TU WIEN
DEPARTMENT OF GEODESY AND GEOINFORMATION
RESEARCH UNIT ENGINEERING GEODESY

Diplomarbeit

Identification of the Driving Dynamics of a Skid-Steered Mobile Robot Based on Geodetic Measurements

zur Erlangung des akademischen Grades

Diplom-Ingenieur

eingereicht am

Department für Geodäsie und Geoinformation
Forschungsbereich Ingenieurgeodäsie
Wiedner Hauptstraße 8-10 / E120.5
A–1040 Wien, Österreich

unter der Leitung von

Univ.Prof. Dr.-Ing. Hans-Berndt Neuner

und

Univ.Ass. Finn Linzer BSc MSc

von

Markus Mikschi BSc
Matrikelnummer: 01526165

Wien, am 26.03.2023

Hans-Berndt Neuner

Affidavit

I declare in lieu of oath, that I wrote this thesis myself in accordance with the recognised principles for scientific papers. All resources used, especially the literature, are listed in this thesis. If text passages from sources are used literally, they are marked as such.

I confirm that this work is original and has not been submitted elsewhere for any examination, nor is it currently under consideration for a thesis elsewhere.

Date: 26.03.2023

Markus Mikschi

Acknowledgements

First and foremost I want to thank my parents without whom I would not be in the position to write these line. Their unwavering and unconditional love and support guided me through my entire life and afforded me great freedom and security, enabling me to pursue the higher education and other interests of my choosing. My brother has also been a great source of inspiration and motivation for which I am very grateful.

I would like to thank my professor Univ.Prof. Dr.-Ing. Hans-Berndt Neuner and my co-supervisor Univ.Ass. Finn Linzer BSc MSc for supporting and guiding me through the process of creating this thesis, offering thought provoking discussions, technical support and motivation. I am especially grateful for the excellent university courses held by mister Linzer, as they peaked my interest for engineering geodesy and set me on a course which would ultimately lead me to creating this thesis.

I am also very grateful for my colleagues at TU Wien, for the very pleasant working environment they created, for the conversations and discussions with them as well as for the laughs and welcome distractions.

Lastly, I am grateful for the opportunities afforded to me by TU Wien and the Austrian education system. The privilege of conducting research in a field I am passionate about with very expensive equipment while paying next to nothing in tuition is not lost on me.

Abstract

The driving dynamics of skid-steered vehicles are difficult to model due to their inherent need for loss of traction for curvilinear motion, which leads to complex wheel-ground interactions. However, such vehicles represent well suited platforms for automated robots with their robust, cost-effective, low maintenance construction and their great off-road performance. One use-case of them is mobile mapping. Such systems can greatly benefit from precise driving dynamic models for pose estimation for both georeferencing measurements and navigation as well as for system control using model predictive control methods.

System identification is a field of applied mathematics for estimating models of dynamical systems based on measured input and output data of the system. The SINDY (Sparse Identification of Nonlinear DYnamics) algorithm is a method that utilizes sparse regression to identify interpretable, parsimonious models in state-space representation, that balance model performance with complexity.

In this thesis the suitability of the SINDY algorithm to conduct system identification for the driving dynamics of the Clearpath Husky A200 based on geodetic measurements was ascertained. The Husky A200 is a medium sized robot for research and prototyping and represents an example of skid-steered unmanned ground vehicle (UGV) well suited for tasks such as mobile mapping. A measurement setup around two laser trackers for collecting the necessary data was created, addressing the challenges of time synchronisation of the different system components and maintaining an uninterrupted line of sight between the laser trackers and their target prism during driving operations.

A preprocessing pipeline to calculate the system identification input data was established, accomplishing time synchronisation, pose calculation and interpolation as well as state vector calculation and transformation. The system identification was successfully conducted, utilizing the integral notation of SINDY and employing an extensive hyperparameter tuning. The point position estimation uncertainty of the best performing model was 14 cm after a 5 second integration period, with the heading estimation uncertainty being 4.7°. These results demonstrate the suitability of identified systems for example for certain applications of state estimation. Potential shortcomings and areas for improvements of the presented measurement setup and methodology were identified and discussed.

Kurzfassung

Die Fahrdynamik von antriebsgelenkten Fahrzeugen ist aufgrund der inhärenten Notwendigkeit des Traktionsverlusts für kurvenförmige Bewegungen, was zu komplexen Reifen-Boden- Interaktionen führt, schwer zu modellieren. Jedoch sind solche Fahrzeuge aufgrund ihrer robusten, kostengünstigen und wartungsarmen Konstruktion sowie ihrer hervorragenden Geländetauglichkeit gut geeignet für automatisierte Roboter. Eine mögliche Anwendung solcher Roboter ist zum Beispiel Mobile Mapping. Dieses kann von einem präzisen Modell der Fahrdynamik für genauere Zustandsschätzungen zur Georeferenzierung von Messungen und Navigation, als auch für die Systemsteuerung profitieren.

Systemidentifikation ist ein Bereich der angewandten Mathematik, der Modelle dynamischer Systeme auf Basis von gemessenen Eingangs- und Ausgangsdaten des Systems schätzt. Der SINDY-Algorithmus (Sparse Identification of Nonlinear DYnamics) ist eine Methode, die sparse Regression verwendet, um interpretierbare und kompakte Modelle zu identifizieren, die einen Kompromiss aus Modellkomplexität und Prädiktionsgenauigkeit darstellen.

In dieser Arbeit wurde die Eignung des SINDY-Algorithmus für die Systemidentifikation der Fahrdynamik des Clearpath Husky A200 auf Basis geodätischer Messungen getestet. Der Husky A200 ist ein mittelgroßer Roboter für Forschung und Entwicklung und stellt ein Beispiel für ein antriebsgelenktes Roboterfahrzeug dar, welches sich gut für Aufgaben wie Mobile Mapping eignet. Ein Messaufbau mit zwei Lasertrackern zur Datenerfassung wurde erstellt, der die Probleme der Zeitsynchronisation der verschiedenen Systemkomponenten und des Aufrechterhaltens einer ununterbrochenen Sichtlinie zwischen den Lasertrackern und ihrem Zielpunkt während des Fahrens löst.

Eine Abfolge von Vorverarbeitungsschritten zur Berechnung der Eingangsdaten für die Systemidentifikation wurde etabliert. Zeitsynchronisation, Posen-Berechnung und Interpolation sowie Zustandsvektor-Berechnung und Transformation wurden dabei durchgeführt. Die Systemidentifikation mit SINDY wurde erfolgreich durchgeführt, wobei die Integralschreibweise von SINDY und ein umfangreiches Hyperparameter-Tuning eingesetzt wurden. Die Unsicherheit der Positionsschätzung des besten Modells betrug nach einer Integrationszeit von 5 Sekunden 14 cm, und die Unsicherheit des Kurswinkels betrug 4,7°. Diese Ergebnisse zeigen die Eignung von durch SINDY identifizierter Modelle für bestimmte Anwendungen der Zustandsschätzung. Potenzielle Schwächen und Verbesserungsmöglichkeiten des vorgestellten Messaufbaus und der Methodik wurden identifiziert und diskutiert.

Contents

1. Introduction

1.1 Motivation

Skid-steered vehicles are often used as mobile platforms for achieving different tasks, especially in off-road environments. Modelling the dynamics of such vehicles is notoriously difficult as their method of curvilinear locomotion requires the loss of traction of the wheels, resulting in complex tire-ground interactions that are heavily dependent on the driving surface characteristics (e.g. Dogru and Marques (2021), Ordonez et al. (2017)). However, models of the driving dynamics are of great importance both for control algorithms in the context of motion planing and path tracking (e.g. Krishna et al. (2017), Srikonda et al. (2022)) as well as for use in system state estimation methods such as the Kalman filter (e.g. Thalmann and Neuner (2016), Yi et al. (2009)).

Traditionally, dynamical systems, for example describing the driving dynamics of a vehicle, are derived from first principals, decomposing complex systems into small parts with known physical models. Depending on the system complexity, deducing accurate models using this approach can be intractable or even impossible. Often many simplifying assumptions need to be made and not all relevant effects are known. Furthermore, it is often necessary to estimate parameters of the resulting models empirically regardless. An alternative approach is system identification which refers to the estimation of system models using data. This is a field of ongoing scientific development, which greatly profits from recent advances in computational performance, mathematics and the era of big data. One of the methods for system identification is SINDY (Sparse Identification of Nonlinear DYnamics), which both identifies the structure of the model and estimate its coefficient with the goal to obtain sparse, parsimonious, interpretable models from noisy data (Brunton et al., 2016a).

In case mobile platforms are used for geospatial data acquisition one speaks of mobile mapping (Li, 2011), which is an important area of ongoing research in geodesy and other fields (e.g. Lehtola et al. (2022), Hui et al. (2022)). On one hand the increased interest in mobile mapping platforms is fueled by the emergence of better and less expensive hardware, such as sensors, as well as better algorithms for path-planning, system control and data analysis. On the other hand the need for data that satisfies criteria such as high temporal and spatial resolution, large spatial extent and economical data collection requires the use and development of modern and efficient measuring approaches.

Accurate models of the driving dynamics of mobile platforms are important for mobile mapping systems as, among other benefits, the pose estimation accuracy can potentially be improved when used with methods such as the Kalman filter. The time series of poses, also called trajectory, consisting of position and attitude of a mobile mapping platform is of great importance. On one hand it can be used for path planing and system control as the current state of the system is crucial for navigation. On the other hand the trajectory is needed for georeferencing the measurements obtained from the instruments mounted on the mobile mapping platform.

The Clearpath Robotics$^{\text{TM}}$ Husky A200 is a skid-steered Unmanned Ground Vehicle (UGV) that is intended as a robotic development platform. Its large payload capacity of 75 kg can accommodate a wide variety of payloads, for example GNSS receivers, Inertial Measurement Units, laser scanners, cameras and more. It therefore represents an interesting example of a platform with difficult to model dynamics that is well suited for mobile mapping applications. The engineering geodesy research group at the TU Wien uses a Husky A200 as the subject of several research projects (e.g. Brandstätter (2022)).

The topic of this thesis is to examine the possibility of using the SINDY system identification algorithm to estimate a dynamical system model representing the driving dynamics of the Husky A200 based on geodetic measurements. The identification of driving dynamics in this thesis is limited to cases of driving on level surfaces and only driving on one specific surface type.

1.2 Aim of the thesis

This theses has the following aims:

- Development of a measurement setup for precise, high frequency 2D pose measurements of ground based mobile platforms
- Development of a processing pipeline for deriving input data for system identification based on the obtained measurements
- Evaluation of the suitability of the SINDY algorithm for system identification of the driving dynamics for the Husky A200 UGV

1.3 Outline of the thesis

This thesis is structured into 8 chapters. Chapter 2 describes important theoretical fundamentals and gives an overview of related work in order to give context for the thesis. It explains the reason why dynamical systems are useful and introduces the field of system identification. In chapter 3 the used hardware and software is listed and explained. The methodology for both calculating the input data for the system identification as well as the system identification itself is explained in chapter 5. Chapter 4 details the developed measurement setup and chapter 6 describes the conducted experiments. Chapter 7 finally presents the obtained results in form of the estimated system model and its performance, while chapter 8 offers the conclusion of the thesis along with an outlook of potential improvements and possible future work.

2. Methodological Fundamentals

This chapter explains theoretical fundamentals in order to aid understanding of the thesis and introduce nomenclature and notations used throughout the thesis. In section 2.1 the concept and importance of dynamical systems is briefly explained. Subsequently, in section 2.2 an introduction into the large, multidisciplinary field of system identification is given. Section 2.3 gives some context of the difficulty modeling of skid steered vehicles, while also presenting existing methods and models for such vehicles. Sections 2.4 and 2.5 describe the method of the least squares adjustment and the used transformation notation respectively.

2.1 Dynamical Systems

Dynamical systems concern the analysis, prediction and understanding of systems whose state evolves over time. They represent a framework for describing dynamic phenomena in the sciences and engineering and are expressed as either differential equations in the time continuous case or iterative mapping functions or difference equations in the time discrete case. Dynamical systems encompass a very broad range of phenomena including classic mechanical systems, fluid dynamics, finance, population dynamics, climate science, epidemiology and many more.

2.1.1 Use cases

As models of dynamical systems describe the behaviour of a wide variety of real-world systems they have many practical and critical use cases. These include these major uses of dynamical system models in modern applications:

- Understanding
- Optimization
- Prediction
- Simulation
- Control

With regard to mobile platforms and especially mobile mapping platforms prediction and control are of special interest. The usage of dynamical system models for control in the context of such platforms pertains to path planning and the calculation of control inputs. It is necessary to understand what actions a system is capable of performing given a certain control input and how a certain control input will affect the system. A dynamical system model can also quantify the controllability of a system in given situations with a precise system model also enabling the creation of sophisticated control laws which can for example be used to minimize the energy expenditure.

On the other hand prediction of the system state using a dynamical system model can be used in sensor fusion algorithms. The Kalman Filter for example is a recursive algorithm, estimating the state of a dynamic system based on series of noisy measurements obtained over time (Groves, 2013). The filter uses knowledge of the underlying system in form of a dynamical system model in order to carry the previously incorporated measurement information forward in time. Therefore measurements that pertained to previous, different system states can be used to improve the estimation of the current system state. Estimating the system state at a certain point in time does not require reprocessing all previous measurements but only updating the previous state vector estimate with the newly arrived measurement information. The more precise the system model is, the more information contained in the previous state estimation can be carried forward in time and the better the state estimation will be over time. This is also apparent in the fact that better models exhibit a better associated system noise covariance matrix which describes the errors of effects not captured by the model on the system state.

2.1.2 Explicit ordinary differential equations

In the case of systems with only one continuous independent variable, most often time, the system is expressed with ordinary differential equations. If there are one or more additional continuous, independent variables, for example physical space coordinates in the case of non-localized, spread-out phenomena, such as heat distribution or any kind of wave, they are expressed as partial differential equations. In this thesis only ordinary differential equations (ODE) are of interest, as the driving dynamics of the Husky A200 UGV represent a time continuous, localized phenomenon.

An explicit ordinary differential equation of the order n is defined as

$$y^{(n)} = f(x, y', y'', ..., y^{n-1}). \tag{2.1}$$

with f being a function of an independent variable x, a dependent functions variable y and its derivatives, with $y^{(i)}$ being the i-th order derivative. If y is a vector, f is a vector-valued function and

$$\mathbf{y}^{(n)} = \begin{bmatrix} y_1^{(n)} \\ y_2^{(n)} \\ \vdots \\ y_m^{(n)} \end{bmatrix} = \mathbf{f}(x, \mathbf{y}', \mathbf{y}'', ..., \mathbf{y}^{\mathbf{n-1}}) = \begin{bmatrix} f_1(x, \mathbf{y}', \mathbf{y}'', ..., \mathbf{y}^{n-1}) \\ f_2(x, \mathbf{y}', \mathbf{y}'', ..., \mathbf{y}^{n-1}) \\ \vdots \\ f_m(x, \mathbf{y}', \mathbf{y}'', ..., \mathbf{y}^{n-1}) \end{bmatrix} \tag{2.2}$$

is an explicit system of ordinary differential equations of order n and dimension m.

Reduction of order

The order of differential equations can sometimes be reduced, which usually facilitates easier solving of the equation. Reducing the order is always possible for explicit ODEs as any explicit ODE of order n can be written as a system of n first-order differential equations by introducing $n-1$ new unknown functions. The explicit order of the ODE in equation 2.1 can be reduced by defining the new function variables

$$y^{(i)} = y_{i+1} \tag{2.3}$$

for $i = 1..n-1$ resulting in the system

$$\frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ \vdots \\ y_n \\ f(x, y_1, y_2, \dots, y_n) \end{bmatrix} \tag{2.4}$$

or more compactly as

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}). \tag{2.5}$$

This concept can be extended to systems of m ODEs of order n, resulting in a system of $m \cdot n$ first-order ODEs.

2.1.3 State space representation

The state-space representation is a mathematical model, especially used in control engineering, which expresses a dynamical system as a system of first-order differential equations relating a set of input, output and state variables. The representation may in some cases be obtained using order reduction as described above. The state of the system is explicitly and fully described by a set of so called state variables. They evolve over time in a way that depends on the values they have at any given time and on the externally imposed forcing described by the ODEs. The number of state variables is the minimum number required to fully describe the dynamical system state at any given point. Together they form the state vector which is a point in the state space. In equation 2.4 the vector $\mathbf{y}$ would be the state vector. The state space has the state variables as the basis axes and encompasses every possible system state as individual point. The state space representation is not unique as the choice of state variables fully describing the system is ambiguous and amounts to selecting a basis for the state space. In the case that the state space is continuous and finite-dimensional, as is the case for the majority of physical systems including the driving dynamics of a vehicle discussed in this thesis, it is referred to as phase space.

For unforced dynamical systems the vector-valued function describing the system of ODEs assigns each point in phase space a vector pointing in the direction the system state represented by that point evolves over time. The progression of a system in time therefore traces a trajectory in phase space that is tangential to the vector field. For forced systems the phase space may be seen as being extended by control input space in which every point, describing the current state and control input, is mapped to the change of the state vector.

As the concept of state space representation stems from control engineering the independent variable is virtually always time, denoted t, and the state vector is commonly denoted $\mathbf{x}$. This notation is adopted henceforth. The general state space representation relevant for this thesis, is thus

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.6}$$

with $\mathbf{u}$ being the optional control inputs representing external forcing.

2.2 System Identification

System identification is a longstanding field in applied mathematics which deals with creating mathematical models based on data that describe dynamical systems (Ljung, 1998). It has increasingly become an important tool with applications in many disciplines both in engineering and science. The most prevalent use case is control system engineering, specifically for model-based feedback control, as more precise models enable the development of better, more efficient control laws for dynamical systems. However, models created with system identification can be used for many applications, among which are the five areas listed in section 2.1.1.

The traditional approach for creating models of dynamical systems is analytical modelling where the system is split into components for which an analytical description based on known natural laws is possible. This technique, also referred to as physical modelling, therefore requires expertise in the studied field and can be very labor intensive (Ljung, 1998). The goal of system identification is to automate the model creation and potentially increase the model accuracy by using machine learning methods to estimate models based on data.

System identification uses measured input and output signals of a system to estimate a model of the dynamical system relating those signals. The modeling can be conducted in either the time or frequency domain of the system. While many different methods of system identification exist, most methods are still restricted by an assumed model form, and sometimes are even limited to linear dynamics, reducing the obtainable accuracy and still requiring expert knowledge about the dynamic system in question (Schmidt & Lipson, 2009). System identification that require an assumption for the model form are parametric methods which can be used to estimate system properties such as physical quantities such as mass, friction coefficients, moment of inertia etc. Non-parametric or free-form approaches are not restricted to a predetermined model structure and enable to also identify the structure in addition to the estimated parameters. They therefore require less knowledge about the system from domain experts. Some methods are not truly non-parametric as they have a certain number of parameters but have so much model capacity that they can model a wide variety of phenomenons. Among those are neural networks which strictly speaking have a certain number of parameters but in practice act like nonparametric models due to the universal approximation theorem.

Usually there is a tradeoff associated with using nonparametric or free-from methods for system identification which is that the resulting models are hard if not impossible to interpret and understand. They are therefore also referred to as "black box" models and occupy the polar opposite on the spectrum compared to analytical modeling. They are mainly aimed at precise system state prediction and not the derivation of compact, interpretable models. However, there have been great efforts for identifying non-linear dynamical systems using nonparametric methods that result in models that are meaningful, parsimonious and interpretable.

Schmidt and Lipson (2009) used symbolic regression based on genetic programming to construct families of candidate nonlinear functions in order to deduce inferred natural laws such as the Hamiltonian, the Lagrangian and equation of motions of mechanical systems in a "breakthrough in nonlinear system identification" (Brunton et al., 2016a). This method achieved to recreate actual known physical laws such as conservation equations from data instead of focusing on pure predictive performance. One potential issue that the usage of symbolic regression has is that it is computationally expensive and may not scale well to large data sets or very complex dynamical systems. In Brunton et al. (2016a) the SINDY algorithm was presented which stands for Sparse Identification of Nonlinear DYnamics. It is a method for identifying models of non-linear dynamical systems expressed in the state space representation. The method places an empathises on creating interpretable, parsimonious models by combining sparsity-promoting techniques and machine learning.

SINDY uses a sparse regression that is linear int the estimated coefficient but non-linear in the state variables to find the fewest terms that still explain the observed data. The core assumption thereby is that the sought system model is sparse when expressed in an appropriate function basis, meaning that only a few terms of a vast function library are active. This is an assumption that holds true for many natural dynamical systems (Brunton et al., 2016a). By using a sparse regression the method automatically balances model complexity as expressed by model terms and model accuracy. It is also more robust against outliers in the data.

The SINDY framework was subsequently extended by Brunton et al. (2016b) to cover actuated dynamical systems, introducing the possibility to include external forcing in the models. In Kaiser et al. (2018) the authors used SINDY in a framework for model predictive control (MPC), comparing the method to neural network based approaches and concluding "the resulting SINDY-MPC framework has higher performance, requires significantly less data, and is more computationally efficient and robust to noise than NN models, making it viable for online training and execution in response to rapid system changes." Schaeffer (2017) used the idea of SINDY to discover governing equations for partial differential equations. The SINDY framework thus represents a universal, well performing method for non-linear system identification that has been already applied to many problems.

The SINDY algorithm itself is described in section 5.3.1 with section 5.3.2 detailing the extension of SINDY to problems with external forcing, which is needed as the Husky A200 UGV experiences such external forcing in the form of motor control inputs. Lastly, section 5.3.4 describes how the method is applied to the specific problem of estimation a model for the driving dynamics of the Husky A200 UGV.

2.2.1 Sparsity and compressed sensing

SINDY applies the mathematical concepts of sparsity and compressed sensing to the application of system identification. Therefore this section, which is predominantly based on Brunton and Kutz (2019), briefly introduces those concepts.

Most natural signals, such as images, audio data and time series measurement data, exhibit inherent structures that make the data highly compressible. If observed natural data is expressed in a well-suited basis, the coordinate vector is sparse with a high probability, meaning that only a few parameters are required to describe the data by characterizing which components of the basis are active and in what proportion. This is the fundamental idea of all data compression algorithms, whereby a signal is represented more efficiently by transforming it into a different basis.

A compressible signal $\mathbf{x} \in \mathbb{R}^n$ can be expressed as a sparse coordinate vector $\mathbf{s} \in \mathbb{R}^n$ with the transform basis $\boldsymbol{\Psi} \in \mathbb{R}^{n \times n}$ using

$$\mathbf{x} = \boldsymbol{\Psi}\mathbf{s}.$$

The coordinate vector $\mathbf{s}$ is called K-sparse if it contains K non-zero elements. The basis $\boldsymbol{\Psi}$ can either be a custom basis, for example obtained by means of the Principal Component Analysis (PCA), or a well suited generic transform basis, such as the Fourier basis or a wavelet basis. The Fourier and wavelets bases are generic or universal bases as most natural data, such as images and audio, is sparse in those bases.

The idea of sparsity can not only be used for signal compression but also to reconstruct a full signal from only a few measurements by reconstructing its sparse representation $\mathbf{s}$ given a chosen basis $\boldsymbol{\Psi}$, instead of the non-sparse representation $\mathbf{x}$. This method is called compressed sensing since it relates the measurements to the compressed representation of the data instead of the full data representation directly.

Furthermore, sparsity can be used for retrieving parsimonious models that avoid overfitting by acting as a regularization. Such models stay interpretable and have limited model capacity due to having the minimum active models terms while still explaining the data used for training the model. A model with fewer terms that explain the data almost as well as a model with more terms follows the Occam's razor principle which states that the most simple explanation is generally the correct one. Sparse optimization has also the benefit of increased robustness with regard to outliers.

2.3 Skid-Steered Mobile Robots

Skid-steering is a method for achieving curvilinear locomotion that is used in many kind of vehicles, especially many all-terrain vehicles such as loaders, farm machinery, mining and military vehicles. It is also used in mobile robots such as the Husky A200 which is the subject of this thesis. Skid-steering uses differential thrust between the left and right sides of the vehicle creating a torque which is turn causes a change in heading. The main benefits of this method are the compact and robust construction, which makes it also very cost effective, and low maintenance. It also gives the vehicle very high maneuverability, even allowing for turning in place (Shamah et al., 2001). The main drawbacks on the other hand are high power consumption when turning as energy needs to be spend to overcome the friction of the tires (Ordonez et al., 2017). This is especially true for tight turns at low speeds or turning in place. The power draw is also hard to predict, which is needed for energy optimizing control algorithms, as it heavily depends on the surface type and inclination (Shamah et al., 2001).

Kinematic or dynamic modelling of skid-steered locomotion is notoriously difficult as the dynamics strongly depend on the complex wheel-ground interactions as turning by definition requires a loss of traction and subsequent sliding and skipping of the wheels over the ground (Wang et al., 2009). This makes the motion of vehicles using this method dependent on the surface properties of the floor and also introduces a strong effect of the slope of the driving surface as the friction of the tires is dependent on normal force of the ground.

In Wang et al. (2009) the authors used a combined kinematic and dynamic modeling approach for skid-steered mobile robots. They utilized similarities between the skid-steered robot and tracked vehicles in order to apply existing models for the kinematic model while the dynamic modeling approach uses existing automotive tire/road interaction models. The result is a complex analytical model for the locomotion which the authors also used in an extended Kalman Filter (Groves, 2013) for robot localization. This approach was subsequently refined in Yi et al. (2009). In Dogru and Marques (2021) a novel, improved kinematic of skid-steered wheeled platforms was presented which took both the geometric properties as well as the location of the center of mass into account. The authors tested the new kinematic model with two different vehicles and on different surfaces.

In order to account for the very strong dependence on the driving surface characteristics many methods rely at least partially on machine learning to do online learning of a terrain dependent model. In Thalmann and Neuner (2016) the authors among other things demonstrated the use of external absolute positioning sensors in the form of a geodetic tachymeter in order to estimate an additional system parameter and it was successfully shown that this parameter captured some physical properties of the tire-ground interactions, resulting in improved dead-reckoning performance. In Ordonez et al. (2017) the authors ”developed a new methodology to perform online learning of terrain dependent robot models that are highly relevant for motion planning applications”. The proposed methodology combines dynamic models for the wheel-terrain interaction with online learning to update the kinematic model and an neural network to update the dynamic model.

Deep learning approaches are also used in the context of skid-steered robots. In Srikonda et al. (2022) a reinforcement learning algorithm was used to obtain a path tracking controller for the dynamic model of a skid-steered vehicle. The authors show that the controller successfully adapted to "the complex nonlinear forces and torques acting at the wheel-ground interface".

In summary there are many approaches for dealing with modeling and controlling skid-steered robots among which many use at least partially machine learning.

2.4 Least squares adjustment

As the least squares adjustment is used at multiple points throughout this thesis a short introduction is presented in this section. This summary is an adoption of Niemeier (2008).

In metrology the least squares adjustment is used to find the most likely parameter estimates based on redundant, noisy observations. As there are redundant observations the estimation of the parameters is over-determined and requires an additional optimization criterion. The least squares adjustment uses the minimization of the sum of squared corrections of the observations for this, giving it its name.

The n observations l_i are combined into the observation vector $\mathbf{L}$ and the u unknown parameters are likewise combined into the parameter vector $\mathbf{X}$:

$$\mathbf{L} = \begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_n \end{bmatrix} ; \mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_u \end{bmatrix} \tag{2.7}$$

The observations and parameters have to be in a functional relation, called the functional model, $\mathbf{\Phi}$, which is a vector of b functions expressing physical, geometrical or mathematical modeled relationships between the observations and parameters. Furthermore, $b > u$ is a necessary conditions for a least squares adjustment as otherwise the problem is not over-determined with $r = b - u$ denoting the redundancy of the adjustment problem. The functional model may be written as

$$\mathbf{\Phi}(\mathbf{L}, \mathbf{X}). \tag{2.8}$$

It is assumed that the true values of the measurements $\tilde{\mathbf{L}}$ and the parameters $\tilde{\mathbf{X}}$ satisfy the functional model exactly, thereby assuming that the functional model is correct, yielding equation 2.9. Furthermore, the estimated parameters $\hat{\mathbf{X}}$ and corrected observations $\hat{\mathbf{L}}$ also satisfy the functional model as is expressed in equation 2.10.

$$\mathbf{\Phi}(\tilde{\mathbf{L}}, \tilde{\mathbf{X}}) = \mathbf{0} \tag{2.9}$$

$$\mathbf{\Phi}(\hat{\mathbf{L}}, \hat{\mathbf{X}}) = \mathbf{0} \tag{2.10}$$

The least squares adjustment uses a linearization of the functional model in order to be able to solve the optimization using the methods of linear algebra. For this adequately precise approximate values of the parameters, $\mathbf{X}_0$, are needed in order to linearize the model around this point. If required, the adjustment is executed iterative until convergence of the parameter estimates is achieved. However, $\mathbf{X}_0$ has to be precise enough for the iteration to converge. The uncorrected observations $\mathbf{L}$ and the approximate values of the parameters $\mathbf{X}_0$ do not satisfy the functional model, resulting in the discrepancies $\mathbf{w}$:

$$\boldsymbol{\Phi}(\mathbf{L}, \mathbf{X}_0) = \mathbf{w} \tag{2.11}$$

The linearization of the functional model is accomplished by terminating a Taylor series expansion after the first degree as is shown in equation 2.12.

$$\underbrace{\boldsymbol{\Phi}(\hat{\boldsymbol{L}}, \hat{\boldsymbol{X}})}_{=\mathbf{0}} \approx \underbrace{\boldsymbol{\Phi}(\boldsymbol{L}, \boldsymbol{X}_0)}_{=\mathbf{w}} + \frac{\partial \phi}{\partial \mathbf{L}}|_{\mathbf{L},\mathbf{X}_0} \cdot (\hat{\mathbf{L}} - \mathbf{L}) + \frac{\partial \boldsymbol{\Phi}}{\partial \mathbf{X}}|_{\mathbf{L},\mathbf{X}_0} \cdot (\hat{\mathbf{X}} - \mathbf{X}_0) \tag{2.12}$$

New variables for the differences between the estimated values $\hat{\mathbf{L}}$ and $\hat{\mathbf{X}}$ to their original values $\mathbf{L}$ and $\mathbf{X}_0$ respectively are introduced:

$$\mathbf{v} = \hat{\mathbf{L}} - \mathbf{L} \tag{2.13}$$

$$\mathbf{x} = \hat{\mathbf{X}} - \mathbf{X}_0 \tag{2.14}$$

$$\mathbf{A} = \frac{\partial \boldsymbol{\Phi}}{\partial \mathbf{X}}\Big|_{\mathbf{L},\mathbf{X}_0} = \begin{bmatrix} \frac{\partial \Phi_1}{\partial X_1}\Big|_{\mathbf{L},\mathbf{X}_0} & \frac{\partial \Phi_1}{\partial X_2}\Big|_{\mathbf{L},\mathbf{X}_0} & \cdots & \frac{\partial \Phi_1}{\partial X_u}\Big|_{\mathbf{L},\mathbf{X}_0} \\ \frac{\partial \Phi_2}{\partial X_1}\Big|_{\mathbf{L},\mathbf{X}_0} & \frac{\partial \Phi_2}{\partial X_2}\Big|_{\mathbf{L},\mathbf{X}_0} & \cdots & \frac{\partial \Phi_2}{\partial X_u}\Big|_{\mathbf{L},\mathbf{X}_0} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \Phi_n}{\partial X_1}\Big|_{\mathbf{L},\mathbf{X}_0} & \frac{\partial \Phi_n}{\partial X_2}\Big|_{\mathbf{L},\mathbf{X}_0} & \cdots & \frac{\partial \Phi_n}{\partial X_u}\Big|_{\mathbf{L},\mathbf{X}_0} \end{bmatrix} \tag{2.15}$$

$$\mathbf{B} = \frac{\partial \boldsymbol{\Phi}}{\partial \mathbf{L}}\Big|_{\mathbf{L},\mathbf{X}_0} = \begin{bmatrix} \frac{\partial \Phi_1}{\partial L_1}\Big|_{\mathbf{L},\mathbf{X}_0} & \frac{\partial \Phi_1}{\partial L_2}\Big|_{\mathbf{L},\mathbf{X}_0} & \cdots & \frac{\partial \Phi_1}{\partial L_n}\Big|_{\mathbf{L},\mathbf{X}_0} \\ \frac{\partial \Phi_2}{\partial L_1}\Big|_{\mathbf{L},\mathbf{X}_0} & \frac{\partial \Phi_2}{\partial X_2}\Big|_{\mathbf{L},\mathbf{X}_0} & \cdots & \frac{\partial \Phi_2}{\partial L_n}\Big|_{\mathbf{L},\mathbf{X}_0} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \Phi_n}{\partial L_1}\Big|_{\mathbf{L},\mathbf{X}_0} & \frac{\partial \Phi_n}{\partial L_2}\Big|_{\mathbf{L},\mathbf{X}_0} & \cdots & \frac{\partial \Phi_n}{\partial L_n}\Big|_{\mathbf{L},\mathbf{X}_0} \end{bmatrix} \tag{2.16}$$

With the introduction of the Jacobian matrices $\mathbf{A}$ and $\mathbf{B}$ containing the partial derivatives of the functional model with respect to the parameters and the observations respectively, as is show in equations 2.15 and 2.16, the following compact notation is possible

$$\mathbf{Ax} + \mathbf{Bv} + \mathbf{w} = \mathbf{0} \tag{2.17}$$

The goal of the adjustment is to alleviate the discrepancies $\mathbf{w}$ under the condition that the sum of the squared corrections to the observations $\mathbf{v}$ is minimized. This optimization condition in vectorized from is

$$\mathbf{v}^T\mathbf{Pv} \rightarrow \min \tag{2.18}$$

wherein $\mathbf{P}$ is an optional weight matrix defining how costly corrections for the different observations should be for the optimization with respect to each other. Most often a stochastic model for the observations is used as defined in the form of a covariance matrix $\mathbf{\Sigma}_{ll}$. As only the relative weighting of the observations influences the position of the optimum, the covariance matrix of the observations is split into the covariance factor σ_0^2 and the the cofactor matrix of the observations $\mathbf{Q}_{ll}$:

$$\mathbf{\Sigma}_{ll} = \sigma_0^2\mathbf{Q}_{ll} \tag{2.19}$$

The value for σ_0^2 may be set in order to achieve more favorable values for numerical stability. Other than that the value may be chosen arbitrarily. The observations are then weighted by the inverse of the cofactor matrix.

$$\mathbf{P} = \mathbf{Q}_{ll}^{-1} \tag{2.20}$$

In the case the covariance matrix of the observations is used for the weighing as described above, the least squares adjustment is a BLUE (Best Linear Unbiased Estimator), guaranteeing an unbiased parameter estimate with the smallest possible estimation variance. If the corrections for the observations follow normal distributions the least squares adjustment is even a MLE (Maximum Likelihood Estimator), estimating the parameters which maximize the likelihood of the occurrence of the observations.

