Martin Løvig Svabø

NAVIGATION AND SLIP DETECTION FOR AN AUTONOMOUS ROBOT IN CROP FIELDS

Master's thesis in Cybernetics and Robotics
Supervisor: Jan Tommy Gravdahl
Co-supervisor: Vegard Line
May 2021

Martin Løvig Svabø

NAVIGATION AND SLIP DETECTION FOR AN AUTONOMOUS ROBOT IN CROP FIELDS

Master's thesis in Cybernetics and Robotics
Supervisor: Jan Tommy Gravdahl
Co-supervisor: Vegard Line
May 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Abstract

Autonomous wheeled mobile robots operating in crop fields are frequently subject to wheel spin. Navigation systems for mobile wheeled robots generally rely on wheel encoder measurements to estimate the robot's linear and angular velocity. The encoder measurements are also frequently used to find the position and orientation of the robot through integration. However, when the robot is subject to wheel spin, the encoder measurements become redundant. Navigation systems for wheeled mobile robots subject to wheel spin must incorporate alternative measurements for estimating the linear and angular velocity when the robot is subject to slippage.

In this thesis, a navigation system that can detect slip and estimate the states of the robot when the robot is subject to slip has been developed. The proposed navigation system compares measurements from an inertial measurement unit and global positioning system with the encoder measurements to detect slippage. The state estimation utilizes the slip detection to switch between encoder measurements when no slip is present and an inertial navigation system using the inertial measurement unit to estimate the robot's velocity. The proposed navigation system is tested and evaluated in simulations through a variety of tests. The results show that the proposed navigation system is effective at detecting slip. However, the state estimation when the robot is subject to slip is varied.

Sammendrag

Autonome roboter som opererer i avlingsfelt, blir ofte utsatt for hjulspinn. Navigasjonssystemer for roboter er som oftest avhengig av målinger fra hjulenkodere for å estimere robotens lineære hastighet og vinkelhastighet. Hastighetsmålingene fra hjulenkodermålingene blir også ofte brukt for å finne posisjonen og orienteringen til roboten gjennom integrering. Når roboten er utsatt for hjulspinn blir hastighetsmålingene fra enkodermålingene feilaktige. Navigasjonssystemer for roboter som er utsatt for hjulspinn må derfor inneholde alternative målinger for å estimere den lineære hastigheten og vinkelhastigheten når roboten blir utsatt for glidning.

I denne oppgaven er det utviklet et navigasjonssystem som kan oppdage glidning og estimere tilstandene til roboten når roboten er utsatt for gliding. Det foreslåtte navigasjonssystemet sammenligner målinger fra en IMU og et globalt posisjoneringssystem med enkodermålingene for å oppdage glidning. Tilstandsestimeringsmetoden bruker glidedeteksjonen til å veksle mellom enkodermålinger når det ikke er noe hjulspinn, og et treghetsnavigasjonssystem som bruker IMU målinger for å estimere robotens hastighet. Det foreslåtte navigasjonssystemet blir testet og evaluert i simuleringer gjennom en rekke tester. Resultatene viser at det foreslåtte navigasjonssystemet effektivt kan oppdage glidning. Imidlertid er tilstandsestimeringen når roboten er utsatt for glid variert.

Preface

With this thesis, my two year Master of Science degree in Cybernetics and Robotics at the Norwegian University of Science and Technology carried out in the spring of 2021 is completed. This project was done in collaboration with Kilter AS over a period of 20 weeks equal to one semester.

I would like to thank my supervisor at NTNU Jan Tommy Gravdahl and my external supervisor from Kilter AS Vegard Line for valuable feedback and guidance. A special thanks to Jarle Dørum and Torgrim Lien for bringing me up to speed with Kilter's software and helping me with technical issues. I would also like to thank Kilter AS for providing the necessary tools for conducting this thesis.

Symbols and Abbreviations

$\boldsymbol{\eta}$:	Position and orientation in NED frame
$\mathbf{p}$:	Position in NED frame
$\boldsymbol{\Theta}$:	Orientation in NED frame
χ :	Heading angle
$\boldsymbol{\nu}$:	Velocity and angular velocity in body frame
$\boldsymbol{\omega}$:	Angular velocity in body frame
u:	Linear velocity in body frame
v:	Lateral velocity in body frame
r:	Angular velocity in body frame
β :	Sideslip-angle
U :	Total speed
γ_i:	Angular displacement of wheel i
ϵ_i:	Lateral slip
ζ:	Longitudinal slip
ω_i :	Angular velocity of wheel i
$\mathbf{R}$:	Rotation matrix
$\mathbf{T}$:	Transition matrix
$\mathbf{J}$:	Jacobain
T_N:	Time constant
C :	Mixing variable
MWR:	Mobile Wheeled Robot
RTK:	Real-time Kinematic

GPS:	Global Positioning System
IMU:	Inertial Measurement Unit
INS:	Inertial Navigation System
NED:	North East Down

Contents

Abstract i

Sammendrag ii

Preface iii

1 Introduction 1

1.1 Problem statement 2

1.2 Acknowledgments 3

1.3 Outline 3

2 Prerequisites 4

2.1 Referance frames 4

2.2 Linear velocity transformation 5

2.3 Angular velocity transformation 6

2.4 Kinematics of a nonholonomic mobile wheeled robot 7

2.5 System overview 8

3 Navigation 10

3.1 Inertial navigation systems for nonholonomic robots 11

3.1.1 Inertial navigation system 11

3.1.2 Wheel slip detection 12

3.1.3 Mixing variables 13

3.1.4 Slip estimation using GPS 14

3.1.5 State estimation 16

3.1.6 Slip control 17

3.2 Dynamic modeling 17

3.2.1 Previous work 18

3.2.2 Dynamic modeling Newtons method 19

3.2.3 Dynamic modeling Lagrange mechanics 21

3.3 Conclusion of literature study 24

4 Hardware and Software 25

4.1 Robot Operating Sytem 2 25

4.2 Simulation 26

4.2.1 Gazebo 27

4.2.2 Rviz 27

4.3 Sensors 28

4.3.1 Wheel Encoder 28

4.3.2 GNSS and RTK-GPS 28

4.3.3 Inertial measurement unit(IMU) 29

5 Implementation 32

5.1 Kalman filter 32

5.2 Tuning 34

5.2.1 Kalman Filter tuning 34

5.2.2 State estimation tuning 35

5.3 Kalman filter testing 38

6 Results and discussion 39

6.1 Linear Velocity Test 39
6.1.1 High friction 40
6.1.2 Low friction 41
6.1.3 Comparison between slip detection using IMU and GPS velocity 45
6.2 Angular velocity test 48
6.2.1 Discussion 51
6.3 Circle test 52
6.3.1 Discussion circle test 58

7 Conclusion **60**
7.1 Overview 60
7.2 Findings 60
7.3 Future work 61

A Appendix **65**
A.1 Linear velocity test 65
A.1.1 High friction 65
A.1.2 Low friction 67
A.1.3 Realistic friction 69
A.2 Angular velocity test 71
A.2.1 High friction 71
A.2.2 Low friction 73
A.2.3 Realistic friction 75
A.3 Circle test 77
A.3.1 High friction 77

A.3.2 Low friction 79

A.3.3 Realistic friction 81

1 Introduction

Row crop production in 2021 represents a notable portion of the world's food supply. As the projected global population in 2050 will be around 9 to 10 billion, the food supply must increase to meet the new demand. As the meat production of the world increases, the grain harvest per capita is decreasing. If the future food supply is to meet the future demand an increase in food produced from agriculture is needed. Unfortunately the expansion of agriculture comes with an environmental price. The environmental price includes soil erosion, surface and groundwater contamination, increase of CO2, increased weed and pest resistance and a decrease in biodiversity which is crucial for the ecosystem[1]. In a world with increased environmental concerns, the need for a revolutionized agricultural industry is evident. One of the main contributors to an increased row crop production is the use of herbicide. An herbicide is a substance that kills or inhibits unwanted plants. The use of herbicide dramatically increases weed production since the weed no longer has to compete with other plants for water supply, nutrients, and sunlight[2]. During the last 20 years, the weeds have become more resistant to herbicide use, while the bans on herbicide use have increased, making it harder for farmers to utilize herbicides to eliminate weeds effectively[3]. With the loss of effective herbicide, some farmers have abandoned using herbicides and have turned to labor-intensive manual hoeing. This increased cost of manual labor sparked the development of new en more effective ways of deploying herbicides.

Kilter, a company located at Langhus in Norway currently develops a robot called AX1 to deploy herbicide in a more effective and environmentally friendly way[4]. The AX1 robot shown in figure 1 is a two wheeled differential drive robot with a caster wheel that is placed off-center to not drive over any crops. The robot utilizes a vision based system together with deep learning to detect and differentiate different weeds and crops. Once a weed is detected, the robot utilizes high-precision nozzles to drop effective herbicide directly on the weed leaves. By only hitting the weed leaves, the herbicide does not affect the crop and soil. The herbicide is only deployed on the weed, and herbicide usage can be reduced by up to 95% compared to conventional herbicide deployment methods. Since the herbicide is not in contact with the crop, the herbicide can be more potent and therefore more effective. Another benefit of using the AX1 robot for the deployment of herbicides is that the crops have better growth conditions due to less or no exposure to herbicide. The AX1 robot can also navigate the fields autonomously, meaning the robot can drop herbicide all day on an entire field with little or no supervision. Consequently the labour force can be significantly reduced, making crop production more effective, environmentally friendly and cheaper. The robot can be a solution to the drop in available herbicides, the increased food demand, the increase in herbicide-resistant plants, and the increasing environmental concern of using herbicides.

One of the challenges in designing and operating the AX1 robot are rooted in the robot being autonomous. Many unpredictable factors like obstacles and uneven and unpredictable terrain, need to be taken into account on a large field when designing the automation system for the AX1 robot. The automation system relies on obtaining as much information as possible from the surrounding field and processing this information to steer and control the robot in a manner that maximizes the effectiveness of the deployment of herbicides without damaging the crops. By utilizing a variety of sensors combined with effective methods such as the Kalman Filter, it is possible to estimate the position, velocity, orientation and angular velocity(states) of the robot enabling reliable control the robot.

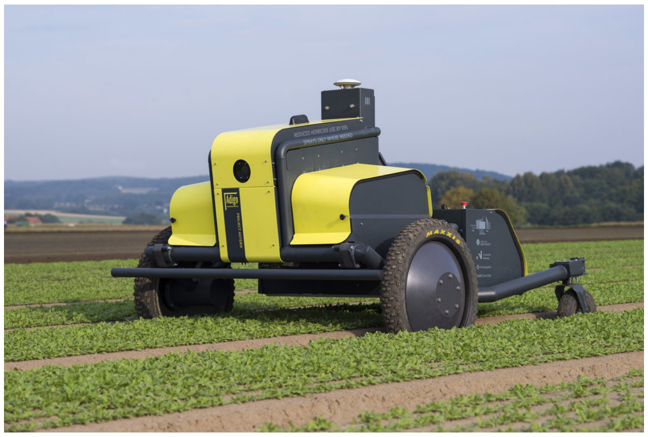


Figure 1: AX1 Robot

1.1 Problem statement

Control of agricultural robots, such as the AX1, are subject to many disturbances such as uneven and rough ground and a slippery surface, making the estimation of the robots states challenging. The slippery surface of a field makes the AX1 robot susceptible to wheel spin. Most state estimation methods rely heavily on encoder measurements that measure the rotational velocity of the wheels to calculate the robot's linear and angular velocity. However, when a mobile wheeled robot(MWR) such as the AX1 robot is subject to slip, the encoders yield poor measurements of the actual velocity of the robot. An alternative method for estimating the states of the robot while subject to wheel spin is therefore required. Available to the Kilter AX1 robot is a wide range of sensors that can be used to estimate the robot's states. Also available are physical parameters such as mass and moment of inertia which can be used to make dynamic models of the AX1 robot, making state estimation from simulation of the dynamic model possible. The goal of the thesis is to develop a system capable of reliably detecting slip and estimating the states of the robot when the robot is subject to wheel spin.

1.2 Acknowledgments

The slip detection and state estimation algorithm in this thesis further tests and builds on the work done in [5]. In the article the accelerometer measurements from the IMU is compared with the linear velocity measurements form the encoder to detect slip. An alternative method for detecting slip for linear velocity by comparing RTK-GPS velocity estimates with the encoder measurements is developed and tested in this thesis. A comparison between the two methods is provided in section 6.1. A derivation of a dynamic representation of the robot is also provided, based on the work done in [6], [7] and [8].

1.3 Outline

This thesis is divided into 7 chapters. A short description of the content of each section will now be presented.

- **Chapter 2** introduces the reference frames where the states of the robot are defined, the transformation matrices used to translate states between reference frames, the kinematic model of the AX1 robot when subject to slip, and an overview of the entire autonomous system.
- **Chapter 3** presents a literature study on inertial navigation systems(INS) and model-based navigation. The literature study is concluded with a discussion of the pros and cons of each method and reason for deciding to use INS.
- **Chapter 4** presents the hardware and software used in this thesis. This includes a presentation of the environment used for communication between the simulator and the state estimator and the sensors used by the AX1 robot.
- **Chapter 5** presents the implementation of the Kalman filter and the tuning of the state estimation method and the kalman filter. The tests performed in order to evaluate the performance of the state estimation are also presented.
- **Chapter 6** presents and discusses the simulation results from the tests presented in chapter 5.
- **Chapter 7** presents a brief overview of the thesis and gives a surmise of the most important findings from the simulations and a discussion on weather the method can be implemented on the real robot. Lastly the future work that can be done to improve the state estimation method is presented.

2 Prerequisites

2.1 Referance frames

When analysing the states of the robot a reference frame must be defined to which the robot's states can be related[9]. When modeling the dynamics of a vehicle it is useful to define several coordinate frames to represent the vehicle movements. The position and orientation of a vehicle is normally represented relative to a point on the ground on a flat plane perpendicular to the ground. This frame is called the North East Down(NED) frame. In the NED frame the x-axis is pointing north, the y-axis is pointing east and the z-axis is pointing down. The frame is stationary and the configuration of the robot represented relative to NED is denoted as

$$\eta = [x^n, y^n, z^n, \phi, \theta, \psi]^T \tag{1}$$

where the position of the robot in NED is denoted as $\mathbf{p} = [x, y, z]^T$ and the orientation in NED as $\mathbf{\Theta} = [\phi, \theta, \psi]^T$.

Velocity is easiest to describe relative to the robot. Therefore, the velocity of the robot is defined in the center of the wheel axle connecting the two front wheels in a frame called the body-fixed frame. In the body-fixed frame, the x-axis is defined pointing in the direction of forward motion, the z-axis down perpendicular to the NED frame and y-axis in the direction that completes the right hand rule for reference frames. The body frame is attached to the robot and moves with the robot. The velocities defined in the body fixed frame is denoted as

$$\nu = [u, v, w, p, q, r]^T \tag{2}$$

where $\mathbf{v} = [u, v, w]^T$ is the velocity and $\omega = [p, q, r]^T$ is the angular velocities of the robot in the body frame. An illustration of the NED frame and the body fixed reference frame in 2D can be found in figure 2.

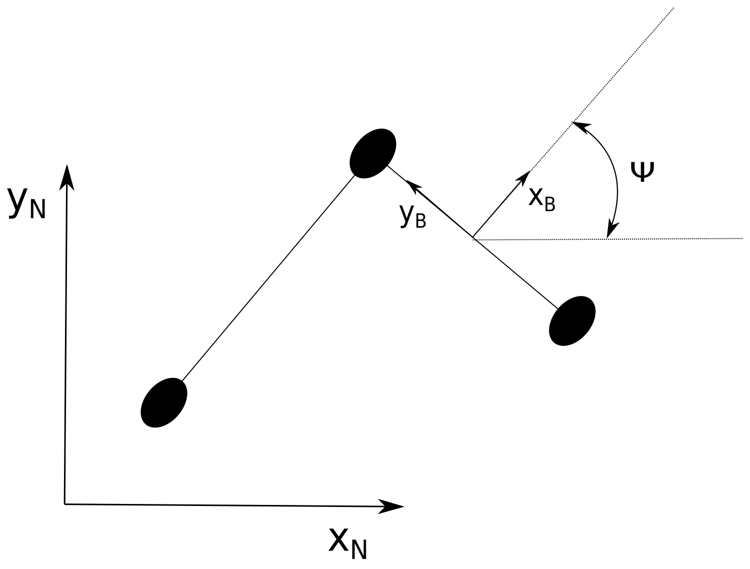


Figure 2: NED and body frame

2.2 Linear velocity transformation

To obtain the velocity of the robot defined in the NED frame the velocity in the body frame is translated to NED[9]. The velocities can be transformed to the NED frame by using Euler angle transformation. The rotation matrix $\mathbf{R}(\mathbf{\Theta}_{nb}) : \mathbf{T}^3 \rightarrow \mathcal{SO}(3)$ denotes the Euler angle rotation matrix which can be denoted as $\mathbf{R}_b^n$. The Euler angle rotation matrix $\mathbf{R}_b^n$ is described in terms of the Euler angles (attitude) $\mathbf{\Theta}_{nb} = [\phi, \theta, \psi]^T$ and can be described by the the principle rotation around the unit vectors. This yields the linear velocity transformation

$$\mathbf{R}_b^n := \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi} \tag{3}$$

By using the rotation matrix the velocity of the robot in the body fixed frame can be translated to NED as follows

$$\dot{\mathbf{p}}^n = \mathbf{R}_b^n \mathbf{v}^b \tag{4}$$

where $\boldsymbol{v}^b$ is the velocity in the body-fixed frame. The velocity can also easily be transformed from the NED frame the Body frame because of the property of the rotation matrix $\mathbf{R}_n^n = (\mathbf{R}_b^n)^T$. This means that the body velocity relative to NED can be written as

$$\mathbf{v}^b = (\mathbf{R}_b^n)^T \dot{\mathbf{p}}^n \tag{5}$$

2.3 Angular velocity transformation

The angular velocity in body $\boldsymbol{\omega}_{nb}^b$ can be transformed to angular rate in NED $\dot{\mathbf{\Theta}}_{nb}$ through the transformation matrix $\boldsymbol{T}(\mathbf{\Theta}_{\boldsymbol{nb}})$ with equation

$$\dot{\mathbf{\Theta}}_{nb} = \boldsymbol{T}(\mathbf{\Theta}_{nb})\boldsymbol{\omega}_{nb}^b \tag{6}$$

where the transition matrix $\boldsymbol{T}(\mathbf{\Omega}_{nb})$ is defined as

$$\boldsymbol{T}(\mathbf{\Theta}_{nb}) = \begin{bmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi / \cos\theta & \cos\phi / \cos\theta \end{bmatrix} \tag{7}$$

These definitions will be used throughout the next sections when the kinematics and dynamics of a differential drive mobile wheeled robot is derived.

2.4 Kinematics of a nonholonomic mobile wheeled robot

Kinematics describes the motion of an object without considering the forces that acts on the object. Fields where the Kilter AX1 operates are mostly flat, the assumption is therefore made that the robot operates on a flat 2D field. In this section the 2D kinematics for the AX1 robot when subject to slip will be derived[9][5].

With the assumption of a flat ground the roll, pitch and height of the robot is constant and can be neglected. The representation of the robot in 2D simplifies to 3-DOF where the coordinates in NED is defined as

$$\eta = [x_n, y_n, \psi_n]^T \tag{8}$$

where η is the position and orientation of the robot in NED defined the same way as in the previous section. The motion of the robot is defined by it's linear velocity u along the x-axis, the lateral velocity v along the y-axis and the angular velocity r around the z-axis. The velocity of a WHR on a 2D plane defined in body-fixed frame can be written as

$$\nu = [u, v, r]^T \tag{9}$$

The velocity of the robot defined in the body frame can be transformed to the NED reference frame by using the rotation matrix

$$R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Velocity in the body frame transformed to NED can then be written as

$$\dot{\eta} = R(\psi)\nu \tag{11}$$

By expanding this expression the kinematic equations for a nonholomonic mobile wheeled robot is obtained

$$\begin{aligned} \dot{x} &= u\cos(\psi) - v\sin(\psi) \\ \dot{y} &= u\sin(\psi) + v\cos(\psi) \\ \dot{\psi} &= r \end{aligned} \tag{12}$$

When the robot is subject to slipping and the robot has both a linear and lateral velocity there is a displacement in the direction in which the wheels of the robot is pointing and the direction which the robot is traveling. This displacement is called the sideslip angle and is defined as follows

$$\beta = \arctan(\frac{v}{u}) = \sin^{-1}(\frac{v}{U}) \tag{13}$$

If the robot has a lateral velocity then the actual angle of which the robot is traveling is the sum of the robot's heading and the sideslip angle. This angle is called course and is defined as

$$\chi = \psi + \beta \tag{14}$$

The total speed of the robot is determined by the speed in the linear and lateral direction of the robot

$$U = \sqrt{u^2 + v^2} \tag{15}$$

2.5 System overview

The system that enables the robot to move in a desired manner is made up of many modules. A module in this contexts is an independent software program with a specific function[9]. A mobile robot needs information about its desired state, its current state, and how to get from its current state to its desired state. All of this information is required to move a robot from A to B in a desired manner. Smart implementation of this system means to make all the modules independent of one another such that one module can be replaced with minor changes and the system can still work. This thesis will focus on the navigation system, but to give some perspective a brief explanation of each part in the system is in order.

Guidance system
The guidance system is used to determine the robot's path and create a reference for the robot to follow. The Kilter AX1 robot follows a path based on a set of waypoints in the field, either manually placed by the robot operators or through accurate GPS measurements from the tractor plowing the field. The guidance system makes paths between the waypoints used as a reference for the robot to follow. Using the tractor's GPS measurements, the waypoints are placed such that the robot's path is between

the crop rows. The path planner is the guidance system's primary function, but it can also include risk assessment and situational awareness.

Navigation system

To follow the path predetermined by the guidance system, the robot needs information about its current states. To obtain this information is the primary function of the navigation system. Navigation systems differs in the way they are implemented but the primary methods is to either use a robot-model or robot-measurements to determine the robot's states. There are advantages and disadvantages to both methods, and both approaches are discussed in detail in section 3.

Control system

Once information about the robot's location and destination is obtained, a control law is implemented to control the robot from its current position to its desired position. The desired control input to the robot is found by allocating the control input to the robot's driving wheels that will move the robot in the desired trajectory. Different control methods have been implemented on the AX1 robot previously, where nonlinear model predictive control(MPC) have yielded good results[10].

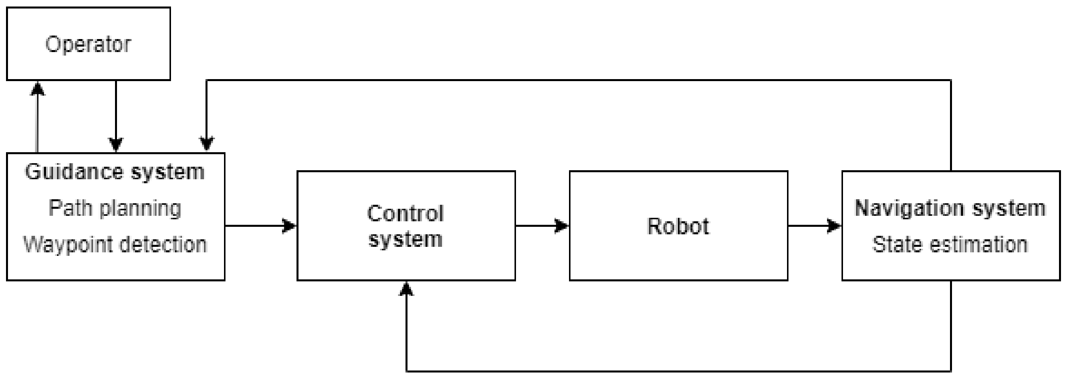


Figure 3: Motion control system

3 Navigation

In order to make a control system for a nonholominic WMR, information about the position, attitude and linear and angular velocity of the robot is needed. This information is provided by the navigation system. These days navigation systems are complex and uses advanced sensors and models, but navigation have existed for many years. According to [11], the first humans to use navigation was the Polynesians, which started to use stars, winds, and birds to determine their position around 330-320 BCE. In cybernetics, there are two common approaches used for navigation, inertial navigation systems(INS) and model-based navigation.

INS uses sensor data together with a kinematic model to estimate the states of the robot. Sensors are devices that can detect physical properties. These physical properties can for example be information about the position of the robot or the soil moisture. The most common sensors used in navigation include GPS, wheel encoder and an IMU. A detailed description of each sensor can be found in section 4.3.

Model based navigation on the other hand uses a physical model of the robot to estimate the states. Even though models are becoming very accurate, most model based navigation schemes also needs the aiding of sensors to give accurate state estimates. There are drawbacks to both these methods which will be discussed further in section 3.3.

The INS scheme discussed in section 3.1 uses an inertial measurement unit(IMU) to measure the acceleration and angular rate of the robot and encoders to measure the robots linear and angular velocity. By integrating these measurements the position and orientation of the robot is found. All sensors have some uncertainty, which can cause large errors in the states obtained through integration. Integration of measurements to obtain the position, orientation and velocity is known as dead reckoning, and the error accumulated by integration is called drift. The method presented in section 3.1 uses an algorithm to switch between using odometry and IMU measurements to estimate the velocity of the robot. The switching scheme based on [5] is implemented in order to estimate the states of the robot even if the robot is subject to wheel spin.

The other approach to obtaining information about the states of the robot is model-based navigation. This method utilizes physical models of the robot in order to estimate the robot's states. The more accurate the model, the more accurate the estimation of the states. A good understanding of the robot's underlying physics and properties is needed to build a reliable model. In most cases, many physical properties are neglected in order to build a model that is computationally efficient. In this project, the main problem is wheel slip, which is one physical property that is hard to model and ignored in most models. A couple of methods for modeling wheeled mobile robots when subject to slip is shown in section 3.2.

