation and during high values, the PSO would be used to approximately estimate the heading.

6
Conclusion

In this thesis, a system that estimates the position and heading with the use of aerial imagery, LiDAR sensors, an IMU sensor and a GNSS sensor, was constructed. The method of how the presented systems use LiDAR data to estimate position and heading can be described as, firstly, an aerial image that has captured the surrounding environment is transformed via Canny edge detection. Secondly, the LiDAR data is projected to a top view and filtered so overlapping points are extracted and used for RANSAC line estimation. The lines and the Canny edge transformed aerial image are then matched using cross-correlation. PSO is used to optimize the heading that maximizes the cross-correlation similarity. The found transformation to the solution is used to update the position estimate. Finally, the position and heading estimate is fused with measurements from an IMU sensor and a GNSS sensor in an EKF to get a more precise estimation. The system's performance was then tested on the sensor data from 6 different scenarios. The testing showed that the system could estimate a position with higher precision compared to the GNSS sensor in a GNSS-challenged environment and that it performed well in estimating the heading. The proposed system showed potential to act as a reliable source of estimating position and heading if the implementation described in future work is made.

6.1 Future work

The modeling of the motion model used in the EKF should be revised since there were indications that the performance of the position algorithm degraded when the vessel was orientated with increased roll magnitude. Implementing an estimation of the roll to the state vector would therefore be an improvement. Other states can also be implemented such as pitch and acceleration noise. Another implementation that can improve the system precision and computational load would be to use the EKFs estimation of the heading covariance to define the search space of the PSO.

As the system relies on parameters that can be tuned such as the RANSAC grid size, overlapping point threshold, PSO parameters and noise covariances, the system could be optimized for better precision. A tuning parameter that should be revised would be the LiDAR covariance $\mathbf{R}_{LiDAR,k}(N_k)$. A better approach than the presented method of modeling the LiDAR covariance would be to make it dependent on the ratio between 2D lines orthogonal to each other. The algorithms parameters could also be tuned for computational efficiency by lowering the number of PSO particles and iterations as well as down-sampling of the LiDAR point cloud.

The computational load could also be lowered by programming the system in a high-performing program language. However, as the aerial images were provided by Matlab, another source of map-database must be used.

References

[1] T. Hoang and V. P. Berntsson, "Localisation using lidar and camera," Master's thesis, Chalmers University of Technology, 2017.

[2] G. Wan, X. Yang, R. Cai, H. Li, Y. Zhou, H. Wang, and S. Song, "Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[3] N. Dalhaug, "Lidar-based localization for autonomous ferry," Master's thesis, Norwegian University of Science and Technology, 2019.

[4] F. Moosmann and C. Stiller, "Velodyne slam," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 393–398.

[5] A. Engsträm and D. Geiseler, "Lidar-based simultaneous localization and mapping in marine vehicles," Master's thesis, Chalmers University of Technology, 2022.

[6] T. Liu, B. Li, L. Yang, G. Chen, L. He, and J. He, "Tightly coupled gnss, ins and visual odometry for accurate and robust vehicle positioning," in *2022 5th International Symposium on Autonomous Systems (ISAS)*, 2022.

[7] O. Pink, "Visual map matching and localization using a global feature map," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008.

[8] Z. Zhao, H. Fu, R. Ren, and Z. Sun, "Real-time intersection detection based on satellite image and 3d lidar point cloud," in *Proceedings of 2021 International Conference on Autonomous Unmanned Systems (ICAUS 2021)*, M. Wu, Y. Niu, M. Gu, and J. Cheng, Eds. Springer Singapore, 2022, pp. 2868–2878.

[9] M. Escourrou, J. A. Hage, and P. Bonnifait, "Ndt localization with 2d vector maps and filtered lidar scans," in *2021 European Conference on Mobile Robots (ECMR)*, 2021.

[10] E. Javanmardi, M. Javanmardi, Y. Gu, and S. Kamijo, "Autonomous vehicle self-localization based on multilayer 2d vector map and multi-channel lidar," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 437–442.

[11] E. Javanmardi, Y. Gu, M. Javanmardi, and S. Kamijo, "Autonomous vehicle self-localization based on abstract map and multi-channel lidar in urban area," *IATSS Research*, vol. 43, no. 1, pp. 1–13, 2019.