The optimization for the estimation of the parameters is carried out using the Lagrange multiplier method, minimizing the sum of squared corrections while ensuring that the (linearized) functional model is satisfied:

$$\Omega = \mathbf{v}^T\mathbf{Pv} - 2\mathbf{k}^T(\mathbf{Ax} + \mathbf{Bv} + \mathbf{w}) \tag{2.21}$$

The solution of the optimization by setting the derivatives of Ω with regard to $\mathbf{v}$, $\mathbf{x}$ and $\mathbf{k}$ to zero leads to the equation system

$$\begin{bmatrix}\mathbf{k}\\ \mathbf{x}\end{bmatrix} = \begin{bmatrix}\mathbf{BPB}^T & \mathbf{A}\\ \mathbf{A}^T & \mathbf{0}\end{bmatrix}^{-1}\begin{bmatrix}-\mathbf{w}\\ \mathbf{0}\end{bmatrix}. \tag{2.22}$$

The corrections for the observations can be calculated using

$$\mathbf{v} = \mathbf{P}\mathbf{B}^T\mathbf{k}. \tag{2.23}$$

The inverse matrix of the equation system describes how the uncertainties of the observations as given by the weighting matrix **P** are propagated into the estimated multipliers **k** and parameters **x** based on the functional model:

$$\begin{bmatrix} \mathbf{B}\mathbf{B}^T & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{Q}_{kk} & \mathbf{Q}_{xk} \\ \mathbf{Q}_{xk} & \mathbf{Q}_{xx} \end{bmatrix} \tag{2.24}$$

The cofactor matrices need to be scaled by the variance factor in order to obtain covariance matrices. The variance factor may be estimated using the corrections of the observations estimated during the adjustment, taking into account whether the calculated corrections which were necessary in order for the functional model to be satisfied are coherent with the covariance information used for the observations:

$$s_0^2 = \frac{\mathbf{v}^T\mathbf{v}}{n_f} \tag{2.25}$$

whereby n_f denotes the degree of freedom which is defined as the difference between the number of equations and parameters. The covariance matrix of the estimated parameters can then be obtained with

$$\mathbf{C}_{xx} = s_0^2\mathbf{Q}_{xx}. \tag{2.26}$$

2.5 Vector and transformation notation

For clarity the following explicit notation for vectors and transformation matrices is adopted throughout this thesis:

$$\mathbf{x}_{a,b}^c \tag{2.27}$$

is a vector pointing from point a to point b expressed with regards to the resolving axis of the frame c. When the name of a laser tracker is used in this context the origin and axis of its measurement frame are referred to. For transformation matrices the notation

$$\mathbf{C}_a^b \tag{2.28}$$

is adopted, referring to a transformation from frame a into the b frame.

3. Hardware and Software

This chapter lists the software and hardware components used for developing the measurement setup and the subsequent processing pipeline, designed for system identification of the driving dynamics of the Husky A200. The choices of the used components were at least partially dictated by the availability of equipment at the research department.

3.1 Hardware

3.1.1 Husky A200 UGV

The Husky A200 from Clearpath Robotics is a medium sized, skid-steered unmanned ground vehicle (UGV) which is intended as a robotic development platform for research and rapid prototyping. Its large payload capacity and power systems can accommodate a wide range of payloads such as sensors and make it suitable for a wide variety of tasks, among them mobile mapping. The Husky A200 is the central subject of this thesis as its driving dynamics are modeled from data using system identification. Some of its important specifications are listed in table 3.1.

The Husky A200 utilizes skid-steering featuring two geared 24V DC motors driving the left and right wheels respectively. The front and rear wheels on either side are mechanically linked by a drivebelt. While the Husky A200 can be programmed to drive autonomously, it can also be controller with a wireless joystick controller. High resolution quadrature rotary encoders are mounted on both motors, delivering precise measurements of wheel position and speed, suitable for dead reckoning and state estimation. The robot has an onboard computer running Ubuntu which controls the motors and receives sensor feedback from the wheel encoders. The Husky A200 is fully integrated into the Robot Operating System (ROS), which is explained in section 3.2.1.

At the Engineering Geodesy research department at the TU Wien the Husky A200 (serial number 0584) is used for a variety of research projects regarding mobile mapping applications. To facilitate this, a frame made out of aluminium extrusions was added to the robot for mounting different sensors. Figure 3.1 shows the Husky A200 of the research department in a typical configuration. The frame was also used in this thesis for mounting the motorized turning hubs, described in section 3.1.7. As the added hardware changes the overall mass of the vehicle and the position of the center of mass, the generalizability of the estimated model of the driving dynamics is limited.

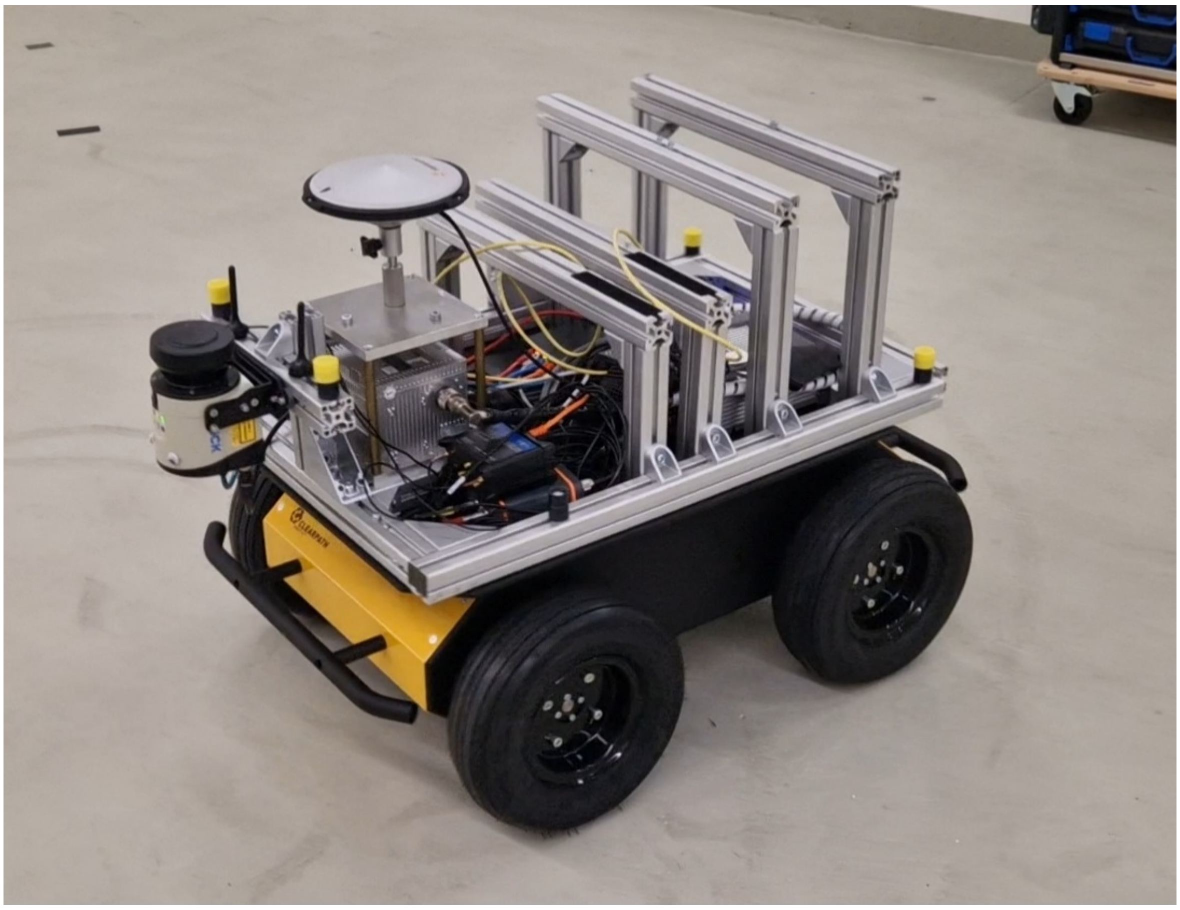

Figure 3.1: Husky A200 in a typical configuration used at the Engineering Geodesy department of the TU Wien

Coordinate system

The choice of the body frame coordinate system is important because the dynamical system will be expressed in body frame coordinates. However, the conventional way of defining vehicle coordinate systems regarding the axis directions is also well suited for this case. The chosen body frame of the Husky A200 is thus defined as follows:

- The origin is the center point of the four wheel hubs. It therefore lies on the symmetry axis of the vehicle and is equidistant from both axles.
- The xy plane of the body frame is the plane (redundantly) defined by the two axles.

Table 3.1: Specifications of the Clearpath Robotics Husky A200 UGV (Clearpath Robotics, 2016)

Parameter	Value
Dimensions	990 x 670 x 390 mm
Weight	50 kg + additional hardware
Max. payload	75 kg
Max. speed	1.0 m/s
Battery	Sealed Lead Acid 24 V, 20 Ah
Encoders	Quadrature, 78,000 pulses/m

- The positive x direction is along the symmetry axis of the vehicle, pointing in the direction of forwards travel.
- The z axis is normal to the xy plane and thus the two axles, pointing upwards.
- The y axis completes the right handed coordinate system, pointing to the left of the vehicle when seen in the forward driving direction.

The resulting reference frame is illustrated in figure 3.2.

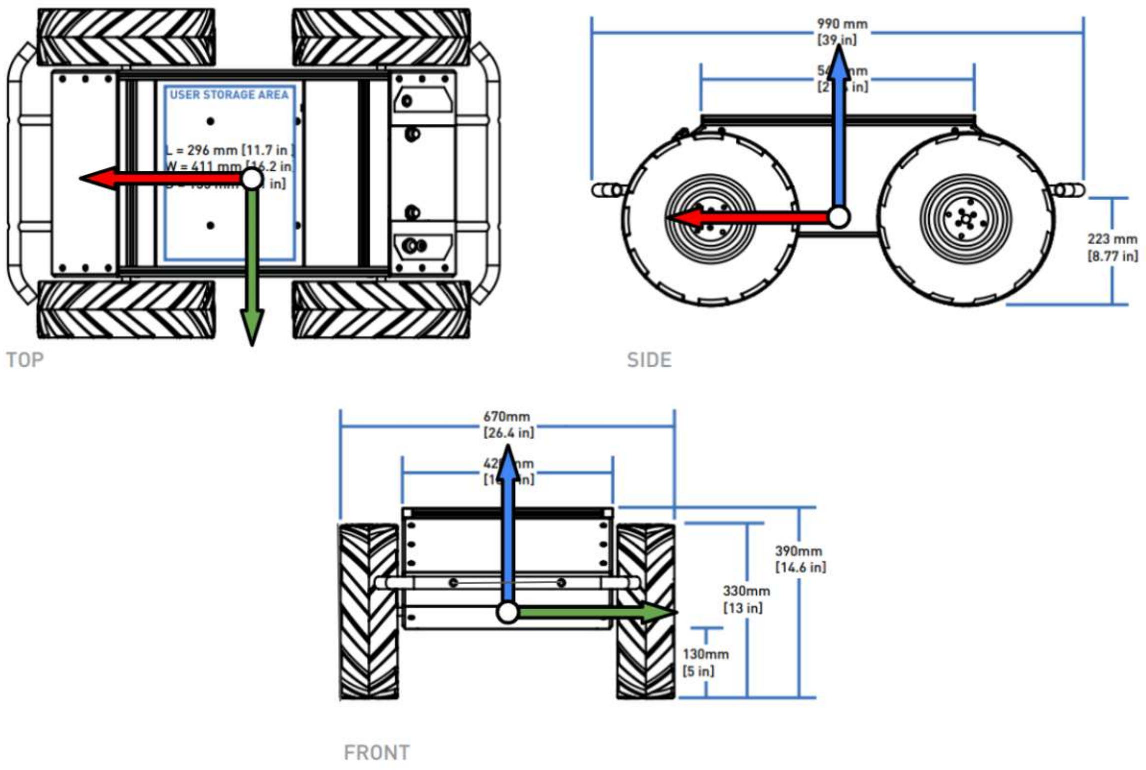

Figure 3.2: Visualization of the chosen body frame. Blueprint from Clearpath Robotics (2016) overlayed with the resulting axes: x - red, y - green, z - blue

Tires

The Husky A200 is designed as a rugged all-terrain vehicle and is delivered with lugged off-road tires. However, the Engineering Geodesy research department at the TU Wien predominantly uses the vehicle as a mobile mapping platform on paved surfaces. Therefore the stock tires were replaced by aftermarket indoor tires with much less pronounced tread pattern. This led to smoother driving dynamics with reduced vibrations of the robot as the tire-ground-interaction is more uniform throughout the rotation of the wheels. The reduced vibrations, which were especially present when turning in place on smooth, grippy surfaces with the original tires, are beneficial for dead reckoning applications using inertial measurement units as well as for the system identification of the driving dynamics discussed in this thesis. Figure 3.3 depicts the original lugged off-road and the smooth aftermarket tires side by side.

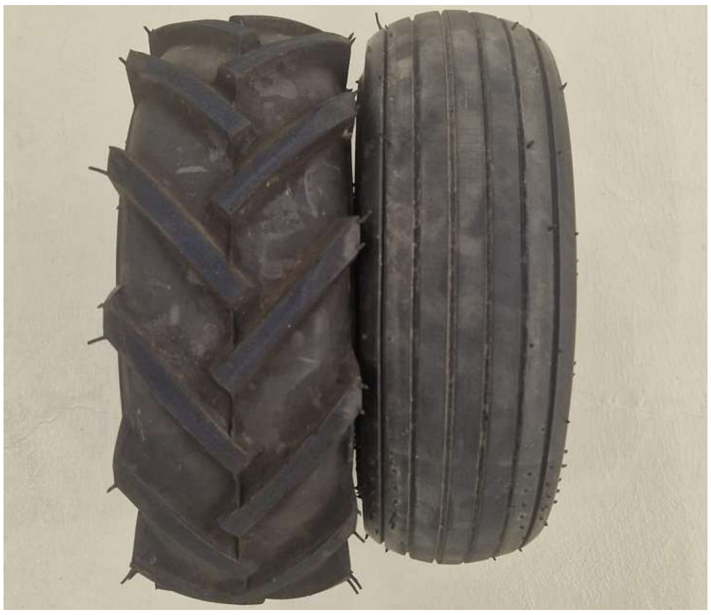

Figure 3.3: Comparison of the original lugged off-road tire (left) and the smooth aftermarket tire which was used for this thesis (right)

3.1.2 Laser trackers

Laser trackers are precise 3D point measurement devices. They measure the spherical coordinates of a retro reflector by means of laser distance measurement and angular encoders. Furthermore, they automatically track the measured target by measuring the returning laser beam on a biaxial Position Sensing Device. The readings of this sensor are used in a closed-loop control loop driving motors (Leica Geosystems, 2020). This capability and their high measurement frequency makes laser trackers suitable for dynamic measurement tasks. In this thesis two Leica laser trackers, the LTD800 and the AT960, were used for collecting a time series of pose data of the Husky A200 during driving as well as for preparatory measurements. The trackers are depicted in figure 3.4 and some of their important properties are listed in table 3.2.

One of the largest error sources for laser tracker measurement are the meteorological influences on the refractive index of the surrounding atmosphere Joeckel et al. (2008). The AT960 features a meteo station integrated in its controller with the possibility of an additional external probe. The station measures air temperature, pressure and humidity and calculates the current refractive index, correcting its influence on the measured distance (Leica Geosystems, 2020). The LTD800 can be connected to an external meteo station working in the same way. However, due to a technical defect this was not possible for the specific unit used and the meteorological measurements were periodically manually performed with a Meteo Station HM30 from Huber Instrumente. The measurements were input in the control software of the LTD800 in order to be used for the corrections.

Both trackers provide a trigger input interface at their respective controller in order to enable external triggering and synchronization of their measurements. This was utilized in the measurement setup in order to synchronize the different system components. They can accept both balanced RS-422 and single wired input signals via their trigger input connector, which is a 15 pin DSUB connector. The trigger signal does not alter the measurement process itself as the different sensors within the laser trackers are polled at regular, predetermined intervals. Instead the trackers register the time stamp of the incoming trigger signal in their internal time system and interpolate the measurements of the individual system components at the trigger time stamp based on the two closest internal measurements (on prior and one past) (Leica Geosystems, 2013). A timestamp of the trackers internal system time is supplied with each measurement based on the internal clock of the tracker controllers. The internal clocks of the trackers can drift with the manuals stating that a drift of 10-20 ppm is "not unusual" and their crystal oscillator has an overall stability of ±100ppm (Leica Geosystems, 2008).

In contrast to the LTD800, the AT960 features a dual-axis Orient-to-Gravity sensor located within the rotating head of the instrument (Leica Geosystems, 2020). This was used in the thesis in order to obtain poses of the Husky A200 with regard to a horizontal local reference frame.

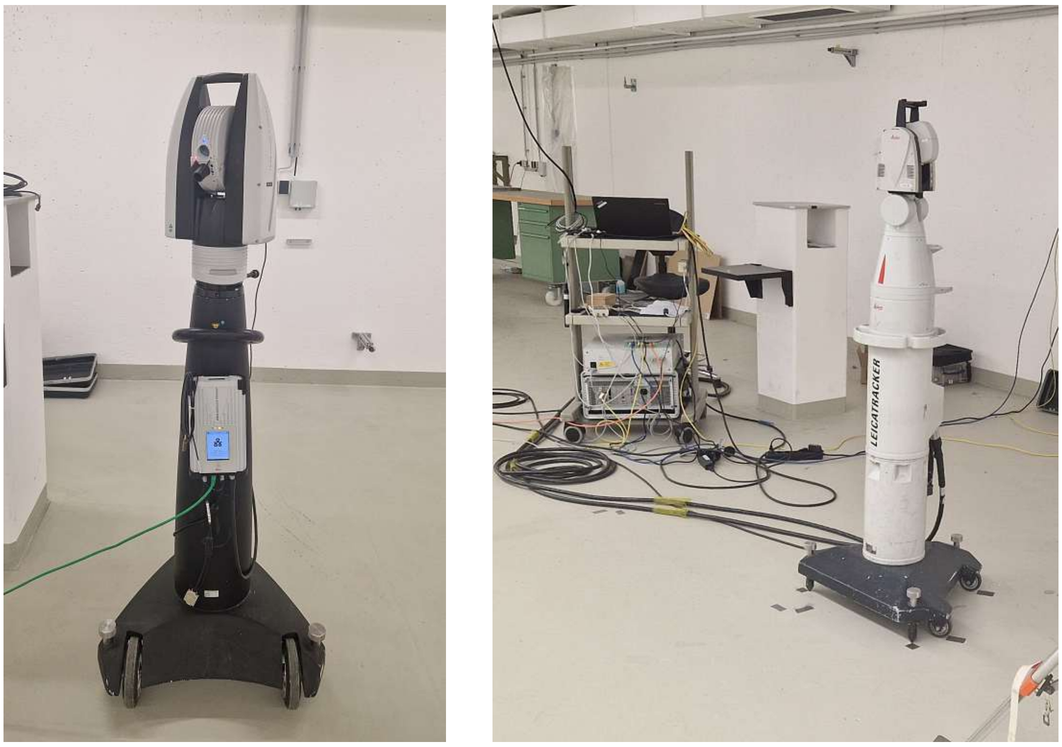

Figure 3.4: The Leica AT960 laser tracker (left) and the Leica LTD800 laser tracker (right)

3.1.3 Red Ring Reflector

The Leica Red Ring Reflector (RRR) is a retro reflector used as a target for Leica laser trackers. It is comprised of an optical corner cube reflector (CCR) housed in a spherical ball made of surface-hardened steel. The reference point of the reflector is designed to coincide with the center of the ball, which enables to turn the target so that the reflector faces the laser tracker without altering the measured point. The ball is ferromagnetic and is often held in consoles with magnets, facilitating easy orientation adjustments. The technical specifications of the used RRR are listed in table 3.3 and a RRR in a magnetic console is depicted in figure 3.5.

Table 3.2: Important parameters of the two trackers from (Leica Geosystems, 2020), (Leica Geosystems, 2003) and (Leica Geosystems, 2013)

	Parameter	AT960	LTD800
Tracking	Maximum target speed		
	At right angles to the beam	$160gon/s$	$> 4m/s$
	In beam direction	$6m/s$	$> 6m/s$
	Maximum Acceleration		
	At right angles to the beam	$1500gon/s^2$	$> 2g$
	In beam direction	$180m/s2$	unlimited
Angular Measurements	Angular Resolution	0.07”	0.14”
	Repeatability		
	Near (0 – 2.5 m)	$\pm 7.5\mu m + 3\mu m/m$	$\pm 12\mu m$
	Far (2.5 m to max distance)		$\pm 5\mu m/m$
	Absolute accuracy		
	Stationary target (0 – 2.5 m)	$\pm 15\mu m + 6\mu m/m$	$\pm 25\mu m$
	Stationary target (2.5to max)		$\pm 10\mu m/m$
	For slow moving target	N/A	$\pm 20\mu m/m$
	For fast moving target		$\pm 40\mu m/m$
Laser Interferometer	Distance resolution	$0.4\mu m$	$1.26\mu m$
	Reproducibility of a coordinate	N/A	$\pm 5\mu m/m$
	Absolute accuracy		
	Wave length stabilization	$\pm 0.4\mu m + 0.3\mu m/m$	$\pm 0.5\mu m/m$
	Initial distance		± 10 micron
Absolute Distance Measurement	Resolution	$0.3\mu m$	$1\mu m$
	Accuracy	$\pm 10\mu m$	$\pm 25\mu m$
Triggering	Internal Clock Resolution	$0.1\mu s$	$1\mu s$
	Trigger time accuracy	$5\mu s$	
	Max Data output	1000 points / s	

Table 3.3: Specifications of the Leica Red Ring Reflector 1.5 inch (RRR) (Hexagon Metrology, n.d.)

Parameter	Value
Size	$38.1\ mm$ ($1.5''$)
Radius tolerance	$\pm 2.5\ \mu m$
Centering of reflector	$\pm 3\ \mu m$
Roundness	$\leq 3\ \mu m$
Acceptable incident angle	$\pm 30°$

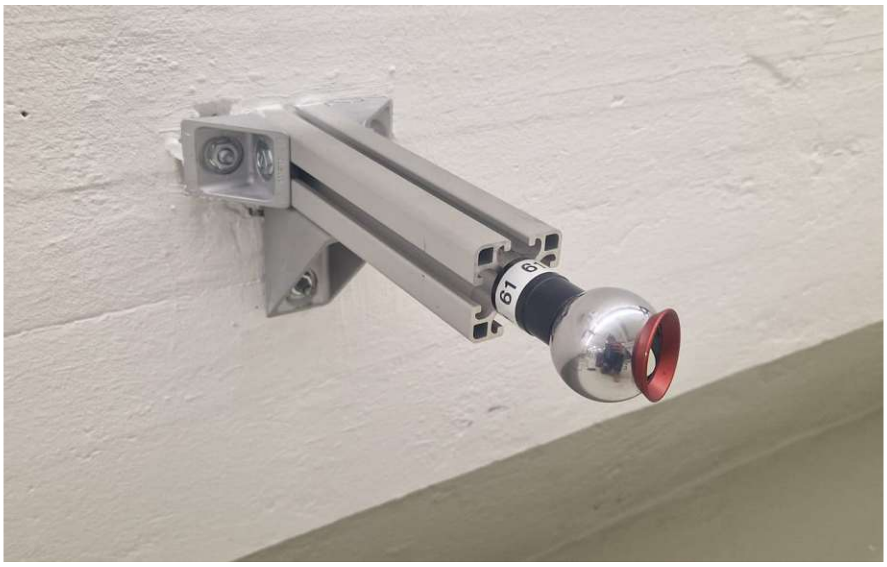
Figure 3.5: RRR in a magnetic console in the measurement laboratory

3.1.4 T-Probe

The Leica T-Probe is a wireless measuring device for probing of hidden, hard-to-reach points, that is used in conjunction with a Leica laser tracker in combination with a T-Cam. It features a long and thin probe whose tip is held against the object that is to be measured. A corner cube retro reflector in the T-Probe is used as a target for the laser tracker, in the case of this thesis the LTD800. In order to obtain the coordinates of the probe tip the leverarm between the corner cube reflector and the tip has to be applied. The T-Probe features multiple infrared LEDs whose positions are measured by the T-Cam, an optional accessory of the LTD800. Those measurements are used to calculate the attitude of the T-Probe which enables to perform a coordinate transformation between the sensor frame of the T-Probe, in which the leverarm is expressed, and the sensor frame of the laser tracker. Thus the position of the probe tip in the sensor frame of the laser tracker can be calculated. As both the position and attitude of the T-Probe are measured it is called a 6 DOF (degree of freedom) measurement technique. In this thesis the T-Probe was used in the context of the Husky A200 body frame realization.

3.1.5 Raspberry Pi 4B

Raspberry Pi is a family of small single-board computers developed by the Raspberry Pi Foundation. Originally intended as learning platforms, their low cost, modularity, and open design has made it popular for other use-cases such as robotics. The Raspberry Pi 4B is the newest version of the computer, originally released in 2019, of which two are utilized in the measurement setup of this thesis. One is used for controlling the motorized turning hubs described in section 3.1.7 and one for the generation of an external trigger signal for the laser trackers.

Raspberry Pis offer a low power draw, which enabled powering one of them directly of an USB port of the onboard computer of the Husky A200. Furthermore, they feature a J8 header consisting of 40 pins amongst which so called general purpose input output (GPIO)

Table 3.4: Selection of specifications of the Raspberry Pi 4B (Ltd, n.d.)

Parameter	Value
CPU	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
RAM	4 GB (the model used)
Network connectivity	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless Gigabit Ethernet

pins facilitate low-level communication such as with the I^{2}C and SPI interfaces. GPIO pins can also be controlled individually as outputs or read incoming signals as inputs by software running on the Raspberry Pi. Therefore Raspberry Pis enable easy interaction between low level hardware and high level programming languages such as Python. In this thesis GPIO pins were utilized for precise GNSS time synchronisation using a pulse per second (PPS) signal as well as for creating an external trigger signal for both laser trackers as is described in section 4.2.

3.1.6 GNSS equipment

GNSS repeater

In order to provide GNSS signals to receivers in the measurement laboratory, located in the second basement of the TU Wien university building at Gußhausstraße 27/29, 1040 Vienna, the GNSS signals received by a GNSS receiver on the roof are transmitted via cable to a transmitter on the ceiling of the laboratory, which can be seen in figure 3.6. There the signal is re-transmitted and reaches GNSS receivers in the laboratory. The additional propagation time through the cable and though the air in the laboratory is equal for the signal from all GNSS satellites. This additional delay is thus absorbed by the estimated receiver clock offset and a GNSS receiver utilizing this setup will behave as being at the location of the receiver on the roof at the point in time of signal reception there.

Septentrio receiver

The Septentrio AsteRx SB3 is a multi-frequency, multi-constellation GNSS receiver. It was used in conjunction with the Septentrio PolaNt-x MF antenna which is a high-precision antenna for geodetic, surveying and machine control application that incorporates low-noise amplifiers, enabling multi-frequency GNSS signal reception. The receiver can communicate via Ethernet (TCP/IP, UDP, LAN 10/100 Mbps) over a hosted web server. Besides position information the receiver can also provide an NTP server for time synchronization. This feature was used to synchronize the onboard computer of the Husky A200. The receiver can be powered by supplying 5 to 36 V DC via the power connector, which made it possible to power it from one of the Husky A200 onboard power outlets.

u-blox receiver

The u-blox C94-M8P is an application board for prototyping which features the u-blox NEO-M8P-2 GNSS module which is a compact, high precision multi GNSS receiver. As interfaces

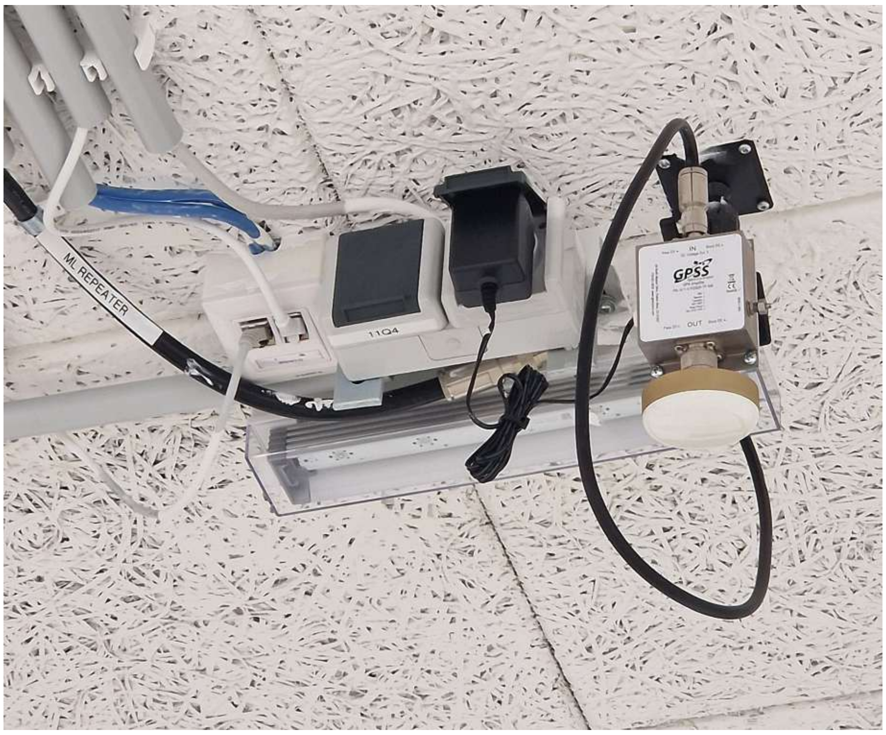

Figure 3.6: GNSS signal repeater on the ceiling of the measurement laboratory

it offers one USB port which is used for GNSS data transmission and to supply power to the receiver. Furthermore is features connection pins for UART communication in the form of a 20-pin J8 connector. One of this pins transmits a pulse-per-second (PPS) timepulse signal. In this thesis the u-blox C94-M8P is used for time synchronization of a Raspberry Pi utilizing the GNSS data via USB port in conjunction with the PPS signal.

3.1.7 CCR Turning Hub

In order to constantly maintain a line of sight between the laser trackers and their respective prism during driving operations of the Husky A200, two motorized turning hubs were designed and built. They consist of a 3D printed hub mounted on the shaft of a NMEA17 stepper motor. The motor was afixed to the alumium extrusions o the husky with a 3D printed mounting plate. The hub and mounting plate were designed using Autodesk Fusion 360 and printed with PLA using a original Prusa i3 MK3S+ 3D printer. The RRR is held in place in the hub by an embedded magnet held in place by a circlip, securing the ferromagnetic RRR while still allowing for quick adjustments of the orientation during the setup process. The assembly can be seen in figure 3.7 and table 3.5 lists important characteristics of the used stepper motor. The stepper motors were driven by two DRV8825 stepper motor drivers on breakout boards which were in turn controlled by an Arduino Nano micro controller board. The control curcuit was realized using a perfboard which is depicted in figure 3.7 and its wiring schematics used are depicted in figure 3.8.