3.1 Inertial navigation systems for nonholonomic robots

INS is used when designing a reliable dynamic model is either to complicated, time consuming or not needed due to high quality sensors. There are different approaches to INS when the robot is subject to wheel-spin. A model of when the robot is subject to spinning is not available, which means that INS must rely on the sensors to determine if the robot is subject to slip. The majority of the methods found in the literature consists of a Kalman-filter to fuse a prediction model based on the kinematic equations with the sensor measurements and a technique for detecting slip. In[12] a switching algorithm that chooses between two different Kalman-filter algorithms is presented. The first Kalman-filter algorithm uses Encoder and GPS measurements when no slipping is detected, and the second algorithm adds an acceleration measurement to the filter when slip is detected.

The INS that is explored in this section will be based on the navigation system found in [5]. This navigation system detects slip by comparing encoder and IMU measurements and uses the slip detection in a selective mixing algorithm that fuses an INS only using IMU measurements with encoder measurements to estimate the states. In [5] this navigation scheme is tested is combination with the MPC algorithm[13] for controlling a MWR subject to slip.

The navigation scheme in [5] was chosen based on the simplicity of the algorithm and the fact that experiments where done in combination with an MPC trajectory tracing algorithm[13] similar to an MPC used on the AX1 robot previously[10].

3.1.1 Inertial navigation system

The INS in [5] uses only IMU(acceleration and angular rate) measurements to estimate the states of the robot when the robot subject to slippage. The accelerometer measuring acceleration is subject to drift due to a high measurement noise and growing bias. The gyro measuring the attitude rate, on the other hand, is a much more reliable sensor and has a low measurement noise and bias compared to the accelerometer. The sampling rate of the IMU is denoted as ΔT_N. The states estimated in this INS are $\mathbf{x} = [u, v, r, a_x, a_y]$ where a_x and a_y is the acceleration along the x and y-axis in the body frame. For obtaining the position and the orientation of the robot the kinematic equations in equation 12 can be used. The discretized nonlinear state space model for the robot can be written as

$$\hat{\mathbf{x}}_{\mathbf{k}} = \begin{bmatrix} \hat{a}_{x,k-1} + \hat{v}_{k-1}\hat{r}_{k-1} \\ \hat{a}_{y,k-1} - \hat{u}_{k-1}\hat{r}_{k-1} \\ 0 \\ 0 \\ 0 \end{bmatrix} T_n + \begin{bmatrix} \hat{u} \\ \hat{v} \\ \hat{r} \\ \hat{a}_x \\ \hat{a_y} \end{bmatrix} \tag{16}$$

and the measurement vector of the IMU can be written as

$$\hat{\mathbf{y}}_k = \begin{bmatrix} \hat{r_k} \\ \hat{a}_x \\ \hat{a}_y \end{bmatrix} \tag{17}$$

Since the state space model in equation 16 is nonlinear an extended Kalman filter(EKF) is used to estimate the states. To use an EKF a linearized model of the system is needed. The complete implementation of the EKF can be found in section 5.1. In order to linearize the system the jacobian is used. The jacobian of the state space model is

$$\mathbf{J}_k = \begin{bmatrix} 1 & r_{k-1}\Delta t & v_{k-1}\Delta t & \Delta t & 0 \\ -r_{k-1}\Delta t & 1 & -u_{k-1}\Delta t & 0 & \Delta t \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

By using an EKF this is a fully functioning navigation system, but the accuracy due to the bias drift and sensor noise makes this a unviable choice for state estimation. In the sections a method for combining INS estimates with wheel encoder measurements to obtain a better state estimate for robots subject to slip is presented.

3.1.2 Wheel slip detection

When the robot is not subject to wheel slip the best forward velocity and angular velocity measurements is obtained from the encoders of the robot. A description of the encoders and how encoder measurements are transformed to velocity and angular velocity measurements can be found in section 4.3.1. The linear and angular velocity measurements found from the encoder measurements are called odometry measurements and will be used henceforth. Since the odometry measurements are very accurate they should be used as linear and angular velocity estimates when the robot is not subject to slip. If both encoder measurements and IMU measurements are available one can determine if wheel slip is present or not by comparing the measurements. When both IMU and odometry measurements are available it is possible to compare the two estimates to determine if wheel slip is present. The two estimates are compared by using the tangential velocity estimate from odometry measurements and comparing this measurement with the IMU measurements over the time intervall ΔT. If the difference is large it is probable that wheel slip is present. This is what is called wheel slip detection. Because both the IMU and encoder measurement are subject to measurement noise the comparison between the odometry measurements and accelerator measurements are performed over a interval of 10 samples. This makes the slip detection more reliable, the downside being that there is a delay in the detection. The slip detection variable for linear velocity $\bar{\Delta} u_k$ is found as follows:

$$\bar{\Delta} u_k = \sum_{k=1}^{10} (u_k^O - u_{k-1}^O - a_{x,k} T_N) \tag{19}$$

where u_k^O is the linear velocity odometry measurement, and a_x the acceleration measured by the IMU along the x-axis[5]. This formula can detect a slipping tendency, but can not detect stationary slip as the method only compares velocity difference and acceleration.

3.1.3 Mixing variables

When slipping is detected the odometry measurements cannot be used to estimate the velocity of the robot. The slip detection variable is used to determine if the linear odometry measurement should be used to estimate the velocity of the robot or if the INS should be used. A double sided logsig function called a mixing variable that uses the detection variable as an input determines which method is used to estimate the velocity[5]. The mixing variable is a value between 1 and 0 where a value of 1 means that the odometry measurement is used, a value of 0 means that INS velocity estimate is used, and values between 1 and 0 means a combination of the two is used. The logsig function C_1 is defined as follows

$$C_{1,K} = pC_{1,K-1} + (1-p)\frac{1}{1+\exp\left(a_1\left(\left|\overline{\Delta v}_K\right| - b_1\right)\right)} \tag{20}$$

where a_1, b_1, and p are tuning variables. The variable a_1 is the steepness of the curve and should be chosen as small as possible without obtaining limit cycle behaviour. Even if the slip detection variable Δu found from an average of 10 samples, the variable is still subject to noise. A threshold variable b_1 is therefore used to ensure that the mixing variable $C1$ does not react to the noise. b_1 should be chosen as small as possible without triggering a changes in the mixing variable due to the noise from the measurements. The p parameter should be chosen as small as possible while avoiding limit cycles. In section 5.2.2 it is shown how these parameters are tuned and how they impact the state estimation scheme.

The same method is used to estimate the angular and lateral velocity of the robot. Lateral velocity is determined by the second mixing variable $C2$ which is a function of the lateral acceleration measurement from the IMU. The second mixing variable is included to stop lateral velocity estimate from reacting to the noise from the IMU. The mixing variable is tuned the same way as the previous mixing variable, where b_2 is the threshold for sideways velocity.

$$C_{2,k} = \frac{1}{1 + \exp\left(a_2\left(|\hat{a}_{\mathrm{y},k}| - b_2\right)\right)} \tag{21}$$

The third mixing variable C3 is used for angular velocity estimation. Mixing variable C3 is tuned the same way as C1 and C2 and uses the difference in angular velocity found from the IMU and odometry measurement to detect rotational slip.

$$C_{3,k} = \frac{1}{1 + \exp\left(a_3\left(\mid \omega_k^{\mathrm{O}} - \omega_k^{\mathrm{I}} \mid - b_3\right)\right)} \tag{22}$$

3.1.4 Slip estimation using GPS

Another method for estimating the velocity of the robot and determining if wheel slip is present not used in [5] is to use a GPS. The Kilter AX1 robot uses a high precision real time kinetic(RTK) GPS which will be described in detail in section 4.3. The RTK-GPS has a precision of 2-4 centimeters[14] contrary to a normal GPS with an average accuracy of around 1 meter[15]. This high accuracy RTK-GPS enables velocity estimation for robots that moves at a low velocity. Velocity estimated from RTK-GPS measurements is estimated by the equation

$$\mathbf{u}_k^{GPS} = \frac{\mathbf{p}_k^{GPS} - \mathbf{p}_{\mathbf{k-1}}^{\mathbf{GPS}}}{\Delta T_{GPS}} \tag{23}$$

where $\mathbf{v}_k^{GPS}$ is the GPS velocity estimate at time-step k, $\mathbf{p}_k^{GPS}$ the position measured by the GPS at time-step k and ΔT_G the frequency of the RTK-GPS measurements. One might assume the velocity estimate to be accurate, because of the low error in the RTK-GPS measurement but the reality is that with a typical GPS sampling rate of

$\Delta T_G = 0.1$ and RTK-GPS accuracy of around 2 centimeters the velocity estimate has an accuracy of around $0.2m/s$. This is great for fast moving robots, but for the kilter AX1 robot which moves at velocities from $0.2m/s$ to $0.5m/s$ the GPS velocity can not be used directly. The proposed solution to this problem is to average GPS velocity over a interval of 6 samples. Larger intervals will yield more accurate estimation when the velocity is constant, but for constant changing velocities this becomes a problem as inaccurate velocity measurements will be included in the average velocity and the estimate will become inaccurate. A balance between accuracy and responsiveness was therefore considered and an interval of 6 was found to be optimal. The average velocity is implemented over the last 6 samples as follows.

$$\bar{\mathbf{u}}_k^{GPS} = \sum_{i=0}^{6} \frac{\mathbf{u}_{k-i}^{GPS}}{6} \tag{24}$$

where k denotes the current RTK-GPS velocity estimate. The RTK-GPS velocity obtained is the average velocity of the robot along the x and y-axis in NED. In order to compare the velocity estimated from the RTK-GPS measurements with the velocity measured from the encoders the GPS velocity needs to be transformed to the body frame using equation 5. After the transformation the forward and sideways velocity of the robot is obtained. In order to preform the transformation the heading of the robot is needed. In the next section it is shown how the the kinematic equations in equation 12 together with the state estimates are used in order to find the position and heading of the robot, and this heading estimate is used in the transformation. The heading angle can also be fund by using a magnetometer described in section 4.3.3. A different method used if the heading is not available is to use the absolute velocity of the GPS velocity estimate. This method will yield a good estimates when the robot is moving and no sideways slip is present. When the velocity estimate is averaged and transformed to the body fixed frame the uncertainty is small enough so that the GPS velocity estimate can be compared with the encoder velocity measurements. Another mixing variable is introduced comparing the two defined as follows

$$C_{4,k} = \frac{1}{1 + \exp(a_4(|\bar{u}_k^{GPS} - u_k^O| - b_4))} \tag{25}$$

If the GPS velocity estimate is used the mixing variable $C4$ replaces the mixing variable $C1$. The two methods for estimating the velocity is shown in section 6.1.

3.1.5 State estimation

When the mixing variables are found the final state estimate is determined. For linear, angular and lateral velocity the mixing variables determine if the odometry or the INS should be used as the final state estimate. For lateral velocity this straight forward approach can not be used since in that case pure INS would be used when the threshold is crossed. This estimate would be very susceptible to noise and would frequently cross the threshold b_2 and then reset to zero. Therefore a new tunable geographical relaxation variable q_2 is introduced to obtain a more consistent and stable lateral velocity estimate[5]. The final state estimates is found as follows

$$\hat{u}_k = C_{1,k}u_k^O + (1 - C_{1,k})u_k^K \tag{26}$$

$$\hat{u}_k = C_{4,k}u_k^O + (1 - C_{4,k})u_k^K \tag{27}$$

$$\hat{\omega}_k = C_{3,k}\omega_k^O + (1 - C_{3,k})\omega_k^K \tag{28}$$

$$\hat{v}_k = C_{2,k}\hat{v}_{k-1}q_2 + (1 - C_{2,k})v_k^K \tag{29}$$

After obtaining the final estimate from utilizing the mixing variables, the robot's velocity in NED and the robots orientation can be found by using forward Euler integration.

$$\begin{aligned} x_{k+1} &= x_k + (u_k \cos\psi_k - v_k \sin\psi_k)\,\Delta T \\ y_{k+1} &= y_k + (u_k \sin\psi_k + v_k \cos\psi_k)\,\Delta T \\ \psi_{k+1} &= \psi_k + r_k \Delta T \end{aligned} \tag{30}$$

If RTK-GPS measurements are used it is also possible to update the position measurements, and if a magnetometer is included the heading can also be estimated more precisely. Another possibility is to include the position and heading in the Kalman filter and add the GPS and magnetometer to the measurement to the measurement vector.

The total speed and sideslip angle can be found by inserting the linear velocities into equations 15 and 13.

3.1.6 Slip control

A slip control algorithm utilizing trajectory tracing as the guidance system is used in [5]. The slip control works by decreasing the step size making the reference point at the next time step closer to the robot. This will make the control system decrease it's velocity and make the robot more stable and less subject to slipping. Slip control is a trajectory tracing algorithm and might be overly complex for the application discussed in this thesis. Path following is a less complex guidance law and utilises waypoints or predetermined path. For path following the guidance algorithm would need to incorporate a decrease in the desired velocity in order to stabilize the system when slip is present. The step size algorithm in [5] might be used as a scaling factor for the velocity reference in the path following guidance law.

Finally the states, side-slip and slip control reference can be fed to the MPC scheme for robust control[13]. As shown in [5] the navigation method using only sensors and kinematic models to control a wheeled mobile robot yielded good results. With the switch from trajectory tracking to path following some of the complexity of the method is decreased and can possibly equally effective.

3.2 Dynamic modeling

Dynamic models are a powerful tool when controlling robots. Model based navigation uses a dynamic model of the system to estimate the states of the robot. A dynamic models is typically used to simulate the system's response from a control input and fusing the model estimate with the sensor measurement using a Kalman filter to obtain reliable state estimates. One advantage of dynamic modeling over INS is that additional state estimates are obtained that can be used if one measurement becomes redundant such as the odometry measurements when the robot is subject to wheel spin[6][9]. Much research is done on modeling wheeled mobile systems, but only a small parentage considers wheel slip. Most articles consider slow-moving robots with high friction surfaces. Some literature focuses on high-speed robots and wheel slip, but there is limited research on slow-speed off-road vehicles. One of the main contributors in this field for off-road model-based navigation is NASA's Mars Rover[16][17]. However, these models are often complicated and beyond the scope of this thesis to explore. The focus of this small literature study has been to find an effective and simple way of modeling an off-road robot subject to wheel slip. There are two equivalent ways of modeling the robot's dynamics, either with Newton laws or with Lagrange mechanics.

There are different approaches and levels of accuracy to modeling a nonholonic robot. Some approaches consider both lateral and longitudinal forces [18][8], some also consider normal force by including the mass of the robot in the model[19][7] a few also consider the angle of the terrain [6]. In [20] it is developed a wheel-ground interaction model for accurate ground speed estimation. The difference between these models is the complexity of the models. In [20] the ground speed estimation requires comprehensive testing to find friction constants for the wheel and surface. Many of the methods use the "Magic formula" developed in 1987 to estimate the ground traction in order to predict when the robot is subject to wheel spin[21][22]. The method is tested and verified over the years and has generally yielded good results. The method is, however, not well suited for applications such as modeling the traction of a robot on a field. Many factors determine ground-wheel interactions, such as the slope of the terrain if there are variations in the surface's friction the method fails. All these factors make it hard to develop reliable models for estimating the states of and MWR in a field environment. Accurate detection of wheel slip is the key to a good state estimate for the AX1 robot. A schematic drawing of the Kilter AX1 robot is shown in figure 4.

The following subsections will present a model previously used by Kilter, a derivation of a model based on newtons laws and one derivation of a model using Lagrange mechanics.

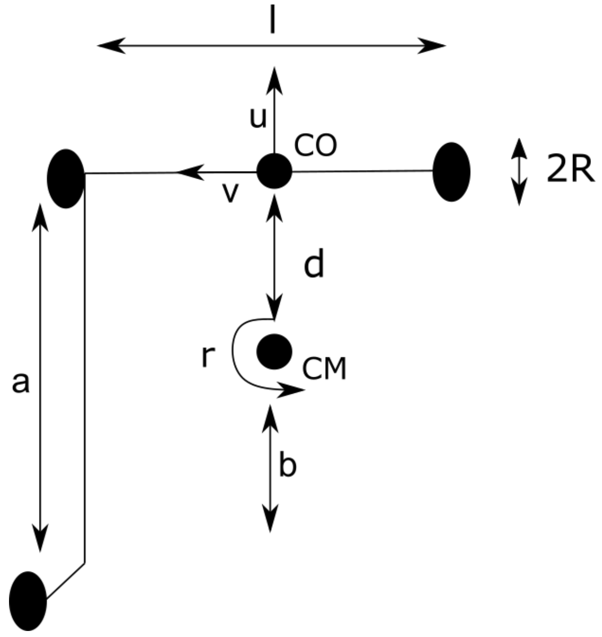


Figure 4: 2D model of the AX1 robot

3.2.1 Previous work

In previous research done on the Kilter AX1 robot [10] a dynamic model based on the works in[23] and [24] is used. In this work the slip of the robot has been implemented as a disturbance in the model. The model is described by the state space representation

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{u} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} ucos\psi - a\omega sin\psi \\ usin\psi + a\omega cos\psi \\ \omega \\ \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}u \\ -\frac{\theta_5}{\theta_2}u\omega - \frac{\theta_6}{\theta_2}\omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{\theta_1} & 0 \\ 0 & \frac{1}{\theta_2} \end{bmatrix} \begin{bmatrix} u_{ref} \\ \omega_{ref} \end{bmatrix} + \begin{bmatrix} \delta_x \\ \delta_y \\ 0 \\ \delta_u \\ \delta_\omega \end{bmatrix} \tag{31}$$

where $\delta = [\delta_x, \delta_y, 0, \delta_u, \delta_\omega]$ are the disturbance from the wheel slip and $\theta_{1,2,3,4}$ are constants including physical parameters of the robot. These constants are given in[10]. The model is a major simplification due to the slippage being modeled as a disturbance, but the model has the major advantage in the reduced complexity compared to other models.

3.2.2 Dynamic modeling Newtons method

The derivation of the model dynamics presented in this section is based on [6] and [7]. The dynamic model in [6] includes a sloping terrain, but in this thesis a flat ground is assumed and the derivation of the forces acting on the robot will be simplified accordingly. The dynamic model provided in [6] uses Newtons method to model all the forces exerted on the robot. Each wheel is subject to a lateral, longitudinal and normal force $F_{i,x}$, $F_{i,y}$ and N_i, where (i=l,r) denotes the left and right wheels. The longitudinal force of the left and right wheel are subject to a traction force and resistance force $F_{i,trac}$ and $F_{i,rollres}$. The forces on the caster wheel are F_{cx}, F_{cy} and N_c. The traction force of the caster wheel can be neglected as the wheel is undriven. The forces and moments exerted on the robot in a body centered reference frame can be written as

$$\begin{aligned} \sum F_x &= F_{rx} + F_{lx} + F_{cx} \\ \sum F_y &= F_{ry} + F_{ly} + F_{cy} \end{aligned} \tag{32}$$

where the longitudinal force $F_{i,x}$ can be decomposed into rolling resistance and traction force

$$F_{i,x} = F_{i,trac} - F_{i,rollres} \tag{33}$$

External forces, such as wind are usually negligible and therefore not included in the models. If we assume that the center of rotation is at the point between the front wheel the momentum of the robot can be written as

$$\sum M_z = I_z\dot{\omega} = \frac{l}{2}(F_{r,trac} - F_{r,rollres} - (F_{l,trac} - F_{l,rollres})) \tag{34}$$

where I_z is the moment of inertia and l is the distances between the front wheels. The schematics of the robot can be found in figure 4. The full dynamic equations based on these equations is found by dividing by the mass of the robot for obtaining linear acceleration and moment of inertia for obtaining angular acceleration.

The normal forces acting on each wheel can be found as follow

$$\begin{aligned} N_r = N_l &= \frac{W}{2}(\frac{b}{a+b}) \\ N_c &= \frac{W}{2(a+b)}a \end{aligned} \tag{35}$$

where a is the distance from the left front wheel to the caster wheel, h is the height to the center of the mass of the robot. b is the horizontal distance from the center of mass to the caster wheel and d is the length of the front wheels axle.

In order to model the traction force most models also include wheel slip(i) and wheel skid(i_s)

$$\begin{aligned} i &= 1 - \frac{u}{R\dot{\gamma}} \\ i_s &= 1 - \frac{R\dot{\gamma}}{u} \end{aligned} \tag{36}$$

where R is the radius of the tire and $\dot{\gamma}$ is the angular velocity of the driving wheels. The velocity of the robot is usually obtained through GPS or IMU measurements, while the angular velocity of the the wheel is gathered from a wheel encoder. After obtaining information about the wheelspin and wheelslip the magic formula can be used to find the friction forces of the wheel. As simplified version of the magic formula is given in [6] where the traction force and rolling resistance are only dependent on three ground related constants in contrast to the six needed in the normal magic formula. The simplified traction force and rolling resistance can be written as

$$F_{\text{traction}} = N\left(\operatorname{sign}(u_{rel})\, C_1\left(1 - \mathrm{e}^{-A_t|u_{rel}|}\right) + C_2 u_{rel}\right) \tag{37}$$

$$F_{\text{roll res}} = -\operatorname{sign}(u_{fwd}) N(R_1(1 - e^{-A_{\text{roll}}\,|u_{fwd}|}) + R_2|u_{fwd}|) \tag{38}$$

where $u_{rel} = R\omega - u_{fwd}$ is the velocity of the tire relative to the ground and $u_{fwd,i} = u \pm 0.5dr$ is the forward velocity of the tire. C_1, A_t, C_2, R_1, A_{roll} and R_2 are all constants related to the tire-ground interaction, each variable is described in [6]. The dynamic equations provided by the derivation of the newton laws are used together with tire dynamics provided in [6] to build an extended Kalman filter and a slip detection algorithm.

3.2.3 Dynamic modeling Lagrange mechanics

Lagrange mechanics are a common tool when modeling mechanical systems. There are many proposed models where Lagrange mechanics is used to model robots when subject to wheel spin [19][18]. Simplified models that reduces the complexity of the slippage models has also been developed[25]. In [8] a dynamic model of the system is developed together with a feedback-linearization algorithm to stabilize the system. The Lagrange mechanics in this section will be based on the dynamic models presented in [8].

The benefit of using Lagrange mechanics is the systematic way in which the models are constructed. To construct the Lagrangian model the potential(P) and kinetic energy(K) is considered separately to find the Lagrangian function $L = K - P$. The Lagrangian models are commonly found in the literature due to the simplicity of the the model construction. Once the potential and kinetic energy has been found the Lagrange equation can be developed. The Lagrangian equation can be written as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}}\right) + \frac{\partial L}{\partial \mathbf{q}} = F - \mathbf{A}^T \boldsymbol{\lambda} \tag{39}$$

where $\mathbf{q}$ is the generalized coordinates of the system, F is the generalized force vector, $A(q)$ is the constraint matrix and $\boldsymbol{\lambda}$ is the Lagrange multipliers associated with the constraints.

The first step in deriving the Lagrange equations is to define generalized coordinates that can describe the configuration of the system. For differential drive three wheeled mobile robots with wheel spin a popular choice of generalized coordinates is

$$\mathbf{q} = [x, y, \psi, \epsilon, \zeta_r, \zeta_l, \gamma_r, \gamma_l] \tag{40}$$

where the lateral and longitudinal slip of the left and right wheels are denotes as ϵ, ζ_r and ζ_l. γ_r and γ_l is the angular displacement of the right and left wheel making $\dot{\gamma}_r$ and $\dot{\gamma}_l$ the velocity of the right and left wheel.

When the generalized coordinates of the robot have been defined the potential and kinetic energy of the system can be derived. If we assume a flat ground then there are no potential energy present in the system and the Lagrangian function simplifies to $\mathcal{L} = K$. The kinetic energy of the robot is the sum of the kinetic energy of the body due to the velocity denoted as K_B and the force exerted on the robot from the left and right wheels denoted as K_L and K_R. The total kinetic energy can be written as $K = K_B + K_L + K_R$. If we denote the center of mass as M and the total mass of the robot as m_M the kinetic energy of the body of the robot can be written as

$$K_m = \frac{1}{2} m_B(\dot{x}_M^2 + \dot{y}_M^2) + \frac{1}{2} I_M r^2 \tag{41}$$

where I_M is the moment of inertia around the center of mass. The kinetic energy of the left and right wheel can be written as:

$$\begin{aligned} K_L &= \frac{1}{2} m_w(R^2\dot{\gamma}_l^2 + \dot{\zeta}_l^2 + \dot{\epsilon}^2) + \frac{1}{2} I_W \dot{\gamma}_l^2 + \frac{1}{2} I_D r^2 \\ K_R &= \frac{1}{2} m_w(R^2\dot{\gamma}_r^2 + \dot{\zeta}_r^2 + \dot{\epsilon}^2) + \frac{1}{2} I_W \dot{\gamma}_r^2 + \frac{1}{2} I_D r^2 \end{aligned} \tag{42}$$

where m_w is the mass of each wheel and I_W and I_D is the moment of inertia about the rotational and diameter axis of the wheels.

