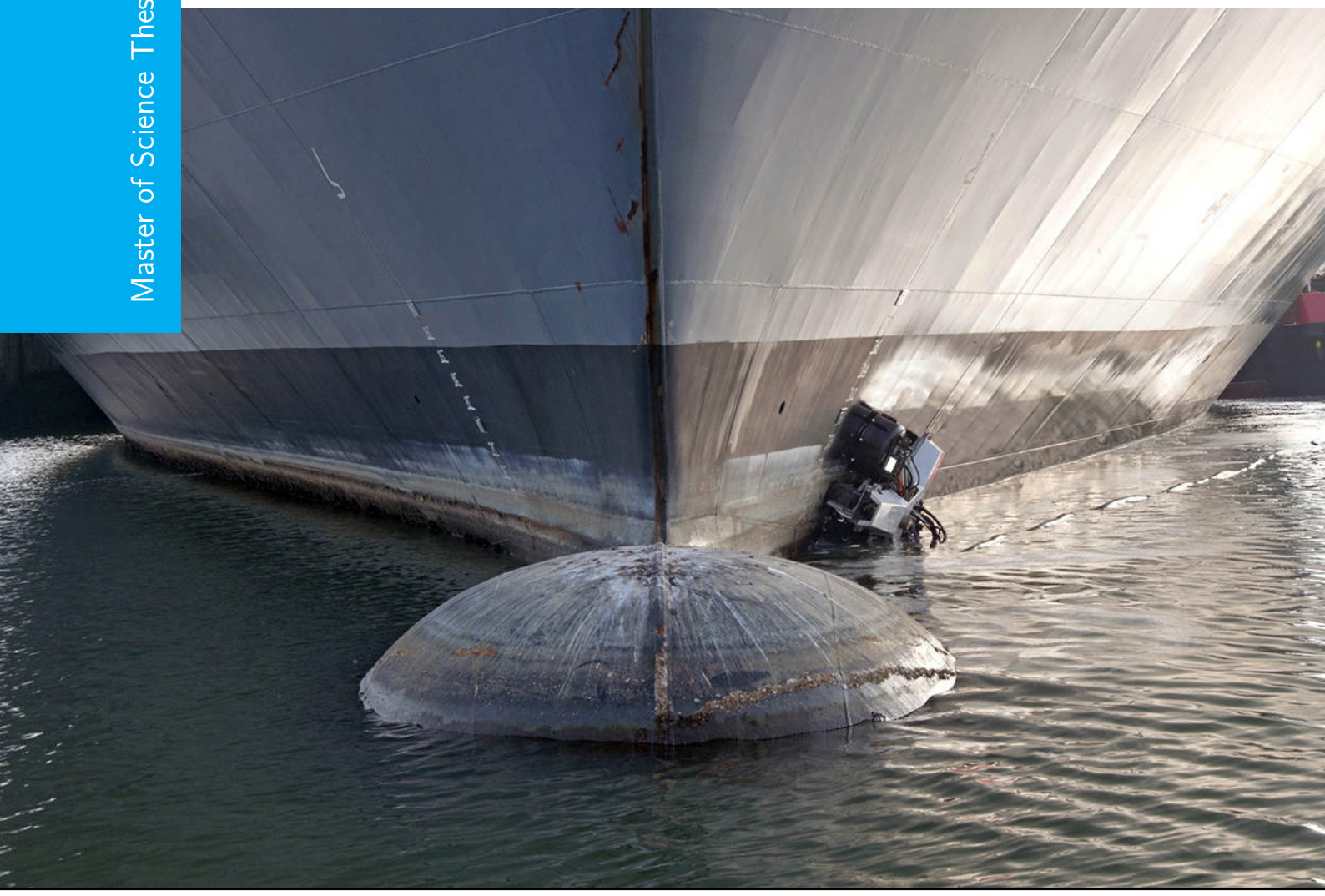
Wheel slip and orientation drift correction for the relative localization system of a ship hull cleaning robot.

K. Cassee

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Wheel slip and orientation drift correction for the relative localization system of a ship hull cleaning robot.

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

K. Cassee

April 13, 2018

CONFIDENTIAL

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Fleet Cleaner

The work in this thesis was supported by Fleet Cleaner B.V.

TU Delft Delft University of Technology

DCSC

Abstract

Fleet Cleaner B.V. is a company that specializes in the cleaning of ship hulls of oceanic trade vessels using a mobile robot. The mobile robot mainly operates out of sight of the operator, making accurate localization of the robot crucial to its operation. However, the absence of absolute localization and sources for error build-up like wheel slip and sensor noise, increase the error between the estimated location and true location over time. Therefore, the goal of this thesis is to *minimize the amount of error build-up between the estimated position and the true position of the robot*, to allow more efficient operation of the robot and to ultimately allow the robot to operate autonomously.

The main sources for error build-up are determined by simulating the position estimator with modeled sensor- and perturbation models and evaluating their individual effect on the position estimator. Algorithms that combat the error build-up are established by synthesizing working principles from literature and extensively testing these principles using simulated data and finally verifying them using real but limited data.

The analysis of the sensor noise and perturbations revealed that the EKF is best suited to estimate the position of the robot and that the main sources for error build-up are wheel slip and IMU heading drift. The addition of a slip detection and velocity correction algorithm reduced the average error build-up per meter traveled by a factor of four. The addition of the heading correction algorithm reduced the error build-up by a factor of three and mitigates the need for intermittent resetting of the robot heading by the operator.

The velocity and heading correction algorithms improve the position estimator allowing the operator to more efficiently control the robot. In order to further reduce the error build-up, it is recommended to add two more wheel encoders to the robot. However, the error will still be unbounded, making the position estimator unsuitable for autonomous operation.

Table of Contents

List of Figures

List of Tables

Preface

This thesis is a continuation on the literature survey, finalized in March 2017, 'Sensor fusion and localization for the positioning system of a ship hull cleaning robot' – that was conducted to provide the necessary information to improve the localization system of the Fleet Cleaner robot.

Acknowledgements

I would like to thank my supervisors dr. S. Wahls from the TU Delft, together with ir. D. Borota, ir. drs. C. de Vet and ir. drs. A. Noordstrand from Fleet Cleaner, for their support and guidance during the writing of this thesis. I would also like to thank my colleagues B. Steensma and D. Verstrate for their commentary during the writing of this thesis.

Delft, University of Technology K. Cassee
April 13, 2018

Chapter 1

Introduction

This chapter serves to motivate and propose a goal for the writing of this thesis. Firstly, some background information about Fleet Cleaner and a motivation for designing a localization system is provided. After the practical introduction, a problem statement is formed in Section 1-3 by examining the current state of the robot's localization system and the desired state. The design of such a localization system is constrained by practical limitations, described in Section 1-4, narrowing the scope of the research topic. In Section 1-5 a set of localization system requirements is proposed to alleviate the robot from the problems described in Section 1-3 and such that the constraints stated in Section 1-4 are satisfied. Finally, the thesis approach to achieve the stated goals is described in Section 1-6.

1-1 Fleet Cleaner ship hull cleaning robot

Fleet Cleaner is a start up company that utilizes a mobile robot, as illustrated in Figure 1-1, to rid ship hulls of fouling such as barnacles and algae, that grow on the ship hull over a period of time. Depending on the amount of fouling, the fuel costs of ships with a fouled hull can increase by 20.4% compared to a hydraulically smooth[1] hull [2]. Fleet Cleaner reports fuel costs savings of up to 5% on ships that they have cleaned in the past. Such fuel cost savings make the Fleet Cleaner robot an attractive cleaning solution for shipping companies and reduces environmental cost of the shipping industry.

The robot propels itself with three hydraulically actuated wheels across the surface of the ship hull, while cleaning the surface with high pressured water jets. The high pressured jets are attached to three rotating cylinders pointed towards the ship hull surface, allowing a total of twelve water jets to clean the ship hull along the entire width of 1.65 $[m]$ of the robot. The robot stays attached to the surface by using three permanent magnets that attract the robot to the metal ship hull.

[1] Hydraulically smooth means that the roughness on the ship hull surface is less the thickness of the laminar sublayer of the turbulent flow [1].

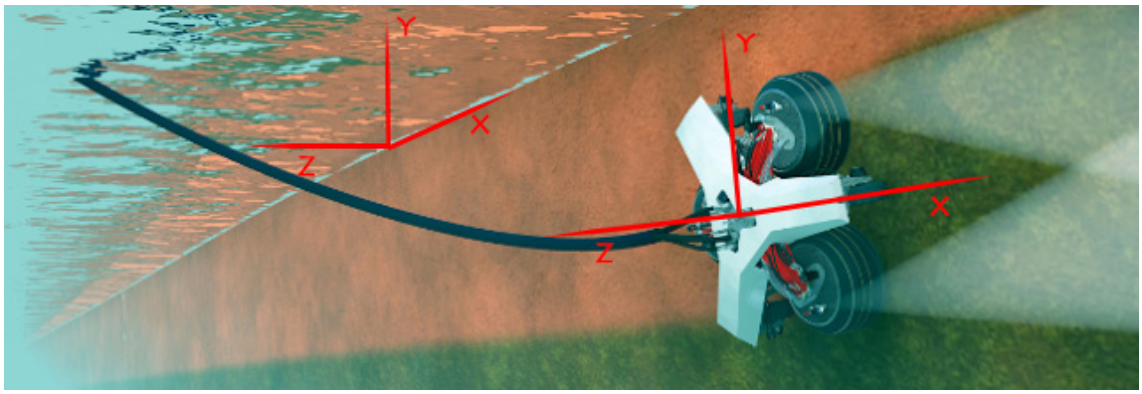


Figure 1-1: Render of Fleet Cleaner robot cleaning the hull of a ship. The figure also displays the convention of the coordinate frames used throughout the thesis [3].

In order to track the robot while it is under water, the location and orientation of the robot is probed by a multitude of sensors, summarized in Table 1-1.

Sensor	Variable	Unit	Description
IMU orientation	$\boldsymbol{q}$, R, or (ϕ, θ, ψ)	$[rad]$	Orientation of robot.
IMU gyroscope	$\boldsymbol{\omega}_{imu}$	$[rad/s]$	Rotational velocity of robot.
IMU accelerometer	$\boldsymbol{a}_{imu}$	$[m/s^2]$	Acceleration of robot.
Depth meter	y_{depth}	$[m]$	Depth of robot.
Wheel encoder	v_{odo}	$[m/s]$	Velocity of front wheel
Steering angle encoder	α	$[rad]$	Steering angle of front wheel

Table 1-1: Table summarizing the available sensor inputs. The orientation can be provided in quaternions, rotation matrices or Euler angles.

Motivation for localization of the robot The localization of the robot on a map of the ship is essential to the operator controlling the robot for a couple of reasons. Firstly, for the vast majority of the time during a cleaning operation, the robot is out of sight of the operator because it is under water. This makes operating the robot without any form of localization difficult since the operator will have to rely on the Forward Looking Sonar (FLS), which has a range of 10 $[m]$, for navigation. Furthermore, the localization of the robot is important to clean the ship hull in a time efficient manner. Fleet Cleaner cleans ships of up to 450 $[m]$ in length so it is essential to minimize the overlap between cleaned sections of the ship hull [4] to maximize time efficiency.

The localization of the robot is not only of importance to the operator. After a cleaning is completed, the customer of Fleet Cleaner is provided with a map of where the robot has cleaned the ship hull. In order to ensure the customer of a thorough cleaning, a realistic trajectory must be presented on a 2D side view of the ship hull. Finally, in order to cut back on operational cost, Fleet Cleaner intends to make the robot autonomous in the future. This again requires a reliable localization of the robot, as it will prevent the robot from running into obstacles on the ship hull.

1-2 Basic concepts

Before a problem statement can be made, it is necessary to explain some basic principles that are used throughout the thesis.

Estimator An estimator is a function $\hat{\theta}$ that estimates an unknown parameter θ on which a *data set* $x[n]$ depends. The estimation of the parameter $\hat{\theta} = g(x[0], x[1], ..., x[n])$ is the problem of parameter estimation [5].

Error build-up In this thesis, *error build-up* is a term to describe the amount of error between the true position and estimated position that accumulates per true meter traveled. The error build-up is defined as

$$EBU = 100\% \cdot \frac{||\boldsymbol{p}_{est}(T) - \boldsymbol{p}_{true}(T)||}{\sum_{i=T-N}^{T} ||\boldsymbol{p}_{true}(t_i) - \boldsymbol{p}_{true}(t_{i-1})||}, \tag{1-1}$$

where $\boldsymbol{p}_{true}(T)$ is the true position at time instance T and $\boldsymbol{p}_{est}(T)$ is the estimated position at time instance T and N is the size of the data sequence over which the error build-up is calculated.

Absolute and relative localization In this thesis a distinction is made between *absolute* and *relative* localization. Absolute localization can only be achieved if the position or orientation with respect to a ship hull fixed frame can be gauged directly by some sensor. Because absolute sensors can gauge the position and orientation directly, integration of position and orientation derivatives is not needed and measurement errors do not accumulate. Relative localization can be used to gauge a position or orientation by integrating a measurement that is a derivative of the position or orientation [6]. Integrating the wheel encoder velocity output in order to obtain a position estimate is an example of relative localization [7]. Because of the accumulation of velocity measurement errors the estimated position drifts away from the true position over time [8]. This amount of drift is quantified by the error build-up score, depicted in Equation (1-1).

1-3 Problem statement

The need for a localization system was motivated in Section 1-1. In this section, it is discussed how error build-up deteriorates the performance of the localization system and why it induces extra workload on the operator. Furthermore it is discussed what the culprits of the error build-up in the current position estimator are.

Operator workload The most time efficient way of cleaning the ship is by maneuvering the robot along the length, or $x-$direction, of the ship between a start marking, like a weld line, and a final marking, like depth markers. This movement is illustrated in Figure 1-2. Ideally, the operator tracks the position of the robot on a 2D side-view map [2] and cross-references this

[2] This 2D side-view map will henceforth be referred to as 'the map'.

position to the sonar imagery once the map shows the robot is near the final marking to ensure it has covered the length of the ship. However, due to the error build-up between the true and estimated position, as shown in Figure 1-2, the map will often indicate that the robot is near the final marking but the true position of the robot will be further away. This forces the operator to constantly monitor the sonar imagery, in order to avoid missing the final marking. Additionally, the error build-up requires the operator to reset the estimated position of the robot on the map intermittently so that when it is provided to the client, it shows a realistic cleaning trajectory. The resetting of the position estimate is done by cross-referencing land marks, detected by the FLS, to the map of the ship. This cross-referencing of the FLS imagery with the map to avoid missing the final mark and to provide a comprehensive map to the client, puts additional work load on the operator, which distracts from other occupations. The main causes for error build-up in the position estimate are discussed below.

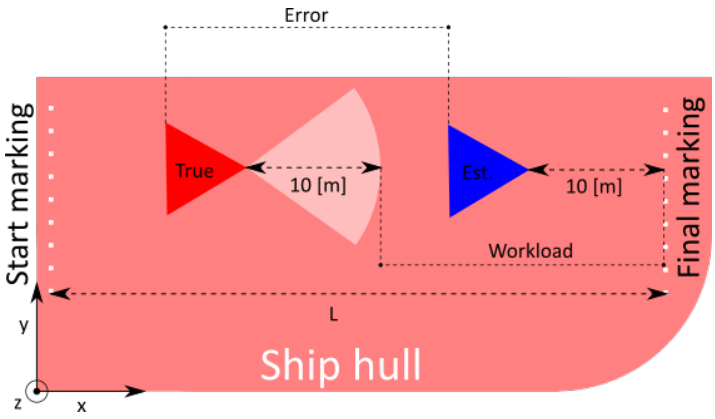


Figure 1-2: Illustration of the error between the true and estimated position of the robot while it is moving between a start and final marking. The red and blue triangles represent the true and estimated robot locations, respectively. The sonar beam of the FLS is represented by the white triangle. The length between the start and final marking is denoted 'L'.

2D position estimator The current position estimation is achieved with the kinematics of a mobile robot on a flat 2D surface with front steering, assuming no-slip conditions. The kinematics of the front wheel location can be described by [9]

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\left(\psi(t) + \alpha(t)\right) \\ \sin\left(\psi(t) + \alpha(t)\right) \\ \frac{\sin\left(\alpha(t)\right)}{l} \end{bmatrix} v_{odo}(t), \tag{1-2}$$

where $\psi(t)$ is the orientation of the robot with respect to a fixed frame, $v_{odo}(t)$ is the wheel speed of the front wheel, $\alpha(t)$ is the steering angle and 'l' is the wheel base, displayed in Figure 1-3.

The position estimator described in (1-2) assumes the robot moves on a plane, which induces errors in the position estimate since the robot actually moves on a 3D curved surface. In order to reduce the error build-up caused by the 2D kinematics, the position estimator must be extended to a 3D position estimator.

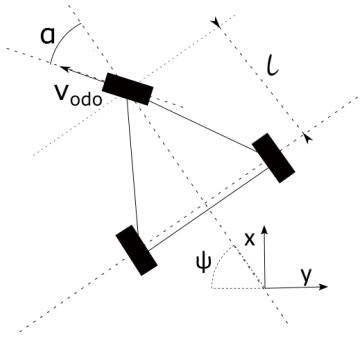


Figure 1-3: Illustration displaying the parameters that are used to describe the 2D robot kinematics. The front wheel steering angle is denoted α, the front wheel speed is denoted v_{odo} and the distance between the front wheel and the rear wheel axis, also called wheel base, is denoted L. The orientation of the robot is denoted ψ and the position is expressed in x and y.

Unreferenced heading Extending the position estimator to three dimensions increases the degrees of freedom of the robot to six, namely three positions (x, y, z) plus three rotations, parameterized here as extrinsic Euler angles, (ϕ, θ, ψ), depicted in Figure 1-4. Due to the presence of three large magnets, the IMU can not employ a magnet, causing the orientation ψ to be unreferenced which induces heading drift over time [10]. Since the coordinate convention is selected as displayed in Figure 1-4, the rotation around the ship hull fixed $y-$axis, θ becomes unreferenced instead.

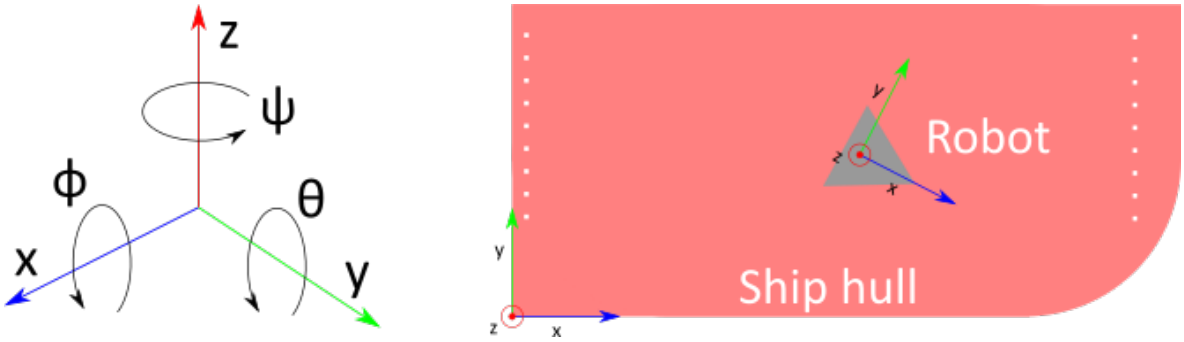


Figure 1-4: Image displaying the coordinate system axes and their associated rotations using the right hand rule. This convention is used throughout the thesis.

Relative localization The position estimator currently only has an absolute measurement input for the $y-$coordinate position estimate, using the depth meter. For the estimation of the $x-$ and $z-$coordinate position, the position estimator solely relies on relative localization. As was previously discussed, such relative localization methods inherently suffer from the problem of error build-up [8]. In most cases, the main source for error build-up of mobile robots that use relative localization to estimate the position on slippery or uneven surfaces is wheel slip [11]. The sensor noise also adds to the error build-up since small errors in the wheel encoder velocity measurements, steering angle measurement, and orientation measurement accumulate over time due to error integration.

To summarize, the main problem is that *the error build-up in the position estimate puts additional workload on the operator*. The main causes for error build-up are listed below.

1. The robot is assumed to be on a flat plane, while in reality it is on a curved 3D surface.

2. The heading orientation of the IMU is unreferenced, causing it to drift over time.
3. The $x-$ and $z-$ coordinate position estimates are estimated using relative localization, which inherently suffers from error build-up.

The constraints to which a solution to these problems is subjected is discussed next.

1-4 Localization system constraints

Together with the system constraints, this section also defines a scope to which the operations of the localization system designed in this thesis limits itself. At the end of this section the system constraints are combined with the problem statement, discussed in the previous section, to formulate a *thesis goal*.

Practical constraints The previously described problem of localization and error build-up is a common topic in the design of localization systems for Unmanned Underwater Vehicle (UUV)s [12, 13, 14] and land based mobile robots [15, 16, 17]. Generally, these localization systems rely on the combination of a model predicted position estimate and absolute localization systems such as acoustic beacons, in the case of UUVs, or Global Positioning System (GPS) in the case of land vehicles, to counter error build-up. In fact, absolute localization systems like an acoustic beacon system [4] have been proposed for the Fleet Cleaner robot in the past. However, the implementation of an external absolute localization systems was never pursued by Fleet Cleaner due to time and financial constraints and will not be available for the localization of the robot during this thesis. The localization system will thus be limited to the sensors listed in Table 1-1 and the computational hardware currently available.

Localization system scope The localization of the robot while it is above water and on the bottom of the ship brings a whole set of different challenges. Firstly, when the robot is above water, the localization system loses its absolute reference in y direction since the depth gauge measurement becomes unavailable. Secondly, when the robot is on the bottom of the ship, it mainly moves in x and z direction, again making the depth gauge measurement futile for tracking the robot position – and the robot mainly rotates around the unreferenced y-axis. It is expected that the set of sensors listed in 1-1 do not suffice to reliably estimate the position of the robot when it is above water or on the bottom of the ship and therefore these modes of operation are disregarded in this thesis. Furthermore, the robot operates on the side of the ship for 75% of the time, so the localization algorithm that functions on the side of the ship is more urgent.

Thesis goal Combining the problem statement of Section 1-3 with the above discussed system constraints, the main goal of this thesis is thus to *reduce the error build-up in the unreferenced (x, z)-position estimates, while the robot is underwater and on the side of the ship, without the use of additional hardware.*

In the next section it is discussed what requirements the localization system must fulfill to achieve this goal to a high enough degree.

1-5 Localization system requirements

The underlying reason of reducing the error build-up in the position estimate is to reduce the workload on the operator, as was discussed in Section 1-3. Since the $x-$ and $z-$coordinate positions are unreferenced, occasional position estimate adjustment of the operator will be required. In order to minimize the workload, it is proposed that such an adjustment should only be required *once* per *cleaning lane*. A cleaning lane is defined as the trajectory between a known start marking at $t = T_1$ and a known final marking at $t = T_2$, depicted in Figure 1-5. The robot performs consecutive cleaning lanes, each at a greater depth, to clean the ship hull.

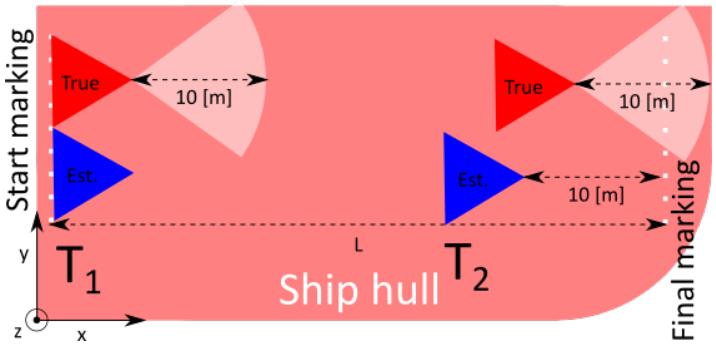


Figure 1-5: Illustration of a cleaning lane executed by the robot on a ship hull. The red and blue triangles represent the true and estimated robot locations, respectively. The sonar beam of the FLS is represented by the white triangle. The length between the start and final marking is denoted 'L'.

In order to satisfy the requirement that the operator should only have to cross-reference the estimated position with the sonar imagery once per cleaning lane, the error between the true position and the estimated position should stay within the range -5 and 5 $[m]$, where a negative error indicates that the true robot position is greater than the estimated position, as depicted in Figure 1-5 at $t = T_2$. If the error stays within this range, the final marking and all other objects specified on the map that lie in between the start and final marking, should be immediately visible in the sonar imagery once the estimated position indicates that the robot is removed 5 $[m]$ from that object or final marking. Once the final marking is detected the operator can reset the position estimate on the map. Any error higher than 5 and lower than -5 $[m]$ counts as workload, since the operator will have to cross-reference the estimated position with the sonar imagery intermittently to detect the final marking.

The position error acquired at $t = T_2$ depends upon the error build-up of the position estimator and the length of the ship. The fore and aft of each ship hull side is usually cleaned separately, so that the FLS always faces the for or aft of the ship. Currently, the largest ship in the world is the the Prelude FLNG, at 488 $[m]$ long. If the side of the ship is cleaned in two parts, the error per cleaning lane needs to stay between -5 and 5 $[m]$ over a cleaning lane of 244 $[m]$. Using Equation (1-1), the error build-up thus needs to stay below 2.05%.

The quantitative and qualitative requirements of the position estimator are summarized below.

- Quantitative

  - The localization system must have an error build-up below 2.05%.
  - The localization system must be able to output a position estimate at a frequency of at least 3 $[Hz]$[3].

- Qualitative
  - The position estimator must require a minimal amount of operator resets.
  - The localization system must output an informative map that displays where the robot has cleaned the ship.

The thesis structure that is used to achieve the goal stated in Section 1-4 is discussed below.

1-6 Thesis approach

In order to achieve the thesis goal, the thesis is divided into two parts: *Analysis* and *System design*. In short, the objective of the first part is to identify and study the sources for error build-up and the objective of the second part is to alleviate the position estimator from these sources.

Part 1: Analysis

The objective of the first part of the thesis is to *gain understanding into the effect that perturbations and sensor noise have on the error build-up in the position estimate*, such that this knowledge can be used to efficiently reduce the error build-up. This objective is achieved by studying sensor data obtained in real life tests to model realistic sensor noise and perturbation models and testing these models on a well tuned position estimator to evaluate their effect on the position estimate. The simulated sensor data is also used to evaluate algorithms that combat the sources for error build-up where real data is unavailable or very limited.

Part 2: System design

The objective of the second part of the thesis is *to reduce the error build-up in the position estimate by countering the identified sources for error build-up*. In Chapter 4, 5 and 6 this is achieved using the following design structure.

Conceptual design Conceptual design specifies the principal solution to the sub-problems. For each respective chapter this means that the objective is *to find and establish a working principle to reduce the error build-up*. This objective is mainly achieved through the review of literature relevant to the identified error build-up sources.

[3]Fleet Cleaner states that in order to ultimately make the robot autonomous, an accuracy of within 10 $[cm]$ is required. The robot travels at 0.3 $[m/s]$, and so the update rate needs to be at least 3 $[Hz]$, otherwise the true position of the robot exceeds the 10 $[cm]$ accuracy limit.

Simulation The objective of the simulation part of each chapter is to *evaluate the performance of the working principles with respect to a simulated ground truth and to identify the limits of the working principles.* It was noted in [18] that it is unlikely that a permanent ground truth system will be available and so in order to make statements about the performance of algorithms, simulated sensor inputs are necessary. The generation of simulated data, which is a demanding task, is justified by the following reasons:

1. The performance of working principles can be evaluated for multiple noise realizations. Under these conditions stronger statements can be made about the robustness to noise of the working principles.
2. Using the simulated ground truth model, the correct implementation of the position estimators can be verified in a noise-free test.
3. The current method of using real data to evaluate algorithms does not allow for continuous tracking but only allows for the algorithm performance to be evaluated at certain points [18].
4. Sources for error build-up identified in Part 1, can be isolated, so that their effect on the position estimate can be studied individually.
5. External perturbations can be manually modified so that the limits of the working principles can be explored and quantitative statements can be made about their robustness.

Validation The validation part of the implementation chapters have as objective to *validate the simulated performance of the working principles using real sensor data.* The validation of the working principles using real data is problematic due to the unavailability of a ground truth measurement and the limited amount of validation data. Validation of the working principles is achieved using ad hoc measures to establish a ground truth measurement using real data that can be used to evaluate the performance of the working principles and assessing if it is consistent with the simulated performance. The tacit assumption is that if it is, stronger statements can be made with respect to the performance of the working principle.

Chapter 2

Sensor data modeling

Recall the objective of Part 1 of the thesis: to *gain understanding into the effect that perturbations and sensor noise have on the error build-up in the position estimate.* The achievement of this objective is impeded by two factors:

1. The lack of knowledge about sources for error build-up such as sensor noise and external perturbations.
2. The lack of a ground truth to evaluate the effect that the sources for error build-up have on the position estimate.

The acquired knowledge about the sensor noise and external perturbations is pivotal, since it is used in Chapter 3 to establish a working principle for the 3D position estimator and used in this chapter to produce sensor noise and perturbation models. Furthermore, the sensor noise and perturbation models are used in Chapter 3 to evaluate the effect that they have on the position estimate. The sensor data models and ground truth model are also used to evaluate the working principles proposed in Chapter 4, 5 and 6 for the reasons listed in Section 1-6 under paragraph 'Simulation'.

The objective of this chapter is thus to *acquire knowledge about the sensor noise and external perturbation characteristics* and to *generate sensor data models and a ground truth model.*

This is achieved through the analysis of the sensor noise characteristics and external perturbations in Sections 2-1 and 2-2. After which noise, perturbation and ground truth models are established in Sections 2-3 and 2-4, respectively. In Section 2-5 the objective stated at the beginning of this chapter is reflected upon.

2-1 Sensor noise characteristics analysis

Currently, Fleet Cleaner only posses sensor noise characteristics specified in the technical documents of the sensors. These characteristics are likely underestimated since they are

recorded in a laboratory and do not capture the noise that is induced by the magnetic field of the ship, the magnets, robot motion and the noise induced by the 150 $[m]$ long umbilical cord that the robot is attached to.

The appropriate selection of a position estimator for the Fleet Cleaner robot partly depends upon how the sensor data is distributed statistically as is discussed by S. Thrun et al., in [17, p. 39] and [17, p. 96]. Likewise, the selection of appropriate modelling techniques for simulated sensor data also depend upon the statistical distribution of that data [19]. Common position estimators like the Extended Kalman Filter (EKF) assume that the sensor inputs are corrupted by Additive White Gaussian Noise (AWGN), meaning completely random noise [17, p. 39]. However, many sensor noise Power Spectral Density (PSD)s are non-white, and may not be normally[1] distributed, which may upset the position estimator. Thus, the statistical distributions and colour of the sensor data sequences must be analyzed to select an appropriate position estimator and to produce noise models to test the position estimator.

The objective of this section is to *determine the statistical distributions of the sensors summarized in Section 1-1* and to determine *if the PSDs of the sensor noise can be accurately approximated by a white noise sequence.*

2-1-1 Sensor noise distribution analysis

The classification of a data sequence distribution is often done by using a Probability Density Function (PDF) as a PDF is a function that describes the distribution of random data and variables [5]. There exists a long list of PDFs that are used to describe data of which the *normal distribution*, also known as the *Gaussian distribution* is the most ubiquitous. If the noise distributions can approximately be represented by a Gaussian distribution this is beneficial since it would make the modeling of sensor noise less time consuming [19] and allow for the implementation of more common position estimators like the EKF [17, p. 39]. So it is checked if the sensor noise distributions can be approximated with a Gaussian distribution.

Sensor noise data acquisition In order to evaluate the sensor noise distributions, data sequences of the sensor outputs are obtained of when the robot is in a stationary, underwater position. To include the effect of wave perturbations on the distribution of the depth sensor, the data sequence used for the depth gauge distribution was taken at a depth of 60 $[cm]$ in rough water conditions.

The histograms of the sensor data sequences displayed in Figure A-1.1 show that only the gyroscope outputs, and depth meter follow an approximate Gaussian distribution. The roll, pitch, yaw and acceleration data in $x-$ and $y-$direction have multi-modal[2] distributions and do not appear to be symmetric and so are not approximately Gaussian.

[1] A normal distribution and a Gaussian distribution are identical and the terms normal and Gaussian are used interchangeably.

[2] a multi-modal distribution is a distribution with two or more maxima.

Sensor	Approximately Gaussian	Std. deviation σ
Depth sensor	Yes	0.0068
Gyroscope x	Yes	0.0031
Gyroscope y	Yes	0.0030
Gyroscope z	Yes	0.0031
Roll	No	0.0039
Pitch	No	0.0038
Yaw	No	0.0031
Steering angle	No	0.0004
Wheel encoder	No	0.0057
Accelerometer x	No	0.0056
Accelerometer y	No	0.0052
Accelerometer z	No	0.0060

Table 2-1: Table summarizing if the histogram of the stationary sensor data sequence is approximately Gaussian distributed – and the standard deviations of the stationary data sequences.

2-1-2 Sensor noise colour analysis

In the frequency domain, the PSD of the sensor noise is often characterized by its 'colour'. If the PSD exhibits so called 'white' behaviour, this means that the noise is completely random, making it less time consuming to model since it can be modeled using a random sequence in Matlab.

To determine the if AWGN is an accurate model, the data sequences are evaluated using the Auto Correlation Function (ACF) [20]. The auto-correlation function is defined as [21, p. 101]

$$R_x(k,l) = E\left[\boldsymbol{x}(k)\boldsymbol{x}(l)^T\right]. \tag{2-1}$$

If the ACF of the sensor data sequence lies within the $R_x(k,l) = \pm\frac{2}{\sqrt{N}}$ confidence bounds where N is the sample size, there is a 95% confidence that the sequence can be assumed to be random and thus white. If the ACF of the sensor data sequence significantly falls outside these bounds, indicating a high correlation between samples, the sensor noise can be assumed to be non-white [22, p. 28].

In order to evaluate the PSDs of the sensor noise, the data sequences used to evaluate the sensor noise distributions are used. Using the ACF of the sensor data sequences, displayed in Figure A-2.1 it is determined that from the set of sensors listed in Section 1-1 only sensor noise PSDs of the gyroscope output can be represented using white noise data sequences. The rest of the sensor noise PSDs are considered to be non-white.

2-1-3 Overview

The objective of this section was to *acquire knowledge about the sensor noise characteristics of the sensors listed in Section 1-1, so that an appropriate position estimator can be selected and sensor noise can be accurately modeled.* In Table 2-2 the sensor noise characteristics are summarized. The table shows that most sensors are not normally distributed and do not have white PSDs.

Sensor	Approximately Gaussian	White noise
Depth sensor	Yes	No
Gyroscope x	Yes	Yes
Gyroscope y	Yes	Yes
Gyroscope z	Yes	Yes
Roll	No	No
Pitch	No	No
Yaw	No	No
Steering angle	No	No
Wheel encoder	No	No
Accelerometer x	No	No
Accelerometer y	No	No
Accelerometer z	No	No

Table 2-2: Table summarizing per sensor if the distributions follow an approximate Gaussian distribution and if the PSD of the noise is white.

Limitations The sensor noise distribution and PSD results have a limited reliability because they do not include the effects of vibrations or motion of the robot. Furthermore, it is assumed that the sensor data distribution are constant over the entire frequency range, which they are not in reality [23].

2-2 External perturbation analysis

Wheel slippage, and the deterioration of the position estimate, is a problem frequently encountered by mobile robots that operate on loose or slippery underground [24, 11]. During cleaning sessions it is frequently observed that the Fleet Cleaner robot also suffers from wheel slip, especially when the ship hull surface is extremely slippery due to heavy fouling. Another major source for error build-up is the IMU heading drift. Usually IMUs use a compass and the earth's magnetic field to estimate the heading. However, the IMU employed by the Fleet Cleaner robot does not posses such a compass, because of the large permanent magnets attached to the robot, and thus the heading estimate drifts away from the true heading over time [10]. Although both wheel slip and heading drift are expected to be major sources for error build-up in the position estimate of the Fleet Cleaner robot, no knowledge exists on precisely how much.

The acquisition of knowledge about the characteristics of the previously described external perturbations is two-fold. Firstly, the knowledge is needed to design perturbation models that can be used in simulations for reasons described in Section 1-6 under 'Simulation'. Secondly, knowledge about the perturbations can be used at a later stage to establish working principles that mitigate error build-up induced by the perturbations.

The objective of this section is thus to *obtain knowledge about what characteristics can be used to describe the external perturbations* and to *quantifying them* such that they can be used later on to generate perturbation models and to combat the perturbations efficiently.

2-2-1 Wheel slip

Before the characteristics of wheel slip can quantified two types of wheel slip are distinguished based upon observations namely *longitudinal slip* and *orthogonal slip*.

Longitudinal slip causes the associated wheel encoder to register revolutions, even though these revolutions do not correspond to a linear displacement of the wheel, inducing an error between the measured velocity and the true velocity. If the wheel registers less revolutions of the wheel compared to the linear displacement, this is called *skidding* [25].

Orthogonal slip occurs when the front wheel slips orthogonal to its rolling direction. If orthogonal wheel slip occurs while the robot is moving in the $x-$direction on the side of the ship, as depicted in Figure 1-5, it will induce and error in the $y-$coordinate position estimate and in the robot fixed ψ orientation. Both these dimensions have an absolute measurement available to them and so it is expected that the error build-up induced by orthogonal wheel slip is negligible. The modelling and evaluation of orthogonal wheel slip is therefore omitted.

Longitudinal wheel slip

In Figure 2-1 longitudinal wheel slip is captured in a sequence of three images. It can be seen that the pattern on the ship hull surface remains the same while the wheel is rotating, thus indicating longitudinal wheel slip.

Figure 2-1: Images captured of wheel slipping in longitudinal direction. The red dots mark the same spoke in a time span of 1 second. The images show that although the wheel is rotating, the robot remains stationary.

Two types of slip that are observed during operations, so called *stationary slip* and *non-stationary slip* or *dynamic slip*. The former is a mode of slippage where the front wheel, or one of the rear wheels slips while the true velocity of the robot is zero. The latter is a mode of slippage where one of the wheels start slipping while the true velocity of the robot is not zero. From Figure 2-2 it can be observed that the magnitude of the wheel encoder output is greater when stationary slip is detected compared to dynamic slip.

Longitudinal wheel slip characteristics From the ranges that are marked as slip in Figure 2-2 it is notable that the output displays oscillatory behaviour. This oscillatory behaviour is especially visible in the range of $[85 - 108]$ seconds. It can be determined from the data associated to Figure 2-2 that the frequency at which slip occurs is about once every 23 seconds[3], with an average slip length of 7.2 seconds and an magnitude of between 0 and 0.15

[3]The frequency of occurrence of wheel slip will be adjusted in Chapter 3 based on multiple data sets. This first requires the development of a position estimator.

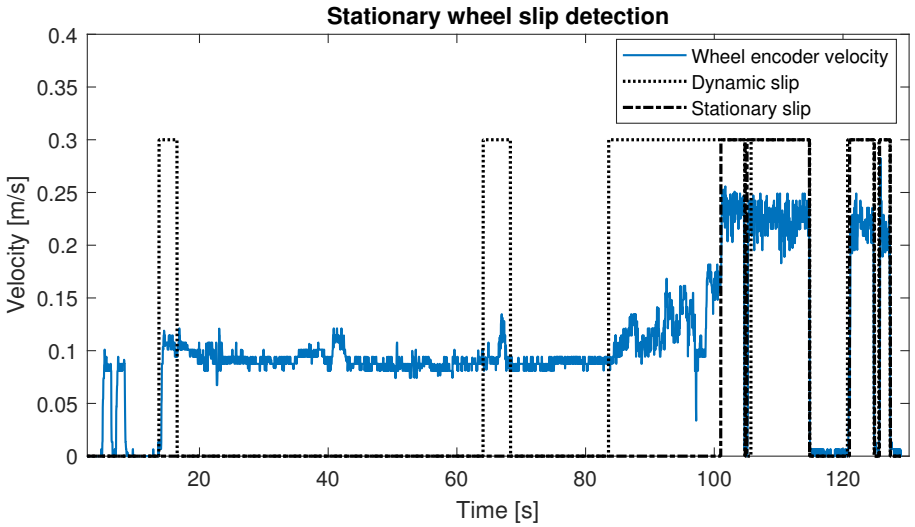


Figure 2-2: Graph displaying the wheel encoder velocity output, stationary and dynamic slip labels. It can be observed that when stationary slip occurs, the wheel encoder output is higher in comparison to when dynamic slip occurs.

$[m/s]$. The transition between dynamic and stationary occurs at approximately 50% of the maximum longitudinal wheel slip magnitude. Using the camera images, the dynamic slip is estimated to be approximately 30% of the velocity when the wheel has traction, denoted the *nominal velocity*. The oscillatory behaviour is quantified using the standard deviation of the signal. When no slip is detected in the data sequence displayed in Figure 2-2 the standard deviation is 0.0066, as opposed to 0.0370 when it is slipping. These values were determined from the data associated to the ranges $[25-34]$ seconds and $[106-115]$ seconds in Figure 2-2, respectively.

2-2-2 Heading drift

The occurence of heading drift is reflected in the roll, pitch, yaw data of the IMU in a stationary situation, displayed in Figure 2-3.

Heading drift characteristics From the data associated to Figure 2-3 it is determined that the heading direction drifts at a rate between 0 and 0.0037 $[rad/s]$, observed at $T = 370$ and $T = 580$, respectively.

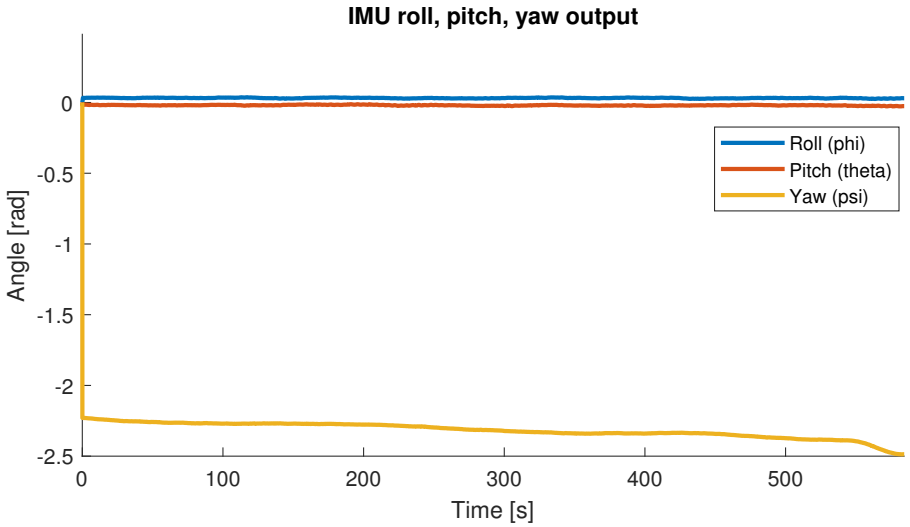


Figure 2-3: Graph of roll, pitch, yaw data of the IMU in a stationary situation on the deck of the ship. Note that the yaw (yellow line) of the robot drifts even though the robot is stationary.

2-2-3 Overview

The sub-objective of this section was to *obtain knowledge about what characteristics can be used to describe the external perturbations* and to *quantifying them* such that they can be used later on to generate perturbation models and to combat the perturbations efficiently. The external perturbation characteristics are summarized in Table 2-3.

Perturbation	Characteristics
Wheel slip	Increase in wheel encoder output of up to 0.15 $[m/s]$.
	Increase in standard deviation of wheel encoder output of about five times the normal value.
	The true velocity is approximately 30% of the nominal velocity during dynamic slip and zero during static slip.
	The slip mode transitions between stationary and dynamic at about 50% of the maximum slip magnitude.
Heading drift	Drift in the yaw output of between 0 and 0.0037 $[rad/s]$.

Table 2-3: Table summarizing the characteristics of the wheel slip and heading drift perturbations

Limitations The sensor data under consideration, displayed in Figure 2-2 originates from operating on the HS Tosca, which was considered a medium fouled ship. The quantification of the perturbation characteristics was based on this data but may vary significantly from ship to ship because of different degrees of fouling. Furthermore, the drift in the IMU yaw orientation was estimated while the robot was stationary on the deck of the ship and so the IMU was unaffected by vibration and electronic interference that may be prevalent during normal operation.

2-3 Sensor noise and perturbation modeling

The testing of algorithms, such as the auto-pilot[4] algorithm and 2D localization algorithm, is currently done using real sensor data obtained from cleaning sessions. Repeating the reasons stated in Section 1-6, using real sensor data to test algorithms is somewhat limited because:

1. The obtained test results cannot be compared against a ground truth.
2. The effects of disturbances like sensor noise, wheel slip and heading drift cannot be tested individually. It is thus unclear if these disturbances are present in the data and how they would affect the algorithm.
3. It is difficult to specify the performance limits of the algorithm with respect to said disturbances since the disturbances are unknown.
4. The algorithms are tested using a small amount of different data sets. This reduces the reliability of the performance results because they are only evaluated for a small amount of noise realizations.

In Chapter 3 the effects of the disturbances on the position estimator are studied. In order to do this and to be able to compare them against a ground truth the disturbances thus need to be applied to the simulated data sequence individually. In Chapters 4, 5 and 6 algorithms are established to combat the disturbances, it is therefore again required that the disturbances are isolated. Furthermore, direct control of the disturbance characteristics allows for the evaluation of the limits of the algorithm's performance. The evaluation of algorithms using multiple noise realizations ensures that the obtained performance results are robust.

The objective of this section is thus to *design sensor noise and perturbation models* so that the disturbances can be isolated in the simulated data sequences, the characteristics of the disturbances can be controlled and multiple sensor noise realizations can be established.

The modeling of real phenomenon like sensor noise and external perturbations can be made increasingly complex as the models approximate the phenomenon closer and closer. To limit the amount of time that is spent on modeling the disturbances a set of requirements is provided that needs to be fulfilled by the perturbation models.

The requirements for the sensor noise models are that they:

- Capture the uncertainty present in the sensor signals. Meaning that the standard deviation of the model should at least be as high as the standard deviation of the sensor signal.
- Approximate the non-white behaviour present in the signals. This requirement ensures that position estimators that assume AWGN are tested on their robustness to non-white noise.

The requirements for the external perturbation models are that they:

[4]This algorithm ensures that the robot stays at the same depth level to reduce the work load on the operator.

- Capture the varying characteristics present in the external perturbations due to variation in external conditions. In Section 2-2 a range was assigned for the magnitude of most perturbations. The models should be able to reflect this range.
- Have the same occurrence frequency and magnitude as the perturbations observed in real data. This ensures that the effect of the perturbations on the position estimator reflects that of the real data.

2-3-1 Sensor noise modeling

Sensor noise modeling approaches can be divided into frequency domain methods [26] and time domain methods [20] of which the frequency domain methods are most often used to model background noise [23]. The frequency domain modeling approach is selected to model the sensor noise since the sensor data was obtained from a stationary situation and can thus be considered to be background noise.