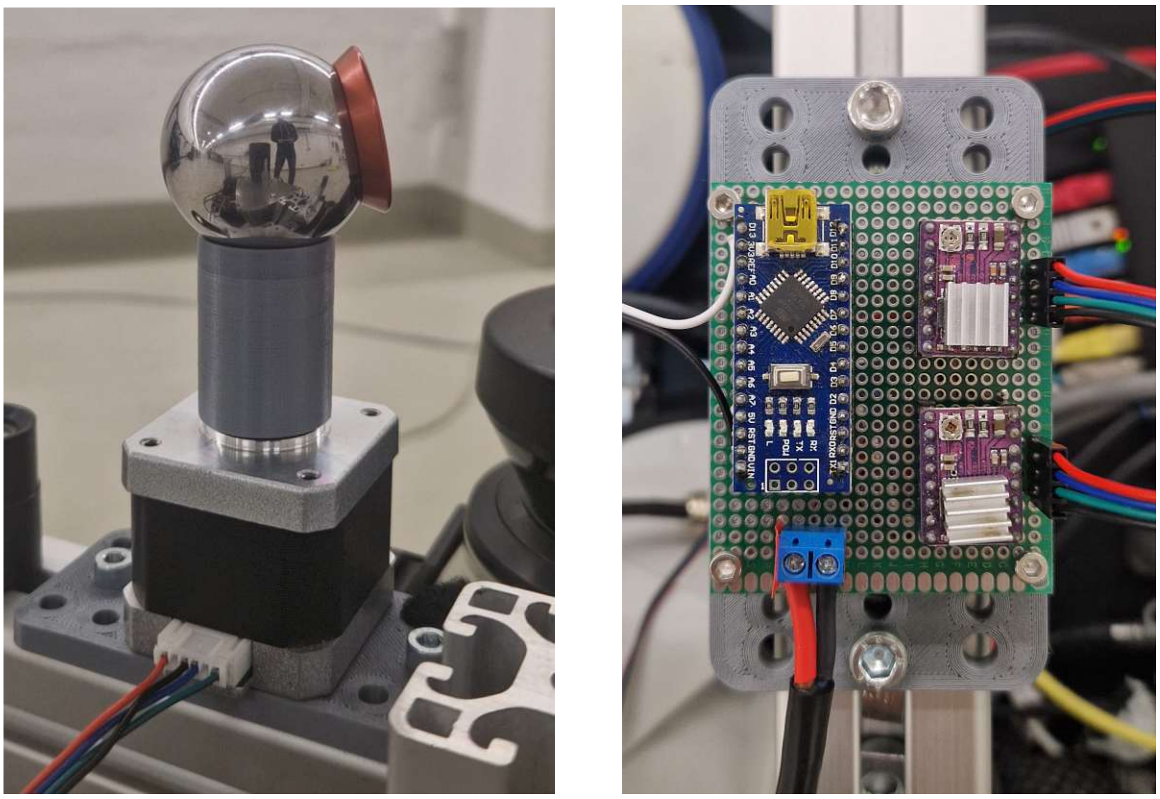

Figure 3.7: The turning hub mounted on the Husky (left) and the the control electronics for the turning hub mounted on the Husky (right)

Table 3.5: Specifications of stepper motors used for the turning hub

Parameter	Value
Step size	1.8°
Size standard	NEMA17
Maximum torque	42 Ncm
Maximum current draw	1.5 A

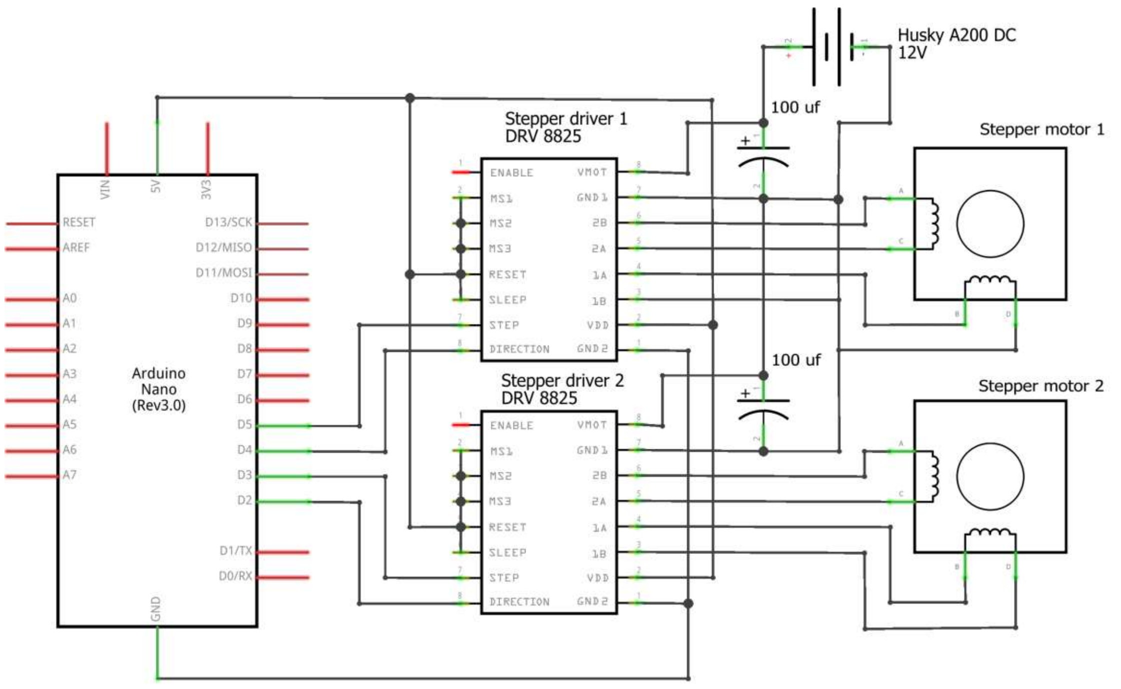

Figure 3.8: Wiring diagram of the controller for the motorized turning hub

3.1.8 Sensor network

The sensor network is a computer network created and used by the geodesy department at the TU Wien to enable the communication between different sensors and computers. It is a multi-building network that is accessible by both Ethernet and WiFi for example in the measurements laboratory where the experiments for this thesis were conducted. The network was used to control different system components such as Raspberry Pis, laser trackers and the Husky A200 remotely, to relay measurement data in real time as needed for the motorized turning hub and to collect and record the measurement data.

3.2 Software

3.2.1 Robot Operating System

The Robot Operating System (ROS) is an open-source robotics software framework providing libraries, visualizers, message-passing, package management, hardware abstraction, low-level device control and more (Stanford Artificial Intelligence Laboratory et al., 2018). It is not a true operating system but a meta-operating system running atop an operating system such as Linux. ROS applications run in a graph network with different program parts running in nodes. The resulting peer-to-peer network of processes communicate among other methods via asynchronous streaming of data as so called messages over topics. ROS features the capability to record the data streams on these topics as so called rosbags. This was used for recording the data needed for this thesis.

The Husky A200 is fully integrated into the ROS framework and its control algorithms are realized as ROS nodes. Therefore ROS was used in order to tab into and record the stream of control inputs given to the robot. Furthermore, the measurement data from the two laser trackers were also recorded as rosbag files and also used to calculate the needed control inputs for the motorized turning hubs. The rosserial ROS package allows for communication between ROS running on a computer and a microcontroller (Ferguson, 2022). It was used in this thesis to relay the calculated control inputs for the motorized turning hubs, described in section 3.1.7, to the Arduino Nano micro controller.

3.2.2 Network Time Protocol

The network time protocol (NTP) is a network protocol used to synchronize all participating computers on a network. The system achieves synchronisation of a couple of milliseconds when synchronizing to a server over the internet and can also achieve sub one millisecond accuracy in local area networks under ideal conditions.

A client polls one or more NTP server while marking the timestamp of its system time of the request. The server receives the request and sends the timestamp of its system time back to the client. The client receives this data and again saving the timestamp of its own system time. The client must then compute the round trip delay of the signal and using this its time offset to the time system of the NTP server. The client then gradually adjust its system time and estimates parameters of a clock correction model.

Accurate synchronization is achieved if the signal propagates from and to the NTP server take the same amount of time. If this is not the case, there will be a systematic bias of half the difference between the to and from travel times. Such asymmetric travel times and network congestion can cause errors of 100 ms or more.

In this thesis the implementations chrony and the Network Time Protocol daemon (ntpd) are used on two Linux machines. Chrony is used on a Raspberry Pi to synchronize the system clock using GNSS data and ntpd is used on the Husky A200 onboard computer to synchronize it with the NTP server supplied by the Septerntrio receiver.

3.2.3 SpatialAnalyzer

SpatialAnalyzer (SA) is a proprietary 3D graphical software for using metrology instruments and analysing the created measurements (New River Kinematics, 2020). SA offers interfaces to a wide range of instruments such as Laser Trackers, GNSS, Total Stations, Laser Scanners and more. It can also interface with multiple instruments simultaneously. The software can fit different geometrical objects to measured data, calculate transformation parameters between different reference frames, compare measurements to reference models and more. The software is intended predominately for large-scale applications of manufacturing, maintenance and inspection in engineering sectors such as automotive, aviation, ship building, etc.

In the context of this thesis is was used for controlling the two laser trackers during the measurements needed for estimating the transform parameters between the trackers and for realizing the Husky A200's body frame. Subsequently it was used to calculate the transformation parameters between the two laser tracker sensor frames, using the USMN (Unified Spatial Metrology Network) function.

Additionally, SA was used to take the necessary measurements for the realization of the Husky A200 body frame and the determination of the leverarms to the two CCRs. The fitting of shapes was then used in conjunction with SA ability to construct frames based on measured points or derived geometric objects to realize the vehicles body frame.

3.2.4 Python

Python is a high level, multi-paradigm programming language. It ranks as one of the most popular languages and is especially used in scientific computing. Its version 3.10.8 was used for the majority of data analysis both with regard to preprocessing the data, creating input data for system identification and conduction the system identification itself. A variety of packages were used in the context of these tasks, the most important of which are listed below.

- Numpy (1.23.5): Provides a multidimensional array object and accompanying derived objects and an assortment of functions for working with matrices, including mathematical, logical, shape manipulation, sorting, selecting, discrete Fourier transforms, basic linear algebra, basic statistics and more (Harris et al., 2020).
- Matplotlib (3.6.2): Data visualization library for creating static, animated, and interactive plots (Hunter, 2007). All plots in this thesis were creating using this package.
- Scipy (1.9.3): SciPy provides fundamental algorithms for scientific computing amongst which are optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations, statistics and more (Virtanen et al., 2020). In this thesis it was used for data interpolation, numerical integration and numerical solutions of initial value problems.
- Sympy (1.11.1): Package for symbolic mathematics (Meurer et al., 2017). It was used for symbolically calculating Jacobian matrices in the context of linearized least-square adjustments.
- PySINDy (1.7.2): PySINDy implements the sparse system identification method SINDY used in this thesis along with several supporting features (de Silva et al., 2020a), (Kaptanoglu et al., 2022).
- RPi.GPIO python library enables python scripts to easily interface with the GPIO pins of Raspberry Pis (Croston, n.d.).
- Scikit-learn (1.2.2): Library for facilitating machine learning in Python (Pedregosa et al., 2011). It was used to conduct a gridsearch with k-fold cross validation during the tuning of the hyperparameters.

4. Measurement Setup

In this chapter the measurement setup of this thesis is described. The goal of this setup was to measure the necessary data in order to conduct system identification of the driving dynamics of the Husky A200. As will be explained in section 5.3.4 this amounts to a time series of poses of the vehicle, also called a trajectory, along with a time synchronized recording of the control inputs given during driving.

A pose, consisting of the position and attitude of a reference frame with regard to another, represents the parameters for transforming between the two. Transformation parameters, or poses, may be estimated from measured points whose coordinates are given in both reference frames, with a two dimensional pose requiring two points and a three dimensional pose requiring three. As the experiments and calculation in this thesis are only conducted for driving on level ground, assumptions about one rotation parameter of the vehicle could be made, enabling to estimate a reduced three dimensional pose based on two points. This process will be explained in section 5.2.4. Furthermore, the system identification of the driving dynamics also requires the control input given to the vehicle during driving.

Thus the goal of the measurement setup was to measure a time series of two points on the vehicle during driving and simultaneously log the control inputs given to drive the vehicle. The coordinate measurements of the two reference points on the vehicle were accomplished using the two laser trackers, Leica AT960 and Leica LTD800, described in section 3.1.2. The reference points were realized as two CCRs mounted on the vehicle, for which RRR, described in section 3.1.3, were used.

The measurement setup was then designed to solve the following challenges:

- Time synchronization of the two laser trackers
- Time synchronisation of the Husky onboard computer for the timestamps of the recorded control inputs
- Creation of a geodetic network in order to obtain the relative orientation of the laser trackers
- Maintaining a line of sight between the two laser trackers and their respective target CCR during driving of the vehicle

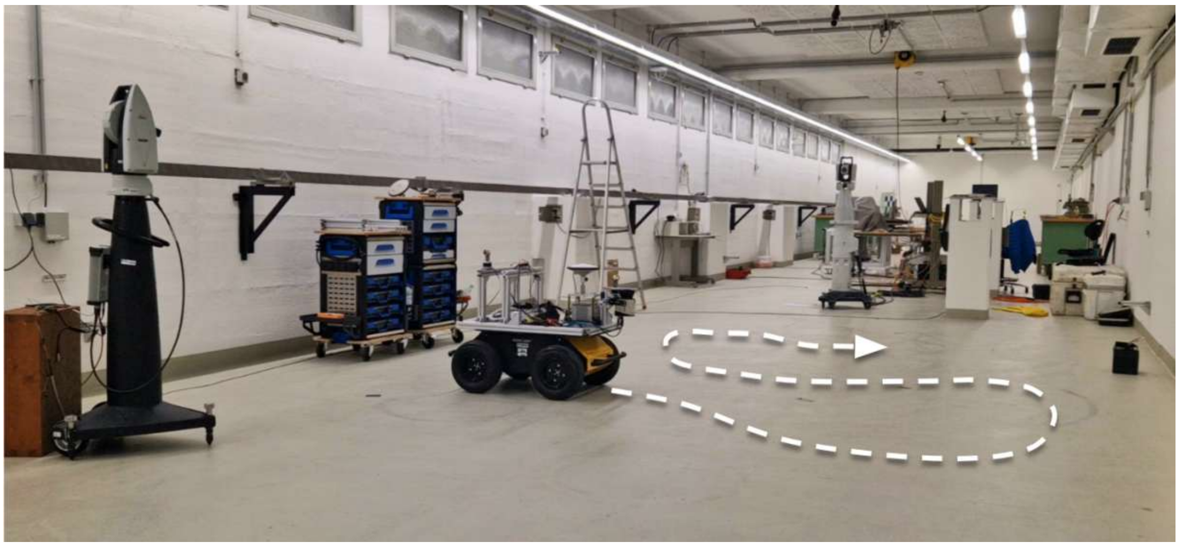

Figure 4.1: Overview of the measurement laboratory with the two laser trackers and the Husky A200 with an illustrated driving path used during the experiments

4.1 Physical layout

All measurements were conducted in the measurement laboratory of the Engineering Geodesy department of the TU Wien, located at Gußhausstraße 25-27, 1040, Vienna, Austria. The two laser trackers were set up around 9 meters apart with an area dedicated to driving of the Husky A200 during experiments between them. Figure 4.1 shows an overview of the measurement laboratory with the two laser trackers and the Husky A200 along with an illustrated driving path. The AT960 laser tracker was leveled using the built in dual-axis Orient-to-Gravity sensor mentioned in section 3.1.2. Both trackers were connected to the sensor network, described in section 3.1.8, via Ethernet in order to be controlled over the network and also to transmit their measurement data. The measurement laboratory features multiple wall mounted magnetic hubs for CCR to be held in, one of which is depicted in figure 4.3. They were used to measure identical points with both laser trackers in order to establish a geodetic network and to calculate transformation parameters between the sensor frames of the two laser trackers. The magnetic hub made it easy to turn the CCR for facing one of the two trackers without changing the reference point. The locations of the used magnetic hubs along with the location of the trackers and the ares used for driving during the experiments are depicted in map in figure 4.2.

4.2 Time synchronization

Three system components of the measurement setup needed to be time synchronized: the two laser trackers AT960 and LTD800 and the onboard computer of the Husky A200. For this the two laser trackers were triggered simultaneously by the same trigger signal created by a Raspberry Pi, described in section 3.1.5. In order to synchronize the tracker measurements with the data recorded by the Husky A200 onboard computer both it and the Raspberry Pi used for creating the trigger signal for the trackers were synchronised to the UTC time system using two different GNSS receivers. Figure 4.4 depicts an overview of the different

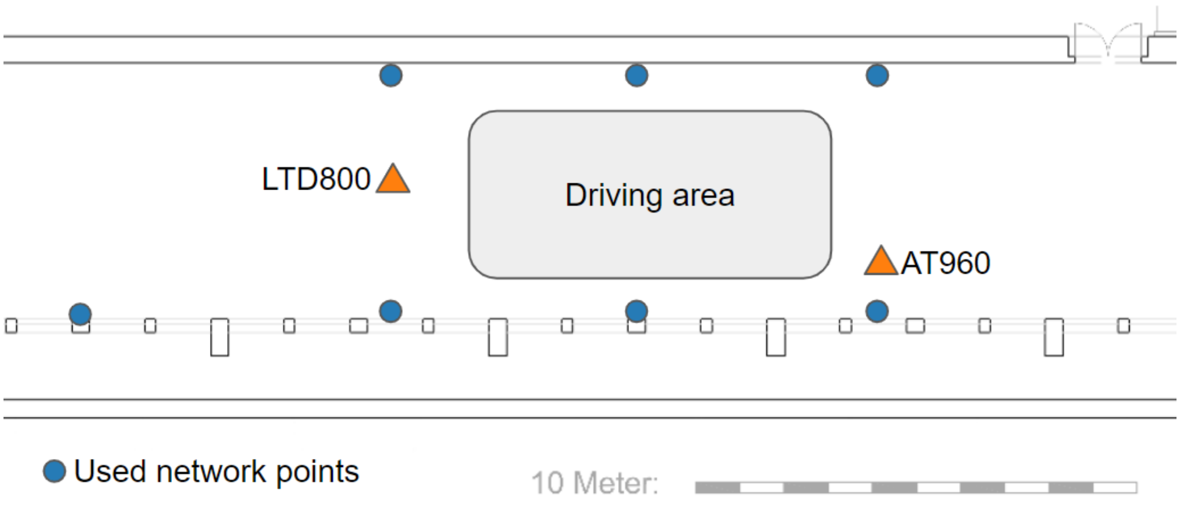

Figure 4.2: Schematic map of the measurement setup in the measurement laboratory

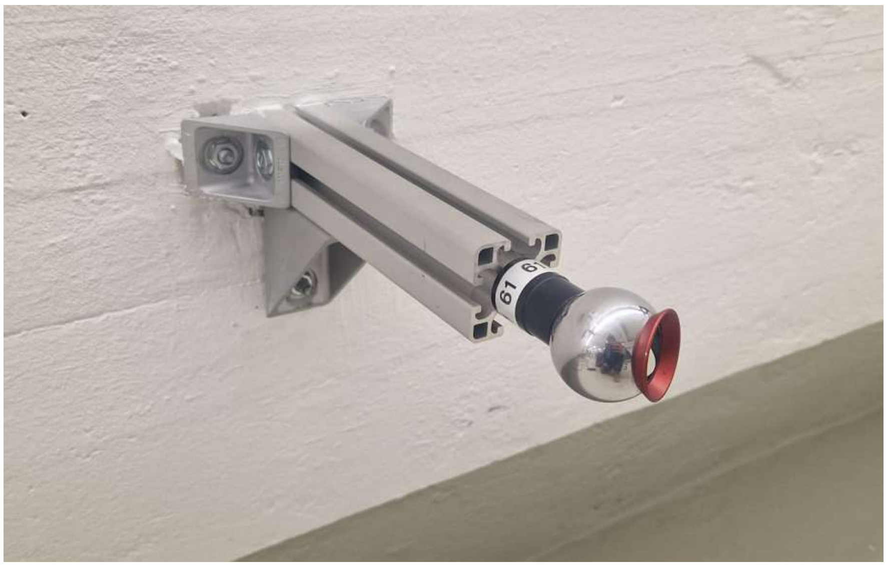

Figure 4.3: RRR in a magnetic console in the measurement laboratory

components to be synchronized and how different signals were used to accomplish this.

4.2.1 Laser tracker triggering and time stamp generation

The measurements of the two laser trackers needed to be synchronised as simultaneous point coordinate measurements of the two reference points on the Husky A200 are required in order to perform the pose calculations. Additionally, the measurements needed precise time stamps for synchronizing them to the recorded control inputs and because the time series of poses is used to calculate the derivatives of its component numerically for the system identification.

In order to synchronise the two laser trackers a Raspberry Pi was utilized to create a square wave trigger signal on one of its GPIO pins, going from 0V to 3.3V. This is accomplished by a custom ROS Python node which runs at 50 Hz and switches the pin high and low using the RPi.GPIO Python library. After the trigger signal is set to high, the script in the node immediately saves the current system time of the Raspberry Pi in a variable and subsequently publishes a timestamp as a ROS message on a dedicated ROS topic. After this is done the

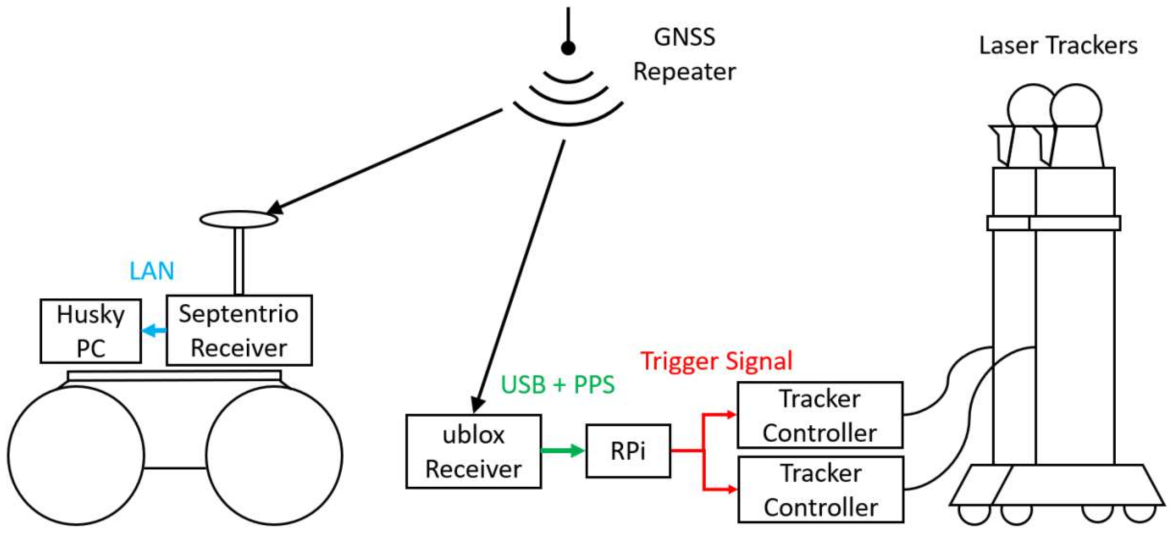

Figure 4.4: Overview of the time synchronization of the different system components

script waits for 1 ms before the pin is set to low again. This led to a slightly varying duty cycle of the trigger signal as the time needed to execute the script varies slightly with the publishing of the ROS message likely contributing most to the variations.

The trigger signal generated by the Raspberry Pi was then transmitted to the controllers of both trackers via cables which were connected to the corresponding pins of the trigger input connectors. The laser trackers were configured to accept external interrupts according to their manuals using custom command line tools utilizing the provided programming interfaces of the two laser trackers. The Raspberry Pi was located close to the the controller of the LTD800 so the cable for its connection was about 1 m long while the cable to the AT960 controller was about 15 m long and was laid around the driving area illustrated in figure 4.2. Both laser trackers can accept a trigger signal in the form of a balanced RS-422 signal. This is a standard of a digital signaling circuit designed for signal transmission over wire that is more robust against interference. Due to missing of the required supporting hardware to create such a signal and time constraints the trigger signal was transmitted as a single wired signal instead. However, the long cable to the AT960 controller picked up noise from interference caused by other devices and signals around it. This led to some trigger events not being recognized by the tracker and some erroneously triggered measurements. It is thus highly recommended to use the RS-422 standard for the transmission of the trigger signal along with the appropriate shielded, twisted pair wires. Such an upgrade is planed for further work using this measurement setup.

As the assignment between the time stamps published by the Raspberry Pi and the laser tracker measurements could not be made using their respective indices, because of the aforementioned interference caused problems, the ROS node script was programmed to skip a trigger event every three seconds. This resulted in easily identifiable gaps in the time line of the trigger events of all three systems which could be used for the synchronisation of their time systems as is described in section 5.2.2.

In order to guarantee an accurate system time of the Raspberry Pi and by extensions the time stamps of the trigger events associated with the trigger signal for the laser trackers the Raspberry Pi was synchronized using GNSS signals. For this the u-blox C94-M8P GNSS receiver development board, described in section 3.1.6, was connected to the Raspberry Pi via USB for the GNSS data and using jumper cables in order to connect the ground and PPS pin of the receiver with the GPIO pins of the Raspberry Pi. As the measurement laboratory is located in the second basement normally no GNSS signals are available there and the GNSS repeater located on the ceiling of the laboratory was used. It relays the GNSS signals received at the roof of the building and it is described in section 3.1.6. The chrony software, described in section 3.2.2, was then used to synchronise the Linux system time of the Raspberry Pi using the GNSS data, utilizing additional Linux packages with the most notable being pps-tools, gpsd and gpsd-clients. By using the PPS signal from the receiver the a sub microsecond synchronization could be achieved. Thus the timestamps published by the Raspberry Pi, corresponding to every rising edge of the created trigger signal, were sufficiently accurate for the measurement task.

4.2.2 AT960 synchronization setup error

Due to an unfortunate user error the AT960 laser tracker was configured to be triggered at the falling edge of the trigger signal instead of the rising edge. Therefore its measurements did not correspond to those of the LTD800 tracker and also not to the recorded time stamps of the Raspberry Pi. The variable duty cycle of the square trigger signal further exacerbated this problem as the offset between the measurements and the time stamps was not constant. Due to a technical defect of the Husky A200 and time limitations further measurements with a corrected measurement setup were not feasible. However, the time stamps of the AT960 could be corrected during the processing of the data as is described in section 5.2.2.

4.2.3 Husky GNSS synchronization

The control inputs for the Husky A200 are recorded using its onboard computer. In order to synchronize this data with the laser tracker measurements the computer system time needs to be synchronized. For this the Septentrio AsteRx SB3 GNSS receiver and the Septentrio PolaNt-x MF GNSS antenna, both described in section 3.1.6, were mounted to the vehicle. Like the GNSS receiver of the Raspberry Pi described above, it also received the GNSS signals from the GNSS repeater mounted on the ceiling of the measurement laboratory. The receiver was connected to the onboard computer using a router also mounted to the Husky A200. The receiver was powered using one of the onboard power outlets. On the onboard computer running Linux the NTP software ntpd, described in section 3.2.2 was configured to only use the NTP server hosted by the GNSS receiver to be used. With the system time of the computer being synchronized the time stamps of the data recorded were also synchronized.

4.3 CCR turning hub

In order to maintain a constant line of sight between the laser trackers and their respective CCR prism, which represent the reference points for the pose calculations, the prisms were seated in the custom designed and manufactured motorized turning hubs. The hubs are

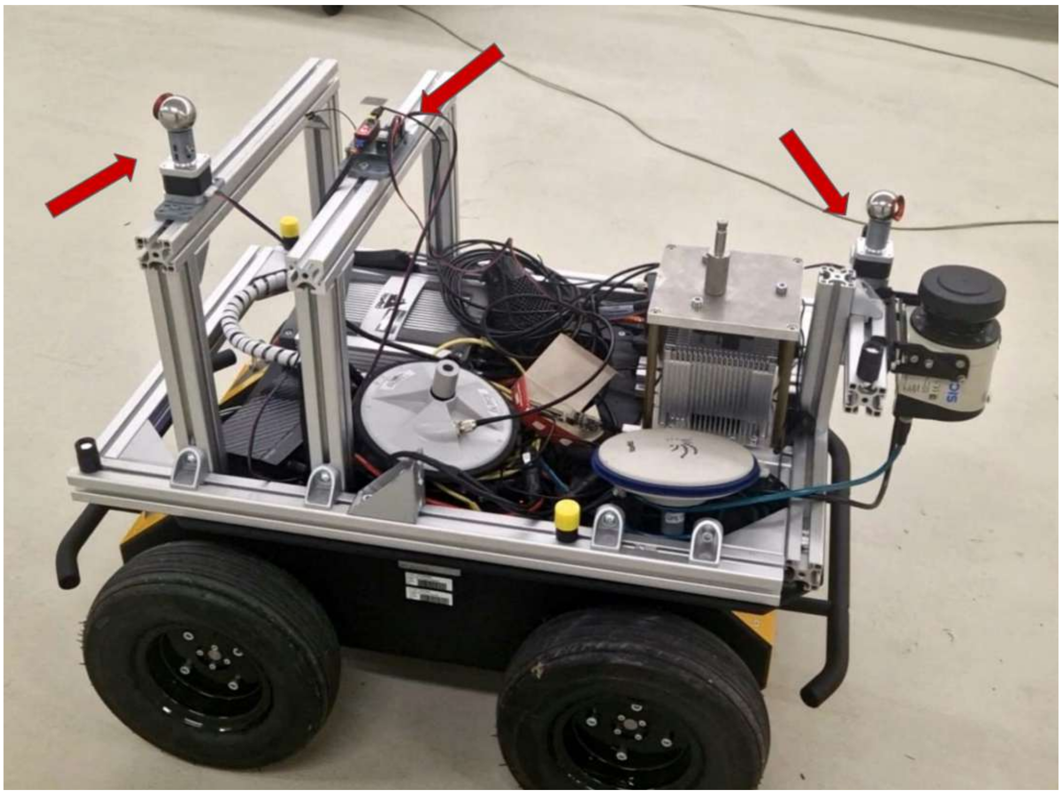
Figure 4.5: The Husky A200 with aluminum extrusions and the two motorized CCR turning hubs (left and right) and their control board (center) highlighted

described in section 3.1.7 and were mounted on the Husky with screws using the aluminum profiles on the vehicle. Figure 4.5 shows the the used setup with the turning hubs in the front and rear of the vehicle. The control PCB for the hubs can also be seen, having likewise been mounted to an aluminium extrusion. The stepper motors of the turning hubs were powered using an 12V onboard power outlet of the Husky A200 while the control PCB received power and the control signals for the motors from a Raspberry Pi on the vehicle via a USB cable. The Raspberry Pi was connected to the sensor network via WiFi and was powered using a USB C cable connected to one of the USB ports of the onboard computer. A custom ROS node running on the Raspberry Pi received the tracker measurements in real time as ROS messages via the sensor network and used them to calculate the required azimuth command value for both turning hubs. This data was then relayed as ROS message to the Arduino Nano of the control PCB of the turning hub assembly using the rosserial package (see section 3.2.1).

A separate computer in form of the Raspberry Pi was employed for calculating and relaying the azimuth control commands for the turning hubs in order to avoid any potential interference with the ROS nodes running on the onboard computer for driving the Husky A200. While the roscore, which is a software acting as a central hub for interactions of nodes and data transfer between them, for the laser trackers, their time synchronization Raspberry Pi and the Raspberry Pi controlling the turning hubs was run on an offsite server, a completely separate roscore running on the Husky's onboard computer was used on the Husky A200 ROS nodes. This ensured that the data traffic and other activity of the measurement setup did not affect the Husky A200 during the measurements.

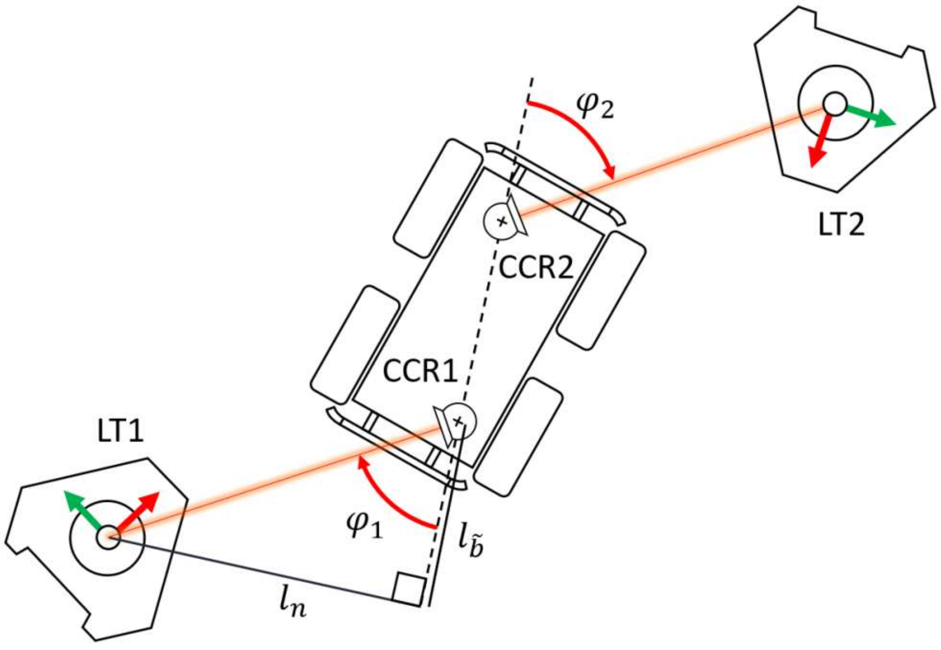

Figure 4.6: Illustration of the turning hub azimuth calculation

As the used CCR can accept a wide range of incidence angles of ±20° (see section 3.1.3) there were no strong requirements for the accuracy of the calculated target azimuth and how quickly the commands were executed by the turning hubs. Thus the assumption of no roll or pitch movements of the vehicle during driving could be made without issues.