The next step in building the Lagrangian equation is to find the constraint matrix $\mathbf{A}(\mathbf{q})$. The nonholomonic constraints of the robot when slip is not included is called the pure rolling constraints and can be written as follows

$$\begin{aligned} R\dot{\gamma}_r &= \dot{x}_M \cos\psi + \dot{y}_M \sin\psi + \frac{l}{2} r \\ R\dot{\gamma}_l &= \dot{x}_M \cos\psi + \dot{y}_M \sin\psi - \frac{l}{2} r \\ 0 &= \dot{y}_M \cos\psi - \dot{x}_M \sin\psi \end{aligned} \tag{43}$$

where R is the radius of the wheel and l is the length of the wheelbase and a is the distance from the wheelbase to the center of mass. No slipping in the model means that the robot has no velocity in the lateral direction. If wheel slip is included, then we introduce longitudinal and lateral displacement for each wheel denoted as (ζ_r, ζ_l), ϵ_r and ϵ_l. The longitudinal displacement for each wheel is equal to the left and right wheel linear displacement $\rho_i = R\gamma_i + \zeta_i$. The relationship between the velocity of the robot and the wheels is

$$\begin{aligned}\dot{\rho}_r &= \dot{x}_M \cos\psi + \dot{y}_M \sin\psi + lr \\ \dot{\rho}_l &= \dot{x}_M \cos\psi + \dot{y}_M \sin\psi - lr \\ \dot{\epsilon} &= \dot{y}_M \cos\psi - \dot{x}_M \sin\psi\end{aligned} \tag{44}$$

To find the constraint matrix $\mathbf{A}(\mathbf{q})$ we solve the kinematic constraint equation

$$\mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{0} \tag{45}$$

where the $\mathbf{A}(\mathbf{q})$ matrix is obtained from comparison between equations (44) and (45). The result is an $\mathbf{A}(\mathbf{q})$ on the form

$$\mathbf{A}(\mathbf{q}) = \begin{bmatrix} \cos\psi & \sin\psi & l/2 & 0 & -1 & 0 & -R & 0 \\ \cos\psi & \sin\psi & -l/2 & 0 & 0 & -1 & 0 & -R \\ -\sin\psi & \cos\psi & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{46}$$

The Lagrange function can be written on the form

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial \mathcal{L}}{\partial \mathbf{q}} = \mathbf{u} + \mathbf{A}^T \lambda \tag{47}$$

where $\lambda = [\lambda_1, \lambda_2, \lambda_3]^T$ is the Lagrange multipliers and $\mathbf{u}$ is the generalized force acting on the system. If we solve the Lagrange equation the dynamics of the robot can be represented as

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{N_1}\tau + \mathbf{N_2}F_y + \mathbf{N_3}\mathbf{F_x} + \mathbf{A^T}\boldsymbol{\lambda} \tag{48}$$

where

$$\begin{aligned}\mathbf{N_1} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\ \mathbf{N_2} &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{N_3} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \\ \boldsymbol{\tau} &= \begin{bmatrix} \tau_r & \tau_r \end{bmatrix}^T\end{aligned} \tag{49}$$

were N are input matrices and $\boldsymbol{\tau}$ is the torque vector, $\mathbf{F}_x = [F_r, F_l]$ and F_y are unknown lateral forces. $\mathbf{F}_x$ is the sum of the traction and slipping force presented in equation

33. The matrix M is a positive definite mass matrix and can be found together with the inertia matrix in [8].

The equation 48 can be rewritten with respect to the wheels angular rate and lateral and longitudinal slip

$$\dot{\mathbf{q}} = \mathbf{S_1(q)v} + \mathbf{S_2(q)}\dot{\nu} + \mathbf{S_3(q)}\dot{\epsilon} \tag{50}$$

where $\mathbf{v} = [\dot{\gamma}_r, \dot{\gamma}_l]$ and $\mathbf{S}_1(\mathbf{q})$, $\mathbf{S}_2(\mathbf{q})$ and $\mathbf{S}_3(\mathbf{q})$ are matrices which can be found in [8]. By taking the derivative of this equation and do some simplifications we get

$$\mathbf{m}\dot{\mathbf{v}} + \mathbf{b}r\dot{\epsilon} + \mathbf{z}\ddot{\boldsymbol{\zeta}} = \boldsymbol{\tau} \tag{51}$$

where $\mathbf{m}$ is a positive definite matrix, $\mathbf{b}$ and $\mathbf{z}$ are constant vectors and matrices that can be found in [8]. The dynamics of the model is given by the equation

$$\dot{\mathbf{v}} = \mathbf{m}^{-1}\left(\boldsymbol{\tau} - \mathbf{b}r\dot{\epsilon} - \mathbf{z}\ddot{\boldsymbol{\zeta}}\right) \tag{52}$$

3.3 Conclusion of literature study

Both INS and model-based dynamics are viable options for modeling of an MWR subject to wheel spin. The sensor-based method has the advantage of not needing robot specific and terrain specific parameters. INS is purely based on the kinematic model of the robot and measurements. The proposed INS in section 3.1 uses the odometry measurement as the default state estimate when no slipping is detected. For slip detection the algorithm compares the IMU, odometry and RTK-GPS measurements and switches to using pure INS when slip is detected. The drawback of this method is the uncertainty associated with using INS over longer intervals.

Model based navigation on the other hand uses a dynamic model of the robot in combined with sensor measurements in a Kalman filter to estimate the robots states. Two proposed methods for developing such a model have been shown in section 3.2. The downside of using model based navigation is that physical parameters of the robot and the terrain is required. Inaccurate model and terrain parameters as well as an inaccurate model can cause large model errors and poor state estimates. The advantage is that model based navigation is not reliant on uncertain measurements such as the IMU. One more advantage is that an accurate model can yield state estimates if a sensor fails. Slip detection using model based is complicated and therefore not shown in the derivation of the model based method, but a similar principle to the slip detection in [5] is used by comparing the model estimate with the odometry measurements[6].

After some consideration the conclusion have been made that the best choice of method for the purposes of this project is to utilize the INS scheme. Simplicity and the fact that a high precision RTK-GPS with the potential of detecting slip reliably, not explored in [5] makes the method an interesting possibility which can yield good results. The method proposed in [5] have been tested with an MPC[13] with good results and MPC have been shown to yield good results for the Kilter AX1 previously robot[10].

4 Hardware and Software

In this section the hardware and software used in this project will be discussed. In section 4.1 the environment which the sensor-based navigation scheme is implemented will be presented. In section 4.3 a detailed description of all the sensors used in the implementation will be presented.

4.1 Robot Operating Sytem 2

The robot operating system(ROS) is an open source software and library tool used for robot applications. The first version of ROS was released in 2007 and have since gone through a number of iterations calumniating in the release of ROS2 which was released in 2017 which have also been through a few additional iterations since then. Although not an operation system, ROS provides a framework for making robot programming modular and versatile. The framework provides functionality which enables communication between the hardware, operating system and user. The modular nature of ROS2 makes it easy for developers to share code and use code from others to suit their application. This makes ROS2 an effective tool for fast development of robot application and the sharing mentality of the ROS2 community makes fast development of a wide range of applications possible[26].

The ROS framework is made up of a workspace which makes it possible to run different versions of ROS and switching between them the easily. Each ROS2 workspace is made up of nodes which are pieces of code written in either C++ or Python. Each node typically has a function which is independent from the other nodes in the workspace. This is how the ROS eco system archives it's modularity. One node can for example simulate the robot, one node can be a controller and another node can process sensor information. If the developer understands the function of a node and what information the node needs, the developer does not need to know precisely how the node works. One of the main features of ROS 2 is the framework for exchanging information between the nodes[27]. Information can be exchanged either through what is called topics or services. Topics acts as a bus where nodes can send information and other nodes can extract information. There is no limit on how many nodes can subscribe or publish to a topic. Nodes that send information to topics are called publishers and nodes that extracts information are called subscribers. The only limitation for sending or receiving data is the datatype that needs to be equal for both subscriber and publisher[28]. An illustration of how ROS2 nodes and topics interact can be found in figure 5. The other method for nodes to exchange information are services. Nodes using services either sends a request or gives an response to the request. The service determines what kind of information can be requested and what response is valid. Nodes that requests information are called service clients and nodes that responds to the requests are called service servers[29]. In this thesis only one node is developed. The node is the implementation of the state estimation method described in section 3.1 and the nodes functiones as a slip detector and estimates the state of the robot. Estimated states are published on the topic /motus/state_estimate that be accessed by other modules in the system such as the controller.

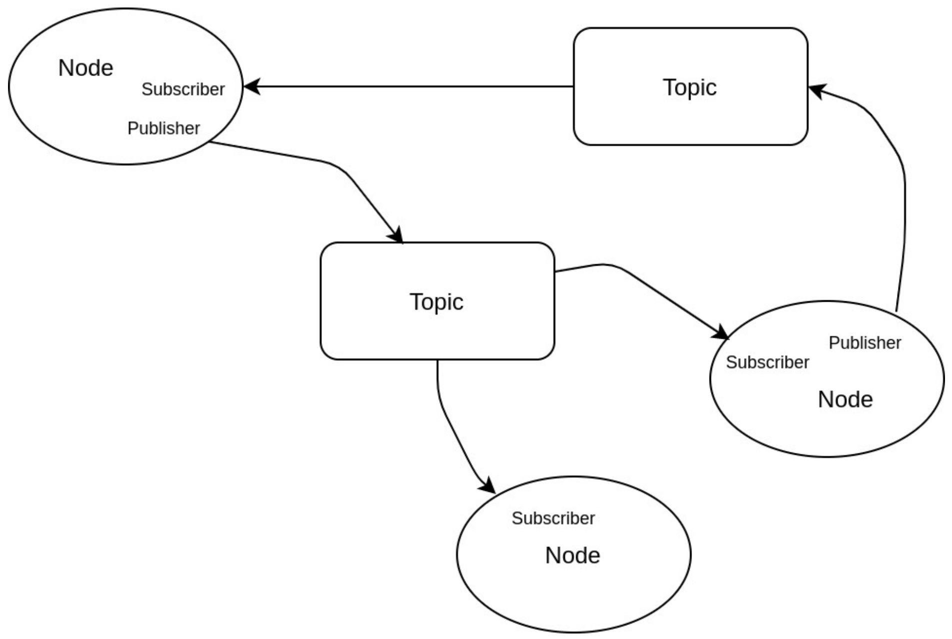


Figure 5: ROS2 topics and nodes

4.2 Simulation

Simulations is a powerful tool when developing new functionalities for robotic systems and other physical systems. Simulations enables testing of new features in a "risk free" environment without using the actual physical robot. If the simulation model has an accurate representation of the robots physics the development process can be a very effective as new features can be tested in a simulator instead of real testing which is typically costly and challenging to achieve. If the simulation has yielded good results the new features can be tested on the real robot. Kilter has implemented a simulation model of the robot in Gazebo. This simulation model is the basis for all the experiments preformed in this project.

4.2.1 Gazebo

Gazebo is an open source simulator which enables fast development of simulation models. The simulation platform can preform accurate dynamic simulations using ODEs, Bullet, Simbody and Dart, making the simulations accurate and reliable. Gazebo also includes a 3D visualization tool together with simple robot model building tools to visualize the simulations. The simulator also generate sensor measurements with different amount of noise to make simulation of robot navigation more realistic[30]. One of the biggest advantages using Gazebo is the vast ROS support. The ROS 2 package gazebo_ros_pkgs provides an interface for 3D simulations in Gazebo for the ROS 2 environment[31].

4.2.2 Rviz

Rviz is a visualization tool used to display all the information available to the robot. This makes the development of robot application more intuitive since the developer can visualize all the data available. In this project this includes the body frame, the coordinate frames attached to the left and right wheel, the IMU measurement frame, and the GPS measurement frame. This makes the measured sensor data easier to visualize and Rviz also provides transformation between the frames which is useful when translating the measurement into body frame.

4.3 Sensors

The Kilter AX1 has three primary sensors used for navigation, GNSS, IMU, and wheel encoders. These three sensors provide position(RTK-GPS), acceleration(IMU), angular velocity(IMU and Encoders), and linear velocity(Encoders) measurements. In this section a detailed description of each sensor will be provided.

4.3.1 Wheel Encoder

Each of the AX1 robots motor driven wheels have an encoder which measures the rotational motions of the wheels. There are four different types of encoders, mechanical, optical, magnetic, and electromagnetic. The most popular of these is the optical encoder. The optical encoder works by mounting a disk with a given number of slots where light can pass through on the wheel axle. On each side of the disk, there is mounted a light emitter and receiver that create pulses as the wheel rotate. The number of output pulses measured can be converted directly to rotational speed[32]. With the knowledge of the rotational speed, wheel radius and length of the wheel axle, it is trivial to find the linear and angular velocity of the robot.

$$u = R\frac{\omega_r + \omega_l}{2} \tag{53}$$

$$\omega = R\frac{\omega_r - \omega l}{L} \tag{54}$$

where u is the linear velocity of the robot, R is the radius of the wheels, ω_r and ω_l is the rotational speed of the left and right wheel, ω is the angular velocity of the robot and L is the length of the wheel axle.

4.3.2 GNSS and RTK-GPS

Global navigation satellite system(GNSS) is the most common navigation sensor for position estimate of robots. A GNSS system estimates a GNSS receiver's longitudinal and latitudinal position by measuring the time it takes for a GNSS signal to travel from a satellite to the receiver and comparing the time to other satellites. There are four GNSS systems for global navigation: Navstar Global Positioning System (GPS), GLONASS, Galileo, and BeiDou. The most famous known and used is the GPS, but GNSS sensors often combine the systems, improving the sensor accuracy and reliability[9].

The Kilter AX1 robot uses Real-Time Kinematic (RTK) GPS to enhance the GPS position measurement precision. RTK-GPS uses a base station with a known position as a reference point. The mobile robot measurements are compared with the more reliable measurements from the base station and are corrected accordingly. By using an RTK-GPS position measurement with a 1-2 cm accuracy is obtained. One of the problems associated with RTK-GPS and GNSS is that obstacles can impair the signal from the satellite or the base-station which may lead to a drop in accuracy or a loss of the measurement all together[14]. As mentioned in section 3.1.4 the high precision of the RTK-GPS enables the possibility of utilizing the measurements to find the velocity of the robot.

4.3.3 Inertial measurement unit(IMU)

An inertial measurement unit(IMU) sensor consists of a three-axis attitude rate sensor(ARS) (or gyroscope) for measuring attitude rate, and a three-axis accelerometer. The IMU used by the AX1 robot is also equipped with a three-axis magnetometer and some IMUs also include a one-axis barometric pressure sensor. The IMU is not placed in the body frame where the acceleration and angular velocity of the robot is defined, but in a measurement reference frame denoted as {m}. IMU measurements must therefore be translated to the body frame in order to be used by the state estimation algorithm[9]. Transformation matrices and vectors can be be obtained in Rviz. A calibration of the IMU measurements are also necessary to compensate for the bias in IMU measurement.

Attitude rate sensor The original ARS is the gyroscope. A gyroscope uses a rotating wheel that can measure momentum and translates the momentum to angular velocity. Gyros are the most accurate ARSs, but Micro Electro-Mechanical Systems(MEMS) have become popular in recent times due to their much lower price and size. The angular velocity measurements are measured in the IMU reference frame and must be translated to the body frame

$$\boldsymbol{\omega}^b_{imu} = \mathbf{T}(\boldsymbol{\Theta}_{mb})\boldsymbol{\omega}^m_{imu} \tag{55}$$

where $\boldsymbol{\omega}^b_{imu}$ is the IMU measurements represented in body, $\boldsymbol{\omega}^m_{imu}$ is the raw IMU measurements and $\boldsymbol{\Theta}_{mb}$ are the Euler angles associated with the rotation from the IMU frame to the body frame. The ARS measurements are subject to both a random walk bias and white noise which can be represented as

$$\begin{aligned}\boldsymbol{\omega}^b_{imu} &= \boldsymbol{\omega}^b_{nb} + \mathbf{b}^b_{ars} + \mathbf{w}^b_{ars} \\ \dot{\mathbf{b}}^b_{ars} &= \mathbf{w}^b_{b,ars}\end{aligned} \tag{56}$$

where $\mathbf{b^b_{ars}}$ is the sensor bias, and $\mathbf{w}^b_{ars}$ and $\mathbf{w}^b_{b,ars}$ is the measurement ant bias noise respectively. By rearanging this equation the angular velocity can easily be obtained. After the transformation there is still an error in the measurement due to the bias and white noise[9]. An initial bias is found by keeping the robot stationary at a flat surface and averaging the transformed ARS measurement over for example 100 samples. The stationary bias used as an initial bias estimate, but the bias is time variant and therefore continually estimated when the robot is online.

Accelerometer There are many different accelerometers, but they are based on the principle of attaching a mass to a spring and measuring the spring force. The accelerometers measure specific force in three dimensions. Accelerometer measurements are transformed from the measurement frame to the body frame as follows

$$\boldsymbol{f}^b_{imu} = \mathbf{R}(\boldsymbol{\Theta}_{mb})\boldsymbol{f}^m_{imu} \tag{57}$$

The accelerometer defined in body can be described mathematically as follows

$$\begin{aligned}\mathbf{f}^b_{imu} :&= \frac{1}{m}\mathbf{f}^b_{non-gravitational} \\ &= \frac{1}{m}(\mathbf{f}^b_{total} - \sum \mathbf{f}^b_{gravitational})\end{aligned} \tag{58}$$

This equation can be interpreted as the measured force equal to the force caused by acceleration and the gravitational acceleration. With this the equation can be simplified to

$$\mathbf{f}^b_{imu} = \mathbf{a}^b_{imu} - \mathbf{g}^b \tag{59}$$

For a accurate model of the sensor readings the sensor bias and sensor uncertainty due to white noise must be added. Since a flat ground is assumed the gravitational force in the body frame is equal to the gravitational force in the NED frame. With this assumption the accelerometer measurement can be described as

$$\begin{aligned}\mathbf{f}^b_{imu} &= \mathbf{a}^b_{imu} - \mathbf{g}^n + \mathbf{b}^b_{acc} + \mathbf{w}^b_{acc} \\ \dot{\mathbf{b}}^b_{acc} &= \mathbf{w}^b_{b,acc}\end{aligned} \tag{60}$$

From this equation the acceleration in body can easily be found. An initial bias estimate and can be found the same way as the ARS bias and the walking bias must be estimated when the robot is online[9].

Magnetometer A magnetometer measures the magnetic field of the earth. The earth's magnetic field is similar to a dipole with one termination point at the north pole and one at the south pole. Along with the globe, there are field lines with varying magnetic strength. This variation in magnetic force is called inclination. When using a magnetometer to measure the heading, the declination angle must be considered. The equation for finding the heading using a magnetometer is

$$\psi = \psi_m + \delta \tag{61}$$

where ψ is the actual heading, ψ_m the measured heading and δ is the declination angle. The declination angle at a specific longitudinal and latitudinal coordinate is found in many mapping services. A magnetometer measures the magnetic field on the three-axis. After rotating the magnetometer measurements to body with

$$\mathbf{m}^b_{mag} = \mathbf{R}(\boldsymbol{\Theta_{mb}})\mathbf{m}^m_{mag} \tag{62}$$

the magnetometer measurements can be described mathematically by

$$\mathbf{m}^b_{mag} = \mathbf{m}^b + \mathbf{b}^b_{mag} + \mathbf{w}^b_{mag} \tag{63}$$

The magnetometer is subject to bias and must therefore be calibrated before use. With the flat ground assumption the magnetometer measurement $m^b = [m_x, m_y, m_z$ can be used to find the heading

$$\psi_m = -atan2(m_x, m_y) \tag{64}$$

and by adding the declination angle the heading ψ is obtained[9].

5 Implementation

5.1 Kalman filter

The Kalman filter(KF) is a recursive filter used to estimate a linear or nonlinear dynamic system's states. The filter combines state measurements and state predictions to obtain an accurate state estimate. Kalman filters enables the navigation schemes such as pure INS since the Kalman filter is capable of reconstructing reconstructing unmeasured states. Kalman filter's also acts a coloured and white noise filter making the Kalman filter an ideal choice when designing navigation systems. The filter can also be operational even if measurements are lost, functioning as a predictor by using the state prediction as the state estimate[9]. The kinematic model of the AX1 robot is a nonlinear and can be written as

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}) \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}, \mathbf{u}) + \epsilon \end{aligned} \tag{65}$$

where $f(x, u, w)$ and $h(x, u)$ are nonlinear vector fields, where the vectos w and ϵ is the process and measurement zero-mean white-noise associated with the covariance matrices

$$\begin{aligned} \mathbf{w} &\sim N(0, \mathbf{Q}) \\ \mathbf{v} &\sim N(0, \mathbf{R}) \end{aligned} \tag{66}$$

States of the system must be discretized to use the Kalman filter. The discretized state space model for a nonlinear system can be obtained as follows

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \Delta t \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \mathbf{0}) \\ \boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) \end{aligned} \tag{67}$$

where x_{k+1} is the predicted state at the next time step, x_k is the states at the current time step and ΔT is the time step. Here forward Euler integration is used to discretize the vector field $\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \mathbf{0})$ and white noise has been neglected[9]. This discretized model is used to give a prediction of the states. The accuracy of the prediction is dependent on the accuracy of the model. This prediction usually gives good estimates of the states over short period's of times, but fails to predict the states over longer periods of time. Therfore the Kalman filter also utilizes sensor measurements to correct the predicted states found by the model. The Kalman filter consists therfore of two steps, one prediction step where the model is used to predict the states in the next time step and one update step where the measurements are used to correct the prediction.

As discussed in section 4.3 sensors are subject to different levels of noise depending on the sensor. One cannot be certain therefore that the sensor measurements portray the true state of what is being measured. The kalman filter gives an estimate of the states based on both the model and the measurements in order to compensate for the inaccuracy of both methods for obtaining a good estimate of the current state. The state estimate given by the Kalman filter is not an average of the state measurements and the predicted states, but a factor based on the predicted accuracy of the state prediction and the covariance of the measurements. For example if there are large model uncertainties, but the sensor are accurate, the Kalman filter will "trust" the sensor measurement more than the state prediction. The covariance matrix used to estimate the uncertainty in the state estimates are updated in each iteration of the Kalman filter algorithm to give continuous estimation of the accuracy in the state estimate. The extended Kalman filter algorithm can be decribed as follows[5]

Algorithm 1 Extended Kalman filter algorithm

```
**Initialize**:$\mathbf{x_0}^-, \mathbf{P_0}^-$ and constant measurement matrix $\mathbf{C}$

State prediction:
$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1} + \Delta T \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)$

State prediction jacobian:
$\mathbf{F}_k = \mathbf{I}_n + \Delta T \frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, 0)}{\partial \mathbf{x}} |_{\hat{\mathbf{x}}_k^-, \mathbf{u}_k}$

Covariance prediction:
$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}$

Kalman gain:
$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{C}^T (\mathbf{C}\mathbf{P}_k^- \mathbf{C}^T + \mathbf{R})$

State estimation:
$\mathbf{x}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-)$

Covariance Estimation:
$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{C})^T + \mathbf{K}_k \mathbf{R} \mathbf{K}_k^-$
```

5.2 Tuning

5.2.1 Kalman Filter tuning

For the Kalman Filter to yield good results, the covariance matrices **Q** and **R** associated with the process and measurement noise respectively, must be determined. A good estimate of the covariance measurement matrix **R** can typically be found in the datasheet for the sensors. The covariances given in the datasheet can be placed on the diagonal of the covariance matrix and will typically yield good results. The process covariance matrix on the other hand is not as straight forward. Its is very difficult to determine the process noise without through analysis. Tuning of the process covariance matrix is more of an art than a science, and is based mostly on experience and knowledge of the process and the important characteristics of the system. The intuition is that high values in the process covariance matrix yields an unstable, but fast convergence of the states in the Kalman Filter. Small values consequently yield a stable filter with slow convergence. The optimal values for the process covariance matrix depends on the system and what characteristics that is important to the system. If the Kalman filter models a slow porcess where accuracy is important, then a **Q** matrix with small values is a good choice. If one where to use a Kalman Filter to model a drone, high values in the **Q** matrix is the best choice. In figure **??** and 7 two different configurations of the **Q** matrix can be observed.

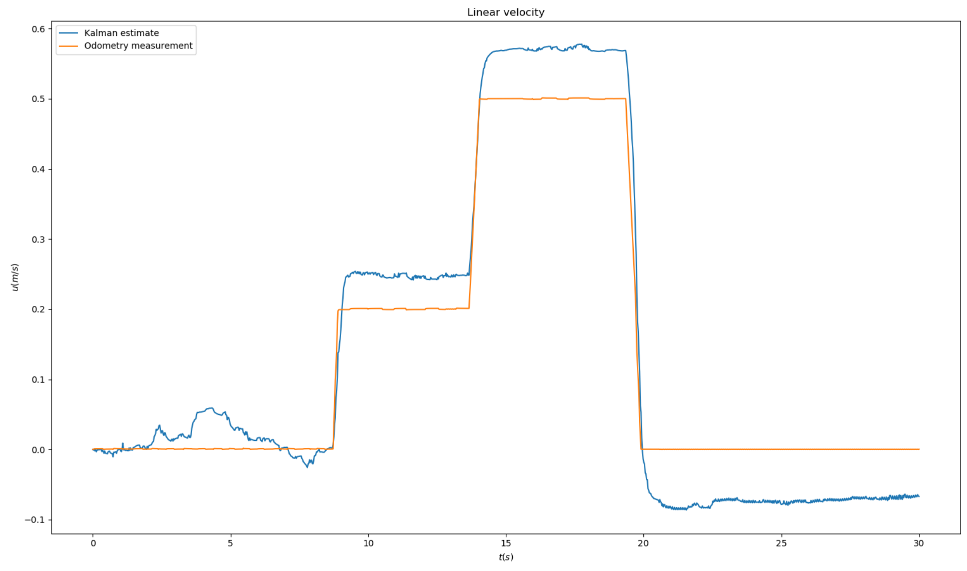


Figure 6: Q = 0.1I

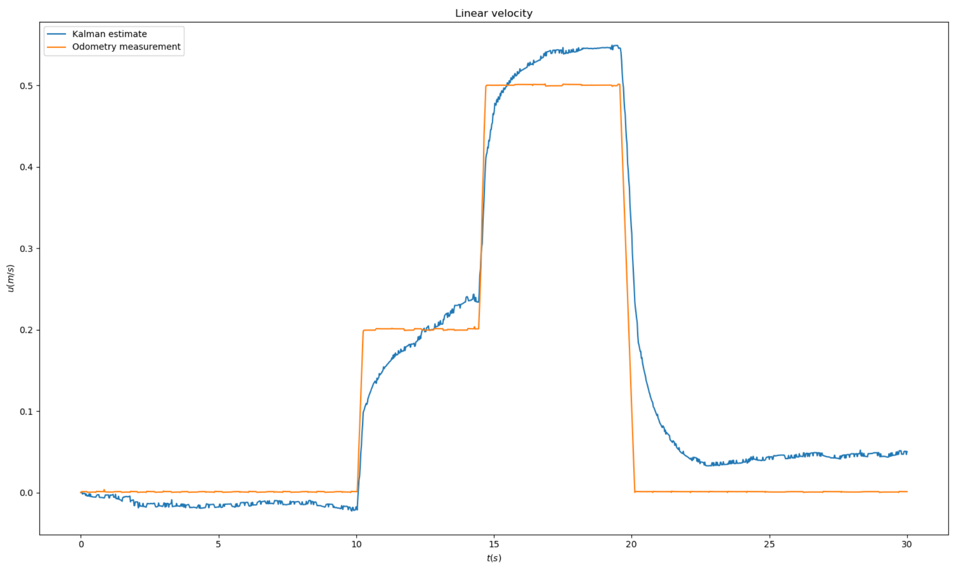


Figure 7: Q = 0.01 I

As one can observe the best choice for the process covariance matrix is $Q = 0.1I$ where I is the identity matrix. This choice gives a fast and stable response to the changes in velocity. One can also note the error in both simulations which is a result of the error obtained by using pure INS.