Spectrum fitting method Modeling of background noise using frequency domain methods can be further divided into spectrum fitting and so called statistical analysis methods. Spectrum fitting methods capture the sensor noise by fitting a Transfer Function (TF) to the PSD of the sensor data to capture the average noise spectrum, assuming that the input of the TF is white Gaussian noise [19, 27]. A downside to this method is that it assumes a constant distribution of the data over all frequencies. In order to include the random behaviour of the noise at each individual frequency, statistical analysis methods can be used [23]. These methods estimate the PDF of the data at different frequency ranges and in doing so obtain more accurate noise models. The downside of statistical analysis methods is that appropriate[5] sensor data needs to be abundant in order to estimate the PDFs over a wide spectrum of frequencies. Examples of this are Meng et al. [23] who, in order to obtain PDFs over a frequency range of 0 to 30 $[MHz]$, used a sensor data set that was obtained over 175 hours and Benyoucef et al., [19] who, used a data set measured using 56 different sensors to obtain 2600 samples per day with a sample size of 401 in each case, over multiple days. The size of the data set that was used was thus in the order of 10^6. Such abundance of appropriate sensor data is not available for Fleet Cleaner and so the noise is modeled using a more simple and time efficient spectrum fitting method.

Spectrum fitting design procedure The sensor noise models are designed by first fitting a continuous time TF to the stationary sensor data PSD. The sensor noise PSD is obtained using Welch's method [28], which can be implemented using the $pwelch()$ function in Matlab. After the continuous time TF is obtained, it is discretized using a zero order hold discretization method [29, p. 307] with the $c2d()$ function in Matlab. An example of the spectrum fitting method, employed to model the sensor noise, is provided in Appendix A-3.

Sensor noise model scaling The application of the spectrum fitting method on the non-Gaussian sensor data reveals its limitations. For instance, the time domain evaluation of the

[5] appropriate meaning that the data was obtained in a static situation, in the case of Fleet Cleaner robot.

simulated sensor noise for the IMU orientation has a standard deviation of 0.0570, whereas the real IMU orientation data has a standard deviation of 0.0031. In order to not overestimate the simulated sensor noise uncertainty, the noise is scaled with scaling factor

$$\xi = \frac{std(\boldsymbol{x})}{std(\hat{\boldsymbol{x}})}, \tag{2-2}$$

where $std(\boldsymbol{x})$ is the standard deviation of the real sensor data and $std(\hat{\boldsymbol{x}})$ is the standard deviation of the simulated sensor data. Using the same methodology to determine the noise model for the depth sensor, the scaled sensor noise model transfer functions summarized in Table 2-4 were obtained.

Sensor	Sensor model TF
Depth gauge	$H_{depth}(z) = \frac{0.002247z-0.0009659}{z^2-1.067z+0.08081}$
IMU Euler angles	$H_{(\phi,\theta,\psi)}(z) = \frac{0.0007999}{z-0.9692}$
IMU acceleration data	$H_{\boldsymbol{a}}(z) = \frac{0.0003205z+0.0002603}{z^2-1.535z+0.5351}$
Steering angle encoder	$H_{\alpha}(z) = \frac{0.0001892z+8.784e-05}{z^2-0.6394z+0.1011}$
Wheel encoder	$H_{odo}(z) = \frac{0.004121}{z-0.5879}$

Table 2-4: Table summarizing the scaled sensor noise models. These models are used throughout the thesis to model the sensor noise.

The simulated sensor signals, corrupted by noise are denoted

$$\tilde{\boldsymbol{s}}_i = \boldsymbol{s}_i + H_i \cdot \eta(k), \tag{2-3}$$

where $\boldsymbol{s}_i$ is the nominal signal of sensor i, H_i is the noise TF associated to sensor i and $\eta(k)$ is a normally distributed, unit standard deviation, zero mean data sequence. The gyroscope noise is simulated by substituting the standard deviation of the gyroscope signal for H_i in Equation (2-3), summarized in Table 2-1.

2-3-2 External perturbation modeling

In this section the characteristics quantified in Section 2-2 are used to produce external perturbation models.

Longitudinal wheel slip modeling

The longitudinal wheel slip characteristics specified in Section 2-2 dictate that:

- The longitudinal wheel slip occurs once every 23 seconds,
- The longitudinal wheel slip lasts for 7.2 seconds on average each occurrence.[6]

[6]The intensity of the wheel slip is scaled such that it occurs once every 10 seconds with a duration of 3.13 seconds.

- The wheel encoder output increases between 0 and 0.15 $[m/s]$ during slippage
- The standard deviation of the signal increases by a factor of five.

Using these characteristics, the combined sensor noise and perturbation model depicted below is formed.

$$v_{odo}(k) = \begin{cases} v_{nom}(k) + \cdot H_{odo} \cdot \eta(k) & \text{if } SlipMag = 0 \\ v_{nom}(k) + 5 \cdot H_{odo} \cdot \eta(k) + SlipMag & \text{if } SlipMag > 0 \end{cases}$$

. where $SlipMag$ is the magnitude of the longitudinal slip and v_{nom} the nominal front wheel velocity under no-slip conditions. The parameter $SlipMag$ varies between 0 and $SlipMagMax = 0.15$ $[m/s]$. The simulated wheel encoder output, corrupted by the sensor noise and longitudinal wheel slip is shown in Figure 2-4.

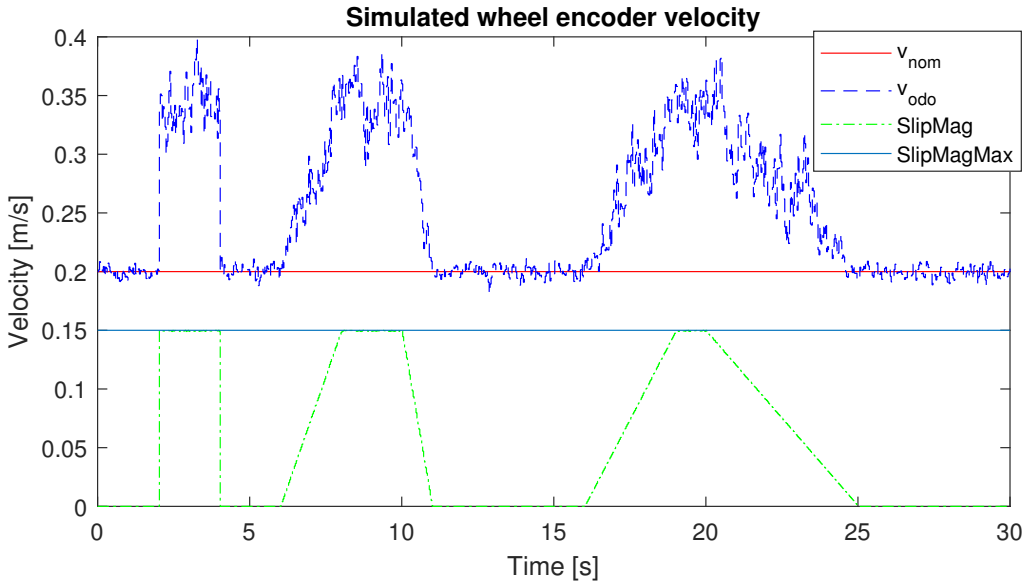


Figure 2-4: Graph of simulated wheel encoder data corrupted by the sensor noise model and longitudinal wheel slip. The velocity profile '$SlipMag$' is fixed, however, its maximum magnitude '$SlipMagMax$' can be varied.

Heading drift modeling

The characteristics that describe the heading drift prescribe that the global heading orientation, provided by the IMU, should drift with a drift rate of between ± 0.0037 $[rad/s]$. The drift is simulated using

$$\theta(k+1) = \theta(k) + Drift \cdot Ts, \tag{2-4}$$

where $\theta(k)$ is the IMU heading output, depicted in Figure 1-4, at sample k, $Drift$ is a parameter between $[-0.0037 \quad 0.0037]$, and Ts is the sampling time.

2-3-3 Overview

The objective of this section was to *design sensor noise and perturbation models* so that the models fulfill the requirements stated at the beginning of this section. A PSD fitting method was used to capture the time correlation of the sensor noise that is prevalent in the sensor output, obtaining the discrete time transfer functions summarized in Table 2-4. Variable parameters, depicted in Table 2-5 are added to the perturbation models such that the characteristics can be modified for algorithm evaluation purposes.

Perturbation	Parameter	Value	Unit
Maximum longitudinal slip magnitude	'$SlipMagMax$'	$[0 \quad 0.15]$	$[m/s]$
Slip mode transition point	'$SlipTrans$'	$[0 \quad 100]$	%
Heading drift rate	'$Drift$'	$[-0.0037 \quad 0.0037]$	$[rad/s]$

Table 2-5: Table summarizing the external perturbation parameter ranges. The parameters are used in Chapters 4, 5 and 6 to simulate the proposed working principles with respect to different parameter settings. The specific tuning of the parameters settings can help gain insight into what limitations the working principles have.

Limitations The frequency spectrum fitting method is limited in the sense that it assumes the noise is normally distributed over the entire frequency spectrum. However, this is not the case and so the influence of the non-Gaussian distribution of the sensor noise can not be studied. Furthermore, the method only captures background noise since the measurements were done in a stationary situation. Both the noise models and the perturbation models are limited as the characteristics were obtained from a single operation and it is likely that these characteristics vary from operation to operation. However, the external perturbation models were designed such that varying conditions can be emulated.

2-4 Ground truth modeling

As was already stated in Section 2-3, Fleet Cleaner currently uses real sensor data to evaluate, among others, new localization algorithms. The downside of this method is that it is difficult or even impossible to evaluate the absolute performance of the localization algorithms because no absolute localization, or ground truth, is available.

A ground truth model is needed that outputs the orientation and position of the robot in three dimensions, given the control inputs $\alpha(t)$ and $v_{true}(t)$. Assuming that this ground truth model is correct, it can be used to verify the correct working of a 3D position estimator by checking if the position estimator tracks the trajectory of the ground truth model. It can also be used to simulate curved surfaces like that of a ship hull to evaluate algorithms designed in coming chapters.

The objective of this section is thus to *produce a ground truth model that can be used to validate the 3D position estimator* such that the effect of the external perturbations and sensor noise on the position estimator and the effectiveness of the working principles that limit the error build-up can be evaluated.

The requirements for the ground truth models are:

- The ground truth model must output an orientation and position given the inputs $\alpha(k)$ and $v_{true}(k)$. It is assumed that if the orientation provided by the ground truth model is used in conjunction with the same inputs by a position estimator and the two trajectories coincide, the position estimator is validated.
- The ground truth model must produce an orientation that changes in all possible direction, i.e. it cannot produce a set of orientations that only span a flat surface.
- The position of the robot must be linked to its orientation. The purpose of this is that it will simulate that the robot is confined to a three dimensional surface.

2-4-1 Ground truth velocity and acceleration models

In Section 2-2 it was shown that the true velocity of the robot during slippage depends on whether dynamic or stationary slip occurs. From the camera images it was observed that the robot velocity during stationary slip is zero. The true velocity during dynamic slippage cannot be gauged due to a lack of an absolute velocity sensor. It is assumed that the true velocity of the robot during slippage is about 30% that of the nominal wheel velocity. The true velocity is modeled using

$$v_{true}(k) = \begin{cases} 0 & \text{if } 0 < SlipMag \geq SlipTrans \cdot SlipMagMax \\ 0.3 \cdot v_{nom}(k) & \text{if } 0 < SlipMag < SlipTrans \cdot SlipMagMax \\ v_{nom}(k) & \text{if } SlipMag = 0 \end{cases}$$

. where '$SlipTrans$' is a parameter between 0 and 1 that determines at what percentage of the maximum longitudinal slip magnitude the slip mode transitions from dynamic to stationary slip. The true acceleration is modeled using with

$$acc(k) = \frac{(v_{true}(k) - v_{true}(k-1))}{Ts}. \tag{2-5}$$

2-4-2 Frenet-Serret model

The use of the Frenet-Serret model [30] allows for the generation of a trajectory on a manifold, given control input $v_{true}(t)$ and $\alpha(t)$ as will be shown next.

The trajectory in x-,y-,z-coordinates on a three dimensional real manifold along with the orientation represented by the Frenet trihedron vectors T,N and $B \in \mathbb{R}^3$, as displayed in Figure 2-5, given the forward velocity $\frac{ds}{dt}$, can be generated using [30]

$$\frac{d}{dt}\begin{bmatrix} \boldsymbol{\mu} \\ T \\ N \\ B \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \kappa & 0 \\ 0 & -\kappa & 0 & \tau \\ 0 & 0 & -\tau & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu} \\ T \\ N \\ B \end{bmatrix} \cdot \frac{ds}{dt}, \tag{2-6}$$

where $\boldsymbol{\mu}(t) \in \mathbb{R}^3$ is the vector representing the x-,y-,z-coordinates of the robot, $\frac{ds}{dt}$ is the velocity in the tangential direction of the space curve and κ and $\tau \in \mathbb{R}$ are the curvature and torsion parameters that describe the shape of the manifold [31].

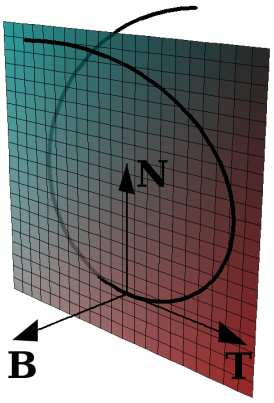


Figure 2-5: Illustration of T,N and B along a space curve [32].

In order to include the 2D kinematics in the Frenet-Serret model, let

$$\boldsymbol{\omega} = \frac{\sin(\alpha) \cdot v_{true}}{l} \cdot N$$

be the vector that describes rotation, induced by the 2D kinematics, around the unit normal vector N. The rate of change of the Frenet trihedron vectors under rotation $\boldsymbol{\omega}$ can then be described by

$$\frac{d}{dt}\begin{bmatrix} T \\ N \\ B \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega} \times T \\ \boldsymbol{\omega} \times N \\ \boldsymbol{\omega} \times B \end{bmatrix}. \tag{2-7}$$

Adding Equation (2-7) to the Frenet-Serret model in Equation (2-6) yields

$$\frac{d}{dt}\begin{bmatrix} \boldsymbol{\mu} \\ T \\ N \\ B \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \kappa & 0 \\ 0 & -\kappa & 0 & \tau \\ 0 & 0 & -\tau & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu} \\ T \\ N \\ B \end{bmatrix} \cdot \frac{ds}{dt} + \begin{bmatrix} 0 \\ \boldsymbol{\omega} \times T \\ \boldsymbol{\omega} \times N \\ \boldsymbol{\omega} \times B \end{bmatrix}. \tag{2-8}$$

In the Frenet-Serret framework the scalar $\frac{ds}{dt}$ specifies the velocity of the Frenet trihedorn in the direction of unit vector T. Furthermore, it is preferred that the vector T, coincides with the robot body axis for all steering angle inputs α. The origin of the trihedron and the direction of T thus must be chosen in a point where the only direction of movement is in the direction of T and where T coincides with the robot body axis. One such point is on the intersection of the robot body axis and the robot wheel axis, denoted point 'O', displayed in Figure 2-6. Let the velocity in the trihedron origin be denoted v_O, the steering angle of the front wheel be denoted α and the forward velocity of the front wheel be denoted v_{odo}. Furthermore, it is assumed that the wheels do not slip and therefore that the wheels share an instantaneous axis of rotation in p_2 with magnitude ω_{p2}, displayed in Figure 2-6.

The velocity in the origin of the trihedron is then calculated with

$$v_o = \omega_{p2} \cdot |p_2 - O|, \quad \in \mathbb{R} \tag{2-9}$$

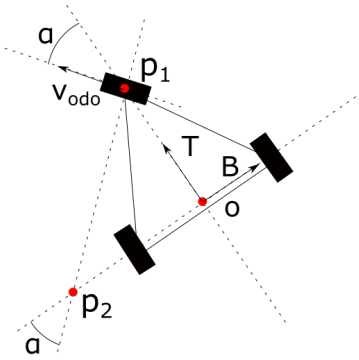


Figure 2-6: Illustration of top-down view on robot. The parameter α describes the steering angle of the robot, v_{odo} the wheel encoder velocity, O the origin of the Frenet trihedron comprised by the vectors (T, N, B), p_1 the rotation axis of the front wheel and p_2 the instantaneous axis of rotation of all the wheels.

using Equation (1-2) it can be seen that

$$\omega_{p2} = \frac{\sin(\alpha) \cdot v_{true}}{|p_1 - O|}. \tag{2-10}$$

Furthermore, using Figure 2-6 it is determined that

$$|p_2 - O| = \frac{|p_1 - O|}{\tan(\alpha)}. \tag{2-11}$$

Substituting Equations (2-11) and (2-10) into Equation (2-9) yields

$$v_o = \frac{\sin(\alpha)}{\tan(\alpha)} \cdot v_{true} = \cos(\alpha) \cdot v_{true}. \tag{2-12}$$

This yields the robot fixed velocity vector

$$v_O = T \cdot v_o = \begin{bmatrix} v_o & 0 & 0 \end{bmatrix}, \quad \in \mathbb{R}^3. \tag{2-13}$$

Substituting $|v_O|$ in Equation (2-8) yields

$$\frac{d}{dt}\begin{bmatrix} \boldsymbol{\mu} \\ T \\ N \\ B \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \kappa & 0 \\ 0 & -\kappa & 0 & \tau \\ 0 & 0 & -\tau & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu} \\ T \\ N \\ B \end{bmatrix} \cdot v_o(t) + \begin{bmatrix} 0 \\ \boldsymbol{\omega} \times T \\ \boldsymbol{\omega} \times N \\ \boldsymbol{\omega} \times B \end{bmatrix}, \tag{2-14}$$

which is the continuous-time Frenet-Serret model, used to model the ground truth trajectories.

2-4-3 Overview

The objective of this section was to *produce a ground truth model that can be used to validate the 3D position estimator and so that the curved surface of a ship hull can be simulated.* Using the Frenet-Serret framework a model was established that produces the position $\boldsymbol{\mu}(t)$ and orientation $R(t)$ using the inputs $v_{true}(t)$ and $\alpha(t)$. Furthermore, $\boldsymbol{\mu}(t)$ and $R(t)$ are not arbitrary, they are confined to the robot moving on a manifold $\mathcal{M}$ parameterized by κ and τ.

2-5 Conclusion

The objective of this chapter was to *acquire knowledge about the sensor noise and external perturbation characteristics* and to *generate sensor data models and a ground truth model.*

The sensor noise characteristics were determined by evaluating the PSDs and histogram distributions of stationary sensor data sequences. Furthermore, the heading drift and longitudinal wheel slip were identified as the most significant external perturbations. The heading drift rate was determined by measuring the rate of change in the IMU heading output while the robot was stationary. The longitudinal wheel slip characteristics were determined by analyzing camera footage of the front wheel and the data associated to that footage.

Using a spectrum fitting method, transfer functions were designed that capture the noise correlation and uncertainty of the sensor noise. External perturbation models were designed that capture the characteristics of the observed external perturbations. In order to emulate the movement of the robot on a curved manifold, the Frenet-Serret model was employed, providing a simulated ground truth trajectory.

Results The analysis of the sensor signals revealed that the only the depth sensor and the gyroscope signal are normally distributed and that only the noise of the gyroscope has an approximately white PSD. The analysis of camera footage revealed that the front wheel of the robot suffers greatly from wheel slip, inducing a bias in the wheel encoder output of up to 0.15 $[m/s]$. The analysis of the IMU heading output revealed that the heading drifts up to a rate of 0.0037 $[rad/s]$. The control inputs to the robot, external perturbation parameters and Frenet-Serret model parameters that are used throughout the thesis to simulate sensor data and ground truth trajectories, are summarized in Table 2-6. A schematic overview of the simulated ground truth and sensor models is provided in Appendix A-4. The model allows for the simulation of ship hull-like curved surfaces, the validation of the 3D position estimator and provides a simulated ground truth for the evaluation the algorithms designed in coming chapters.

Description	Parameter	Value	Unit
Robot control inputs			
Desired nominal forward velocity	v_{nom}	$[0 \quad 0.3]$	$[m/s]$
Steering angle	α	$[0 \quad 0.6]$	$[rad]$
External perturbation parameters			
Max. longitudinal wheel slip magnitude	'$SlipMagMax$'	$[0 \quad 0.15]$	$[m/s]$
Slip mode transition point	'$SlipTrans$'	$[0 \quad 100]$	[%]
Heading drift rate	'$Drift$'	$[-0.0037 \quad 0.0037]$	$[rad/s]$
Frenet-Serret manifold parameters			
Curvature	κ	$\mathbb{R}$	-
Torsion	τ	$\mathbb{R}$	-
Length of simulation	$Tfinal$	$[0 \quad \infty]$	$[s]$

Table 2-6: Table summarizing simulated robot control inputs, external perturbation parameters and Frenet-Serret model parameters.

Limitations The spectrum fitting method, used to obtain the sensor noise models, is limited because it assumes that the noise is normally distributed over the entire spectrum, which was not the case. The method was chosen because it is an easy first approximation of the sensor noise PSDs and it does not require the evaluation of the sensor noise distribution over the entire frequency range and the evaluation of the sensor noise PSD over time. More modest external perturbations like orthogonal wheel slip, wheel deformation, sensor misalignment and water pressure waves were left out of consideration. The observed external perturbation characteristics were obtained from camera footage originating form a single cleaning operation, reducing the reliability of the characteristics. Furthermore, the characteristics were modeled using parameters and no dynamics were taken into consideration.

Recommendations If more realistic sensor noise models are required, it is recommended that large data samples are taken at different time instances to model the sensor noise using more complex statistical analysis methods [23]. The reliability of the identified longitudinal wheel slip characteristics can be improved by adding a ground truth velocity sensor to the robot. This will allow for the evaluation of the wheel encoder output with respect to a ground truth velocity. If such a ground truth velocity is available it will improve the analysis of the longitudinal wheel slip characteristics analysis. Furthermore, it will be possible to identify a dynamic model between the wheel encoder output and the true velocity.

Chapter 3

Position estimation

Currently, the position of the robot is estimated in only two dimensions using the non-linear kinematics equation depicted in Equation (1-2). This induces error build-up since the robot operates on a three dimensional curved surface and is incompatible with aspirations to making the robot autonomous. Furthermore, it is yet unknown to what degree the perturbations, discussed in Chapter 2 affect the position estimate.

Fleet Cleaner thus requires a 3D position estimate to reduce error build-up and so that the robot can ultimately be made autonomous. Additionally, knowledge of how much the 3D estimated trajectory is corrupted by the perturbations discussed in Chapter 2 is valuable information in itself, but will also help to set-up a strategy to effectively combat the error build-up in the position estimate.

The objective of this chapter is thus to *design a 3D position estimator* and *evaluate the effect of the simulated perturbations on the position estimator*, designed in Chapter 2.

In Section 3-1 literature is reviewed to establish a working principle that can be used to estimate the position in three dimensions. After a working principle is established, its workings are validated in Section 3-2 and the effects of the simulated perturbations designed in Chapter 2 on the position estimator are studied. Finally, in Section 3-3 the obtained results will be summarized and a strategy for reducing error build-up is proposed.

3-1 3D position estimators

As was previously discussed, the current position is estimated in only two dimensions and in order to reduce error build-up - and to ultimately make the robot autonomous - a *suitable* 3D position estimator is required. In order to design a suitable position estimator several facets of position estimator design need to be reviewed, which is elaborated on next.

Position estimation for mobile robots is almost always achieved using some sort of *Bayesian filter* [33, 34, 35] depicted in Algorithm 1 [17, p. 27], as it accommodates the fusion of a

dynamic or kinematic model prediction with sensory inputs and uses their respective uncertainty to obtain an optimal estimator. Essentially a Bayesian filter estimates the belief[1] of the orientation and position (or *pose*) of the mobile robot, denoted μ_t, by alternating between predicting the belief using control input u_t in **step 2** with a *state transition model*

Algorithm 1 Bayes filter

1: **function** BF($bel(\mu_{t-1})$,u_t,z_t)
2: $bel(\bar{\mu}_t) = \int p(\mu_t|\mu_{t-1}, u_t)bel(\mu_{t-1})d\mu_{t-1}$
3: $bel(\mu_t) = \eta \cdot p(\boldsymbol{z}_t|\mu_t)bel(\bar{\mu}_t)$
4: **return** $bel(\mu_t)$

and adjusting this predicted belief with a sensor measurement z_t in **step 3**. These steps are called the *prediction step*, yielding a *prior distribution* of the state, and *measurement update*, yielding the *posterior distribution* of the state, respectively [36]. The Bayes filter framework is very general and leaves a couple of degrees of freedom for the design of a position estimator. Firstly a choice has to be made about what algorithm will be used to estimate the prior density and the posterior density of the state. Secondly, possible ways of fusing sensor data in **step 3** to improve the estimate need to be reviewed. And thirdly, based on the two previous design steps sensor and process noise models that are used to estimate the prior and posterior densities of the state need to be designed.

The objective of this section is thus *to establish a working principle for a 3D position estimator suitable for the Fleet Cleaner robot* by answering the following questions:

1. What is a suitable position estimator for the Fleet Cleaner robot given the sensor noise characteristics, external perturbation characteristics and non-linear state transition function?
2. What fusion capabilities are there to improve the position estimate?
3. How should the process and measurement noise models be set-up, given the choices made in the previous two points?

3-1-1 Position estimator comparison

A multitude of position estimators have been used to localize underwater mobile robots, the more popular ones [37] ranging from *parametric filters* like the EKF [38] and Unscented Kalman Filter (UKF) [39] to *non parametric filters* like the Particle filter (PF) [36]. The EKF, PF and UKF workings are depicted in Appendices A-5, [17, p. 96] and [17, p. 65], respectively. In this section the three different filters compared and a selection is made among them.

EKF The EKF is popular because it is an extension on the famous Kalman Filter (KF) for non-linear state transition models, it is relatively easy to design and code and often provides good prediction accuracy [40]. Additionally, the EKF is computationally efficient, where each

[1] The belief reflects the knowledge about the orientation and position of the robot that is not gauged directly by a sensor measurement but is rather inferred from data using a model.

update requires time $\mathcal{O}(k^{2.4}+n^2)$, where $k = dim(\boldsymbol{z}_t)$ and $n = dim(\boldsymbol{\mu}_t)$. In the face of highly non-linear state transition models the EKF produces bad estimates because the linearization step doesn't adequately approximate the non-linear system [40], [17, p. 61]. If a Gaussian with a low uncertainty is mapped by a non-linear function, the image may still approximate a Gaussian. However, the higher the uncertainty of the Gaussian that is being mapped by the non-linear state transition function, the less its image will approximate a Gaussian [17, p. 61], making the EKF unsuitable for systems with highly non-linear state transition functions. The EKF framework is based on the assumption that the measurement noise PSD is white.

Particle filter Non-parametric filters like the PF are desirable when the state transition function is highly non-linear [41], because they don't rely on a first-order approximation like the EKF. Another advantage of PFs is that they allow for arbitrary sensor noise characteristics and the state and measurement inputs do not have to have a Gaussian distribution [36]. This does, however, require the modeling of the sensor noise distributions. In situations where the states my abruptly change because of a sudden displacement, the PF is able to recover from this so called 'robot kidnapping' [17, 36, 42], whereas the EKF will not. This is a significant feature especially in *global localization*, where the initial location of the robot is unknown.

A drawback of the PF is that the improved robustness and versatility is burdened with an high computational cost of $\mathcal{O}(c^n), \quad c > 1$ [40, 43], this is especially prominent when the state dimension is high [41], meaning 8 or higher [40].

Unscented Kalman filter It is often asserted that due to the non-linearity in the state-transition model of mobile robots, the UKF is superior to the EKF when localizing mobile robots [44, 45, 46, 47]. This is because the non-linear approximation properties of the UKF are generally better than those linear approximation of the of the EKF [48]. A drawback of the UKF with respect to the EKF is that it has a higher computational complexity [49].

The advantages of all the position estimators are summarized below to get a good overview.

Comparison

The main advantage of the UKF and the PF over the EKF seem to be that they can better handle the non-linear state transition model of the robot kinematics, depicted in Equation (1-2). Advantages that solely the PF has over the EKF is that it can handle non-Gaussian sensor inputs, that it can approximate the non-Gaussian distribution of the posterior distribution and that it can recover from robot kidnapping. Below the advantages and disadvantages are compared to each other and a selection among the position estimators is made.

System non-linearity L. D'Alfonso et al., conducted a comparative experiment on a mobile robot similar to the Fleet Cleaner robot and showed that the linearization errors induced by the first order Taylor approximation made in the EKF are negligible [48]. They state that this is probably due to the fact that the non-linearities in the system model 'are not bad enough to highlight any substantial difference'. The same conclusion was reached by M. St-Pierre et al., in a comparative study between the UKF and EKF in localizing a mobile robot [39]. The estimation performance of the UKF and the EKF in estimating quaternions using noisy

sensor signals was studied in [49] and they also conclude that the UKF, although it has higher computational complexity, has equivalent performance to that of the EKF. They ascribe this to the fact that the covariances of the quaternions are significantly smaller (10^{-4} to 10^{-6}) than unity, making the higher order moments very small and thus a Gaussian approximation of the distributions more appropriate. Furthermore, if the sampling rate is sufficiently high, meaning 25 $[Hz]$ or higher, the quaternion dynamics behave in a quasi linear fashion, making the linearization error of the EKF minimal. The fact that the Fleet Cleaner robot is physically bigger and slower moving than the aforementioned researcher's robots, and that the sampling frequency 32 $[Hz]$, makes the linear approximation even more apt at capturing the necessary dynamics.

Colored noise An argument is to be made in favor of the PF because it can handle the arbitrary noise distributions that were observed in Section 2-1. However, in order to take advantage of this feature the sensor noise distributions need to be modeled first and as was already stated in Section 2-1, the distributions are difficult to obtain in dynamic situations and are not constant. Furthermore, the EKF has been applied with great success in state estimation problems that violate the underlying assumptions of the noise being AWGN [17, p. 61]. Adding to this, the covariances of the sensor inputs are in the order of 10^{-5} to 10^{-6}, making them more suitable to approximate with a Gaussian [49].

Robot kidnapping The case of robot kidnapping also does not really apply to the Fleet Cleaner robot since if the robot is dislodged from he ship hull, the states are initialized on the side of the ship hull, using the depth gauge, absolute orientation measurements and operator input.

Selection The PF and UKF thus do not have a significant advantage over the EKF in dealing with the system non-linearity and non-Gaussian sensor distributions and so the EKF is used to estimate the position of the robot. It is not argued that the EKF is the best position estimator, however comparing different position estimators using simulations would take up too much time compared to the amount of performance increase it is expected to yield.

3-1-2 Sensor fusion

The current fusion scheme is displayed in Appendix A-6, Figure A-6.1 (Left). This method of position estimation leaves a lot of available sensor inputs unused, namely the rotational velocity provided by the 2D kinematics, depicted in Equation (1-2) and the linear acceleration output of the IMU. Methods for incorporating the acceleration data and rotational velocity data in the position estimate are discussed next.

Rotational velocity fusion The accuracy of the position estimate, to some degree, relies on the accuracy of the orientation estimate, which is currently provided by the IMU. In their paper, Zhou et al., [50] have shown that the position estimate error of an indoor mobile robot

can be reduced by up to 4.5% by fusing[2] an orientation estimate obtained by a model such as the one depicted in Equation (1-2) with an absolute orientation input. Such a fusion scheme is displayed in Appendix A-6, Figure A-6.1 (Right).

The way in which the rotational velocity is fused with the orientation output of the IMU depends on the parameterization of the attitude[3] of the robot. Common attitude representations are exponential coordinates, Euler angles and Quaternions. Like exponential coordinates, Euler angles have the downside that the rotation matrix becomes singular for some rotations [51, p. 31]. Fundamentally, such singularities arise using 3-dimensional attitude representations [51, p. 33]. Quaternions give a global parametrization of $SO(3)$ meaning that, unlike the Euler angle representation, singularities do not arise [51, p. 33]. Quaternions are commonly applied in spacecraft for on-board inertial navigation [52, p. 412] and have been proven effective in fusing inertial data from a gyroscope with orientational IMU data [53], which is equivalent to fusing the rotational velocity, predicted by the steering angle and forward wheel velocity, with IMU data as is the case for the Fleet Cleaner robot. Considering the previously discussed advantages and disadvantages of different representations the quaternion is most preferable for the Fleet Cleaner robot.

Linear acceleration fusion A common use for the accelerometer data is to design a so called 'strap-down' [54] filter that fuses the gyroscopic output with that of the accelerometer to obtain a better attitude[4] estimation [55, 56]. The orientation provided by the IMU is already calculated using such a strap-down filter [10, p. 16-18] and so using the acceleration data to improve the attitude estimation is superfluous. However, fusing the accelerometer data with that of the wheel encoder may improve the position estimate since the acceleration data does not become biased like the wheel encoder output when wheel slip occurs.

Fusion schemes

The two different fusion schemes previously discussed leave a total of four fusion schemes to be evaluated, namely:

1. The current fusion scheme, displayed in Appendix A-6, Figure A-6.1 (Left).
2. The rotational velocity - IMU orientation fusion scheme, displayed in Appendix A-6, Figure A-6.1 (Right).
3. The wheel encoder - IMU acceleration fusion scheme, displayed in Appendix A-6, Figure A-6.2 (Left).
4. The wheel encoder - rotational velocity - IMU scheme, which is basically a combination of the two previous fusion schemes, displayed in Appendix A-6, Figure A-6.2 (Right).

The models for the fusion schemes are derived next.

[2]Sensor fusion is the practise of combining sensory data from different sensors such that the resulting data has less uncertainty than those data sources would have individually [41].

[3]Attitude is a synonym for orientation.

[4]The attitude of a mobile robot refers to its orientation with respect to a global frame.

Fusion scheme 1 A vector $\boldsymbol{x}$ in $\mathbb{R}^3$ can be rotated by a quaternion using [57][5]

$$\boldsymbol{x}' = R(\boldsymbol{q})\boldsymbol{x}, \tag{3-1}$$

where

$$R(\boldsymbol{q}) = \begin{bmatrix} (1-2q_2^2-2q_3^2) & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & (1-2q_1^2-2q_3^2) & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & (1-2q_1^2-2q_2^2) \end{bmatrix}. \tag{3-2}$$

The time derivative of the location of the robot can then be calculated by rotating the robot fixed velocity vector v_O, in Equation (2-13), to the ship hull frame using the model

$$\begin{aligned} \dot{\boldsymbol{x}}(t) = g(\boldsymbol{x}(t), \boldsymbol{u}(t)) = & \quad \begin{bmatrix} \dot{\mu}_1 \\ \dot{\mu}_2 \\ \dot{\mu}_3 \end{bmatrix} = \begin{bmatrix} v_{odo} \cdot \cos(\alpha)(1-2q_2^2-2q_3^2) \\ 2 \cdot v_{odo} \cdot \cos(\alpha)(q_1q_2-q_0q_3) \\ 2 \cdot v_{odo} \cdot \cos(\alpha)(q_1q_3+q_0q_2) \end{bmatrix} \\ \boldsymbol{y}(t) = h(\boldsymbol{x}(t), \boldsymbol{u}(t)) = & \quad y = \begin{bmatrix} \mu_2 \end{bmatrix}, \end{aligned} \tag{3-3}$$

where $\boldsymbol{u}(t) = [v_{odo}(t) \quad \alpha(t) \quad q_0 \quad q_1 \quad q_2 \quad q_3]^T$ and $\boldsymbol{x}(t) = [\mu_1(t) \quad \mu_2(t) \quad \mu_3(t)]^T$. The quaternion components are denoted q_i, the coordinates $[x(t) \quad y(t) \quad z(t)]^T = [\mu_1(t) \quad \mu_2(t) \quad \mu_3(t)]^T$, $v_{odo}(t)$ is the wheel encoder output and $\alpha(t)$ is the steering angle.

Fusion scheme 2 Let $\boldsymbol{\omega} = [\omega_x \quad \omega_y \quad \omega_z]^T = [\omega_1 \quad \omega_2 \quad \omega_3]^T$ be the rotational velocities expressed in the robot frame and $\boldsymbol{q} = [q_0 \quad q_1 \quad q_2 \quad q_3]^T$ the quaternion describing the orientation of the robot with respect to the ship hull fixed reference frame. The time evolution of the quaternion [58] can then be described by

$$\dot{q} = \frac{1}{2}\boldsymbol{q} \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}, \tag{3-4}$$

using the *quaternion product* [59] between $\boldsymbol{q}$ and $\boldsymbol{\omega}$. The non-linear model describing the time-evolution of the quaternion can be represented by

$$\begin{aligned} \begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} &= \begin{bmatrix} \frac{1}{2||q||}(-\omega_1q_1-\omega_2q_2-\omega_3q_3 \\ \frac{1}{2||q||}(-\omega_3q_2+\omega_1q_0-\omega_2q_3) \\ \frac{1}{2||q||}(\omega_2q_0+\omega_1q_2-\omega_3q_1) \\ \frac{1}{2||q||}(-\omega_1q_2+\omega_2q_1+\omega_3q_0) \end{bmatrix} \\ \boldsymbol{y} &= \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \end{aligned} \tag{3-5}$$

By inserting the rotational velocity predicted by Equation (1-2) into Equation (3-5) for ω_3, a model prediction for the quaternions is obtained and can thus be fused with the IMU quaternion output using the standard EKF framework.

[5]The notation indicating time dependency of the variables is omitted

Combining Equations (3-5), (1-2) and (3-1) yields the centralized non-linear model that describes the time evolution of the quaternions and positions:

$$\begin{aligned}
\dot{\boldsymbol{x}}(t) = g(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \quad & \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{\mu}_1 \\ \dot{\mu}_2 \\ \dot{\mu}_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2||q||}(-\omega_1 q_1 - \omega_2 q_2 - v_{odo}\frac{\sin(\alpha)}{l} q_3 \\ \frac{1}{2||q||}(-v_{odo}\frac{\sin(\alpha)}{l} q_2 + \omega_1 q_0 - \omega_2 q_3) \\ \frac{1}{2||q||}(\omega_2 q_0 + \omega_1 q_2 - v_{odo}\frac{\sin(\alpha)}{l} q_1) \\ \frac{1}{2||q||}(-\omega_1 q_2 + \omega_2 q_1 + v_{odo}\frac{\sin(\alpha)}{l} q_0) \\ v_{odo} \cdot \cos(\alpha)(1 - 2q_2^2 - 2q_3^2) \\ 2 \cdot v_{odo} \cdot \cos(\alpha)(q_1 q_2 - q_0 q_3) \\ 2 \cdot v_{odo} \cdot \cos(\alpha)(q_1 q_3 + q_0 q_2) \end{bmatrix} \\
\boldsymbol{y}(t) = h(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \quad & \boldsymbol{y} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ \mu_2 \end{bmatrix}.
\end{aligned} \tag{3-6}$$

where $\boldsymbol{u}(t) = [v_{odo}(t) \quad \alpha(t) \quad \omega_1(t) \quad \omega_2(t)]^T$, $\boldsymbol{x}(t) = [q_0 \quad q_1 \quad q_2 \quad q_3 \quad \mu_1(t) \quad \mu_2(t) \quad \mu_3(t)]^T$ and l is the wheel base, depicted in Figure 1-3.

Fusion scheme 3 The acceleration data is fused by making the forward velocity of the robot origin a state of the system and using the wheel encoder velocity as a sensor input to the system, depicted in Equation (3-7) and shown in Appendix A-6, Figure A-6.2 (Left). In this fusion scheme the quaternions are not estimated and are used as a model input.

$$\begin{aligned}
\dot{\boldsymbol{x}}(t) = g(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \quad & \begin{bmatrix} \dot{v}_o \\ \dot{\mu}_1 \\ \dot{\mu}_2 \\ \dot{\mu}_3 \end{bmatrix} = \begin{bmatrix} a_x \\ v_o \cdot (1 - 2q_2^2 - 2q_3^2) \\ 2 \cdot v_o \cdot (q_1 q_2 - q_0 q_3) \\ 2 \cdot v_o \cdot (q_1 q_3 + q_0 q_2) \end{bmatrix} \\
\boldsymbol{y}(t) = h(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \quad & \boldsymbol{y} = \begin{bmatrix} v_o \\ \mu_2 \end{bmatrix}.
\end{aligned} \tag{3-7}$$

where $\boldsymbol{u}(t) = [a_x(t) \quad q_0(t) \quad q_1(t) \quad q_2(t) \quad q_3(t)]^T$ and
$\boldsymbol{x}(t) = [v_o(t) \quad \mu_1(t) \quad \mu_2(t) \quad \mu_3(t)]^T$. The robot frame fixed velocity is denoted $v_o = \cos(\alpha) v_{odo}$, as shown in Equation (2-9).

Fusion scheme 4 The rotational velocity and acceleration fusion schemes can also be combined as displayed in Appendix A-6, Figure A-6.2 (Right) and depicted in Equation (3-8).

$$\dot{\boldsymbol{x}}(t) = g(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \begin{bmatrix} \dot{v}_o \\ \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{\mu}_1 \\ \dot{\mu}_2 \\ \dot{\mu}_3 \end{bmatrix} = \begin{bmatrix} a_x \\ \frac{1}{2||q||}(-\omega_1 q_1 - \omega_2 q_2 - v_{odo}\frac{\sin(\alpha)}{l} q_3 \\ \frac{1}{2||q||}(-v_{odo}\frac{\sin(\alpha)}{l} q_2 + \omega_1 q_0 - \omega_2 q_3) \\ \frac{1}{2||q||}(\omega_2 q_0 + \omega_1 q_2 - v_{odo}\frac{\sin(\alpha)}{l} q_1) \\ \frac{1}{2||q||}(-\omega_1 q_2 + \omega_2 q_1 + v_{odo}\frac{\sin(\alpha)}{l} q_0) \\ v_o \cdot (1 - 2q_2^2 - 2q_3^2) \\ 2 \cdot v_o \cdot (q_1 q_2 - q_0 q_3) \\ 2 \cdot v_o \cdot (q_1 q_3 + q_0 q_2) \end{bmatrix} \tag{3-8}$$

$$\boldsymbol{y}(t) = h(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \boldsymbol{y} = \begin{bmatrix} v_o \\ q_0 \\ q_1 \\ q_2 \\ q_3 \\ \mu_2 \end{bmatrix},$$

where $\boldsymbol{u}(t) = [v_{odo}(t) \quad \alpha(t) \quad \omega_1(t) \quad \omega_2(t)]^T$ and
$\boldsymbol{x}(t) = [v_o \quad q_0 \quad q_1 \quad q_2 \quad q_3 \quad \mu_1(t) \quad \mu_2(t) \quad \mu_3(t)]^T$.

Linearization and discretization The EKF framework depicted in Appendix A-5 requires the calculation of the linearized system matrices G_t, F_t and H_t. Furthermore, the linearized system matrices need to be discretized such that they can be implemented on Fleet Cleaner computer hardware. The linearized system matrices are calculated by first calculating the Jacobian of $g()$ and $h()$ with respect to the states and model inputs using the Matlab function *jacobian()*. The Jacobian of the non-linear system is then evaluated at each time step t for the non-linear model predicted states $\bar{\boldsymbol{x}}_t$, and $\boldsymbol{u}_t$ to obtain the linearized system matrices. The linear system matrices are then discretized using the forward Euler integration method.

3-1-3 Noise matrix design

The process noise matrices of the EKF is often tuned by trial-and-error. Because this method can take up a considerable amount of time, the noise matrix is often made diagonal so that a minimal amount of parameters require tuning. This however negates the cross-correlation between states and can result in poor filter performance [60]. Furthermore, improper design of the noise matrices Q and R can greatly reduce the performance of the EKF, and can even cause it to diverge [61]. The sensor noise matrix R in all cases is set as [60]

$$R = \begin{bmatrix} var(s_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & var(s_M) \end{bmatrix}, \tag{3-9}$$

where $var(s_j)$ is the variance of the j^{th} sensor input, estimated in Chapter 2.

As was determined in the previous section, the position estimator may have up to 8 states. The process noise matrix will thus be an 8×8 matrix with $8(8+1)/2$ tuneable noise matrix entries, assuming Q is symmetric [21]. It is thus apparent that an automated method for designing the process noise matrix is needed to limit the amount of time spent tuning the matrix. Furthermore, estimating the cross-correlations between states using such an approach may improve the estimator performance beyond that of an estimator using a diagonal process noise matrix.

Methods for estimating the process noise covariance can be broadly classified into four categories: covariance matching, correlation techniques, Bayesian and maximum likelihood methods, among which the maximum likelihood methods seem to be the most promising for estimating process noise covariance matrices associated with non-linear systems [62]. Valappil and Georgakis [60] developed an algorithm for determining a time-varying process noise matrix, depending on the uncertainty in the process model parameters.

The time-varying process noise matrix is calculated with

$$Q(t) = k_Q J_{\hat{p}}(t) C_{\hat{p}} J_{\hat{p}}^T(t), \tag{3-10}$$

where

$$J_{\hat{p}}(t) = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} \right)_{\boldsymbol{x}(t), \boldsymbol{u}(t), \hat{\boldsymbol{p}}}, \tag{3-11}$$

$\boldsymbol{f}$ is the non-linear state transition function, $\hat{p}$ is the parameter, $C_{\hat{p}}$ is the covariance matrix associated with the identification of the parameters and k_Q is a tuning parameter, usually set at 1. Instead of using a time-varying process noise matrix $Q(t)$, Schneider et al. construct a time-invariant process noise matrix

$$Q = \frac{1}{KN} \sum_{j=1}^{K} \sum_{k=0}^{N-1} Q_{ii}^j(t_k), \tag{3-12}$$

where $Q_{ii}^j(t_k)$ are the diagonal entries of the time-varying Q matrix at time step t_k of simulation run j, K is the amount of simulation runs and N the amount of samples per run. They conclude that the time-invariant process noise matrix achieves similar performance to that of the time-varying process noise matrix [61], while being less computationally demanding and more simple. However, the time-invariant process noise matrix does not capture the cross-correlation of the states. In Section 3-2 the performance of the two design methods is compared in order to determine if it is worthwhile using a time-varying process noise matrix with off diagonal entries.

3-1-4 Overview

The objective of this section was to *establish a working principle for a 3D position estimator suitable for the Fleet Cleaner robot*, by determining a suitable filter to estimate the states, determine fusion possibilities and to determine a suitable method to determine the process noise models.

In Table 3-1 the position estimator selection and possible fusion scheme and process noise matrix designs are summarized.

Position estimator selection	**EKF**	~~PF~~	~~UKF~~
Fusion schemes	FS1	FS2	FS3	FS4
Process noise matrix	Time-invariant Q	Time-varying Q

Table 3-1: Table summarizing position estimator selection and possible combinations of fusion schemes and process noise matrix designs to be tested. Fusion scheme 1 is the current fusion scheme, fusion scheme 2 with added rotational velocity data fusion, fusion scheme 3 with added acceleration data fusion and fusion scheme 4 is a combination of fusion scheme 2 and 3.