The set point for motorized turning hubs was calculated using the laser tracker measurements arriving over the sensor network at the Raspberry Pi in real-time. The measured points are $\mathbf{x}^{\mathrm{LT1}}_{\mathrm{LT1,\ CCR1}}$ and $\mathbf{x}^{\mathrm{LT2}}_{\mathrm{LT2,\ CCR2}}$ which denote the position vector from the origin of the measurement frame to the respective CCR reference point expressed using the axes of the sensor frame of one of the laser trackers. The subscripts of 1 and 2 are used in this section for the laser trackers and their respective CCR as they can be used interchangeably. Also for this reason the calculations are shown only for one prism with the second calculations being completely analogous.

For the calculation of the turning hub azimuths the position of the reference point of one laser tracker needs to be known in the sensor frame of the other laser tracker, thus requiring a transformation between the two sensor frames. The estimation of the parameters for this transformation and the transformation of the points is described in section 5.2.1. Therefore the transformed points, $\mathbf{x}^{\mathrm{LT2}}_{\mathrm{LT1,\ CCR1}}$ and $\mathbf{x}^{\mathrm{LT1}}_{\mathrm{LT2,\ CCR2}}$, are seen as given for the purpose of the explanations in this section.

A geometric representation of the relevant geometry for calculating the azimuths is given in figure 4.6. The two reference points form a baseline vector, here given with respect to the sensor frame of laser tracker 1:

$$\mathbf{b}^{\mathrm{LT1}}_{\mathrm{CCR1,\ CCR2}} = \mathbf{x}^{\mathrm{LT1}}_{\mathrm{LT1,\ CCR2}} - \mathbf{x}^{\mathrm{LT1}}_{\mathrm{LT1,\ CCR1}} \tag{4.1}$$

which was projected into the xy-plane for the calculations resulting in

$$\tilde{\mathbf{b}}^{\mathrm{LT1}}_{\mathrm{CCR1,\ CCR2}} = \begin{bmatrix} x_b \\ y_b \\ 0 \end{bmatrix} \tag{4.2}$$

The vector perpendicular to this vector and the z-axis is calculated using

$$\mathbf{n}^{\mathrm{LT1}}_{\mathrm{CCR1,\ CCR2}} = -\tilde{\mathbf{b}}^{\mathrm{LT1}}_{\mathrm{CCR1,\ CCR2}} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.3}$$

Subsequently the vector from the prism to its measuring laser tracker sensor frame origin is projected onto the baseline vector and its normal vector resulting in its components in those two directions using

$$l_{\tilde{b}} = -\mathbf{x}^{\mathrm{LT1}}_{\mathrm{LT1,\ CCR1}} \cdot \frac{\tilde{\mathbf{b}}^{\mathrm{LT1}}_{\mathrm{CCR1,\ CCR2}}}{\|\tilde{\mathbf{b}}^{\mathrm{LT1}}_{\mathrm{CCR1,\ CCR2}}\|} \tag{4.4}$$

and

$$l_n = \mathbf{x}^{\mathrm{LT1}}_{\mathrm{LT1,\ CCR1}} \cdot \frac{\mathbf{n}^{\mathrm{LT1}}_{\mathrm{CCR1,\ CCR2}}}{\|\mathbf{n}^{\mathrm{LT1}}_{\mathrm{CCR1,\ CCR2}}\|}. \tag{4.5}$$

With those values the azimuth for the motorized turning hub could be calculated using

$$\varphi = \mathrm{atan2}(l_n, l_{\tilde{b}}) \tag{4.6}$$

whereby atan2 is a function that calculates the arc-tangent while taking the quadrant in which the points lies into account and it is defined as

$$\mathrm{atan2}(y, x) = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \pi/2 - \arctan(\frac{y}{x}) & y > 0 \\ -\pi/2 - \arctan(\frac{y}{x}) & y < 0 \\ \arctan(\frac{y}{x}) + \pi & x < 0 \\ \text{undefined} & x = 0 \text{ and } y = 0 \end{cases} \tag{4.7}$$

This calculation was conducted for both laser tracker and the obtained azimuth angles were used as set points for the turning hubs. For initialization the CCRs were manually turned in the magnetic hubs to face in the direction of their respective laser tracker. The system calculated the azimuths and set the first calculated values as the current rotation angle of the turning hubs.

5. Methodology

In this chapter the employed methodology for the different tasks is described. There are three main areas that need to be covered. First, the method of determining the leverarms of the CCR on the Husky A200 with regard to its body frame and by extension the necessary realization of the body frame is described in section 5.1. Subsequently the preprocessing of the driving experiments measurement data for obtaining the input data for the system identification is presented in section 5.2. Lastly the system identification itself using the SINDY algorithm is described along with the methodology of hyperparameter tuning and model evaluation in section 5.3.

5.1 Body frame realization & CCR leverarm determination

Determining the pose of the Husky A200 equates to estimating the transform parameters between the body frame of the robot and the chosen local reference frame. For this, the coordinates of two points need to be known in both reference frames. While their coordinates in the local reference frame are ever-changing and are constantly measured by the laser trackers during driving, the coordinates with regard to the body frame are assumed to be constant and are determined beforehand. The position vector of the reflectors with regard to the body frame are henceforth also referred to as leverarms.

In order to measure the position of the reference points with regard to the body frame if first had to be realized. The realization of the Husky A200 body frame, as defined in section 3.1.1, was accomplished by measuring the coordinates of the heads of the hex key screws used for mounting the wheels to their wheel hubs. As the four screw holes of every wheel are arranged circularly around the axle in the wheel hub, the measured points theoretically lie on a circle with its center on the center line of the axle. With the measurements of the screw heads, the two end points of the center-lines of both axles are thus defined and due to the definition of the chosen body frame, it can be fully constructed using those points.

The measurements, described in section 6.2, were conducted with the software Spatial Analyser (SA), which is described in section 3.2.3. Its functions were also used to fit a circle to the four measured points per wheel hub. Subsequently a plane was fitted to the four resulting center points of the circles. The body frame was then created in SA using its frame wizard function with the fitted plane defining the xy-plane, the center of the four center points defining the origin position with that plane and the connection of the center points of the left front and rear tire defining the direction of the x axis.

The measured coordinates of the reference point could then be transformed into the newly created reference frame in SA and thus represented the required lever arms.

5.2 Preprocessing

5.2.1 Transformation into common coordinate system

The instrument reference frame of the AT960 laser tracker was used as local reference frame for the calculations of this thesis. The measurements of the LTD800 therefore had to be transformed into the measurement frame of the AT960.

The estimation of the needed transform parameters was accomplished by measuring identical points with both laser trackers (see sections 4 and 6.1) and subsequently using the USMN (Unified Spatial Metrology Network) function of the Spatial Analyser software, described in section 3.2.3. This function yielded the transformation parameters between the instrument frames.

With the notation introduced in section 2.5 the transformation of the measured points of the LTD800 laser tracker to its respective CCR target into the measurement frame of the AT960 laser tracker may be expressed as

$$\mathbf{x}^{\text{AT960}}_{\text{AT960, CCR}} = \mathbf{t}^{\text{AT960}}_{\text{AT960, LTD800}} + \mathbf{C}^{\text{AT960}}_{\text{LTD800}}\ \mathbf{x}^{\text{LTD800}}_{\text{LTD800, CCR}} \tag{5.1}$$

with $\mathbf{t}^{\text{AT960}}_{\text{AT960, LTD800}}$ being the translation vector of the transformation. The transformation matrix $\mathbf{C}^{\text{AT960}}_{\text{LTD800}}$ is a 3D rotation matrix that is defined in equations 5.2 to 5.5.

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{5.2}$$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{5.3}$$

$$\mathbf{R}_z(\gamma) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

$$\mathbf{C}^{\text{AT960}}_{\text{LTD800}} = \mathbf{R}_z(\gamma)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha) \tag{5.5}$$

5.2.2 Time synchronization

Due to both missed and erroneous trigger events of the laser tracker measurements caused by interference on the trigger cable (see section 4.2), the measurements could not be matched with the time stamps provided by the GNSS synchronized Raspberry Pi by their index.

Therefore a synchronisation of the time systems of the Raspberry Pi and the two laser trackers was necessary. Furthermore, the triggered laser tracker measurements have a timing accuracy of $5\mu s$ with respect to the incoming trigger signal, see section 3.1.2, and a synchronisation of the time systems may improve the timing accuracy regardless of the presence of interference caused issues. The trigger events and the measurements recorded by the two laser trackers have time stamps in their respective time system and are denoted $t_{r,i}^{(\mathrm{RPi})}$, $t_{r,i}^{(\mathrm{AT960})}$ and $t_{r,i}^{(\mathrm{LTD800})}$, whereby $t_{r,i}^{(\mathrm{LT}\ 1/2)}$ is used to refer to the time stamps of either one of the two trackers. The indices r and i refer to the i-th event during the r-th run. It is important to note that the indices of the events, i, do not correspond with each other over the different time systems due to the aforementioned reasons.

The synchronisation was performed in three steps, whereby they were performed for each experimental run, r, as well as for both trackers individually. First a discrete cross correlation was used in order to coarsely align the individual laser tracker time system with that of the Raspberry Pi. Then a linear clock drift model was fitted which mapped the corrected laser tracker time stamps onto the raspberry time system. Lastly a correction step, that was necessitated by the measurement setup error described in section 4.2.2, was applied to the time stamps of the AT960 measurements.

As is described in section 4.2 a trigger event of the 50 Hz trigger signal was skipped every three seconds. This resulted in distinctive gaps in the series of time stamps of all three systems that could be utilized for the time synchronisation. A discrete signal was created based on those gaps by mapping the time of a timestamp $t_{r,i}$ to the time difference to the subsequent time stamp $\Delta t_{r,i}$:

$$f_r : t_{r,i} \mapsto \Delta t_{r,i} = t_{r,i+1} - t_{r,i} \tag{5.6}$$

This signal was created for the time stamps of the Raspberry Pi and both laser trackers and are denoted $f_r^{(\mathrm{RPi})}$, $f_r^{(\mathrm{AT960})}$ and $f_r^{(\mathrm{LTD800})}$ respectively whereby $f_r^{(\mathrm{LT}\ 1/2)}$ is used to refer to either one of the two laser trackers signals. The resulting signals are well suited for discrete cross correlation due to the sharp changes in the time differences of subsequent time stamps. They also enable easy and robust detection of the gap locations by thresholding with a value of 35 ms, resulting in the time stamps of the gap locations in the different time systems $t_{r,\mathrm{gap}_j}^{(\mathrm{RPi})}$ and $t_{r,\mathrm{gap}_j}^{(\mathrm{LT}\ 1/2)}$ respectively, whereby the latter one is again used to refer to either one of the two laser trackers. An illustrative example of the signal based on actual measurement data along with the detected gap locations is depicted in figure 5.1.

A discrete cross correlation between the signal of the Raspberry Pi timestamps, $f_r^{(RPi)}$, and of the laser trackers, $f_r^{(\mathrm{LT}\ 1/2)}$, was performed. As the signals f of the different components are not strictly equidistant, they were linearly interpolated at 1000 Hz using the `interp` numpy function, resulting in the signals $\bar{f}_r^{(RPi)}$ and $\bar{f}_r^{(\mathrm{LT}\ 1/2)}$. The interpolation also increased the time resolution of the cross correlation result. The equation for the discrete cross correlation is given in equation 5.7 and the calculations were performed using the scipy Python package. The location of the maximum of the cross correlation function gives the shift between the time system of the laser tracker and the Raspberry Pi. The shift is denoted $\tau_r^{\mathrm{LT}\ 1/2}$ and was

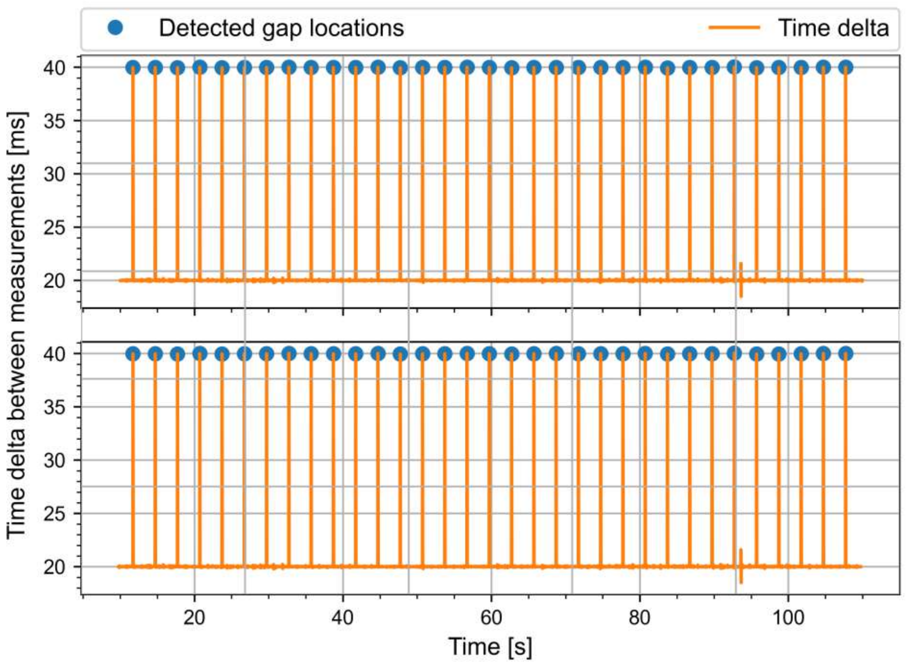

Figure 5.1: The created signal used for time synchronisation and the detected gap locations of the LTD800 (top) and the Raspberry Pi (bottom)

calculated using equation 5.8. An exemplary results of the cross correlation for one run is depicted in figure 5.2.

$$\left(\bar{f}_{ts}^{(\mathrm{LT\ 1/2})} * \bar{f}_{ts}^{(RPi)}\right)[n] = \sum_{m=1}^{N} \bar{f}_{ts}^{(\mathrm{LT\ 1/2})}[m]\bar{f}_{ts}^{(RPi)}[m+n] \tag{5.7}$$

$$\tau_r^{(\mathrm{LT\ 1/2})} = \arg\max_n \left(\bar{f}_{ts}^{(\mathrm{LT\ 1/2})} * \bar{f}_{ts}^{(RPi)}\right)[n] \cdot \frac{1}{1000Hz} \tag{5.8}$$

The coarse alignment of the different time systems using the cross correlation is performed in order to aid the matching of corresponding gap locations in the different time systems. It only results in an coarse alignment because the internal clocks of the laser trackers exhibit a drift as is mentioned in section 3.1.2 that had not yet been corrected at this stage. The timestamps of the identified gap locations in the time systems of the two laser trackers were shifted using

$$t_{r,\mathrm{gap}_j}^{*(\mathrm{LT\ 1/2})} = t_{r,\mathrm{gap}_j}^{(\mathrm{LT\ 1/2})} - \tau_r^{(\mathrm{LT\ 1/2})}. \tag{5.9}$$

After this correction correspondences between the timestamps of identified gap locations in the time systems of the Raspberry Pi and laser trackers were identified. This was done by searching for the closest laser tracker gap timestamp to each Raspberry Pi gap timestamp. In order to eliminate incorrect matches every correspondence where the timestamps of the two time systems were further than 4 ms apart were discarded. The resulting lists of corresponding time stamps of gap locations in both the Raspberry Pi and laser trackers time systems were used to estimate the parameters of a linear clock drift model:

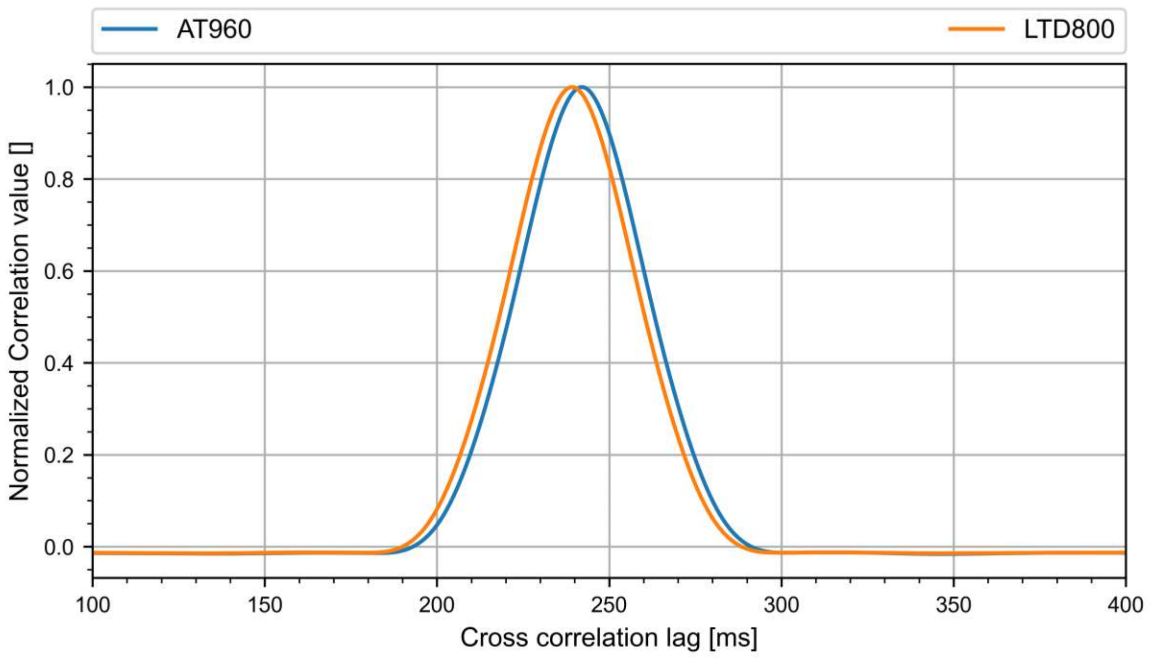

Figure 5.2: Example of the cross correlation output for the coarse alignment of the laser tracker time systems with that of the Raspberry Pi for one driving experiment

$$t^{(RPi)} = k \cdot t^{(\text{LT 1/2})} + d \tag{5.10}$$

The estimation of the parameters was conducted using a least squares estimation and was accomplished using the `polyfit()` function of the numpy Python package. The used model equation is

$$t^{(RPi)}_{r,\text{ gap}_i} = k^{(\text{LT 1/2})}_r \cdot t^{(\text{LT 1/2})}_{r,\text{ gap}_i} + d^{(\text{LT 1/2})}_r + \epsilon_{r,\text{ gap}_i} \tag{5.11}$$

wherein $k^{(\text{LT 1/2})}_r$ and $d^{(\text{LT 1/2})}_r$ denote the clock drift and clock offset of the first or second laser tracker during the r-th run. $t^{(RPi)}_{r,\text{gap}_i}$ and $t^{*(\text{LT 1/2})}_{r,\text{ gap}_i}$ refer to the i-th corresponding gap time stamp of the r-th run in the time system of either the Raspberry Pi or either one of the laser trackers. $\epsilon^{(\text{LT 1/2})}_{r,\text{ gap}_i}$ denotes the remaining error at the i-th gap location of the r-th experimental run for the first or second laser tracker. Figure 5.3 shows an example of the linear clock model fit for both trackers for one of the driving experiment runs.

After the estimation of the clock model parameters, the time stamps of the measurements of both laser trackers were corrected using the linear model, resulting in the time stamps of the measurement within the time system of the Raspberry Pi $t^{(\text{RPi})}_{\text{LT 1/2}}$:

$$t^{(\text{RPi})}_{\text{LT 1/2}} = k^{(\text{LT 1/2})}_r \cdot \left(t^{(\text{LT 1/2})} - \tau^{(\text{LT 1/2})}_r\right) + d^{(\text{LT 1/2})}_r \tag{5.12}$$

The synchronisation of the LTD800 tracker was completed at this point. For the AT960 an additional correction was necessary.

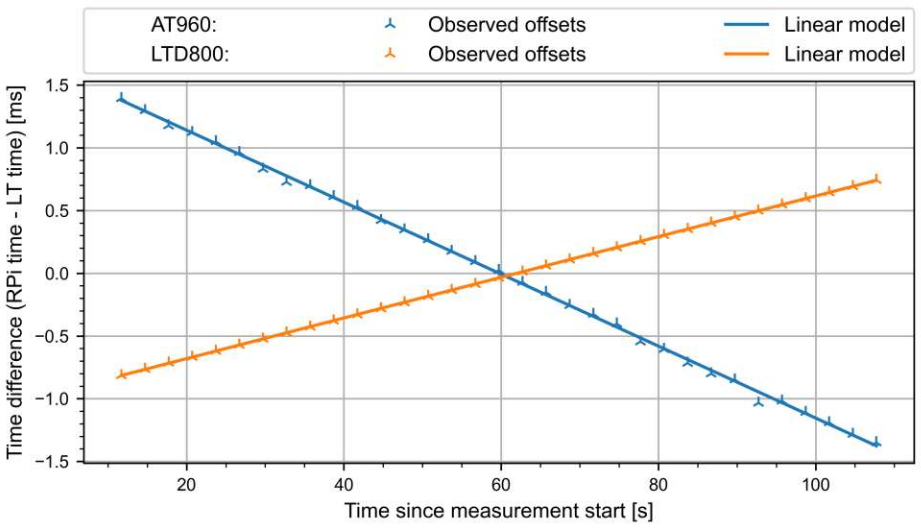

Figure 5.3: Example of the fitted linear clock models for the two internal laser tracker clocks for one driving experiment

Correction of AT960 measurement setup error

Due to the measurement setup error described in section 4.2.2 the measurements of the AT960 and their time stamps did not correspond to the trigger timestamps of the Raspberry Pi and by extensions to the measurements and timestamps of the LTD800. The AT960 time system still had an unknown offset with regard to the Raspberry Pi time system of about 1 ms after the corrections described above, which corresponds to the average duty cycle length of the trigger signal. Said offset could not be estimated using solely the timestamps of the different system or the signals f created from them described in equation 5.6. The variations in the offset caused by the varying duty cycle of the trigger signal, as explained in section 4.2.2, are accounted for by the measurement timestamps in the time system of the AT960. Figure 5.4 depicts the remaining differences between the laser tracker time stamps transformed into the time system of the Raspberry Pi. While the magnitude of the discrepancies of the LTD800 timestamps are as expected (see section 3.1.2), the AT960 time stamps exhibit large deviations caused be the measurement setup error (see section 4.2.2) and the varying duty cycle length of the trigger cable (see section 4.2.1). However, this means that once the time system of the AT960 is correctly mapped onto that of the Raspberry Pi, the variations in time offsets to the trigger times is also solved.

The two reference points on the Husky A200 which are realized using CCRs and measured with the two laser trackers form a baseline. The baseline length is constant besides flexing of the physical structure the reference points are mounted upon. If the measurements of the laser trackers are synchronised correctly and interpolated at common time stamps, the distance between the measured points should only vary in accordance with the measurement uncertainty. During preliminary calculations variations of the measured baseline length over time were discovered. A strong correlation of this variation with the forwards velocity of the vehicle with a Pearson correlation coefficient of about 0.9 led to the discovery of the time synchronisation issue. However, the variations of the observed baseline length could in turn also be used to estimate the remaining clock offset of the AT960 time system. This was

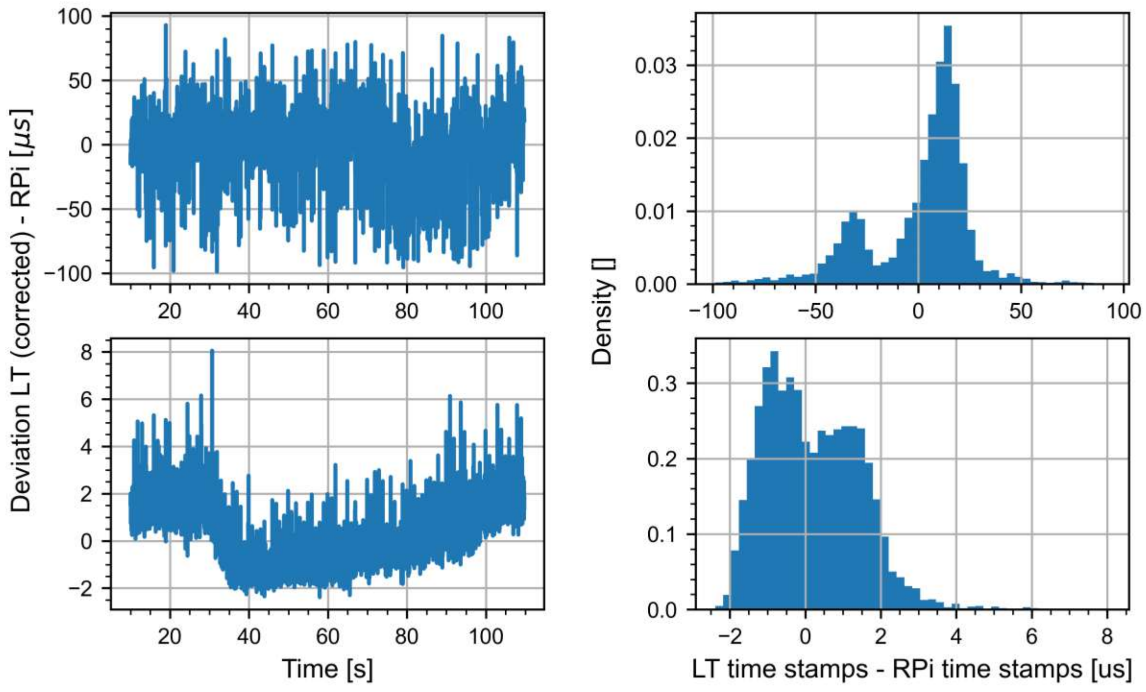

Figure 5.4: The remaining time difference between the measurement time stamps of the laser trackers transformed into the Raspberry Pi time system and the Raspberry Pi trigger time stamps for the AT960 (top) and LTD800 (bottom)

accomplished by minimizing the observed base line length variations for the different runs individually by shifting the AT960 measurements in time.

The laser tracker point measurement for any point in time can be obtained by interpolating the measured coordinates with their time stamps in the Raspberry Pi time system $t^{(\mathrm{RPi})}_{\mathrm{LT}\ 1/2}$ at point t, resulting in the point coordinate vectors $\mathbf{x}_{AT960}(t)$ and $\mathbf{x}_{LTD800}(t)$. The observed baseline length $l(t)$ can hence be calculated using

$$l(t) = \sqrt{\left(\mathbf{x}_{LTD800}(t) - \mathbf{x}_{AT960}(t)\right)^T\left(\mathbf{x}_{LTD800}(t) - \mathbf{x}_{AT960}(t)\right)}. \tag{5.13}$$

Introducing an offset for the AT960 measurements denoted Δt the observed baseline length becomes

$$l(t, \Delta t) = \sqrt{\left(\mathbf{x}_{LTD800}(t) - \mathbf{x}_{AT960}(t+\Delta t)\right)^T\left(\mathbf{x}_{LTD800}(t) - \mathbf{x}_{AT960}(t+\Delta t)\right)}. \tag{5.14}$$

For the calculation the timestamps of the Raspberry Pi, $t^{(\mathrm{RPi})}_{r,i}$, were used as the values for t which resulted in the baseline length of the i-th timestamp during the r-th run, see equation 5.15. As the additional offset of the AT960 is calculated for every run individually, the subscript r is also introduced to the offset resulting in Δt_r.

$$l(t_{r,i}^{(\mathrm{RPi})}, \Delta t_r) = \sqrt{\left(\mathbf{x}_{LTD800}(t_{r,i}^{(\mathrm{RPi})}) - \mathbf{x}_{AT960}(t_{r,i}^{(\mathrm{RPi})} + \Delta t_r)\right)^T \left(\mathbf{x}_{LTD800}(t_{r,i}^{(\mathrm{RPi})}) - \mathbf{x}_{AT960}(t_{r,i}^{(\mathrm{RPi})} + \Delta t_r)\right)} \tag{5.15}$$

The standard deviation was used as a metric for quantifying the variability of the observed baseline length during the r-th run:

$$s_r(\Delta t_r) = \frac{1}{N_r - 1}\sqrt{\sum_{i=1}^{N_r}(l(t_{r,i}^{(\mathrm{RPi})}, \Delta t_r) - \bar{l}(t_{r,i}^{(\mathrm{RPi})}, \Delta t_r)} \tag{5.16}$$

whereby $\bar{l}(t_{r,i}^{(\mathrm{RPi})}, \Delta t_r)$ denotes the average observed baseline length calculated from the used samples:

$$\bar{l}(t_{r,i}^{(\mathrm{RPi})}, \Delta t_r) = \frac{1}{N_r}\sum_{i}^{N_r} l(t_{r,i}^{(\mathrm{RPi})}, \Delta t_r) \tag{5.17}$$

The additional clock offset of the AT960 tracker could then be calculated by using s_r as a minimization criteria as follows

$$\Delta t_r = \arg\min_{\Delta t_r} s_r(\Delta t_r). \tag{5.18}$$

In order to accomplish this the function s_r was sampled at multiple values for Δt_r. It was found that the function could be approximated very well with a quadratic polynomial around the minimum. This approximation is defined in equation 5.19. The estimate of the time offset Δt_r for each run could then be calculated analytically as is shown in equation 5.20. The coefficient for each run, a_r, b_r and c_r were estimated using a least squares adjustment based on ten samples around the minimum using the numpy function `polyfit`. Figure 5.6 shows an exemplary plot of the approximation of one of the experimental runs.

$$s_r(\Delta t) \approx h_r(\Delta t) = a_r \cdot \Delta t^2 + b_r \cdot \Delta t + c_r \tag{5.19}$$

$$\Delta t_r \approx \arg\min_{\Delta t} h_r(\Delta t) = -\frac{b_r}{2a_r} \tag{5.20}$$

The covariance information about the estimated parameters obtained from the least squares adjustment in an error propagation in order to calculate the variance of the calculated minimum:

$$s_{\Delta t_r} = \mathbf{F}\mathbf{C}_{xx}\mathbf{F}^T \tag{5.21}$$

with $\mathbf{C}_{xx}$ being the estimated covariance matrix of the parameters and $\mathbf{F}$ being the Jacobian of the calculation in equation 5.20 given as

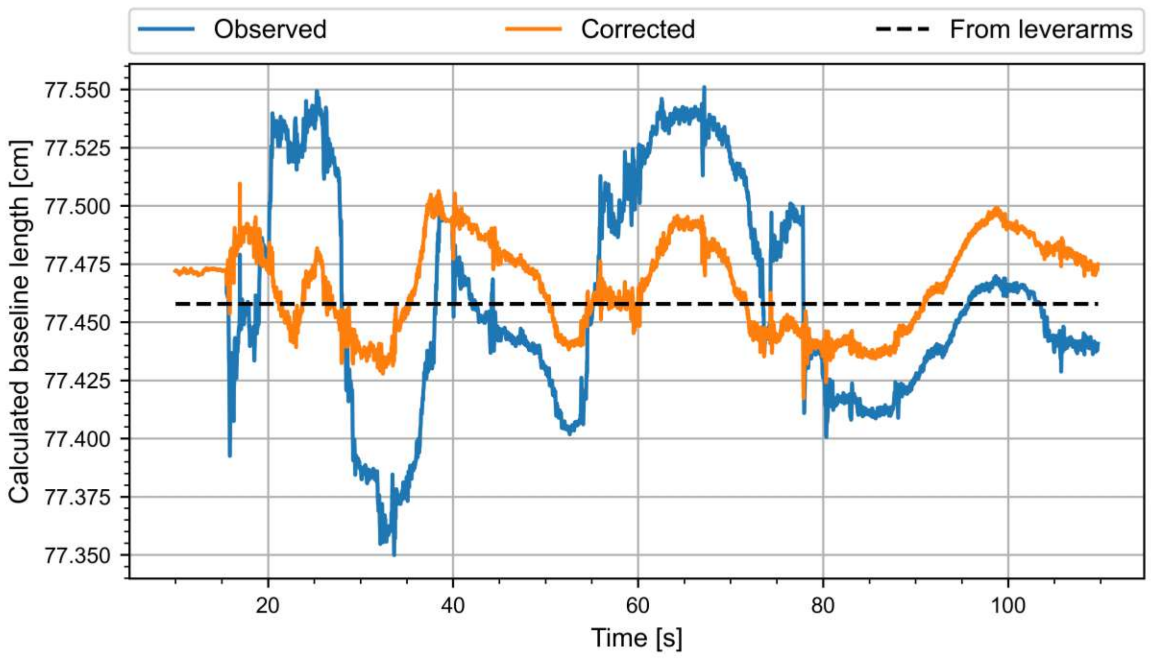

Figure 5.5: Example of the baseline length over time before and after the correction of the AT960 offset for one driving experiment

$$\mathbf{F} = \begin{bmatrix} \frac{b}{2a^2} & -\frac{1}{2a} & 0 \end{bmatrix}. \tag{5.22}$$

Figure 5.5 depicts an example of the baseline length variations prior and after the correction of the AT960 offset.