The covariance matrix **P** is constantly estimated in the Kalman Filter but good initial condition is important for the initial stability of the filter. An easy way to determine the initial covariance is to keep the robot stationary, and use the converging covariances as the initial covariance estimate. This ensures that the state converge quickly and the state estimation remains accurate during the beginning of the startup sequence.

5.2.2 State estimation tuning

In section 3.1.3 the mixing variables that determine which method for estimating the states is to be used is presented. The mixing variables are double-sided sigmoid functions with tunable parameters a, b, and the linear velocity mixing variable also uses p. Parameter a determines the slope of the function, which means how fast the algorithm switches from one state estimation method to another. The b parameter is the threshold for the change between the estimation methods and is tuned to ensure that the mixing variables do not react to the noise of the measurements used to estimate the states. Thus, in some sense, parameter b is a noise filter that is built into the model and limits the interference of sensor noise on the state estimation. Unfortunately, the parameter also comes with some downsides discussed in section 6. The parameters used in the simulations is given in table 1

	C1	C2	C3	C4
a	500	100	100	500
b	0.01	0.2	0.1	0.1
p	0.01	-	-	-

Table 1: Mixing variable parameters

The parameters have been found from extensive testing, and an illustration of all the mixing variables can found in figure 8. It is easy to observe the effect of the different parameters. The mixing variable $C1$ is steep and slim while $C2$ is shallower and larger.

For fast switching between two state estimation methods, the a parameters are large which makes state estimation rarely a combination of state estimates but a pure estimate from one method. The b variable must be large enough that noise does not trigger this change. As one can observe from figure 9 where the mixing variable b_1 is very small, the mixing variable is reacting to the noise of the system. In figure 10 on the other hand the mixing variable holds a slightly higher value and the mixing variable does not react to the sensor noise. The b variable should be as small as possible while not reacting to the noise of the system and at the same time be as small as possible in order to detect slippage as fast as possible.

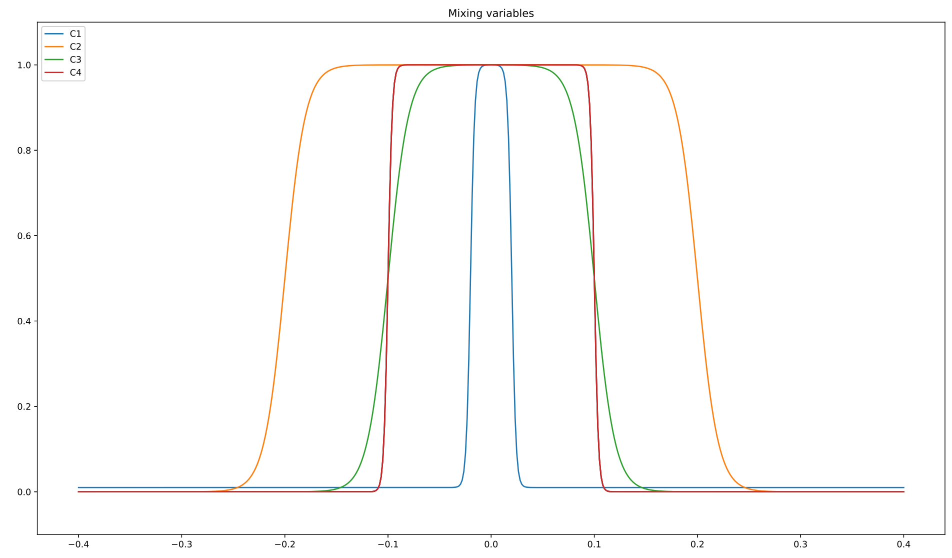


Figure 8: Mixing variables

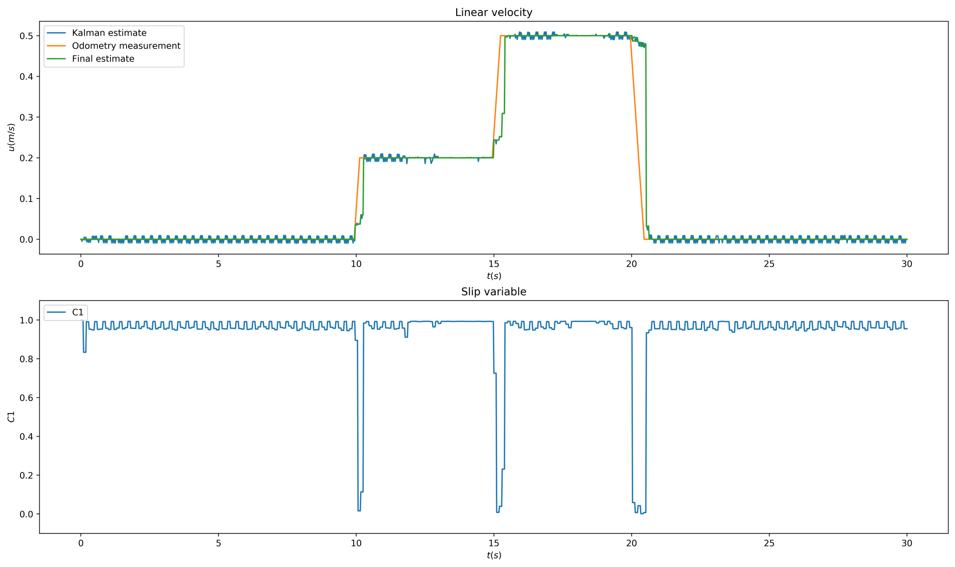


Figure 9: b1 $= 0.01$

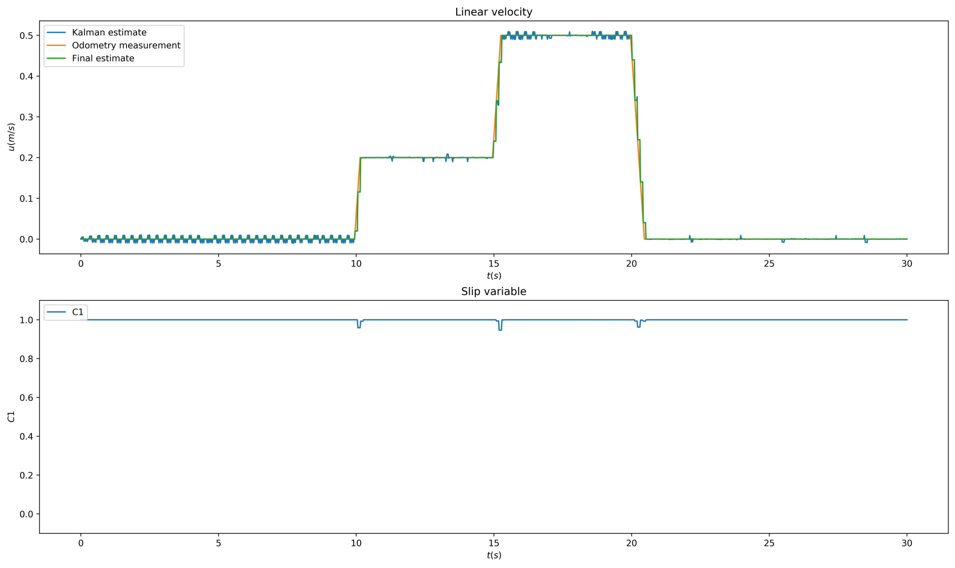


Figure 10: b = 0.03

5.3 Kalman filter testing

Test scenarios are implemented to measure and validate the performance of the state estimation algorithm. Three different test scenarios where the commanded velocity of the robot is changed to move the robot in desired motions like motions performed in a field have been developed. The performance will be measured based on the algorithm's ability to detect slip, estimate the correct state, and the stability of the estimate. Some states like linear velocity are more critical than other states. Prioritization of performance of these states is therefore in order. The test scenarios are simulated with three levels of friction to understand better how the state estimation algorithm performs and analyze its limitations. The robot usually operates velocities between 0.2m/s-0.5m/s and angular velocities between 0.2m/s-0.5m/s, which is reflected in the tests. The tests used to validate the performance of the state estimation algorithm are given below.

- **Linear Velocity Test:** Change the commanded linear velocity from 0.0m/s to 0.2m/ to 0.5m/s to 0.0m/s with an interval of 5 seconds between each change.
- **Angular Velocity Test:** Commanded angular velocity is changed from 0.0m/s to 0.2m/s to 0.5m/s to 0.0m/s with five seconds between each interval change.

- **Circle Test:** The linear and angular velocity is initially set to 0.2m/s, then the linear velocity is set to 0.5m/s, then both the linear and angular velocity is changed to 0.0m/s. All the changes are done with an interval of 5 seconds.

These tests represent typical motions of the robot when operational in a field. The tests simulate forward movements from one end of the field to the other. During the tests, the robot is subjected to friction levels that are either very high, very low, or realistic to see how the state estimation behaves under different circumstances. For example, when performing the tests with high friction, the robot will move in the same way as the commanded velocity. However, when friction is very low, the robot will be almost at a standstill with the wheels turning. With realistic friction, the robot will slip during acceleration, but once the desired velocity is reached no slipping will be present. Different levels of friction in Gazebo can be implemented by changing the mu1 and mu2 parameters in the SDF Gazebo file. For low friction the friction parameters used are mu1 = 0.0001 and mu2 = 0.0002 and for high friction mu1 and mu2 are default Gazebo friction. Real world friction coefficients between rubber wheels and soil can be estimated to be around 0.02 based on the table found in [33].

6 Results and discussion

In this section, the results from the test cases given in section 5.3 are presented. Simulations are performed, giving the same commanded velocity to the robot both with very high, very low, and realistic friction settings in the Gazebo simulator. Simulations are done to test the performance of the state estimation algorithm given in 3.1 by triggering behavior where the performance can be analyzed easily. The actual states of the robot where unavailable in the Gazebo simulator and the performance of the state estimate can only be evaluated by observations done during the simulations.

First, the linear velocity test results will be presented, followed by the angular velocity test, then the circle test. Only the states that are influenced in each test are presented, but all the simulation data for each test can be found in the appendix.

6.1 Linear Velocity Test

In this section, the linear velocity test is preformed with low, high and realistic friction settings. Both the the proposed method for estimating linear velocity found in [5] using the IMU to detect slip and the method using the GPS velocity estimate to detect slip is shown for the three levels of friction. A comparison of the two methods for detecting slip will be conducted, and a concussion as to which slip detection method performs the best in simulations, will be given at the end of this section.

6.1.1 High friction

With high friction, the robot's actual velocity and the commanded velocity are equal. The optimal velocity estimate is therefore using the odometry measurement for the entirety of the linear velocity test.

In figure 11 the linear velocity test is performed with very high friction in Gazebo and using the IMU to detect slip. One can observe the Kalman estimate, final estimate, odometry measurement for linear velocity, and the slip variable C1. One can observe that the encoder measurement, Kalman filter, and final estimate closely follow the odometry measurement. The slip variable C1 is steady at a value of one, meaning that the encoder measurement is used as the final state estimate during the entire test.

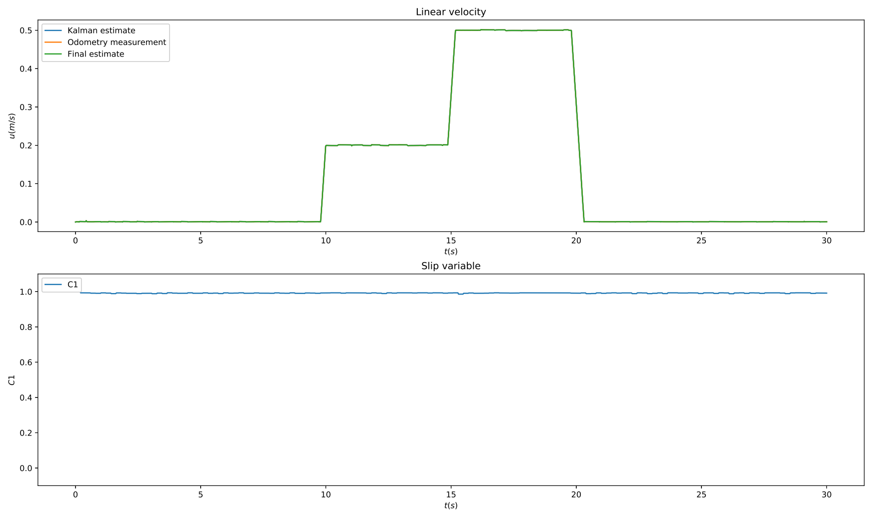


Figure 11: Linear velocity test with high friction using IMU for detecting slip

In figure 12 the same test is performed as in figure 11, but the slip detection is performed with the GPS velocity estimate. As one can observe, the change in velocity triggers a change in the mixing variable C4 from 1 to 0, meaning the INS is used to estimate the velocity in the intervals where C4 is 0. The trigger of the mixing variable C4 results from the delay in the GPS velocity estimate, which uses the average velocity over the last six samples. Therefore it takes sometime before the GPS estimates can react to the change in velocity, and the difference between odometry and GPS velocity estimate becomes large enough to trigger a change in the mixing variable C4.

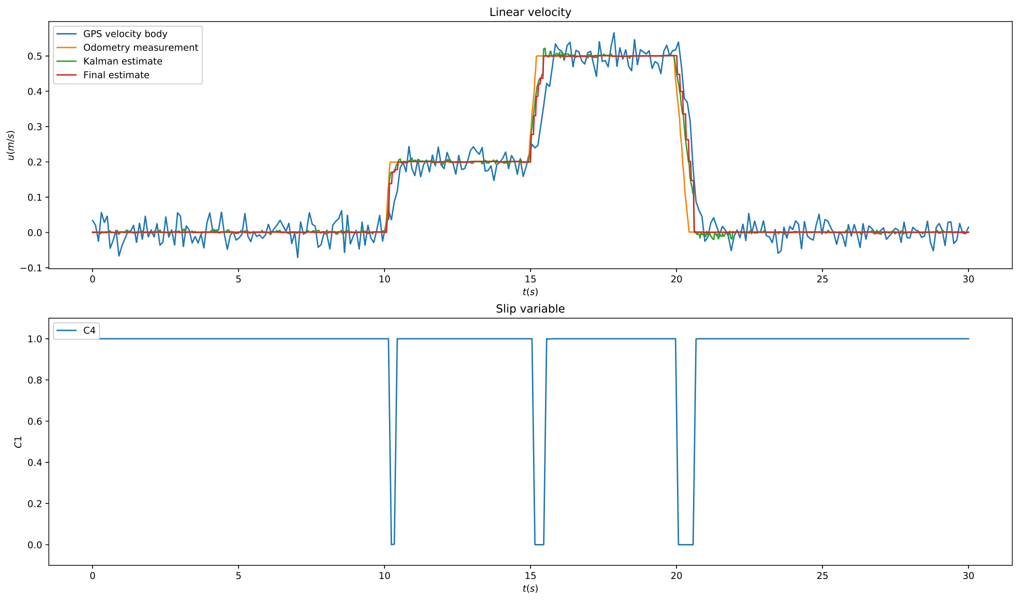


Figure 12: Linear velocity test with high friction using GPS for detecting slip

6.1.2 Low friction

When the robot is subject to very low friction and performs the linear velocity test, the robot is almost stationary while the wheels are spinning. The optimal velocity estimation is to use the INS as soon as the wheels start spinning and switch back to using the odometry measurement when the wheels stop turning.

In figure 13 the linear velocity test is performed with very low friction, and the IMU is used for slip detection. As one can observe the odometry measurement, Kalman filter estimate, and state estimate still follow the commanded velocity closely. The slip variable C1 spikes in the areas where there is a change in commanded velocity. In the interval where there are spikes in C1, the final state estimate uses INS as one can be observed from the flat velocity estimates in these areas. The slip detection is also somewhat delayed due to the slip detection being performed over an interval of 10 samples. As one can observe the flatting of the state estimate happens some time after the change in commanded velocity.

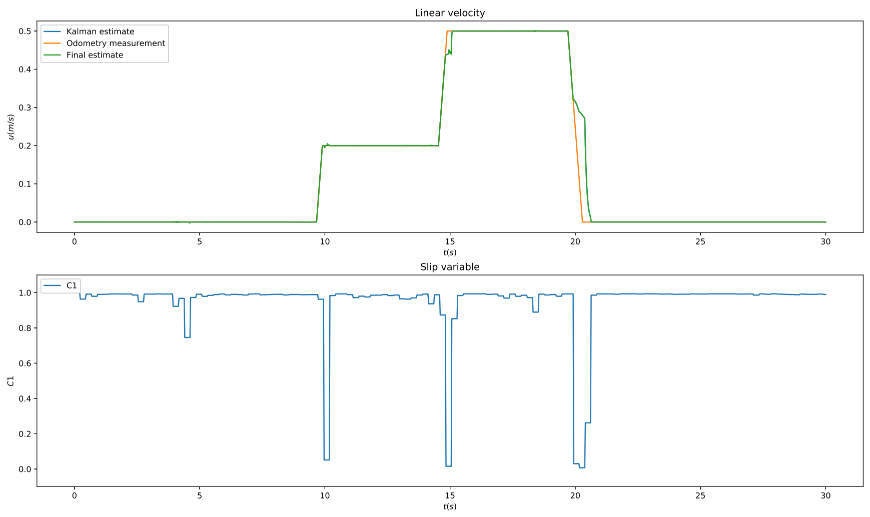


Figure 13: Linear velocity test with high friction with IMU

In figure 14 the same test is performed using the GPS velocity estimate for slip detection. The mixing variable C4 jumps from 1 to 0 when the encoder measures a velocity of around 0.1 and remains zero until the encoders measures a velocity of around 0.1. C4 is zero in this interval, meaning that the INS is used as the final state estimate for linear velocity during this time. The state estimate does not change to INS before the odometry measurement is over 0.1 because of the mixing variable tuning parameter b_4, determining the threshold for where the odometry measurement should be used. The tuning parameter is set to b_4=0.1, meaning that INS is not used until the difference between odometry measurements and GPS velocity is over this threshold.

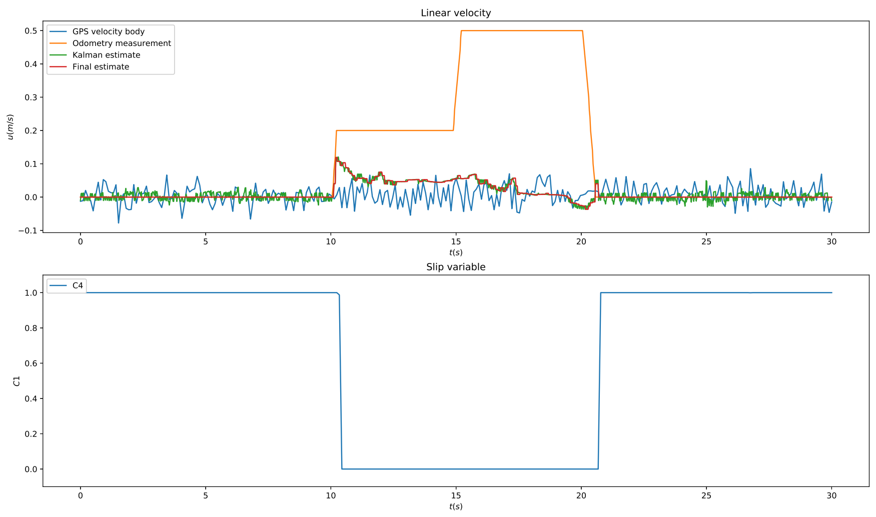


Figure 14: Linear velocity test with high friction

Realistic friction With realistic friction settings, the robot will have some wheel spin during acceleration. When reaching the commanded velocity, the wheel spin ends. The ideal case is for the state estimation scheme to use INS while the robot is accelerating and then switch to using odometry measurements when the robot reaches the commanded velocity.

Figure 15 shows the simulation data for the linear velocity test when the IMU is used to detect slip. The simulation starts with some smaller spikes in the mixing variable. Since the spikes are a value between 0 and 1, the final estimate is a combination of the odometry measurement and the state estimated by the INS. As one can observe the slip detection is triggered when the robot is accelerating and the state estimation uses INS during this time. Despite wheel spin being present the robot is still accelerating and the final state estimate show a more moderate velocity than the commanded velocity but a steeper curve than the one found in figure 13 where the velocity of the robot is close to zero.

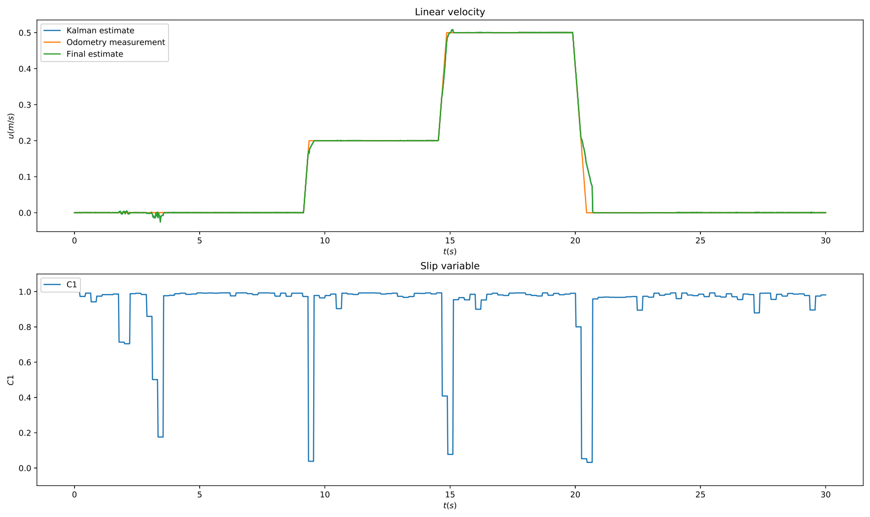


Figure 15: Linear velocity test with high friction with IMU

The simulation of the linear velocity test where the friction settings are realistic and where GPS velocity is used is shown in figure 16. The velocity estimate uses INS in the intervals where slip is detected and switches back to using the odometry measurements when the robot is close to completing the acceleration. Again there are spurts in the velocity estimate as the difference in GPS and odometry measurements crosses the threshold of the tuning parameter b_4. The jumps can be observed in all the intervals where the robot is accelerating.

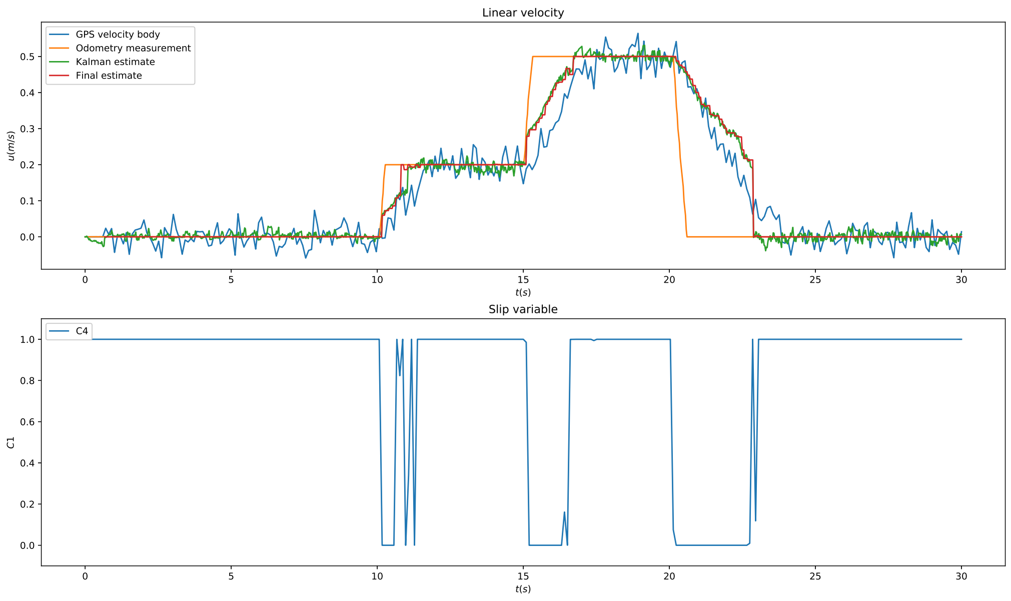


Figure 16: Linear velocity test with normal friction

6.1.3 Comparison between slip detection using IMU and GPS velocity

This section will discuss the performance, advantages, and disadvantages of detecting linear slip with velocity estimated from the GPS and acceleration measured from the IMU for the linear velocity test.

When the robot is subject to high friction, the slip detection using IMU performs well with no detected slip and uses the odometry measurement for the entirety of the simulation, which is ideal.

Slip detection using GPS velocity, despite using the INS while the robot is accelerating, performs well. When switching to INS, the velocity estimate is very close to the odometry measurement. The performance of the INS is very reliable during accelerations, and the result, as can be observed in figure 12 is very satisfactory. So satisfactory that the real-world performance of running the robot on a high friction surface, the difference in using the IMU or GPS velocity to estimate linear velocity would be negligible.