3-2 Extended Kalman Filter simulation

In the previous section, an EKF in conjunction with several fusion schemes, using a time-variant process noise matrix, was proposed a suitable 3D position estimator. This leaves open the following questions:

1. Does the EKF properly estimate the position in three dimensions?
2. Does the linear approximation of the EKF sufficiently capture the non-linear kinematics?
3. Is the time-varying process noise matrix design method an improvement over the time-invariant process noise matrix design method in reducing estimation error induced by noise?
4. What fusion scheme provides the best estimator performance with respect to the simulated sensor noise, and external perturbations?
5. What source for error build-up - i.e. sensor noise, wheel slip and heading drift - contributes most to the error build-up between in the position estimate?

The first two questions will be answered by comparing the EKF estimated position against a trajectory generated by the ground truth model designed in Chapter 2, without sensor noise. The third question will be answered by comparing the estimator performance using the time-varying and time-invariant process noise matrices, with added sensor noise on the sensor inputs. The fourth and fifth question will be answered by corrupting the sensor inputs with the noise and perturbation models, designed in Chapter 2, and comparing the fusion scheme estimation performance. In Appendix A-7 a schematic overview is given of the possible different process noise and fusion schemes combinations.

3-2-1 Ground truth trajectory

Generating a ground truth trajectory using the Frenet-Serret model of Equation (2-6) requires the specification of the intrinsic parameters (κ, τ) and the inputs $(\alpha(t), v_{odo}(t))$. In order to evaluate if the EKF properly estimates the position in three dimensions it suffices to generate a trajectory on a sphere by setting $\tau = 0$ and $\kappa = 1/R$, where R is the radius of the sphere. The accuracy of the linear approximation of the EKF reduces as the radius of the sphere becomes smaller. In order to provide a worst case scenario for the linear approximation of the EKF, the curvature parameter κ is set at $1/2.5$ $[m]$, which is the minimal curvature radius

that the robot is designed to traverse. The Fleet Cleaner technical documentation specifies that the maximum velocity of the robot is 0.3 $[m/s]$ and that the maximum steering angle is 0.61 $[rad]$. Using this information the modeling parameters are set to the values shown in Table 3-2. Furthermore, the sensor noise is switched off.

$v_{nom}(t)$	$\alpha(t)$	$SlipMagMax$	$SlipTrans$	$Drift$	κ	τ	$Tfinal$
$0.3 \cdot \cos(0.01t)$	$0.61 \cdot \sin(0.1t)$	off	off	off	$1/2.5$	0	600

Table 3-2: Table summarizing the control, perturbation and Frenet-Serret inputs used to evaluate the EKF working.

3-2-2 EKF workings validation

The workings of the 3D position estimator and the accuracy of the linear approximation of the EKF is evaluated by comparing the *open loop* EKF estimated position using noise free sensor inputs to the simulated ground truth trajectory provided by the Frenet-Serret model, displayed in Figure 3-1 by the red line. The prediction accuracy is evaluated using the RMSE, which is a common metric for evaluating estimator performance, defined as [63]

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \left(\hat{\boldsymbol{\mu}}_i - \boldsymbol{\mu}_i\right)^2}{N}}, \tag{3-13}$$

where $\hat{\boldsymbol{\mu}}_i$ is the estimated position at instance i and $\boldsymbol{\mu}_i$ is the simulated ground truth position at time instance i [49]. As an example, the noise free sensor inputs generated in the previous section are used to estimate the trajectory components displayed in Figure 3-1 using fusion scheme 1. The estimation of the position using fusion scheme 1 yielded an RMSE score of 0.0052 $[m]$. The RMSE scores of all the fusion schemes are summarized in Table 3-3. From Figure 3-1 and Table 3-3 it is concluded that all the fusion schemes correctly estimate the position in three dimensions and also that the linear approximation of the EKF induces very little estimation error. The fact that the linear approximation induces very little estimation error in part justifies the selection of the EKF as a suitable position estimator since this is often advertised as a main drawback of the EKF.

	Fusion scheme 1	Fusion scheme 2	Fusion scheme 3	Fusion scheme 4
RMSE	0.0052 $[m]$	0.0096 $[m]$	0.0050 $[m]$	0.0071 $[m]$

Table 3-3: Table summarizing the RMSE scores between the simulated ground truth trajectory and the estimated position of the four different fusion schemes. The estimated trajectory was obtained using the modeling parameters specified in Table 3-2 and the sensor inputs were void of noise.

3-2-3 Process noise matrix design method comparison

The performance of the fusion schemes is evaluated using the time-invariant process noise matrix of Equation (3-12) and the time-variant process noise matrix of Equation (3-10) to determine the best process noise matrix design method.

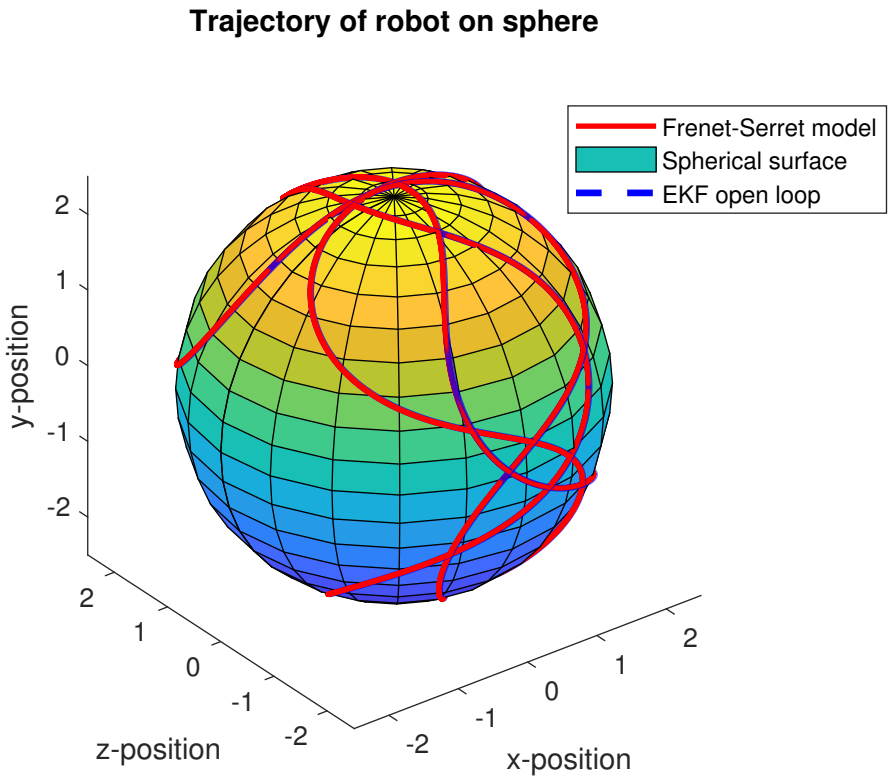


Figure 3-1: Frenet-Serret generated trajectory using intrinsic parameters $(\tau, \kappa) = (0, \frac{1}{2.5})$ and inputs $(\alpha(t), v_{odo}(t)) = (0.61 \cdot \sin(0.1t), 0.3 \cdot \cos(0.01t))$, for a duration of 600 seconds.

The modeling parameter settings summarized in Table 3-2 are used. Also, noise is added to the sensor signals using the sensor noise models depicted in Table 2-4. The realization of the noise affects the estimator performance, so to incorporate this randomness in the RMSE score of the estimators each simulation is iterated 1000 times[6].

The test results shown in Figure 3-2 show that the fusion schemes have comparable estimator performance. The time-varying process noise matrix slightly improves the estimator performance for the second and third fusion schemes and slightly decreases performance for the second and fourth fusion schemes. From these results it is concluded that the added estimation performance of the time-varying process noise matrix is doubtful for this set-up. Furthermore, the fact that it is time-varying makes it computationally more expensive, making it less desirable. For these reasons the time-invariant process noise matrix is selected as most appropriate process noise matrix design method.

Fusion scheme comparison

In order to select the best fusion scheme for the position estimator, the fusion schemes are simulated with sensor inputs corrupted by noise and external perturbations. The simulation

[6]Any amount of iterations beyond $K = 250$ approximated the same RMSE distribution in terms of the mean and standard deviation of the distribution.

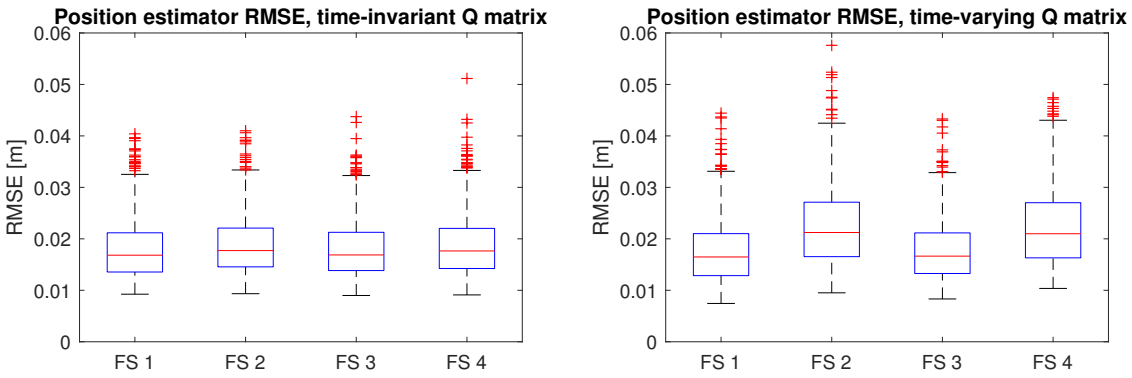


Figure 3-2: Graph displaying RMSE score distributions of the fusion schemes using a time-invariant process noise matrix (Left) and time variant process noise matrix (Right) using boxplots, consisting of $K = 1000$ samples. The sensor inputs used by the estimators were corrupted by the sensor noise models.

parameters are set to the values displayed in Table 3-4. Noise is added to the sensor signals using the sensor noise models. Each simulation is iterated $K = 1000$ times to incorporate the randomness of the sensor noise into the RMSE scores.

$v_{nom}(t)$	$\alpha(t)$	$SlipMagMax$	$SlipTrans$	$Drift$	κ	τ	$Tfinal$
$0.2 \cdot \cos(0.01t)$	$0.61 \cdot \sin(0.1t)$	0.15	50	0.0037	1/2.5	0	60

Table 3-4: Table summarizing the control, perturbation and Frenet-Serret inputs used to compare the fusion schemes with respect to the external perturbations.

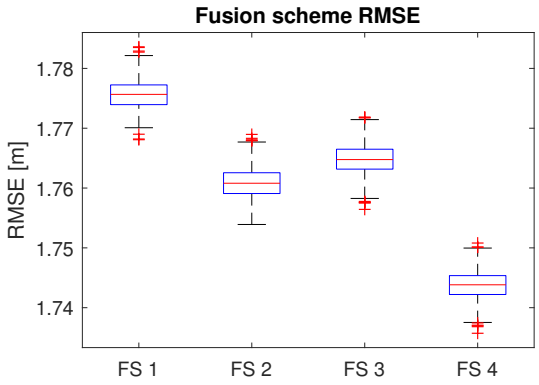


Figure 3-3: Graph displaying RMSE score distributions of the fusion schemes, using the time-invariant process noise matrix. The sensor inputs were corrupted by both noise and perturbations. The boxplots are assembled from $K = 1000$ RMSE values.

In Figure 3-3 the distributions of the RMSE scores are displayed. The figure shows that fusion scheme 4 obtains the best estimator performance compared to the other fusion schemes. Fusion scheme 4 is thus henceforth used to estimate the position.

Source for error build-up comparison

The effect of the sources for error build up on the position estimate is evaluated individually by corrupting the noise free sensor inputs, using the perturbation parameters displayed in Table 3-5. In order to evaluate the simulated error build-up scores to the error build-up scores using real data, the simulated ground truth trajectory is a straight line, with $\alpha(t) = 0$ and $\kappa = 0$.

$v_{nom}(t)$	$\alpha(t)$	$SlipMagMax$	$SlipTrans$	$Drift$	κ	τ	$Tfinal$
0.2	0	0.15	50	0.0037	0	0	60

Table 3-5: Table summarizing the control, perturbation and Frenet-Serret inputs used to evaluate error build-up of the EKF with respect to each perturbation individually.

The error build-up scores between the estimated position and the simulated ground truth trajectory are summarized in Table 3-6. The table shows that wheel slip is the main contributor to the overall error build-up score of the position estimator.

	None	Noise	Wheel slip	Heading drift	All
Error build-up	0.19%	0.99%	**114.47**%	6.18%	118.22%

Table 3-6: Table summarizing the error build-up scores between the estimated positions, obtained by the EKF in conjunction with fusion scheme 4 and a time-invariant process noise matrix, and simulated ground truth trajectory with the sensor inputs corrupted by noise, wheel slip and heading drift.

Source for error-build up validation

A way that the simulated error build-up values methods can be validated, without the use of additional resources, is by comparing the EKF estimated position (or travel distance) to that of trajectory with a known distance. The distance of such a ground truth trajectory can be determined by using the FLS images to detect and trace the weld lines on the ship hull surface, as shown in Figure 3-4, and by linking them to the associated sensor input.

Validation data The data that is selected to validate the performance of the velocity correction methods is only suitable if

- there exists FLS imagery that can be associated to the data,
- it is associated to a situation where the robot is traveling in a straight line such that it can be compared against the measured distance between the weld lines detected using the FLS and so that it can be compared against the simulated error build-up scores.

Using these restrictions, five different data sets summarized in Table 3-7 are obtained, originating from three different cleaning operations.

Data set	Ship	Date	Run time
1	Mineral China	10^{th} of September 2017	13:02:06.8-13:04:21.5
2	Mineral China	12^{th} of September 2017	17:49:00.8-17:49:52.6
3	OOCL	1^{st} of August 2017	18:53:03.9-18:54:13.7
4	HS Tosca	7^{th} of July 2017	00:02:14.6-00:02:56.2
5	HS Tosca	7^{th} of July 2017	00:56:43.0-00:58:29.2

Table 3-7: Table summarizing the real data sets used to evaluate the error build-up of the position estimators.

Only a limited amount of ground truth data sets are available since only a small portion of the FLS imagery is stored due to a limited amount of storage space.

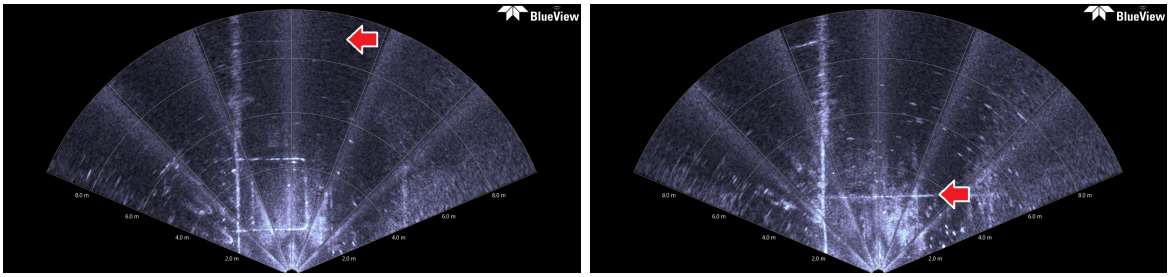


Figure 3-4: Images of robot being at a distance of 8.7 $[m]$ and 3.0 $[m]$ from the weld line. Images originate from operation on the HS Tosca on the 7^{th} of July 2017, 00:02:14.6-00:02:56.2.

True error build-up The EKF estimated travel distances and true travel distances, using the sensor data associated to the validation data sets listed above, are summarized in Table 3-8. The respective error build-up values, as defined in Equation (1-1), are summarized in Table 3-9. Tables 3-6 and 3-9 show that the simulated error build-up scores (118.22%) overestimate the maximum true error build-up (57.14%), which is probably mainly due an overestimate in the wheel slip frequency of occurrence. In order to have the simulated error build-up scores closer to the scores obtained using real data, the longitudinal wheel slip occurrence is scaled by a factor of $57.14/118.22 = 0.4833$ from once every 10 seconds to once every 20 seconds, yielding the simulated error build-up scores summarized in Table 3-10.

Data set $\rightarrow$	1	2	3	4	5
True	4.4 $[m]$	3.5 $[m]$	6.5 $[m]$	5.7 $[m]$	11.9 $[m]$
Estimated	6.8 $[m]$	5.5 $[m]$	8.7 $[m]$	6.5 $[m]$	11.6 $[m]$

Table 3-8: EKF predicted travel distances compared to the true travel distances, determined using the FLS.

Data set $\rightarrow$	1	2	3	4	5	Mean
Error build-up	54.55%	57.14%	33.85%	14.04%	2.52%	32.42%

Table 3-9: Error build-up scores between the EKF estimated and true position using real data.

	None	Noise	Wheel slip	Heading drift	All
Error build-up	0.19%	0.99%	**46.02%**	6.19%	54.72%

Table 3-10: Table summarizing the corrected simulated error build-up scores between the estimated positions, obtained by the EKF in conjunction with fusion scheme 4 and a time-invariant process noise matrix, and simulated ground truth trajectory with the sensor inputs corrupted by noise, wheel slip and heading drift. The wheel sleep occurrence has been decreased from once every 10 seconds to once every 20 seconds, to better match the error build-up scores obtained using real data.

3-2-4 Overview

The objective of this section was to *verify the workings of the EKF, evaluate the performance of the fusion schemes and time-invariant process noise matrix, and determine the effect that the perturbations have on the position estimate.*

Using an open loop noise and perturbation free simulation it was shown that all fusion schemes correctly estimate the position in three dimensions. The selection of the fusion scheme and process noise model design method is summarized in Table 3-11.

Position estimator selection	**EKF**	~~PF~~	~~UKF~~
Fusion scheme selection	~~FS1~~	~~FS2~~	~~FS3~~	**FS4**
Process noise matrix design selection	**Time-invariant** Q	~~Time-varying Q~~

Table 3-11: Table summarizing position estimator, fusion scheme and process noise matrix design selection.

The individual simulation of the perturbations revealed that wheel slip induces the most error build-up, followed by heading drift, inducing an error build-up of 46.02% and 6.19%, respectively.

3-3 Conclusion

The objective of this chapter was to *design a 3D position estimator* and *evaluate the effect of the simulated perturbations on the position estimator* designed in Chapter 2. This knowledge can then be exploited to develop a strategy for reducing the error build-up in the position estimate.

The review of literature on non-linear position estimators, possible fusion schemes and process noise matrix design was used to establish working principles for estimating the position of the robot. Simulations were used to compare the performance of the working principles - which revealed that an EKF fusing the acceleration data with model predicted rotational velocity data, using a time-invariant process noise matrix - obtains the lowest RMSE values with respect to the simulated ground truth.

The effects of the sensor noise and perturbation were studied by applying the noise and perturbation models designed in Chapter 3 on the EKF and comparing the error build-up scores between the estimated positions and a simulated ground truth trajectory.

Results The simulations revealed that the time-varying process noise matrix did not add significant performance to either of the four fusion schemes. Furthermore, it was revealed that fusion scheme 4 yields the lowest RMSE score when faced with sensor noise and external perturbations. Fusion scheme 4 in combination with a time-invariant process noise matrix is henceforth used to estimate the position of the robot. The simulations also revealed that the main culprit of the error build-up in the position estimate is wheel slip, which induces an error build-up up to 46.02%. The heading drift of 0.0037 $[rad/s]$ induced an error build-up of 6.18%. The noise and linear approximation of the EKF induced an error build-up of 0.99% and 0.19%, respectively. Using real data error build-up scores between 2.52% and 57.14%, with a mean of 32.42%, were obtained.

Limitations The conclusion that the main source for error build-up is wheel slip, was based upon simulated results, which has inherent limitations due to sensor noise and perturbation modeling assumptions. The claim does bear some merit as many researchers point at longitudinal wheel slip as a main source for error in mobile robots localization using odometry [64, 65]. The ground truth used to validate the error build-up scores was obtained by visual inspection of the FLS imagery, which induces errors of approximately 10 $[cm]$. This chapter concludes the analysis phase of this thesis, of which it was the objective to *gain understanding into the effect that perturbations and sensor noise have on the error build-up in the position estimate*, such that this knowledge can be used to efficiently reduce the error build-up. Based on the results obtained in this chapter it is proposed that in order to reduce the error build-up to within 2.05% the wheel slip and heading drift must both be accounted for.

Chapter 4

Wheel slip Detection

In the previous chapter it was shown using simulations that wheel slip is the main source for error build-up in the position estimate. The error build-up in the position estimate violates the requirement of a maximum error build-up of 2.05% and puts additional workload on the operator, as was discussed in Section 1-3. In order to reduce error build-up in the position estimate, which in turn reduces workload on the operator and improves the cleaning trajectory presented to the client, the wheel slip must be detected and accounted for.

In most cases, the detection of wheel slip is achieved by training a *classifier* on real labeled[1] sensor data. The trained classifier is then used to detect wheel slip by labeling incoming data in slip (1) and non-slip (0) classes. As was shown in Section 2-2, the slip labeling process involves the cross-referencing of camera footage to sensor data, which is very labour intensive making real labeled sensor data limited in numbers but also restricted to a certain type of ship hull condition. Due to these restrictions of real labeled data, it is expected that the classifier label prediction accuracy will degrade when faced with varying ship hull conditions.

The objective of this Chapter is thus to *design a wheel slip detection algorithm that attains a high as possible slip label prediction accuracy and retains this accuracy when faced with varying ship hull conditions.*

The design of the wheel slip detector is achieved by firstly reviewing literature on wheel slip detection in comparable robots to establish working principles. The working principles designed in Section 4-1 are simulated in Section 4-2, using the sensor and perturbation models designed in Chapter 2, so that they can be tested under varying ship hull conditions. Finally, the working principles are validated in Section 4-3 to verify the workings and performance of the principle using real sensor data.

[1] Data is labeled when data points have been assigned a label denoting whether they belong to an instance where the robot is slipping (1) or where it is not (0).

4-1 Wheel slip detection working principle

Wheel slip detection is a common topic in the design of localization systems for mobile robots that use odometry to estimate the position of the robot. This is especially true for robots that operate on loose, uneven or slippery surfaces [66, 67]. As was mentioned before, the detection of slip is often achieved using a *classifier* that labels incoming sensor data as belonging to the slip class (1) or non-slip class (0), denoted $delta_t$. The incoming sensor data is often arranged in *feature vectors*, denoted $\boldsymbol{f}_t$. These feature vectors are obtained by transforming the input data using some function such that a higher *prediction accuracy* can be attained by the classification algorithm [68]. In order to account for the imbalance between positive and negative labels the balanced accuracy metric is used as the prediction accuracy metric, denoted [69]

$$Accuracy = \frac{(TP/P + TN/N)}{2}, \tag{4-1}$$

were TP is the amount of true positive predictions, TN the amount of true negative predictions, and P and N the total amount of positive negative class labels respectively. In order to reduce the error build-up as much as possible, the slip detection algorithm must attain a high as possible slip label prediction accuracy.

Wheel slip detector accuracy requirement In Chapter 3, Table 3-9 it was estimated that 46.02% of the error build-up was due to wheel slip and 0.99% due to combined non-linearity and noise. In order to reduce the error build-up to 2.05%, assuming the error build-up due to heading drift can be completely eliminated and the true velocity can be perfectly estimated, $\frac{46.02-(2.05-0.99)}{46.02} = 97.7\%$ prediction accuracy is required at minimum. Typically, slip detection algorithms used in mobile robots attain a label prediction accuracy of between 94% and 98% [11, 25, 70], where an accuracy of 98% is only achieved when the researchers have access to multiple wheel encoders [11]. Due to the limited amount of training data and access to only a single wheel encoder, it is unlikely that an accuracy of 97.70% can be achieved. Instead, emphasis is put maximizing the label prediction accuracy given that limited training data is available and the ship hull surface conditions are varying.

Supervised versus unsupervised An important distinction in types of classification methods is the amount of supervision required to train the classifier. In general, supervised classification methods attain higher label prediction accuracy than unsupervised classification methods. However, the former requires labeled data to train the classifier whereas the latter does not. The manual labeling of sensor data is very time consuming as was discussed in Section 2-2, limiting the amount of labeled sensor data available to training the classifier. Furthermore, it was also established in Section 2-2 that the slip characteristics may vary significantly depending upon the ship hull surface conditions. If the limited classifier training set is not general enough to encompass all the varying ship hull conditions, a unsupervised method may be preferred since they can be trained on a larger, more general, unlabeled data set and have better adaptive capabilities.

Because of this trade-off, in this section three classification methods, ranging from supervised to unsupervised, as displayed in Figure 4-1 are proposed in this section.

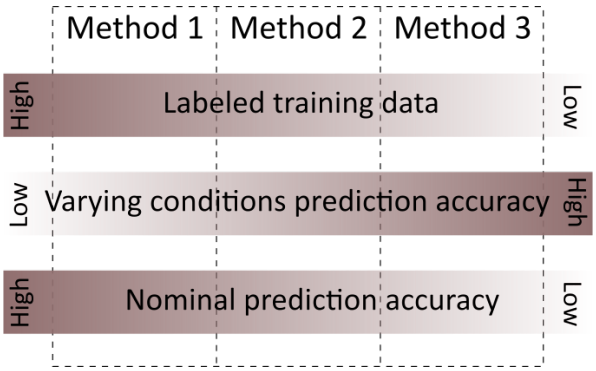


Figure 4-1: Schematic overview of expected prediction accuracy of classification methods in descending order of supervision under unvarying and varying conditions. For instance, Method 1 is a supervised classification algorithm, thus requiring labeled data to train the algorithm. Since labeled data is scarce and therefore limited to one type of ship hull condition, it is expected that the supervised classifier prediction accuracy degrades when faced with varying ship hull conditions. Under nominal conditions, meaning that the ship hull conditions associated to the incoming data are roughly the same to that of the training data, it is expected that the supervised method will attain the highest prediction accuracy.

The objective of this section is to *design features that can be used for classification and to propose three classifiers, ranging from supervised to unsupervised*, such that the most appropriate classifier can be selected based on simulations, simulating the varying ship hull conditions, in Section 4-2.

Firstly, features that are used to detect slip are designed in Section 4-1-1 - and secondly, three classification methods are proposed in Section 4-1-2. At the end of the section an overview of the design choices is given.

4-1-1 Feature engineering

In order to get better classifier prediction accuracy, the sensor data must first be transformed into features. Features that are commonly used in wheel slip detection for mobile robots are described below. The features that are discussed have all been employed in slip detection for mobile robots, where a label prediction accuracy of 94% or higher was attained.

Model predicted velocity feature

The detection of wheel slip can be achieved by comparing the wheel encoder output to a model predicted wheel speed output, given the input to the motor or, in the case of the Fleet Cleaner robot, to the hydraulic valves. The model predicts the wheel speed under the conditions that the wheel is not slipping. If the wheel starts slipping, the model prediction of the wheel speed and the wheel encoder output will diverge. This information can in turn be used to detect slip. This feature was shown to be effective by Ojeda et al., [24] in their design of a slip detection algorithm for a mobile robot operating on a loose sandy surface.

A Single Input Single Output (SISO) ARX model is used to predict the true velocity under no slip conditions, using the signal to the hydraulic valve as an input. The ARX model is one of the most simple models used in system identification, making its implementation time efficient. The ARX model assumes a model of the form [71]

$$y_t + a_1 y_{t-1} + ... + a_{na} y_{t-na} = b_1 u_{t-1-nk} + ... + b_{nb} u_{t-nb-nk} + e_t, \tag{4-2}$$

where y_t is the output at time instance t, u_{t-1} is the input at time instance $t-1$, 'na' is the number of poles, 'nb' is the number of zeros [2], 'nk' is the delay before the input occurs in the output, and 'e_t' is a white noise disturbance. The parameters a_i and b_j are estimated using the LLS solution [21, p. 28]. The tuning of the ARX has three degrees of freedom, namely the amount of zeros 'nb', the amount of poles 'na' and the input delay 'nk'. The tuning of these parameters is discussed next.

ARX model tuning In order to reduce the chances of overfitting the ARX model to the tuning data set [72], a K-fold cross-validation on a data set with sample size $N = 1281$ in combination with a *grid search* is used to tune the parameters set (na, nb, nk). More advanced, faster methods exist for finding the optimal parameter set, however, the grid search is easy to implement and can yield the same classifier performance [73].

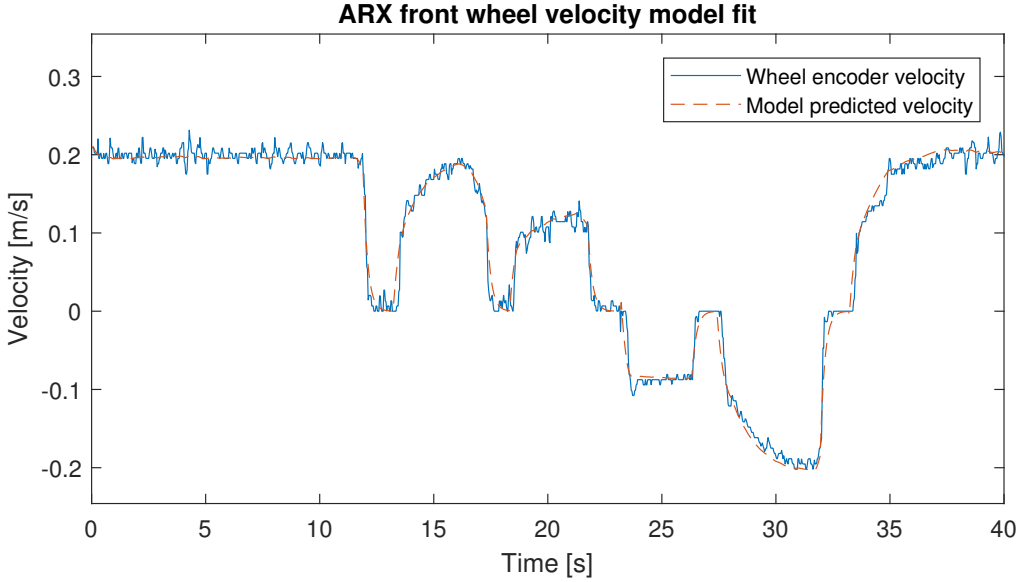


Figure 4-2: Graph of model fit between wheel encoder velocity and ARX model. The graph shows that despite a RMSE score of 0.0128 $[m/s]$ on the validation set, the model predicted velocity has little bias with respect to the wheel encoder velocity output.

The grid search is conducted by selecting different combinations for (na, nb, nk) and 4-fold cross-validating the ARX model on the tuning data set, as displayed in Figure A-8.1. This yields four model performance evaluations per parameter combination. The optimal parameter set (na, nb, nk) is found by evaluating the RMSE, depicted in Equation (3-13) for parameter sets $na = [1:10]$, $nb = [1:na-1]$ and $nk = [1:10]$[3].

Using a 4-fold cross-validation the optimal parameters were found to be $(na, nb, nk) = (2, 1, 2)$, yielding an average RMSE score of 0.0122 $[m/s]$ on the tuning data set and an RMSE score of 0.0128 $[m/s]$ on the validation data set. The tuned ARX model scored an average error of

[2] The amount poles and zeros are equivalent to the amount of terms in the denominator and numerator of a transfer function, respectively.

[3] In this thesis, the notation $[a:b]$ is used to describe a vector containing only integer values which increase from a to b with increments of 1. e.g. $[1:4] = [1 \quad 2 \quad 3 \quad 4]$

$-1.6355 \cdot 10^{-6}$ and $2.4679 \cdot 10^{-5}$ on the tuning and validation data set respectively, indicating there is little bias between the model prediction and the wheel encoder output. The model fit between the wheel encoder velocity and ARX model is displayed in Figure 4-2.

The discrete time TF model used to predict the velocity is

$$G(z) = \frac{2.278 \cdot 10^{-6} z^{-2}}{1 - 0.9467 \cdot z^{-1} + 0.0822 z^{-2}}, \quad Ts = 0.03125, \tag{4-3}$$

where $G(z)$ is a discrete-time transfer function between the hydraulic valve input $U(z)$ and the robot velocity $V(z)$, denoted v_{model}, z is the lag operator - and Ts is the sampling time.

The feature that gauges the discrepancy between the model predicted velocity and the wheel encoder velocity is denoted

$$f_1(k) = |v_{odo}(k) - v_{model}(k)|. \tag{4-4}$$

Model mismatch Changing environmental parameters, like the current velocity of the water and roll resistance of the front wheel, cause a model mismatch between the predicted velocity and the wheel encoder output. It is expected that this mismatch will degrade the performance of a classification algorithm that is using f_1 as a feature. Such a model mismatch is displayed in Figure 4-3, where the I/O model of Equation (4-3) is used to predict the front wheel velocity using a data set from a different cleaning operation. From the figure it can be seen at $T = 326.3$ $[s]$ a steady state model mismatch occurs of $100\% \cdot ((0.11/0.092) - 1) = 19.56\%$.

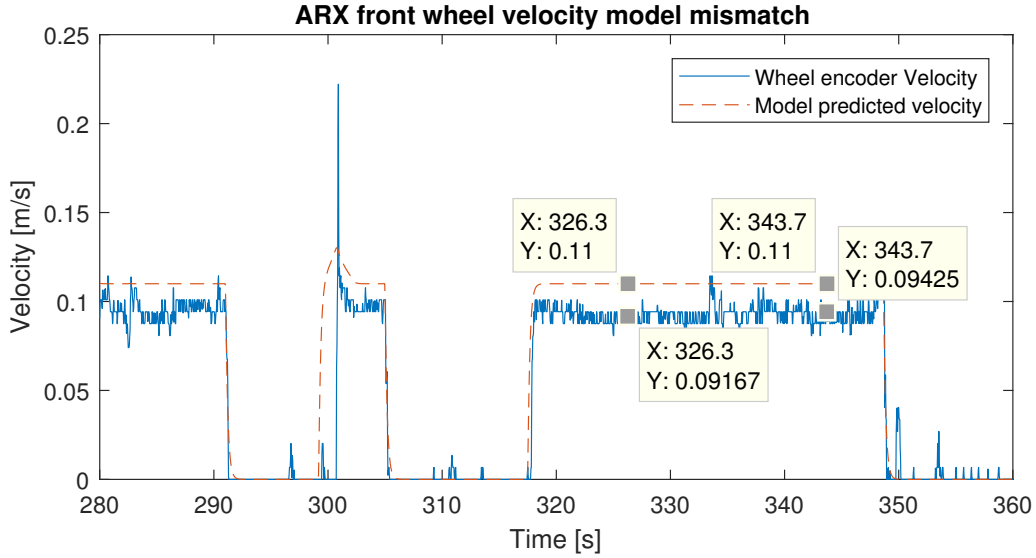


Figure 4-3: Graph displaying mismatch between the wheel encoder velocity output and the model predicted velocity. At $T = 325$ a steady state mismatch of 20% is observed.

The model mismatch between the model predicted velocity and wheel encoder velocity is modeled such that it can be used in Section 4-2 to evaluate its effect on the classifier prediction accuracy. The model mismatch is modeled with

$$v_{model}(k) = v_{nom}(k) + MM \cdot v_{nom}(k), \tag{4-5}$$

where $\tilde{v}_{nom}(k)$ is the nominal velocity without longitudinal wheel slip perturbation and $MM = [-20 \quad 20]\%$ is the parameter specifying the amount of model mismatch. The parameter is added to the user specified modeling parameters under 'external perturbation parameters' in Table 2-6.

Model predicted rotational velocity feature

If the wheel encoder output is perturbed by longitudinal wheel slip, using Equation (1-2) it can be seen that

$$\begin{aligned} \omega^z_{true} = v_{nom} \cdot \frac{\sin(\alpha)}{l} \neq \omega^z_{model} = v_{odo} \cdot \frac{\sin(\alpha)}{l}, \\ \forall \alpha, v_{nom}, v_{odo} \quad s.t \quad \alpha, (v_{nom} - v_{odo}) \neq 0 \end{aligned} \tag{4-6}$$

where ω^z_{true} and ω^z_{model} are the true and model predicted rotational velocity around the robot fixed frame $z-$axis, respectively. Since ω^z_{true} is gauged by the IMU gyroscope, these quantities are often used to detect slip [66, 25, 74]. The feature that gauges the difference between the model predicted and gyroscope rotational velocity is denoted

$$f_2(k) = |\omega^z_{model}(k) - \omega^z_{gyro}(k)|. \tag{4-7}$$

Variance of sensor inputs feature

The variance of the sensors signals is often used as a feature to detect robot immobilization [11, 70, 75]. The variance is calculated over a sliding window with

$$Feat^{var}_x(k) = \frac{1}{N} \sum_{i=k-N}^{k} (x(i) - \mu)^2, \tag{4-8}$$

where μ is the sliding window average and N the size of the sliding window. In order to get the best label prediction accuracy, the parameter N needs to be tuned. The sliding window variance is applied to the wheel encoder velocity output and the gyroscope rotational velocity output, yielding the features

$$f_3(k) = \frac{1}{N} \sum_{i=k-N}^{k} \left(v_{odo}(i) - \frac{1}{N} \sum_{j=k-N}^{k} v_{odo}(j) \right)^2, \tag{4-9}$$

and

$$f_4(k) = \frac{1}{N} \sum_{i=k-N}^{k} \left(\omega^z_{gyro}(i) - \frac{1}{N} \sum_{j=k-N}^{k} \omega^z_{gyro}(j) \right)^2, \tag{4-10}$$

A tuning method to find an optimal window length 'N' is discussed next.

Feature parameter tuning A simple method to check the 'goodness' of a feature is by evaluating its correlation to the class labels using the Pearson Correlation Coefficient (PCC) [68], defined as

$$PCC = \frac{cov(\boldsymbol{f}_i, Y)}{\sqrt{var(\boldsymbol{f}_i) \cdot var(Y)}}, \tag{4-11}$$

where Y are the true class labels of the data points and $\boldsymbol{f}_i$ the feature vectors. The PCC is a measure of correlation between a feature and the hand labeled data. The PCC takes values in the range of $[-1 \quad 1]$, where PCC> 0 indicates a positive correlation, PCC< 0 a negative correlation and $PCC = 0$ indicates zero correlation. The more correlated a feature is to the truth labels, the better it can be used by a classification algorithm to classify the data [68]. The PCC method is used in Section 4-2 to evaluate the optimal value of the window length 'N' for the features in Equation (4-8).

To summarize, the *standardized* feature vector [4] that is used to detect slip is denoted

$$\boldsymbol{\zeta} = \begin{bmatrix} std(\boldsymbol{f}_1)^{-1} & 0 & 0 & 0 \\ 0 & std(\boldsymbol{f}_2)^{-1} & 0 & 0 \\ 0 & 0 & std(\boldsymbol{f}_3)^{-1} & 0 \\ 0 & 0 & 0 & std(\boldsymbol{f}_4)^{-1} \end{bmatrix} \cdot \left(\begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \boldsymbol{f}_3 \\ \boldsymbol{f}_4 \end{bmatrix} - \begin{bmatrix} mean(\boldsymbol{f}_1) \\ mean(\boldsymbol{f}_2) \\ mean(\boldsymbol{f}_3) \\ mean(\boldsymbol{f}_4) \end{bmatrix} \right). \tag{4-12}$$

Feature standardization[5] the input features is important when features with different units are used and has shown to increase classifier performance [76].

4-1-2 Supervised machine learning

Supervised machine learning algorithms can be broadly divided into logic based algorithms such as decision trees [77] and rule based classifiers [25], perceptron based classifiers such as the Feed Forward Neural Network (FF-NN) [78], statistical learning algorithms such as the naive Bayes classifier [79], instance based learning such as the kNN [80] and State Vector Machine (SVM) learning [81]. In general, the SVM and FF-ANN attain the highest prediction accuracy when dealing with multi-dimensional continuous features, given that enough training data is available, whereas decision trees and rule based classifiers perform better with discrete or categorical features [82]. The naive Bayes approach is insensitive to overfitting on the training data as it is a high-bias learning algorithm [82]. The flip side is that it has lower prediction accuracy than high-variance learning algorithms like the SVM and the FF-ANN. The kNN is a simplistic classification method with very short training time. It is easy to implement since it only requires the tuning of a single parameter k [80]. It generally has slightly higher prediction accuracy than the naive Bayes method and slightly lower prediction accuracy than the FF-ANN and the SVM [82]. Among these possibilities the FF-ANN and the SVM are expected to have the highest label prediction accuracy.

In a study, conducted by Gonzalez et al., 2016, comparing several classification methods against each other in a pursuit to detect wheel slip of a Mars rover it is concluded that the SVM has a slightly better label prediction accuracy than the FF-ANN, 95.5% versus 94.0% when the classifiers are used to detect wheel slip [70]. Another advantage of the SVM over

[4] Meaning that the feature is zero mean and unit variance.
[5] In literature it is also called 'normalization'.

FF-NN is that the optimization required is a convex optimization, mitigating the possibility of converging to a local minimum [82]. For the reasons discussed the SVM is used as the supervised classification method.

SVM algorithm SVMs were introduced by V.Vapnik in 1992 as a novel method for classifying data [81]. The SVM segregates data by finding a hyperplane that leads to the maximum separation between clusters in the input space. This classification line is denoted

$$y(i) = sign(\boldsymbol{w}^T \Phi(\boldsymbol{\zeta}(i)) + b), \tag{4-13}$$

where $y(i)$ is the class label that takes the values $(1, -1)$, $\boldsymbol{w}$ a weighting factor, $\boldsymbol{\zeta}$ an input feature vector, $sign()$ is a function that maps $\mathbb{R}^+ \rightarrow 1$ and $\mathbb{R}^- \rightarrow -1$ – and $\Phi()$ is a non-linear kernel, discussed in the next section. The class labels -1 are mapped to 0 using

$$delta(i) = max(0, y(i)). \tag{4-14}$$

The optimization that needs to be conducted to find the weight $\boldsymbol{w}^T$ and b is

$$\begin{aligned} &\min \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \\ s.t \quad &y(i)(\boldsymbol{w}^T\Phi(\boldsymbol{\zeta}(i)) + b) \geq 1 \quad i = 1, ..., n \end{aligned} \tag{4-15}$$

where $y(i)$ is the hand labeled output, and n the number of samples.

If the data points in the input space are not linearly separable the linear classification SVM is easily extended to a nonlinear classifier, by mapping the input data into a high-dimensional feature space. By choosing an adequate mapping, the data points can become (mostly) linearly separable [83]. The performance of the SVM thus largely depends upon the mapping, or *kernel*, that is used. There are, however, no theories concerning how to choose a good kernel in a data dependent-way [83], making the selection an empirical exercise.

SVM kernel selection The three most popular kernels used for the SVM are the Radial Basis Function (RBF) (or Gaussian kernel), defined as [84]

$$K(\boldsymbol{\zeta}, \boldsymbol{\zeta}(i)) = \exp\left\{-\gamma||\boldsymbol{\zeta}_j - \boldsymbol{\zeta}(i)||^2\right\}, \tag{4-16}$$

where $\boldsymbol{\zeta}_j$ are the support vectors [84] and γ the reciprocal of the width of the Gaussian. There is also a polynomial kernel, denoted

$$K(\boldsymbol{\zeta}, \boldsymbol{\zeta}(i)) = (\boldsymbol{\zeta}_j^T\boldsymbol{\zeta}(i) + a)^d, \tag{4-17}$$

and the linear kernel, which is just a first order polynomial kernel. The Gaussian kernel is often preferred above the polynomial kernel because it has less *hyper parameters* that need tuning [73] and it usually outperforms the polynomial kernel in both accuracy and convergence time [85]. There are some situations in which a Gaussian kernel is not suitable, particularly when the feature vector is of high dimension. A linear kernel is then preferred [73].

If the data set is separable, the largest margin is found by minimizing Equation (4-15). If the data set is not separable, even after using a kernel transformation, some data points will

fall within the margin calculated by the SVM, called *margin errors*. To allow for such margin errors, a slack variable ξ_i is introduced into the inequality constraints to allow a data point to be in the margin ($0 \leq \xi_i \leq 1$) or to be misclassified ($\xi_i > 1$). The objective function of Equation (4-15) is extended to

$$\begin{aligned} & \min \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}\xi(i), \\ s.t \quad & y(i)(\boldsymbol{w}^T\Phi(\boldsymbol{\zeta}(i)) + b) \geq 1 - \xi(i), \\ & \xi(i) \geq 0, \end{aligned} \tag{4-18}$$

where the tune-able parameter C puts a penalty on the amount of data points that are misclassified or within the margin. Since the classification problem at hand is of low dimension and the Gaussian kernel has been shown to produce good results, it is used as the SVM kernel. To optimize the prediction accuracy of a SVM using a Gaussian kernel, two parameters need to be tuned: the reciprocal of the width of the Gaussian γ and the soft margin parameter C.

SVM hyper parameters The tuning of the Gaussian parameter γ greatly influences the accuracy and robustness of the SVM classifier. Choosing a small value for γ will generally lead to a smooth decision boundary, which counters overfitting. Choosing a high value for γ allows more flexibility in the decision boundary which can obtain higher prediction accuracy. However, a high γ value may induce overfitting on the training data [85]. As discussed before the margin parameter C penalizes the margin error data points. Choosing a high C will lead to less margin errors, however this may also reduce the margin seperating the rest of the data points. If C is reduced, allowing for some margin error, the optimization may find a hyperplane with a bigger margin separating the rest of the data points [85]. The tuning method that is used to tune the SVM hyper parameters is discussed at the end of this section.

Threshold classification

Thresholding is a classification technique often used to detect wheel slip, the difference between thresholding techniques often being the way in which the threshold is determined. A binary threshold classifier classifies incoming features using the rule

$$delta(i) = \begin{cases} 1 & \text{if } \sum_{j=1}^{N}(\boldsymbol{\zeta}_j(i) \geq T_j) > 1, \\ 0 & \text{if } \sum_{j=1}^{N}(\boldsymbol{\zeta}_j(i) \geq T_j) = 0, \end{cases}$$

where $\boldsymbol{\zeta}_j$ is the j^{th} entry of the feature vector and T_j the to that dimensions corresponding threshold, both at data sample i. A popular way of determining the threshold for the maximum torque that can be exerted on the surface, is by using the Mohr-Coulomb tire-soil interaction model [24, 25, 86]. The threshold is calculated by first calculating the maximum torque τ_{max} that can be exerted on the soil without slipping,

$$\tau_{max} = c + \sigma_{max}\tan(\phi), \tag{4-19}$$

where c is the cohesion of the soil, ϕ the internal friction angle of the soil and σ_{max} the normal component of the stress region at the wheel-terrain interface. The maximum allowable torque

is then related to the input current to the wheel motors, which is linearly proportional to the torque exerted by the motor [24]. However, it is pointed out by C. Ward el al., that determining the parameters in the Mohr-Coulomb model requires accurate measurements and the technique is terrain-specific [87], they like others [67, 66, 87] instead opt for tuning the threshold empirically, which mitigates the issue of parameter estimation. The empirical tuning the threshold requires labeled data, since the prediction accuracy needs to be determined for each specific threshold, making it a supervised method. The threshold approach does however retain some adaptability compared to the previously discussed SVM, as the thresholds may be tuned online by the operator if it is observed that they have bad label prediction accuracy. The threshold method requires the tuning of the four thresholds (T_1, T_2, T_3, T_4). The tuning method that is employed to tune the threshold is discussed at the end of this section.