[12] M. Khader and S. Cherian, "An introduction to automotive lidar," Texas Instruments, accessed: May 24, 2023. [Online]. Available: https://www.ti.com/lit/wp/slyy150a/slyy150a.pdf?ts=1676608487924

[13] S. Harris, "Inertial measurement unit (imu) – an introduction . [online]." 2023, accessed: May 24, 2023. [Online]. Available: https://www.advancednavigation.com/tech-articles/inertial-measurement-unit-imu-an-introduction/

[14] V. Passaro, A. Cuccovillo, L. Vaiani, M. D. Carlo, and C. E. Campanella, "Gyroscope technology and applications: A review in the industrial perspective." *Sensors*, vol. 17, 2017.

[15] M. Dadafshar, "Accelerometer and gyroscopes sensors: Operation, sensing, and applications," 2014, accessed: May 24, 2023. [Online]. Available: https://www.analog.com/en/technical-articles/accelerometer-and-gyroscopes-sensors-operation-sensing-and-applications.html

[16] "Icao. global navigation satellite system (gnss) manual," 2005, accessed: May 24, 2023. [Online]. Available: https://www.icao.int/Meetings/PBN-Symposium/Documents/9849_cons_en%5B1%5D.pdf

[17] "Gps," nationalgeographic, accessed: May 24, 2023. [Online]. Available: https://education.nationalgeographic.org/resource/gps/

[18] M. Petovello and S. Gaglione, "How does a gnss receiver estimate velocity?" 2015, accessed: May 24, 2023. [Online]. Available: https://insidegnss.com/wp-content/uploads/2018/01/marapr15-SOLUTIONS.pdf

[19] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Springer, 2009.

[20] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[21] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, 1986.

[22] R. C. Bolles and M. A. Fischler, "A ransac-based approach to model fitting and its application to finding cylinders in range data," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, p. 637–643.

[23] "2-d cross-correlation," MathWorks, accessed: May 24, 2023. [Online]. Available: https://se.mathworks.com/help/signal/ref/xcorr2.html

[24] E. Weisstein, "Cross-correlation theorem," MathWorld–A Wolfram incollection Resource., accessed: May 24, 2023. [Online]. Available: https://mathworld.wolfram.com/Cross-CorrelationTheorem.html

[25] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[26] M. Wahde, *Biologically Inspired Optimization methods: An Introduction*. WIT Press, 2008.

[27] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge University Press, 2013, vol. 3.

[28] S.-L. Sun and Z.-L. Deng, "Multi-sensor optimal information fusion kalman filter," *Automatica*, vol. 40, pp. 1017–1023, 2004.

[29] J. Daintith, *A Dictionary of Physics*, 6th ed. Oxford University Press, 2016.

[30] "Inertial measurement unit," CPAC Systems AB, accessed: May 24, 2023. [Online]. Available: https://playground.cpacsystems.se/en/inertial-measurement-unit/
[31] "V200s vector user guide," Hemisphere, 2021, accessed: May 24, 2023. [Online]. Available: https://www.hemispheregnss.com/wp-content/uploads/2022/01/875-0395-10_a6-v200s-vector-gnss-compass-user-guide.pdf.
[32] European Space Agency, "Accuracy," ESA GNSS Science Support Centre (GSSC) Navipedia, accessed: 05 24, 2023. [Online]. Available: https://gssc.esa.int/navipedia/index.php/Accuracy
[33] "Rs-bpearl," RoboSense, accessed: May 24, 2023. [Online]. Available: https://www.robosense.ai/en/rslidar/RS-Bpearl
[34] "Rs-helios," RoboSense, accessed: May 24, 2023. [Online]. Available: https://www.robosense.ai/en/rslidar/RS-Helios
[35] J. S. Subirana, J. J. Zornoza, and M. Hernández-Pajares, "Gnss measurement features and noise," 2011, accessed: May 24, 2023. [Online]. Available: https://gssc.esa.int/navipedia/index.php/GNSS_Measurement_features_and_noise
[36] ——, "Receiver noise," 2011, accessed: May 24, 2023. [Online]. Available: https://gssc.esa.int/navipedia/index.php/Receiver_noise
[37] "Mapping toolbox," Mathworks, accessed: May 24, 2023. [Online]. Available: https://se.mathworks.com/help/map/index.html?s_tid=CRUX_topnav
[38] "Edge detection," MathWorks, accessed: May 24, 2023. [Online]. Available: https://se.mathworks.com/help/images/ref/edge.html
[39] A. Becker, *Kalman Filter: From the Ground Up*, 2023.