Slip detection performed on a very low friction surface should be an ideal case for slip detection to perform well. If the robot is stuck while the wheels are still turning, slippage must be detected so the robot can be stopped and avoid the whole system from failing. A system can be developed based on slip detection to give warnings if the estimated velocity of the robot is close to zero over extended periods indicating that the robot has gotten stuck the robot can be turned off and a warning can be sent to the operator.

Using the IMU to detect slip with very low friction, the result is an estimated velocity closely related to the odometry measurements. This estimate yields a massive error in the linear velocity estimation that should be close to zero. The reason is that the IMU slip detection can only detect slippage when there are changes in the odometry measurements or the robot's velocity. For the low friction simulation, the robot is almost at a standstill, and the wheels are turning at a constant velocity meaning the tangential velocity is close to zero. Slippage is detected by the difference in tangential velocity and acceleration. If the difference between the tangential velocity measured by the encoders and the accelerometers is zero, no slip is detected. The slip is detected throughout ten samples. With the change in the velocity of the wheels happening so fast, slip detection cannot work very effectively and is delayed, as one can observe from the delayed response of the mixing variable C1.

Slip detection using GPS velocity performs well with a switch to INS when the commanded velocity is initialized. However, there is a slight delay before the slip is detected, as one can observe from figure 14. One can observe that the final estimate uses the odometry measurement in the interval where the odometry measurement is between 0 and 0.1, corresponding to the mixing variable threshold b_4=0.1, before switching to INS. This means the initial INS estimate is 0.1. Velocity estimation using INS relies on the IMU measurements of the robot and means that the INS is unable to perceive the actual velocity of the robot, which is zero. The real acceleration of the robot is close to zero for the entire simulation, and one would expect the estimated velocity of holding a constant value of 0.1 during the entire simulation. This is, however, not the case.

The IMU used in the INS is subject to large amounts of noise and drift and is therefore subject to significant levels of uncertainty. In order to obtain an estimate of the robot's forward linear velocity, the linear, sideways, and angular velocity estimates are combined and integrated. Based on the fact that all these estimates stem from the IMU measurements, which are subject to much noise, it is not surprising that the performance of the velocity estimate is unstable.

Long intervals where the INS is used result in a substantial drift in the velocity estimate. In the simulation shown in figure 14, the estimate given by the INS is converging towards zero(the actual state of the robot), but the INS might as well have drifted the other way, estimating a velocity higher than the one measured by the encoders. It becomes clear that the reliability of the INS is a significant problem if the slip detection is to be implemented on the real robot where the IMU measurements will be subjected to other disturbances from rugged terrain where the AX1 robot is operating. In this simulation, the INS is used for approximately 20 seconds and can result in significant errors during this time. If the slip detection and state estimation algorithm are to be used, a limit on how long the robot can use pure INS in its current iteration must be set. Another option is to improve the INS by adding measurements. An option is to add the position and velocity estimate from the GPS measurement. The GPS velocity estimate has a large uncertainty, but its addition could yield good results during stationary slippage when the GPS velocity estimate is stable. However, when the robot is accelerating, the GPS velocity estimate would be subject to a delay and could cause a delay in the estimate.

Slip detection performed on the linear velocity test with realistic friction will ideally use the INS for velocity estimation during acceleration as the wheels are observed to be spinning during acceleration.

The linear velocity test with realistic friction using IMU for slip detection performs inadequately, with slip detected over short intervals where the odometry measurements change. The result is therefore not correct compared to what is observed during simulations. The problem is again connected with the delay in the detection and the fast change in odometry measurement. In addition, the slow acceleration of the robot due to slippage makes the difference between accelerometer readings and odometry measurements too small to detect slippage over a longer interval. The IMU slip detection is also somewhat unstable, as one can observe from the distorted line of C4 during constant odometry readings. The slip detection variable b_1 is tuned very aggressively, and is triggered by the large amount of noise in the IMU measurements.

The results are satisfactory when using GPS velocity to estimate slip. The one flaw with the method is the jump in velocity estimate when the estimated velocity of the GPS and odometry measurement crosses the threshold b_4. The INS estimates are reasonable if the GPS velocity estimates are used as a performance reference. It can be observed that the estimated velocity and the GPS velocity have a similar slope with the GPS velocity slightly ahead of the INS as expected because of the GPS velocity delay. The only deviation from this observation happens during the deacceleration when the commanded velocity changes from 0.5 to 0.0. Here the INS velocity estimate lags behind the GPS velocity and indicates that the INS's performance is suboptimal during longer intervals, as explained earlier. The difference causes a massive jump in the velocity estimation and is not optimal. The issue could be improved by a less aggressive tuning of the a_4 parameter. A less aggressive tuning would result in a smoother transition by combining the INS and odometry measurements when the threshold b_4 is crossed. However, this would also result in a slower slip detection, which would be unfortunate in circumstances like the simulation with low friction.

By analyzing the difference in performance between the slip detection using IMU measurement and GPS velocity estimate, it is clear that the slip detection using GPS velocity estimation is superior. If slip is present, the fast changes in the odometry measurements make it hard for IMU slip detection to detect slip. The IMU cannot detect stationary slip like in the low friction test and makes it very hard to recommend using this method for the AX1 robot, which relies on detecting slip if it is stuck on the field. Slip detection using the GPS velocity estimate performs well in all the tests. The fast switch between the velocity estimates during the test with realistic friction might be the state estimations' most significant problem. Still, the problem can be solved with less aggressive slip detection parameters. Other methods for estimating the GPS velocity for a smoother estimation might also be a solution for making the method more reliable. The INS is also a concern, and improvements such a drift compensation and noise filtering might be a solution for a smoother and more reliable velocity estimate when using INS.

6.2 Angular velocity test

In this section the angular velocity test with high, low and realistic friction settings is presented.

High friction With high friction the optimal is for the state estimate to use the odometry measurement during the entire angular velocity test.

In figure 17 the angular velocity test with high friction is performed. The slip variable is almost at a constant value of one with a few minor spikes that do not have a tangible impact on the performance of the state estimation. The final estimation and the odometry are therefore almost identical during the entire simulation.

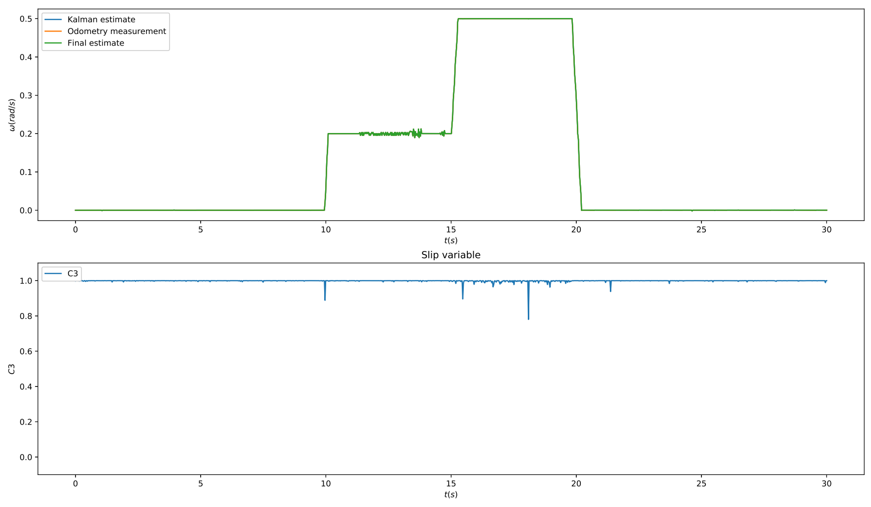


Figure 17: Angular velocity test with high friction

Low friction For low friction the optimal estimate is the same as for linear velocity with the optimal behavior being a switch to INS as the commanded velocity is initialized.

In figure 18 the friction in the simulator is very low. As can be observed, there is a slight delay in the slip detection when the commanded angular velocity is changed from 0 to 0.2. Shortly after that, the state estimation algorithm switches to using INS, and the estimate converges to zero before yielding another spike in the estimate when the commanded angular velocity is changed from 0.5 to 0.0.

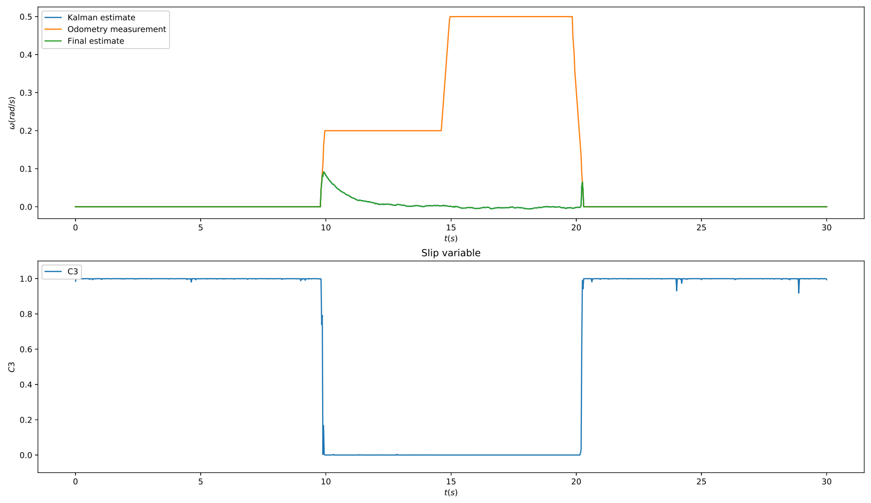


Figure 18: Angular velocity test with low friction

Realistic Test The angular velocity during the angular velocity test with realistic test is shown in figure 19. The behavior of the state estimation is similar to the linear velocity estimate in the linear velocity test. During acceleration, the state estimation algorithm switches from using odometry measurements to INS for angular velocity estimation with a delay caused by the mixing parameter b3. The delay is present every time the robot is accelerating.

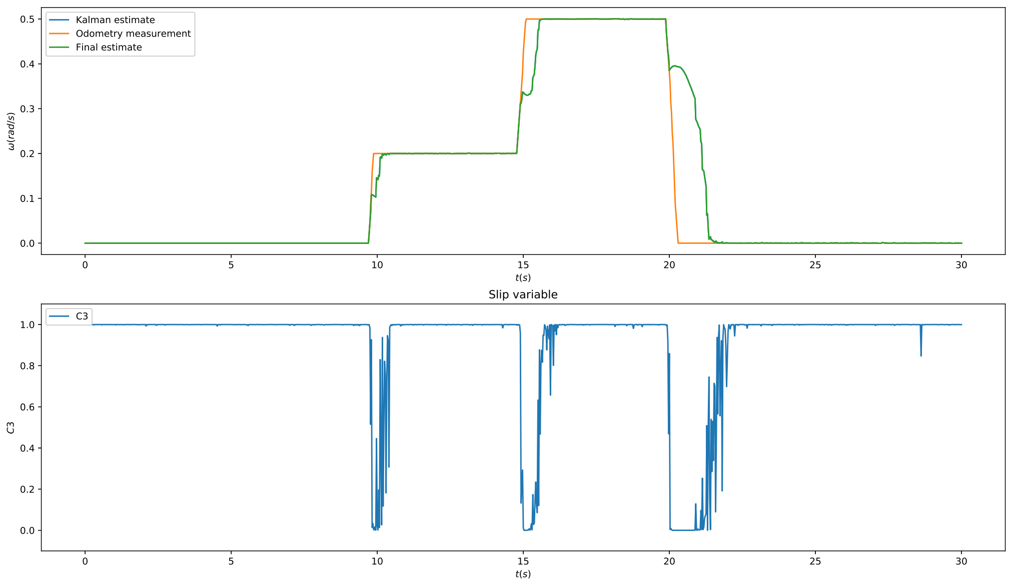


Figure 19: Angular velocity test with normal friction

6.2.1 Discussion

The angular velocity test performed with high friction yield a good result as the mixing variable was almost at a constant value of 1 during the entire simulation. The odometry measurement, as is the optimal estimate, was consequently used. The reason for the excellent performance is that the odometry angular velocity measurements used are very reliable. The direct comparison between the odometry measurement and the angular velocity measurements will therefore yield a very small deviation between the two measurements. The angular velocity measurement follows the odometry measurement closely, as can be observed in figure 17 in the appendix, resulting in a very stable slip detection.

Similar to the linear velocity test with low friction, there is a slight delay in the switch to INS in the angular velocity test. The delay results from the tuning parameter b3 that stops the INS from being used until the difference between the odometry measurement and INS angular velocity estimate precedes the b3 parameter. In this simulation, the parameter b3 is 0.1, and as can be observed from figure 18 the state estimation algorithm switches to INS when the estimated angular velocity reaches 0.1.

Contrary to using INS for linear velocity estimation, the performance of the INS for angular velocity is very good. The estimate converges to zero, which is the correct estimate. The tunable process noise parameters in the Kalman filter used, results in a stable convergence. The reason for the superior performance of the angular velocity estimation over the linear estimation is that the angular velocity measurements have lower noise and are used directly and not integrated.

6.3 Circle test

In this subsection the circle test is performed. The circle test is designed to trigger changes in multiple states to analyse if the performance of the state estimation is affected. Like the other tests, the circle test will be performed with high, low and realistic friction settings. The linear, lateral and angular velocity are all affected during the circle test and will be presented for each friction setting.

High friction Figure **??** shows the forward linear velocity during the circle test with high friction. Like the linear velocity test, there are some spikes in the mixing variables during acceleration where INS is used as the state estimate. A couple of spikes can also be observed when the robot has a velocity of 0.5m/s.

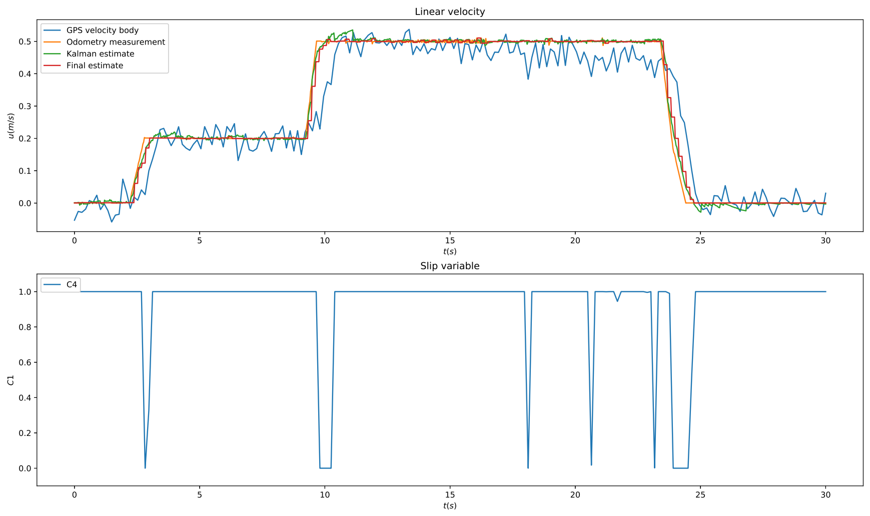


Figure 20: Linear forward velocity in circle test with high friction

In figure 21 the angular velocity during the circle test with high friction is shown. One can observe that the mixing variable is relatively stable at a value of 1, hence the odometry measurement is used during the entire test. Some vibrations in the estimate can also be observed when the estimated velocity is 0.5m/s.

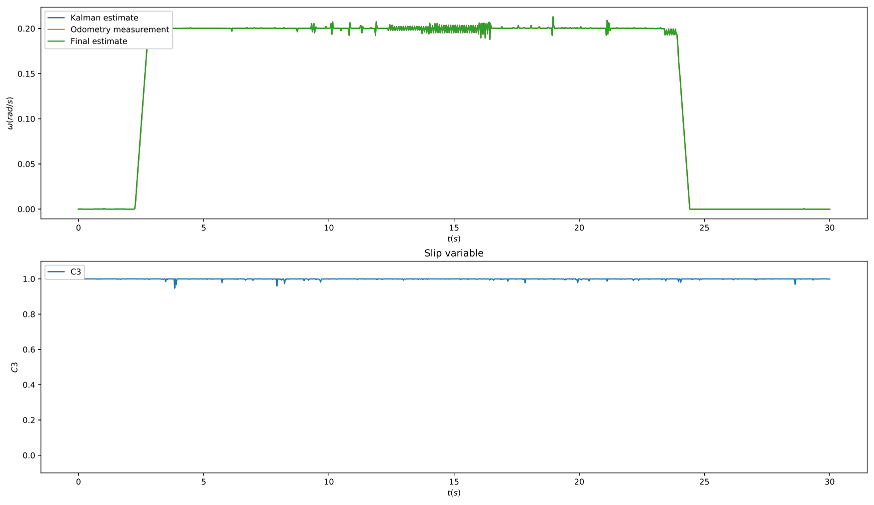


Figure 21: Angular velocity in circle test with high friction

Figure 22 shows the estimated lateral velocity of the robot during the circle test with high friction. As can be observed there are many spikes in the state estimate as the mixing variable C2 dips below 1 but the spikes are small with a maximum value of 0.005 and can therefore be considered insignificant.

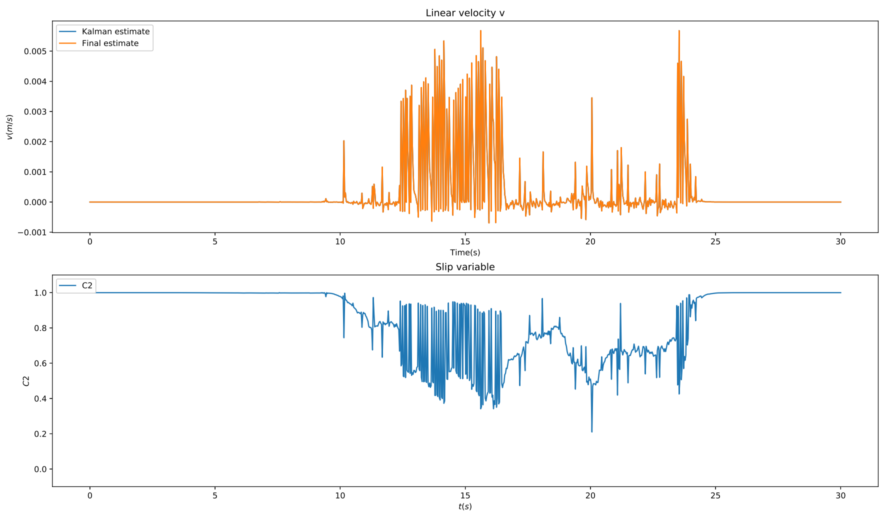


Figure 22: lateral velocity in circle test with high friction

Low friction Figure 23 shows the linear velocity estimation on a low friction setting. As can be observed the mixing variable C4 changes from 1 to 0 when the odometry measures a linear velocity of 0.1. The INS is used until the encoders measures a velocity of 0.1 and then switches back to using odometry measurements.

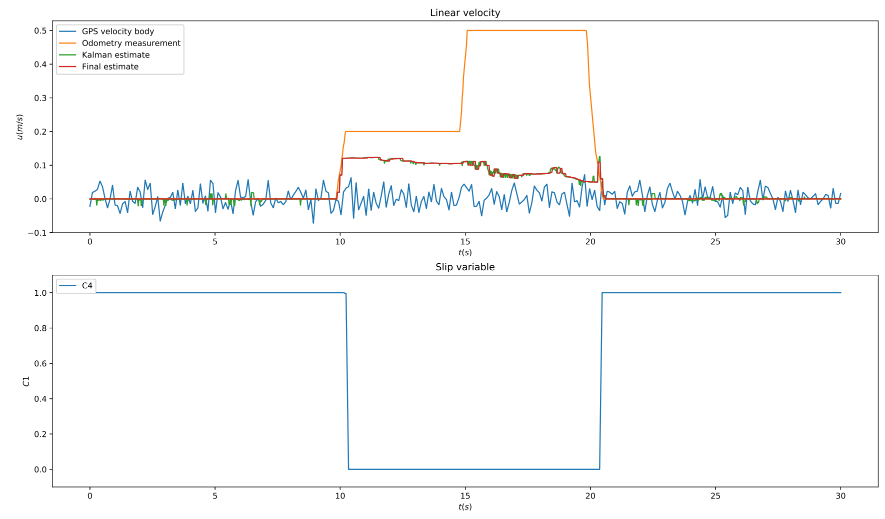


Figure 23: Forward linear velocity in circle test with low friction

In figure 24 the measured and estimated angular velocity of the robot during the circle test with low friction is presented. As for the angular velocity test the slip variable switches from one to zero when the odometry measurement reaches 0.1 and the final state estimation starts to use the INS before switching back to using the odometry measurement when it measures 0.1.

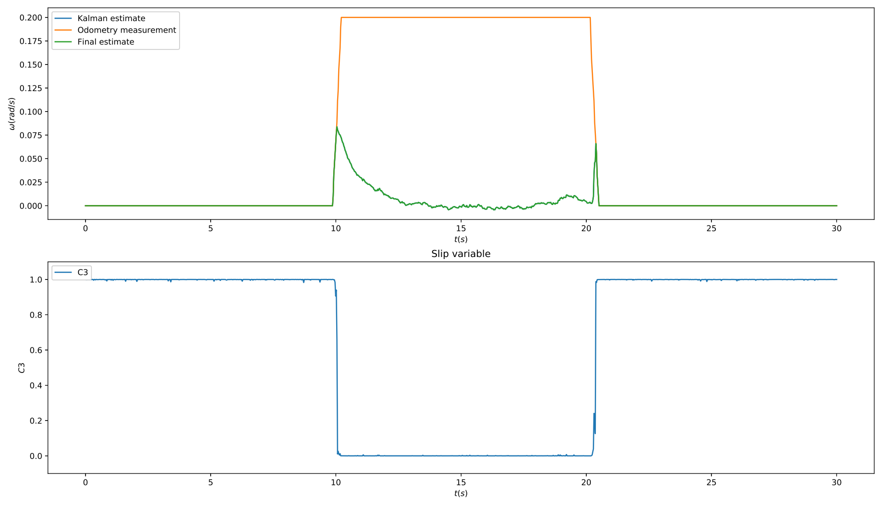


Figure 24: Angular velocity in circle test with low friction

Figure 25 shows the lateral angular velocity estimate of the robot during the circle test with low friction. The final state estimate and the Kalman filter estimate remains zero during the whole test as the slip variable C2 remains constant during the test.

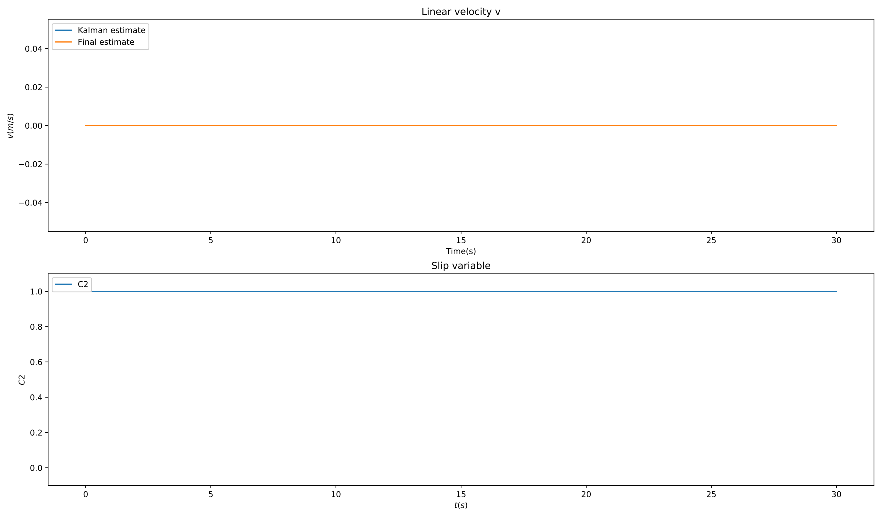


Figure 25: Angular velocity in circle test with low friction

Realistic friction The linear velocity in the circle test with realistic friction shown in figure 26 show much of the same behavior as found in the linear velocity test. The state estimate switches to INS during accelerations. Contrary to the linear velocity test some spikes in the mixing variable C4 can be observed when the linear velocity estimate reaches 0.5.

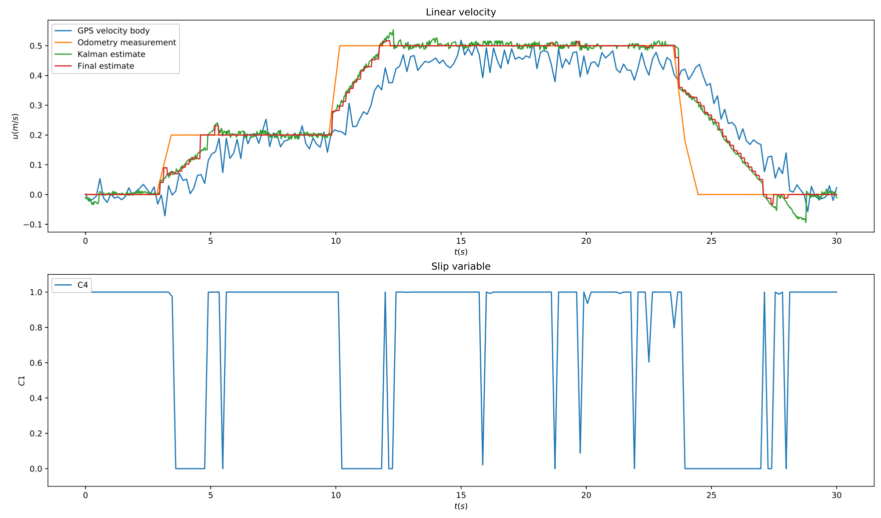


Figure 26: Circle test with normal friction linear velocity

The angular velocity estimate in the circle test with realistic friction is shown in figure 27. The estimate switches for using the odometry measurement to using INS during accelerations when the difference in IMU and odometry measurements reaches 0.1.