Application of synchronisation

Lastly, the measurements of both the LTD800 and AT960 could be brought into the time system of the Raspberry Pi.

$$t^{(\mathrm{RPi})}_{\mathrm{LTD800},r,i} = k^{(\mathrm{LTD800})}_r \cdot (t^{(\mathrm{LTD800})}_{r,i} - \tau^{(\mathrm{LTD800})}_r) + d^{(\mathrm{LTD800})}_r \tag{5.23}$$

$$t^{(\mathrm{RPi})}_{\mathrm{AT960},r,i} = k^{(\mathrm{AT960})}_r \cdot (t^{(\mathrm{AT960})}_{r,i} - \tau^{(\mathrm{AT960})}_r) + d^{(\mathrm{AT960})}_r + \Delta t_r \tag{5.24}$$

In order to get simultaneous point coordinates of the two reference points on the Husky A200 the time synchronised laser tracker measurements were interpolated at the time stamps of the Raspberry Pi. Due to the high measurement frequency and the relatively slow speed of the vehicle a linear interpolation of the coordinate time series was sufficient. The resulting point coordinates are henceforth referred to as $\mathbf{x}^l_{l,\mathrm{AT960}}(t_{r,i})$ and $\mathbf{x}^l_{l,\mathrm{LTD800}}(t_{r,i})$.

5.2.3 Turning hub error model

During the estimation of the additional time offset of the AT960 time system the variations in the base line length over time were used as a minimization criteria. However, even after

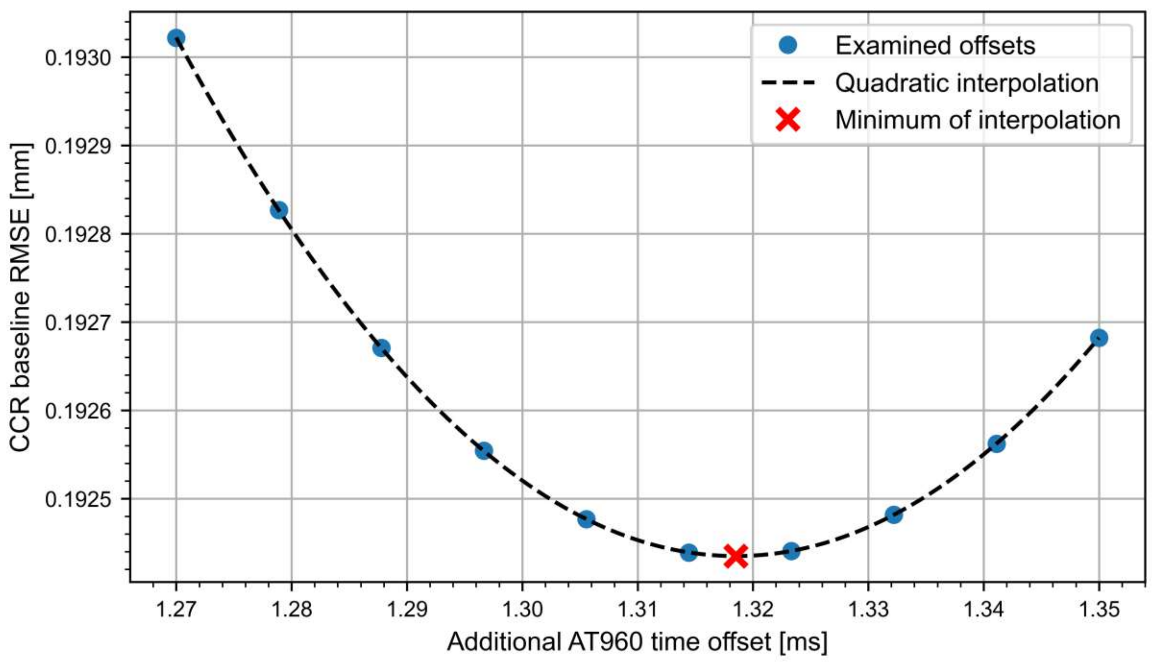

Figure 5.6: Example of the cubic spline fit of the base line length variability as a function of time offset for one driving experiment

this correction step there remained significant variations of the observed base line lengthm as can be seen in figure 5.5. One potential source of such variations is an eccentricity of the motorized turning hubs, described in section 3.1.7, on which the CCRs are mounted. In order to ascribe a source to the cause of the remaining variations and preclude potentially remaining large time synchronisation errors, a model for the eccentricity caused baseline length variation was created and its parameters fitted to the data:

$$l(t_i) + \epsilon_i = c + r_1 \cos\left(\varphi_1(t_i) + \Phi_1\right) + r_2 \cos\left(\varphi_2(t_i) + \Phi_2\right) \tag{5.25}$$

The model assumes a circular motion of both measured CCR reference points with the turning hub azimuths acting as the phase angles. The azimuths, φ_1 and φ_2, were calculated as already described in section 3.1.7. The baseline length variations were modeled by an constant length c and two variations caused by the two circular motions with radii r_1 and r_2. Phase offsets Φ_1 and Φ_2 had to be introduced into the model as the initial position of the turning hub is not consistent between the different runs as the CCRs were manually adjusted to face the laser trackers before every run. Additionally they were occasionally adjusted during the experimental runs, limiting the achievable fit quality. These are also the reasons why the estimation of the parameters were conducted for every run individually and subsequently compared. However, while the phase offset parameters had to be estimated for every run individually the radii and the constant term of the model could be estimated for all runs simultaneously in a future improvement. Equation 5.25 gives the used model. The estimation of the parameters was conducted with a least squares estimation using the `scipy.optimize.curve_fit` function. Figure 5.7 shows an example of the remaining baseline length variations after the time synchronisation and the modeled variations for one experi-

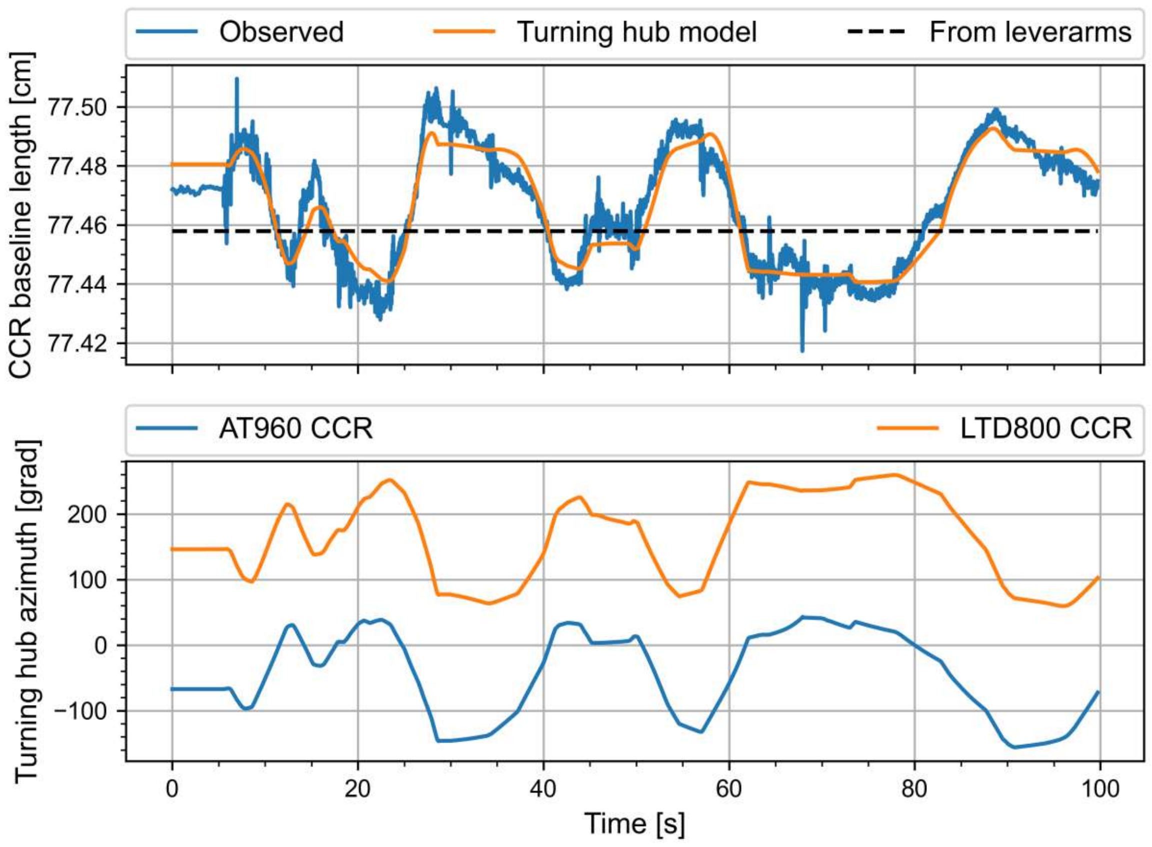

Figure 5.7: Example of the turning hub error model for one driving experiment with the observed baseline length variations along with the resulting model (top) and the turning hub azimuths used as input for the model (below)

mental run. Alongside the calculated azimuths used as independent variables in the model are also shown. The azimuths are not treated as stochastic quantities in the estimation. However, they exhibit systematic errors because the calculated azimuth is not identical to the true turning angle of the turning hub due to delays in the transmission and processing of the data, the reaction time of the PD controller and the deadband implemented in the controller of the turning hubs.

The model generally showed good agreement with the data so that other sources of large errors could be precluded. However, the model was not used to adjust the measured coordinates of the CCR reference points. The main reason for this is that the measurements for the determination of the leverarms of the CCRs were conducted with an unknown phase angle. The information obtained from the correction model of the baseline length is not sufficient to determine at what angle the turning hubs were during the measurement and how to adjust the measurements of both CCRs in all three dimensions. Therefore it was elected to not change the measurements as one systematic effect would be replaced with different, potentially larger, systematic errors. Nevertheless, the model could be used to ascribe an error source to the remaining observed base line length variations and to quantify the limits of the turning hubs.

5.2.4 Pose calculation

Based on the time synchronized laser tracker measurements the poses of the Husky A200 could be calculated which equates to estimating the transformation parameters between the vehicle's body frame and the local reference frame which was chosen to be the measurement frame of the AT960 laser tracker. The estimation had to be conducted for each data sample point individually. For clarity the calculations in the following section are presented for a single pose, omitting any time or pose indices.

2D Pose calculation

In a first step the roll and pitch movements of the Husky A200 during the driving experiments were assumed to be neglectable and the problem was reduced to two dimensions in order to obtain adequate initial parameter estimates for a subsequent iterative linearized least squares adjustment. This was accomplished by projecting the coordinates of the measured CCR points into the xy plane of both the local reference frame and the vehicle body frame. The transformation equation for either one of the two reference points thus becomes

$$\begin{bmatrix} x^l_{l,\text{ccr } 1/2} \\ y^l_{l,\text{ccr } 1/2} \end{bmatrix} = \begin{bmatrix} t^l_x \\ t^l_y \end{bmatrix} + \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x^b_{b,\text{ccr } 1/2} \\ y^b_{b,\text{ccr } 1/2} \end{bmatrix} \tag{5.26}$$

with t^l_x and t^l_y being the components of the 2D translation vector from the origin of the local reference frame to the origin of the vehicle's body frame and α denoting the heading of the vehicle in the simplified 2D case. The variables x and y are the coordinates of either one of the measured reference points on the Husky A200 in either the local reference frame l or the body frame b, adopting the notation introduced in section 2.5.

This transformation is non-linear in the parameters. However, by utilizing overparametrization by introducing the parameters $a = \cos\alpha$ and $b = \sin\alpha$ the equation system becomes linear:

$$\begin{bmatrix} x^l_{l,\text{ccr } 1/2} \\ y^l_{l,\text{ccr } 1/2} \end{bmatrix} = \begin{bmatrix} t^l_x \\ t^l_y \end{bmatrix} + \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x^b_{b,\text{ccr } 1/2} \\ y^b_{b,\text{ccr } 1/2} \end{bmatrix} \tag{5.27}$$

Combining the transformation equations of the two reference points and rearranging the equation system in order to combine the four parameters of the overparameterized transformation into one vector yields

$$\begin{bmatrix} 1 & 0 & x^b_{ccr_1} & -y^b_{ccr_1} \\ 0 & 1 & y^b_{ccr_1} & x^b_{ccr_1} \\ 1 & 0 & x^b_{ccr_2} & -y^b_{ccr_2} \\ 0 & 1 & y^b_{ccr_2} & x^b_{ccr_2} \end{bmatrix} \begin{bmatrix} t^l_x \\ t^l_y \\ a \\ b \end{bmatrix} = \begin{bmatrix} x^l_{ccr_1} \\ y^l_{ccr_1} \\ x^l_{ccr_2} \\ y^l_{ccr_2} \end{bmatrix} \tag{5.28}$$

which is a linear equation system of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{5.29}$$

which can be solved by inverting the matrix **A**

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}. \tag{5.30}$$

The parameter α may be obtained approximately using

$$\alpha \approx \text{atan2}(b, a) \tag{5.31}$$

whereby atan2 is a function that calculates the arc-tangent while taking the quadrant in which the points lies into account and it is defined as

$$\text{atan2}(y, x) = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \pi/2 - \arctan(\frac{y}{x}) & y > 0 \\ -\pi/2 - \arctan(\frac{y}{x}) & y < 0 \\ \arctan(\frac{y}{x}) + \pi & x < 0 \\ \text{undefined} & x = 0 \text{ and } y = 0 \end{cases} \tag{5.32}$$

3D Pose least squares adjustment

After the 2D poses were calculated a subsequent calculation of 3D poses was conducted. A full 3D coordinate transformation which equates to a 3D pose has 6 parameters, three for translation and three for rotation. Despite the two measured reference points on the Husky A200 representing 6 observations the estimation of the 6 parameters is underdetermined. This is because the observed coordinates are invariant to a rotation around the axis that is represented by the baseline. One of the six observations is redundant because the length of the baseline is known from the leverarms in the body frame of the vehicle. Therefore a full 3D pose could not be calculated from the observations. Instead a 3D pose with only two rotations, corresponding to the yaw or heading and the pitch of the vehicle was used. Those two rotations are around the z and y axis of the vehicle body frame respectively, see section 3.1.1. The resulting transformation equation is given in equations 5.33 to 5.35.

$$\mathbf{x}^l_{ccr_i} = \mathbf{t}^l + \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{x}^b_{ccr_i} \tag{5.33}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{5.34}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.35}$$

The rational behind the choice of this parametrization is that the vehicle does experience some noticeable pitch movements, especially during acceleration and braking. If not accounted for by a own parameter and rotation those changes would alter the estimated position and could adversely affect the system identification of the driving dynamics. Furthermore, the length of the projection of the xy plane of the local reference frame is altered by a pitch movement. The estimated uncertainties of the parameters resulting from a least squares adjustment would therefore not be representative as the corrections for the observations would have to increase in order to account for the inadequate functional model used.

The parameters of the pose were calculated using a least squares adjustment, which is described in section 2.4. The functional model $\mathbf{\Phi}$ consists of the transformation equation given in equation 5.33 for both reference points on the Husky A200. The functional model for one point is thus

$$\mathbf{\Phi}_{\text{ccr }1/2} = \mathbf{t}^l + \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{x}^b_{ccr_i} - \mathbf{x}^l_{\text{ccr }1/2} \tag{5.36}$$

The functional model is non-linear and is thus linearized. While calculating the derivatives of the functional model are trivial they are lengthy which makes working with them cumbersome and error-prone. Therefore they were calculated analytically using the sympy Python package (see section 3.2.4). For this the 2D pose information obtained in the previous section was used as approximate values of the parameters. For the pitch angle θ zero was used as an approximate value. For the z component of the translation vector likewise no pitch was assumed an the z coordinates of the leverarms in the body frame were subtracted from the measured CCR points in the local reference frame and subsequently averaged. The vector of approximate values for the parameters to be estimated then becomes

$$\mathbf{X}_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ \theta_0 \\ \psi_0 \end{bmatrix} = \begin{bmatrix} x_{2D} \\ y_{2D} \\ \frac{1}{2}(z^l_{\text{ccr}1} - z^b_{\text{ccr}1} + z^l_{\text{ccr}2} - z^b_{\text{ccr}2}) \\ 0 \\ \psi_{2D} \end{bmatrix} \tag{5.37}$$

with the suffix $2D$ denoting the values obtained from the 2D pose estimation described in the previous section. During the analysis of the data it was found that these approximate values were sufficiently accurate for the iterative linearized least squares adjustment to converge quickly, typically within one or two iterations. For the adjustment all measurements were assumed to have the same uncertainties as the uncertainty estimates were not recorded from the laser trackers during the experiments and the systematic effects of the turning hub eccentricity (see section 5.2.3) has a much larger effect than the measurement uncertainties of the trackers.

5.3 System identification

5.3.1 The SINDY Algorithm

The SINDY algorithm is a dynamical system identification algorithm that both identifies the active terms in the differential equations from a potentially very large library of candidate functions and estimates their coefficients. It therefore represents a quasi non-parametric estimation approach, as the maximum number of active terms is predetermined and the form of the dynamical system model is determined by setting the coefficients of inactive terms to zero. SINDY stands for Sparse Identification of Non-linear DYnamics and the algorithm was first published in Brunton et al. (2016a). It utilizes concepts of sparsity and compressed sensing, which were introduced in section 2.2.1, to find the active terms by means of a sparse regression. The SINDY algorithm places an emphasis on parsimonious models that balance accuracy and model complexity witch in turn avoids overfitting. The sparse regression also make the method more robust against outliers in the data used for training the model. In particular SINDY uses the fact that most physical dynamical systems only exhibit a few relevant terms in their governing equations (Brunton et al., 2016a). A dynamical system in state space representation, as explained in section 2.1, is given as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) \tag{5.38}$$

whereby $\mathbf{x} \in \mathbb{R}^d$ is the state vector fully describing the system at time t. The vector-valued function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ expresses the dynamics of the system and for many systems will only consist of a few terms, making the function sparse in a suitable high-dimensional function basis.

Pairs of input and output data in the form of the state vector $\mathbf{x}(t_i)$ and its derivative $\dot{\mathbf{x}}(t_i)$ of the dynamical system are needed for the system identification. The sampling of the data can be done in arbitrary fashion and subsequent samples do not need to lie on the same trajectory through the phase space. However, a uniform sampling of the whole phase space representing relevant and realistic system states is desirable for improved model performance and generalizability. Furthermore, data based on observations of a real physical system automatically represent a time series of the state vectors and their derivative of a trajectory through phase space as the state of the dynamical system by definition evolves along such a trajectory. In case the derivative of the state vector were not measured directly they may be calculated by means of numeric differentiation. In this case the state vector data must consist of a densely sampled time series.

The measured state vectors and their derivatives are combined to form the data matrices $\mathbf{X} \in \mathbb{R}^{m\times n}$ and $\dot{\mathbf{X}} \in \mathbb{R}^{m\times n}$ respectively, as is shown in equations 5.39 and 5.40.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T(t_1) \\ \mathbf{x}^T(t_2) \\ \vdots \\ \mathbf{x}^T(t_m) \end{bmatrix} = \overset{\xrightarrow{\text{State variables}}}{\begin{bmatrix} x_1(t_1) & x_2(t_1) & \dots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \dots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \dots & x_n(t_m) \end{bmatrix}} \Big\downarrow \text{Time} \tag{5.39}$$

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}^T(t_1) \\ \dot{\mathbf{x}}^T(t_2) \\ \vdots \\ \dot{\mathbf{x}}^T(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \dots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \dots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \dots & \dot{x}_n(t_m) \end{bmatrix} \tag{5.40}$$

A feature library matrix $\boldsymbol{\Theta}(\mathbf{X}) \in \mathbb{R}^{m \times p}$ of non-linear candidate functions of the columns of $\mathbf{X}$ is constructed. The choice of candidate functions is basically unlimited and the choice of the family of functions may be informed from domain expertise of the specific problem at hand. However, polynomials and trigonometric functions represent a function basis that is well suited for many problems as many functions are either sparse when expressed in those bases or may be approximates well using them. As an example a feature library containing constant terms, polynomial terms and trigonometric terms is shown in equation 5.41.

$$\boldsymbol{\Theta}(\mathbf{X}) = \begin{bmatrix} | & | & | & | & & | & | & | & | & \\ 1 & \mathbf{X} & \mathbf{X}^{\mathbf{P}_2} & \mathbf{X}^{\mathbf{P}_3} & \dots & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \sin(2\mathbf{X}) & \cos(2\mathbf{X}) & \dots \\ | & | & | & | & & | & | & | & | & \end{bmatrix} \tag{5.41}$$

$\mathbf{X}^{\mathbf{P}_i}$ is used to denote all monomials of degree i that can be constructed with all possible combinations of the state variables.

Every column of $\boldsymbol{\Theta}(\mathbf{X})$ represents one candidate function, evaluated at all system states represented as state vectors in the data. SINDY then uses a model consisting of a linear combinations of the candidate functions, resulting in a non-linear dynamical system model that is linear in its parameters. In vector notation the model is given as

$$\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi} \tag{5.42}$$

with $\boldsymbol{\Xi} \in \mathbb{R}^{p \times n}$ being the matrix containing the model parameters. Each column represents the coefficients for the model of one equation of the differential equation system. The model of the i-th equation can be written as

$$\dot{x}_i = f_i(\mathbf{x}) \approx \sum_{k=1}^{p} \theta_k(\mathbf{x})\xi_{k,i} = \boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\xi}_i \tag{5.43}$$

whereby $\theta_k(\mathbf{x})$ denotes the k-th candidate function that was used to generate the data matrix $\mathbf{\Theta}$ and $\boldsymbol{\xi}_i$ denotes the i-th column of the coefficient matrix with $\xi_{k,i}$ referring to k-th element of that column.

SINDY uses sparse regression in order to solve for $\mathbf{\Xi}$. For this the model equation is augmented by a normal distributed noise term allowing for discrepancies between the regression target and achieved model output:

$$\dot{\mathbf{X}} = \mathbf{\Theta}(\mathbf{X})\mathbf{\Xi} + \eta\mathbf{Z} \tag{5.44}$$

with $\mathbf{Z}$ being is a matrix of independent identically distributed normal distributed entries with zero mean, and η being the noise amplitude.

There exist multiple options to achieve sparse regression such as the least absolute shrinkage and selection operator (LASSO) (Tibshirani, 2011) which is an ℓ1-regularized regression. An alternative is sequential thresholded least-squares estimation which is computationally less expensive and thus scales better to large amount of data. In the case of this thesis the SINDY algorithm implementation pysindy (Silva et al., 2020b), see also section 3.2.4, was used, which is an open source project developed by the team that developed the SINDY algorithm. The sequentially thresholded least-squares optimization function of this library was used for the calculation. This functions internally uses a sequentially thresholded Ridge regression (Hoerl & Kennard, 1970) which is a method for estimating the coefficients of models from data where the independent variables might be highly correlated.

The optimization criterion for this optimisation is given by

$$\boldsymbol{\xi}_k = \underset{\boldsymbol{\xi}'_k}{\operatorname{argmin}} \left\| \dot{\mathbf{X}} - \mathbf{\Theta}(\mathbf{X})\boldsymbol{\xi'}_k \right\|_2 + \alpha \left\| \boldsymbol{\xi}'_k \right\|_2 \tag{5.45}$$

whereby α is a regularization parameter which represents a hyperparameter that has to be tuned. The Ridge regression then estimates the parameters by

$$\boldsymbol{\xi}_k = (\mathbf{\Theta}(\mathbf{X})^T\mathbf{\Theta}(\mathbf{X}) + \alpha^2\mathbf{I})^{-1}\mathbf{\Theta}(\mathbf{X})^T\dot{\mathbf{X}} \tag{5.46}$$

with $\mathbf{I}$ denoting the identity matrix. In order to promote sparsity a sequentially thresholded Ridge regression is used. Thereby the regression is performed repeatedly while all parameters with an amplitude below the threshold T are set to zero. This is done until convergence of the parameters is achieved or the maximum number of iterations is reached. The threshold T constitutes an additional hyperparameter to be tuned.

The SINDY framework has four major aspects important for the user:

- Choice of state vector coordinates
- Choice of candidate function in the feature library

- Differentiation method in case the derivative of the state vector was not observed directly
- Type of sparse regression method

5.3.2 SINDY with control input

The SINDY framework has been extended to be applied to problems with external forcing (Brunton et al., 2016b), (Fasel et al., 2021). The differential equation of such problems has the form

$$\dot{\mathbf{x}}_k = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{5.47}$$

whereby $\mathbf{u}$ denotes a vector with control inputs that actuate the system. For the system identification of such a system the control inputs are also needed which have to be recorded during the data acquisition. The SINDY model is extended to incorporate the external forcing influence by augmenting the feature matrix to contain not only the candidate functions based on the state vector but also the control input vector $\mathbf{u}$ and combinations of elements of the two.

As done with the state vector and its derivative the control input vectors for the different data points are combined to form the control input data matrix $\mathbf{U} \in \mathbb{R}^{u \times k}$:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}^T(t_1) \\ \mathbf{u}^T(t_2) \\ \vdots \\ \mathbf{u}^T(t_m) \end{bmatrix} = \begin{bmatrix} u_1(t_1) & u_2(t_1) & \dots & u_k(t_1) \\ u_1(t_2) & u_2(t_2) & \dots & x_k(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_m) & u_2(t_m) & \dots & u_k(t_m) \end{bmatrix} \tag{5.48}$$

Continuing the example for the feature library from equation 5.41, the matrix becomes

$$\boldsymbol{\Theta}(\mathbf{X}, \mathbf{U}) = \begin{bmatrix} | & | & | & | & | & & | & | & | & \\ 1 & \mathbf{X} & \mathbf{U} & \mathbf{X} \otimes \mathbf{X} & \mathbf{X} \otimes \mathbf{U} & \dots & \sin(\mathbf{X}) & \sin(\mathbf{U}) & \sin(\mathbf{X} \otimes \mathbf{U}) & \dots \\ | & | & | & | & | & & | & | & | & \end{bmatrix} \tag{5.49}$$

whereby $\mathbf{x} \otimes \mathbf{y}$ denotes the vector with all possible product combinations. The model then becomes

$$\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X}, \mathbf{U})\boldsymbol{\Xi} \tag{5.50}$$

with the optimization for the estimation of the parameters being exactly identical to the case without external forcing:

$$\boldsymbol{\xi}_k = \underset{\boldsymbol{\xi}'_k}{\operatorname{argmin}} \left\|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X},\mathbf{U})\boldsymbol{\xi'}_k\right\|_2 + \alpha \left\|\boldsymbol{\xi}'_k\right\|_2 \tag{5.51}$$

$$\boldsymbol{\xi}_k = (\boldsymbol{\Theta}(\mathbf{X},\mathbf{U})^T\boldsymbol{\Theta}(\mathbf{X},\mathbf{U}) + \alpha^2\mathbf{I})^{-1}\boldsymbol{\Theta}(\mathbf{X},\mathbf{U})^T\dot{\mathbf{X}}_k \tag{5.52}$$

5.3.3 SINDY with integral notation

For better numerical stability in the presence of noisy data, the integral notation of the SINDY algorithm was used. This concept was published in Schaeffer and McCalla (2017). The notation enables to avoid the differentiation of the state vector in order to get its derivative in use cases where the derivative was not measured directly but had to be calculated numerically. There exist many methods for numerical differentiation, for example using spline interpolation or numerical derivative processes that utilize total variation regularization Chartrand (2011). Nevertheless, numerical differentiation in the presence of noise remains difficult. Additionally there might be high frequency components in the time series data used for system identification that are not part of system dynamics of interest, but get amplified during the differentiation process and influence the estimated model parameters. In the context of this thesis this might be vibrations caused by flexing of the physical structure on which the CCRs are mounted to or by the steps taken by the stepper motors of the turning hub. Such effects can largely cancel out using the integral notation.

The model introduced in the previous section

$$\dot{x}_i(t) = \sum_{k=1}^{p} \theta_k(\mathbf{x},\mathbf{u})\xi_{k,i} \tag{5.53}$$

is integrated over a time span τ from a start time t_j:

$$\int_{t_j}^{t_j+\tau} \dot{x}_i(t)\, dt = \int_{t_j}^{t_j+\tau} \sum_{k=1}^{n} \xi_{i,k}\Phi_k(\mathbf{x}(t),\mathbf{u}(t))dt \tag{5.54}$$

Due to the linearity of the model with regards to the parameters only the feature library terms have to be integrated. Furthermore, the state vector derivative is replaced by the difference of the state vector variables at the integral bound times:

$$x_i(t_j+\tau) - x_i(t_j) = \Delta x_{i,j} = \sum_{k=1}^{n} \xi_k \int_{t_j}^{t_j+\tau} \Phi_k(\mathbf{x}(t),\mathbf{u}(t))dt \tag{5.55}$$

The estimated parameters are the same as for the differential notation. Therefore this methodology can be used to estimate the same differential equations of a dynamical system with the added benefit of more numeric stability due to the usage of differences in the state variables instead of their derivative and partial cancellation of high frequent noise such as caused by vibrations due to the integration of the feature library terms. A new variable for the integrated feature matrix is introduced:

$$\mathbf{\Theta}'(\mathbf{X},\mathbf{U}) = \begin{bmatrix} \int_{t_1}^{t_1+\tau}\Theta_1(\mathbf{x}(t),\mathbf{u}(t)) & \int_{t_1}^{t_1+\tau}\Theta_2(\mathbf{x}(t),\mathbf{u}(t)) & \dots & \int_{t_1}^{t_1+\tau}\Theta_k(\mathbf{x}(t),\mathbf{u}(t)) \\ \int_{t_2}^{t_2+\tau}\Theta_1(\mathbf{x}(t),\mathbf{u}(t)) & \int_{t_2}^{t_2+\tau}\Theta_2(\mathbf{x}(t),\mathbf{u}(t)) & \dots & \int_{t_2}^{t_2+\tau}\Theta_k(\mathbf{x}(t),\mathbf{u}(t)) \\ \vdots & \vdots & \ddots & \vdots \\ \int_{t_p}^{t_p+\tau}\Theta_1(\mathbf{x}(t),\mathbf{u}(t)) & \int_{t_p}^{t_p+\tau}\Theta_2(\mathbf{x}(t),\mathbf{u}(t)) & \dots & \int_{t_p}^{t_p+\tau}\Theta_k(\mathbf{x}(t),\mathbf{u}(t)) \end{bmatrix} \tag{5.56}$$

The parameter estimation can then be done as described previously. The regression problem becomes

$$\mathbf{\Delta X} = \mathbf{\Theta}'(\mathbf{X})\mathbf{\Xi} + \eta\mathbf{Z} \tag{5.57}$$

with the optimization problem being

$$\boldsymbol{\xi}_k = \underset{\boldsymbol{\xi}'_k}{\operatorname{argmin}} \left\|\mathbf{\Delta X}_k - \mathbf{\Theta}'(\mathbf{X},\mathbf{U})\boldsymbol{\xi'}_k\right\|_2 + \alpha\left\|\boldsymbol{\xi}'_k\right\|_2 \tag{5.58}$$

and the parameters being estimated using a sequentially thresholded Ridge regression as described in section 5.3.1. The regression of one iterations step then becomes:

$$\boldsymbol{\xi}_k = (\mathbf{\Theta}'(\mathbf{X},\mathbf{U})^T\mathbf{\Theta}'(\mathbf{X},\mathbf{U}) + \alpha^2\mathbf{I})^{-1}\mathbf{\Theta}'(\mathbf{X},\mathbf{U})^T\mathbf{\Delta X}_k \tag{5.59}$$

For this thesis the integrals were approximated numerically using the trapezoidal rule.

5.3.4 Applying SINDY to Husky A200

For the application of SINDY for modeling the driving dynamics of the Husky A200 the state vector coordinates describing the system had to be defined. The choice of state vector coordinates is very restricted as they are only a few possible ways to fully describe the system. Furthermore the choice is partly dictated by the likely use cases of such a model. Applications of the model such as navigation or path planing usually operate on position and velocity as description of the system state. The used state vector representation of the system therefore contains the pose of the vehicle, consisting of its position and attitude. As only 2D dynamics on level driving surfaces is discussed in this thesis, the pose has three values with two coordinates and the heading angle. In order to fully describe the state of the vehicle the first derivative, representing the linear and angular velocities, is also included in the state vector. This is consistent with the reasonable assumption that the control inputs for robot which direct the motor drivers affect the robot in the form of accelerations. This leads to a system of second order differential equations in order to relate the vehicle's pose and control inputs with the acceleration which represents the second derivative of the pose. By including the first derivative of the pose in the state vector the order of the differential equations is reduced, as is explained in section 2.1. The resulting state vector is then

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \psi \\ \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} x \\ y \\ \psi \\ v_x \\ v_y \\ \omega \end{bmatrix}. \tag{5.60}$$

Due to the state vector containing the pose and its derivative, estimating a model for the derivative of the pose based on the state vector is trivial, as the state vector by definition contains the sought values:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \psi \\ \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \ddot{x} \\ \ddot{y} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{x},\mathbf{u}) \\ f_2(\mathbf{x},\mathbf{u}) \\ f_3(\mathbf{x},\mathbf{u}) \\ f_5(\mathbf{x},\mathbf{u}) \\ f_5(\mathbf{x},\mathbf{u}) \\ f_6(\mathbf{x},\mathbf{u}) \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ f_5(\mathbf{x},\mathbf{u}) \\ f_5(\mathbf{x},\mathbf{u}) \\ f_6(\mathbf{x},\mathbf{u}) \end{bmatrix} \tag{5.61}$$

This also results directly from the reduction of order of the differential equation system. Additionally the driving dynamics to be modeled are assumed to be homogeneous and isotropic which means that the system response is invariant with regard the position and heading of the vehicle. A dependence of the driving dynamics model on the attitude and position within the measurement laboratory is undesirable as the training data was collected while driving on level ground with the same surface over the whole test area. Even if small changes in the driving dynamics over the measurement area were to exist, including these effects into the system model is undesirable as the location and orientation of the used local reference frame is arbitrary and the model would not generalize. Thus the state vector is truncated for the system identification to only contain the first derivatives of the pose:

$$\tilde{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} \tag{5.62}$$

The system identification problem then becomes

$$\frac{d}{dt}\tilde{\mathbf{x}} = \mathbf{f}(\tilde{\mathbf{x}},\mathbf{u}) \tag{5.63}$$

However, this differential equation system still depends on the orientation of the used local reference frame as the linear velocity and by extension the linear acceleration is expressed with regard the the frame's axes. Under the assumption of homogeneity and isotropy it is self-evident that the driving dynamics are inherently a body frame phenomenon as the the control inputs effect changes in the body frame. Therefore the state vector representation is transformed from a local reference frame into the vehicle's body frame. Without this transformation the model resulting from system identification would have to discover the 2D transform relationship between the body frame and the local reference frame in addition to the actual driving dynamics.