A

Appendix

The transformation matrix for each LiDAR, IMU and GNSS. Transformation $T_B^{L_1}$ represents transformation from LiDAR L_1 to base B.

$$T_B^{L_1} = \begin{bmatrix} 0.0008 & -0.0002 & -1.0000 & -7.1700 \\ -0.0010 & -1.0000 & 0.0002 & 0.4800 \\ -1.0000 & 0.0010 & -0.0008 & 0.3960 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$T_B^{L_2} = \begin{bmatrix} 0.2202 & -0.9699 & 0.1039 & 3.7887 \\ -0.8345 & -0.2425 & -0.4948 & -1.8122 \\ 0.5051 & 0.0222 & -0.8628 & 1.0779 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$T_B^{L_3} = \begin{bmatrix} 0.2393 & 0.9662 & 0.0957 & 3.5665 \\ 0.7856 & -0.2506 & 0.5657 & 1.8901 \\ 0.5706 & -0.0602 & -0.8190 & 1.0172 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$T_B^{L_4} = \begin{bmatrix} -0.0220 & -0.9977 & 0.0648 & -3.2191 \\ 0.9727 & -0.0064 & 0.2320 & 1.4394 \\ -0.2310 & 0.0681 & 0.9706 & 2.3910 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$T_B^{L_5} = \begin{bmatrix} 0.0004 & 0.9995 & 0.0311 & -3.1667 \\ -0.9838 & 0.0060 & -0.1791 & -1.4083 \\ -0.1792 & -0.0305 & 0.9833 & 2.3929 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$T_B^{GNSS} = \begin{bmatrix} 1.0000 & 0 & 0 & -2.2078 \\ 0 & 1.0000 & 0 & 0.4281 \\ 0 & 0 & 1.0000 & -2.2789 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$R_B^{IMU} = \begin{bmatrix} 0 & -1.0000 & 0 \\ 1.0000 & 0 & 0 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

B

Appendix

Visualizations of the vessels coordinate system and the sensors coordinate systems.

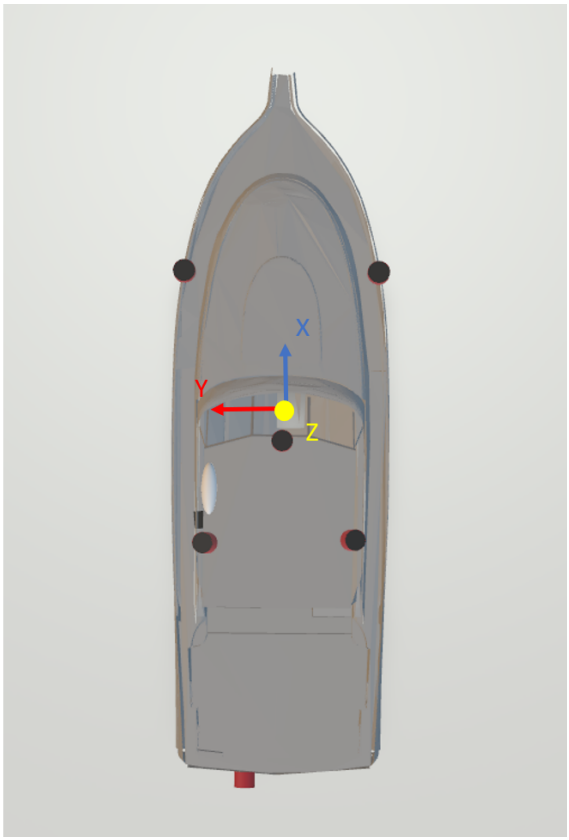

Figure B.1: Visualisation of the vessels coordinate system.