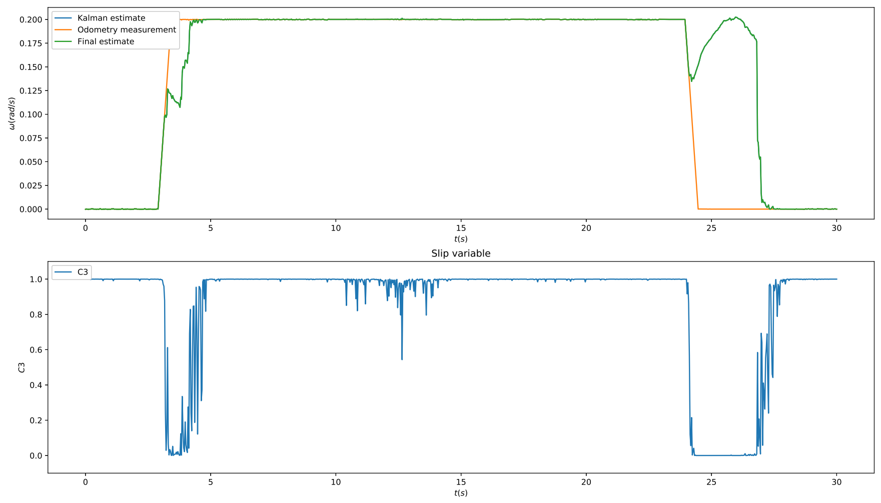


Figure 27: Circle test with normal friction angular velocity

Figure 28 show the lateral velocity of the robot during the circle test with realistic friction. There are spikes in the lateral velocity when the robot's linear velocity is estimated to be 0.5. The largest of the lateral velocity spikes almost reaches 0.25.

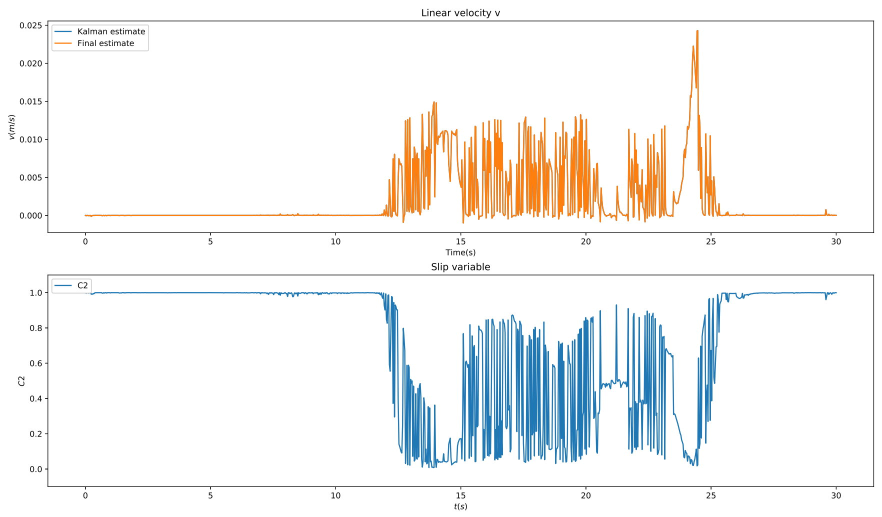


Figure 28: Circle test with normal friction side slip

6.3.1 Discussion circle test

During the circle test, the state estimation performance is very similar to the performance in the linear velocity and angular velocity tests. The delay caused by threshold variables and averaging in the GPS velocity estimate are still present. Lateral velocity has not yet been discussed as lateral velocity is not present in the other testes. However, in the circle test, the robot's lateral force acting on the robot is more significant than in the other tests.

The linear velocity estimate in the circle test is similar to the one performed in the linear velocity test. The one noticeable difference is the spikes in the mixing variable C4 when the velocity is 0.5 during the circle test. As can be observed in figure **??** and 26 the GPS velocity is slightly lower than the odometry measurements suggests. If compared with the GPS linear velocity estimate in figure 12 and 16 the difference becomes evident. The reason behind the slightly inaccurate GPS velocity estimates during the circle test is an error that stems from the transformation from GPS velocity in NED to GPS velocity in the body frame. The origin of the error is the noise and drift from the magnetometer measurement used to estimate the robot's heading. Magnetometers are fairly accurate, but the slight error will accumulate in the transformation and results in a minor error in the GPS velocity estimate.

The lateral velocity estimate is found much like the other state estimates by using mixing variables. Ideally, the robot's lateral velocity is zero when no lateral slip is present. Mixing variable C2 is used to archive a lateral velocity estimate that is only non-zero when the robot is slipping. The mixing variable for lateral velocity is only dependent on the lateral acceleration measured by the IMU. IMU measurements for lateral velocity are subject to large amounts of noise, and the threshold parameter b_2 acts as a noise filter and as a threshold for slip detection. As long as the robot's lateral acceleration is below the threshold b_2, the lateral velocity converges to zero. Thus, large amounts of noise from the IMU do not affect the estimate.

In figure 22 the friction settings are high, and one can observe a slight change in the mixing variable C2, but the change is slight and does not yield a significant impact on the lateral velocity, which is approximately zero. No lateral velocity is expected as the high friction in the simulator makes it nearly impossible for the robot to slip, and no lateral velocity can occur.

In the low friction test, the robot's acceleration is near zero, and the lateral velocity cannot be triggered due to the threshold.

In figure 28 the lateral velocity is triggered when the robot reaches its maximum velocity. Lateral velocity is found by integrating the lateral acceleration and the linear and angular velocity estimates, making for a somewhat unstable estimate. One can observe that the mixing variable is in the region between 0 and 1, resulting from a measured lateral velocity around the threshold. The unstable mixing variable manifests itself in the state estimate because of the unstable mixing variable C2. The estimate jump between 0 and the INS velocity estimate with the same frequency as C2. A solution might be to use a more significant q2 variable which might result in a smoother estimate. The estimate might have also been better if the a_2 variable was tuned more aggressively, making the jump to INS faster. Lateral velocity is at its peak when the commanded velocity changes from 0.2 to 0.0, making the robot slide sideways.

As can be observed in figure 27 when the angular velocity estimate decline from 0.2, there is a sudden rise in angular velocity as the state estimation switches to INS. This behavior is because stems from the commanded angular velocity turning zero, but the robot keeps rotating. The IMU measures the correct angular velocity, but the state estimation does not switch to INS before the difference in angular velocity is over 0.1. After the switch, the INS will converge towards the IMU measurement, but the transition takes some time due to the Kalman filter's slow response. By the time the INS has caught up with the IMU measurement, the robot's rotation slows down, and the estimate drops rapidly to zero. When the INS approach the odometry measurement, a combination of the state estimate and the odometry measurement is used, as can be observed from the mixing variable C2 transition from 0 to 1.

7 Conclusion

7.1 Overview

In this thesis, an algorithm for estimating the linear, lateral, and angular velocity of the Kilter AX1 robot when subject to wheel spin was developed. A literature study was conducted, concluding that INS would be the most viable and most accessible navigation system for the Kilter AX1 robot. Then, a state estimation algorithm using INS was implemented in the ROS2 environment working with the Gazebo simulator to test the possibility of using INS for the AX1 robot. The algorithm works by comparing a velocity estimate from the GPS measurements and IMU measurements with odometry measurements to determine if and when the robot is subject to wheel spin. Odometry measurements were used as the standard linear and angular velocity estimate when no slip was detected. When slip was detected, the algorithm would change the state estimate to using INS until no slip was detected. When no slip was detected the algorithm would switch back to using odometry measurements as the state estimate.

7.2 Findings

A set of tests were developed in order to verify the performance of the state estimation algorithm. The tests were performed with three different friction settings in the Gazebo simulator. By taking advantage of the AX1s RTK-GPS, using the velocity estimated from the RTK-GPS to detect slip was made possible. Comparing the slip detection using an RTK-GPS and IMU for estimating the linear velocity, it was evident that GPS velocity was more effective at detecting wheel spin for linear velocity. Furthermore, the techniques for finding angular and lateral velocity proposed in [5] were found to be accurate for detecting slip.

The state estimation performed well in the tests with high and realistic friction, where the state estimation algorithm manages to capture the essential behavior of the robot. When the simulator was in a low friction setting, the angular velocity state estimation performed well. However, for linear velocity the state estimation uses INS for more extended periods, resulting in an uncertain and unstable estimate due to the noise levels and drift of the accelerometers. The lateral state estimation were unstable due to poor tuning and direct integration of noisy estimates.

The algorithm's most significant limitation is the threshold for which slip is detected. The threshold results in a jump in the state estimates and can cause problems for other parts of the system, like the controller, which does not perform well with sudden jumps in the state estimate.

7.3 Future work

One of the assumptions made in this thesis is flat ground. A flat ground assumption is unlikely to hold on fields where the AX1 robot is to operate. Modifications to the kinematic equations and GPS velocity could be made to incorporate pitch and roll. Pitch and roll are readily available through IMU measurements.

Methods to replace the GPS velocity estimate for a more accurate slip detection that does not suffer from delay will further improve the slip detection. Methods for detecting slip, like using the kinetic equations to predict the future position of the robot and comparing the position prediction to the RTK-GPS measurement, are a possibility.

Another method for detecting slip not mentioned in this thesis is using an additional encoder on the caster wheel. With the inclusion of this additional encoder, detecting slip becomes trivial as a direct comparison between the caster wheel encoder and the driving wheel encoders can be used. Kilter does not currently have an encoder installed on the caster wheel of the robot, but with its inclusion, the slip detection might become even more reliable.

References

[1] Catherine Badgley et al. “Organic Agriculture and the Global Food Supply”. In: *Renewable Agriculture and Food Systems* 22 (June 2007), pp. 86–108. DOI: 10.1017/S1742170507001640.

[2] The Editors of Encyclopaedia Britannica. *Herbicide.* URL: https://www.britannica.com/science/herbicide. (accessed: 28.01.2021).

[3] Trygve Utstumo et al. “Robotic in-row weed control in vegetables”. In: *Computers and Electronics in Agriculture* 154 (Nov. 2018), pp. 36–45. DOI: 10.1016/j.compag.2018.08.043.

[4] Kilter AS. *Asterix Project.* URL: https://www.asterixproject.tech/. (accessed: 19.09.2021).

[5] M. Seyr and S. Jakubek. “Proprioceptive Navigation, Slip Estimation and Slip Control for Autonomous Wheeled Mobile Robots”. In: *2006 IEEE Conference on Robotics, Automation and Mechatronics.* 2006, pp. 1–6. DOI: 10.1109/RAMECH.2006.252627.

[6] C. C. Ward and K. Iagnemma. “Model-Based Wheel Slip Detection for Outdoor Mobile Robots”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation.* 2007, pp. 2724–2729. DOI: 10.1109/ROBOT.2007.363877.

[7] M. Partovibakhsh and G. Liu. “Slip ratio estimation and control of wheeled mobile robot on different terrains”. In: *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER).* 2015, pp. 566–571. DOI: 10.1109/CYBER.2015.7288002.

[8] N. V. Tinh et al. “Modeling and feedback linearization control of a nonholonomic wheeled mobile robot with longitudinal, lateral slips”. In: *2016 IEEE International Conference on Automation Science and Engineering (CASE).* 2016, pp. 996–1001. DOI: 10.1109/COASE.2016.7743512.

[9] Thor Inge Fossen. *HANDBOOK OF MARINE CRAFT HYDRODYNAMICS AND MOTION CONTROL.* Norwegian University of Science and Technology, 2021.

[10] Trygve Utstumo, Therese Berge, and Jan Gravdahl. “Non-linear Model Predictive Control for constrained robot navigation in row crops”. In: (Mar. 2015). DOI: 10.1109/ICIT.2015.7125124.

[11] Norvald Kjerstad. *Navigasjon.* URL: https://snl.no/navigasjon. (accessed: 25.01.2021).

[12] D. M. G. A. I. Sumanarathna et al. “Simulation of mobile robot navigation with sensor fusion on an uneven path”. In: *2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]* (2014), pp. 388–393.

[13] Felipe Kühne, Walter Fetter Lages, and João Manoel Gomes da Silva Jr. "Mobile Robot Trajectory Tracing Using Model Predictive Control". In: (Sept. 2005).

[14] Lars Mæhlum. *RTK - Real Time Kinematic*. URL: `https://snl.no/RTK_-_Real_Time_Kinematic`. (accessed: 29.01.2021).

[15] Navigation National Coordination Office for Space-Based Positioning and Timing. *GPS Accuracy*. URL: `https://www.gps.gov/systems/gps/performance/accuracy/`. (accessed: 25.05.2021).

[16] L. Ding et al. "Slip ratio for lugged wheel of planetary rover in deformable soil: definition and estimation". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 3343–3348. DOI: `10.1109/IROS.2009.5354565`.

[17] D. M. Helmick et al. "Slip compensation for a Mars rover". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 2806–2813. DOI: `10.1109/IROS.2005.1545178`.

[18] N. Ghobadi and S. F. Dehkordi. "Dynamic modeling and sliding mode control of a wheeled mobile robot assuming lateral and longitudinal slip of wheels". In: *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*. 2019, pp. 150–155. DOI: `10.1109/ICRoM48714.2019.9071913`.

[19] Shyamprasad Konduri et al. "Effect of Wheel Slip in the Coordination of Wheeled Mobile Robots". In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 8097–8102. ISSN: 1474-6670. DOI: `https://doi.org/10.3182/20140824-6-ZA-1003.02716`. URL: `http://www.sciencedirect.com/science/article/pii/S1474667016428909`.

[20] Damien Lhomme-Desages et al. "Doppler-Based Ground Speed Sensor Fusion and Slip Control for a Wheeled Rover". In: *Mechatronics, IEEE/ASME Transactions on* 14 (Sept. 2009), pp. 484–492. DOI: `10.1109/TMECH.2009.2013713`.

[21] Yu Tian and Nilanjan Sarkar. "Control of a Mobile Robot Subject to Wheel Slip". In: *Journal of Intelligent Robotic Systems* 74 (June 2013), pp. 915–929. DOI: `10.1007/s10846-013-9871-1`.

[22] Egbert Bakker, Lars Nyborg, and Hans B. Pacejka. "Tyre Modelling for Use in Vehicle Dynamics Studies". In: *SAE International Congress and Exposition*. SAE International, Feb. 1987. DOI: `https://doi.org/10.4271/870421`. URL: `https://doi.org/10.4271/870421`.

[23] C. De La Cruz and R. Carelli. "Dynamic Modeling and Centralized Formation Control of Mobile Robots". In: *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*. 2006, pp. 3880–3885. DOI: `10.1109/IECON.2006.347299`.

[24] Yulin Zhang et al. "Dynamic model based robust tracking control of a differentially steered wheeled mobile robot". In: *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*. Vol. 2. 1998, 850–855 vol.2. DOI: `10.1109/ACC.1998.703528`.

[25] Rached Hatab. "Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework". In: *Advances in Robotics Automation* 02 (Jan. 2013). DOI: 10.4172/2168-9695.1000107.

[26] Open Robotics. *About ROS*. URL: https://www.ros.org/about-ros/. (accessed: 30.05.2021).

[27] Open Robotics. *Understanding ROS 2 nodes*. URL: https://docs.ros.org/en/foxy/Tutorials/Understanding-ROS2-Nodes.html. (accessed: 12.04.2021).

[28] Open Robotics. *Understanding ROS 2 topics*. URL: https://docs.ros.org/en/foxy/Tutorials/Topics/Understanding-ROS2-Topics.html. (accessed: 12.04.2021).

[29] Open Robotics. *Understanding ROS 2 services*. URL: https://docs.ros.org/en/foxy/Tutorials/Services/Understanding-ROS2-Services.html. (accessed: 12.04.2021).

[30] *Gazebo*. URL: http://gazebosim.org/. (accessed: 13.04.2021).

[31] *gazebo_ros_pkgs*. URL: http://wiki.ros.org/gazebo_ros_pkgs. (accessed: 13.04.2021).

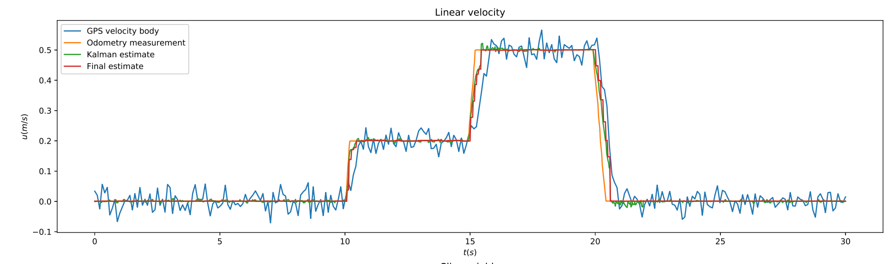
[32] Howard Austerliz. *Data Acquisition Techniques Using PCs*. 2003.

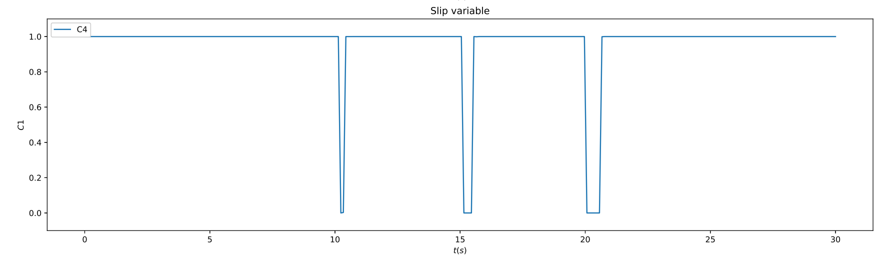
[33] *Rolling Resistance*. URL: https://www.engineeringtoolbox.com/rolling-friction-resistance-d_1303.html. (accessed: 19.05.2021).

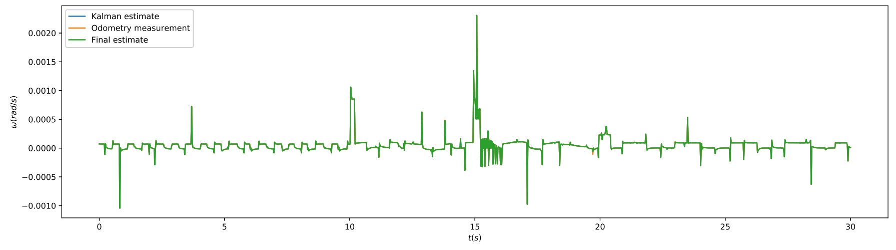
A Appendix

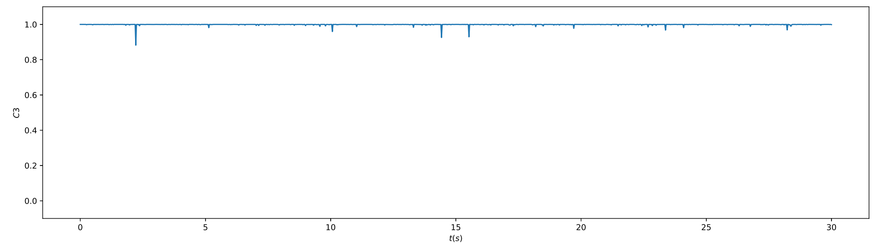
A.1 Linear velocity test

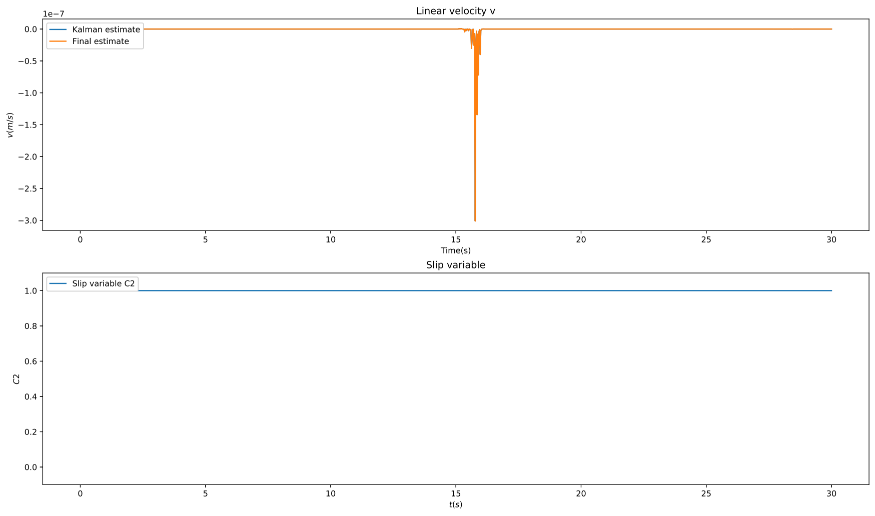
A.1.1 High friction

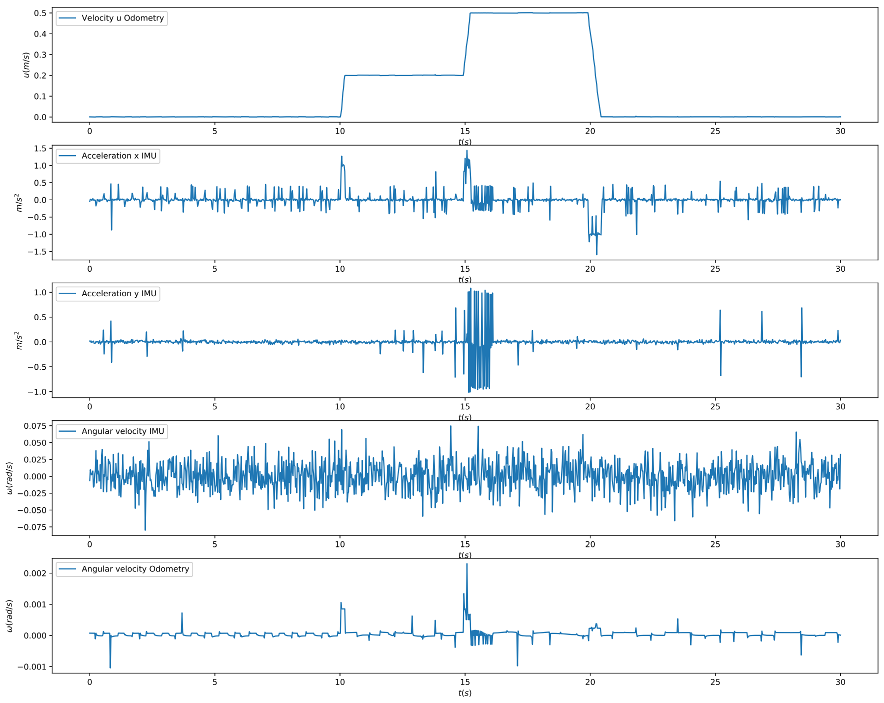




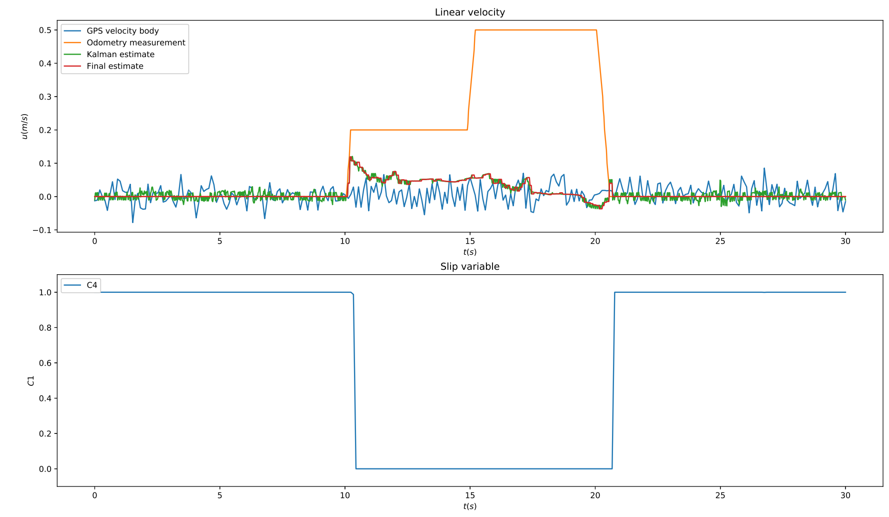


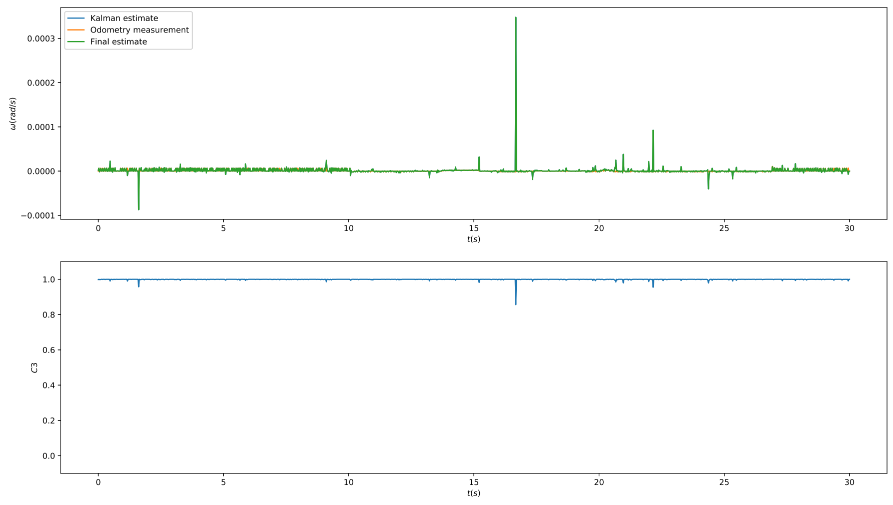


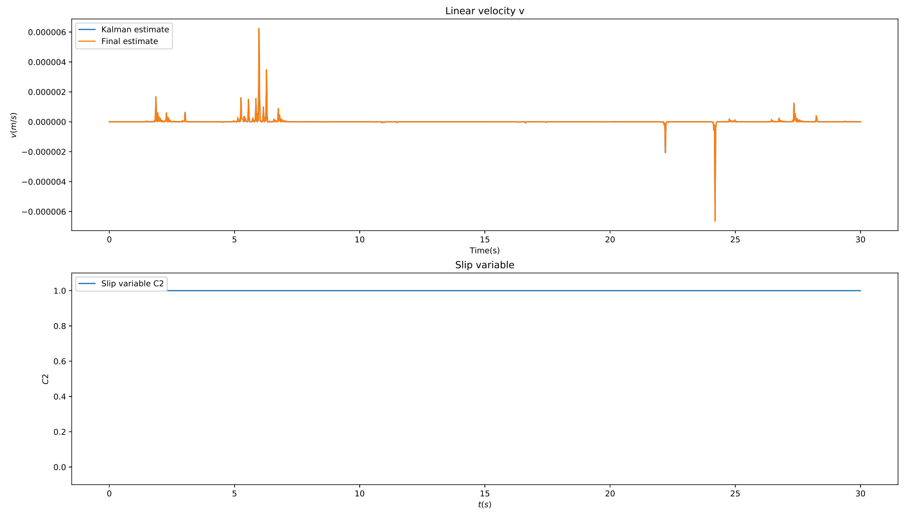

Linear velocity v
1e−7
Kalman estimate
Final estimate
v(m/s)
Time(s)
Slip variable
Slip variable C2
C2
t(s)