4-1-3 Unsupervised machine learning

Since the training of the unsupervised classifier is not restricted to the use of labeled data, the training data set is of large scale, requiring a classification method that can cope with large scale data. Furthermore, the dimension of the data is low, since only four features are being used. An added requirement to the selection of the unsupervised classification method is that it must also posses good adaptive properties.

The K-means algorithm is suitable for large scale data of low dimension [88] and has a low computational complexity. Furthermore, it has been widely applied as an adaptive clustering method [89, 90, 91] and has even been used for detecting wheel slip [70]. Among all the clustering techniques it is one of the most simplest and easily implemented algorithms which made it a popular classification method for researchers and have made it a base for more complex variants [80].

K-means algorithm

The K-means algorithm partitions an n-dimensional data set in K clusters such that some distance norm between the empirical mean of each cluster and the points in the cluster is minimized. The incoming data is assigned to a cluster by minimizing

$$\min_j \quad d(\boldsymbol{\zeta}(i), \boldsymbol{\mu}_j), \tag{4-20}$$

where $\boldsymbol{\mu}_j$ is the j^{th} cluster center of the K clusters, $d(\cdot)$ is some distance norm and $\boldsymbol{\zeta}(i)$ is the input feature vector point – and assigning the input vector label j that minimizes the function.

The cluster centers $\boldsymbol{\mu}_j$ are trained by initializing the cluster centers at an arbitrary or specified initial points $\boldsymbol{\mu}_j(0)$ and updating the cluster centers by assigning the training data $\boldsymbol{\zeta}(i)$ to the cluster center j that minimizes

$$\min_j \quad d(\boldsymbol{\zeta}(i), \boldsymbol{\mu}_j) \tag{4-21}$$

and updating the cluster centers using

$$\boldsymbol{\mu}_j = \frac{1}{N_j} \sum_{i=1}^{N_j} \boldsymbol{\zeta}^j(i), \tag{4-22}$$

where N_j is the number of samples belonging to the same cluster and $\boldsymbol{\zeta}^j(i)$ the i^{th} sample belonging to cluster j. Using the K-means algorithm as described above would only partition the data. In order to assign slip labels $\{1, 0\}$ to the data, the cluster j that minimizes

$$\min_j \quad d(\mathbf{0}, \boldsymbol{\mu}_j) \tag{4-23}$$

is assigned as the non-slip cluster. Note that this is the cluster where the difference between the model predicted velocities v_{model} and ω_{model} - and the velocities provided by the sensors v_{odo} and ω_{gyro} is the smallest. This cluster j is denoted '$NoSlip$' and the other clusters '$Slip$'. Incoming data points $\boldsymbol{\zeta}(i)$ can now be classified using

$$delta(i) = \begin{cases} 1 & \text{if } \min_j \quad d(\boldsymbol{\zeta}(i), \boldsymbol{\mu}_j) \neq d(\boldsymbol{\zeta}(i), \boldsymbol{\mu}_{NoSlip}), \\ 0 & \text{if } \min_j \quad d(\boldsymbol{\zeta}(i), \boldsymbol{\mu}_j) = d(\boldsymbol{\zeta}(i), \boldsymbol{\mu}_{NoSlip}), \end{cases}$$

K-means distance function Typically the Euclidean distance metric

$$d_j = ||\boldsymbol{\zeta}(i) - \boldsymbol{\mu}_j|| \tag{4-24}$$

is used to determine what cluster center is nearest to the input feature $\boldsymbol{\zeta}(i)$. As a result, the K-means clustering algorithm finds ball shaped clusters in the data [80]. It is unclear to what extent the features have the same order of magnitude, so to account for the mismatch between the cluster shape and the distribution of the feature points, the standardized Euclidean distance is used, defined as [88]

$$d_j = \frac{||\boldsymbol{\zeta}(i) - \boldsymbol{\mu}_j||}{std_j}, \tag{4-25}$$

where std_j is the standard deviation of cluster j. Note that this is equivalent to standardizing the features.

The K-means algorithm converges to a local minimum depending on its cluster initializations [80]. To overcome this problem the algorithm can be started multiple times and the the best solution, in terms of the sum of squared errors, can be chosen as the best partition [92]. Furthermore, the K-means algorithm only has a single parameter that needs to be tuned, namely the amount of clusters 'K'. The selection of the amount of clusters is simply achieved by evaluating the label prediction accuracy for different values of 'K'. The method employed to find the optimal amount of clusters 'K' is discussed next.

Classifier parameter tuning method

The proposed classification methods require the tuning of method specific parameters. Table 4-1 summarizes the method specific tuning parameters.

	SVM	Threshold	K-means
Parameters	(C, γ)	K	(T_1, T_2, T_3, T_4)

Table 4-1: Table summarizing the classification specific tuning parameters.

The grid search method used to tune the ARX model parameters is also used to tune the classifier parameters summarized in Table 4-1. The optimization of the parameter sets is achieved by a K-fold cross-validation on the tuning data set for different combinations of tuning parameters [73]. The parameter set that yields the highest label prediction accuracy, averaged over the K folds, is then selected as the optimal parameter set. The classifier prediction accuracy is validated on a separate validation data set. After the moving window sizes N have been determined the feature order is randomized to reduce the chances of overfitting. In Figure A-9.1 a schematic overview is given, explaining the steps used to tune classifier parameter sets and to validate the classifier label prediction accuracy.

4-1-4 Overview

The objective of this section was to *design features used for classification and to propose three classifiers, ranging from supervised to unsupervised*, such that the most appropriate classifier can be selected with respect to the limited amount of labeled data and the varying ship hull surface conditions. In Table 4-2 the selected features and classifiers that will be evaluated in the next section are summarized.

Features	
Velocity feature	$f_1(k) = \lvert v_{odo}(k) - v_{model}(k) \rvert$
Rotational velocity feature	$f_2(k) = \lvert \omega^z_{model}(k) - \omega^z_{gyro}(k) \rvert$
Velocity variance	$f_3(k) = \frac{1}{N} \sum_{i=k-N}^{k} \left(v_{odo}(i) - \frac{1}{N} \sum_{j=k-N}^{k} v_{odo}(j) \right)^2$
Rotational velocity variance	$f_4(k) = \frac{1}{N} \sum_{i=k-N}^{k} \left(\omega^z_{gyro}(i) - \frac{1}{N} \sum_{j=k-N}^{k} \omega^z_{gyro}(j) \right)^2$
Classifiers	
Supervised	SVM
	Threshold
Unsupervised	K-Means

Table 4-2: Table summarizing the selected features and the classifiers to be evaluated in Section 4-2.

Limitations A set of four features was selected based upon their respective successful implementations in literature. The selection of the features leaves out of consideration different combinations of the four features and other possible features.

4-2 Slip detection simulation

Features that can be used to detect wheel slip and three classification methods, ranging from supervised to unsupervised, were established in the previous section. In Section 2-2 it was established that the maximum longitudinal wheel slip magnitude will vary depending upon the ship hull surface conditions and in Section 4-1 it was established that there will be a mismatch between the model predicted velocity and the wheel encoder velocity output, depending on surface conditions. These variations affect the features designed in the previous section and thus affect the classifier prediction accuracy.

It is expected that the SVM will achieve the highest prediction accuracy under *nominal conditions* meaning that the ship hull conditions corresponding to the input data remain similar to the data used to train the classifiers. If the input sensor data characteristics depart too far from the training data characteristics, the K-means classifier may achieve a higher prediction accuracy assuming it can be made adaptive so that it can account for the variations in ship hull conditions.

The objective is thus to *determine what classification method is best suited to detect wheel slip with respect to the limited amount of labeled data and the varying ship hull conditions.*

Firstly, the nominal performance of the classifiers will be compared. The nominal evaluation of the classification methods will show the respective label prediction accuracy of the classification methods in the ideal case of unchanging ship hull characteristics. The nominal evaluation of the classifiers thus serves as a best case scenario of their respective label prediction accuracy.

Secondly, the performance of the classification methods will be evaluated under variable slip characteristics and model parameters, simulating variable ship hull conditions.

4-2-1 Nominal performance

The nominal performance of the classifiers is evaluated to establish what the performance of the classifiers is in the best case scenario of unchanging perturbation parameters, mimicking unchanging ship hull conditions. Furthermore, the perturbation parameters are set such that the classifier prediction accuracy is high, relative the other parameter possible settings specified in Table 4-3. The prediction accuracy of the classifiers under nominal conditions can thus be regarded as the best case scenario.

Simulation data

In order to pose a best case scenario, the modeling parameters are set such that the features become easily separable by the classifiers. For instance, maximizing the slip magnitude parameter values also maximizes the segregation between the slip and non-slip clusters in standardized *feature space*[6], displayed in Figure 4-4, increasing classifier prediction accuracy. The parameter settings for the nominal evaluation of the classifiers are summarized in Table 4-3.

$v_{nom}(t)$	$\alpha(t)$	$SlipMagMax$	$SlipTrans$	$Drift$	MM	κ	τ	$Tfinal$
$0.2 \cdot \cos(0.01t)$	$0.61 \cdot \sin(0.1t)$	0.15	50	0	0	0	0	30

Table 4-3: Table summarizing the perturbation parameter settings for the slip detector classifier training data and nominal performance evaluation.

[6]Feature space is the co-domain of the sensor data after it has been transformed using the feature functions.

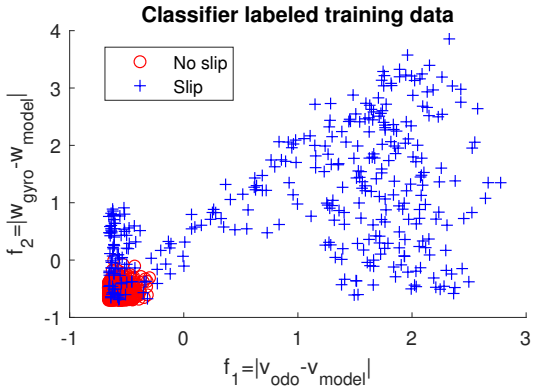


Figure 4-4: Scatter plot of labeled feature vector data points of training data set. The red circles represent the feature data points when the wheel is not slipping and the blue plus markers represent the feature data points when the wheel is slipping.

Feature parameter tuning

Using the labeled training data set, displayed in Figure 4-4, the optimal value for 'N' in Equation (4-8) is determined by evaluating the PCC between the features $\boldsymbol{f}_3$ and $\boldsymbol{f}_4$ – and the slip labels for different 'N'. Evaluating the PCC for $N = 2:100$, the highest PCC values are obtained at $N = 38$ and $N = 16$ for $\boldsymbol{f}_3$ and $\boldsymbol{f}_4$, respectively, as displayed in Figure 4-5.

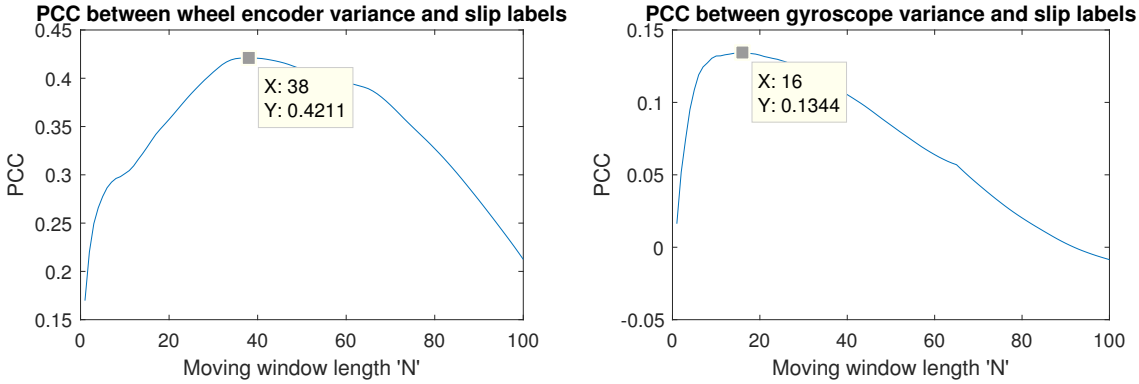


Figure 4-5: Graphs displaying the evaluation of the PCC between $\boldsymbol{f}_3$, $\boldsymbol{f}_4$ and the slip labels for an increasing moving window length 'N'.

Classifier parameter tuning

Using the previously discussed grid search method the grids specified in Table 4-4 are searched, obtaining the classifier parameters summarized in Table 4-5. In order to find the optimal parameters for the SVM and the threshold classifier, three consecutive grid searches are conducted, using a higher resolution grid each step. The parameter values that correspond to the maximum prediction accuracy are used as the midpoint of the search grid of the next iteration. As was previously discussed, the cluster centers of the K-means classifier also depend upon the initial clusters μ_0. To prevent overfitting, the K-means training is multistarted 100 times for each K.

	SVM	Threshold	K-means
Parameters	(C, γ)	(T_1, T_2, T_3, T_4)	K
Grid 1	$(10^{[-5:5]}, 10^{[-5:5]})$	$(-2:6, -2:6, -2:6, -2:6)$	$(2:20)$
Grid 2	$(50:150, 5:1.5)$	$(0.5:1.5, -0.5:0.5, 3.5:4.5, 1.5:2.5)$	-
Grid 3	$(85:95, 4.5:5.5)$	$(-0.25:0.35, -0.10:0.00, 3.75:3.85, 1.75:1.85)$	-

Table 4-4: Table summarizing search grids used to determine optimal classifier parameters.

	SVM	Threshold	K-means
Parameters	(C, γ)	(T_1, T_2, T_3, T_4)	K
Values	$(92, 5.0)$	$(-0.34, -0.09, 3.77, 1.77)$	3

Table 4-5: Table summarizing optimal classifier parameters obtained using the grid search ranges specified in Table 4-4.

Nominal performance evaluation

The nominal performance of the classification methods is evaluated by generating 1000 data sets, using the modeling parameters displayed in Table 4-3. The prediction accuracy, calculated using Equation (4-1), of the classifiers is evaluated on the same 1000 data sets. This ensures that the respective classifier performance does not depend on differences between data sets.

The results are summarized in Table 4-6. From the table it is clear that the SVM and the threshold method have the highest label prediction accuracy, followed by the K-means classifier.

Classification method	Minimum	1^{st} quartile	Median	3^{rd} quartile	Maximum
SVM	**95.63**%	**95.94**%	**96.05**%	**96.15**%	**96.36**%
Threshold	95.11%	95.53%	95.73%	95.84%	96.25%
K-means	88.03%	88.24%	88.35%	88.45%	88.76%

Table 4-6: Table summarizing the nominal label prediction accuracy of the classification methods.

It can be concluded that under nominal conditions the SVM performs best. Under nominal conditions all the classification methods except the K-means classifier fall within the range of 94% to 98% prediction accuracy that is normally seen in wheel slip detection, as was discussed at the beginning of this chapter. It is assumed that the K-means algorithm can retain its nominal performance under varying conditions if made adaptable and thus the 88.35% prediction accuracy serves as a lower bound to compare the SVM and Threshold algorithm against.

Limitations The data that was used posed a best case scenario for the classification algorithms, so the label prediction accuracy that was obtained in this section is probably at the high end of what is realistic. Furthermore, the labeled data that was used to tune the classifiers was labeled perfectly, which also results in an overestimate the the classifier prediction accuracy since this is not the case in a realistic setting. This perfect labeling of data is not

realistic when real sensor data is used, since the data is labeled by visually cross-referencing camera footage to data sequences.

The next section evaluates the effect of varying ship hull surface conditions on the prediction accuracy of the classification methods.

4-2-2 Performance under variable characteristics

The environmental conditions within which the Fleet Cleaner robot operates are variable, which in turn affect the features used by the classifiers to detect wheel slip. In this section the prediction accuracy of the classifiers under variable environmental conditions is tested by evaluating the prediction accuracy for varying values of slip magnitude and mismatch between the wheel encoder velocity and model predicted velocity. The same classification models as in the previous section are used in this section. There is thus a mismatch between the training and validation data, emulating the scenario where the ship hull conditions corresponding to the sensor data on which the classifier was trained severely deviate from the ship hull conditions corresponding to the input sensor data. Firstly, the results of the classifier prediction accuracy will be shown. At the end of the section the results, and their limitations, are discussed.

Performance under varying model mismatch and maximum longitudinal wheel slip

The prediction accuracy of the classifiers under model mismatch between the model predicted velocity and the wheel encoder velocity is simulated by setting all the perturbation parameters at their nominal values, summarized in Table 4-3 and letting the model mismatch parameter 'MM'increase from -30% to 30% in $K = 61$ steps. Similarly, the prediction accuracy of the classifiers under varying longitudinal wheel slip magnitude is simulated by letting the maximum longitudinal wheel slip parameter '$SlipMagMax$' increase from 0 to 0.2 in $K = 21$ steps. Each step is iterated 1000 times to simulate different noise realizations.

The label prediction accuracy under varying model mismatch and maximum longitudinal wheel slip are displayed in Figures 4-6 and 4-7, respectively. In theoretical label prediction accuracy of an adaptive K-means classifier is also plotted in the figures.

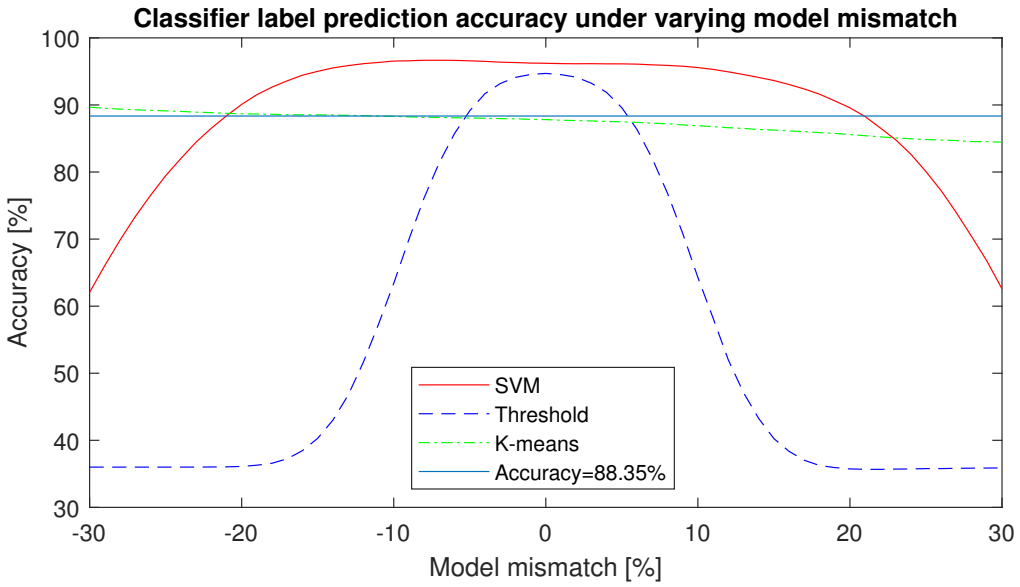


Figure 4-6: Graph displaying the label prediction accuracy of the classification methods under a model mismatch ranging from -30% to 30%.

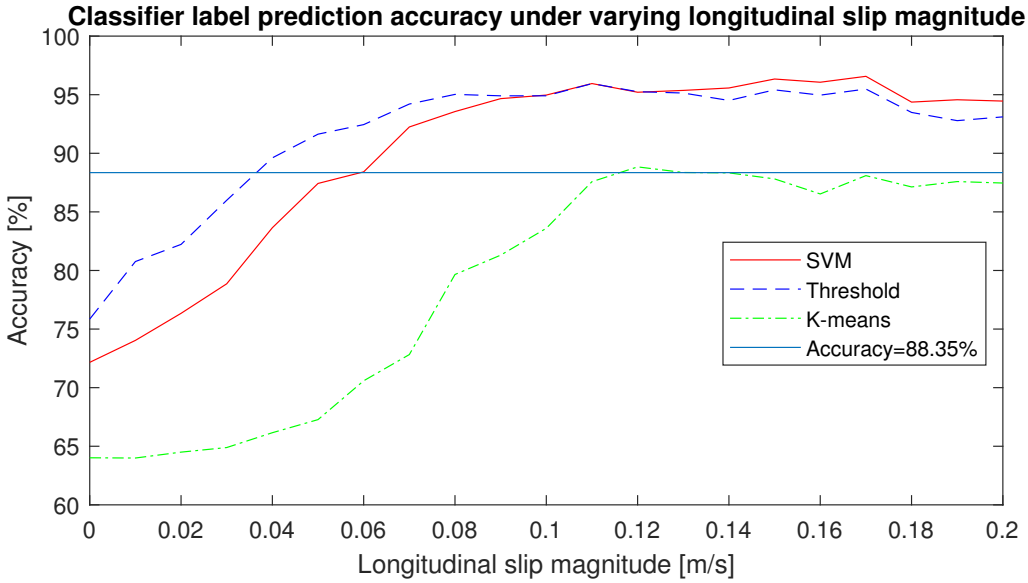


Figure 4-7: Graph displaying the label prediction accuracy of the classification methods under a longitudinal slip magnitude ranging from 0 to 0.15 $[m/s]$.

Discussion

SVM vs. K-means As was previously discussed, under nominal conditions the SVM has a far better prediction accuracy than that of the K-means classifier. It was argued that the K-means may be preferred if the prediction accuracy of the SVM under variable conditions drops below that of the K-means classifier under nominal conditions, assuming the K-means

classifier can be made adaptive such that it retains its nominal prediction accuracy under all conditions. The prediction accuracy of the SVM only falls below that of the nominal prediction accuracy of the K-means classifier in the presence of a longitudinal slip magnitude lower than 0.06 $[m/s]$ and a model mismatch beyond 21%. The advantage of the K-means classifier over the SVM with respect to the model mismatch can be disregarded since a model mismatch of up to 20% is expected. Only in the range of 0 to 0.06 longitudinal slip magnitude improvements can be made by using an adaptive K-means classifier. Since the SVM performs considerably better in all other ranges, this does not justify the design of an adaptive K-means classifier.

SVM vs. Threshold Under nominal and varying conditions, except for varying longitudinal slip magnitude, the SVM has a better label prediction accuracy than the threshold classifier. It can however be argued that, like the K-means classifier, the threshold classifier can be adapted such that it can retain its nominal prediction accuracy. In the case of the threshold classifier this assertion is less credible than in the case of the K-means algorithm since it requires an operator to adjust the thresholds online. The amount of online adjustment depends upon how sensitive the prediction accuracy of the threshold classifier is with respect to the varying conditions. Unfortunately, the prediction accuracy is very sensitive to model mismatch. Any model mismatch beyond $\pm 5\%$ would require the operator to tune the thresholds, since a theoretical adaptive K-means classifier would be preferred beyond that point. The adapting of the thresholds puts additional work load on the operator, which is in direct violation with one of the objectives of this thesis, namely to reduce the workload on the operator.

Although the SVM is the best classifier compared to the other classifiers, it does not retain its nominal performance under varying conditions. The label prediction accuracy falls below its minimal nominal performance of 95.63% at an longitudinal slip magnitude of less than 0.1 $[m/s]$ and a model mismatch beyond -14% to 13%. It is therefore expected that an SVM trained on the limited real data set currently available will not suffice to accurately detect slip when faced with varying ship hull conditions.

On the basis of the simulated results it is thus concluded that the SVM classifier is best suited to detect wheel slip for the Fleet Cleaner robot. In the Section 4-3 the results are validated using real sensor data.

4-2-3 Overview

The objective of this section was to *determine what classification method is best suited to detect wheel slip with respect to the limited amount of labeled data and the varying ship hull conditions.*

The median label prediction accuracy of the classifiers under nominal conditions is summarized in Table 4-7. Furthermore, it is summarized if the classifiers can retain their nominal performance under varying conditions.

Limitations Although the simulated results indicate that the SVM is likely to have the best performance, the results also have some limitations. In evaluating the performance of the classifiers under varying conditions it was tacitly assumed that all variations are equally

	SVM	Threshold	K-means
Nominal label prediction accuracy	**96.05**%	95.73%	88.35%
Retain performance under varying ship hull conditions?	No	No	No

Table 4-7: Table summarizing the nominal median label prediction accuracy of the classifiers and if the classifiers can retain their nominal performance under varying ship hull conditions.

as likely. This was done because in the Chapter 2 the occurrence rate of the perturbation values was not studied. For instance, the performance with respect to model mismatch was evaluated in the range of -30% to 30% and based on the entire range it was concluded that the threshold classifier has the worst performance. However, it could be the case that a model mismatch of $\pm 5\%$ is far more likely than one of, for instance, 10%. In that case the performance of the threshold method is not as bad as was concluded now. Furthermore, the classifiers were trained on a perfectly labeled data set. This limits the evaluation of the classifier performance since under realistic circumstances, where the data is labeled by visually cross-referencing camera footage to data sequences, the data will not be perfectly labeled.

4-3 Slip detection validation

In the previous section it was determined that the SVM classifier has the best overall performance with respect to the varying ship hull conditions and a limited amount of labeled training data. However, the results were obtained using simulated data which has inherent limitations and so to validate the results, real sensor data is used to evaluate the classifiers.

The objective of this section is thus to *validate the simulated classifier performance results using real sensor data.*

Firstly, a data set is selected for evaluation and the parameter 'N' of the variance features $\boldsymbol{f}_3$ and $\boldsymbol{f}_4$ is tuned in Sections 4-3-1 and 4-3-2, respectively. After the features have been tuned, the classifier parameters are tuned in Section 4-3-3. Finally, the performance of the classifiers is validated in Section 4-3-4.

4-3-1 Validation data

The hand labeled data that is used to train and evaluate the classifiers originates from an operation on the HS TOSCA, on the 6^{th} of July, 2017. The data corresponds to the wheel encoder velocity output displayed in Figure 2-2. This data set is currently the only available labeled data set.

4-3-2 Feature tuning

Using the same technique as described in the previous section, the optimal moving window length 'N' for the features $\boldsymbol{f}_3$ and $\boldsymbol{f}_4$ are determined by evaluating the PCC between the features and the slip labels for a varying amount of 'N'. As is shown in Figure 4-8, the optimal values are found to be $N = 127$ and $N = 36$ for $\boldsymbol{f}_3$ and $\boldsymbol{f}_4$, respectively.

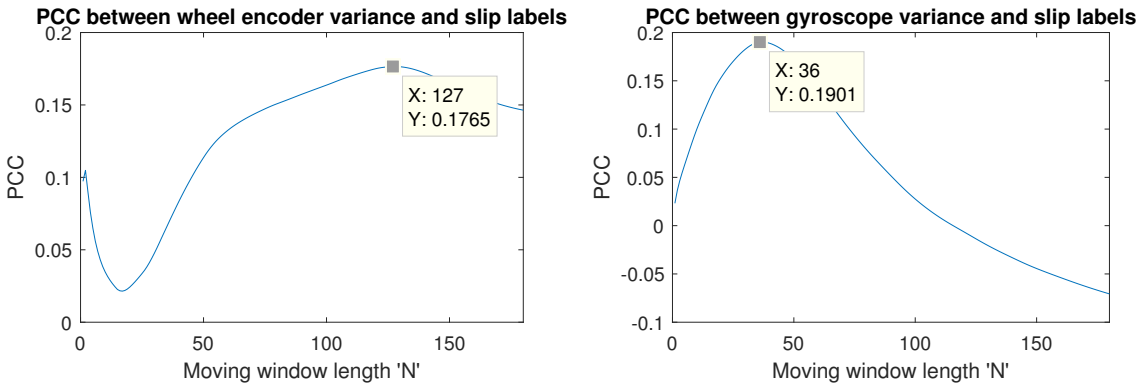


Figure 4-8: Graphs displaying the evaluation of the PCC between $\boldsymbol{f}_3$, $\boldsymbol{f}_4$ and the slip labels for an increasing moving window length 'N'.

4-3-3 Classifier parameter tuning

The parameters of the classifiers are tuned a grid search, 4-fold cross-validation method, as discussed at the end of Section 4-1. The grid ranges that are used to find the optimal parameters are summarized in Table 4-8 and the optimal classifier parameters are summarized in Table 4-9.

	SVM	Threshold	K-means
Grid 1	$[10^{-4:4}, 10^{-4:4}]$	$[-2:6, -2:6, -2:6, -2:6]$	$[2:20]$
Grid 2	$[5:15, 0.5:1.5]$	$[0.5:1.5, -0.5:0.5, 4.5:5.5, 4.5:5.5]$	-
Grid 3	-	$[0.85:0.95, -0.10:0.10, 4.50:4.60, 4.80:4.90]$	-

Table 4-8: Table summarizing search grids used to determine optimal classifier parameters.

	SVM	Threshold	K-means
Parameters	(C, γ)	(T_1, T_2, T_3, T_4)	K
Values	$(10, 0.5)$	$(0.93, -0.08, 4.54, 4.83)$	8

Table 4-9: Table summarizing optimal classifier parameters obtained using the grid search ranges specified in Table 4-8.

4-3-4 Classifier performance

The trained classifiers are evaluated on a separate validation data set that also originates from the data set displayed in Figure 2-2. The classifier label prediction accuracy scores summarized in Table 4-10 are obtained. As was expected from the simulated results, the SVM attains the highest prediction accuracy followed closely by the threshold classifier. In both cases the K-means classifier has the lowest prediction accuracy. Based on the simulated and validation results, the SVM is selected as the most suitable classifier to detect wheel slip for the Fleet Cleaner robot.

	SVM	Threshold	K-means
Label prediction accuracy	**96.85**%	94.06%	84.97%

Table 4-10: Table summarizing the classifier prediction accuracy on the validation data set.

Limitations Although the prediction accuracy of the SVM falls in the range specified at the beginning of the chapter, it cannot be assumed that this prediction accuracy will be attained in all situations. This is because the data that was used to train the classifiers originates from a single operation and therefore does not encompass all possible ship hull conditions. Furthermore, the data was labeled using camera, so it includes an unknown amount of falsely labeled data. This again reduces the reliability of the validation results.

4-4 Conclusion

The objective of this chapter was to *design a wheel slip detection algorithm that attains a high as possible slip label prediction accuracy and retains this accuracy when faced with varying ship hull conditions.* In order to satisfy the requirement of a maximum error build-up score of 2.05% a label prediction accuracy of 97.70% was required. Typically, such prediction accuracy is only attained by slip detectors with access to multiple wheel encoders and so more emphasis was put on designing a slip detector that maintains a high prediction accuracy under varying ship hull conditions. Three classification methods, in descending order of supervision, were evaluated in determining what the best method is for detecting slip, namely

- SVM,
- Threshold,
- K-means.

In order to evaluate the classifier performance with respect to varying ship hull conditions the classifiers were trained on a simulated data set emulating a real data set obtained from an operation with one type of ship hull condition - and tested on data sets emulating the entire range of expected ship hull conditions. The simulated results were validated using real sensor data.

Results and Conclusion The simulations results show that under nominal conditions the SVM performs best, achieving a label prediction accuracy of 96.05%. The simulated results were validated using real sensor data, which revealed that the SVM attains a prediction accuracy of 96.85%. When faced with varying longitudinal and model mismatch, the SVM has the highest prediction accuracy compared to the other classifiers except for conditions with a longitudinal slip magnitude beneath 0.1 $[m/s]$. Based on the simulated and validation results the SVM was selected as the best suited classifier to detect wheel slip.

The simulated results show that an SVM trained on limited data can only maintain its nominal performance for a small range of varying external perturbation parameters. So although the validation results yield a prediction accuracy of 96.85%, it cannot be expected that it will

attain this accuracy under all ship hull conditions, since the training data originates from a single ship with limited ship hull surface conditions.

The simulations also reveal that for low wheel slip magnitudes, the prediction accuracy of the SVM deteriorates. It is expected that this decrease in accuracy can not be omitted, due to the fact that it only has access to a single wheel encoder. If multiple wheel encoders had been available, the difference in wheel encoder output could have been used to detect slip even when the wheel slip magnitude is low.

Limitations The simulations only included variations in model mismatch, and slip magnitude. This limits the analysis of the robustness of the classifiers since other factors that may change with varying ship hull conditions were not included. Such factors may for instance be the PSD of the noise, the noise magnitude or other unforeseen factors.

The main drawback of the classifier validation procedure is that the training and tuning of the classifiers were based upon hand labeled data, originating from a single ship, that was obtained using a camera. This caused the amount of available data to be very limited, which is why it cannot be assumed that the SVM will have good performance in all conditions. Furthermore, the use of a camera to label the data is not very accurate so the SVM is trained on a set where an unknown amount of data was falsely labeled.

Recommendations It is recommended that two more wheel encoders are added to the robot such that a higher label prediction accuracy can be maintained in case of low wheel slip magnitude. Based on literature [11] it is anticipated that with the addition of two more wheel encoders a label prediction accuracy of 97.70% can be achieved.

In order to train the classifier such that it encompasses most of the possible surface conditions, a time efficient method for labeling the sensor data is required. One proposition is to first add two wheel encoders and to only actuate two of the three wheels. The third wheel will function as an absolute velocity measurement, which can be used to label data as slip (1) when a discrepancy between the third wheel and either of the two other wheels is detected.

Chapter 5

Velocity Correction

As was previously discussed, the main source for error build-up in the position estimate is wheel slip. This violates the requirement that the error build-up should stay within 2.05%. In this chapter a velocity correction algorithm is designed such that it can be used in conjunction with the previously designed EKF position estimator and SVM slip detector to improve the position estimate.

In Chapter 4 it was concluded that it is unlikely that an error build-up of 2.05% can be achieved, since the SVM slip detector cannot maintain its prediction accuracy over a wide range of varying ship hull conditions.

The objective of this chapter is thus to *design a velocity correction algorithm, that can be used in conjunction with the EKF and the SVM, that attains an as low as possible error build-up score.*

In Section 5-1, literature on velocity correction for mobile robots using odometry is reviewed and a design direction for the velocity estimator is chosen. In Sections 5-2 and 5-3, the velocity correction algorithm is tested on simulated and real data and compared against more common velocity correction methods.

5-1 Velocity correction working principle

The literature on possible ways to correct velocity is vast and so three system requirements are listed below to narrow down the search for a adequate working principle.

Velocity correction working principle requirements

1. The working principle must work in conjunction with the SVM slip detector and EKF position estimator.
2. Absolute data measurements using additional hardware is not needed to train the working principle.

The objective of this section is thus to *find a velocity correction method consistent with the system requirements that achieves an as low as possible error build-up score.*

Firstly, velocity correction methods used in literature are reviewed in Section 5-1-1. Based on the velocity correction methods used in literature and on their shortcomings a velocity correction method is proposed in Section 5-1-2. Finally, the proposed working principle is reflected upon with respect to the set objectives and its limitations are discussed.

5-1-1 Interactive multi model approach

The literature on velocity estimation and slip correction can broadly be divided into the Interacting Multiple Model (IMM) approaches and a centralized approaches. IMM's operate by alternating between two (or more) models that predict the position of the robot in non-slip and slip conditions [93, 74, 94], whereas centralized approaches [95, 96, 97] rely on a single model that is continuously adapted using some external reference to correct the velocity. Considering the above stated requirements, an IMM approach is best suited to estimate the true velocity and will be further considered.

The IMM estimates the true velocity by taking linear combinations of multiple prediction models i.e. one that estimates the states when the robot is not slipping and on that estimates the states when it is. The general workings of the conventional IMM are described in Figure 5-1. It is common that both prediction models use a Bayesian filter like the KF, EKF or PF to predict the states, in which case the IMM switches between the models or takes a weighted average of the models by comparing the likelihood of the predicted states given their respective measurement inputs. When applied to mobile robots that are prone to slip, it is common that the model that predicts the position of the robot in non-slip conditions uses the wheel encoder velocity as a measurement update for the states, whereas the slip model uses the IMU acceleration as a measurement update [94, 93, 74]. Researchers that have employed the IMM to estimate the position of a mobile robot on uneven or slippery surfaces typically report an accuracy improvement of 60.53% to 84.44% [94, 93, 74], depending on how uneven or slippery the surface is. The accuracy improvement is defined as

$$Accuracy - improvement = \frac{Acc_1 - Acc_2}{Acc_1} \cdot 100\%, \tag{5-1}$$

where Acc_i is the error between method i and the ground truth [94].

This range will be used to compare the velocity correction methods designed in this chapter against.

SVMIMM Although it is common practice to use the likelihood of the innovation step as a model selection method, J. Jung et al., [74] have shown that using an external classification method to detect slip, which in their case was an SVM, improves the performance of the IMM over the conventional method. Since an SVM has already been implemented, the SVMIMM proposed by J Jung et al., [74], may prove to be a cost effective velocity correction method. Their approach is to estimate the velocity by taking a weighted average between the wheel encoder output and the IMU velocity output, using the posterior likelihood of the class labels conditioned on the feature input as a weight, depicted in Appendix A-10-1. R. Sidharthan

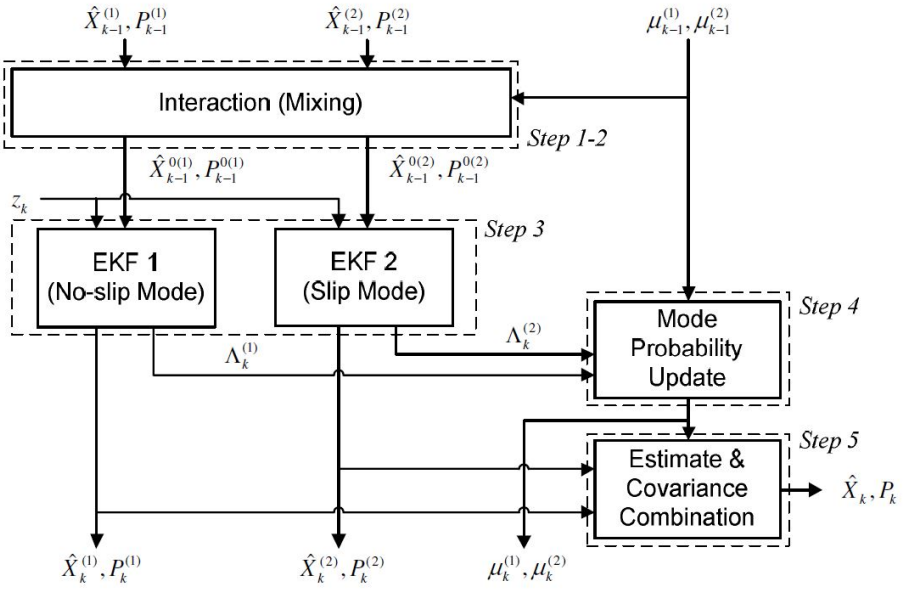


Figure 5-1: Schematic overview of IMM framework [74]. In **step 1-2** a linear combination of the states estimated by EKF 1 and EKF 2 in the previous time step is taken, using the mode probability $\boldsymbol{\mu}_{t-1}$ as a weight. In step 3 the linear combination is used updated by both EKF 1 and EKF 2, yielding the respective covariances $\Lambda_k^{(i)}$ and the state predictions $X_k^{(i)}$. The mode probability is updated in **step 4** using the covariance matrices of EKF 1 and EKF 2. Finally, in **step 5** the optimal state is calculated using a linear combination of the covariance matrices and the states.

et al., [94] observe that this slip detection method may be inadequate because wheel slip characteristics differ from one robot to another vary depending on the surface that the robot is on. This is in agreement with the simulations conducted in Chapter 4, which showed that the SVM slip detector is susceptible to varying ship hull conditions. If the modes are miscalculated, valuable acceleration data may not be incorporated in the velocity estimate. Another pitfall of the conventional IMM is the use of the IMU to estimate the velocity during slippage, which drifts due to acceleration error accumulation. In Chapter 2 it has been observed that prolonged slippage occurs during operation. In Figure 2-2 for instance, it can be seen that the front wheel is slipping for periods greater than 20 seconds. It is expected that because of this long slip duration, the use of IMU by the SVMIMM to estimate the velocity becomes inaccurate due to integration error accumulation.

Velocity correction method proposal In order to improve the velocity estimate during slippage it is proposed to make use of so called *weak constraints*. Weak constraints can be viewed as virtual measurements, used as measurement inputs to the position estimator, based on physical insight into the system [87]. The covariance associated to the constraints decides how 'weak' the constraint is and is used in the measurement noise matrix R. Opposed to this is the so called *strong constraints* where the covariance is zero. An example of a strong constraint is if the input to the hydraulic actuators of the wheels is zero, the input velocity to the EKF may be assumed to be zero as well. This mitigates the possibility of integrating wheel encoder noise while the robot is stationary, theoretically improving the position estimate.

The use of weak constraints instead of integrated IMU accelerations to estimate the velocity during slippage mitigates the chance of error accumulation and is therefore expected to yield more accurate velocity estimates over long periods of slippage. This expectation is tested by comparing the SVMIMM and the proposed *constrained IMM* in Sections 5-2 and 5-3. The constrained IMM is designed in the next sections.

5-1-2 Constrained IMM design

In Chapter 2, two types of slip were distinguished, so called *stationary slip* and *non-stationary slip* or *dynamic slip*. The former is a mode of slippage where the front wheel, or one of the rear wheels slips while the true velocity of the robot is zero. The latter is a mode of slippage where one of the wheels start slipping while the true velocity of the robot is not zero. It is important that the two are distinguished because both modes affect the true velocity very differently. Following the standard IMM framework, the two slip modes are made interactive using weights calculated by a mode probability predictor. In Figure 5-2 the design structure of the proposed constrained IMM is displayed. The mode probability predictor and dynamic slip model are designed in the next sections.

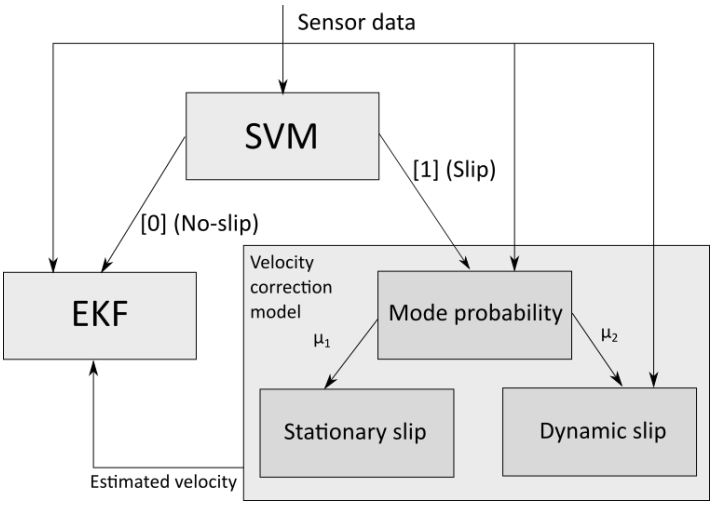


Figure 5-2: Schematic overview of proposed constrained IMM design. The incoming sensor data is classified by the previously designed SVM slip detector assigning the class labels [1] and [0] denoting slip and non-slip, respectively. If the wheel is not slipping, the EKF uses the wheel encoder velocity output to estimate the position. If it is slipping, the velocity is estimated by a velocity correction model and used as a weak constraint input to the EKF. The velocity correction model distinguishes between stationary and dynamic slip and takes a weighted average of the two, to estimate the velocity while slipping. The weights μ_1 and μ_2, weighting the slip models, are estimated with a so called mode probability predictor.

Mode probability predictor

As was previously discussed, in the normal IMM framework the mode probability is calculated using the covariances of the states. As was shown in [74], the use of an external classifier to calculate the mode probabilities can improve the position estimate. The use of a classifier requires the selection of features and the classifier itself, discussed next.

Feature selection As was discussed in Chapter 2, Figure 2-2, it can be observed that the wheel encoder velocity is greater when stationary slip is detected compared to dynamic slip. This can be understood by considering the following.

The rotary motion of a slipping wheel with mass m, friction coefficient c, angle $\phi(t)$ and torque input τ can be described by the second order ODE

$$\ddot{\phi}(t)I_m + \dot{\phi}(t)c = \tau(t), \quad I_m, c > 0. \tag{5-2}$$

Given the step input $\tau(t) = \tau_{step}$, the rotational velocity of the system will converge to its steady state after some transient such that

$$\dot{\phi}_{ss}(t)c = \tau_{step}(t). \tag{5-3}$$

From Equation (5-3) it can now be seen that if the friction coefficient c decreases, the steady state rotational velocity will increase. Thus, the lesser the grip, which is equivalent to a low friction coefficient, the higher the rotational velocity of the wheel will be, given a constant $\tau(t)$.

Using this insight that the difference between the model predicted velocity and the wheel encoder velocity is proportional to the friction coefficient, the feature $\boldsymbol{f}_1$, described by Equation (4-4), is used to differentiate between stationary and dynamic wheel slip. In Figure 5-3 it is shown that the difference between the model predicted velocity and wheel encoder output is greater during stationary slip compared to dynamic slip.

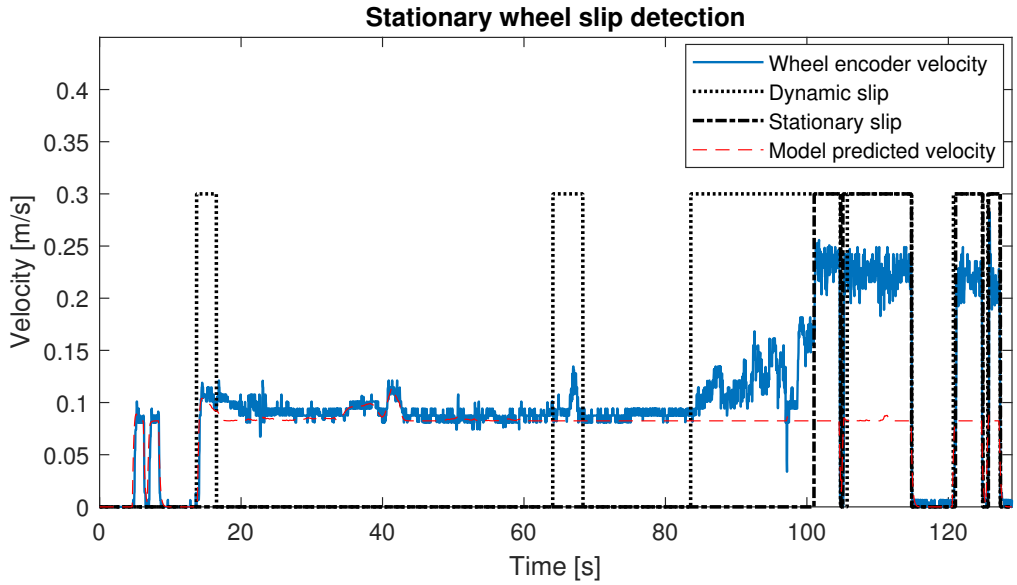


Figure 5-3: Graph displaying the wheel encoder velocity output, model predicted velocity and stationary slip labels. In the left image, stationary slip occurred due to the front wheel encountering ship hull fouling while in the right image it occured due to an encounter with a weld line.