Following the notation introduced in section 2.5 the transformation of the linear velocities is given by

$$\mathbf{v}^b = \begin{bmatrix} v_x^b \\ v_y^b \end{bmatrix} = \mathbf{C}_l^b \mathbf{v}^l = \mathbf{R}(\psi) \begin{bmatrix} v_x^l \\ v_y^l \end{bmatrix} \tag{5.64}$$

wherein the superscript b denotes the body frame of the Husky A200 and l denotes the local reference frame. The transformation matrix $\mathbf{R}(\psi)$ is a 2D rotation matrix given by

$$\mathbf{R}(\Psi) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{5.65}$$

The angular velocity and acceleration of the vehicle's heading is invariant to this transformation. The state vector expressed in the body frame is then given by

$$\mathbf{x}^b = \begin{bmatrix} v_x^b \\ v_y^b \\ \dot{\psi} \end{bmatrix}. \tag{5.66}$$

This representation was ultimately used to conduct the system identification of the driving dynamics.

The Husky A200 represents a dynamic system with external forcing as it is actuated by its drivetrain. Therefore SINDY with control, described in section 5.3.2, is employed to estimate the governing equations that incorporate the control inputs given to the vehicle during driving. For the system identification in this thesis the inputs from the handheld controller which the Husky A200 received were used as the control input $\mathbf{u}$. This input consists of the two values which controlled the forwards/backwards and turning motion of the vehicle respectively. Both inputs have a range of $[-1, 1]$. How these values are interpreted and used to drive the two motors was seen as part of the dynamical system to be identified. An input that corresponds more closely to the real motor driver output may improve the model performance as the control algorithm which calculates those signals would not have to be discovered by the system identification in addition to the driving dynamics. Future experiments regarding this possibility are planned. However, the input signals from the hand held controller used are those that end users of the vehicle would find readily available. The Husky supports two drive modes which correspond to different maximal driving speeds and can be controlled by pressing different buttons on the controller for "arming" the robot. Only the mode with the higher maximum driving speed is treated in this thesis.

For the feature library only polynomials up to degree three were used for the system identification in this thesis as the combination of polynomials and trigonometric functions was found to lead to ill-conditioned equation systems. An investigation using a broader range of non-linear candidate functions is needed to assess its impact on the model performance for the driving dynamics in this specific use case.

The need for numeric calculation of the state vector derivatives was circumvented by using the integral notation of SINDY which is described in section 5.3.3.

5.3.5 Calculation of input data

For the system identification three types of input data are required. On one hand the system input both in form of the state vector and optionally the control vector describing the external forcing is required. On the other hand the system output which represents the regression target is needed. In the case of the integral notation explained in section 5.3.3 the output is not the derivative of the state vector but the difference between two state vectors a certain integration length τ apart in time.

As shown in the previous section the system to be identification in this thesis uses the first derivative of the 2D pose of the Husky A200 expressed in the vehicles body frame, representing the linear and angular velocities, as the state vector. Without the integral notation the second derivative of the 2D pose, constituting the linear and angular accelerations would also be needed.

In order to obtain this the components of the pose data calculated as described in section 5.2.4, x, y and ψ, were interpolated using cubic splines. Cubic splines are a localized, C^2 continuous interpolation and therefore well suited for the calculation of the pose derivatives. Differentiating a spline approximation is also straight forward and computationally efficient. The interpolation was conducted using the `UnivariateSpline` function from the SciPy package (see section 3.2.4). The resulting spline approximation functions were then evaluated at the trigger timestamps, resulting in a approximately 50 Hz time series of the state vectors:

$$\mathbf{x}^l(t_i) = \begin{bmatrix} \dot{x}^l(t_i) \\ \dot{y}^l(t_i) \\ \dot{\psi}^l(t_i) \end{bmatrix} \tag{5.67}$$

These state vectors when then transformed into the body frame as described in the previous section while using the current azimuth of the pose as a transformation parameter.

The recorded control inputs for the robot were linearly interpolated at the times of the Raspberry Pi trigger time stamps as they are needed at the same times as the state vector data. This resulted in the data points $\mathbf{u}(t_i)$.

Despite the high measurement frequency and precise measurement method minute, high frequency variations in the calculated pose time series persisted and got amplified by the differentiation. These high frequency components likely are not part of the desired macroscopic driving dynamics and are caused by vibrations, the flexing of the physical structure the CCRs are mounted to, the discrete steps taken by the stepper motor and other phenomena. Therefore it was tested whether a smoothing of the interpolation resulting in an approximation would result in a better model predictive performance with regard to the multi-step trajectory prediction in a local reference frame. For the smoothing the employed Scipy function `UnivariateSpline` increases the number of knots of the spline interpolation until the smoothing condition

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sigma_i^2}(y_i - f(x_i))^2 \leq s \tag{5.68}$$

is met. y_i denotes the i-th data point and $f(x_i)$ the corresponding approximation value. The the individual data points were weighted by the inverse of their variance σ_i^2 obtained by the least squares adjustment during the pose estimation explained in section 5.2.4. Different values for s, the smoothing parameter, were tested. If s is zero the data is interpolated exactly, with the spline going through all data points.

The smoothing has an effect that is complimentary to that of the usage of the integral notation for the SINDY algorithm. It was found to be particularly important in preliminary calculations that used the differential notation of the SINDY algorithm. However, it was found to also have an effect on the parameter estimation with the integral notation, so different values for the smoothing parameter s were tested.

Figure 5.8 depicts a short sample of the time series of the three pose components and their first and second derivative. With the exception of the approximation using the largest amount of smoothing with s set to 0.01, the approximations follow the pose data points quite closely. However, the derivatives resulting from the different interpolations are very different. The blue curves are based on the exact interpolation and demonstrate that the second derivatives of the pose components have large, high frequency oscillations resulting on very hard to model regression targets with alternating sign over even very short time frames during smooth driving operations. Figure 5.9 highlights this by showing the amplitude spectrum of the three state vector components and of their derivatives for the four tested smoothing values for s. The amplification of noise and likely also other effects such as vibrations can be clearly seen in the second derivative, showcasing the benefits of using the integral notation of SINDY, introduced in section 5.3.3, as it alleviates the necessity of calculating the second derivative.

5.3.6 Hyperparameter tuning

There are four hyperparameters that may be tuned in the methodology outlined above. Two of them pertain to the data preprocessing in the form of the smoothing factor s used in the pose interpolation for the derivative calculation (see section 5.3.5) and in the form of the integration window length τ (see section 5.3.3). The other two concern the sparse regression in the context of SINDY and are the regularization parameter α and the coefficient threshold T (see section 5.3.1). Table 5.1 lists the tested values for each hyperparameter.

The first class of hyperparameter is more difficult to tune because they alter the regression target itself whereby a better prediction performance in the body frame may not translate to a more accurate prediction in a local reference frame which ultimately is the most important performance metric. A full grid search with cross validation for all four hyperparameter si-

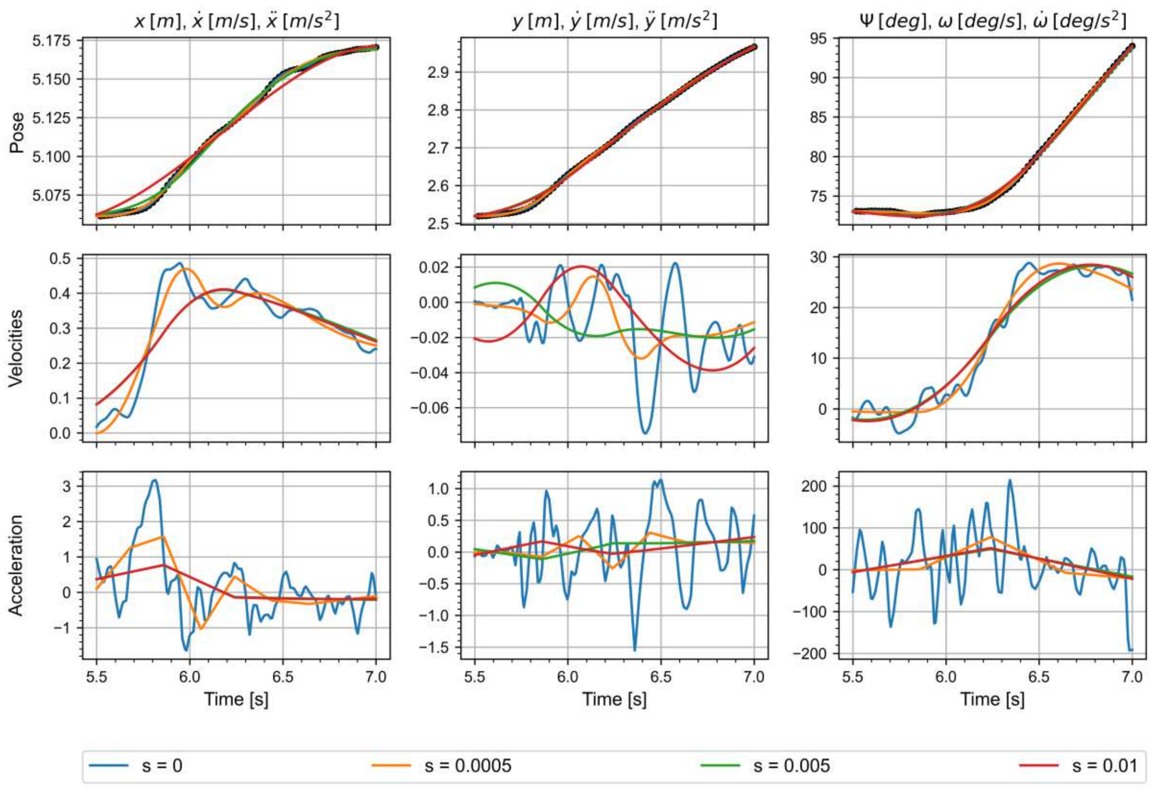

Figure 5.8: Short (≈ 1.5 s) sample of the pose data along side with their interpolations using different smoothing values as well as the resulting first and second derivative

Table 5.1: List of all tested values for the four different hyperparameters during tuning

Hyperparameter	Value
s	0, 0.0005, 0.005, 0.01
τ	3 s
α	0, 1, 5, 10, 30, 50
T	0, 0.25, 0.50 1, 3

multaneously evaluated on the multi-step predictive performance in the local reference frame was computationally intractable given the available computational hardware and time constraints. Therefore a selection of values for s were selected and used to create different data sets with the preprocessing steps. For the parameter τ 3 seconds were used for all experiments. Subsequently a gridsearch for the parameters α and T using 5-fold cross validation was conducted for the different data sets individually using the functionality of the `GridSearchCV` class of the sklearn Python library. The gridsearch used the predictive performance of the body frame model with regard to $\mathbf{\Delta x}$ as a performance metric. The values and by extension the results of this step are not comparable across the different data sets resulting from the different preprocessing parameters. This is because the preprocessing alters the data on which the grid search operates. For example a longer integration window length τ will cause larger predictions errors as a longer integration causes more errors to accrue. Likewise a stronger smoothing indicated by a larger smoothing factor s will lead to smaller prediction errors. Therefore resulting models based on different data sets can only be compared based on their multi-step prediction of the trajectory in the local reference frame when compared

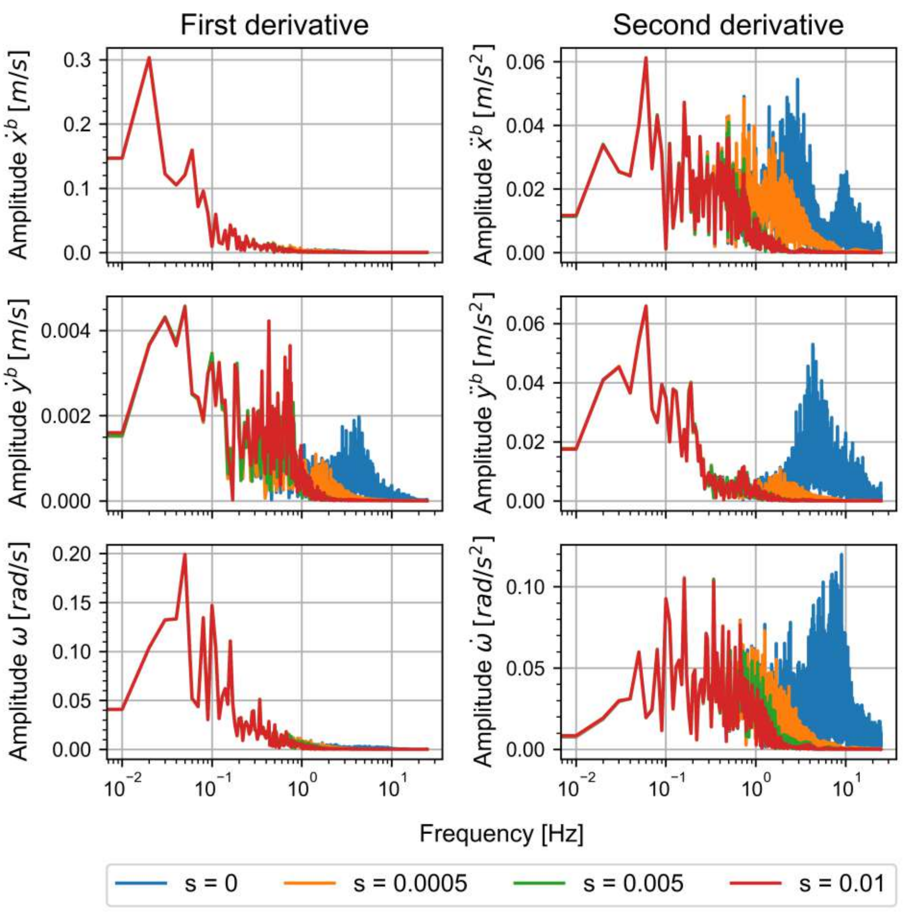

Figure 5.9: Amplitude spectra of the first and second derivative of the spline interpolation of the pose data in the body frame for different smoothing values

to the ground truth of measured poses.

Therefore some models were selected for each data set individually based on the associated gridsearch results. Those models were trained on the whole training data set and subsequently used to make predictions for the pose trajectory in local reference frame. Those results were ultimately compared. For this the different experimental runs are split into a training and test data set with an training-test split of about 75%-25%.

5.3.7 Using model in multi-step prediction

As the driving dynamics are modeled with regard to the vehicle's body frame an additional step is necessary when using the model in order to predict the path of the vehicle. The model resulting from the system identification predicts the derivative of the state vector in the body frame of the vehicle:

$$\dot{\mathbf{x}}^b \approx \mathbf{f}(\mathbf{x}^b, \mathbf{u}) \tag{5.69}$$

Using a numerical integration method, such as the Runge-Kutta family of methods, discrete points along an approximate trajectory through phase space can be calculated. For example using the Runge-Kutta method of order 4 would be applied to the problem like this:

$$\begin{aligned}\mathbf{x}^b_{n+1} &= \mathbf{x}^b_n + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right)h \\ t_{n+1} &= t_n + h\end{aligned} \tag{5.70}$$

whereby

$$\begin{aligned}\mathbf{k}_1 &= \mathbf{f}\left(\mathbf{x}^b(t_n), \mathbf{u}(t_n)\right) \\ \mathbf{k}_2 &= \mathbf{f}\left(\mathbf{x}^b(t_n) + \frac{h}{2}k_1, \mathbf{u}\left(t_n + \frac{h}{2}\right)\right) \\ \mathbf{k}_3 &= \mathbf{f}\left(\mathbf{x}^b(t_n) + \frac{h}{2}k_2, \mathbf{u}\left(t_n + \frac{h}{2}\right)\right) \\ \mathbf{k}_4 &= \mathbf{f}\left(\mathbf{x}^b(t_n) + hk_3, \mathbf{u}\left(t_n + h\right)\right).\end{aligned} \tag{5.71}$$

In this thesis the explicit Runge-Kutta method of order 5(4) (Dormand & Prince, 1980) is used as implemented by the `integrate.solve_ivp` function of the Scipy Python library (Virtanen et al., 2020). This function adapts the step size automatically for smaller time steps where needed to maintain adequate precision.

The control inputs $\mathbf{u}$ are interpolated linearly in order to evaluate the function $\mathbf{f}$. The result of the calculation is a list of state vectors in the body frame $\mathbf{x}^b_n$ with $n = 0..N$. An initial condition $\mathbf{x}^b_0$ is needed for the initial value calculation. In many applications of the model the initial condition will be given as a state vector with respect to the local reference frame. The initial condition with regard to the body frame is then calculated using the transformation introduced in section 5.3.4.

The obtained state vectors are truncated state vectors, as is described in section 5.3.4. They therefore only contain the linear and angular velocities in the body frame. Their transformation into the local reference frame depends on the current pose which is constantly changing based on the estimated derivatives. Therefore the transformation of the results of the numeric model integration and calculation of the trajectory of the vehicle in the local reference frame represents a differential equation itself. The body frame state vector may be transformed into the local reference frame using the inverse of the transform introduced in section 5.3.4, resulting in the truncated state vector with respect to the local reference frame written here as

$$\tilde{\mathbf{x}}^l_n = \mathbf{C}^l_b(\psi_l)\mathbf{x}^b_n. \tag{5.72}$$

The full, not truncated state vector expressed in the local reference frame introduced in equation 5.3.4 is now explicitly denoted $\mathbf{x}^l$ in order to highlight the difference between it and the state vector of the driving dynamics expressed in the body frame. The initial condition of this state vector is then denoted

$$\mathbf{x}^l_{t_0} = \begin{bmatrix} x^l_{t_0} \\ y^l_{t_0} \\ \psi^l_{t_0} \\ \dot{x}^l_{t_0} \\ \dot{y}^l_{t_0} \\ \dot{\psi}^l_0 \end{bmatrix}. \tag{5.73}$$

The state vector time series in the local reference frame is then calculated one step at a time by transforming the current $\mathbf{x}^b_n$ using the current heading estimation ψ_n with equation 5.72 and subsequently updating the full state vector with regard to the local reference frame:

$$\mathbf{x}^l_{n+1} = \begin{bmatrix} x^l_k + \dot{x}^l_k \Delta t \\ y^l_k + \dot{y}^l_k \Delta t \\ \psi^l_k + \dot{\psi}^l \Delta t \\ \tilde{\mathbf{x}}^l_n \end{bmatrix} \tag{5.74}$$

This equates to the explicit Euler method. A possible improvement is the creation of a new, combined differential equation which contains the transformation from the body to the local reference frame:

$$\mathbf{g}(\mathbf{x}^l, \mathbf{u}) = \mathbf{C}^l_b(\psi^l_n)\mathbf{f}(\mathbf{C}^b_l(\psi^l_n)\mathbf{x}^l, \mathbf{u}) \tag{5.75}$$

This approach could potentially improve the estimation by enabling a combined numerical integration using high order methods, such as the Runge-Kutta methods, instead of the two step approach presented above. The evaluation of this multi-step prediction method for the dynamical model was not inside the scope of this thesis and requires additional investigations.

5.3.8 Evaluation of model performance

For the evaluation of the model predictive performance it is used in a multi step prediction and the estimated poses are compared to the ground truth provided by the laser tracker measurements. The testing data set was divided into 5 second long integration intervals whereby subsequent intervals overlapped by 50% in order to calculate more integration runs an thus get a more reliable result. Figure 5.10 shows an example of a run from the testing data set being used for the evaluation with the different integration periods illustrated. The first datapoint of every interval was used as an initial value for the numerical integration of the model as was described in the previous section. This estimated trajectory for the robot was then compared to the poses obtained from the laser tacker measurements which resulted in the errors of the 2D pose estimates denoted ϵ_x, ϵ_y and ϵ_ψ. As the location and orientation of the local reference frame in which those quantities are expressed is arbitrary the coordinate errors where combined to form the point position error defined as

$$\epsilon_p = \sqrt{\epsilon_x^2 + \epsilon_y^2}. \tag{5.76}$$

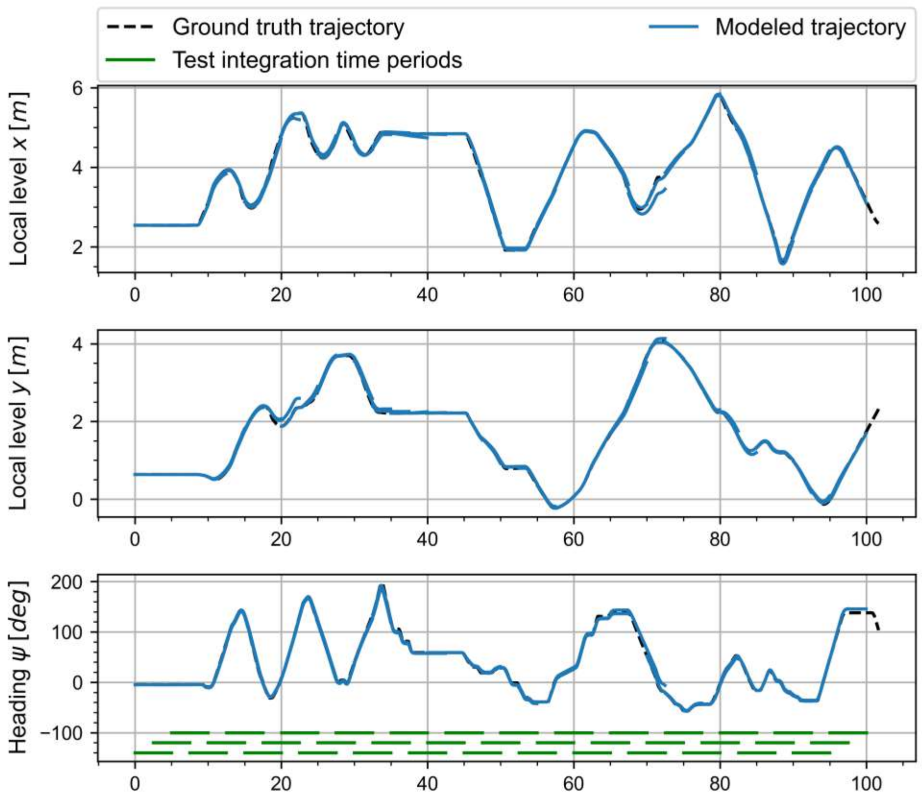

Figure 5.10: Example of the model evaluation using a multi-step prediction w.r.t the local reference frame

The time series of point position error and absolute heading error of the different 5 second intervals are then stacked. Based on this the standard deviation of the position and heading estimation as a function of integration time can be calculated:

$$s_p(t_i) = \sqrt{\frac{1}{N}\sum_{j=1}^{N} \epsilon_{p,j}(t_i)^2} \tag{5.77}$$

$$s_h(t_i) = \sqrt{\frac{1}{N}\sum_{j=1}^{N} \epsilon_{h,j}(t_i)^2} \tag{5.78}$$

Figure 5.11 shows an example of the evaluation of a specific model.

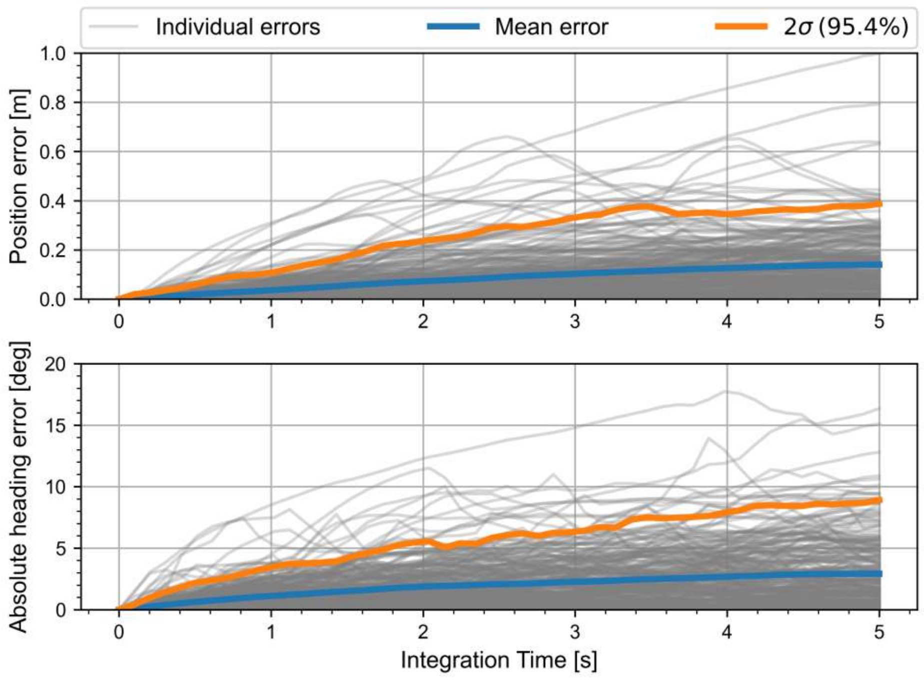

Figure 5.11: Example of the stacked deviations of the predicted trajectory and derived metrics

5.3.9 Least squares adjustment of model

The possibility of adjusting the estimated parameters of the active terms identified and estimated using SINDY using a subsequent least squares adjustment was tested. The concept of a least squares adjustment was introduced in section 2.4. There are three possible ways to introduce corrections into the functional model. The simplest way is to only consider the difference of state vectors $\Delta\mathbf{x}$ as stochastic quantities and thus only introducing correction for those terms. This results in

$$\Delta\mathbf{x}_i + \boldsymbol{v}_{\Delta x,i} = \sum_{k=1}^{p} \boldsymbol{\theta}'_k(\mathbf{x})\xi_{k,i} = \boldsymbol{\Theta'}(\mathbf{x})\boldsymbol{\xi}_i. \tag{5.79}$$

The next step would be to also apply corrections for each integrated library term which results in

$$\Delta\mathbf{x}_i + \boldsymbol{v}_{\Delta\mathbf{x},i} = \sum_{k=1}^{p} (\boldsymbol{\theta}'_k(\mathbf{x}) + v_{k,i})\xi_{k,i} = (\boldsymbol{\Theta'}(\mathbf{x}) + \boldsymbol{\epsilon}_{\boldsymbol{\Theta}'})\boldsymbol{\xi}_i. \tag{5.80}$$

The most rigorous way is to introduce corrections for the components of the state vectors used in the integration used for constructing the feature library matrix for the integral notation of SINDY (see section 5.3.3). This results in

$$\Delta\mathbf{x}_i + \boldsymbol{v}_i = \sum_{k=1}^{p} \theta'_k(\mathbf{x} + \boldsymbol{v})\xi_{k,i} = \sum_{k=1}^{p} \int_{t_j}^{t_j+\tau} \theta_k(\mathbf{x} + \boldsymbol{v}\mathbf{x})dt \;\; \xi_{k,i} = \boldsymbol{\Theta'}(\mathbf{x} + v_{\mathbf{x}})\boldsymbol{\xi}_i. \tag{5.81}$$

It was found that the equation system for solving the least squares adjustment was badly conditioned and that there was a very high correlation between estimated parameters. This issue was circumvented during the system identification process with SINDY by the Ridge regression. The cross correlation matrix resulting from the adjustment using the functional model described in equation 5.79 is still presented in section 7.9 for the discussion of the results.

The use of a least squares adjustment for improving the coefficients estimated using SINDY with rigorous stochastic modeling of the observations is highly desirable and needs further investigations. The conditioning problem may be alleviated by reducing the potential candidate functions in the feature library or perhaps by targeted subsampling of the training data.

6. Experiments

This section details the experiments conducted for obtaining the data needed for the system identification. The three major steps were geodetic network measurements, leverarm determination and the driving experiments. All data was collected on 21.12.2022 in the Engineering Geodesy measurement laboratory located at Gußhausstraße 25-27, 1040 Vienna, Austria.

6.1 Geodetic network

In order to calculate the transformation parameters between the two laser trackers a geodetic network was created with seven points realized as CCRs in the magnetic hubs mounted to the walls. Those points were measured with both laser trackers while being controlled through the Spatial Analyzer software. The "Precise Point" preset was utilized with a 5 second averaging period. The CCR for the measurements with the LTD800 was initialized in the bird bath for interferometric measurements.

Figure 6.1 shows the measured points forming the geodetic network and the two laser trackers along with their sensor frames. The resulting coordinates along with their uncertainties are listed in tables 6.1 and 6.2 for the AT960 and LTD800 respectively.

6.2 Body frame realization and leverarm determination

The position of the CCR reference points on the Husky A200 had to be known with respect to its body frame. For this the body frame, following the definition introduced in section 3.1.1, had to be realized and the position vectors of the two reference points, referred to as leverarms, calculated. The screw heads of the screw of the wheel hubs were used as measurement points in order to estimate the axle endpoint position. This allowed for the

Table 6.1: Measurement data of the geodetic network points from the AT960

Points	x [m]	y [m]	z [m]	$\sigma_x[\mu m]$	$\sigma_y[\mu m]$	$\sigma_z[\mu m]$
31	11.83188391	15.66555826	1.11614100	83.5	76.13	93.1
41	7.68306683	9.96930859	-1.05500903	53.8	45.95	59.4
43	11.55237925	7.14983024	0.84172574	48.6	59.55	66.1
53	7.44493518	2.26783293	-1.06306672	28.6	36.82	38.2
63	3.83942444	-1.99454397	0.85086516	19.1	19.93	21.2
61	-0.20351163	1.38697634	-1.02545438	8.3	11.03	10.0
51	3.62148080	5.92743913	1.30458020	33.1	29.17	34.3

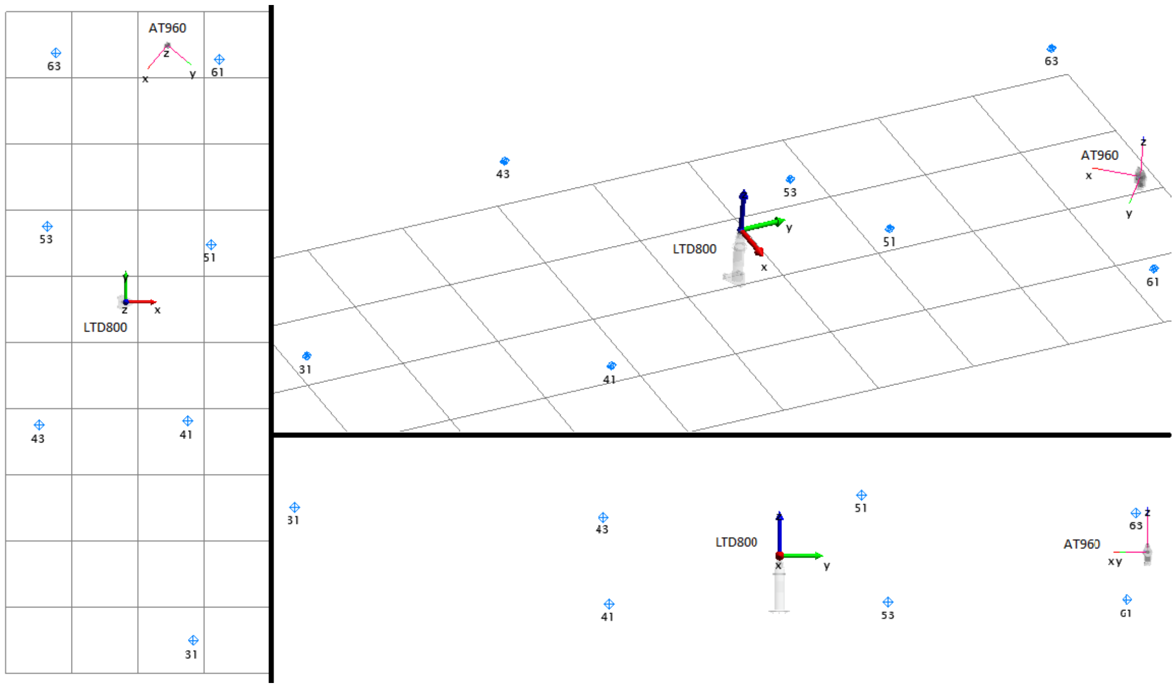

Figure 6.1: Overview of the measured network points and the sensor frames of the two laser trackers

Table 6.2: Measurement data of the geodetic network points from the LTD800

Points	x [m]	y [m]	z [m]	$\sigma_x[\mu m]$	$\sigma_y[\mu m]$	$\sigma_z[\mu m]$
43	-2.81484982	-3.93203020	0.86077272	22.0	20.8	24.1
31	2.11812526	-10.88127131	1.07420271	53.7	36.3	53.4
41	1.96576801	-3.82171493	-1.05034765	21.1	19.4	21.6
53	-2.50638871	2.45261077	-1.00324475	17.5	17.0	17.5
63	-2.21038951	8.01523260	0.94620957	39.8	29.0	40.0
61	3.04863959	7.78313096	-0.94827391	41.2	29.9	41.5
51	2.75969342	1.83857674	1.34382657	16.8	17.0	18.0

realization of the body frame as is described in section 3.1.1. The measurements were taken using the laser tracker LTD800, described in section 3.1.2, and the T-Probe, described in section 3.1.4, utilizing the 'Precise Stable Point' preset in Spatial Analyzer with a five second averaging period. The Husky A200 was placed about five meters away from the laser tracker, with its x-axis pointing towards the tracker. This enabled to measure the bolt heads of all four wheels with the same vehicle and instrument position. The ball tip of the T-Probe was slightly pressed into the screw heads for the measurements, where it rested securely due to the shape of the hex key screw head. The measurement process is depicted in figure 6.2. Subsequently, the CCR reference points were measured using the LTD800 and a CCR seated in the respective turning hub, using the 'Precise Stable Pt' preset in Spatial Analyser with a five second averaging period. The resulting coordinates are listed in table **??**.