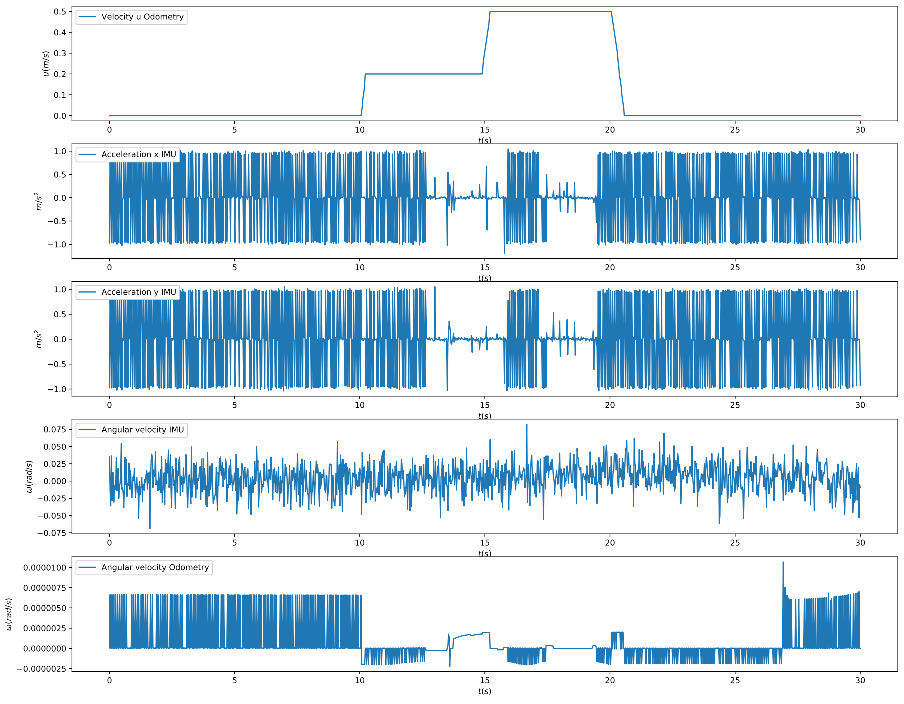

Velocity u Odometry
u(m/s)
Acceleration x IMU
m/s²
Acceleration y IMU
m/s²
Angular velocity IMU
ω(rad/s)
Angular velocity Odometry
ω(rad/s)
t(s)

A.1.2 Low friction

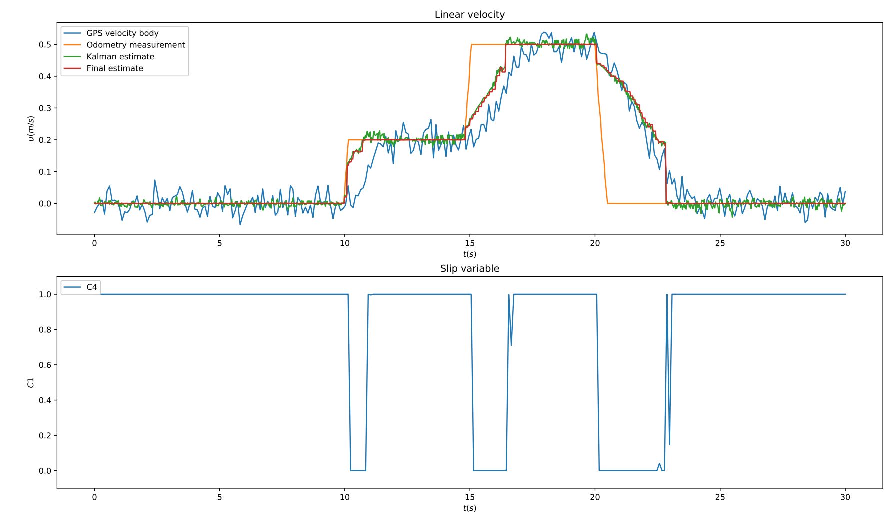


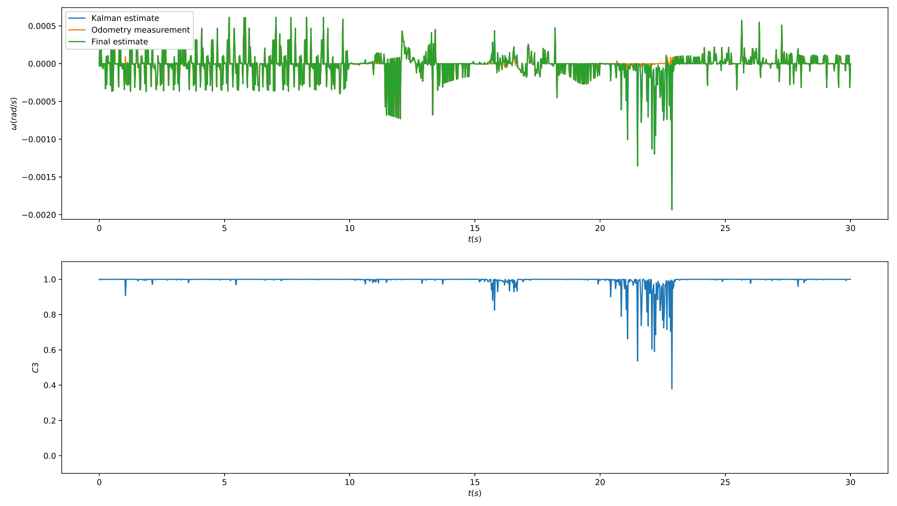


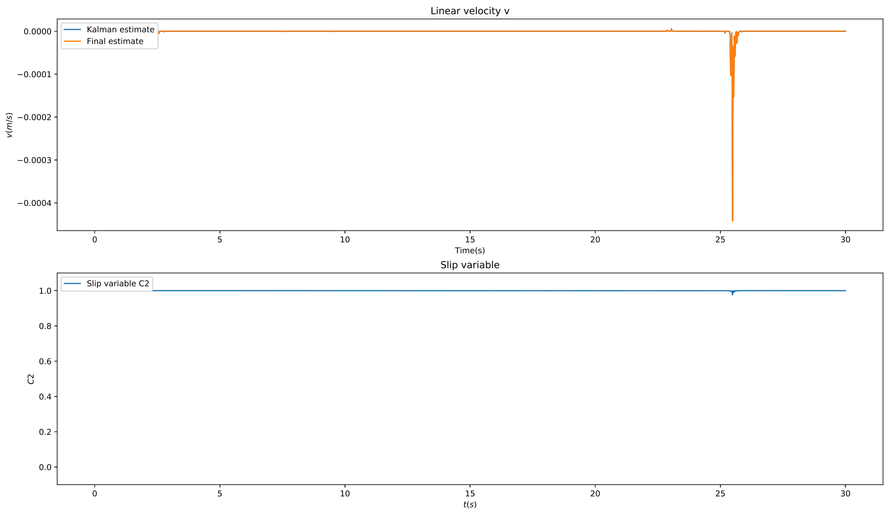

Linear velocity v
Kalman estimate
Final estimate
v(m/s)
Time(s)
Slip variable
Slip variable C2
C2
t(s)

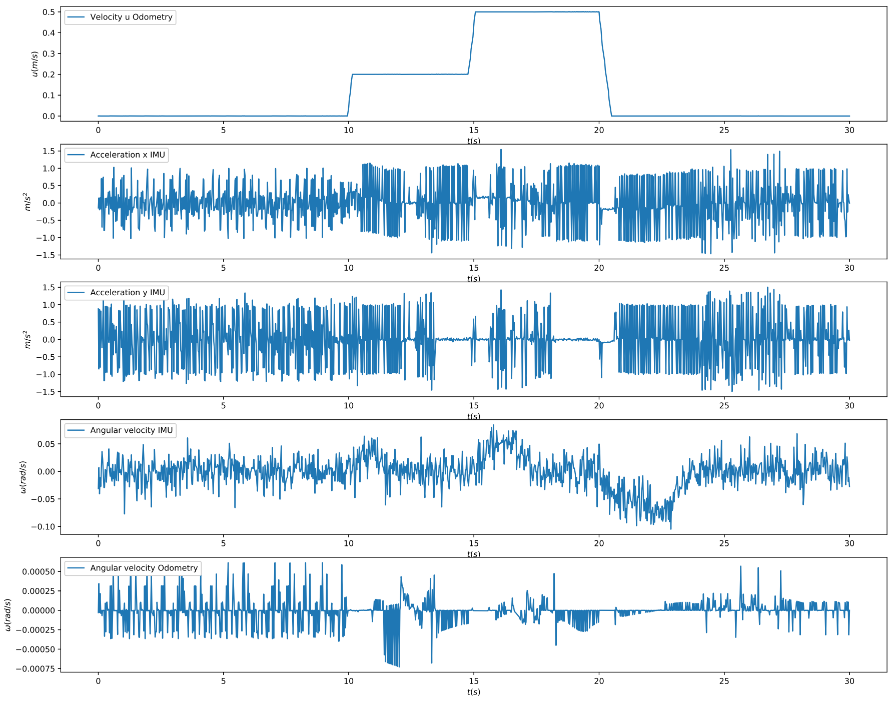

Velocity u Odometry
u(m/s)
t(s)
Acceleration x IMU
m/s²
Acceleration y IMU
Angular velocity IMU
ω(rad/s)
Angular velocity Odometry

A.1.3 Realistic friction



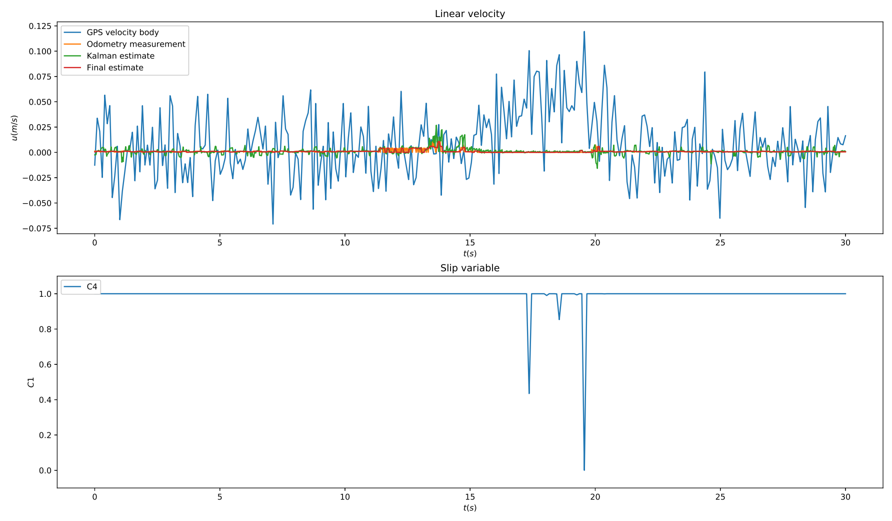


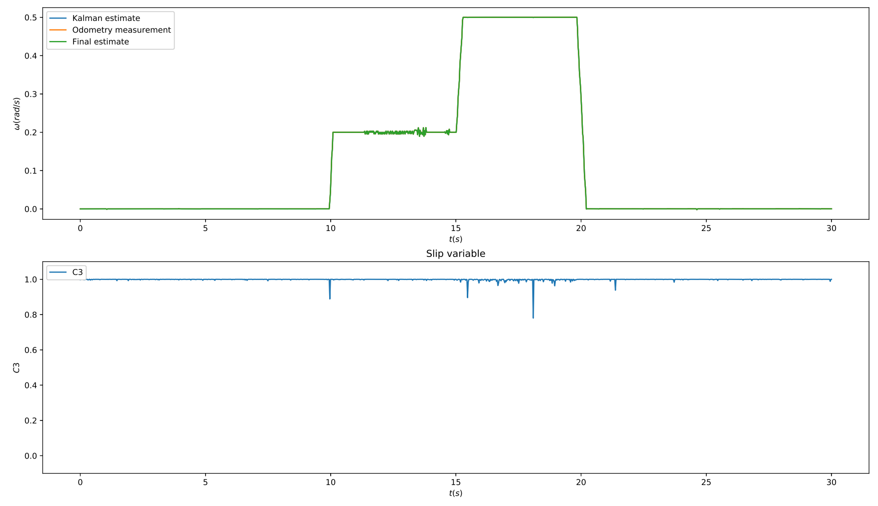

Linear velocity v
Kalman estimate
Final estimate
v(m/s)
Time(s)
Slip variable
Slip variable C2
C2
t(s)

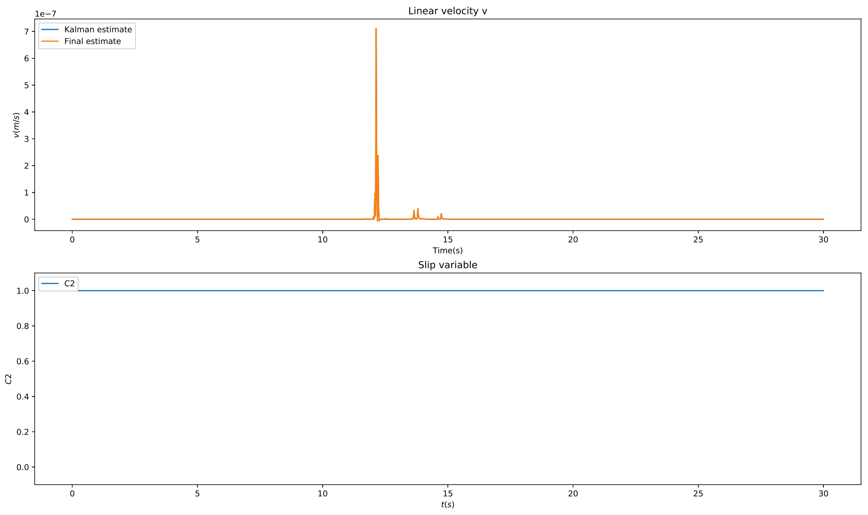

Velocity u Odometry
u(m/s)
Acceleration x IMU
m/s²
Acceleration y IMU
m/s²
Angular velocity IMU
ω(rad/s)
Angular velocity Odometry
ω(rad/s)
t(s)

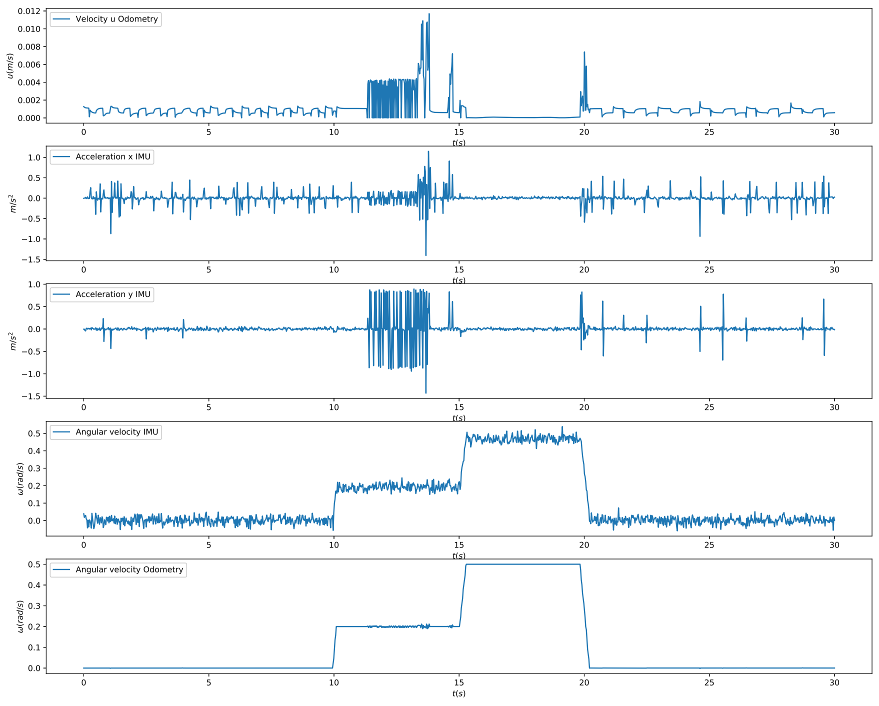
A.2 Angular velocity test

A.2.1 High friction

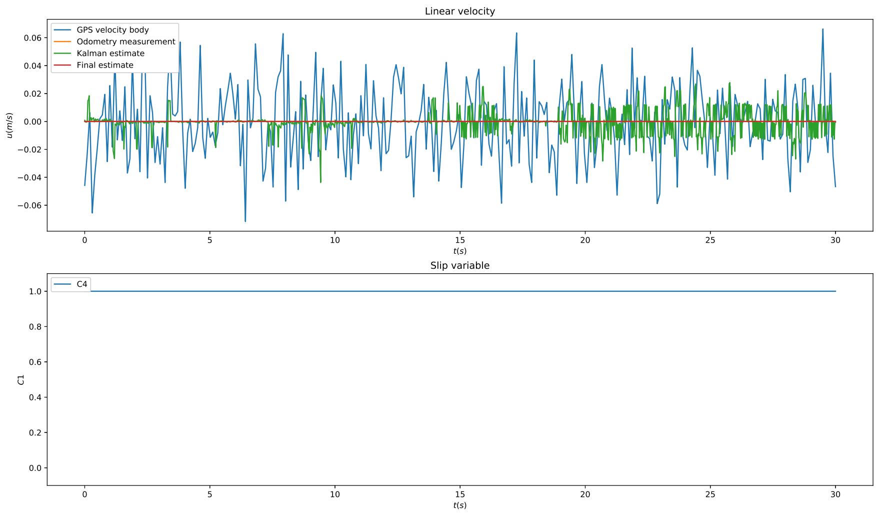


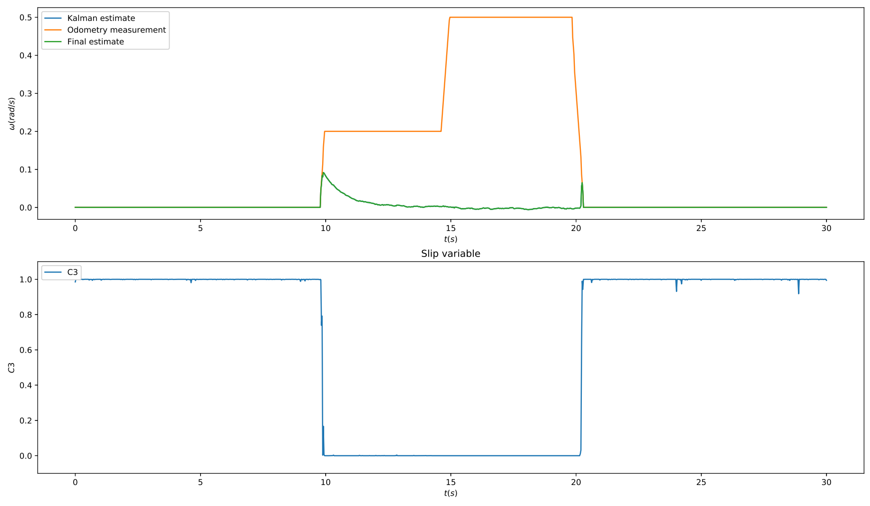


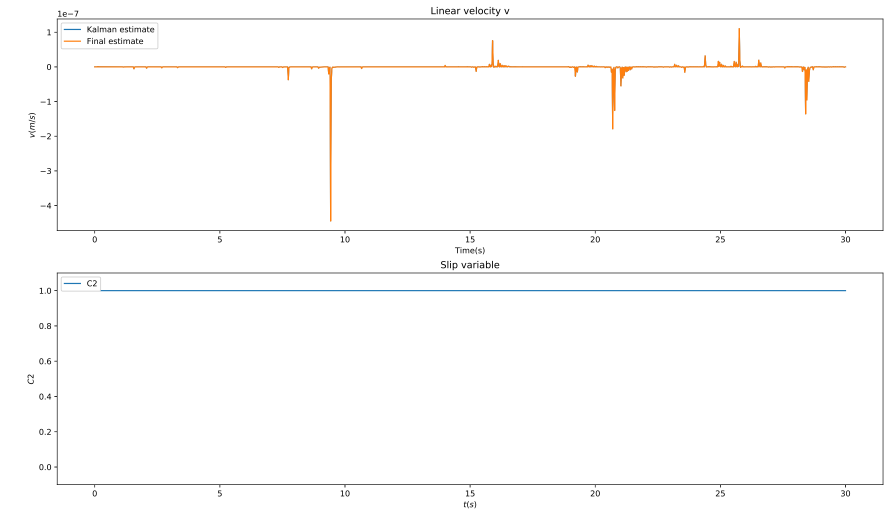

Linear velocity v
1e−7
Kalman estimate
Final estimate
$v(m/s)$
Time(s)
Slip variable
C2
$C2$
$t(s)$

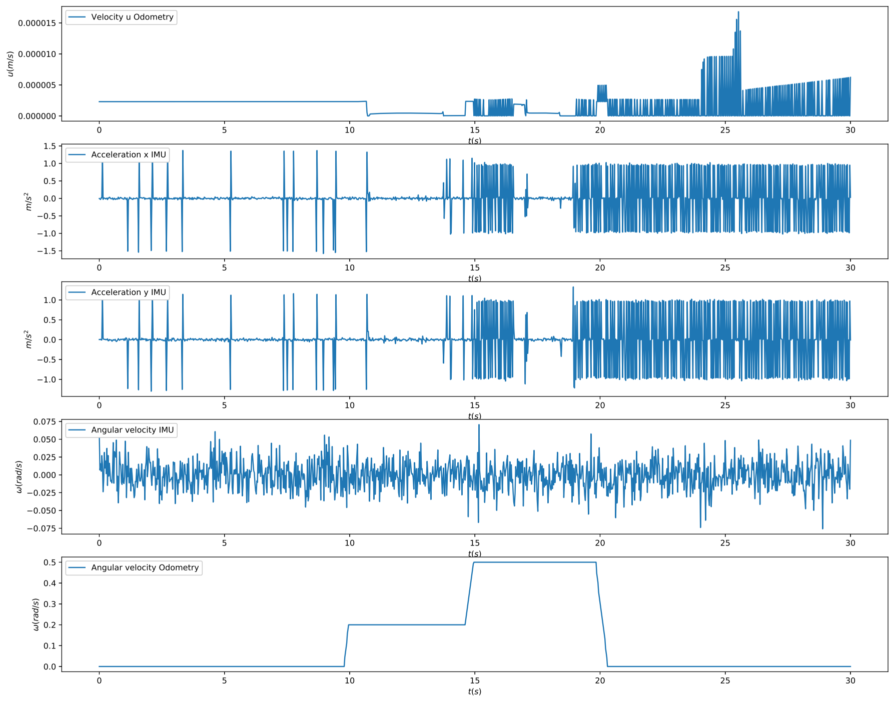

Velocity u Odometry
$u(m/s)$
$t(s)$
Acceleration x IMU
m/s^2
$t(s)$
Acceleration y IMU
m/s^2
$t(s)$
Angular velocity IMU
$\omega(rad/s)$
$t(s)$
Angular velocity Odometry
$\omega(rad/s)$
$t(s)$