Label probability calculation The selection of a classifier to calculate the mode probabilities is quite arbitrary since there is only one feature. The only requirement is that the classifier is also able to output the posterior probability $P(C_k|\boldsymbol{f}_1)$, where C_k is the class label. This ability has been developed for most classifiers. The SVM with a linear kernel is used to calculate the label probabilities since it produced good label prediction accuracy in the previous chapter

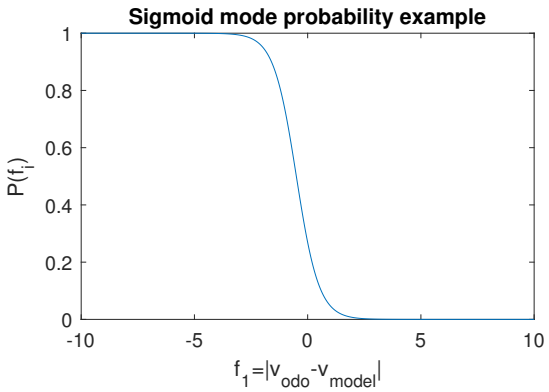


Figure 5-4: Graph of sigmoid with $A = 2$ and $B = 1$. The figure shows that the probability output input f_1 belonging to class C_k depends upon the value of f_1.

and it has already been employed by [74] for this application. Using the Matlab function *fitcsvm()* a sigmoid described by

$$p(f_i) = \frac{1}{(1 + \exp(Af_i + B))}, \tag{5-4}$$

where f_i is the feature value and A and B are estimation parameters, is fitted to the $P(C_k|\boldsymbol{f}_1)$ data points, as described in [98]. This yields a sigmoid, like the one shown in Figure 5-4, which outputs the likelihood of the label C_k, conditioned on the input feature value f_i. Using the estimated sigmoid, the weights of the modes are calculated with [74] $\mu_1 = p(f_i)$ and $\mu_2 = 1 - p(f_i)$. The corrected velocity and velocity covariance is then calculated with

$$\begin{aligned} \hat{v} &= \mu_1 \cdot Stationary - slip + \mu_2 \cdot Dynamic - slip, \\ \hat{\Sigma}_v &= \mu_1 \cdot \Sigma_{ss} + \mu_2 \cdot \Sigma_{ds}. \end{aligned} \tag{5-5}$$

Since it is assumed that the robot velocity is zero when stationary slip occurs, Equation (5-5) reduces to

$$\begin{aligned} \hat{v} &= \mu_2 \cdot Dynamic - slip, \\ \hat{\Sigma}_v &= \mu_2 \cdot \Sigma_{ds}. \end{aligned} \tag{5-6}$$

In the next section the dynamic slip model will be derived.

Dynamic slip velocity correction

The derivation of the true velocity of the robot during dynamic slip is complicated since there is no absolute velocity reference. However, under some circumstances it can be assumed that the wheel encoder velocity does provide an absolute velocity reference while the robot is slipping.

When the robot is moving backward during normal operation, it is common that one of the rear wheels will be exposed to the fouling, making them more likely to slip, as displayed in Figure 5-5.

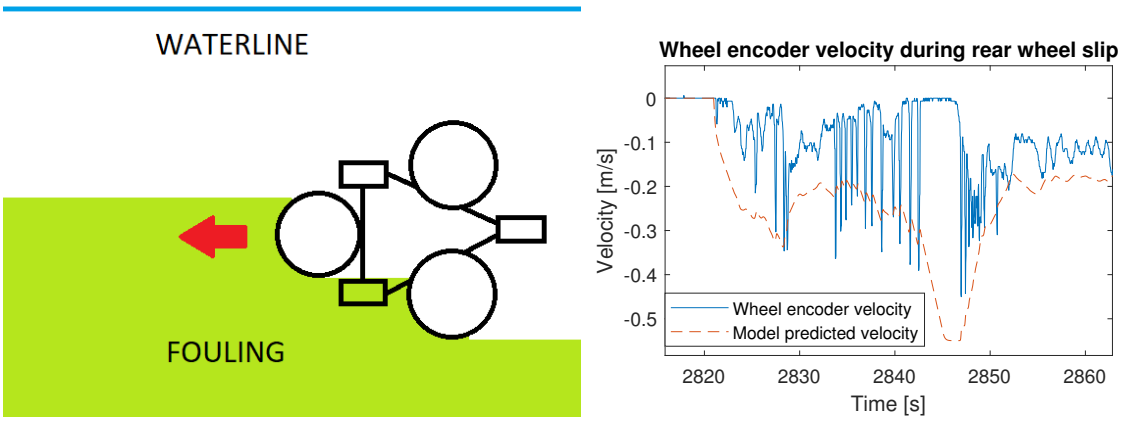


Figure 5-5: Illustration of circumstances in which rear wheel slip likely to occur (Left) and graph displaying data associated to rear wheel slip (Right). In the left image, the robot is moving backward, with its right rear wheel on a section that is fouled and thus likely to slip. Note that in the right image the magnitude of the model predicted output is higher than that of the wheel encoder output while moving backward, indicating rear wheel slippage.

In Figure 5-5 (Right) data associated to such occurrences is displayed.

Hydraulic wheel actuation To understand why the wheel encoder velocity is lower than the model predicted velocity when one of the rear wheels starts slipping, consider the following:

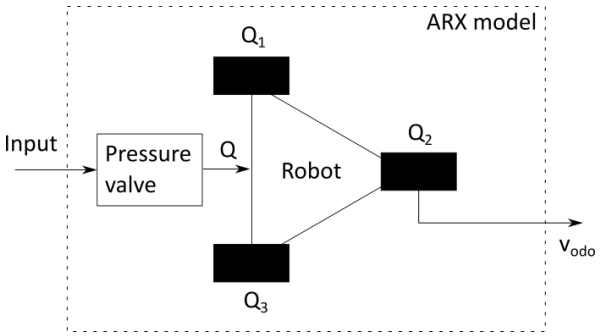


Figure 5-6: Illustration of distribution of hydraulic fluid flow over the robot wheels.

All three wheels share the same hydraulic pressure valve, as shown in Figure 5-6, such that the flow rate Q_i to each wheel satisfies

$$Q = Q_1 + Q_2 + Q_3, \quad Q_i \propto v_i, \quad Q \propto u \tag{5-7}$$

where v_i is the rotational velocity of each individual wheel, u is the input to the pressure valve and Q is the flow rate coming from the pressure valve. The ARX model of Equation (4-3) was identified under circumstances where the wheels were not slipping such that

$$Q_1 = Q_2 = Q_3 = \frac{Q}{3}, \tag{5-8}$$

$$u \cdot PV_{gain} = Q, \tag{5-9}$$

and

$$v_{model} = c \cdot \frac{Q}{3} = c \cdot Q_2 = v_{odo}, \tag{5-10}$$

where is c a constant.

However, if the right rear wheel starts slipping, such that

$$Q_3 > Q_1, Q_2, \quad Q_1 = Q_2, \tag{5-11}$$

it follows that

$$v_{model} = c \cdot \frac{Q}{3} > c \cdot Q_2 = v_{odo}, \tag{5-12}$$

assuming a constant pressure valve input u.

Under these conditions it can thus be assumed that one of the rear wheels is slipping and the front wheel is not, providing an absolute velocity measurement. The wheel encoder data associated to such conditions can thus be used to identify an model between the hydraulic valve input and the true velocity during wheel slippage.

Identification data In order to make the identification suitable for identifying a dynamic slip model, the data displayed in Figure 5-5 (Right) is modified such that it:

- Excludes data where the wheel encoder speed is zero or near to zero, since this behaviour is already captured by the stationary slip model,
- Only includes data points that have been labeled as 'slip' by the SVM slip detector,
- Only includes data points where the model predicted velocity magnitude is higher than that of the wheel encoder,
- Only includes data samples with a sample size higher than 100 samples[1].

In order to incorporate varying surface conditions into the identification data, data from operations on the HS Tosca, OOCL and Mineral China is used, yielding a total of 21 data samples with a sample size of 100 each. One data set is used as a validation set, and the other data sets are used to train the dynamic slip model using a 20-fold cross-validation, similar to the K-fold tuning method described in Section 4-2. In Appendix A-11.1 a schematic overview of the dynamic slip model identification is provided.

Dynamic slip model identification As was stated in the system requirements in Chapter 1 a maximum error build-up of 2.05% is allowed. If the robot moves in only one direction, the error between the ground truth position and the estimated position can be calculated with

$$d_{error}(T) = \left| \int_0^T v_{est}(t)dt - \int_0^T v_{true}(t)dt \right|, \tag{5-13}$$

[1]Since identification data is abundant, this reduces the amount of samples and also ensures that the slower dynamics are captured by the model.

and the error build-up with

$$error \quad build-up = 100 \cdot \left| \int_0^T v_{est}(t)dt - \int_0^T v_{true}(t)dt \right| / \int_0^T v_{true}(t)dt. \tag{5-14}$$

A Single Input Single Output (SISO) ARX model is used to predict the true velocity, using the signal to the hydraulic valve as an input. In order to find the numerator and denominator size nb and na and input delay nk, a 20-fold cross-validation is conducted on a search grid of $na = [2 : 20]$ by $[nb = 2 : na - 1]$ by $[nk = 0 : 10]$. Essentially, the same training method as in Section 4-1 to develop a model prediction for the velocity is employed. For each parameter combination, the RMSE score between the model prediction and the ground truth is evaluated 20-fold after which the average RMSE score over the 20-fold evaluation is calculated. Using this method, the parameter combination yielding the highest RMSE score is $(na, nb, nk) = (18, 7, 1)$. The model is validated on a separate validation set, yielding the model fit displayed in Figure 5-7. The error build-up calculated using Equation (5-14) is 0.15%, satisfying the requirement of 2.05%.

The average of the 20-fold RMSE scores between the model predicted velocity and the ground truth velocity, using the optimal parameters, is 0.0084 $[m/s]$, which equals the average standard deviation. The square of this value, $0.0084^2 = 7.0560 \cdot 10^{-5}$ which equals the variance, is used as the uncertainty of the weak constraint Σ_{ds}, depicted in Equation (5-5).

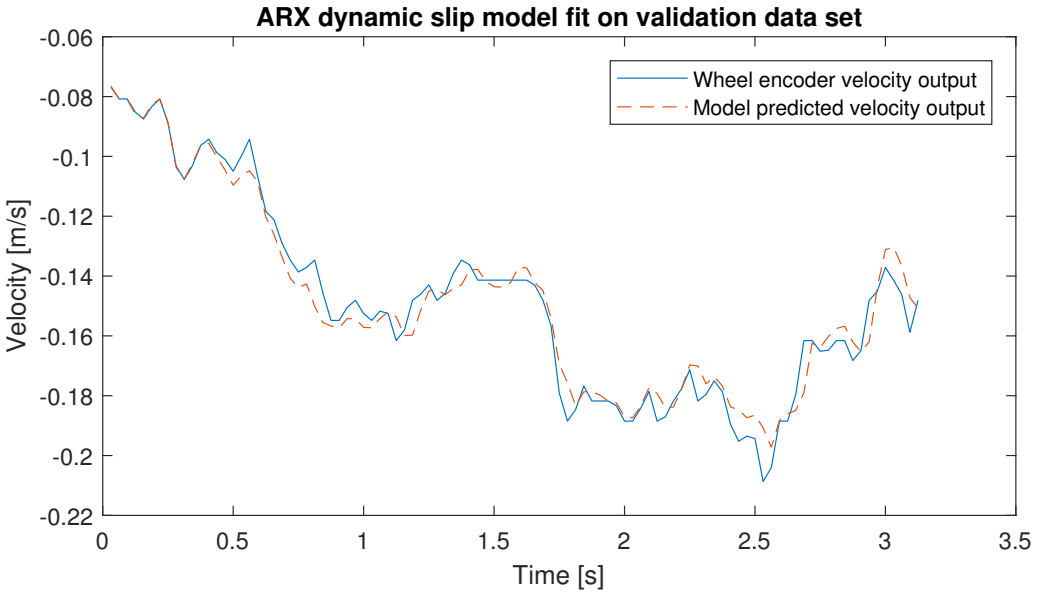


Figure 5-7: Graph of wheel encoder velocity and dynamic slip predicted velocity during slippage. The error build-up induced by the mismatch between the true wheel encoder velocity and the modeled wheel encoder velocity is 0.15%.

Now that the mode probability predictor and the dynamic slip model have been designed, the design of the constrained IMM algorithm is completed. The algorithm is depicted in Appendix A-10.

5-1-3 Overview

The objective of this section was to *find a velocity correction method consistent with the system requirements that achieves an as low as possible error build-up score.*

The constrained IMM is proposed as an alternative to the more conventional IMM for the reasons summarized in Table 5-1.

	Characteristics
Conventional IMM	Acceleration data error accumulation
	Performance affected by mode miscalculation
Constrained IMM	No use of noisy acceleration data
	Less affected by mode miscalculation

Table 5-1: Table summarizing advantages of the proposed constrained IMM over the conventional IMM.

Limitations Like the SVM designed in the previous chapter, the tuning of the mode predicting SVM relies on hand labeled data, which was labeled using camera footage. This induces the mislabeling of data, reducing the label prediction accuracy by an unknown amount.

The dynamic slip model used by the constrained IMM was based on the very strong assumption that the wheel encoder provides a ground truth velocity measurements under specific conditions. Because of this, the error build-up score attained using the dynamic slip model is probably overestimated, since the wheel encoder output under these specific conditions does not necessarily equal the 'true' ground truth velocity. A simple linear SISO ARX model was used to estimate the velocity during dynamic slip, which achieved the set requirement of an error build-up of 2.05%. However, due to the non-linear behaviour of the robot velocity during slippage it is expected that the model is only valid under limited conditions.

5-2 Velocity estimator simulation

In the previous section a constrained IMM was proposed as a working principle for estimating the velocity while slipping. The SVM that is used to calculate the mode probabilities makes use of the feature $\boldsymbol{f}_1$, depicted in Equation (4-4), which is susceptible to model mismatch and variations in longitudinal wheel slip magnitude, which are caused by varying ship hull conditions.

In order to evaluate the performance deterioration of the constrained IMM when faced with such variations in ship hull surface conditions, the constrained IMM is simulated using varying perturbation model parameters. The constrained IMM, depicted in Appendix A-10-2 is also compared against the SVMIMM, depicted in Appendix A-10-1, proposed in [74]. Block schemes of the simulation environments of the constrained IMM and the SVMIMM are provided in Appendix A-12.1 and A-13.1, respectively. The IMM designed by Jung et al., [74] is used as a comparison because it uses the IMU acceleration data to estimate the true velocity while slipping, which is a common approach in IMM design. It is expected that the use of IMU acceleration data to estimate the true velocity yields inaccurate results due to the long

duration of wheel slip. By comparing the constrained IMM to a conventional IMM it can thus be shown that the conventional method is unsuitable for the Fleet Cleaner robot.

The objective of this section is thus to *determine if the proposed constrained IMM is a significant improvement over the conventional IMM under varying ship hull surface conditions.*

In order to evaluate the velocity correction methods, firstly a ground truth velocity is modeled. Secondly, the SVM mode predictor is trained and thirdly, the correction methods will be evaluated under nominal and varying conditions. At the end of the section, the obtained results are discussed in an overview and the objective of the section is reflected upon.

5-2-1 Simulation data modeling

Similar to Chapter 4, the point of interest is to evaluate what the performance of the velocity corrector is under nominal conditions and if the performance is preserved when faced with varying external perturbation parameters that affect the velocity corrector. To reiterate, nominal conditions are conditions where the modeling parameters used to evaluate the velocity corrector are the same as the modeling parameters used to generate the training data, providing an upper bound for the expected performance.

SVM mode predictor training data The nominal perturbation parameters used to generate training data for the SVM mode predictor are summarized in Table 5-2.

$v_{nom}(t)$	$\alpha(t)$	$SlipMagMax$	$SlipTrans$	$Drift$	MM	κ	τ	$Tfinal$
0.2	0	0.15	$[0 \rightarrow 100]$	0	0	0	0	60

Table 5-2: Table summarizing the nominal perturbation parameter settings for training the mode predictor.

A combination of the data sets displayed in Figure 5-8 is used for the training data set to avoid the labeled data to be perfectly separable by the SVM mode predictor. A perfectly separable data set would yield a step function class label probability function $p(C_k|f_1(k))$ instead of a sigmoid, yielding mode weights $\mu_1, \mu_2 = \{0, 1\}$ instead of $[0, 1]$.

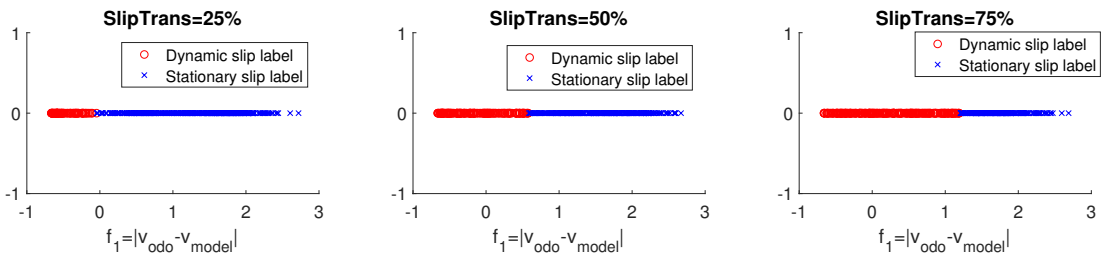


Figure 5-8: Scatter plot illustrating the effect of shifting slip transition parameter '$SlipTrans$'. The graphs show that as '$SlipTrans$' increases from 25% to 75%, the stationary and dynamic slip labels are shifted in feature space.

5-2-2 Simulated SVM mode predictor tuning

The simulated SVM slip mode predictor is tuned using a 4-fold cross-validation grid search, as was previously done for the SVM slip detector. Since a linear kernel is used, only the box constraint parameter 'C', discussed in Section 4-1-2, needs to be tuned. A one dimensional grid search, discussed in Section 4-1-3, is conducted for $C = [10^{-4} : 10^{4}]$ and $C = [0.5 : 1.5]$, consecutively. The grid search revealed an optimal prediction accuracy on the test data set of 96.57% with $C = 1$. On the validation set the tuned SVM scores a label prediction accuracy of 94.11%. The sigmoid parameters that are obtained by fitting a sigmoid, described by Equation (5-4), to the posterior distribution, as was discussed in Section 5-1-2, are $(A, B) = (-2.33, 0.27)$.

5-2-3 Velocity correction performance

As was discussed at the beginning of this section, the performance of the constrained IMM is expected to be susceptible to variations in ship hull conditions. To repeat, the changing ship hull conditions affect the model mismatch parameter 'MM' in Equation (4-5) and the longitudinal slip magnitude represented by '$SlipMagMax$' in Equation (2-3-2). These parameters in turn affect the feature $\boldsymbol{f}_1$, which is used by the velocity correction method to estimate the true velocity. The performance of the proposed constrained IMM is tested with respect to these changing parameters. In order to evaluate the effectiveness of the proposed constrained IMM, the velocity correction method is also compared against the SVMIMM of Jung et al., [74] and against the EKF without any velocity correction method.

Velocity correction working validation

Firstly, it is validated if the velocity correction methods are working in an optimal situation. The modeling parameters are set to the values summarized in Table 5-3. The noise is switched off and the true slip labels are used. Additionally, both velocity correction methods have a perfect dynamic slip and stationary slip model. In figure 5-9 the estimated velocities using the velocity correction methods are shown. The figure shows that when the sensor inputs are noise free and the data is perfectly labeled, both velocity correction methods work properly.

$v_{nom}(t)$	$\alpha(t)$	$SlipMagMax$	$SlipTrans$	$Drift$	MM	κ	τ	$Tfinal$
0.2	0	0.15	50	0	0	0	0	30

Table 5-3: Table summarizing modeling parameters used for velocity correction method validation.

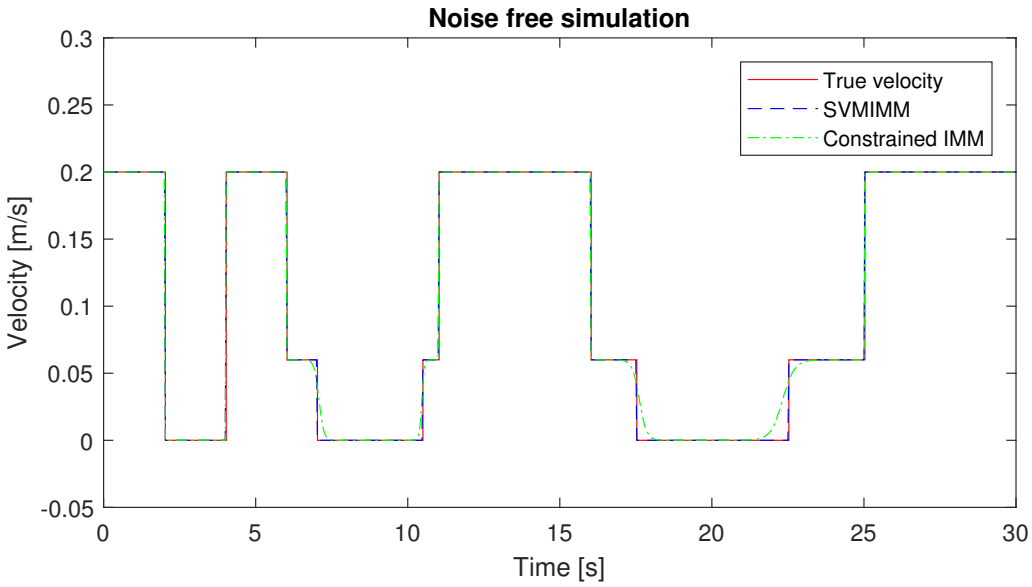


Figure 5-9: Graph illustrating the estimated velocities using the velocity correction methods.

Performance under varying model mismatch

The influence of a model mismatch between the model predicted velocity v_{model} and the wheel encoder velocity v_{odo} is simulated by fixing perturbation parameters at the values specified in Table 5-4, except the model mismatch parameter, which is increased from -30% to 30% in $K = 12$ steps.

$v_{nom}(t)$	$\alpha(t)$	$SlipMagMax$	$SlipTrans$	$Drift$	MM	κ	τ	$Tfinal$
0.2	0	0.15	50	0	0	0	0	60

Table 5-4: Table summarizing modeling parameters used for velocity method validation.

Each step is iterated $N = 1000$ times to simulate different noise realizations. The error build-up per meter traveled between the simulated ground truth and the EKF estimated position in conjunction with the two velocity correction methods is shown in Figure 5-10. The figure shows that the constrained IMM has the lowest error build-up except in the range beyond $\pm 20\%$. However, this is off less concern since a maximum model mismatch of $\pm 20\%$ is expected, as was discussed in Section 4-1. Overall, the constrained IMM has the best performance with respect to varying model mismatch and therefore attains the highest accuracy improvement over the EKF without velocity correction, as shown in Table 5-5.

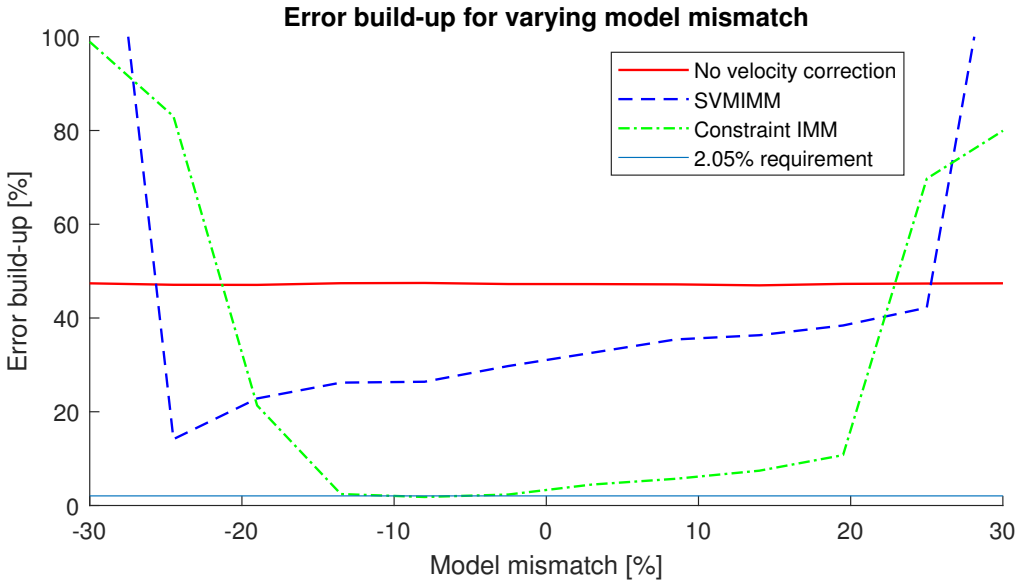


Figure 5-10: Graph of error build-up scores between simulated ground truth position and estimated position using different velocity correction methods under a varying model mismatch parameter 'MM'.

	No VC	SVMIMM	Cons. IMM
Nominal error build-up	47.24%	31.26%	**3.64%**
Average error build-up	47.24%	31.00%	**7.04%**
Accuracy improvement	-	34.38%	**85.10%**

Table 5-5: Table summarizing error build-up and the accuracy improvement of the EKF in conjunction velocity correction methods under varying model mismatch. The average error build-up scores are obtained by taking the average error build-up score in the range of $[-20 \quad 20]\%$.

Performance under varying longitudinal slip magnitude

The influence of a varying longitudinal slip magnitude '$SlipMagMax$' on the velocity correction method is evaluated by fixing the perturbation parameters at the values specified in Table 5-4 and letting '$SlipMagMax$' increase from 0 to 0.15 $[m/s]$ over a series of 11 steps, iterating each step $N = 1000$ times to simulate different noise realizations. In Figure 5-11 the error build-up scores between the simulated ground truth and the estimated position using the velocity correction methods are plotted for an increasing longitudinal slip magnitude. The figure shows that for longitudinal slip magnitudes close to 0.0 $[m/s]$, the three position estimator attain similar error-build up scores. The figures also shows that as the longitudinal slip magnitude increases, the position estimator using the constrained IMM as the velocity estimator, has significantly better performance than the SVMIMM. This increase in performance is reflected in Table 5-6, summarizing the respective average accuracy improvements of the velocity correction methods.

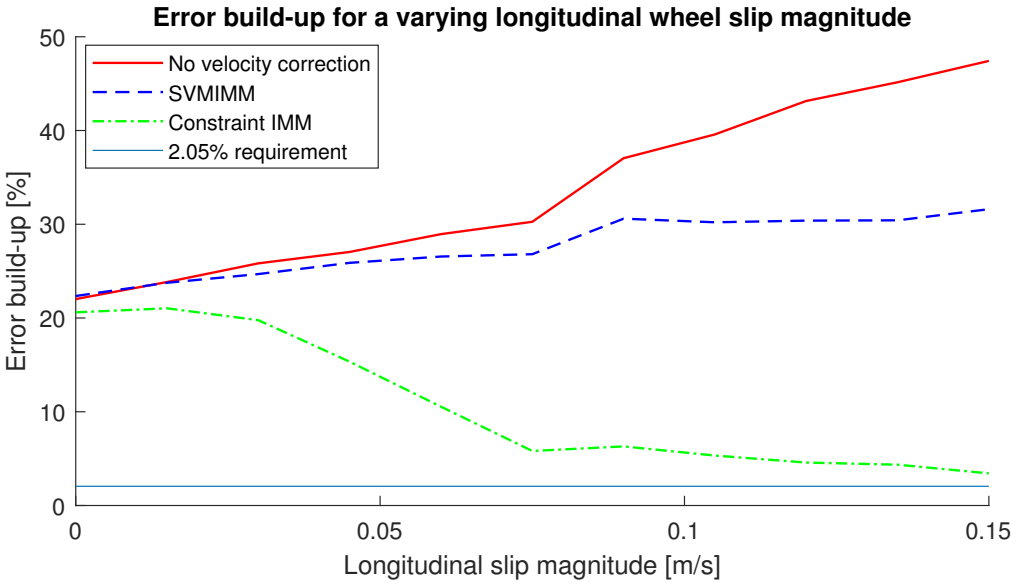


Figure 5-11: Graph of error build-up scores between simulated ground truth position and estimated position using different velocity correction methods under varying longitudinal slip magnitude parameter '$SlipMagMax$'.

	No VC	SVMIMM	Cons. IMM
Nominal error build-up	47.42%	31.61%	**3.43%**
Average error build-up	33.66%	27.57%	**10.65%**
Accuracy improvement	-	18.09%	**68.36%**

Table 5-6: Table summarizing the nominal and average error build-up – and the accuracy improvement of the EKF in conjunction velocity correction methods under a varying longitudinal wheel slip magnitude.

Discussion

Constrained IMM The simulated results show that under nominal conditions error build-up of between $3.43 - 3.64\%$ is attained by the constrained IMM and so it does not satisfy the requirement of a maximum error build-up of 2.05%. The simulated results also show that the performance of the constrained IMM degrades with respect to external perturbation parameters. The performance of the constrained IMM is mainly degraded when the longitudinal slip magnitude is below 0.1 $[m/s]$ or when the model mismatch exceeds $\pm 20\%$. As was shown in Chapter 4, Figure 4-7, the accuracy of the SVM slip detector starts to degrade for longitudinal slip magnitudes below 0.1 $[m/s]$. As a result, the SVM slip detector fails to detect slip causing the performance of the constrained IMM to approach that of the EKF without velocity correction. An increase in model mismatch beyond $\pm 20\%$ also decreases the prediction accuracy of the SVM slip detector, as shown in Figure 4-6. However, the degrading performance when faced with, for instance varying model mismatch, is not all due to the SVM slip detector. For instance, at a model mismatch of -20% the SVM slip detector has an accuracy of 90%, as displayed in Figure 4-6. Using the values in Table 3-9, a perfect

velocity corrector would reduce the error build up by $46.02\% \cdot 90\% = 41.42\%$ to approximately $50.50\% - 41.42\% = 9.08\%$. However, Figure 5-10 shows that the error build-up is only reduced to 32.84%, indicating that the varying external perturbation parameters also degrade the velocity corrector. Increases in model mismatch produce higher values for $\boldsymbol{f}_1$, depicted in Equation (4-4), causing the velocity corrector to more frequently detect stationary slip, thus underestimating the true velocity. Similarly, a reduction in longitudinal wheel slip causes the velocity corrector to more frequently detect dynamic slip, thus overestimating the true velocity.

SVMIMM The SVMIMM does not attain the performance indicated by [74] since under nominal conditions the SVMIMM only achieves an error build-up score of $36.20\% - 37.65\%$. Similar to the constrained IMM, the performance severely degrades for a model mismatch beyond $\pm 20\%$. This because the SVM slip detector over estimates the amount of slip, causing the SVMIMM to integrate the acceleration in order to estimate the true velocity. Since slip is detected during prolonged a duration, errors in the acceleration data accumulate, inducing large position errors. An additional problem of the SVMIMM is that when the SVM slip detector fails to detect slip, the SVMIMM misses crucial acceleration data it needs to approximates the true velocity. The performance of the SVMIMM is thus much more sensitive to degrading performance of the SVM slip detector compared to the constrained IMM, since the constrained IMM simply makes an assumption on the true velocity when slip is detected.

5-2-4 Overview

The objective of this section was to *determine if the proposed constrained IMM is a significant improvement over the conventional IMM under varying ship hull surface conditions.*

The average error build-up scores obtained by the velocity correction methods under nominal and varying ship hull conditions are summarized in Table 5-7.

Average error build-up	No VC	SVMIMM	Cons. IMM
Nominal conditions	47.42%	32.61%	**3.43**%
Varying model mismatch	47.42%	31.00%	**7.04**%
Varying longitudinal wheel slip	33.66%	25.57%	**10.65**%

Table 5-7: Table summarizing the average error build-up scores of the velocity correction methods under nominal, varying model mismatch and varying longitudinal wheel slip conditions.

Limitations The evaluation of the velocity correction methods is limited. Firstly, the constrained IMM is assumed to have a perfect model of the dynamic slip mode, modelled in Section 2-4-1. In comparing the velocity estimators to each other this is obviously in the favor of the constrained IMM. Another limitation is that the velocity correction methods were only evaluated for a fixed longitudinal slip magnitude '$SlipMagMax$' and a varying model mismatch 'MM' and vice versa. The correction methods were thus not evaluated for all possible combinations of those two parameters. Like in the previous chapter, the SVM slip detector used by both the SVMIMM and the constrained IMM - and the SVM mode predictor used solely by the constrained SVM as a mode predictor, were trained on a perfectly labeled

training data set. This is unrealistic, since the real sensor data is hand labeled using camera footage, which is susceptible to mislabeling.

5-3 Velocity estimator validation

In the previous section it was determined that the EKF using the constrained IMM as a velocity estimator yields the lowest error build-up between the simulated ground truth and the estimated position and the highest accuracy improvement with respect to the EKF without velocity correction. The results were however obtained in a simulated environment which suffers from inherent limitations, discussed in the previous section.

The objective of this section is thus to *verify the simulated performance of the EKF using the constrained IMM as a velocity estimator, using real sensor data.*

Firstly, the SVM mode predictor used by the constrained IMM is tuned. Secondly, the EKF estimated position using the two velocity correction methods and the EKF without velocity correction are compared using the real data sets also employed in Section 3-2-3. Finally, The results are discussed at the end of this section.

5-3-1 SVM mode predictor tuning

The data that is used to tune the mode prediction SVM is labeled using camera footage and originates from operation on the HS TOSCA, on the 6^{th} of July, 2017. The labeled data used to tune and validate the classifier is depicted in Figure 5-3. The data displayed in the figures is adjusted such that only the data points where slip is observed remain, yielding a sample size of $N = 1516$. The data is then labeled such that when dynamic slip occurs the class label is 1 and when stationary slip occurs, the class label is 0.

The box constraint parameter 'C' is tuned by evaluating the average prediction accuracy of a 4-fold cross-validation on a tuning data set, as displayed in Figure A-9.1, for different C. A one dimensional grid search was conducted for $C = [10^{-3} : 10^{3}]$, followed by a grid search from $[0.2 : 1.8]$, obtaining the optimal box constraint parameter $C = 1$, with an average label prediction accuracy of 94.14% on the tuning data set. Validating the tuned SVM on the separate validation set yielded a label prediction accuracy of 96.37%. The sigmoid parameters, used in Equation (5-4) to predict the mode probabilities are $(A, B) = (-1.826, -0.480)$.

5-3-2 Velocity estimator performance

The same validation data sets as used in Section 3-2-3 are used here to evaluate the velocity estimator performance. The EKF estimated travel distances, using the two different velocity correction methods, are summarized in Table 5-8. The respective error build-up values, as defined in Equation (1-1), are summarized in Table 5-9. The tables show that the constrained IMM in all but one case improves the error build-up quite significantly over both the EKF without the use of velocity correction and the EKF in conjunction with the SVMIMM.

Data set	No correction	SVMIMM	Cons. IMM	True
1	6.8 $[m]$	6.7 $[m]$	**3.9** $[m]$	4.4 $[m]$
2	5.5 $[m]$	4.9 $[m]$	**3.6** $[m]$	3.5 $[m]$
3	8.7 $[m]$	9.8 $[m]$	**7.1** $[m]$	6.5 $[m]$
4	6.5 $[m]$	6.3 $[m]$	**5.5** $[m]$	5.7 $[m]$
5	**11.6** $[m]$	11.2 $[m]$	10.0 $[m]$	11.9 $[m]$

Table 5-8: Table summarizing position estimates using various velocity estimation methods.

Data set	No correction	SVMIMM	Cons. IMM
1	54.55%	52.27%	**12.82**%
2	57.14%	40.00%	**2.86**%
3	33.85%	50.77%	**9.23**%
4	14.04%	10.53%	**3.51**%
5	**2.52**%	5.88%	15.97%
Average error build-up	32.42%	31.89%	**8.88**%

Table 5-9: Table summarizing error build-up in position estimates per true meter traveled using various velocity estimation methods.

Discussion

The validation results depicted in Table 5-9 confirm that the constrained IMM attains the lowest error build-up score, reducing the average error build-up of the EKF without velocity correction by a factor of four to 8.88%.

The results also show that the SVMIMM hardly improves the EKF without velocity correction, only improving the error build-up score by 0.53%.

The variation in error build-up score and the predicted travel distances summarized in Table 5-8 suggest that the high error build-up scores of the constrained IMM are mainly due to the SVM slip detector detecting slip when the robot is not slipping, undershooting the true distance traveled.

The results show that the constrained IMM reduces the error build-up in most cases with respect to the EKF using no velocity correction method and the EKF using the SVMIMM as a velocity correction method, thus confirming the simulated results. Overall, the constrained IMM produced an accuracy improvement of 72.60% over the EKF without velocity correction. This accuracy improvement falls in the middle of the accuracy improvement range of 60.53% to 84.44%, encountered in comparable mobile robots, discussed at the beginning of this chapter. Contrary to what was expected from the simulations, the SVMIMM does not improve the EKF position estimate without any velocity correction method. It is expected that this is due to the noise present on the acceleration data, and the fact that the SVMIMM is prone to missing valuable acceleration data if slip is not detected by the SVM slip detector.

5-4 Conclusion

The objective of this chapter was to *design a velocity correction algorithm, that can be used in conjunction with the EKF and the SVM, that attains an as low as possible error build-up score.* This objective is achieved by first proposing working principles by reviewing literature on velocity correction methods for mobile robots - and testing and validating these working principles using simulated data and real sensor data, respectively.

Since the velocity correction method has to work in conjunction with the EKF and also the SVM designed in the previous chapter the IMM was selected as the most suitable working principle to correct the velocity during slippage. Conventionally, IMMs estimate the velocity during slippage by integrating the IMU acceleration output. However, due to the prolonged duration of slippage observed during several operations and the amount of sensor noise on the IMU acceleration output of the Fleet Cleaner robot, this solution seemed questionable. Instead, a constrained IMM making use of a mode predicting SVM, mixing a dynamic and stationary slip model was proposed to estimate the velocity during slippage.

Like the SVM designed in Chapter 4, the mode predicting SVM makes use of a feature that is susceptible to varying ship hull conditions and so the constrained IMM was simulated using data emulating these conditions. The constrained IMM was also juxtaposed against a conventional IMM using the IMU to estimate the velocity during slippage. Finally, the simulated results were validated using real sensor data and the accuracy improvement was compared to that of wheeled mobile robots in comparable circumstances.

Results and conclusions The simulated results showed that the EKF in conjunction with the constrained IMM retains an error build-up score of 3.64% to 32.84% under varying ship hull conditions, yielding an average accuracy improvement range of $68.36\% - 85.10\%$ over the EKF without velocity correction. The simulated results also suggested that the constrained IMM is an improvement over the conventional IMM which only achieved an accuracy increase in the range of $18.09\% - 34.38\%$. The simulated results indicate that the decreasing performance of the constrained IMM coincides with the decrease in performance of the SVM slip detector. However, the simulated results also show that the increase in error build-up when faced with varying ship hull conditions is due to a reduction in velocity corrector performance. So even if a perfect slip detector was available, the constrained IMM would not suffice to reduce the error build-up to within 2.05%.

The average error build-up using real sensor data of five independent data sets, originating from three different cleaning operations, was reduced from 32.42% to 8.88%. The constrained IMM thus does not satisfy the requirement of a maximum error build-up of 2.05% but it does reduce the error build-up of the EKF by a factor of 3.65, significantly reducing workload on the operator.

Limitations The design of the constrained IMM that is proposed in this chapter is limited since it made use of an SVM that was trained on data hand labeled using camera footage. As was already seen in Chapter 4 such data is susceptible to mislabeling. Furthermore, the dynamic model was trained using data that was assumed to be a ground truth velocity. However, this assumption has not been verified on the basis of experimentation due to the lack of resources. The performance of the velocity correction methods was tested using simulated data

mimicking the varying conditions of a ship hull. The variability of the conditions was however limited to model mismatch 'MM' and the longitudinal slip magnitude '$SlipMagMax$'. The performance of the velocity correction methods was only evaluated for a fixed 'MM' and a variable '$SlipMagMax$' - and vice versa, since evaluating the performance for each possible combination would be too time consuming. The dynamic slip model data is limited because it is very questionable if the 'true' velocity used to identify the model is in fact the true velocity. However, the identification of the model was limited to this data since no true velocity data is available.

Recommendations Following the previously discussed limitations it is recommended that additional wheel encoders are installed to track the true velocity of the robot. If wheel slip occurs for all but one wheel, wheel encoder output associated to the wheel that is not slipping can be used to estimate the true velocity. If no wheel encoders are added to the robot it is recommended that some test is conducted to obtain better true velocity data with which the dynamic slip model is identified. Both the simulated results in Chapter 4 and in this chapter showed that the performance of the slip detector and velocity corrector will steeply degrade when faced with model mismatch. So if no additional wheel encoders are added to the robot, it is recommended that the ARX model that estimates the velocity of the robot when it is not slipping is made adaptive.

Chapter 6

Heading Correction

In Chapter 3 it was shown that the position estimation is corrupted due to drift in the IMU heading output, inducing an error build-up of 6.19% over a time period of 60 seconds. This is problematic as it induces large errors between the estimated position and the true position and puts additional workload on the operator of whom it is currently required to reset the estimated heading intermittently.

In order to alleviate the position estimate of the aforementioned problems, the drift in the IMU heading output must be accounted for. The reduction of the IMU heading drift will not only improve the heading estimate of the IMU, but also reduce the error build-up in the x- and z-position estimates, thus reducing workload on the operator and producing a more realistic trajectory map for the customer.

The objective is thus to *design a heading correction algorithm that accounts for the heading drift of the IMU output, and reduces the error build-up position estimate.*

In Section 6-1 a working principle is proposed that accounts for the IMU heading drift and reduces error build-up. In Section 6-2 this working principle is tested using simulated data to verify the workings of the principle and to determine its shortcomings. In Section 6-3 the simulated results are validated using real data.

6-1 Heading correction working principle

Although IMUs with a referenced heading, using the earths magnetic field exist they cannot be employed by the Fleet Cleaner robot as it utilizes large magnets which will distort the magnetometer [10]. Another approach, employed by Barshan et al., [99], is to estimate a drift model by evaluating the sensor drift when the IMU is known to be stationary. This would yield a constant drift rate value but as was shown in Section 2-2, the drift rate varies with time. A different method for accounting for the heading drift thus needs to be devised.

In this section it is proposed that knowledge about ship hull shapes can be used to combat the drift in the IMU heading orientation. The general idea is based on the fact that most

oceanic trade vessels, that Fleet Cleaner has cleaned so far, have a flat ship hull side section[1] – and the assertion that this section can be detected using sensor data. If the ship side is detected, the heading orientation, defined by θ in Figure 1-4, can be reset to its initial value, assuming the IMU was initialized on the ship side. Incidentally, if the ship side of the ship is detected, the $z-$coordinate position estimate can also be reset to its initial value, eliminating the error build-up in the $z-$coordinate position estimate, also shown in Figure 1-4.

Heading corrector requirement Resetting the heading and z-coordinate position estimate can improve the position estimator will reduce the error build-up in the position estimate. However, if the heading and $z-$coordinate position estimate are reset while the robot is on the bow of the ship, a so called false positive – it may in fact induce very large errors and increase error build-up. In order to decrease the chances of this happening the ship side detector is required to

1. have an false positive rate of less than 0.0005 per bow crossing [2],
2. retain the nominal error build-up score obtained by the constrained IMM when faced with heading drift rate in the range of $[-0.0037 \quad 0.0037]$ $[rad/s]$.

Proposed heading corrector workings The workings of the proposed heading corrector are displayed in Figure 6-1. As is shown in the figure, the detection of the ship side using sensor data requires the selection of an appropriate classifier, the design of features used by the classifier to detect the flat side of the ship and a state resetting method.

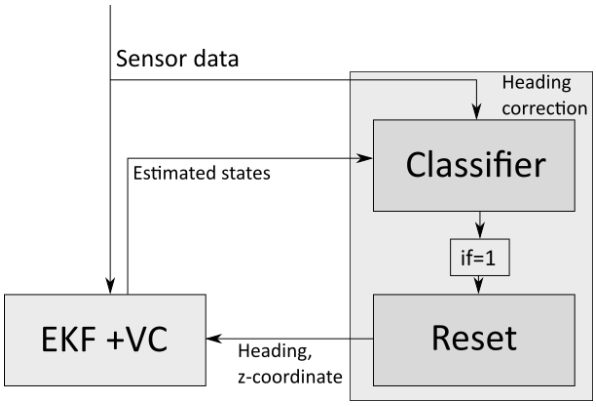


Figure 6-1: Illustration of proposed heading correction algorithm. The incoming sensor data and the corrected wheel encoder velocity is used by a classifier to determine if the robot is on the flat side of the ship hull. If the flat side of the ship hull is detected, the heading and the $z-$coordinate position estimate can be reset to their respective initial values, as long as the robot has been initialized on the flat side of the ship hull.

The objective of this section is thus to *design features and select a suitable classifier to detect the flat side of the ship hull such that the IMU heading and $z-$coordinate position estimate can be corrected.*

The section is structured as follows. In Section 6-1-1 features are selected that can be used to detect the side of the ship. In Section 6-1-1 a classifier is selected that suits the selection of features and other to be determined requirements. In Section 6-1-4 a method for resetting the heading and $z-$ coordinate position estimate is proposed. Finally, in Section 6-1-5 an overview is given of the design choices that were made and their possible shortcomings

[1] This flat ship hull side section will be referred to simply as 'the side of the ship' or 'ship side'.
[2] This failure rate was discussed with Fleet Cleaner

6-1-1 Feature engineering

In Chapter 4 the features were determined by reviewing research done on wheel slip detection for mobile robots on uneven or loose surfaces. This tactic can not be used to determine appropriate features for the detection of the flat side of the ship hull as the topic is uncommon. Instead the features are determined based upon physical insight into the shape of ship hulls.

Roll feature

A property of the side of the ship is that it is vertical, and so an obvious way to detect the flat side of the ship hull is by gauging the extrinsic roll ϕ_t. If the robot is initialized on the side of the ship such that $\phi_0 = 0$, the roll will deviate from zero when it approaches the underside of the ship or the bow of the hull, as displayed in Figure 6-2 (Left).

Figure 6-2: Images of oceanic trade vessels, displaying the ship hull shape. *Left:* Mc-Kinney Møller, container transport vessel [100]. *Right:* Teseo, crude oil tanker [101].

A drawback to this feature is that it can only be used for the detection of the side of the ship if the side of the ship is the only vertical ship hull section. As can be seen in Figure 6-2 (Right) ship hull shapes exist where the bow of the ship, although curved, is still vertical. If a classifier is solely using the roll feature, it would in this case not be able to distinguish between the curved front part of the ship hull and the flat side of the ship hull. Therefore an additional feature that measures the curvature of the surface the robot is on is proposed next.