As the T-Probe and CCR measurements were conducted with the same vehicle and laser tracker positioning, the collected wheel hub coordinates could be used to construct the Husky A200 body frame in the measurement frame of the LTD800. The coordinates of the CCR could subsequently be transformed from the measurement frame of the LTD800, in which

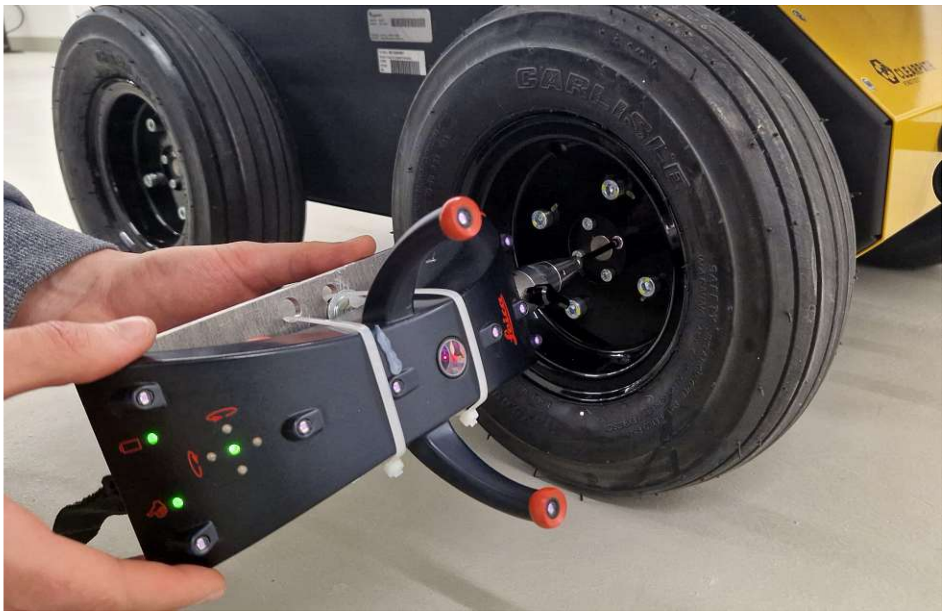

Figure 6.2: Measuring process of the wheel hub screw heads for the body frame realization

they were measured, into the constructed body frame, resulting in the required leverarms to the reference points on the Husk A200.

Table 6.3: Measurement data along with uncertainties for the body frame realization and leverarm determination

Points	x [m]	y [m]	z [m]	$\sigma_x[\mu m]$	$\sigma_y[\mu m]$	$\sigma_z[\mu m]$
Front left wheel	1.7458041	4.1128551	-1.1781067	20.6	18.4	20.7
	1.7466556	4.1301023	-1.1958863	21.1	18.9	21.0
	1.7459489	4.1122499	-1.2130364	20.9	19.1	20.7
	1.7451135	4.0950384	-1.1954413	20.1	18.5	20.2
Rear left wheel	1.7687535	4.6175709	-1.1724643	23.0	19.5	23.4
	1.7697480	4.6406026	-1.1822952	24.1	21.4	23.9
	1.7696002	4.6306282	-1.2048744	23.6	20.0	22.8
	1.7685448	4.6078053	-1.1949982	23.2	20.4	22.9
Rear right wheel	1.1874569	4.6530965	-1.1691820	22.8	19.9	23.0
	1.1862731	4.6328405	-1.1833869	22.5	20.2	24.1
	1.1877104	4.6673030	-1.1897254	23.3	19.7	23.1
	1.1865796	4.6471386	-1.2037590	23.3	19.2	23.3
Front right wheel	1.1622637	4.1299999	-1.1765262	21.8	19.1	21.1
	1.1613931	4.1241303	-1.2005666	20.9	19.2	20.1
	1.1624256	4.1480904	-1.2065659	21.1	18.8	20.1
	1.1632388	4.1539587	-1.1824637	21.4	18.4	21.0
CCR front	1.5132630	3.9057401	-0.6796971	19.7	17.7	19.1
CCR rear	1.3182130	4.6406633	-0.5319953	21.9	18.9	22.4

6.3 Driving experiments

In order to collect the needed data for the system identification there were nine driving experiment runs conducted, totalling about 45 minutes of driving time with the breakdown of the different run lengths given in table 6.4. During those runs the Husky A200, controlled

Table 6.4: Duration (rounded) of the different driving experiment runs and their allocation to either training of testing data

Run	Length [s]	Training data	Testing data
1	100		✓
2	390	✓	
3	340	✓	
4	300	✓	
5	100		✓
6	470	✓	
7	125	✓	
8	440	✓	
9	460		✓
Sum	45.4 min	34.4 min	11.0 min
		75.8%	24.2 %

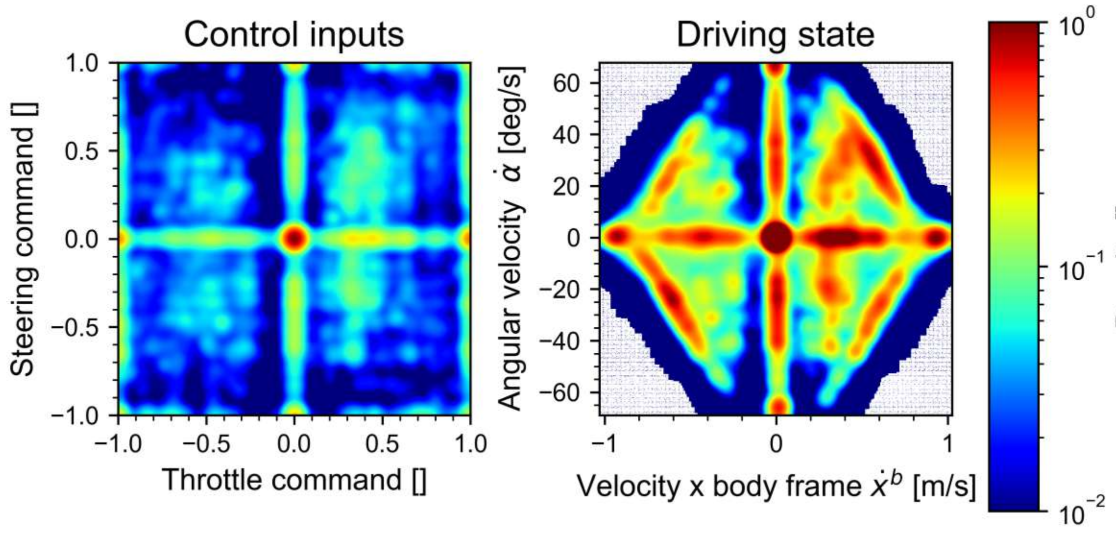

Figure 6.3: Density of the control inputs and driving states during all driving experiment runs

manually via the controller, was driven in between the two laser trackers within the area illustrated in figure 4.2.

It is important to sample the combined space of phase space and control input space as completely as possible for good system identification results that generalize well beyond the training data. Therefore many different driving scenarios including sharp changes in control input along with different system driving states were used during the experiments. Figure 6.3 depicts the density of different control inputs of all measured data as a heatmap. Alongside it the density of the distribution of driving state, in form of the velocity in driving direction, $\dot{x}^b$ and the rate of heading change, $\dot{\alpha}$, is shown. It is can be seen that the maximum forwards velocity of the vehicle and is decreased during cornering which is to be expected. The same is likewise true for the maximum turning speed depending on the forward velocity. The limitations might either be due to power limitations of the motors or limitations imposed by the driving control algorithm steering the motors.

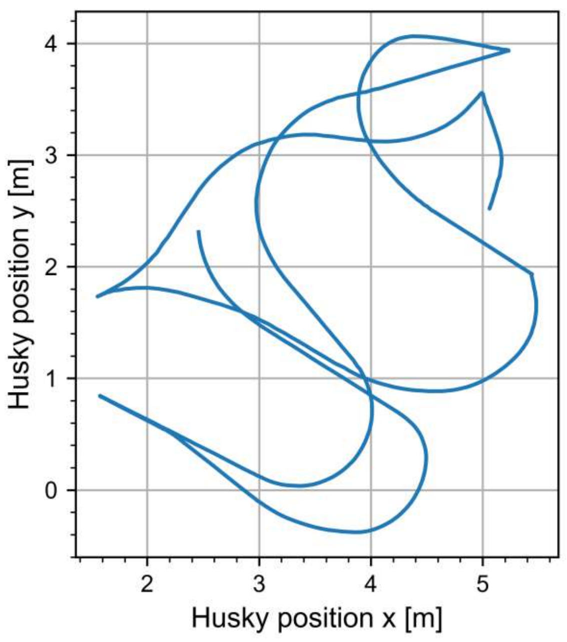

Figure 6.4: Example of a driving path during one of the driving experiment runs in the AT960 sensor frame which was used as local reference frame

The NTP software (see section 3.2.2) running on the Husky A200 onboard computer and the Raspberry Pi for the laser tracker triggering (see section 4.2.1) was given an one hour ”warm-up” period for synchronising their respective system clock. In order to avoid battery drainage of the Husky A200 during this process a Y-splitter was used to connect the vehicle with its battery and its wall charger simultaneously. This was necessary as the system computer could not be powered down as otherwise the system synchronization would had to start over again. The same configuration was also used in between different driving experiment runs in order to recharge the battery and slow down the battery depletion over the course of all experiments.

The vehicle could not be driven to complete a full rotation as this would lead to a loss of line of sight between the trackers and their respective prism as parts of the vehicle would obscure the view. This necessitated the alternating forward and backwards driving within the driving zone. An example of a driving path followed during one of the driving experiment runs is depicted in figure 6.4.

Before every driving experiment run the laser tracker beams were captured with their respective prism in case the line of sight was previously lost. Subsequently the prisms, placed in the motorized turning hubs, were oriented such that the laser beam would enter the prism with a central incidence direction. After this step the ROS node for the turning hub calculation was started and the first calculated turning hub azimuths were used as initial values for the turning hub position. At the beginning of every driving run the recording of rosbags for the relevant data in form of ROS messages was started both on the Husky A200 onboard computer for the control inputs and on the ROS server for the laser tracker data and the associated timestamps from the Raspberry Pi. After this step the driving experiment could commence, after which the data recording was stopped.

7. Results

7.1 Leverarm determination

Based on the measurements described in section 6.2 the body frame of the Husky A200 could be realized using the methodology described in section 5.1. Figure 7.1 depicts the measured points, the geometric objected fitted to them and the resulting body frame. Table 7.1 gives the leverarms of the two CCR reference points with regard to this frame. As the constructed body frame is seen as exact and not stochastic, the leverarm uncertainties only reflect the uncertainties of the CCR measurements.

Table 7.1: Resulting leverarm vectors of the two CCR reference point in the body frame

Points	x [m]	y [m]	z [m]	$\sigma_x[\mu m]$	$\sigma_y[\mu m]$	$\sigma_z[\mu m]$
Front (LTD800)	0.4671355	0.0655364	0.5167336	17.7	19.6	19.1
Rear (AT960)	-0.2600713	-0.1627405	0.6546911	18.9	22.0	22.4

7.2 Laser tracker transformation parameters

The parameter for the transformation from the sensor frame of the LTD800 into that of the AT960 could be calculated using the USMN function of Spatial Analyzer as described in

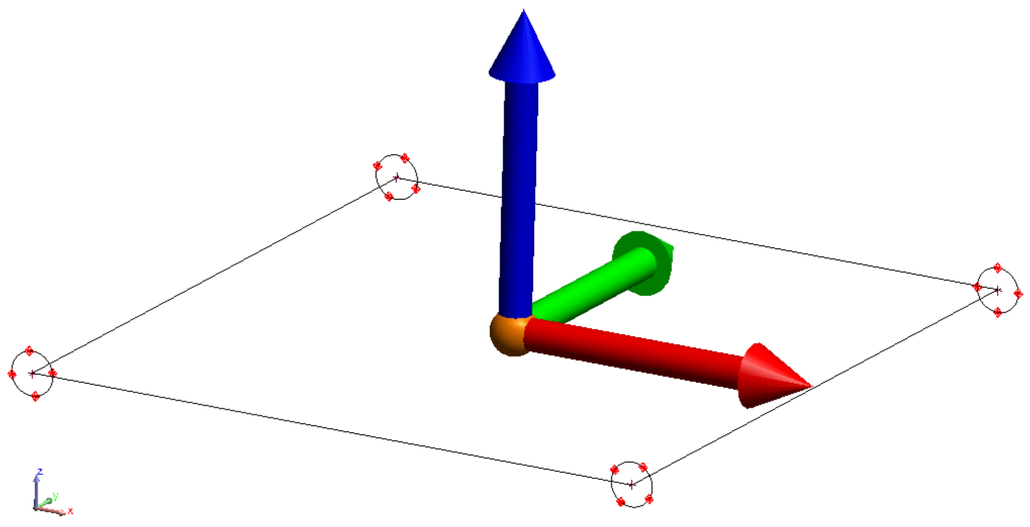

Figure 7.1: Visualization of the fitted geometry of the wheel hubs and plane through their centers along with the resulting body frame

section 5.2.1 based on the measurements described in section 6.1. The resulting parameter values are listed in table 7.2 along with their uncertainties. The uncertainties are of limited use as the mathematical model employed by Spatial Analyzer to combine the measurements of different sensors into one common frame is not transparent. The manual states "Instead of using a best-fit, USMN takes an intelligent weighted bundle approach. USMN examines each common measurement and considers the characteristics of the instruments that measured them and their positions in space." (New River Kinematics, 2020). In Lettner (2022) discrepancies between the estimated uncertainties estimated by the USMN function and a least squares adjustment of the same network were ascertained. The values are still useful for pertaining strong systematic errors in the data and are of the expected magnitude given the measurement uncertainties of the laser trackers.

Table 7.2: Parameters for the transformation from the sensor frame of the LTD800 into that of the AT960

	Translation			Rotation		
	t_x	t_y	t_z	R_x	R_y	R_z
Value	-2.050360 m	-8.757229 m	-0.086725 m	-0.374896°	-0.180820°	142.749476°
1σ	20.2 μm	25.9 μm	36.1 μm	1.058"	1.205"	0.700"

7.3 Time synchronization

The measurements of the two laser trackers obtained during the driving experiments, as described in section 6.3, were synchronized using the methodology outlined in section 5.2.2. The results from the cross correlation is omitted here as the found offset depends on the initial system clock offset which is not meaningful for the discussion.

7.3.1 Linear clock model

Table 7.3 lists the results of the linear clock model fit for the individual driving experiment runs. The estimated clock drift of the two internal laser tracker clocks are very stable over the whole measurement campaign and could be very well estimated using the model as is

Table 7.3: Estimated parameters of the linear clock model for the two laser tracker internal clocks for each driving experimental run

	LTD800			AT960		
Run	Drift	Drift STD	Offset STD	Drift	Drift STD	Offset STD
	[ppm]		[μs]	[ppm]		[μs]
1	16.2078	0.007	0.47	-28.6908	0.1524	10.08
2	16.2395	0.001	0.26	-28.4536	0.0191	4.26
3	16.2986	0.001	0.19	-28.4882	0.0293	5.71
4	16.2387	0.001	0.21	-28.5271	0.0321	5.58
5	16.2179	0.005	0.70	-28.6187	0.2333	33.51
6	16.2666	0.001	0.19	-28.5428	0.0156	4.21
7	16.2801	0.002	0.15	-28.6317	0.0826	6.09
8	16.2976	0.001	0.18	-28.5384	0.0205	5.31
9	16.3236	0.001	0.15	-28.5425	0.0184	4.98

Table 7.4: Estimated parameter for the additional AT960 offset caused by the measurement setup error along with the reduction in base line length variability

Run	Additional offset		Baseline length RMSE		
	Offset [ms]	Offset STD [ns]	Before [mm]	After [mm]	Reduction [%]
1	1.3180	2.95	0.452	0.193	57
2	1.3179	7.82	0.455	0.122	73
3	1.3537	5.29	0.479	0.137	71
4	1.3312	4.24	0.532	0.131	75
5	1.3247	4.29	0.515	0.149	71
6	1.3536	9.36	0.534	0.099	81
7	1.4718	2.45	0.461	0.136	70
8	1.3454	13.03	0.550	0.093	83
9	1.3517	17.18	0.741	0.094	87

evident by the corresponding standard deviations. However, the uncertainty of the estimated parameters if significantly worse for the AT960 clock. This is due to the measurement setup error described in section 4.2.2 in combination with the varying duty cycle of the trigger signal which is explained in section 4.2.1. The varying offset caused by this error caused the data points to not only be influenced by an offset and a linear click drift. However, despite the uncertainties of the estimates being higher as the functional model is not strictly fulfilled, the model parameters could still be estimated reliably due to the high number of data points used. Furthermore, errors in the estimated offset parameter would be counteracted by the estimated additional offset for the AT960 (see section 7.3.2).

The offset estimated is not listed for the same reason as for the cross correlation and because the offset of the linear clock model is completely complementary to that of the cross correlation.

7.3.2 Additional AT960 offset

The estimation of the additional offset of the AT960 time system that was necessitated by the measurement setup error described in section 4.2.2 was conducted as is explained in section 5.2.2. The estimated offset parameters for the different driving experiment runs are listed in table 7.4 alongside with the associated change in base line length variability.

The observed baseline length variability could be strongly reduced for all driving experiment runs. The lower reduction of the first run is likely because this run consisted of mainly of slower driving which reduced the effect an error in the time synchronisation of the laser trackers has on the observed baseline length.

The listed standard deviation of the estimated additional offset pertain to the minimum calculation of the interpolating quadratic polynomial function. As they are very small the baseline length variability could be interpolated very well and the additional time offset of the AT960 laser tracker could be eliminated based on the chosen methodology. However, these standard deviations are not reflective of the overall time synchronisation quality between the two laser trackers.

Table 7.5: The estimated coefficients of the turning hub error model along with their uncertainties

Run	Model parameters			Parameter STD [μm]			Baseline length RMSE	
	c [cm]	r_1 [mm]	r_2 [mm]	σ_c	σ_{r1}	σ_{r2}	before [mm]	after [mm]
1	77.46361	0.23	0.09	1.45	2.97	3.34	0.192	0.063
2	77.46268	0.26	0.11	1.07	2.96	2.93	0.121	0.068
3	77.46231	0.22	0.13	0.97	2.95	2.77	0.136	0.071
4	77.46118	0.19	0.12	1.16	4.16	3.71	0.130	0.080
5	77.46313	0.27	0.12	1.46	4.09	4.09	0.148	0.067
6	77.46290	0.19	0.09	0.87	2.08	2.09	0.098	0.080
7	77.46578	0.09	0.17	1.29	3.42	3.65	0.136	0.068
8	77.46210	0.19	0.10	1.04	3.09	3.18	0.091	0.079
9	77.46241	0.21	0.11	1.70	3.49	3.52	0.092	0.082
Mean	77.46290	0.21	0.12	1.23	3.25	3.25	0.127	0.073
STD	0.00128	0.053	0.02	0.27	0.64	0.60	0.032	0.007

7.4 Turning hub error model

The parameters of the turning hub error model introduced in section 5.2.3 were estimated for each driving experiment run individually. The results are listed in table 7.5. As can be seen the model resulted in a reduction of the unaccounted variations in the observed baseline length for all runs. The estimated constant term of the model is very consistent between the different runs, exhibiting a standard deviation of about 13μm between the different estimates. The estimated radii are with values of approximately 0.21 and 0.12 mm within the expected tolerance of the manufactured turning hub (see section 3.1.7). The results of the turning hub error model highlight the need for a improved measurement setup. Improved nests for the CCR on the motors of the turning hub were manufactured out of turned aluminum. However, a technical defect unrelated to this thesis rendered the Husky A200 inoperable and no experiments utilizing the improved version of the turning hub could be conducted. The ”Super cateye reflector” from Leica could alleviate the need for the motorized turning hubs alltogether as it offers 360° incidence angle of the laser beam of the laser trackers.

7.5 Pose calculations

After the laser tracker measurements were transformed into a common local horizontal reference frame, for which the sensor frame of the AT960 laser tracker which had been previously leveled was used, and the measurements were synchronised the poses of the Husky A200 could be calculated. The used methodology is described in section 5.2.4. Figure 7.2 and 7.3 show the time series of the pose parameters for one driving experiment along with their standard deviation obtained from the least squares adjustment as an example.

The estimated uncertainties of the pose component are larger than would be expected based solely on the uncertainties of the laser tracker measurements. This points to systematic effects being present which is also corroborated by smooth progression of the uncertainties over time. The variations in baseline length caused by turning hub error described in section 5.2.3 is likely the biggest cause. However, remaining time synchronisation issues may still

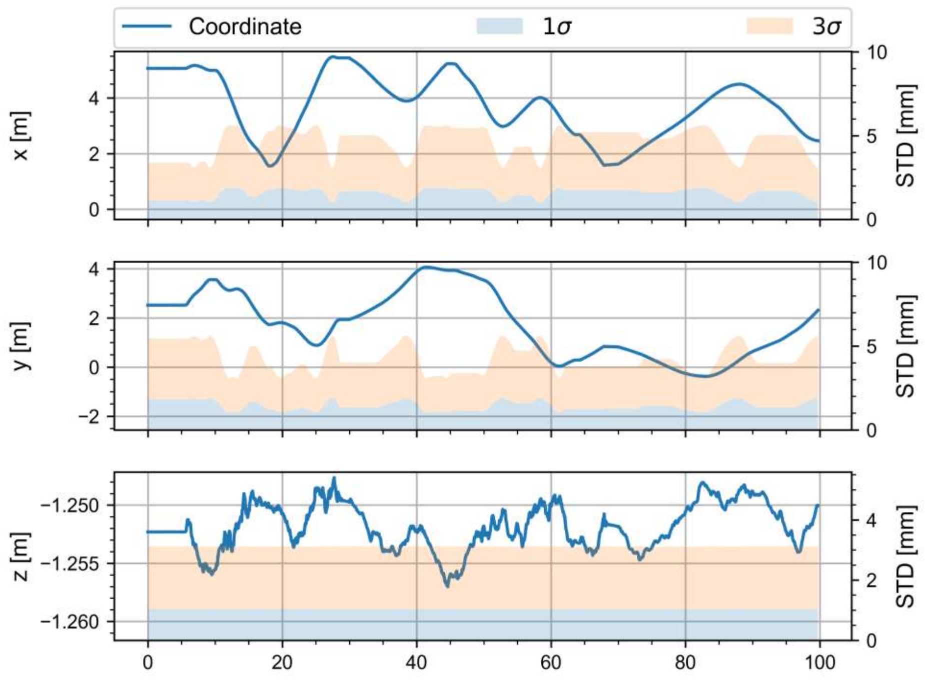

Figure 7.2: Example of the positional components of the Husky A200 poses for one driving experiment run

persist. The uncertainties of the z, θ and ψ components are very stable over time, as they are less sensitive to coordinate errors due to the geometry. It is important to note that the adjustment has a degree of freedom of only one (see section 5.2.4), making the estimation of the parameters and their uncertainties not very reliable.

The estimated parameter uncertainties highlight the potential and need to improve the measurement setup both in terms of eliminating the time synchronisation setup error and reducing the systematic effects caused by the turning hub eccentricity.

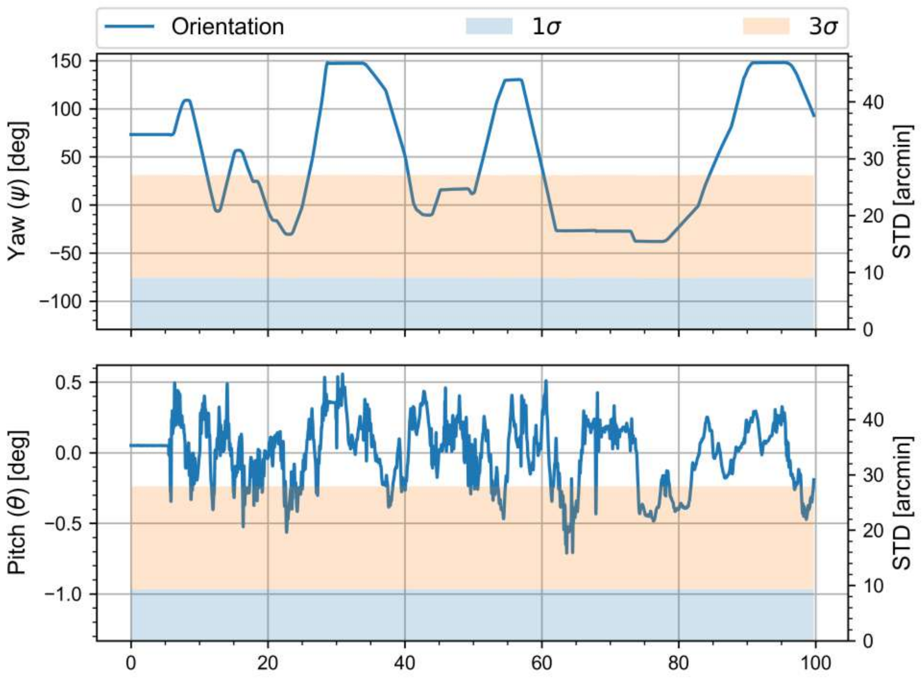

Figure 7.3: Example of the attitude components of the Husky A200 poses for one driving experiment run

7.6 Hyperparameter tuning

The time series of calculated pose data was used to calculate the required input data for the system identification, as is described in section 5.3.5. Thereby different values for the smoothing factor of the interpolation (see section 5.3.5) were used for creating different data sets. The other two hyperparameters α and T, pertaining to the sparse regression employed as part of the SINDY algorithm (see section 5.3.1), were tuned using individual grid searches using 5-fold cross-validation as is described in section 5.3.6. Figures 7.4 through to 7.7 show the grid search results for the different data sets. As metrics for evaluating the gridsearch the mean absolute error, the coefficient of determination (also referred to as R2 score), the model complexity and the R2 score per model complexity were chosen and are plotted in the figures. The mean absolute error was chosen in accordance with the sparsity promoting principle of SINDY. It is less sensitive to outliers which otherwise could potentially skew the results towards less restrictive hyperparameterization with more model terms not representative of the true system dynamics. The R2 score describes how much of the variations in the data were captured by the model. It is defined as

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \tag{7.1}$$

with

$$SS_{\text{res}} = \sum_{i=1}^{N} (f(x_i) - y_i)^2 \tag{7.2}$$

$$SS_{\text{tot}} = \sum_{i=1}^{N} (y_i - \bar{y})^2 \tag{7.3}$$

whereby $\bar{y}$ denotes the average of all y_i. The best possible R2 score is 1 with 0 denoting no predictive power.

Based on the different metrics multiple models from each data set were selected to be evaluated. The goal of the selection was to pick models with a good balance between predictive performance and model complexity. Ultimately 14 models were selected, which are highlighted in the grid search result plots, covering a large span of model complexities of 12 to 60 terms. With exception of the extreme cases of either none or all potential model terms being active, the progression of the different performance metrics is rather smooth with only small differences between neighbouring parametrisations. The parametrisation of $\alpha = 50$ and $T = 3$, resulting in model with fewer than 10 terms, was so restrictive that no terms for the equation for $\dot{x}^b$ were estimated regardless of which smoothing parameter was used. Those models were therefore not further investigated.

The examined hyperparameter space was broad enough that some parametrization resulted in either all terms available in the feature library (see section 5.3.1) being active or all terms being eliminated, as can be seen in the plots. Furthermore, none of the selected models resulted from hyperparameter values that are at the edge of the tested parameter space. Thus the selected models are around a perceived optimum in the hyperparameter space.

It is important to note that the values of the different gridsearch results pertaining to different smoothing factor values can not be compared amongst each other as the different smoothing parameters altered the target of the regression for both the training of the models as well as for their evaluation during the gridsearch, as is eluded to in section 5.3.6. However, it is notable that the hyperparameters $\alpha = 30$ and $T = 1$ resulted in promising gridsearch results for all evaluated smoothing parameters and the associated models were all selected to be investigated.