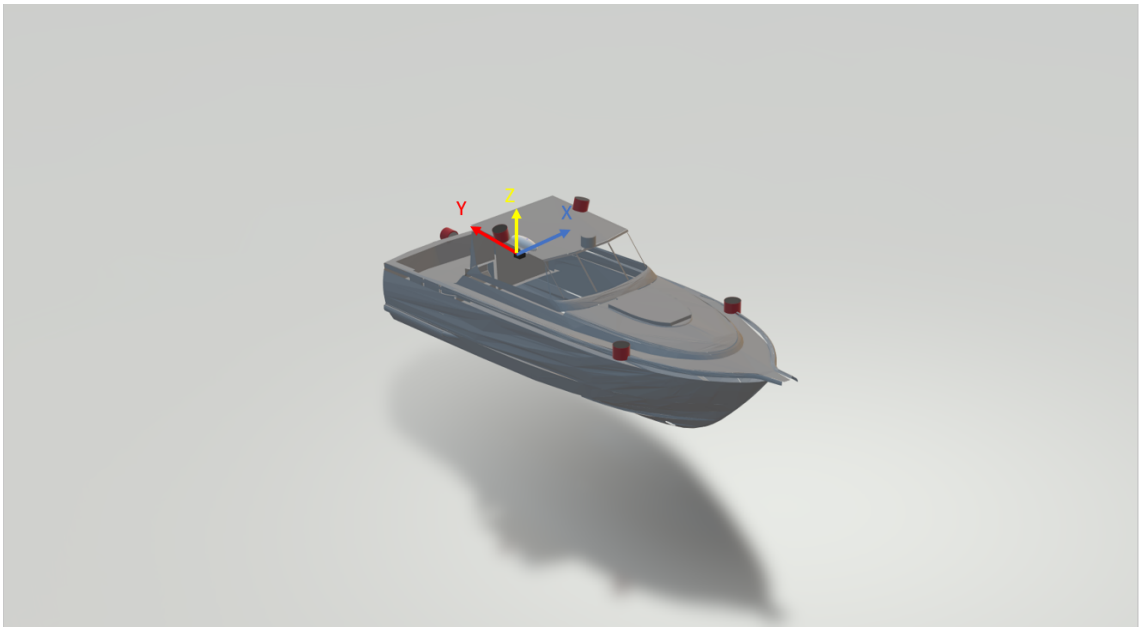

Figure B.2: Visualisation of IMU position and its coordinate system.

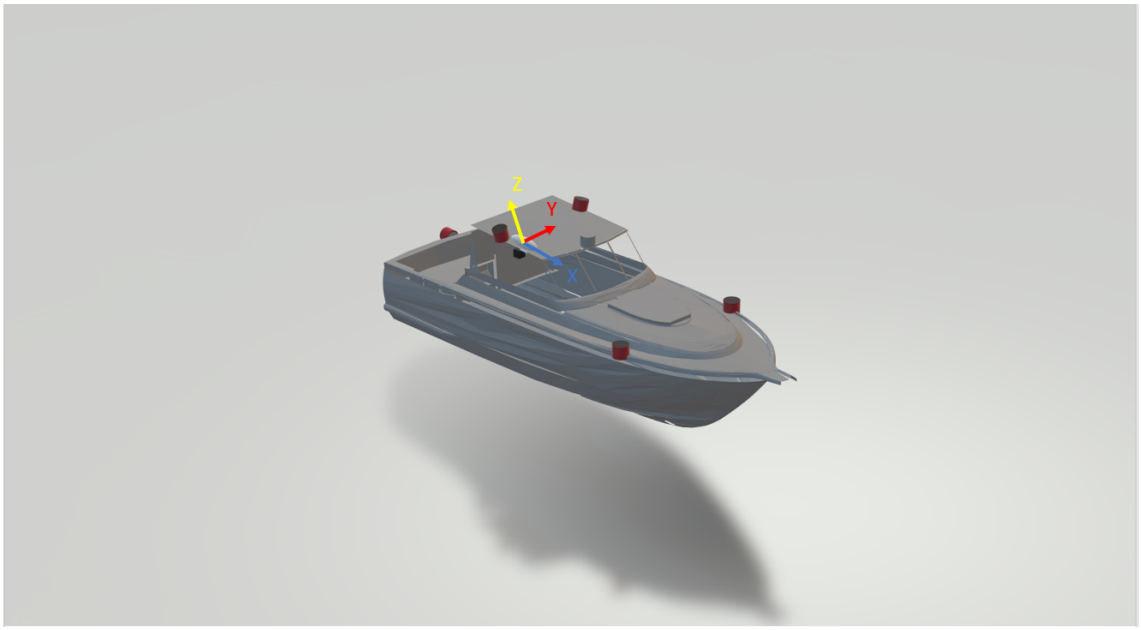

Figure B.3: Visualisation of GNSS position and its coordinate system.