Curvature feature

The difference between the side and the bow of ship displayed in Figure 6-2 (Right), is that movement over the bow will yield a *curved* trajectory and movement over the side will yield a straight trajectory. The curvature 'κ' of a path $\boldsymbol{q}(s)$, parameterized in the path length 's', is defined as the directional change of the tangent '$T(s)$' of $\boldsymbol{q}(s)$ in the direction of the surface normal vector '$N(s)$' [102], as shown in Figure 2-5. A flat surface will have a curvature of $\kappa = 0$. There are different ways to estimate this curvature, and the selection of the appropriate estimation method depend upon the feature requirements, discussed next.

Feature requirement It is required of the feature that it can be used to distinguish the side of the ship from the bow of the ship. This task becomes more difficult as the curved section of the ship hull is less curved, and $\kappa \to 0$. Such small curvatures occur on very large oceanic oil tankers, such as the HS Teseo displayed in Figure 6-2 (Right). The HS Teseo has a width of 44 $[m]$ [103], and so its curvature is estimated to be approximately $\kappa = \frac{1}{R} = 0.0455$. In order to be able to distinguish flat surfaces from curved surfaces, the curvature feature estimate '$\hat{\kappa}$' requires at maximum bias of $E[\hat{\kappa}] - \kappa < 0.0455$. Furthermore, the lower the variance of the estimate, the better the clusters become separated, the higher the prediction accuracy of the classifier will be. A low estimator variance is thus also required.

Estimator selection Curvature estimation can be broadly divided into *analytic* and the more simplistic *numerical estimation methods*. Analytic estimation methods rely on fitting a surface to a spatial point cloud near the point of interest and determining the derivatives of that surface at the point of interest to determine the curvature. Numerical estimation methods estimate the curvature using the difference in normal direction of the surface between adjacent spatial points. In their research P. Flynn et al., [104] compare analytic curvature estimation methods to numerical methods by simulating trajectories to obtain orientational vectors in each point along the trajectory, similar to what was done in this thesis using the Frenet-Serret model, adding zero mean Gaussian noise with standard deviation 0.03 to the orientational vectors and using different methods to estimate the true curvature. P. Flynn et al., have shown that numerical curvature estimation methods can be as accurate as analytic techniques. In their research they also point out that it is difficult to get the curvature estimates better than 10% accuracy if the surface is not flat. However, surfaces with zero curvature were estimated to within $2.57 \cdot 10^{-2}$ using a numerical approach, calculating the curvature using the change of the surface normal. Flynn et al., added noise to the simulated sensor signals with a standard deviation of 0.03. As was summarized in Table 2-1, the noise on the orientation data provided by the IMU is an order of magnitude smaller. It is therefore expected that the results of Flynn et al., can be reproduced.The curvature is thus estimated numerically, by calculating the change in the surface normal.

Research [102, 105] has shown that noise severely deteriorates the curvature estimation accuracy. To get reliable results using numerical curvature estimation, a Gaussian filter is must used to reduce the noise on the input data [102, 105].

Surface normal change estimator The surface normal change method used to estimate the curvature is described below [106]. Let p be the point of interest for calculating the curvature and $q_{n,m}$ be a point in a two dimensional $N \times M$ neighbourhood of p. The curvature at p in the direction of $q_{n,m}$ is

$$\hat{\kappa}_{p,q} = \begin{cases} \frac{||n_p - n_q||}{||p-q||} & \text{if } ||p-q|| < ||(n_p+p)-(n_q+q)|| \\ -\frac{||n_p - n_q||}{||p-q||} & \text{if } ||p-q|| > ||(n_p+p)-(n_q+q)|| \end{cases}$$

.

where n_p and n_q are the surface normals at point p and q, respectively. In order to increase the estimator accuracy, $\hat{\kappa}$ is calculated over an $N \times N$ [104] range and estimated using the LLS [21, p. 28] solution. A downside of this method is that the length of the chord between

q and p is taken as the distance between points instead of the geodesic distance, reducing estimation accuracy [106].

A way to alleviate this problem is by using the Frenet-Serret model described by Equation (2-6), as it relates the change in the tangential vector T_t in the surface normal direction to the distance traveled over the manifold $d_t = \hat{v}_{t-1} \cdot h + d_{t-1}$ through intrinsic manifold parameters κ and τ, which represent curvature and torsion respectively.

Using forward Euler integration to discretize the continuous time Frenet-Serret model yields

$$\begin{bmatrix} T(k+1) \\ N(k+1) \\ B(k+1) \end{bmatrix} = \begin{bmatrix} 0 & \kappa & 0 \\ -\kappa & 0 & \tau \\ 0 & -\tau & 0 \end{bmatrix} \begin{bmatrix} T(k) \\ N(k) \\ B(k) \end{bmatrix} \cdot \hat{v}(k) \cdot T_s + \begin{bmatrix} T(k) \\ N(k) \\ B(k) \end{bmatrix}, \tag{6-1}$$

where $\hat{v}(k)$ is the estimated velocity using the EKF in conjunction with the constrained IMM and T_s is the sampling time. Rearranging yields

$$\underbrace{\begin{bmatrix} T(k+1) \\ N(k+1) \\ B(k+1) \end{bmatrix} - \begin{bmatrix} T(k) \\ N(k) \\ B(k) \end{bmatrix}}_{\boldsymbol{y}} = \underbrace{\begin{bmatrix} N(k) \cdot \hat{v}(k) \cdot T_s & 0 \\ -T(k) \cdot \hat{v}(k) \cdot T_s & B(k) \cdot \hat{v}(k) \cdot T_s \\ 0 & -N(k) \cdot \hat{v}(k) \cdot T_s \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} \kappa \\ \tau \end{bmatrix}}_{\boldsymbol{x}}, \tag{6-2}$$

which can be solved using LLS solution [21, p. 28]. The vector $\boldsymbol{y}$ can be obtained using the IMU orientation output and H using the IMU orientation output and the estimated velocity. In order to attenuate the noise in the curvature estimate $\hat{\kappa}$ [104], $\boldsymbol{y}$ and H can be calculated over the sliding window ranging from $k - N$ to k, where N is the window size. A way for determining the window size N is discussed in Section 6-1-1.

Gaussian filter smoothing In order to increase the curvature estimation performance the input data used in $\boldsymbol{y}$ and H is filtered using a Gaussian filter.

$$x_n = \frac{1}{\sqrt{2\pi}\sigma_1} \sum_{i=k-N_1}^{k} \exp \frac{(n-i)^2}{2\sigma_1^2} \cdot x_i, \tag{6-3}$$

where, x_i is the input at time instance i, 'N_1' is the window size and σ_1 the standard deviation of the Gaussian. For each sensor input the moving window size N_1 and the standard deviation σ_1 is a tuneable parameter, the optimization of which is discussed in Section 6-1-3.

Curvature feature variance

As was discussed in [104], the curvature estimation of flat surfaces using the surface normal change estimator yields more accurate and precise results compared to the curvature estimation of non-flat surfaces. It is thus anticipated that the variance of the estimator $\hat{\kappa}$ can be used to distinguish between flat surfaces and non-flat surfaces. The uncertainty of the LLS solution of

$$\boldsymbol{y} + \boldsymbol{\epsilon} = H\boldsymbol{x}, \tag{6-4}$$

where ϵ is the standard deviation of the measurement vector $\boldsymbol{y}$, then the uncertainty of the LLS solution is [21, p. 110]

$$\begin{aligned} E[\hat{\boldsymbol{x}} - \boldsymbol{x}] &= (H^T H)^{-1} H^T E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] H (H^T H)^{-1} \\ &= (H^T H)^{-1} H^T \Sigma_{\boldsymbol{y}}^2 H (H^T H)^{-1}, \end{aligned} \tag{6-5}$$

where $\Sigma_{\boldsymbol{y}}^2 = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})(\boldsymbol{y}_i - \bar{\boldsymbol{y}})^T$, N is the moving window size and $\bar{\boldsymbol{y}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i$. The sliding window size N is a tuneable parameter, the optimization of which is discussed in Section 6-1-3.

The previously designed features are combined to form the standardized feature vector

$$\boldsymbol{\zeta}_2 = \begin{bmatrix} std(|roll|)^{-1} & 0 & 0 \\ 0 & std(|\hat{\kappa}|)^{-1} & 0 \\ 0 & 0 & std(|var(\hat{\kappa})|)^{-1} \end{bmatrix} \left(\begin{bmatrix} |roll| \\ |\hat{\kappa}| \\ |var(\hat{\kappa})| \end{bmatrix} \right) - \begin{bmatrix} mean(|roll|) \\ mean(|\hat{\kappa}|) \\ mean(|var(\hat{\kappa})|) \end{bmatrix}, \tag{6-6}$$

with a total of 3 tuneable parameters, namely the Gaussian filter parameters (σ_1, N_1) and the sliding window length N.

Feature parameter tuning

The smoothing using a Gaussian filter and the calculation of $\hat{\kappa}$ over a moving window induces a delay between $\hat{\kappa}$ and the true class labels. In order to account for this delay, the true class labels are shifted by N_{delay}. Using this delay, the class labels are predicted as

$$delta(k - N_{delay}) = Classifier(\boldsymbol{\zeta}_2(k)). \tag{6-7}$$

Adding N_{delay} to the tunable feature parameters, yields the tunable parameter set displayed in Table 6-1.

	Gaussian filter $(\hat{v}, T, N, B)$	Delay	$\hat{\kappa}$, $var(\hat{\kappa})$
Parameters	(σ_1, N_1)	(N_{delay})	(N)

Table 6-1: Table summarizing the tuneable parameters of the selected flat ship hull side detection features.

In Chapter 4 the feature parameters were tuned by evaluating the PCC, depicted in Equation 4-11, of the features with respect to the true class labels for different parameter values. The SVM slip detector required only the tuning of 2 feature parameters, affecting two of the total of four features. In this cases there are four tuneable feature parameters affecting two of the three features, making correct tuning of the feature parameters more urgent. So to get optimal classifier performance it is decided to tuned the feature parameters simultaneous to the classifier parameters using a K-fold cross-validation grid search, discussed in Section 4-1-3.

Next, a classifier is selected that is suitable for detecting the flat side of the ship hull using the features designed in this section.

6-1-2 Heading correction classifier

The classification method that is suitable to distinguish between the side of the ship and the bow of the ship, is subjected to different requirements than the slip detection classifier, designed in Chapter 4. The requirements that the selection of the classifier is subjected to are discussed next.

Classifier requirements The previously designed SVM was mainly selected on its high prediction accuracy making the true positive rate and false positive rate approximately evenly high if there are an equal amount of (1) class labels and (0) class labels. High prediction accuracy is only of secondary importance for the detection of the ship side, and a low false positive rate is far more important. This is because if a false detection of the ship side is made while the robot is on the bow of the ship, the EKF states will be reset, inducing large position estimate errors. Another requirement of the classification method is that is has a short training time. This is because the feature vector designed in the previous section requires the tuning of 4 parameters using a grid search, thus requiring at least $K \cdot c^4$ training iterations, where K is the amount of folds of the K-fold cross-validation and c are the amount of points per grid dimension. To summarize, the classification method is required to-

1. acquire a maximum false positive rate of 0.0005,
2. have a short training time such that larger grids can be searched,
3. have a small amount of tuning parameters such that the grid search consumes less time.

Feature characteristics The selection of an appropriate classifier also in large depends upon the characteristics of the features that are used. As was discussed in [104] and [105], the numerical estimation of the curvature is sensitive to sensor noise and so it is expected that the feature $|\hat{\kappa}|$ will be noisy. Furthermore, the EKF position estimate can be used to determine if the robot is on the side of the ship or on the bow, up to some safety margin proportional to the error build-up of the position estimate, making labeled data abundant. The feature characteristics are summarized below.

1. The curvature estimate $\hat{\kappa}$ is expected to be noisy.
2. The feature vector is low dimensional.
3. The classifier has access to a large labeled training data set.

From the classifier requirements and the third feature characteristics it is determined that a supervised, *lazy*, probabilistic classifier is most suitable to detect the flat side of the ship hull. A lazy learning algorithm has zero training time, as it simply stores the labeled data and uses it directly to classify incoming data. Like the mode probability prediction classifier employed in Chapter 5, a probabilistic classifier is used to determine the class likelihood $P(C_k|\boldsymbol{f}_i)$. A threshold can be set beyond which $P(C_k|\boldsymbol{f}_i)$ is classified as (1) (side of the ship hull), reducing the false positive rate.

An algorithm that fits all these requirements is the k-Nearest Neighbours (kNN) classification algorithm as it supervised, lazy, has only one tuning parameter and can produce a probabilistic output. Furthermore, the kNN can be tuned to be made robust against noise [107] and has a label prediction accuracy similar to that of the SVM for classification problems with low (<50) feature dimensionality [108]. A downside to the kNN being lazy is that it requires more storage space than an *eager* classification algorithm, like the SVM, to store its training data and the classification time increases as the training data sample size increases [109]. However, this is not problematic if the classification time stays within the sample time of $Ts = 0.03125$. For the reasons mentioned, the kNN classifier is used to detect the flat side of the ship hull.

k nearest neighbours algorithm

The kNN is trained by providing data pairs $\langle \boldsymbol{f}_i, delta_i \rangle$, where $\boldsymbol{f}_i$ is a feature vector and $delta_i$ is the class label $\{0, 1\}$ associated to feature $\boldsymbol{f}_i$. Given a set of data pairs $\langle \boldsymbol{f}_i, delta_i \rangle$, unlabeled feature points $\boldsymbol{f}_q$ are classified by evaluating the class labels of its 'k' *nearest* neighbours and assigning the the most common class among its neighbours to $\boldsymbol{f}_q$ [110]. The most common definition of what the nearest neighbours of point $\boldsymbol{f}_q$ are, is the minimum Euclidean distance, denoted

$$d(\boldsymbol{f}_i, \boldsymbol{f}_q) = ||\boldsymbol{f}_i - \boldsymbol{f}_q||. \tag{6-8}$$

Although the use of the Euclidean distance is common as a distance norm, it has been often shown that using distance norms based on labeled training data can significantly improve the kNN prediction accuracy [111]. The selection of k nearest neighbours is not a trivial matter and is typically a trade-off. Selecting a low k may provide a higher prediction accuracy on some data sets but is prone to over fitting on the training set and thus is in most cases not general enough and not robust to noise. Choosing a high k has the inverse consequences, the kNN will be more robust against noise but will have a lower prediction accuracy on the training set [108]. The proposed kNN thus has two degrees of freedom for design: the distance function and the selection of the amount of nearest neighbours k. The selection of a distance function is discussed next and the tuning of k is discussed in Section 6-1-3.

kNN distance function Common distance norms used in conjunction with the kNN include the Euclidian norm, discussed above, the Chi-square norm, cosine norm, Minkowski norm and Mahalnobis norm. In their paper researching the label prediction accuracy of the kNN using the former first four distance norms, Hu et al., [112] conclude that the Euclidean distance norm produces the best kNN prediction accuracy if the feature data is numerical. It is reported by K. Q. Weinberger et al., that using the Mahalnobis norm instead of the Euclidean norm can vastly improve kNN performance [113]. In their paper comparing the Mahalanobis norm to the Euclidian norm, they show that for classification problems with low sample size and input dimension, using a Mahalanobis norm instead of Euclidian norm can improve the prediction accuracy of the classifier. Although in this case the sample size is not low, the input dimension is and so the Mahalanobis norm is used as the distance function. The Mahalanobis norm is denoted

$$d(\boldsymbol{f}_i, \boldsymbol{f}_q) = (\boldsymbol{f}_i - \boldsymbol{f}_q) C_{\boldsymbol{f}} (\boldsymbol{f}_i - \boldsymbol{f}_q)^T, \tag{6-9}$$

where $C_{\boldsymbol{f}}$ is the covariance matrix of the training feature data $\boldsymbol{f}$,

6-1-3 Feature and classifier optimization

One of the classifier requirements stated that the classifier must have a low false positive rate in order to prevent large position estimate errors. The false positive rate can be influenced by adjusting the probability threshold, denoted $pThresh$, which is the class probability $P(C_k|\boldsymbol{f}_i)$ required for an input feature to be classified as (1), which means the flat side of the ship has been detected. In order to obtain the highest accuracy this threshold is set at 0.5, however, for a low false positive rate it is set at for instance 0.8. The consequence of lowering the false positive rate is a decrease in true positive rate. By increasing the probability threshold the amount of flat ship hull side detection thus decreases. Note that there is thus a trade-off between the certainty with which an input feature can be classified as (1) and the amount of true positive classifications of the flat side of the ship hull. This trade-off is reflected in the Receiver Operating Characteristics (ROC) of the probabilistic output of a classifier which evaluates the true positive rate with respect to the false positive rate [114], displayed in Figure 6-3. The ROC plots the true positive rate of the classifier on the $y-$axis and the false positive rate of the classifier on the $x-$axis, obtained using a probability threshold increasing from 0 to 1. The Area Under the Curve (AUC) of the ROC is maximized in order to maximize true positive rate while false positive rate is reduced to acceptable levels. The AUC of the ROC has been shown to be an effective metric to evaluate classifier performance if the probability threshold has not been determined yet [115]. If setting the probability threshold $pThresh$ is

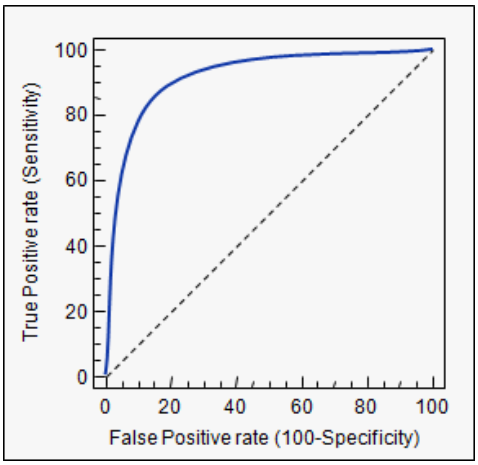


Figure 6-3: Graph displaying ROC curve of a classifier. The ROC curve depicts the trade-off between the true positive rate and false positive rate of a classifier. The graph shows that as the false positive rate is reduced, which is desirable for this application, the true positive rate decreases as well. The sensitivity of the true positive rate to a change in the false positive rate depends upon the classifier. Ideally, the true positive rate should remain as high as possible for a decreasing false positive rate, which is achieved by maximizing the AUC value of the ROC. https://www.medcalc.org/manual/roc-curves.php

not sufficient to acquire a low false positive rate, the rule that at least an '$NrCons$' amount of input data points must be labeled (1) before a data point is actually labeled (1), is employed. The feature and classifier parameters, summarized in Table 6-2, are optimized by evaluating the average AUC of a 4-fold cross-validation for different parameter sets $(\sigma_1, N_1, N_{delay}, N, K)$, which are varied using a grid search method. The grid point that yields the highest AUC value is selected as the optimal parameter set.

	Gaussian filter $(\hat{v}, T, N, B)$	Delay	$\hat{\kappa}$, $var(\hat{\kappa})$	kNN
Parameters	(σ_1, N_1)	(N_{delay})	(N)	K

Table 6-2: Table summarizing the tuneable parameters of the selected flat ship hull side detection features and classification method.

6-1-4 Heading and position correction

Using the flat side detection algorithm designed in the previous section the position in z-direction and the heading can be corrected. The objective phrased at the beginning of chapter states that the heading correction algorithm should reduce the error build-up of the position estimate. In order to achieve this a *strong constraint* is used as a sensor input to the EKF designed in Chapter 3. In contrast to the weak constraint employed in Chapter 5, the strong constraint has zero uncertainty and so the states are forced to the value of the strong constraint.

Heading strong constraint

Let θ_0 be the heading of the robot after it has been initialized on the flat side of the ship hull – and ϕ_t and ψ_t the roll and pitch of the robot expressed in the ship hull fixed frame, depicted in Figure 1-4. Once the flat side of the ship hull is detected, the heading of the robot is corrected using [116]

$$\hat{\boldsymbol{q}}_t = \boldsymbol{q}_{cor} \frac{\hat{\boldsymbol{q}}_t^*}{||\hat{\boldsymbol{q}}_t^2||} \hat{\boldsymbol{q}}_{t-1}, \tag{6-10}$$

where $\hat{\boldsymbol{q}}_t$ are the EKF estimated quaternion states at time instance t, and

$$\boldsymbol{q}_{cor} = \begin{bmatrix} \cos(\phi/2)\cos(\theta_0/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta_0/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta_0/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta_0/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta_0/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta_0/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta_0/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta_0/2)\cos(\psi/2) \end{bmatrix}. \tag{6-11}$$

z-coordinate strong constraint

In order to combat the drift in z- coordinate position estimate, the state of the EKF representing the $z-$ coordinate is reset to its initial value z_0 when the flat side is detected. As soon as the flat side of the ship hull is detected, the sensor model of the EKF changes to

$$y = \begin{bmatrix} v_{odo} & \hat{q}_0 & \hat{q}_1 & \hat{q}_2 & \hat{q}_3 & \mu_2 & z_0 \end{bmatrix}^T, \tag{6-12}$$

where $\hat{q}_i$ are the corrected quaternion inputs, μ_2 the depth gauge input, and z_0 the strong constraint for the $z-$coordinate position estimate. The heading corrector algorithm is depicted in Appendix A-14.

6-1-5 Overview

The objective of this section was to *design features and select a suitable classifier to detect the flat side of the ship hull such that the IMU heading and $z-$coordinate position estimate can be corrected.*

The features used to detect the side of the ship and the selection criteria for the kNN classifier are summarized in Table 6-3.

Features	
Roll feature	$f_1(k) = \lvert\phi(k)\rvert$
Curvature feature	$f_2(k) = \lvert\hat{\kappa}(k)\rvert$
Curvature feature uncertainty	$f_3(k) = \lvert var(\hat{\kappa})(k)\rvert$
Classifier selection	
kNN	Short training time
	Small amount of tuneable parameters
	Low dimensional feature vector
	Can be made robust against noise

Table 6-3: Table summarizing the features used to detect the side of the ship and the classifier selection criteria.

Limitations A drawback of the position estimator resetting procedure, used when the flat side is detected, is that in case of a false positive detection, it severely degrades the position estimate. In order to prevent false positive detections, additional detection rules, such as an '$NrCons$' amount of detections must occur in a row before the the flat side is detected, may be added. Lastly, the heading corrector performance will be limited to ships that have a flat side.

6-2 Heading and position corrector simulation

In the previous section a working principle was devised that detects when the robot is on the flat side of the ship hull and resets the heading and $z-$coordinate position estimate when the flat side of the ship hull is detected.

As was discussed previously, the drawbacks of the heading corrector are the high noise of the curvature feature [104], making it difficult to distinguish between the side of the ship and the curved sections when the curvature is low – and the false positive detection of the side of the ship, inducing large errors in the position estimate. The false detection of the side of the ship becomes likely when the robot is operating on a large oil tanker. This is because the roll feature can not be employed to distinguish the side from the fore of the ship and the SNR of the curvature feature is low. In order to explore the limitations of the proposed heading corrector, the algorithm is tested on simulated data emulating the data originating from a large oil tanker. Furthermore, the heading corrector is evaluated with respect to various ship hull curvatures and heading drift rates such that the limit of the algorithm can be determined. The heading correction algorithm is evaluated using the previously discussed requirement and objective, listed below.

- The heading corrector may not falsely detect the flat side of the ship if it induces a position error greater than 10 $[cm]$.
- The heading corrector must attain the nominal error build-up score obtained by the constrained IMM, under varying heading drift rates in the range of $[-0.0037 \quad 0.0037]$ $[rad/s]$.

The objective of this section is to *validate the workings – and determine the performance limits of the heading corrector with respect to varying ship hull curvature and heading drift, while satisfying the set requirement.*

Firstly, sensor data is simulated emulating the shape of a large oil tanker using the previously designed sensor noise models and Frenet-Serret model. Secondly, the kNN classifier and feature parameters are tuned on the simulated data and a suitable probability threshold is selected. Finally, the performance of the heading corrector is evaluated with respect to the varying ship hull curvature and heading drift rate. An overview summarizing the simulated results and its limitations is provided at the end of the section.

6-2-1 Simulation data modeling

The simulated data is used to train the kNN classifier and to evaluate the heading corrector and EKF performance with respect to varying ship hull curvature. The training data and heading corrector evaluation data are subjected to the following requirements.

The requirements of the training data are that it emulates the sensor data originating from a large oil tanker, since this is the worse case scenario from the viewpoint of the classifier, as was previously discussed. The training data must thus emulate the data from a ship with a curvature of $\kappa = 0.0455$ and a vertical bow section. Furthermore, the training data must simulate the corruption by sensor noise, wheel slip and heading drift, since these factors all affect the curvature estimate.

The requirements of the evaluation data are that the curvature can be made varying such that the detection limits of the classifier with respect to the curvature of the ship hull can be evaluated. Like, the simulated training data, the evaluation data is also corrupted by sensor noise, wheel slip and heading drift perturbations.

The Frenet-Serret model devised in Chapter 2 does not suffice to model the varying ship hull curvature, since the curvature is set as a constant. Therefore, a hybrid Frenet-Serret model that simulates the transfer from the side of the ship to the bow of the ship is devised next.

Hybrid Frenet-Serret model

In order to simulate the changing curvature of an oil tanker, the hybrid Frenet-Serret model combines a flat manifold, where

$$\begin{aligned} \kappa &= 0 \\ \tau &= 0, \end{aligned} \tag{6-13}$$

with a cylindrical manifold, where

$$\begin{aligned} \kappa &= \frac{\cos^2 \beta}{R} \\ \tau &= \frac{\cos \beta \sin \beta}{R}, \end{aligned} \tag{6-14}$$

where β is the pitch of the robot with respect to the cylinder and R is the cylinder radius, set at $R = 22[m]$ simulating the approximate curvature of a large vessel like the HS Teseo[3].

[3] http://maritime-connector.com/ship/teseo-9038866/

The hybrid model switches between the flat manifold curvature parameters and cylindrical manifold parameters using

$$\begin{bmatrix} \kappa_t \\ \tau_t \end{bmatrix} = \begin{cases} \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & if \quad \mu_1 < 0, \\ \begin{bmatrix} \frac{\cos^2 \beta_t}{R} \\ \frac{\cos \beta_t \sin \beta_t}{R} \end{bmatrix}, & if \quad \mu_1 \geq 0. \end{cases} \tag{6-15}$$

where μ_1 is the true position of the robot in x direction. The classifier training data trajectory is generated using the hybrid Frenet-Serret model using

$$\begin{bmatrix} \alpha_t \\ v_{in} \end{bmatrix} = \begin{cases} \begin{bmatrix} 0 \\ 0.2 \end{bmatrix}, & if \quad t < 0.5 \cdot Tfinal, \\ \begin{bmatrix} 0 \\ -0.2 \end{bmatrix}, & if \quad t \geq 0.5 \cdot Tfinal \end{cases} \tag{6-16}$$

as the steering angle and velocity input. This yields the trajectory displayed in 6-4 emulating a bow section crossing. The remaining perturbation parameter values are summarized in Table 6-4.

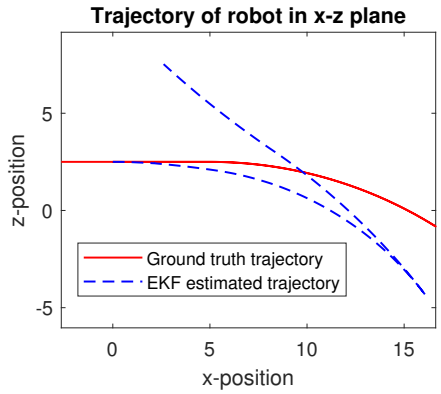


Figure 6-4: Graph displaying the trajectory in $x-, z-$coordinates of the robot during a bow crossing. The trajectory starts at $(x, z) = (-2.5, 2.5)$. The sensor inputs used by the EKF were corrupted by the perturbation models, using the perturbation parameters specified in Table 6-4, and a heading drift of -0.0037 $[rad/s]$. The error build-up between the EKF estimated trajectory and the simulated ground truth trajectory is 19.6%.

$v_{nom}(t)$	$\alpha(t)$	$SlipMagMax$	$SlipTrans$	$Drift$	MM	κ	τ	$Tfinal$
0.2	0	0.15	50	$[-0.0037 \rightarrow 0.0037]$	0	0.0455	0	200

Table 6-4: Table summarizing modeling parameters used for kNN classifier tuning.

6-2-2 Feature and classifier parameter tuning

The optimal feature and classifier parameters are found using a K-fold cross-validation grid search, discussed in Section 6-1-1 and 6-1-3. Firstly, the feature vector in Equation (6-6) is computed for some combination of feature parameters $(N_1, \sigma_1, N_{delay}, N, k)$. Secondly, the features are ordered randomly and partitioned, as shown in Figure A-9.1. Finally, the kNN is trained, using some k amount of nearest neighbours and 4-fold cross-validation on the training and test data, as displayed in Figure A-9.1. This process is repeated for each point in the search grids specified in Table 6-5. Using this method, the optimal parameter set is found to

be $(N_1, \sigma_1, N_{delay}, N, k) = (50, 150, 100, 80, 100)$, scoring an AUC score of 0.9922 on the test set and 0.9600 on the validation set.

	N_1	σ_1	N_{delay}	N	k
Grid 1	$10^{[1:3]}$	$10^{[1:3]}$	$10^{[1:3]}$	$10^{[1:3]}$	$10^{[1:3]}$
Grid 2	$10 \cdot [1:5]$	$100 \cdot [1:5]$	$100 \cdot [1:5]$	$10 \cdot [1:5]$	$100 \cdot [1:5]$
Grid 3	$10 \cdot [5:9]$	$50 \cdot [1:5]$	$50 \cdot [1:5]$	$10 \cdot [5:9]$	$50 \cdot [1:5]$

Table 6-5: Table summarizing the grid search ranges for kNN tuning. The optimal parameter set was found to be $(N_1, \sigma_1, N_{delay}, N, k) = (50, 150, 100, 80, 100)$, scoring an average AUC score on the test data set of 0.9922.

Probability threshold and consecutive detection tuning The probability threshold '$pThresh$' and the amount of consecutive detections '$NrCons$' required for a data point to be labeled 1, affects the true positive rate and the false positive rate of the kNN classifier. The parameter set $(NrCons, pThresh)$, used to assign class labels to the input features, is determined by evaluating the true positive rate and false positive rate of the tuned kNN for a varying $pThresh$ and $NrCons$, on the training data set, yielding the true positive rate and false positive rate graphs displayed in Figure 6-5.

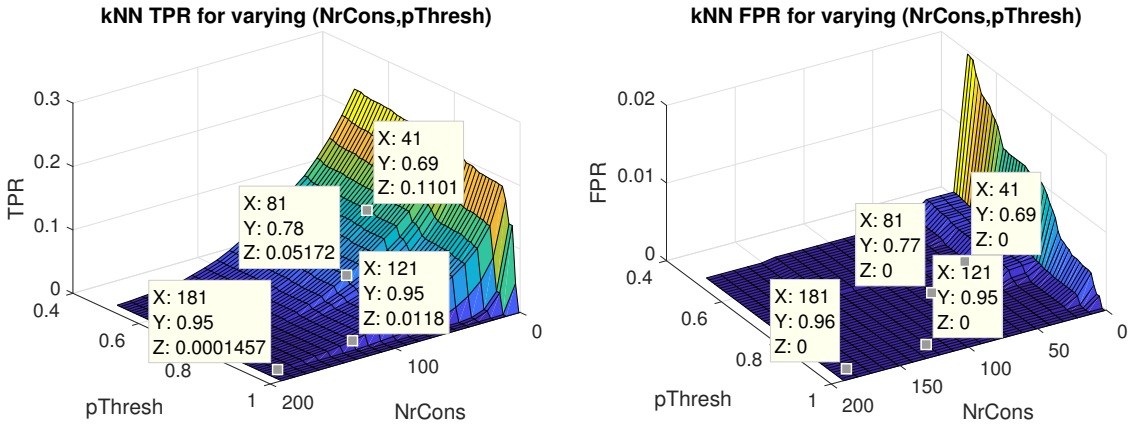


Figure 6-5: Graphs displaying the shifting true positive rate and false positive rate of the kNN classifier for a shifting $pThresh$ and $NrCons$. The four grid points selected all have a zero false positive rates, however, they have different true positive rates. It is expected that the false positive rate parameter set $(NrCons, pThresh) = (181, 0.95)$ will be most robust against false positive detections. However, compared to the parameter set $(NrCons, pThresh) = (41, 0.69)$ it will also be less able to consistently detect the side of the ship, thus increasing error build-up.

The figure shows that at $(NrCons, pThresh) = (1, 0.99)$ the false positive rate is non-zero and so it is necessary to add the rule that a number of consecutive detections must occur before input data can be labeled as (1), such that the false positive rate is reduced below 0.0005. Four threshold parameter combinations, varying in false positive rate and thus varying in robustness to false ship side detections as displayed in Figure 6-5, are selected to be further evaluated in the next section.

6-2-3 Heading correction performance simulation

In the beginning of this section it was discussed that the heading corrector is susceptible to variations in ship hull curvature and heading drift. Furthermore, it was shown that the robustness to false detections of the side of the ship depends upon the selection of the threshold parameters $(NrCons, pThresh)$. In this section a set of threshold parameters is selected that fulfill the heading correction requirements stated in Section 6-1 and it is evaluated to what extent the heading corrector improves the position estimate of the EKF with respect to the variations in ship hull curvature and heading drift.

Simulation data The modeling parameters used to generate the simulation data are set at the values summarized in Table 6-6. The parameters $Drift$ and $\kappa = \frac{1}{R}$ are varied over a to be specified K amount of steps. Each such step is iterated $K = 1000$ times to simulate different noise realizations.

$v_{nom}(t)$	$\alpha(t)$	$SlipMagMax$	$SlipTrans$	$Drift$	MM	κ	τ	$Tfinal$
0.2	0	0.15	50	0	0	0.0455	0	200

Table 6-6: Table summarizing modeling parameters used for heading corrector simulation.

Performance under varying ship hull curvature

It is expected that the false positive rate of the kNN classifier will increase as the bow curvature radius increases. The kNN has been trained on simulated data set with a bow curvature of $R = 22$ $[m]$. The influence of a varying ship hull curvature 'R' on the kNN is evaluated by fixing the perturbation parameters at the values summarized in Table 6-6 except for the bow curvature, which is increased from 2.2 to 48.4 $[m]$ in 11 steps. In Figure 6-6 it is shown that the threshold parameter sets $(NrCons, pThresh) = (121, 0.95)$ and $(NrCons, pThresh) = (181, 0.95)$ maintain a zero false positive rate up to a bow curvature radius of 48 $[m]$, whereas the less restrictive threshold parameter settings only maintain a zero false positive rate below a bow curvature of 22 $[m]$.

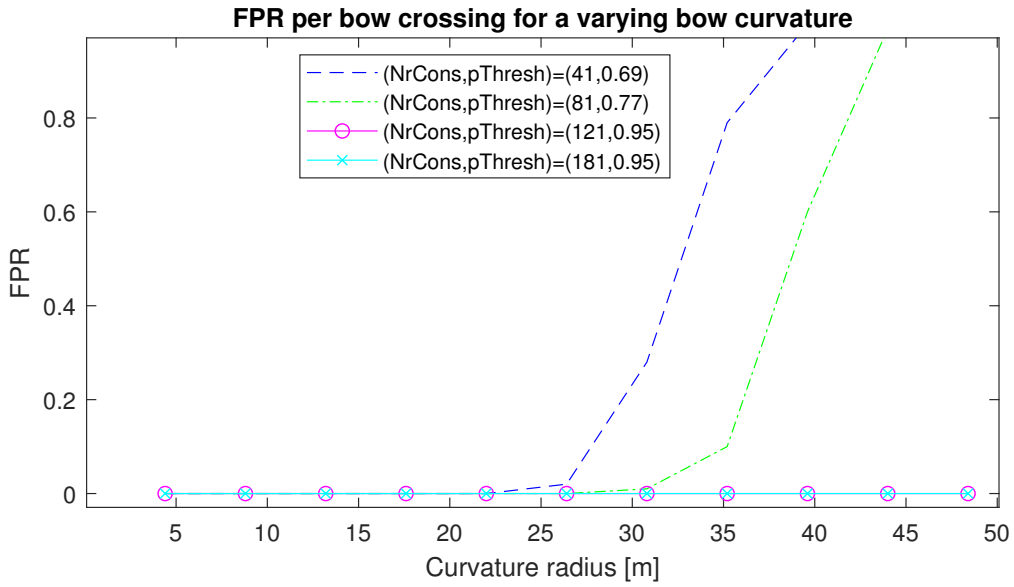


Figure 6-6: Graph showing the average false positive rate per bow crossing for variable bow curvature radius.

Performance under varying heading drift

The influence of a varying IMU heading drift on the EKF, described in A-12.1, in conjunction with the heading corrector is tested by fixing the perturbation parameters as shown in Table 6-6 except for the heading drift rate, which is increased from -0.0037 to 0.0037 in $K = 21$ steps. From Figure 6-7 it can be seen that as the threshold parameters $(NrCons, pThresh)$ go to higher values the average error build-up scores increase. In Table 6-7 it can be seen that only the threshold parameter sets $(NrCons, pThresh) = (121, 0.95)$ and $(NrCons, pThresh) = (181, 0.95)$ maintain an false positive rate below 0.0005. From Figure 6-7 and Table 6-7 it can be seen that the heading corrector using the threshold values $(NrCons, pThresh) = (121, 0.95)$ achieves the lowest error build-up while maintaining a zero false positive rate over the entire range.

$(NrCons, pThresh) \rightarrow$	$(41, 0.69)$	$(81, 0.77)$	$(121, 0.95)$	$(181, 0.95)$
FPR	0.2133	0.0190	0	0

Table 6-7: Table summarizing the average false positive rate per bow crossing over the entire variable heading drift range, for various threshold parameter settings.

Using the detection thresholds $(NrCons, pThresh) = (121, 0.95)$, an average error build-up over the entire heading drift range of 8.10% is achieved, which is an accuracy improvement of 41.66% over the EKF without heading corrector, while maintaining a zero false positive rate thus not increasing the workload on the operator. The figure also shows that for high heading drift rates, the heading corrector hardly improves the EKF without heading correction. This is because the true positive rate is very low at high heading drift rates. The figure also shows that the heading corrector manages to maintain the nominal performance of 3.64%, attained using the constrained IMM, up to a heading drift range of approximately ± 0.002 $[rad/s]$.

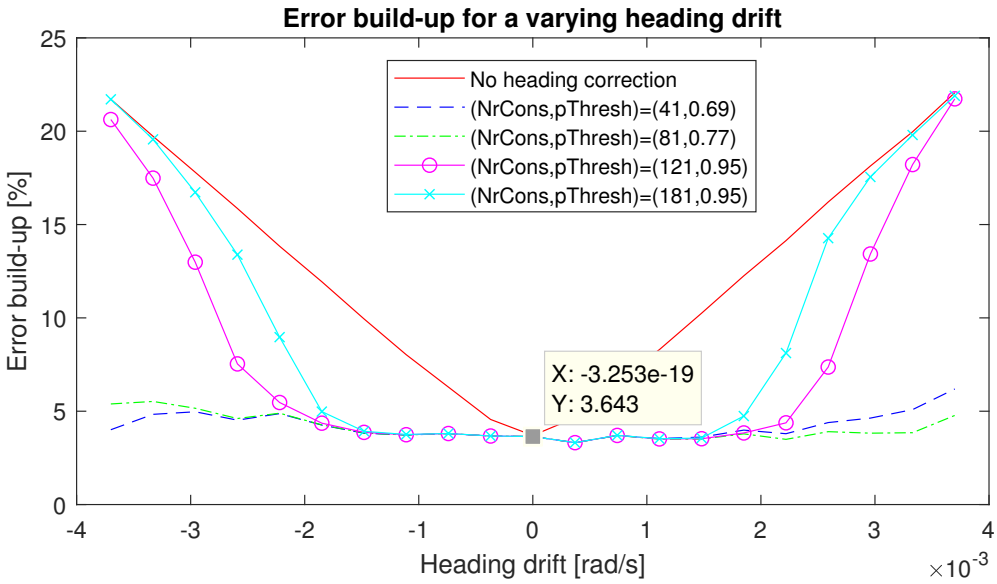


Figure 6-7: Graph displaying the error build-up of the constrained IMM in conjunction with and without heading correction.

6-2-4 Overview

The objective of this section was to *validate the workings – and determine the performance limits of the heading corrector with respect to varying ship hull curvature and heading drift, while achieving the set requirement.*

In Table 6-8 the improvements in error build-up score are summarized using the heading corrector. The table also shows the heading drift rates up until where the heading corrector attains the nominal performance of the constrained IMM.

	No HC	Heading correction
Average error build-up	41.66%	**8.10%**
Performance limits	-	$[-0.002 \quad 0.002]$ $[rad/s]$

Table 6-8: Table summarizing the average error build-up score obtained using the heading corrector in the heading drift rate range of $[-0.0037 \quad 0.0037]$, and the performance limits of the heading corrector.

Limitations The results are limited since perfectly labeled data was used to evaluate the heading corrector performance. Furthermore, the performance of the heading corrector was only evaluated with respect to varying bow curvature and heading drift individually. The consecutive labeling rule that was employed to reduce the false positive rate is quite simplistic and reduces both the true positive rate and the false positive rate. More research can be done to find rules that to a greater degree reduce the false positive rate compared to the true positive rate, which would yield a lower error build-up while maintaining a zero false positive rate.

6-3 Heading and position corrector validation

The simulated results in the previous section showed that the addition of the heading corrector reduces the error build-up to 8.10% for heading drift rates between -0.0037 and 0.0037 $[rad/]$ while maintaining a zero false positive rate for bow curvature radii up to 48 $[m]$ and drift rates in the range of $[-0.0037 \quad 0.0037]$. The results were however obtained in a simulated environment which inherently suffers from limitations, previously discussed.

The objective of this section is to *verify the simulated results obtained by the EKF in conjunction with the heading corrector, using real sensor data.*

Firstly, a suitable data set will be selected and labeled for the training of the kNN classifier and the validation of the simulated results. Secondly, the kNN classifier will be tuned and finally, the simulated results are validated using real sensor data.

6-3-1 Validation data acquisition

Data from a cleaning operation on the HS Tosca, displayed in Figure 6-8, is used to validate the simulated results. The robot was moved back and forward between a weld line and depth marking on the ship hull. Using this information, a technical drawing of the ship and the position estimator, the data displayed in Figure 6-9 is labeled as the side of the ship (1) and the bow of the ship (0). The data used corresponds to a trajectory of the robot where it stays

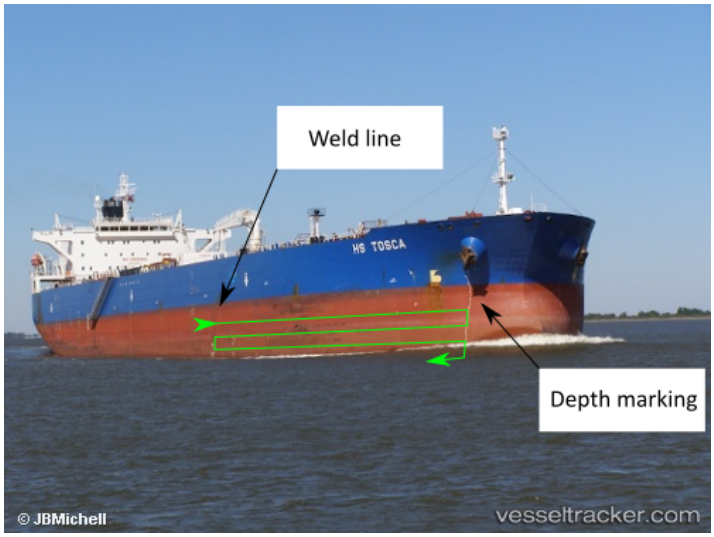


Figure 6-8: Image of robot cleaning trajectory between a weld line and depth marking on the hull of the HS Tosca.

vertical. The roll feature can thus not be used, making the classifier reliant solely on the curvature features. This data is selected to pose a 'worst case scenario' from the viewpoint of the classifier. The data sequence displayed in Figure 6-9 consists of $N = 65664$ data points, spanning a time frame of 34.2 minutes. The data is partitioned into 10 partitions by assigning one in each ten data points to a single partition. One partition is used to train and tune the classifier and the remaining nine partitions are used to validate the classifier.

The performance of the EKF in conjunction with the heading corrector is evaluated by calculating the error build-up between the two error build-up point pairs, displayed in Figure 6-9. Since the hull is vertical and the robot is maneuvered between two fixed landmarks, it is assumed that the $x-$ and $z-$ coordinate positions in these point pairs is the same.

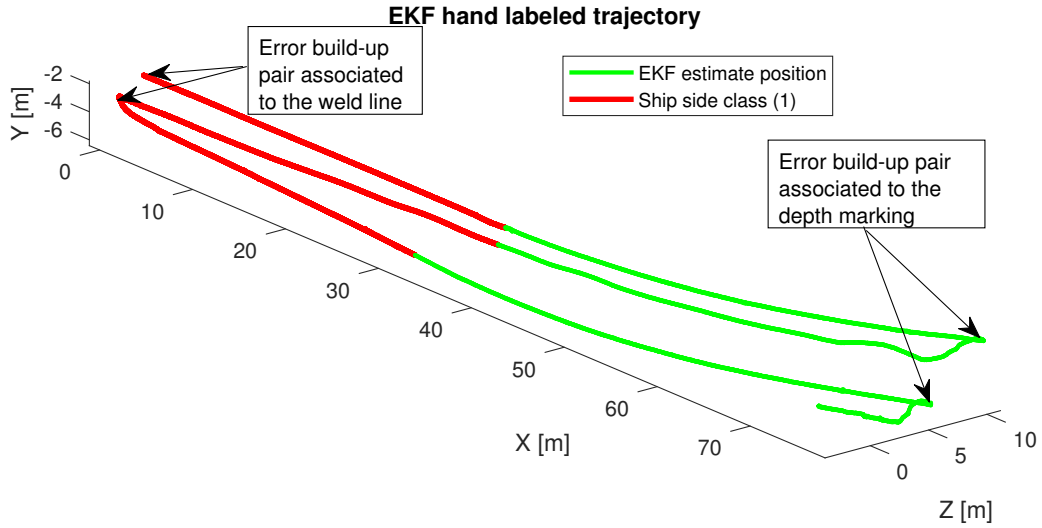


Figure 6-9: Graph of labeled training data to train the kNN classifier, associated to the trajectory in Figure 6-8. The error build up is calculated between the two error build-up pairs on the left and the right of the figure. It is assumed that the $x-$ and $z-$coordinate position in these point pairs is the same.

6-3-2 Feature and classifier parameter tuning

Similar to Section 6-2, the parameter set $(N_1, \sigma_1, N_{delay}, N, k)$ is optimized by evaluating the kNN AUC score for different combinations of classifier and feature parameters. Each AUC evaluation is the average of a 4 fold-cross validation using the tuning data set as depicted in Figure A-9.1. Using the grid search sequences summarized in Table 6-9, the optimal parameter set was found to be $(N_1, \sigma_1, N_{delay}, N, k) = (600, 100, 10, 300, 400)$, scoring an AUC score on the test set of 0.9776.