A.2.2 Low friction

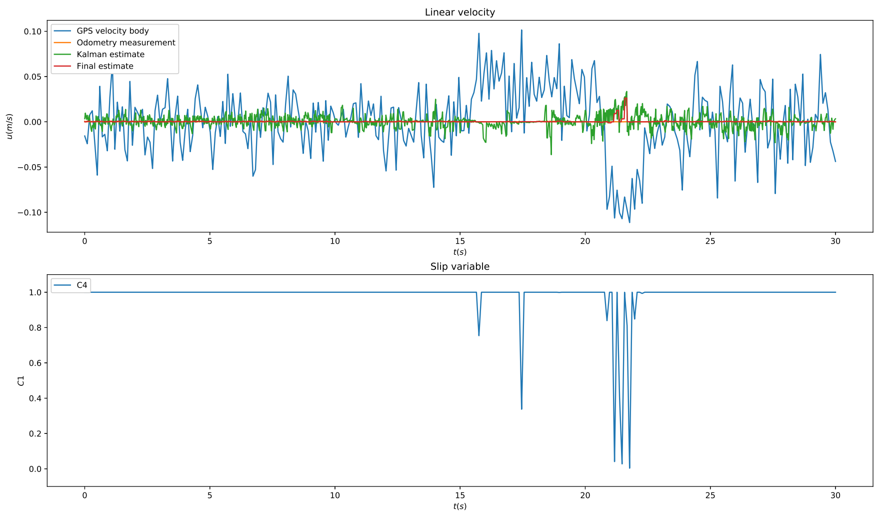


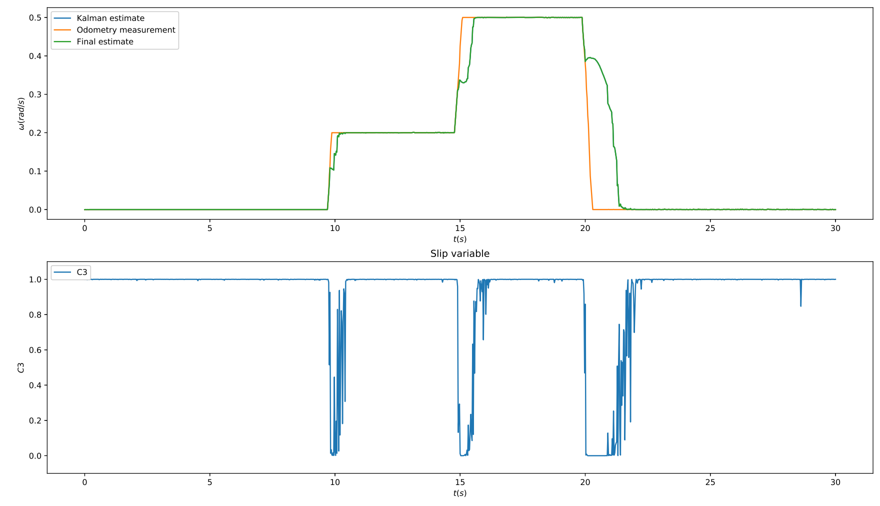


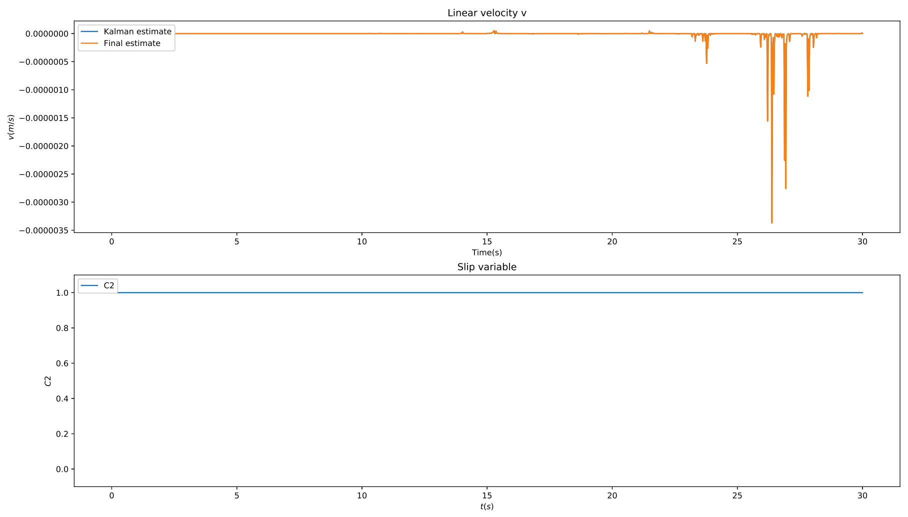

Linear velocity v
1e−7
Kalman estimate
Final estimate
v(m/s)
Time(s)
Slip variable
C2
C2
t(s)

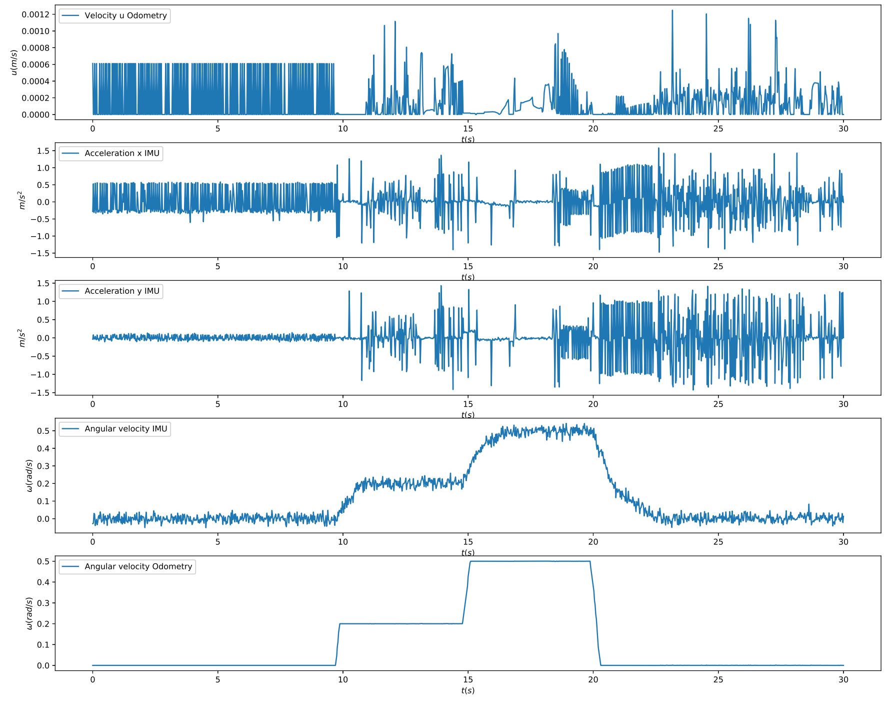

Velocity u Odometry
u(m/s)
t(s)
Acceleration x IMU
m/s^2
t(s)
Acceleration y IMU
m/s^2
t(s)
Angular velocity IMU
ω(rad/s)
t(s)
Angular velocity Odometry
ω(rad/s)
t(s)

A.2.3 Realistic friction



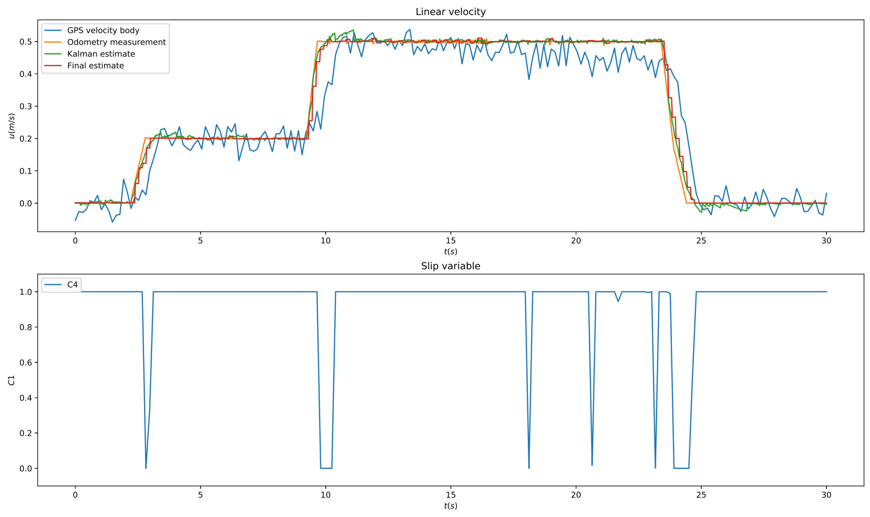


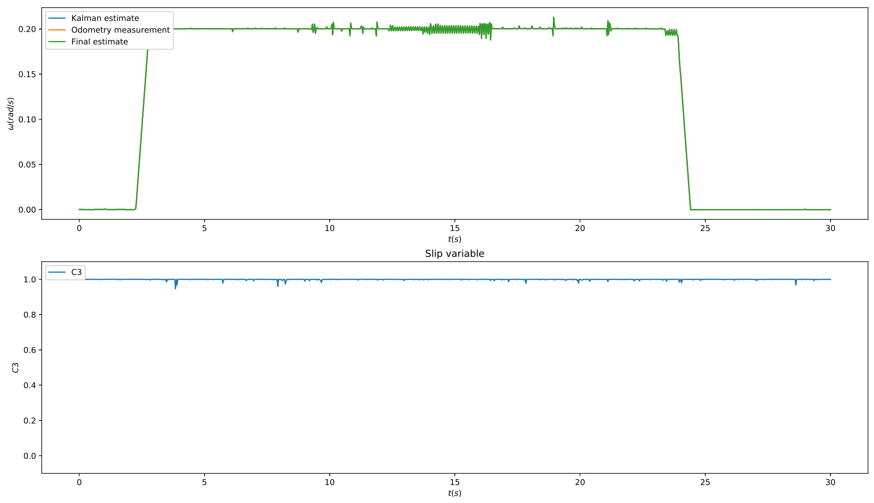

Linear velocity v
Kalman estimate
Final estimate
v(m/s)
Time(s)
Slip variable
C2
C2
t(s)

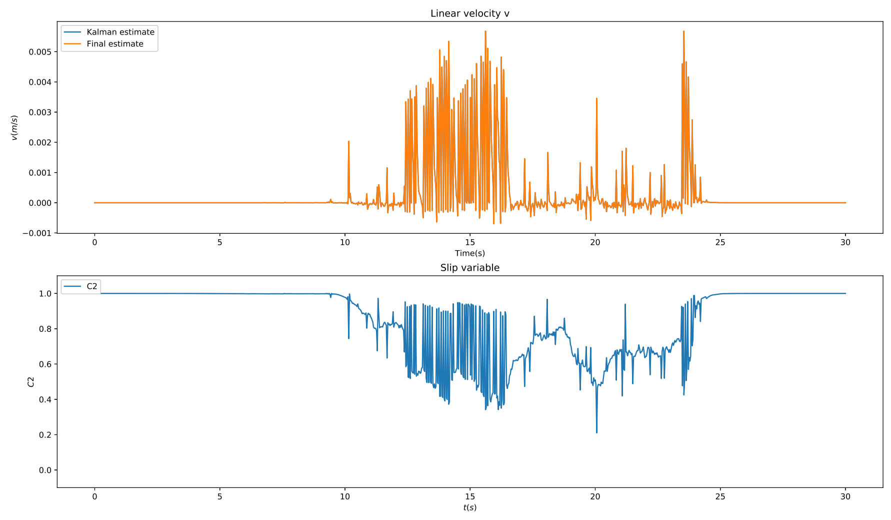

Velocity u Odometry
u(m/s)
t(s)
Acceleration x IMU
m/s^2
t(s)
Acceleration y IMU
m/s^2
t(s)
Angular velocity IMU
ω(rad/s)
t(s)
Angular velocity Odometry
ω(rad/s)
t(s)

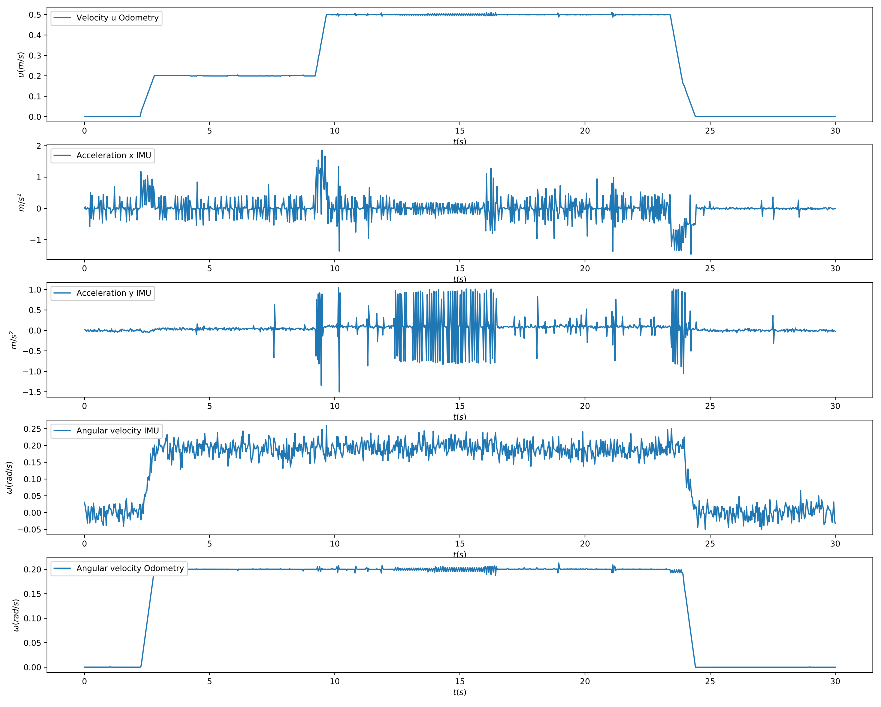
A.3 Circle test

A.3.1 High friction

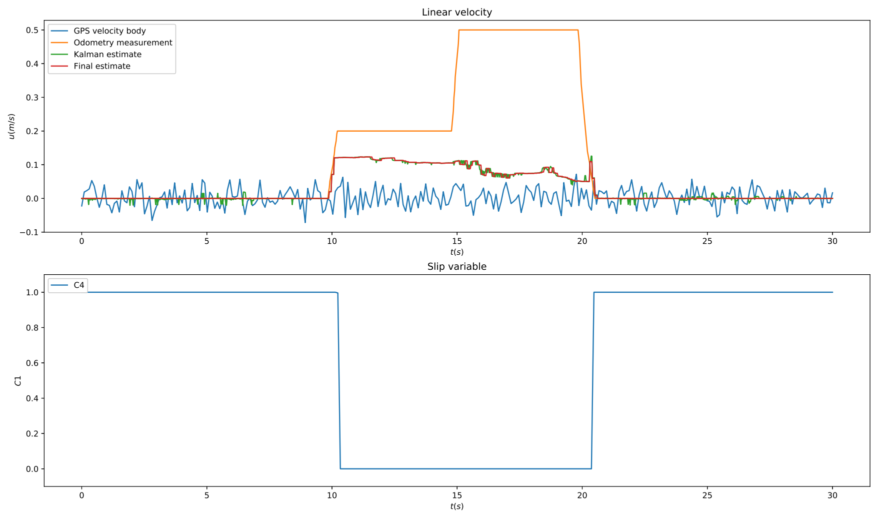


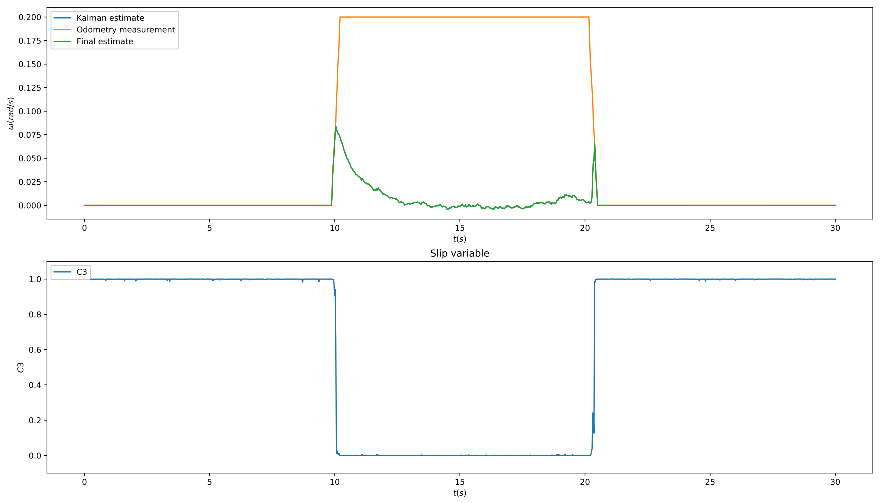


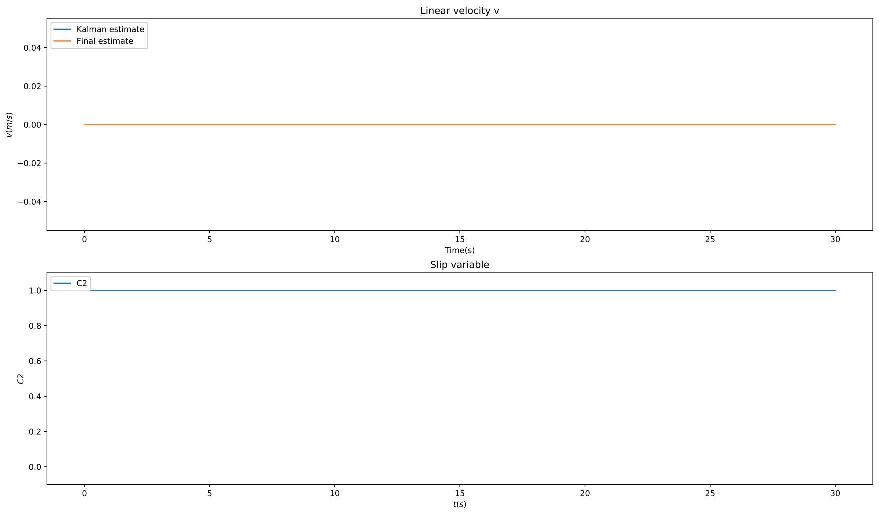

Linear velocity v
Kalman estimate
Final estimate
v(m/s)
Time(s)
Slip variable
C2
C2
t(s)


Velocity u Odometry
u(m/s)
t(s)
Acceleration x IMU
m/s²
t(s)
Acceleration y IMU
m/s²
t(s)
Angular velocity IMU
ω(rad/s)
t(s)
Angular velocity Odometry
ω(rad/s)
t(s)

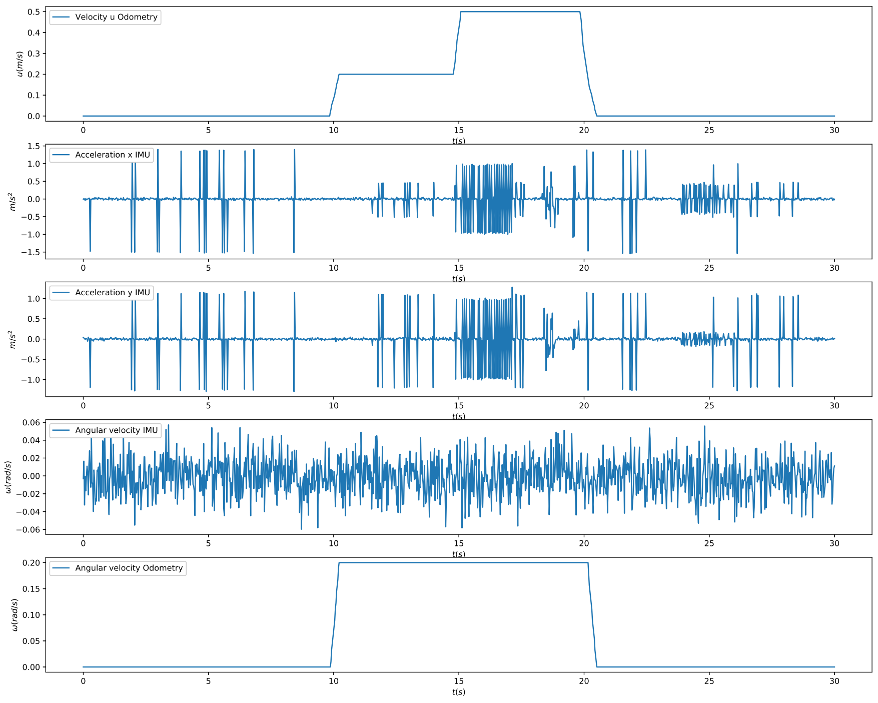
A.3.2 Low friction





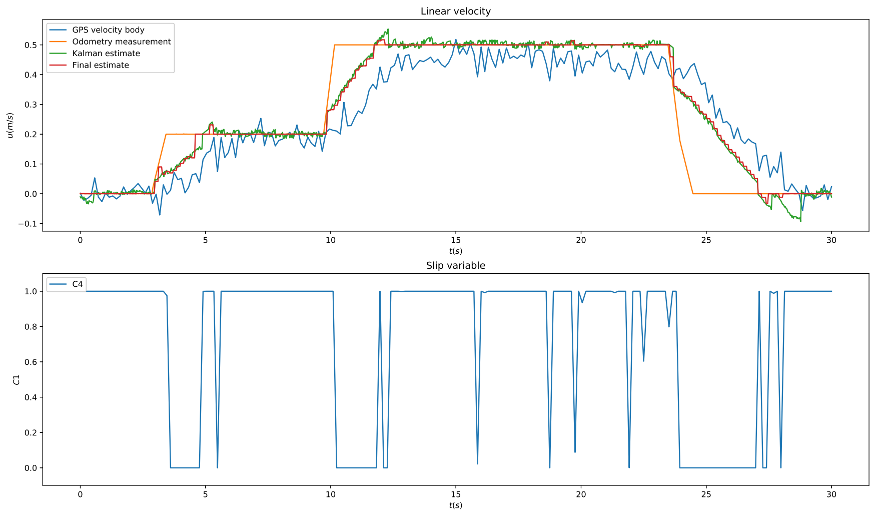


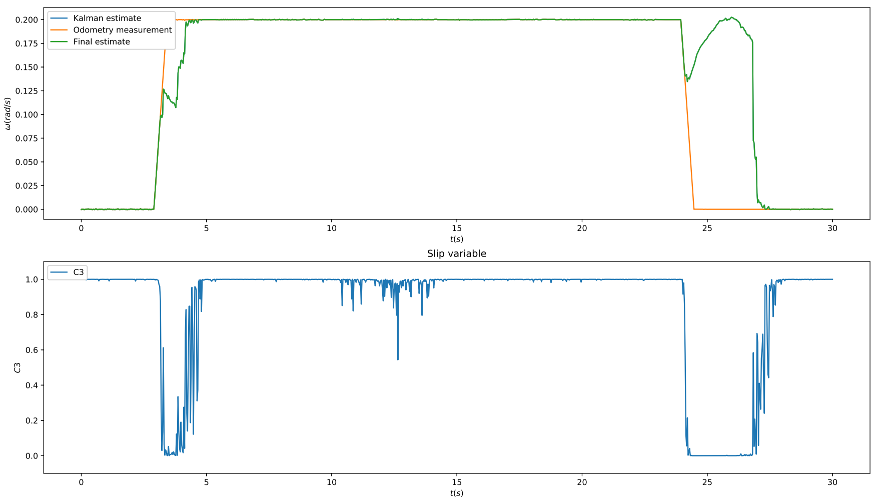
Linear velocity v
Kalman estimate
Final estimate
v(m/s)
Time(s)
Slip variable
C2
t(s)

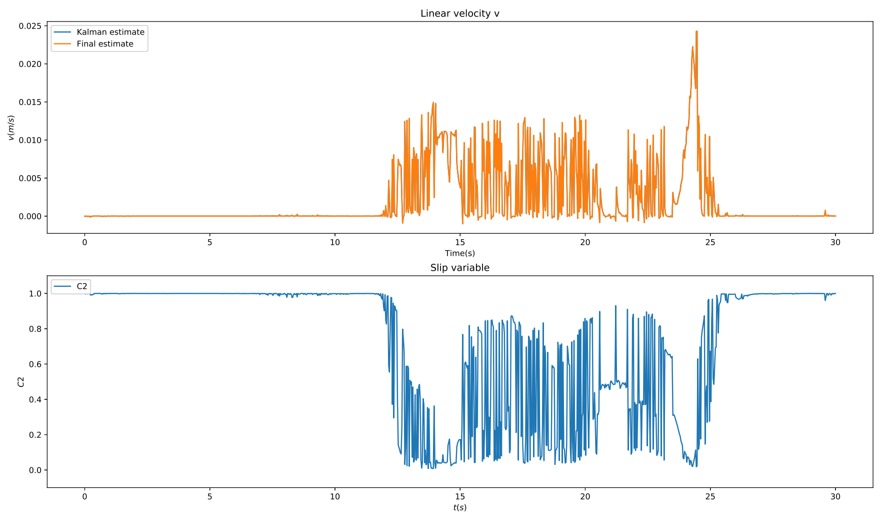


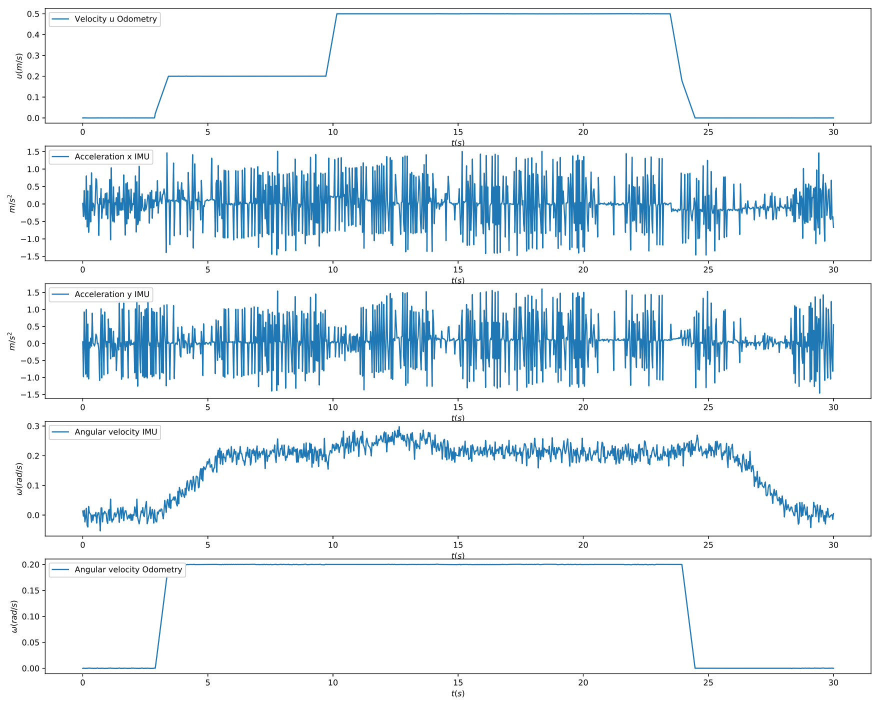
Velocity u Odometry
u(m/s)
Acceleration x IMU
Acceleration y IMU
m/s²
Angular velocity IMU
Angular velocity Odometry
ω(rad/s)
t(s)

A.3.3 Realistic friction






Linear velocity v
Kalman estimate
Final estimate
v(m/s)
Time(s)
Slip variable
C2
C2
t(s)


Velocity u Odometry
u(m/s)
t(s)
Acceleration x IMU
m/s^2
t(s)
Acceleration y IMU
m/s^2
t(s)
Angular velocity IMU
ω(rad/s)
t(s)
Angular velocity Odometry
ω(rad/s)
t(s)