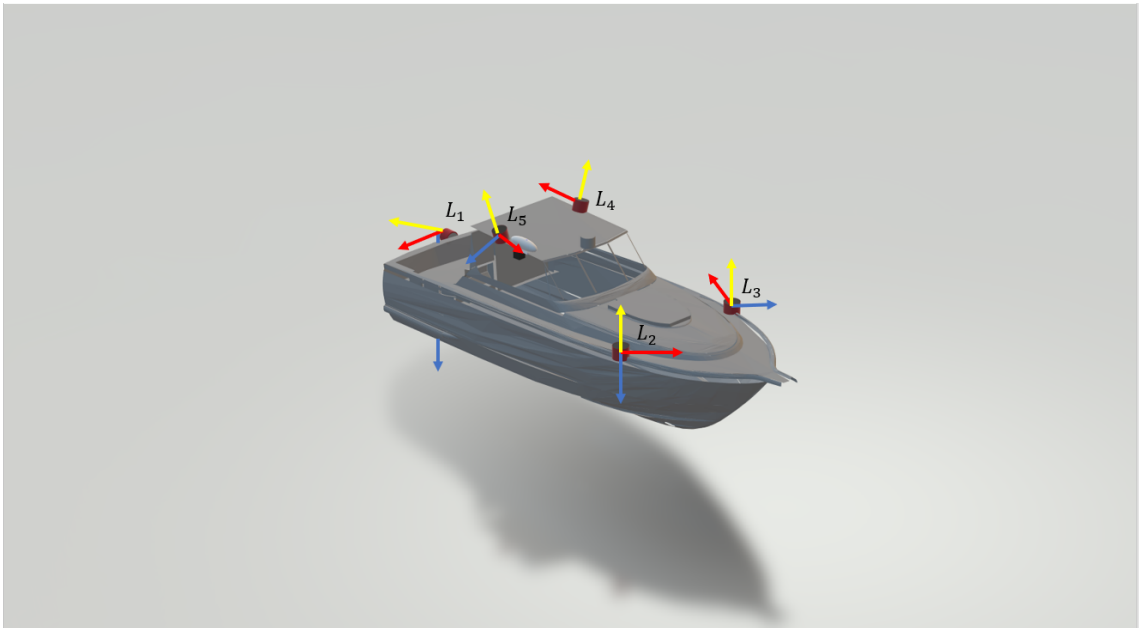

Figure B.4: Visualisation of LiDARs positions and coordinate systems; blue x-axis, red y-axis and yellow z-axis.

C

Appendix

The process noise matrix $\mathbf{Q}_k$.

$$
\mathbf{Q}_k = \alpha_1 \begin{bmatrix}
\delta^3 \sin(\psi_k)^2 \sigma_{Acc}^2 & \cos(\psi_k)\delta^3 \sin(\psi_k)\sigma_{Acc}^2 & \cos(\psi_k)\delta^2 \sigma_{Acc}^2 & -\delta^2 \sin(\psi_k)\sigma_{Acc}^2 & 0 \\
\cos(\psi_k)\delta^3 \sin(\psi_k)\sigma_{Acc}^2 & \cos(\psi_k)^2 \delta^3 \sigma_{Acc}^2 + \delta^3 \sin(\psi_k)^2 \sigma_{Acc}^2 & \delta^2 \sin(\psi_k)\sigma_{Acc}^2 & \cos(\psi_k)\delta^2 \sigma_{Acc}^2 & 0 \\
\cos(\psi_k)\delta^2 \sigma_{Acc}^2 & \delta^2 \sin(\psi_k)\sigma_{Acc}^2 & \delta\sigma_{Acc}^2 & 0 & 0 \\
-\delta^2 \sin(\psi_k)\sigma_{Acc}^2 & \cos(\psi_k)\delta^2 \sigma_{Acc}^2 & 0 & \delta\sigma_{Acc}^2 & 0 \\
0 & 0 & 0 & 0 & \delta\sigma_{Gyro}^2
\end{bmatrix}
$$

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

CHALMERS
UNIVERSITY OF TECHNOLOGY