	N_1	σ_1	N_{delay}	N	k
Grid 1	$10^{[1:3]}$	$10^{[1:3]}$	$10^{[1:3]}$	$10^{[1:3]}$	$10^{[1:3]}$
Grid 2	$100 \cdot [1:5]$	$10 \cdot [1:5]$	$100 \cdot [1:5]$	$100 \cdot [1:5]$	$100 \cdot [1:5]$
Grid 3	$100 \cdot [5:9]$	$50 \cdot [1:5]$	$10 \cdot [1:5]$	$100 \cdot [1:5]$	$100 \cdot [1:5]$

Table 6-9: Table summarizing the grid search ranges for kNN parameter tuning. The optimal parameter set was found to be $(N_1, \sigma_1, N_{delay}, N, k) = (600, 100, 10, 300, 400)$, scoring an average AUC score on the test data set of 0.9776.

Probability threshold and consecutive detection tuning Similar to Section 6-2, four different combinations of the probability threshold $pThresh$ and number of consecutive detections $NrCons$, displayed in Figure 6-10 are selected to be evaluated in combination with the EKF. The different combinations all have a zero false positive rate, but as was shown in Section 6-2, they have a different amount of robustness to false detections of the side of the ship. The threshold parameter combination $(NrCons, pThresh) = (181, 0.78)$ is expected to yield the most true positive detections but may also be most susceptible to false positive detections. On the other hand, the combination $(NrCons, pThresh) = (601, 0.84)$ is expected to yield the lowest amount of false positive detections but may also fail to detect the side of the ship at all. The threshold parameter combinations are evaluated in conjunction with the EKF in the next section. The combination that yields the lowest error build-up score while retaining a zero false positive rate is selected as the most appropriate threshold parameter combination.

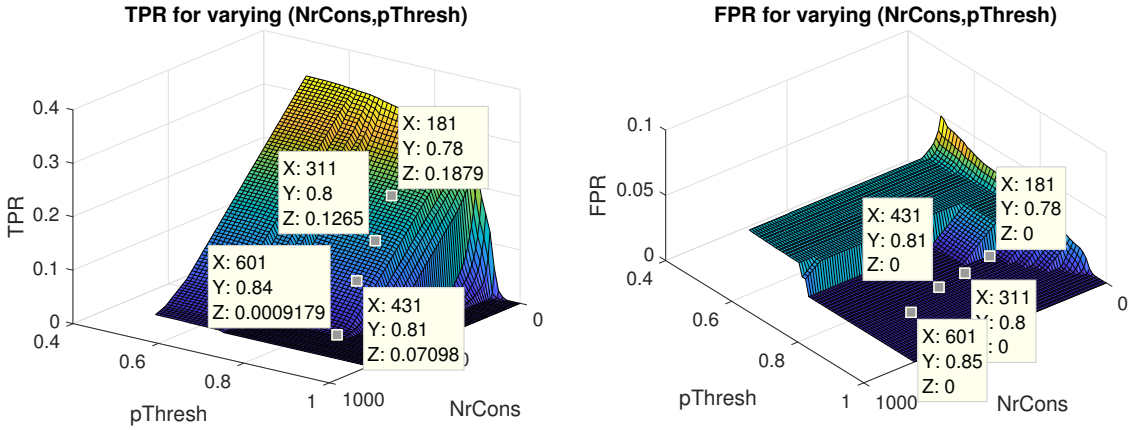


Figure 6-10: Graphs displaying the shifting true positive rate and false positive rate of the kNN classifier for a shifting $pThresh$ and $NrCons$. The four grid points selected all have a zero false positive rates, however, they have different true positive rates. It is expected that the false positive rate parameter set $(NrCons, pThresh) = (601, 0.84)$ will be most robust against false positive detections. However, compared to the parameter set $(NrCons, pThresh) = (181, 0.78)$ it will also be less able to consistently detect the side of the ship, thus increasing error build-up.

6-3-3 Heading corrector performance validation

The performance of the kNN classifier and the four different threshold parameter combinations are evaluated on the nine remaining validation data sets. The data sets are used by the kNN to predict 4×9 different class label prediction sequences, which in turn are used to correct the heading and $z-$coordinate position estimate of the EKF predicted trajectory displayed in Figure 6-9 (Right). For each threshold parameter combination, the error build-up in the position estimate per 'P EBU' pair, is calculated by taking the average over the nine error build-up scores originating from the nine different class label prediction sequences per threshold parameter combination. The average error build-up scores estimate for the four parameter combinations are summarized in Table 6-10. The table shows that the threshold parameter combination $(NrCons, pThresh) = (181, 0.78)$ yields the lowest error build-up while retaining a zero false positive rate. The trajectory generated by the EKF using the heading corrector is displayed in Figure 6-11.

	Error build-up pair 1	Error build-up pair 2	FPR
No heading correction	1.06%	3.75%	-
$(NrCons, pThresh) = (181, 0.78)$	**0.61**%	**0.94**%	0
$(NrCons, pThresh) = (311, 0.8)$	0.61%	0.95%	0
$(NrCons, pThresh) = (431, 0.81)$	0.61%	0.96%	0
$(NrCons, pThresh) = (601, 0.85)$	1.33%	3.77%	0

Table 6-10: Table summarizing the average error build-up of the two point pairs – and the false positive rate using different parameter threshold combinations.

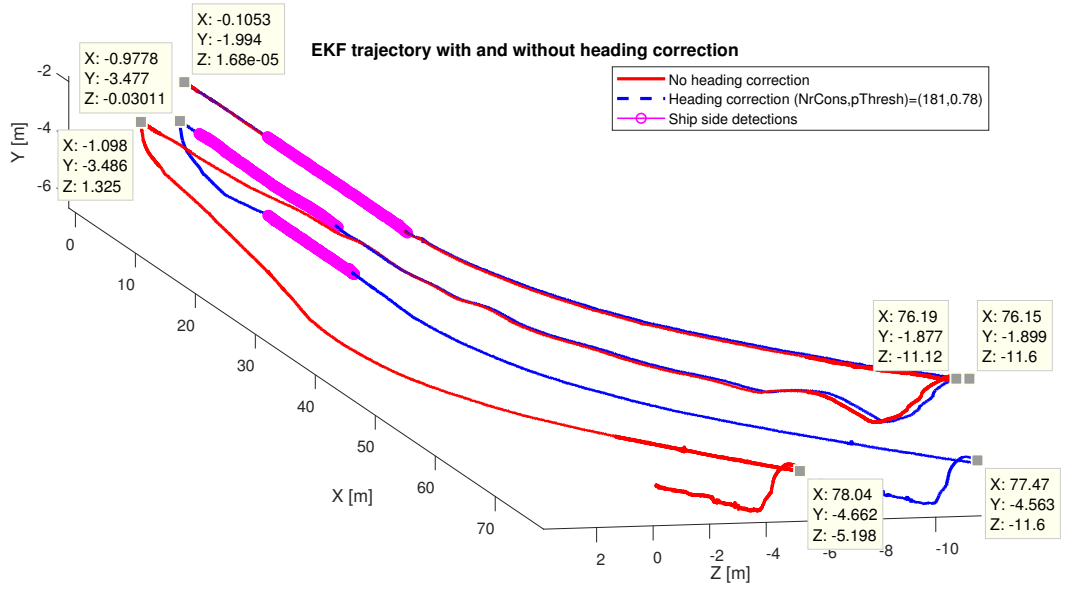


Figure 6-11: Graph of EKF generated trajectory with and without heading correction.

Figure 6-11 shows that the error build-up in the position estimate using the heading corrector is mainly due to the error in the $x-$ coordinate position estimate. The error in the $z-$coordinate position estimate is reduced to $0.00 - 0.03$ $[m]$. Furthermore, since the $z-$coordinate position estimate is reset every time the robot crosses the ship side, the $z-$coordinate position estimate error is bounded. The average error build-up obtained using the most favorable heading corrector settings is 0.77% while the average error build-up without heading corrector is 2.39%.

6-3-4 Overview

The objective of this section was to *verify the simulated results obtained by the EKF in conjunction with the heading corrector, using real sensor data.*

In Table 6-11 the error build-up scores of the constrained IMM with and without heading correction are summarized.

	No HC	Heading corrector
Average error build-up	2.39%	**0.77**%

Table 6-11: Table summarizing average error build-up scores of the constrained IMM without heading correction and with heading correction.

Limitations The error build-up in the $z-$coordinate is not completely bounded, since a correction of the heading and position estimate only occurs when the algorithm detects the side of the ship. While the robot is on the bow, the error in the $z-$coordinate position estimate is still unbounded. Although the results were based on 9 different data sets, the data sets originate from the same cleaning operation. It can thus be argued that ships with larger bow curvatures or if the drift rate is higher, the error build-up will increase. It was assumed that the $x-$ and $z-$coordinate positions of the error build-up pairs, displayed in Figure 6-8 are the same however, in reality they are not. This also limits the reliability of the results.

6-4 Conclusion

The objective of this chapter was to *design a heading correction algorithm that accounts for the heading drift of the IMU output, and reduces the error build-up position estimate.* This objective was achieved by proposing a working principle, and testing this principle using simulated and real data.

A ship side detection algorithm was proposed as a heading corrector. The algorithm corrects the heading and $z-$coordinate position estimate of the EKF when the side of the ship is detected. The main challenge of this detector was to be able to distinguish the side of the ship from the bow, when the ship hull remains vertical from side to bow, which is common for oil tankers – and to maintain a zero false positive rate, such that the resetting of the position estimate does not induce large errors in case of a false ship side detection. For such a hull shape, the extrinsic IMU roll input can not be used to distinguish between the side and the bow of the ship and so a curvature feature was devised that estimates the curvature of the ship hull. By assuming the bow of the ship is curved and the side of the ship is flat, the two ship hull sections can be distinguished. A zero false positive rate was ensured by tuning the probability threshold '$pThresh$' of the kNN classifier output and the number of consecutive ship side detections '$NrCons$'that are required before a data point is labeled as the side of the ship.

Results The simulated results showed that the performance of the heading corrector decreases with an increasing drift rate, but that the false positive rate can be maintained at zero if the kNN thresholds parameter $(NrCons, pThresh)$ are sufficiently high. The simulated results showed that on average, the error build-up can be reduced from 41.66% to 8.10%, assuming the heading drift varies between -0.0037 and 0.0037 $[rad/s]$. Furthermore, the simulated results showed that the heading corrector can eliminate the error build-up induced by

the heading drift up to a drift rate of ± 0.002 $[rad/s]$. Using real sensor data, the EKF in conjunction with the heading corrector achieved an average error build-up of 0.77%, while maintaining a zero false positive rate. Furthermore, the error at the measurement points in the $z-$coordinate position estimate were between 0.03 and 0.00 $[m]$, indicating the the error in the $x-$coordinate position estimate is mostly responsible for the error build-up and that the error build-up in the $z-$coordinate position estimate is bounded. Based on the validation results, it can be seen that the heading corrector not only reduces the error build-up, but also bounds the error of the $z-$coordinate position estimate, which is ultimately required of the localization system to make the robot autonomous.

Limitations The current curvature estimator is limited since it only estimates the curvature locally, and it does not take into account that the ship hull has a fixed shape for each position. This information can potentially be used to increase estimator performance even further.

The resetting method for the heading and the $z-$coordinate position estimate is quite simplistic and it does not update the $x-$coordinate position estimate. Resetting the $z-$coordinate position estimate either increases or decreases the estimated trajectory length over some N past horizon. Assuming the estimated length of the traveled trajectory has zero error, the $x-$coordinate position should thus also increase or decrease, to preserve the estimated trajectory length.

The data that was used to validate the heading corrector performance originated from a cleaning operation on a single ship. The performance of the heading corrector thus has not been validated for ships with different shapes, and surface conditions.

Due to a lack of absolute localization the error build-up of the position estimator in conjunction with the heading corrector could only be evaluated at two locations. It can thus not be stated that the estimated $z-$ coordinate position of the robot remains within 10 $[cm]$ at all times.

Recommendations In order to improve the curvature estimate, the estimator can be made recursive such that it estimates the curvature based on the surface normal change estimator and based on its position, using a previously estimated curvature associated to that position. Essentially, such a recursive curvature estimator assumes that the robot is on a fixed manifold.

Assuming the velocity correction method can achieve zero error between the estimated velocity and the true velocity, the $x-$coordinate position estimate can be improved after the side of the ship has been detected. This can be done by recalculating the robot trajectory over a past horizon, such that the trajectory ends at the reset $z-$coordinate position estimate and heading.

In order to improve robustness and reliability, the heading corrector should be trained and validated on data originating from multiple ships. It is recommended that data is used originating from container vessels and oil tankers, since they differ in bow shape. Also, the smallest and largest $150 - 400$ $[m]$ ships should be included in the data set.

Chapter 7

Closing remarks

In this chapter it will be reviewed to what extent the objectives set at the beginning of the thesis have been met and, insofar they have not been met, what can be done to fulfill them.

Firstly the thesis goal and thesis structure are reviewed. Secondly, the obtained results are shown and discussed and a main conclusion is stated. Finally, the shortcomings of the results with respect to the objectives are used to formulate recommendations.

7-1 Thesis goal

Recall the goal of the thesis, to *reduce the error build-up in the unreferenced* (x, z)*-position estimates, while the robot is underwater and on the side of the ship, without the use of additional hardware*. To achieve the objective the thesis was split in two parts, both of which had their sub-objectives. In order to achieve this goal the requirements listed below had to be achieved.

- Quantitative
  - The localization system may have a maximum error build-up of 2.05%.
  - The localization system must be able to output a position estimate at a frequency of at least 3 $[Hz]$[1].
- Qualitative
  - The position estimator must require a minimal amount of operator resets.
  - The localization system must output an informative map that displays where the robot has cleaned the ship.

[1] Fleet Cleaner states that in order to ultimately make the robot autonomous, an accuracy of within 10 $[cm]$ is required. The robot travels at 0.3 $[m/s]$, and so the update rate needs to be at least 3 $[Hz]$, otherwise the true position of the robot exceeds the 10 $[cm]$ accuracy limit.

Part 1, Analysis

The objective of the first part of the thesis was to *gain understanding into the effect that perturbations and sensor noise have on the error build-up in the position estimate*, such that this knowledge can be used to efficiently reduce the error build-up. This objective was achieved by studying sensor data obtained in real life tests to model realistic sensor noise- and perturbation models and testing these models on a well tuned non-linear filter to evaluate their effect on the position estimate.

Part 2, Implementation

The objective of the second part of the thesis was *to reduce the error build-up in the position estimate by countering the, in Part 1, identified culprits for error build-up.* In Chapter 4, 5 and 5 this was achieved using the following design structure.

Conceptual design Conceptual design specifies the principal solution to the sub-problems. For each respective chapter this meant that the objective was *to find- and establish a working principle to reduce the error build-up.* This objective was achieved through the review of literature relevant to the identified error build-up sources.

Simulation The objective of the simulation part of each chapter is to *evaluate the performance of the working principles with respect to a ground truth and to identify practical boundaries of the working principles.* Working principles are extensively tested by numerous simulations using the sensor noise- and perturbation models designed in Part 1. This ensured the performance of the working principles under varying sensor noise- and perturbation conditions.

Validation The validation part of the implementation chapters had as objective to *validate the simulated performance of the working principles using real sensor data.* The validation of the working principles was problematic due to the unavailability of a true reference. Validation was achieved using ad hoc measures that made use of existing knowledge of environment conditions.

7-2 Results and conclusions

Part 1, Analysis

7-2-1 Sensor data modeling

Results and Conclusions The static analysis of the sensor data revealed that all the sensor data sequences listed in Table 1-1, except the gyroscope and depth meter, are non-Gaussian distributed. Furthermore, all sensors except the gyroscope output of the IMU, have a non-white noise spectrum. Camera observations during operation on the HS Tosca on the 7^{th} of

July, 2017 revealed that the front wheel, to which the wheel encoder is attached, suffers from wheel slip which increases error build-up in the position estimate. Using the measured sensor noise PSDs and external perturbation characteristics, models were generated that simulate the correlated sensor noise and emulate the wheel slip and heading drift perturbations.

Limitations Only a limited amount of data was available for the evaluation of the sensor noise PSDs, so the modeling was done using a simplistic spectrum fitting method. The external perturbation characteristics of the wheel slip were determined using by cross-referencing camera footage to real sensor data, which limits the accuracy with which characteristics can be determined and modeled.

7-2-2 Non-linear filters

The simulation of the robot trajectory with sensor inputs corrupted by the noise models and perturbation models revealed the individual simulated error build-up values summarized in Table 7-1 and the error build-up values using real data, shown in Table 7-2.

	None	Noise	Wheel slip	Heading drift	All
Error build-up	0.19%	0.99%	**46.02**%	6.19%	54.72%

Table 7-1: Table summarizing the error build-up scores between the estimated position and the simulated ground truth for various sources of error build-up.

Data set →	1	2	3	4	5	Mean
Error build-up	54.55%	57.14%	33.85%	14.04%	2.52%	32.42%

Table 7-2: Error build-up scores between the EKF estimated and true position.

The error build-up values summarized in Table 7-1 led to the conclusion that the best way to reduce the error build-up in the position estimate is to account for the slip perturbation in the wheel encoder output and account for the drift in the heading direction of the IMU.

Part 2, Implementation

Wheel slip detection

The detection of wheel slip is achieved by classifying sensor input data and labeling it $\{1, 0\}$, representing the slip label and non-slip label respectively. In order to reduce the error build-up to 2.05% a label prediction accuracy of at least 98.11% was required. Due to the cumbersome method that had to be employed to hand label the training data, only a limited amount of labeled training data was available. This merited the comparison of three classifiers that had required different amounts of supervision and adaptability. The three classification methods that were compared are:

1. SVM classifier. This method is supervised with no online adaptive capabilities.

2. Threshold classifier. This method is most commonly used in literature. This method is supervised and can be adjusted online by the operator.

3. K-means classifier. This is an unsupervised method which can be made adaptable.

Results and Conclusions The simulated results revealed that the SVM classifier achieves the highest prediction accuracy under nominal conditions (96.05%) and maintains the highest prediction accuracy over a wide range of simulated ship hull conditions compared to the other classifiers. The simulations also reveal that if the SVM is trained on a limited data set, it will not be able to maintain a high prediction accuracy when faced with the expected variations in ship hull surface conditions. The simulated performance results were validated using real hand labeled data, summarized in Table 7-3.

	SVM	Threshold	K-means
Label prediction accuracy	**96.85**%	94.06%	84.97%

Table 7-3: Table summarizing the slip detection classifier prediction accuracy results on real data.

Limitations The validation results are limited since the classifiers were trained and validated on data originating from the same cleaning operation. Furthermore, the data was labeled using camera footage, which is prone to mislabeling.

Velocity correction

A constrained IMM was proposed that estimates the true velocity with a weighted average between two velocity models. This negates the possibility of error accumulation in the velocity estimate due to integration of the IMU acceleration.

Results and Conclusions Simulated results reveal that the constrained IMM can reduce the error build-up from 47.24% to 3.71% under nominal conditions. However, when faced model mismatch the error build-up go as high as 32.84%. Using real sensor data it was shown the constrained IMM reduces the average error build-up by a factor of four to 8.88%, as shown in Table 7-4. However, it cannot be assumed the constrained IMM always attains this performance since an error build-up score of 15.97% was measured on a single data set.

Data set	No correction	SVMIMM	Cons. IMM
Average error build-up	32.42%	31.89%	**8.88**%

Table 7-4: Table summarizing error build up in position estimates per true meter traveled using various velocity estimation methods.

Limitations The simulated and validation results show that the constrained IMM is a significant improvement over both the EKF without velocity correction and the SVMIMM. However, the simulated results also show that even if a perfect slip detector was available, the constrained IMM would not reduce the error build-up to within 2.05% under varying ship hull conditions. Both the evaluation of the performance of the slip detector and velocity corrector reveal that performance significantly decreases with respect to a varying model mismatch.

Heading correction

The heading orientation and $z-$coordinate estimate are corrected by detecting the flat side of the ship hull using the kNN as a classification algorithm and resetting the heading orientation and $z-$coordinate to their respective initial conditions.

Results and Conclusions The simulated results showed that using the heading corrector, the error build-up score can be brought back 3.64% for a heading drift rate between ± 0.002 $[rad/s]$. Furthermore, the simulated results showed that a false positive rate per bow crossing below 0.0005 can be maintained for ship hull curvatures up to at least 48 $[m]$ and a heading drift rate between ± 0.0037 $[rad/s]$, which is important since a false detection would severely upset the position estimator and the cleaning trajectory presented to the client. Using real data, the addition of the heading correction improved the error build-up by a factor as shown in Table 7-5.

	No heading correction	EKF with heading correction
Average error build-up	2.39%	**0.77**%

Table 7-5: Table summarizing the average error build-up scores for the EKF with and without heading correction.

Limitations Although the results are promising, data was used from a single cleaning operation from a particularly clean ship, thus limiting the reliability of the results.

Main conclusion

The desired error build-up score of 2.05% could not be achieved since the available training data is too limited for the SVM slip detector and velocity correction models utilized by the Constrained IMM to maintain a high enough prediction with respect to all the possible varying ship hull conditions. Even though the set objective was not satisfied, the addition of the SVM slip detector and constrained IMM velocity corrector did reduce the error build-up from 32.42%, to an average of 8.88%, significantly reducing the workload on the operator.

The addition of the heading corrector reduced the error build-up of the EKF in conjunction with the constrained IMM from 2.39% to 0.77%. The heading corrector mitigates the need for intermittent resetting of the heading orientation by the operator, thus also reducing the workload on the operator.

Even though the positioning error was reduced, it is still unbounded, making the current position estimator not suitable for autonomous operation of the robot.

7-3 Recommendations

The results obtained in this thesis partially complete the set objectives of bounding the error build-up in the position estimate. To further reduce the error build-up in the position estimate, or to remove error build-up completely, the following procedures – that fall within and outside the thesis scope – are recommended.

7-3-1 Within thesis scope

Adaptive model predicted velocity Both the evaluation of the performance of the slip detector and velocity corrector reveal that performance significantly decreases with respect to a varying model mismatch. The SVM slip detector and constrained IMM performance can thus be improved by accounting for the model mismatch, for instance by making the velocity prediction model adaptive.

kNN training data Although it has been shown that the implementation of a heading corrector is feasible and beneficial, it is recommended that data from more ships is included in the classifier training data. This will ensure that the heading corrector can be employed on different ship hull shapes.

7-3-2 Outside thesis scope

Additional wheel encoders The current setup of the robot only allows slip detection using the sensor input data of the front wheel. Mobile robots comparable to the Fleet Cleaner robots often use wheel encoders on all wheels, which greatly simplifies the detection of wheel slip. Furthermore, adding additional wheel slip allows for more easy hand labeling of slip data and slip model identification.

Training data acquisition The slip detection SVM, the slip mode predicting SVM and the dynamic slip model were trained using a limited hand labeled data set. Using the added wheel encoders in the previous recommendation, the labeling of slip data can be made more easy. For instance, if the power to one wheel is cut, it simply tracks the movement of the robot. Using this data, the classifiers can be better trained and a non-linear model can be trained to estimate the true velocity during dynamic slip.

Feature analysis The extended data set that is provided by the previous recommendations will also allow for the uncovering of more features that can be used by the SVM to detect slip. Adding more features can make the slip detector label prediction accuracy more robust against varying ship hull conditions.

Appendix A

Appendix

A-1 Sensor noise distribution

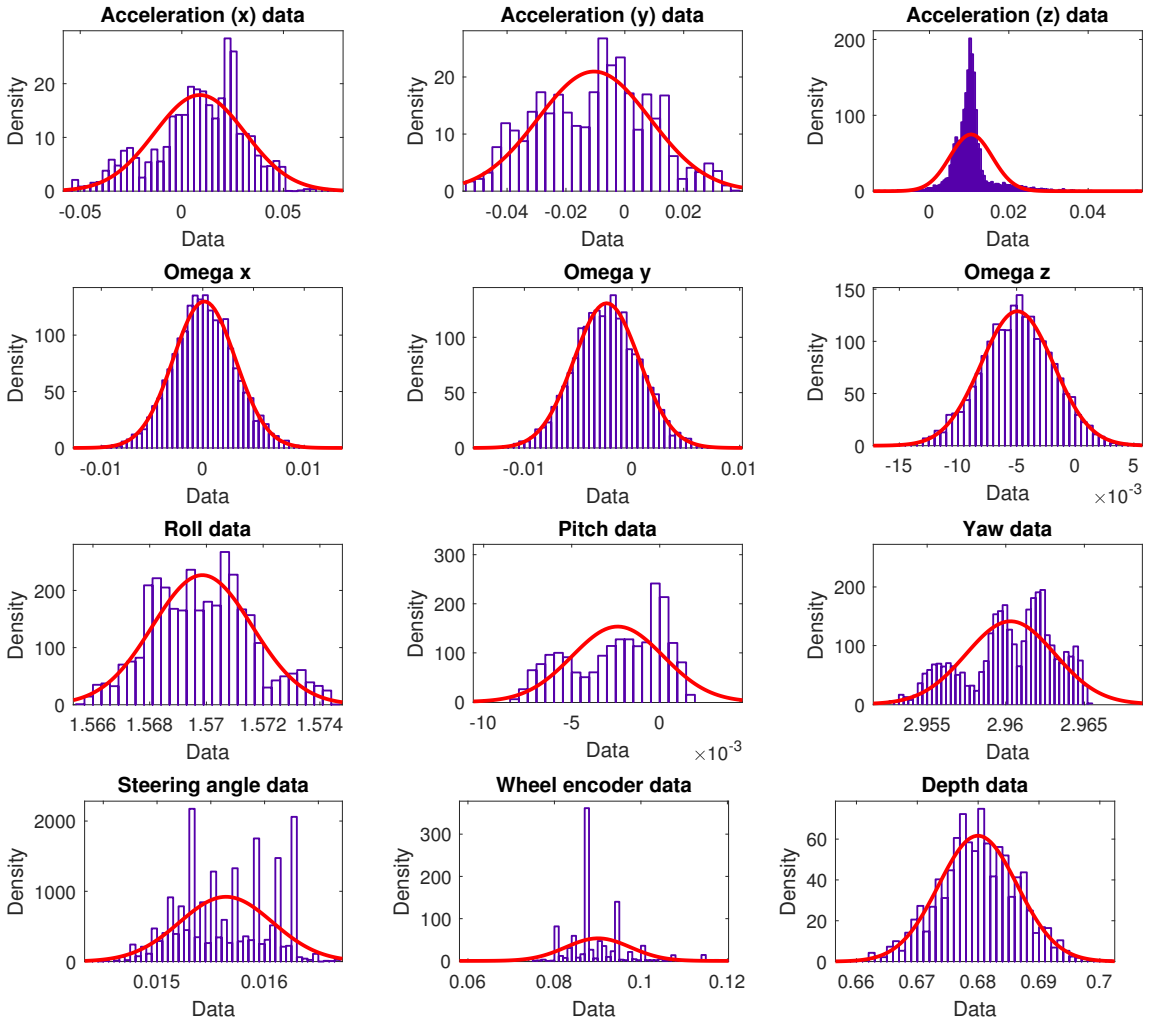


Figure A-1.1: Graphs of sensor data distributions with normal distribution fitted to the data. The data was obtained in a stationary situation such that the distribution of the sensor noise can be probed.

A-2 Auto-correlation functions

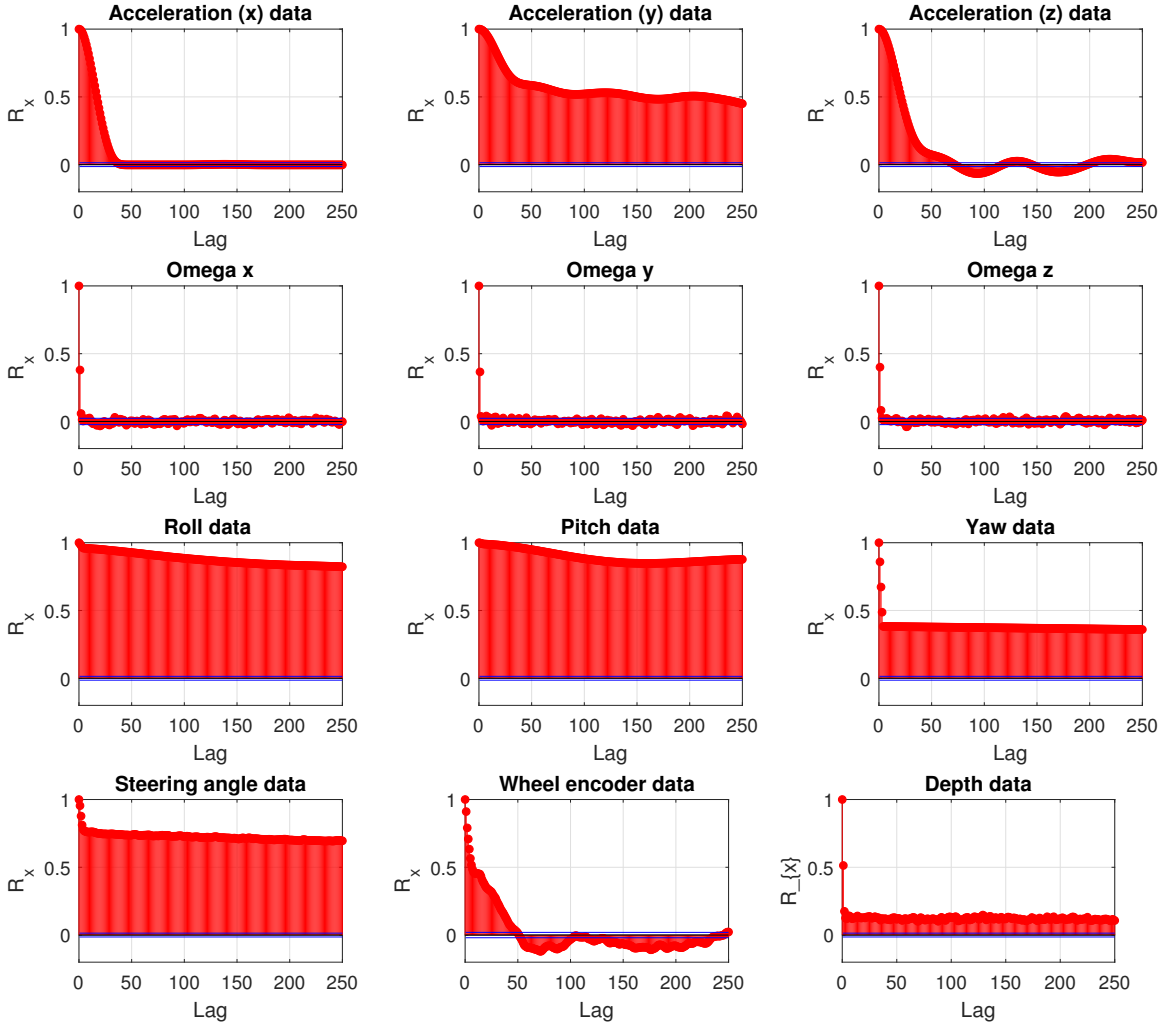


Figure A-2.1: Auto-correlation graphs of sensors used for position estimation. The data was obtained in stationary situations such that the stationary noise can be evaluated. The figures show that only the samples of the rotational velocity data, $\omega_x, \omega_y, \omega_z$, have little auto-correlation. The stationary noise associated to the rotational velocities can thus be accurately approximated with white noise.

A-3 Sensor noise modeling example

The ATM 1st-N depth meter is employed by Fleet Cleaner to gauge the water depth of the robot and will be used as an explanatory case for designing a sensor noise model. The data that is used to further evaluate the sensor noise characteristics is displayed in Figure A-3.1. This particular data sequence is selected because it was recorded in very rough water conditions while the robot was only 65 $[cm]$ deep in the water. Because of that, it is expected that in most situations the magnitude of the noise induced by waves will be less than in the data sequence displayed in Figure A-3.1.

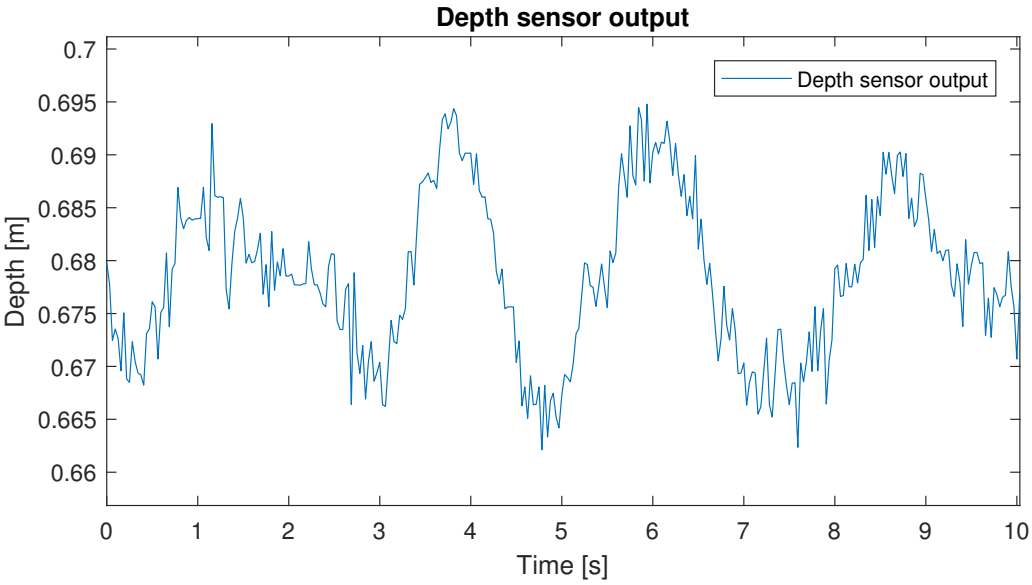


Figure A-3.1: Sensor output of depth meter in a static situation. The data was obtained during a test on the HS Tosca.

The PSD of the depth meter data, displayed in Figure A-3.2, was obtained using the *pwelch*() [28] function in Matlab. As was expected from the sensor analysis conducted in Section 2-1, the PSD is non-white.

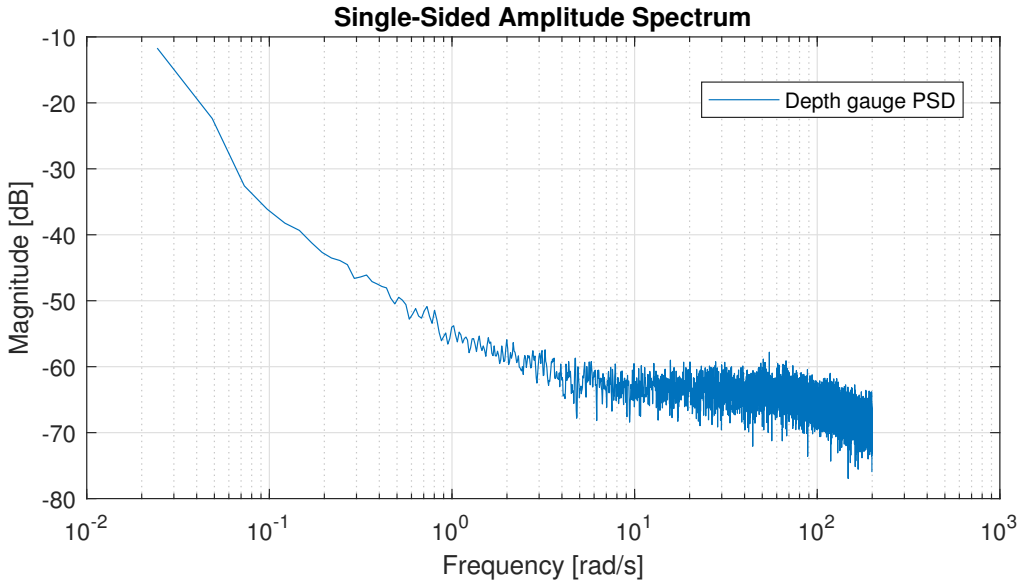


Figure A-3.2: Power spectral density of the depth meter data sequence. In the range of 10^{-1} to $3 \cdot 10^{0}$ $[\frac{rad}{s}]$ the magnitude of the PSD decreases with approximately $-20[\frac{dB}{dec}]$. In the range of $8 \cdot 10^{0}$ to $6 \cdot 10^{1}$ $[\frac{rad}{s}]$ the PSD has a slope of $0[\frac{dB}{dec}]$ and after that it again decreases to about $-20[\frac{dB}{dec}]$.

A TF resembling the depth sensor data PSD is obtained by estimating from Figure A-3.2

that there is a zero at $\omega = 3[\frac{rad}{s}]$ and a pole at $\omega = 80[\frac{rad}{s}]$ and that the TF has one pure integrator. In order to limit the amount of drift in the signal due to pure integration, a pole is used instead of the integrator at $\omega = 0.5[\frac{rad}{s}]$. This pole eliminates the integrator drift, but preserves the oscillations induced by waves which have a frequency of approximately 2 $[rad/s]$. In order to match the magnitude of the real signal, the simulated signal is scaled with a factor of 0.08, obtaining the continuous time TF

$$H_{depth}(s) = 0.08 \cdot \frac{(\frac{1}{30}s + 1)}{(2s + 1)(\frac{1}{80}s + 1)}. \tag{A-1}$$

In order to simulate the sensor noise, the continuous-time TFs need to be discretized. The TFs are discretized using the zero order hold discretization method with the $c2d()$ function in Matlab. The sensor data is simulated by inputting a unit variance WGN sequence into the transfer function H_{depth}, shown in Figure A-3.3. From the figure it is determined that the simulated PSD approximates the real PSD.

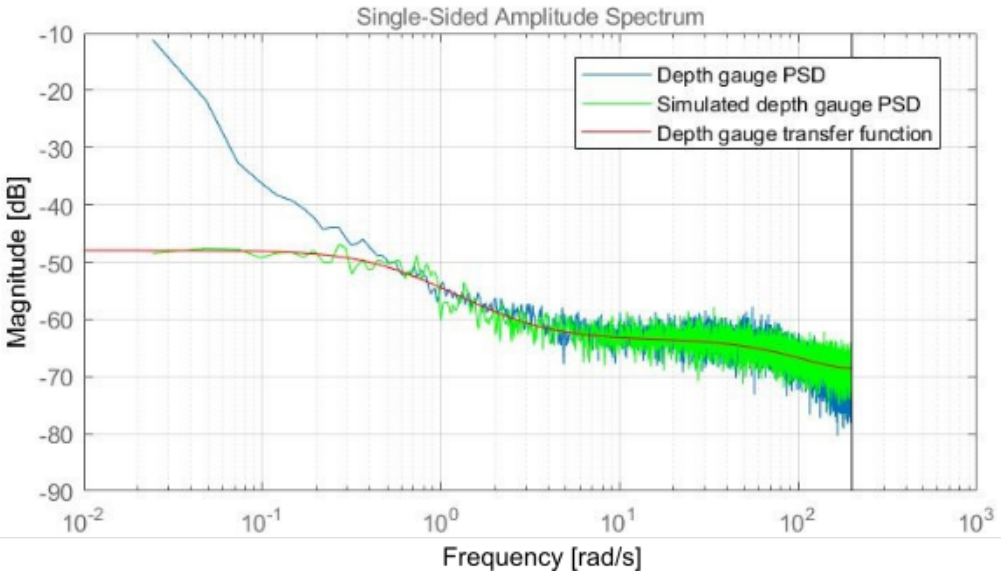


Figure A-3.3: Graph of depth meter data sequence PSD (blue), with $H_{depth}(s)$ bode plot (red) and simulated noise PSD (green). The simulated PSD approximates the real PSD in the frequency range of $[5 \cdot 10^{-1} \quad 2 \cdot 10^{2}]$ $[rad/s]$. This ensures that the simulated PSD includes the wave oscillations at 1.8 $[rad/s]$ and excludes the integration drift.

A-4 Simulated ground truth and sensor model schematic overview

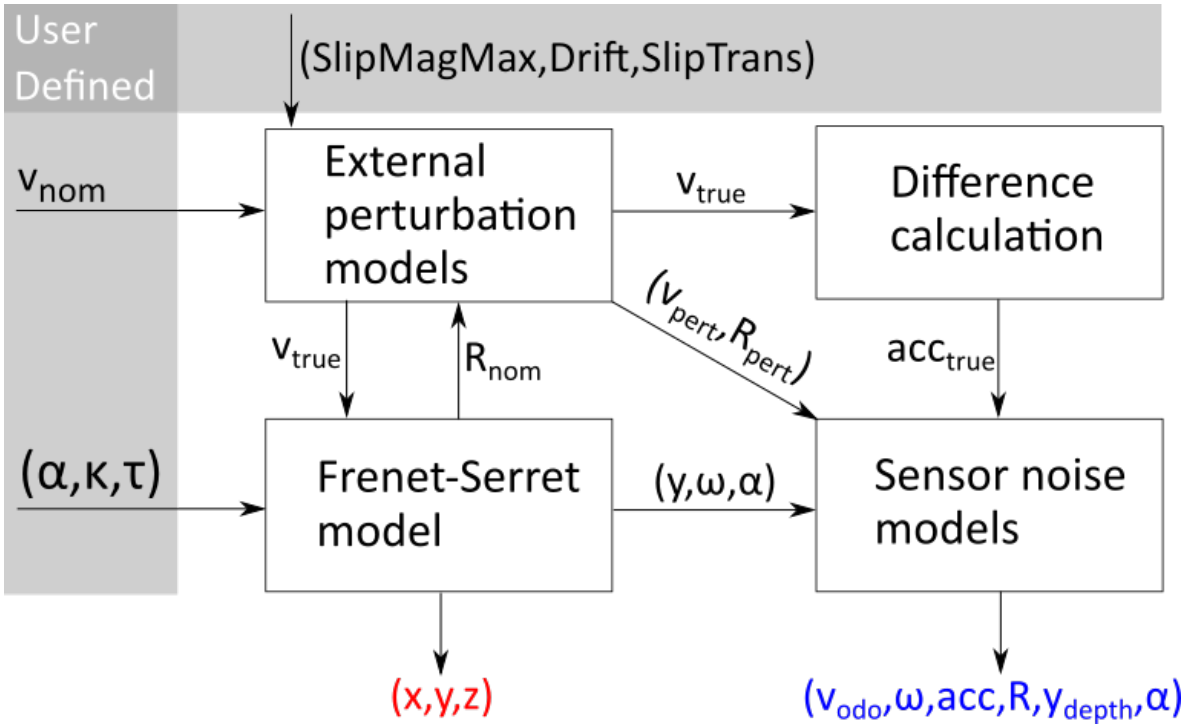


Figure A-4.1: Schematic overview of simulated ground truth and sensor signal generator. The control inputs, external perturbation parameters and Frenet-Serret parameters are all user defined. The output (red) of the Frenet-Serret model is the simulated ground truth and the output of the sensor noise models (blue) are used by the position estimator to estimate the true position.

A-5 Extended Kalman filter

The EKF is an extension of the KF such that it is applicable to non-linear models. In essence the EKF functions the same as the KF, with an added linearization step in each iteration. A derivation of the EKF can be found in [17, p.59].

EKF assumptions Suppose the kinematics of a robot are represented by the continuous time non-linear model

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= g(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{y}(t) &= h(\boldsymbol{x}(t), \boldsymbol{u}(t)), \end{aligned} \tag{A-2}$$

where $\boldsymbol{x}(t)$ represents the states, $\boldsymbol{u}(t)$ the input, $g(\boldsymbol{x}(t), \boldsymbol{u}(t))$ the non-linear state transition function and $h(\boldsymbol{x}(t), \boldsymbol{u}(t))$ the non-linear model output. The non-linear state-transition function of the robot $g(\boldsymbol{x}(t), \boldsymbol{u}(t))$ can be approximated by a first order Taylor-expansion and

evaluated at $\boldsymbol{u}_t$ and $\boldsymbol{x}_{t-1}$ to obtain the linear system matrices,

$$\begin{aligned} G_t &= \frac{\partial g(\boldsymbol{x}(t), \boldsymbol{u}(t))}{\partial \boldsymbol{x}(t)}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t), \\ F_t &= \frac{\partial g(\boldsymbol{x}(t), \boldsymbol{u}(t))}{\partial \boldsymbol{u}(t)}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t), \end{aligned} \tag{A-3}$$

where $\boldsymbol{u}_t$ is the control input and $\boldsymbol{x}_{t-1}$ the mean of the state PDF of the previous iteration. The non-linear model output $h(\boldsymbol{x}(t), \boldsymbol{u}(t))$ is approximated by a first order Taylor-expansion and is evaluated at $\bar{\boldsymbol{x}}_t$, which yields

$$H_t = \frac{\partial h(\boldsymbol{x}(t), \boldsymbol{u}(t))}{\partial \boldsymbol{x}(t)}(\bar{\boldsymbol{x}}_t, \boldsymbol{u}_t), \tag{A-4}$$

where $\bar{\mu}_t$ is the state transition function $g(x(t), u(t))$ evaluated at the posterior mean μ_{t-1} and current input u_t. The non-linear model in Equation (A-2) can now be approximated by

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= G_t(\boldsymbol{x}(t) - \boldsymbol{x}_{t-1}) + F_t(\boldsymbol{u}(t) - \boldsymbol{u}_t) \\ \boldsymbol{y}(t) &= H_t(\boldsymbol{x}(t) - \bar{\boldsymbol{x}}_t). \end{aligned} \tag{A-5}$$

EKF algorithm workings The EKF algorithm is depicted in Algorithm 2. The EKF algorithm has as inputs the mean of the states of the previous time step $\boldsymbol{x}_{t-1}$, its corresponding covariance matrix Σ_{t-1}, the current time step control input $\boldsymbol{u}_t$ and the sensor measurement $\boldsymbol{y}_t$. In **step 2**, the non-linear state transition model from Equation maps $\boldsymbol{x}_t$ and Σ_t into the *prior distribution*, where the notation $\bar{\boldsymbol{x}}_t$ indicates that the (in this case) mean has not been updated by a measurement yet. In **step 3** and **4** the non-linear state transition model is linearized around the current state and control input. In **step 5** the error covariance matrix is updated using the linearized model, which is used in **step 6** to calculate the Kalman gain. The prior distribution is then fused with the sensor measurements $\boldsymbol{y}_t$ in **step 7**, called the *innovation step*, followed by an update of the error covariance matrix in **step 8**.

Algorithm 2 Extended Kalman filter

1: **function** EKF($\boldsymbol{x}_{t-1}$,Σ_{t-1},$\boldsymbol{u}_t$,$\boldsymbol{y}_t$)
2: $\quad \bar{\boldsymbol{x}}_t = g(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$
3: $\quad G_t = \frac{\partial g(\boldsymbol{x}(t), \boldsymbol{u}(t))}{\partial \boldsymbol{x}(t)}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$
4: $\quad H_t = \frac{\partial h(\boldsymbol{x}(t), \boldsymbol{u}(t))}{\partial \boldsymbol{x}(t)}(\bar{\boldsymbol{x}}_t, \boldsymbol{u}_t)$
5: $\quad \bar{\Sigma}_t = G_t \Sigma_{t-1} {G_t}^T + Q_t$
6: $\quad K_t = \bar{\Sigma}_t {H_t}^T (H_t \bar{\Sigma}_t {H_t}^T + R_t)^{-1}$
7: $\quad \boldsymbol{x}_t = \bar{\boldsymbol{x}}_t + K_t(\underbrace{\boldsymbol{y}_t - h(\bar{\boldsymbol{x}}_t)}_{innovation})$
8: $\quad \Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
9: $\quad$ **return** $\boldsymbol{x}_t$,Σ_t

A-6 Fusion schemes

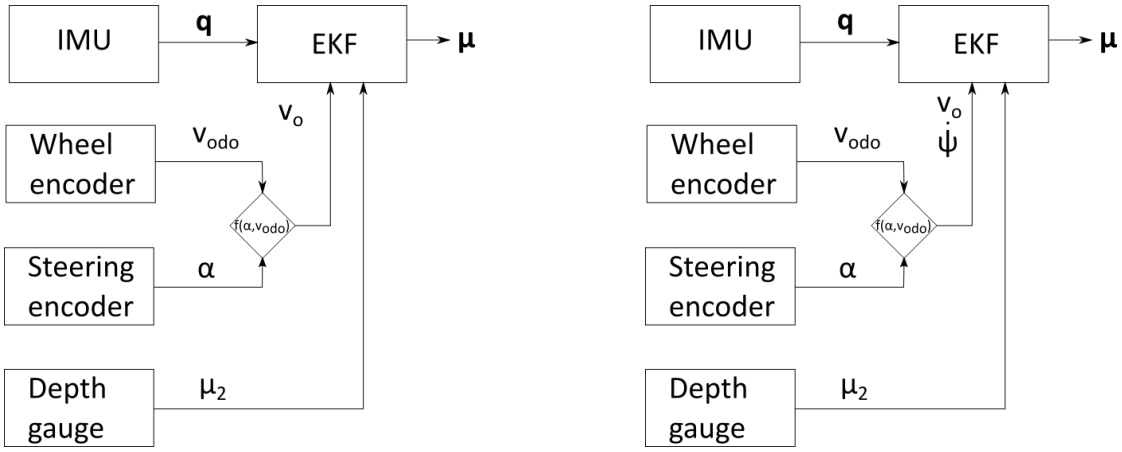


Figure A-6.1: Illustration displaying Fusion schemes 1 (Left) and 2 (Right). Fusion scheme 1 is the scheme currently employed by the Fleet Cleaner robot. Fusion scheme 2 is obtained by fusing the model predicted rotational velocity with the IMU orientation estimate, proposed in [50].

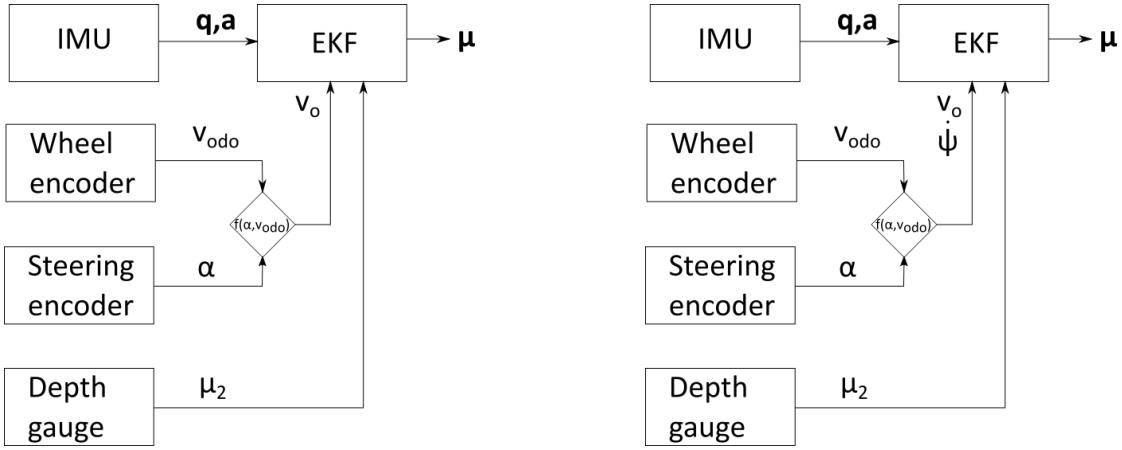


Figure A-6.2: Illustration displaying Fusion schemes 3 (Left) and 4 (Right). Fusion scheme 3 is obtained from fusion scheme 1 by additionally fusing the IMU acceleration with the wheel encoder velocity. Fusion scheme 4 is a combination of fusion scheme 2 and 3.

A-7 EKF position estimation versus simulated ground truth

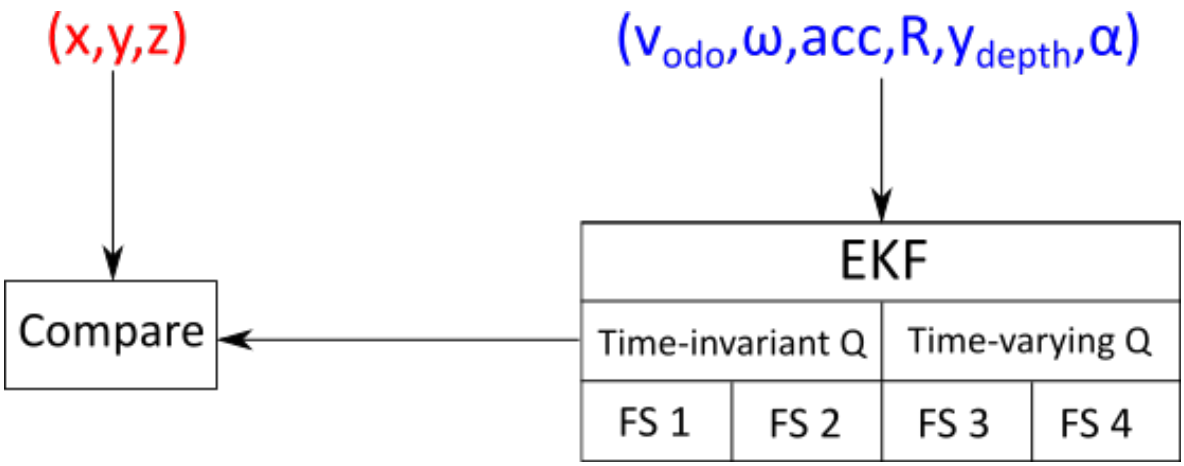


Figure A-7.1: Illustration of the different possible set-ups for the EKF and how it is compared to the simulated ground truth. Note that the red and blue inputs at the top are an extension of Figure A-4.1.

A-8 ARX tuning schematic overview

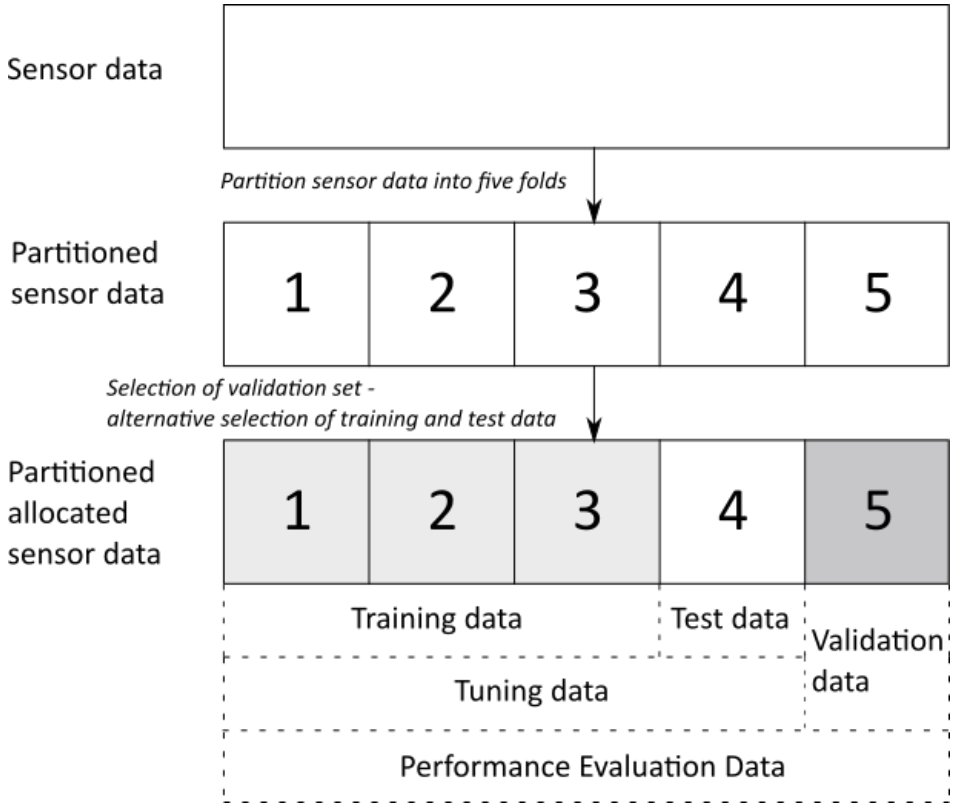


Figure A-8.1: Image illustrating the data preparation and allocation of ARX model training data. Firstly, the data is partitioned into five folds, leaving out one for validation. Secondly, the ARX model parameters are tuned 4-fold on the remaining tuning data set. Finally, the performance of the tuned ARX is evaluated on the separate validation data set.

A-9 Classifier tuning data allocation

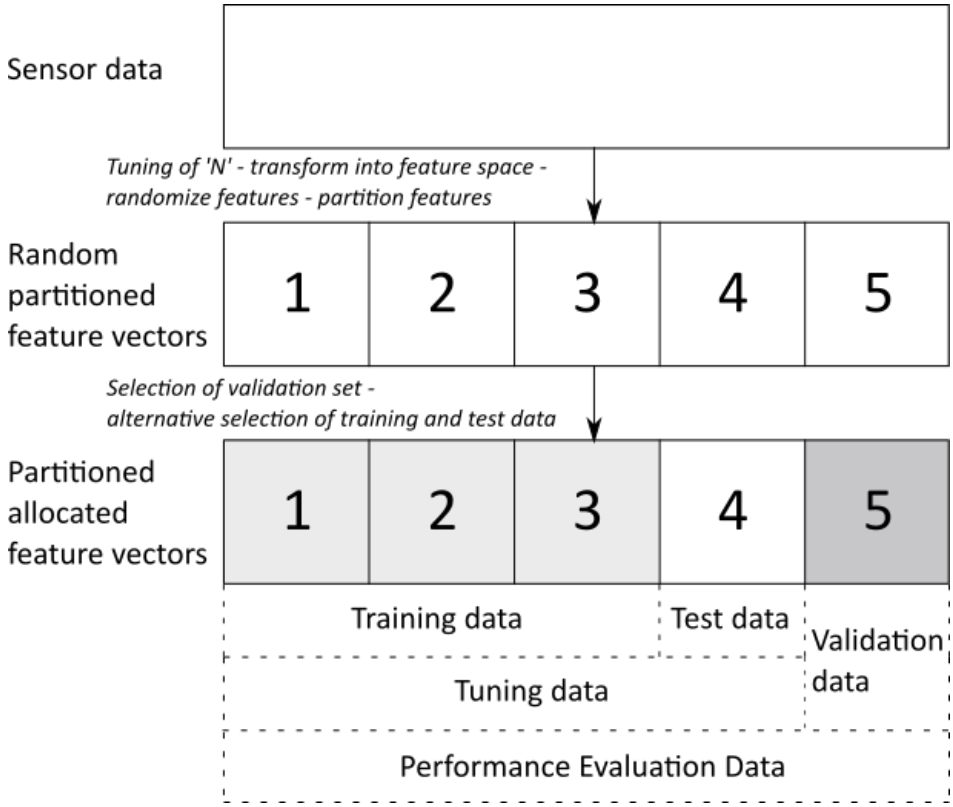


Figure A-9.1: Schematic overview of classifier data design and usage. In the first step the original sensor data is used to train the moving window length 'N' for the variance features. After the feature parameters have been tuned the sensor data is transformed into feature space. The order of the features is then randomized and the entire data set is partitioned, yielding the random partitioned feature vectors. A selection is then made among the five data sets, partitioning it into a training set, test set and validation set. The classifiers are trained using the training set and tuned by evaluating the trained classifier on the test set so that the optimal classifier parameters can be found. The performance of the classifiers is then evaluated on the validation set.

A-10 Velocity correction methods

A-10-1 SVMIMM

Let $S(t)$ be the complete set of available sensor inputs. From this set, the wheel encoder output v_t^{odo}, and gyroscope output ω_t^{gyro} together with the model predicted velocity v_t^{model} and rotational velocity ω_t^{model} are mapped to standardized feature space in **step 2**, yielding the feature vector ζ_t. In **step 3** the class likelihood of the slip and no slip class conditioned on the input feature is calculated using the SVM classifier. In **step 4 - 5** the states and covariance matrices are predicted using an EKF with the acceleration data as control input and the wheel encoder output as a measurement output – and an EKF with only the acceleration data as control input. In **step 6-7** a weighted average between the EKF models predicted states and covariance matrices is calculated using the class likelihood $p(\zeta_t)$.

Algorithm 3 SVMIMM

1: **function** SVMIMM($S_t, \hat{x}_{t-1}, P_{t-1}$)
2: $\zeta_t = \Phi(v_t^{odo}, v_t^{model}, \omega_t^{gyro}, \omega_t^{model})$
3: $p(\zeta_t) = SVM(\zeta_t)$
4: $x_1, P_1 = EKF(S_t)$
5: $x_2, P_2 = EKF_{IMU}(S_t)$
6: $\hat{x}_t = p(\zeta_t)x_1 + (1 - p(\zeta_t))x_2$
7: $P_t = p(\zeta_t)P_1 + (1 - p(\zeta_t))P_2$
8: **return** $\hat{x}_t, P_t$

A-10-2 Constrained IMM

Let $S(t)$ be the complete set of available sensor inputs. From this set, the wheel encoder output v_t^{odo}, and gyroscope output ω_t^{gyro} together with the model predicted velocity v_t^{model} and rotational velocity ω_t^{model} are mapped to standardized feature space in **step 2**, yielding the feature vector ζ_t. In **step 3** the class label is predicted using the SVM classifier with ζ_t as input. If the class label is 0, meaning no slip is detected, the velocity input to the EKF is set to equal v_t^{odo}. If the class label is 1, meaning slip is detected, the constrained IMM is used to correct the velocity. In **step 7** the wheel encoder output v_t^{odo} and model predicted velocity v_t^{model} are mapped to standardized feature space, yielding the feature ξ_t. The posterior class likelihood, conditioned on the input feature is calculated using the mode probability prediction SVM in **step 8**. In **step 9** the corrected velocity are calculated by taking a weighted average between the stationary slip ($\hat{v} = 0$) and dynamic slip model output $v_t^{DynSlip}$, using the class likelihood as a weight. The variance calculated in **step 10** is used in the sensor noise matrix R of the EKF.

Algorithm 4 Constrained IMM

function ConsIMM(S_t)
  $\zeta_t = \Phi(v_t^{odo}, v_t^{model}, \omega_t^{gyro}, \omega_t^{model})$
  $delta_t = SVM(\zeta_t)$
  if $delta_t = 0$ **then**
    $\hat{v}_t = v_t^{odo}$
  if $delta_t = 1$ **then**
    $\xi = \Phi_2(v_t^{odo}, v_t^{model})$
    $p_2(\xi_t) = SVM_2(\xi_t)$
    $\hat{v}_t = v_t^{DynSlip} p_2(\zeta_t)$
    $\hat{\Sigma}_t = 7.056 \cdot 10^{-5} \cdot p_2(\zeta_t)$
  return $\hat{v}_t, \hat{\Sigma}_t$

A-11 Dynamic slip ARX model identification

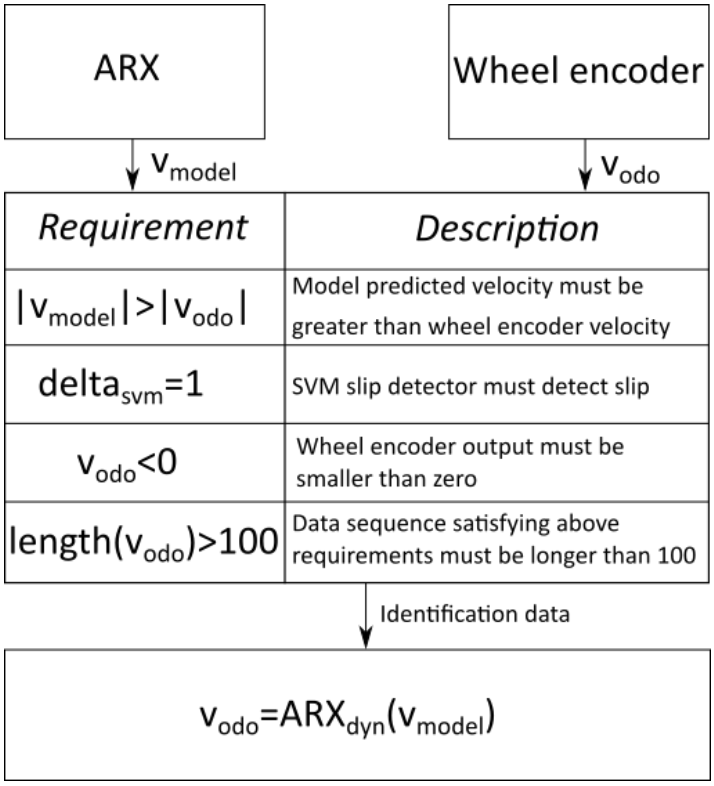


Figure A-11.1: Illustration of dynamic slip model identification procedure. Firstly, data from five different cleaning operations is selected based on the four requirements stated in the figure. The data that satisfies all four requirements is used to identify an ARX model between the input v_{model} and the output v_{odo}. This ARX model is used to estimate the true velocity when dynamic wheel slip is detected.

A-12 Constrained IMM schematic overview

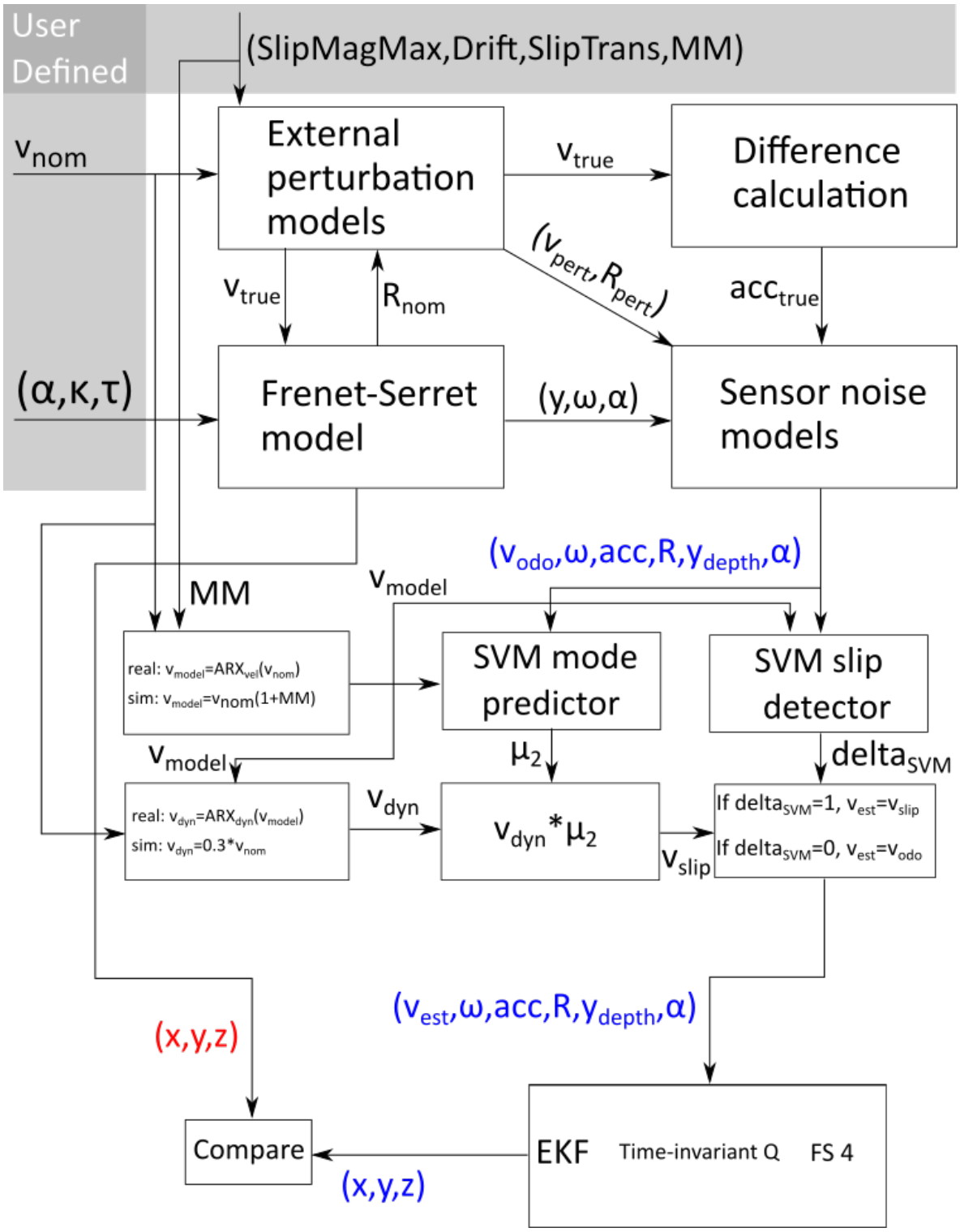


Figure A-12.1: Schematic overview of simulated environment for the constrained IMM. The figure has two blocks where the output depends upon whether real or simulated data is used. If real data is used, the ARX models designed in this thesis are used.

A-13 SVMIMM schematic overview

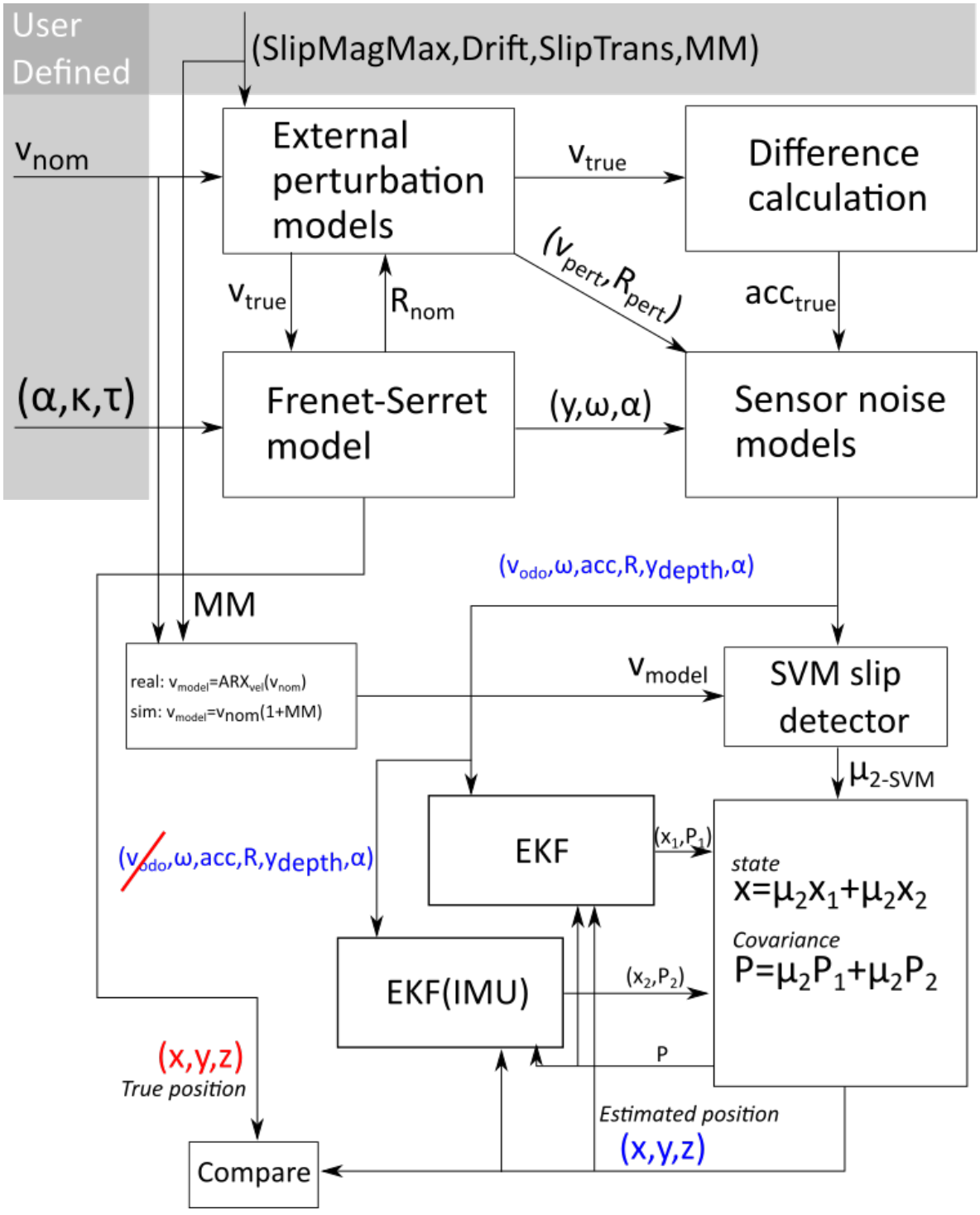


Figure A-13.1: Schematic overview of simulated environment for the SVMIMM.

A-14 Heading corrector algorithm

The heading corrector has as an input the sensor data $S(k)$, Constrained IMM estimated velocity $\hat{v}(k)$, $z-$coordinate position estimate $\mu_3(k)$, orientation correction quaternion $\boldsymbol{q}_{cor}(k-1)$ and estimated quaternions $\hat{\boldsymbol{q}}(k-1)$. In **step 2** the kNN classifier is used to label the input data as the side of the ship (1) or not the side of the ship (0). In **step 3-5**, if the ship side is not detected, the $z-$position coordinate update is set to the current position estimate for the $z-$position, $\mu_3(k)$ and the orientation correction quaternion $\boldsymbol{q}_{cor}(k-1)$ is left unchanged. In **step 6-8**, if the side of the ship is detected, the $z-$position coordinate update is set to $z_0 = 0$ and the orientation correction quaternion is updated. In **step 9** the quaternion is updated using the orientation correction quaternion.

Algorithm 5 Heading Corrector

1: **function** HC($S(k)$,$\hat{v}(k)$,$\mu_3(k)$,$\hat{\boldsymbol{q}}_{cor}(k-1)$,$\boldsymbol{q}(k-1)$)
2: $delta_{HC}(k) = kNN(\hat{v}(k), T(k), N(k), B(k))$
3: **if** $delta_{HC}(k) = 0$ **then**
4: $z(k) = \mu_3(k)$
5: $\boldsymbol{q}_{cor}(k) = \boldsymbol{q}_{cor}(k-1)$
6: **if** $delta_{HC}(k) = 1$ **then**
7: $z(k) = 0$
8: $\boldsymbol{q}_{cor} = fcn(\phi(k), \theta_0, \psi(k))$
9: $\hat{\boldsymbol{q}}(k) = \boldsymbol{q}_{cor} \frac{\hat{\boldsymbol{q}}(k)^*}{||\hat{\boldsymbol{q}}(k)^2||} \hat{\boldsymbol{q}}(k)$
10: **return** $\hat{\boldsymbol{q}}(k), z(k), \boldsymbol{q}_{cor}(k)$

Bibliography

[1] "Hydraulically smooth hull." http://www.borouge.com/IndustrySolution/. Accessed: 16-11-2017.

[2] M. Schultz, J. Bendick, E. Holm, and W. Hertel, "Economic impact of biofouling on a naval surface ship," *Biofouling*, vol. 27, no. 1, pp. 87–98, 2011.

[3] "innovation-awards." http://innovation-awards.nl/concept/fleet-cleaner/. Accessed: 4-10-2016.

[4] D. Borota, "Design of a position determination system for a ship's hull maintenance robot," Master's thesis, TU Delft, 2014.

[5] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. New Jersey,Prentice Hall PTR, 1993.

[6] P. Goel, S. I. Roumeliotis, and G. S. Sukhatme, "Robust localization using relative and absolute position estimates," in *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, vol. 2, pp. 1134–1140, IEEE, 1999.

[7] B. Yamauchi, "Mobile robot localization in dynamic environments using dead reckoning and evidence grids," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 2, pp. 1401–1406, IEEE, 1996.

[8] K. S. Chong and L. Kleeman, "Accurate odometry and error modelling for a mobile robot," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 4, pp. 2783–2788, IEEE, 1997.

[9] M. Hashimoto, H. Kawashima, T. Nakagami, and F. Oba, "Sensor fault detection and identification in dead-reckoning system of mobile robot: interacting multiple model approach," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 3, pp. 1321–1326, IEEE, 2001.

[10] "Mti user manual." https://www.xsens.com/wp-content/uploads/2013/12/MTi-10-series.pdf, 2014.

[11] C. C. Ward *et al.*, "Classification-based wheel slip detection and detector fusion for outdoor mobile robots," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 2730–2735, IEEE, 2007.

[12] A. Vasilijevic, B. Borovic, and Z. Vukic, "Underwater vehicle localization with complementary filter: Performance analysis in the shallow water environment," *Journal of intelligent & robotic systems*, vol. 68, no. 3-4, pp. 373–386, 2012.

[13] A. Matos, N. Cruz, A. Martins, and F. L. Pereira, "Development and implementation of a low-cost lbl navigation system for an auv," in *OCEANS'99 MTS/IEEE. Riding the Crest into the 21st Century*, vol. 2, pp. 774–779, IEEE, 1999.

[14] H.-P. Tan, R. Diamant, W. K. Seah, and M. Waldmeyer, "A survey of techniques and challenges in underwater localization," *Ocean Engineering*, vol. 38, no. 14, pp. 1663–1676, 2011.

[15] E. J. Krakiwsky, C. B. Harris, and R. V. Wong, "A kalman filter for integrating dead reckoning, map matching and gps positioning," in *Position Location and Navigation Symposium, 1988. Record. Navigation into the 21st Century. IEEE PLANS'88., IEEE*, pp. 39–46, IEEE, 1988.

[16] M. Wei and K. Schwarz, "Testing a decentralized filter for gps/ins integration," in *Position Location and Navigation Symposium, 1990. Record. The 1990's-A Decade of Excellence in the Navigation Sciences. IEEE PLANS'90., IEEE*, pp. 429–435, IEEE, 1990.

[17] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. The MIT press, 2006.

[18] W. Feitsma, "Design of an above water position determination system for a ship's hull maintenance robot," Master's thesis, TU Delft, 2016.

[19] D. Benyoucef, "A new statistical model of the noise power density spectrum for power-line communication," *Proc. IEEE ISPLC*, pp. 136–141, 2003.

[20] J. Wang, S. Y. Chao, and A. M. Agogino, "Sensor noise model development of a longitudinal positioning system for avcs," in *American Control Conference, 1999. Proceedings of the 1999*, vol. 6, pp. 3760–3764, IEEE, 1999.

[21] M. Verhaegen and V. Verdult, *Filtering and System Identification*. Cambridge University Press, 2007.

[22] C. Chatfield, *The analysis of time series: an introduction*. CRC press, 2016.

[23] H. Meng, Y. L. Guan, and S. Chen, "Modeling and analysis of noise effects on broadband power-line communications," *IEEE Transactions on Power delivery*, vol. 20, no. 2, pp. 630–637, 2005.

[24] L. Ojeda, D. Cruz, G. Reina, and J. Borenstein, "Current-based slippage detection and odometry correction for mobile robots and planetary rovers," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 366–378, 2006.

[25] G. Reina, L. Ojeda, A. Milella, and J. Borenstein, "Wheel slippage and sinkage detection for planetary rovers," *IEEE/Asme Transactions on Mechatronics*, vol. 11, no. 2, pp. 185–195, 2006.

[26] O. G. Hooijen, "On the channel capacity of the residential power circuit used as a digital communications medium," *IEEE Communications Letters*, vol. 2, no. 10, pp. 267–268, 1998.

[27] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using allan variance," *IEEE Transactions on instrumentation and measurement*, vol. 57, no. 1, pp. 140–149, 2008.

[28] P. Welch, "The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Transactions on audio and electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.

[29] L. Keviczky, *Control Engineering*. Széchenyi University Press, 2011.

[30] P. Petersen, *Riemannian geometry*, vol. 171. Springer, 2006.

[31] K.-R. Kim, P. T. Kim, J.-Y. Koo, and M. R. Pierrynowski, "Frenet-serret and the estimation of curvature and torsion," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 4, pp. 646–654, 2013.

[32] "Frenet-serret model coordinate frame." https://commons.wikimedia.org/w/index.php?curid=2656238. Accessed: 29-3-2017.

[33] L. Drolet, F. Michaud, and J. Cote, "Adaptable sensor fusion using multiple kalman filters," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, vol. 2, pp. 1434–1439 vol.2, 2000.

[34] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, pp. 343–349, 1999.

[35] T. Nicosevici, R. Garcia, M. Carreras, and M. Villanueva, "A review of sensor fusion techniques for underwater vehicle navigation," in *OCEANS'04. MTTS/IEEE TECHNO-OCEAN'04*, vol. 3, pp. 1600–1605, IEEE, 2004.

[36] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1, pp. 99–141, 2001.

[37] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A survey of underwater vehicle navigation: Recent advances and new challenges," in *IFAC Conference of Manoeuvering and Control of Marine Craft*, vol. 88, 2006.

[38] L. L. Menegaldo, M. Santos, G. A. N. Ferreira, R. G. Siqueira, and L. Moscato, "Sirus: A mobile robot for floating production storage and offloading (fpso) ship hull inspection," in *Advanced Motion Control, 2008. AMC'08. 10th IEEE International Workshop on*, pp. 27–32, IEEE, 2008.

[39] M. St-Pierre and D. Gingras, "Comparison between the unscented kalman filter and the extended kalman filter for the position estimation module of an integrated navigation information system," in *Intelligent Vehicles Symposium, 2004 IEEE*, pp. 831–835, IEEE, 2004.

[40] F. Daum, "Nonlinear filters: beyond the kalman filter," *IEEE Aerospace and Electronic Systems Magazine*, vol. 20, no. 8, pp. 57–69, 2005.

[41] H. Durrant-Whyte and T. C. Henderson, "Multisensor data fusion," in *Springer Handbook of Robotics*, pp. 585–610, Springer, 2008.

[42] P. S.-h. Won, M. Biglarbegian, and W. Melek, "Development and performance comparison of extended kalman filter and particle filter for self-reconfigurable mobile robots," in *Robotic Intelligence In Informationally Structured Space (RiiSS), 2014 IEEE Symposium on*, pp. 1–6, IEEE, 2014.

[43] E. Kiriy, H. Michalska, and G. Michaud, "Particle filter application to localization," in *Harbour Protection Through Data Fusion Technologies*, pp. 317–327, Springer, 2009.

[44] N. Houshangi and F. Azizi, "Accurate mobile robot position determination using unscented kalman filter," in *Electrical and Computer Engineering, 2005. Canadian Conference on*, pp. 846–851, IEEE, 2005.

[45] N. Houshangi and F. Azizi, "Mobile robot position determination using data integration of odometry and gyroscope," in *Automation Congress, 2006. WAC'06. World*, pp. 1–8, IEEE, 2006.

[46] F. Azizi and N. Houshangi, "Sensor integration for mobile robot position determination," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 2, pp. 1136–1140, IEEE, 2003.

[47] F. Azizi and N. Houshangi, "Mobile robot position determination using data from gyro and odometry," in *Electrical and Computer Engineering, 2004. Canadian Conference on*, vol. 2, pp. 719–722, IEEE, 2004.

[48] L. D'Alfonso, W. Lucia, P. Muraca, and P. Pugliese, "Mobile robot localization via ekf and ukf: A comparison based on real data," *Robotics and Autonomous Systems*, vol. 74, pp. 122–127, 2015.

[49] J. J. LaViola, "A comparison of unscented and extended kalman filtering for estimating quaternion motion," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3, pp. 2435–2440, IEEE, 2003.

[50] J. Zhou and L. Huang, "Experimental study on sensor fusion to improve real time indoor localization of a mobile robot," in *Robotics, Automation and Mechatronics (RAM), 2011 IEEE Conference on*, pp. 258–263, IEEE, 2011.

[51] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[52] J. R. Wertz, *Spacecraft attitude determination and control*, vol. 73. Springer Science & Business Media, 2012.

[53] J. L. Marins, X. Yun, E. R. Bachmann, R. B. McGhee, and M. J. Zyda, "An extended kalman filter for quaternion-based orientation estimation using marg sensors," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 4, pp. 2003–2011, IEEE, 2001.

[54] J.-H. Chen, S.-C. Lee, and D. B. DeBra, "Gyroscope free strapdown inertial measurement unit by six linear accelerometers," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 2, pp. 286–290, 1994.

[55] Y. Fuke and E. Krotkov, "Dead reckoning for a lunar rover on uneven terrain," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 1, pp. 411–416, IEEE, 1996.

[56] A. M. Sabatini, "Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 7, pp. 1346–1356, 2006.

[57] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[58] J. B. Kuipers *et al.*, *Quaternions and rotation sequences*, vol. 66. Princeton university press Princeton, 1999.

[59] T. D. Howell and J.-C. Lafon, "The complexity of the quaternion product," tech. rep., Cornell University, 1975.

[60] J. Valappil and C. Georgakis, "Systematic estimation of state noise statistics for extended kalman filters," *AIChE Journal*, vol. 46, no. 2, pp. 292–308, 2000.

[61] R. Schneider and C. Georgakis, "How to not make the extended kalman filter fail," *Industrial & Engineering Chemistry Research*, vol. 52, no. 9, pp. 3354–3362, 2013.

[62] V. A. Bavdekar, A. P. Deshpande, and S. C. Patwardhan, "Identification of process and measurement noise covariance for state and parameter estimation using extended kalman filter," *Journal of Process control*, vol. 21, no. 4, pp. 585–601, 2011.

[63] T. Chai and R. R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature," *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[64] J. Yi, J. Zhang, D. Song, and S. Jayasuriya, "Imu-based localization and slip estimation for skid-steered mobile robots," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 2845–2850, IEEE, 2007.

[65] C. C. Ward and K. Iagnemma, "Model-based wheel slip detection for outdoor mobile robots," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 2724–2729, IEEE, 2007.

[66] T. Nagata and G. Ishigami, "Gyro-based odometry associated with steering characteristics for wheeled mobile robot in rough terrain," *Advanced Robotics*, vol. 30, no. 23, pp. 1495–1508, 2016.

[67] S. Maeyama, N. Ishikawa, and S. Yuta, "Rule based filtering and fusion of odometry and gyroscope for a fail safe dead reckoning system of a mobile robot," in *Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on*, pp. 541–548, IEEE, 1996.

[68] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[69] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *Pattern recognition (ICPR), 2010 20th international conference on*, pp. 3121–3124, IEEE, 2010.

[70] R. Gonzalez, S. Byttner, and K. Iagnemma, "Comparison of machine learning approaches for soil embedding detection of planetary exploration rovers," in *International Conference of the ISTVS (International Society for Terrain-Vehicle Systems), Detroit, Michigan, USA, 12-14 September, 2016*, 2016.

[71] T. Soderstrom, H. Fan, B. Carlsson, and S. Bigi, "Least squares parameter estimation of continuous-time arx models from discrete-time data," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 659–673, 1997.

[72] A. Y. Ng, "Preventing" overfitting" of cross-validation data," in *ICML*, vol. 97, pp. 245–253, 1997.

[73] C.-W. Hsu, C.-C. Chang, C.-J. Lin, *et al.*, "A practical guide to support vector classification," 2003.

[74] J. Jung, H.-K. Lee, and H. Myung, "Slip compensation of mobile robots using svm and imm," in *Robot Intelligence Technology and Applications 2012*, pp. 5–12, Springer, 2013.

[75] K. Iagnemma and C. C. Ward, "Classification-based wheel slip detection and detector fusion for mobile robots on outdoor terrain," *Autonomous Robots*, vol. 26, no. 1, pp. 33–46, 2009.

[76] H. K. Ekenel and R. Stiefelhagen, "Analysis of local appearance-based face recognition: Effects of feature selection and feature normalization," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pp. 34–34, IEEE, 2006.

[77] S. Singh and P. Gupta, "Comparative study id3, cart and c4. 5 decision tree algorithm: a survey," 2014.

[78] P. Gaudiano, E. Zalama, and J. L. Coronado, "An unsupervised neural network for low-level control of a wheeled mobile robot: noise resistance, stability, and hardware implementation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 3, pp. 485–496, 1996.

[79] C. Bielza and P. Larrañaga, "Discrete bayesian network classifiers: a survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 5, 2014.

[80] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[81] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[82] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," 2007.

[83] S.-i. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, no. 6, pp. 783–789, 1999.

[84] B. Scholkopf, K.-K. Sung, C. J. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing support vector machines with gaussian kernels to radial basis function classifiers," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2758–2765, 1997.

[85] A. Ben-hur and J. Weston, "A userâĂŹs guide to support vector machines," 2007.

[86] K. Yoshida and H. Hamano, "Motion dynamics of a rover with slip-based traction model," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3, pp. 3155–3160, IEEE, 2002.

[87] C. C. Ward and K. Iagnemma, "A dynamic-model-based wheel slip detector for mobile robots on outdoor terrain," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 821–831, 2008.

[88] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.

[89] S. N. Sulaiman and N. A. M. Isa, "Adaptive fuzzy-k-means clustering algorithm for image segmentation," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 4, 2010.

[90] B. C. Patel and G. Sinha, "An adaptive k-means clustering algorithm for breast image segmentation," *International Journal of Computer Applications*, vol. 10, no. 4, pp. 35–38, 2010.

[91] M. Sahu, K. Parvathi, and M. V. Krishna, "Parametric comparison of k-means and adaptive k-means clustering performance on different images," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 2, pp. 810–817, 2017.

[92] D. Steinley, "K-means clustering: a half-century synthesis," *British Journal of Mathematical and Statistical Psychology*, vol. 59, no. 1, pp. 1–34, 2006.

[93] H.-K. Lee, K. Choi, J. Park, Y.-H. Kim, and S. Bang, "Improvement of dead reckoning accuracy of a mobile robot by slip detection and compensation using multiple model approach," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1140–1147, IEEE, 2008.

[94] R. K. Sidharthan, R. Kannan, S. Srinivasan, and V. E. Balas, "Stochastic wheel-slip compensation based robot localization and mapping," *Advances in Electrical and Computer Engineering*, vol. 16, no. 2, pp. 25–32, 2016.

[95] G. Bayar, M. Bergerman, A. B. Koku, *et al.*, "Improving the trajectory tracking performance of autonomous orchard vehicles using wheel slip compensation," *Biosystems Engineering*, vol. 146, pp. 149–164, 2016.

[96] Y. Koubaa, M. Boukattaya, and T. Dammak, "An adaptive control for uncertain mobile robot considering skidding and slipping effects," in *Systems and Control (ICSC), 2016 5th International Conference on*, pp. 13–19, IEEE, 2016.

[97] H. Gao, X. Song, L. Ding, K. Xia, N. Li, and Z. Deng, "Adaptive motion control of wheeled mobile robot with unknown slippage," *International Journal of Control*, vol. 87, no. 8, pp. 1513–1522, 2014.

[98] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[99] B. Barshan and H. F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 328–342, 1995.

[100] "Mearsk mc-kinney." http://www.nieuwsbladtransport.nl/Nieuws/Article/ArticleID/50391/ArticleName/MaerskLineboektverliesvan151miljoendollar. Accessed: 26-7-2017.

[101] "Hs teseo." http://maritime-connector.com/ship/teseo-9038866/. Accessed: 26-7-2017.

[102] M. Worring and A. W. Smeulders, "Digital curvature estimation," *CVGIP: Image understanding*, vol. 58, no. 3, pp. 366–382, 1993.

[103] "Teseo width." http://www.marinetraffic.com/nl/ais/details/ships/shipid:274851/mmsi:247071700/imo:6519041/vessel:TESEO. ,Accessed: 11-03-2018.

[104] P. J. Flynn and A. K. Jain, "On reliable curvature estimation.," in *CVPR*, vol. 88, pp. 5–9, 1989.

[105] P. J. Besl, "Geometric modeling and computer vision," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 936–958, 1988.

[106] P. J. Besl and R. C. Jain, "Invariant surface characteristics for 3d object recognition in range images," *Computer vision, graphics, and image processing*, vol. 33, no. 1, pp. 33–80, 1986.

[107] N. Bhatia *et al.*, "Survey of nearest neighbor techniques," *arXiv preprint arXiv:1007.0085*, 2010.

[108] I. Hmeidi, B. Hawashin, and E. El-Qawasmeh, "Performance of knn and svm classifiers on full word arabic articles," *Advanced Engineering Informatics*, vol. 22, no. 1, pp. 106–111, 2008.

[109] D. W. Aha, "Editorial," in *Lazy learning*, pp. 7–10, Springer, 1997.

[110] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[111] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[112] L.-Y. Hu, M.-W. Huang, S.-W. Ke, and C.-F. Tsai, "The distance function effect on k-nearest neighbor classification for medical datasets," *SpringerPlus*, vol. 5, no. 1, p. 1304, 2016.

[113] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in neural information processing systems*, pp. 1473–1480, 2006.

[114] M. H. Zweig and G. Campbell, "Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine.," *Clinical chemistry*, vol. 39, no. 4, pp. 561–577, 1993.

[115] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[116] M. Planning, "Analysis division," *Onboard Navigation Systems Characteristics," NASA JSC-14675*, 1981.

Glossary

List of Acronyms

EKF	Extended Kalman Filter
FLS	Fuzzy Logic Supervisor
GPS	Global Positioning System
KF	Kalman Filter
PDF	Probability Density Function
PF	Particle filter
UKF	Unscented Kalman Filter
UUV	Unmanned Underwater Vehicle
PSD	Power Spectral Density
SVM	State Vector Machine
IMM	Interacting Multiple Model
AWGN	Additive White Gaussian Noise
ACF	Auto Correlation Function
PCC	Pearson Correlation Coefficient
ROC	Receiver Operating Characteristics
AUC	Area Under the Curve
kNN	k-Nearest Neighbours
RBF	Radial Basis Function
FLS	Forward Looking Sonar

TF Transfer Function

SISO Single Input Single Output

FF-NN Feed Forward Neural Network