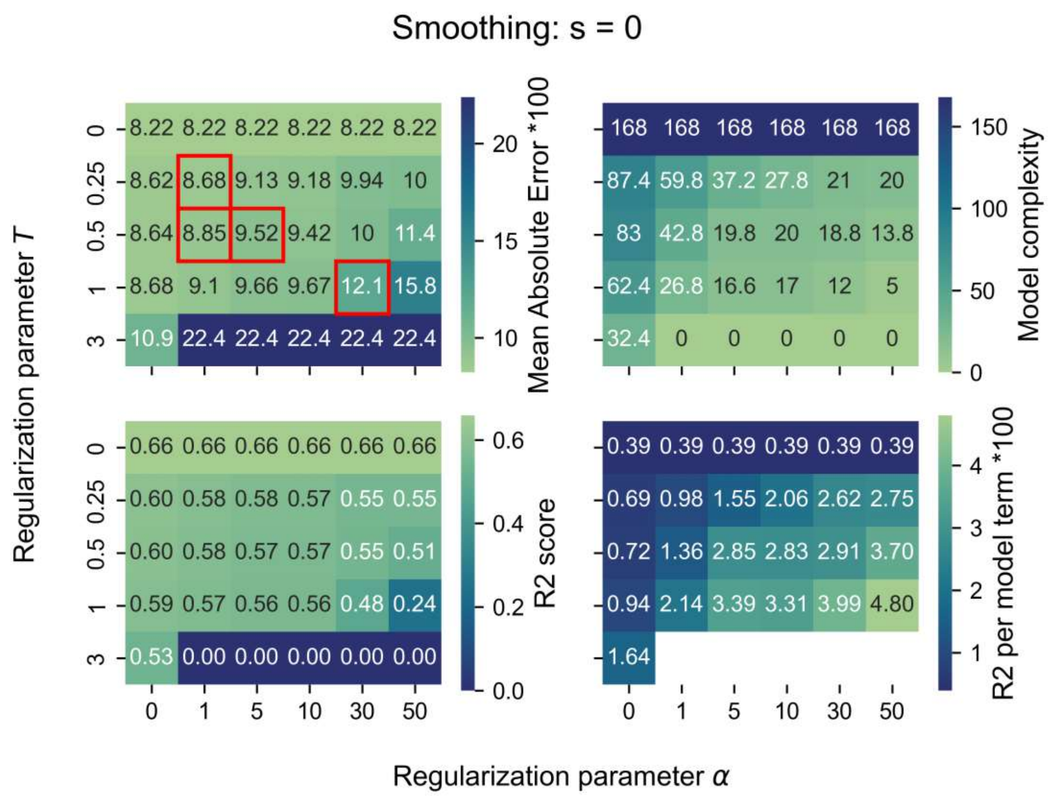

Figure 7.4: Grid search result for the data set created using a smoothing factor of $s = 0$

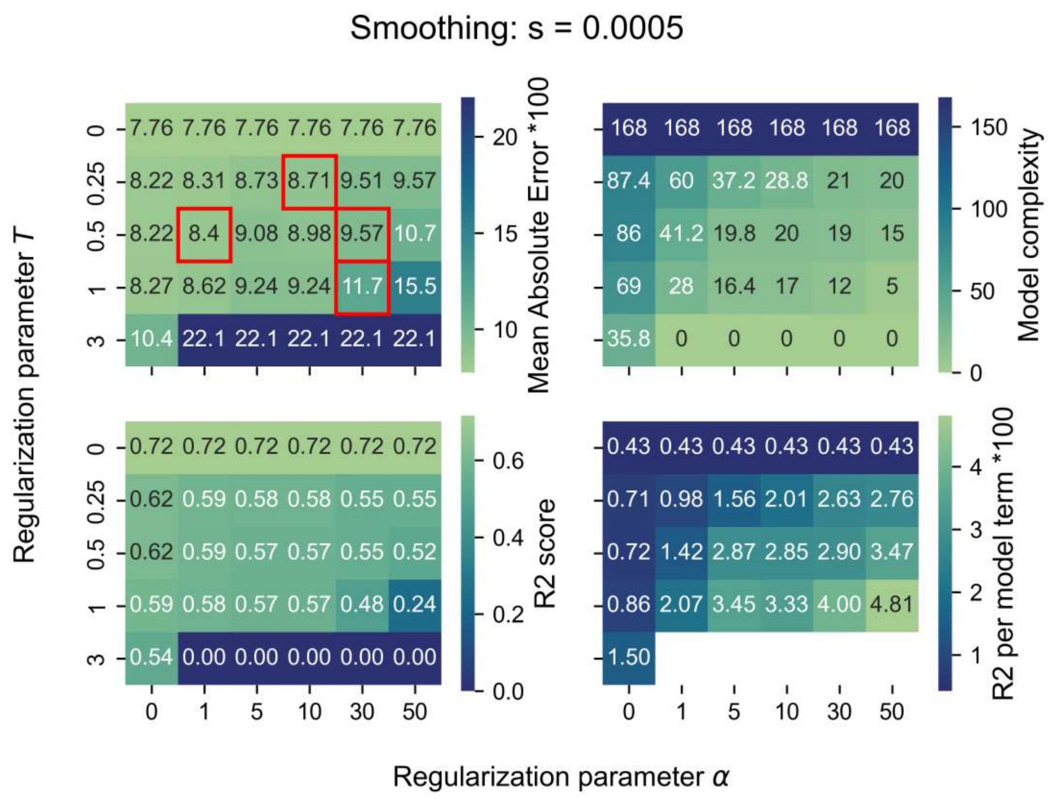

Figure 7.5: Grid search result for the data set created using a smoothing factor of $s = 0.0005$

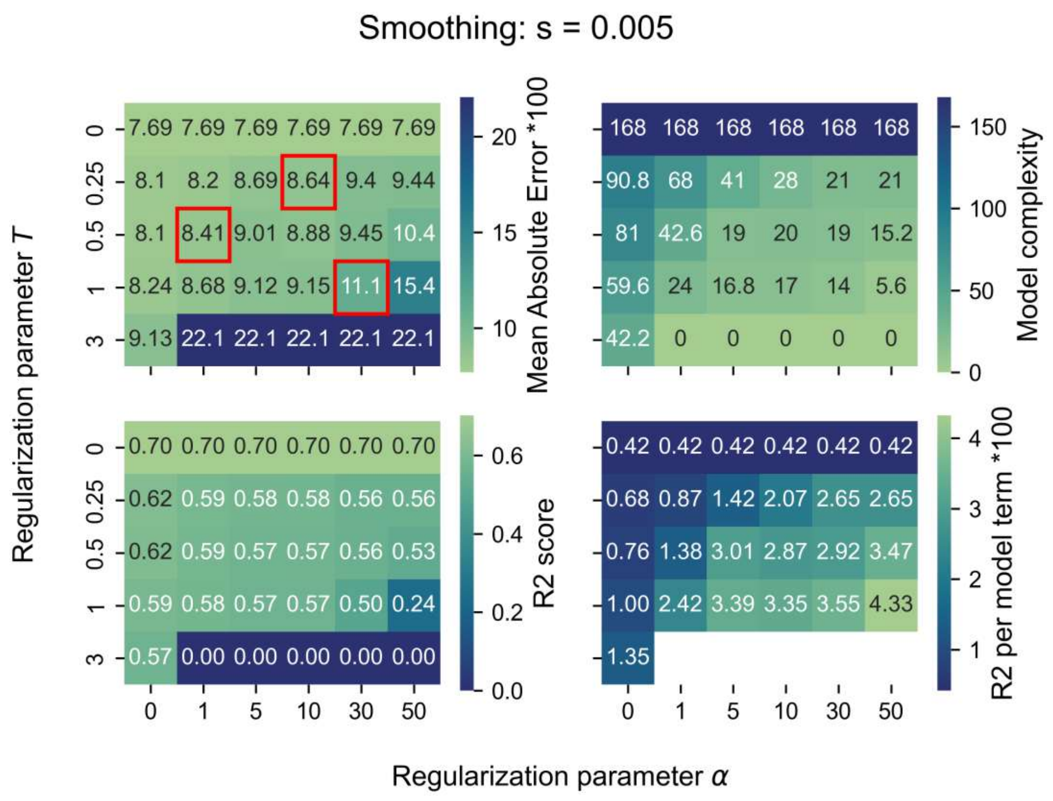

Figure 7.6: Grid search result for the data set created using a smoothing factor of $s = 0.005$

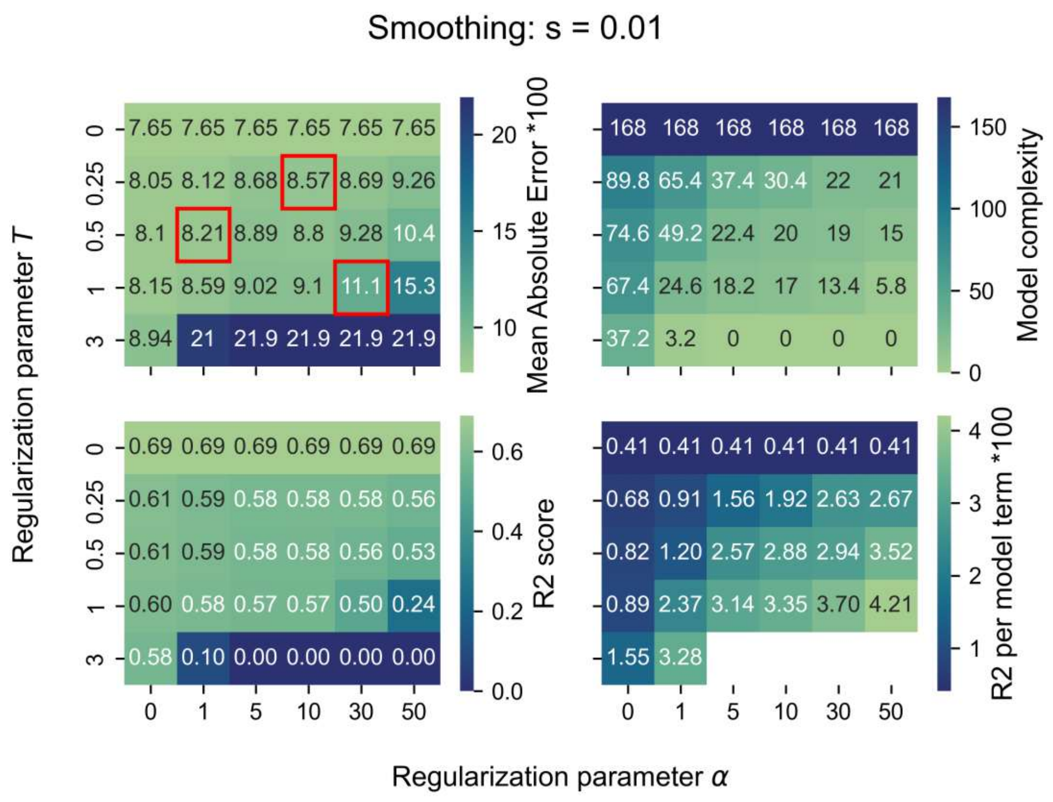

Figure 7.7: Grid search result for the data set created using a smoothing factor of $s = 0.01$

7.7 Model comparison

The 14 models which were selected based on the gridsearch results, as is described in the previous section, were evaluated based on their predictive performance with regard to the local reference frame, as is described in section 5.3.8. For this the three driving experiment runs reserved as testing data (see table 6.4), were partitioned into 126 5 second integration windows with 50% overlap. The resulting estimated standard deviation of the position and heading estimation as a function of integration length for all evaluated models are depicted figure 7.8. As can be seen despite there being a trend of more complex models exhibiting better predictive performance, the models with the most terms do not perform the best. This reinforces the validity of the use of sparsity promoting techniques for the system identification as it avoids overfitting.

All of the 14 evaluated models did not exhibit any active terms in the equation for $\dot{y}^b$. This could have multiple reasons amongst which are that the movements along this axis are not predictable given the available input data. Potential small variations in the driving surface properties or variations in tire temperature and dust accumulation on the contact patches of the tire could have led to varying driving dynamics that presented themselves disproportionally in the side-to-side movements of the vehicle. Figure 6.4 depicts the components of the body state vector components for one of the driving experiment runs. While $\dot{x}^b$ and $\dot{\psi}$

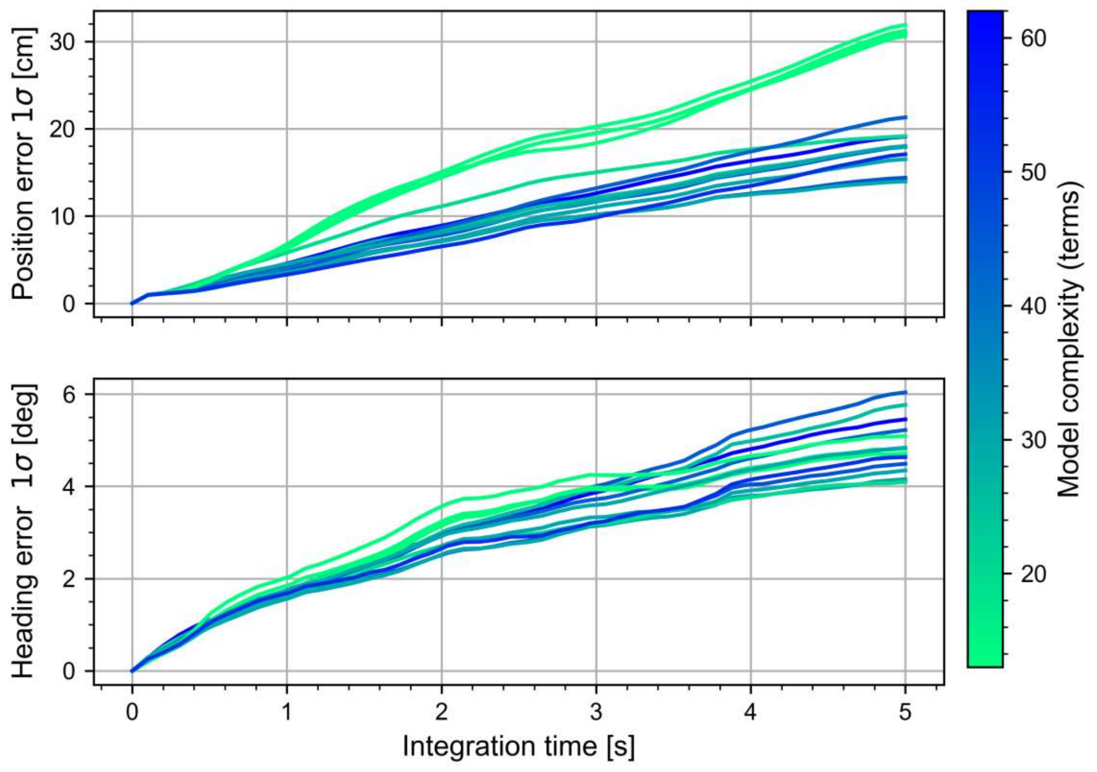

Figure 7.8: Estimates standard deviation of the position and heading estimation of the different models as a function of integration time based on the prediction errors on the testing data set

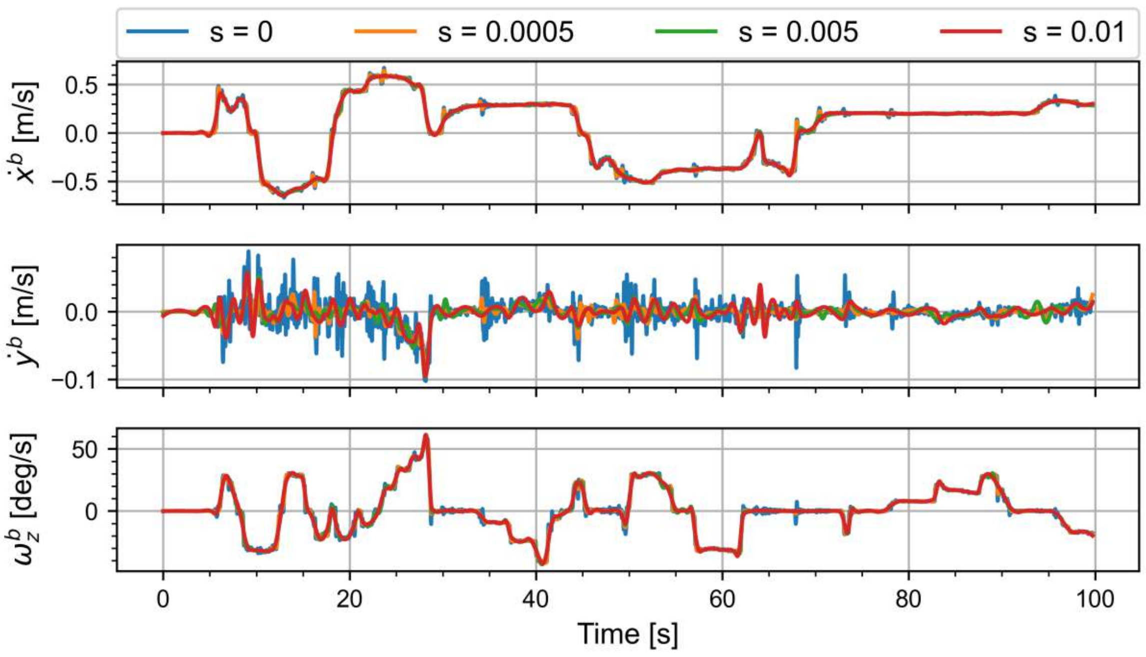

Figure 7.9: Body frame state vector components over time of one driving experiment run for different smoothing values

exhibit a strong signal with gradual changes, $\dot{y}^b$ has rapid sign changes and appears to be more random and noise-like. This is corroborated by the fact that the different smoothing parameters greatly alter the course of the function, indicating a poor signal to noise ratio. Another possibility is that the measured data was not of sufficient quality, either due to systematic effects, the pose component uncertainties or the measurement frequency, to infer this aspect of the dynamics.

While the examined models estimate the rate of change of $\dot{y}^b$ to be zero, some use this quantity as input for their other equations. This does not disqualify those models as values for $\dot{y}^b$ can still be updated in the context of using the models for a state estimation via Kalman filter. Furthermore, the value $\dot{y}^b$ may simply be initialized to 0 for solving initial value problems.

Figure 7.10 depicts the standard deviation of the position and heading estimation after a five second integration period as a function of model complexity. Furthermore, the data points are color coded to show which value for the smoothing factor s was used in the training of the model. The results show that a the smoothing of the pose component interpolations led to an improved model performance, with the smallest smoothing value for s, except for 0, performing the best. The especially bad heading prediction performance of the models with more complexity that were trained on the unsmoothed data strongly suggest that the pose data contains some effects, such as perhaps vibrations or flexing of the physical structure etc., that are not part of the desired driving dynamics of the Husky A200. It can also be seen that the models estimated with very restrictive hyperparameters, resulting in only 13 terms being active, suffer significant in terms of prediction performance when compared to more complex models. However, it can also be seen that too loose hyperparameters give the model to much capacity and lead to overfitting and in turn worse predictive performance with regard to the vehicle trajectory in the local reference frame. Models with around 30 terms exhibit both

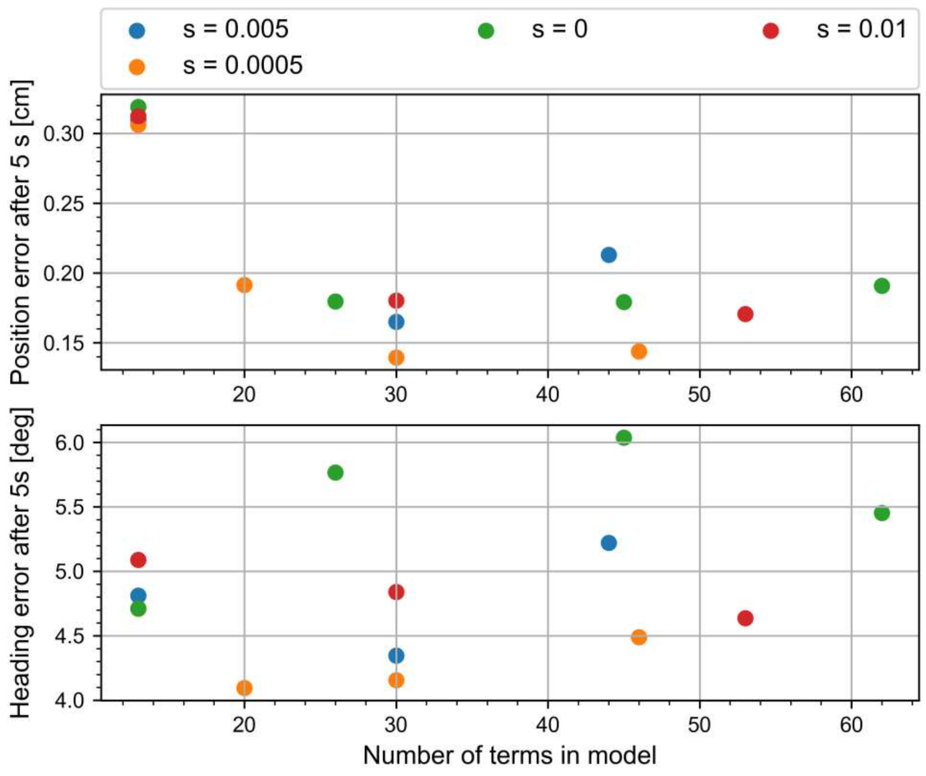

Figure 7.10: Comparison of the multi-step prediction uncertainties after a five second integration period

good performance and a reasonable model complexity. These results highlight the need for thorough hyperparameter tuning during the system identification process with the SINDY algorithm.

Table 7.6 lists the 14 evaluated models along with their complexity and the associated used hyperparameters ordered according to their position prediction accuracy after a five second integration duration. Models 11 trough to 14 enable an easy comparison of the influence of the smoothing parameter as the other hyperparameters are the same for those models. The model trained on unsmoothed data performs the worst, with the model based on very lightly smoothed training data with $s = 0.0005$ performing the best. This value for the smoothing parameter is also presented in the models 1 and 2, beating models with other hyperparameterisations with comparable or higher model complexity.

Overall the model performance as measured by the multi-step prediction in a local reference frame are promising. A standard deviation of 14 cm after 5 second integration period strongly suggests that the obtained model would be beneficial in use-cases such as state estimation with for example imprecise GNSS receivers. In this context of compensating intermittent loss of GNSS signals the model may also be beneficial.

Table 7.6: List of evaluated models along with their respective hyperparameter, model complexity and position and heading errors after 5 seconds

	Hyperparameters			Complexity	Position error after 5 s [cm]	Heading error after 5s [deg]
	s	α	T			
1	0.0005	10	0.25	30	14.0	4.2
2	0.0005	1	0.5	46	14.4	4.5
3	0.005	10	0.25	30	16.5	4.3
4	0.01	1	0.5	53	17.1	4.6
5	0	1	0.5	45	17.9	6.0
6	0	5	0.5	26	18.0	5.8
7	0.01	10	0.25	30	18.0	4.8
8	0	1	0.25	62	19.1	5.5
9	0.0005	30	0.5	20	19.1	4.1
10	0.005	1	0.5	44	21.3	5.2
11	0.0005	30	1	13	30.6	4.7
12	0.005	30	1	13	31.0	4.8
13	0.01	30	1	13	31.2	5.1
14	0	30	1	13	31.9	4.7

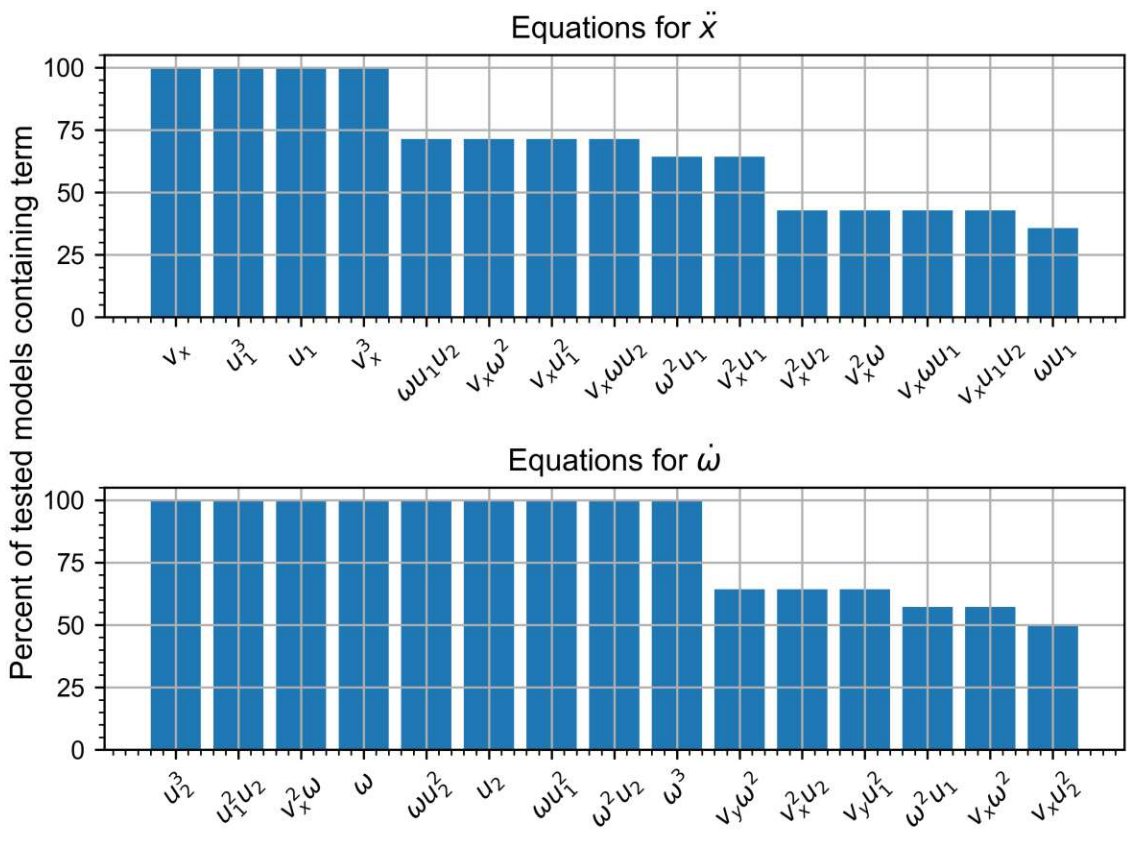

Figure 7.11: Most prevalent terms over all evaluated models and the percentage with which they appear in them

7.8 Model interpretation

The core idea of the SINDY algorithm is to identify compact, interpretable and parsimonious models of dynamical systems (Brunton et al., 2016a). Therefore it was attempted to ascribe the identified model terms a meaning or cause. For this, the best performing model with 13 terms (number 11 in table 7.6) and the best performing model overall (number 1 in the table), as measured by the point position error, were used. The model number 11 is referred to as model A and the model with number 1 is referred to as model B for the purposes of this discussion.

Model A has the equations

$$\dot{v}_x = -4.812\, v_x + 4.356\, u_1 + 3.065\, v_x^3 - 2.204\, u_1^3 \tag{7.4}$$

$$\dot{v}_y = 0.000 \tag{7.5}$$

$$\begin{aligned}\dot{\omega} = &-6.551\,\omega + 7.473\, u_2 + 2.978\, v_x^2\omega + 2.914\,\omega^3 - 4.298\,\omega^2 u_2 - 0.720\,\omega u_1^2 \\ &+ 2.654\,\omega u_2^2 - 2.799\, u_1^2 u_2 - 1.856\, u_2^3\end{aligned} \tag{7.6}$$

while model B has the equations

$$\begin{aligned}\dot{v}_x = &-7.770\, v_x + 7.190\, u_1 - 0.369\, v_x\omega + 0.312\,\omega u_1 + 6.780\, v_x^3 - 6.357\, v_x^2 u_1 \\ &+ 7.730\, v_x\omega^2 - 5.104\, v_x\omega u_2 + 3.459\, v_x u_1^2 - 3.752\,\omega^2 u_1 + 1.125\,\omega u_1 u_2 \\ &- 3.129\, u_1^3 - 0.435\, u_1 u_2^2\end{aligned} \tag{7.7}$$

$$\dot{v}_y = 0.000 \tag{7.8}$$

$$\begin{aligned}\dot{\omega} = &-6.951\,\omega + 7.900\, u_2 + 7.690\, v_x^2\omega - 3.251\, v_x^2 u_2 + 1.394\, v_x\omega^2 \\ &- 0.817\, v_x u_1 u_2 - 0.907\, v_x u_2^2 + 2.315\, v_y\omega^2 - 4.976\, v_y u_1^2 + 2.896\,\omega^3 \\ &- 1.409\,\omega^2 u_1 - 4.349\,\omega^2 u_2 - 3.143\,\omega u_1^2 + 3.226\,\omega u_2^2 - 0.795\, u_1^2 u_2 \\ &+ 0.838\, u_1 u_2^2 - 2.322\, u_2^3\end{aligned} \tag{7.9}$$

whereby u_1 and u_2 denote the throttle and steering input respectively.

The equation for $\dot{v}_x^b$ from model A is very simple. The terms v_x and v_x^3 can be ascribed to friction forces, stemming from motor resistance, rolling resistance from tire deformations etc., The terms u_1 and u_1^3 represent the throttle response. Model B also exhibits those four terms while also exhibiting a cubic term u_1^3 for the throttle response. Furthermore, model B introduces dependencies of the forwards acceleration on both angular velocity and steering input. Those dependencies are present in the true driving dynamics of the Husky A200. In figure 6.3 a limitation of the maximum forwards velocity based on the angular velocity can be seen. Therefore the forward acceleration needs to be lower when turning so that the equilibrium of propulsion and friction forces is at a lower speed, depending on the turning speed. A dependence on the steering input is also expressed in the terms $u_1u_2^2$. Such a dependency is to be expected as the motors need to create differential thrust and therefore not all of their potential torque is used for linear acceleration. Of special interest are the terms $-0.369\,v_x\omega + 0.312\,\omega u_1$, $-5.104\,v_x\omega u_2$ and $+1.125\,\omega u_1 u_2$ as they introduce asymmetries in the dynamics as the acceleration is dependent on which direction the vehicle turns and in which direction the steering put is. However, due to strong correlations between the linear and angular velocities and their associated control input, described in the next section, those asymmetries may largely cancel out. The need for the system identification to use quadratic terms of certain inputs in order to maintain symmetry showcases the potential benefits of augmenting the feature library with the absolute values of the input values. Likewise, the introduction of the terms like $a \cdot |a|$ would allow to use quadratic terms that maintain the sign of the input values. The tuning of the feature library therefore requires additional research.

It appears that the rotational component of the driving dynamics is more complex than the translational component as both model exhibit considerably more terms for the $\ddot{\psi}$ equation than for the $\ddot{x}$ equation. This is expected as turning requires the overcoming of traction, resulting in complex tire-ground interactions especially compared to driving in a straight line where virtually no slippage occurs.

Regarding the angular acceleration model A and B both exhibit terms likely representing friction forces analogous to that of the linear case: ω and ω^3. The steering input response is modeled with many terms even in the simpler model A, introducing dependencies on the current rotation speed as well as the linear velocity despite the more restrictive hyperparameterisation. Model B exhibits additional cross terms for example a product term with the linear velocity and the steering input. Overall the interpretation of the terms in the equations for $\ddot{psi}$ proves to be more difficult which is to be expected as the driving dynamics are so difficult that analytical modeling remains challenging for skid-steered robots. The terms $v_y\omega$ and $v_y u_1$ are of special interest as they introduce a dependency of the lateral movement of the vehicle, despite the rate of change of this state variable being estimated as zero by all models. Further investigations into whether this is overfitting to the data or truly a part of the underlying driving dynamics need to be conducted.

In figure 7.11 the 15 most prevalent terms in all elevated models for the equations for both $\ddot{x}$ and $\ddot{\psi}$ are presented, ordered by the percent of models they appear in. For the linear acceleration four terms relating to the throttle response and friction forces were discovered by all examined models, regardless of the hyperparameterisation. Models with less restrictive regularizations introduced dependencies on the turning speed and steering input. For the angular acceleration equation all models identified more terms, highlighting the complex dynamics, exhibiting terms for linear velocity and throttle input. A significant amount of models exhibit a dependence of the lateral movement of the vehicle.

7.9 Least squares model adjustment

As described in section 5.3.9 it was attempted to refine the estimated model coefficients using a least squares adjustment (see section 2.4) after the relevant terms were identified by SINDY. However, early experiments revealed that the resulting equation system had insufficient separability of parameters based on the data, high resulting correlations and poor conditioning. Therefore the coefficients were not updated using the least squares adjustment

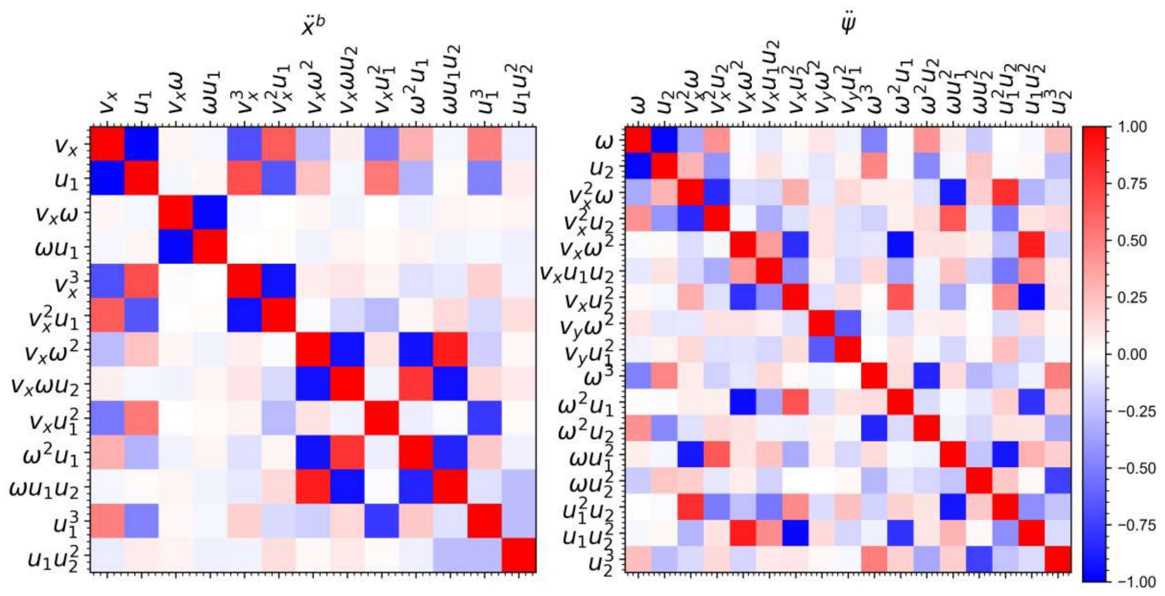

Figure 7.12: Correlation matrices of the model coefficients resulting from a least squares adjustment

estimates. This issue showcases the benefit of using the Ridge regression which can estimate parameters in cases with strong correlation between parameters.

The correlation matrix of the adjustment for model number 1 in table 7.6 using the simplest functional model introduced in section 5.3.9 is depicted in figure 7.12. The correlation matrix was split for the different equations for better readability. There were no correlations between the parameters of different equations using the simplest functional model. It can be seen that there are some very strong correlations between certain parameters. Table 7.7 lists the terms with 15 strongest correlations. The strongest correlations are between the term with the forwards velocity of the vehicle and the throttle input and the rotation rate and the steering input. While it is clear that those quantities correlate during driving they are independent and important for the system identification. The other strong correlations listed are just variations of terms that contain the aforementioned strongly correlated terms.

The refinement of the model parameters using a least squares adjustment is highly desirable due to the possibility to incorporating the measurement uncertainties as well as for the covariance information for the estimated parameters. The issue of the correlation could be circumvented with targeted subsampling of the collected data so that the amount of data representing driving states with low acceleration, both linear and angular, is limited. Furthermore it might be possible to design input sequences that are targeted for producing data that is well suited for system identification by sampling the phase-input-space as uniformly and completely and avoiding correlations. This was also envisioned by the authors of Brunton et al. (2016b), writing "[...] likely possible to design input sequences that optimally probe complex systems to extract high-value information that will be useful to characterize the system". The possibilities of targeted data collection or subsampling for obtaining data well suited for system identification using SINDY and subsequent least square adjustments needs additional research. Furthermore, an augmentation of the used feature library with

Table 7.7: The 15 strongest correlation between parameters estimated using a least squares adjustment

	Correlation	Term A	Term B
1	-0.99	u_1	v_x
2	-0.98	u_2	ω
3	-0.98	ωu_1	$v_x\omega$
4	-0.97	$u_1 u_2^2$	$v_x u_2^2$
5	-0.96	$\omega^2 u_1$	$v_x\omega^2$
6	-0.93	$\omega u_1 u_2$	$v_x\omega u_2$
7	-0.93	$v_x\omega u_2$	$v_x\omega^2$
8	-0.93	$v_x^2 u_1$	v_x^3
9	-0.92	$\omega^2 u_1$	$v_x\omega^2$
10	-0.91	$u_1^2 u_2$	ωu_1^2
11	-0.90	ωu_1^2	$v_x^2\omega$
12	0.89	$u_1 u_2^2$	$v_x\omega^2$
13	0.88	$\omega u_1 u_2$	$v_x\omega^2$
14	-0.86	$\omega^2 u_2$	ω^3
15	-0.86	$\omega u_1 u_2$	$\omega^2 u_1$

absolute values of the inputs as well as quadratic terms retaining the sign of the input value, as discussed in the previous section may reduce restrictions imposed by the feature library and also reduce correlations.

8. Conclusion

The aim of this thesis was to evaluate the suitability of system identification in the form of the SINDY algorithm for estimating a model of the driving dynamics of the Husky A200 UGV based on geodetic measurements. This resulted in the three major tasks of creating a suitable measurement setup, establishing a preprocessing pipeline for obtaining the input data for the system identification task and ultimately conducting and evaluating the system identification.

The measurement setup was realized using two laser trackers measuring two reference points on the vehicle represented as CCRs in specifically designed and manufactured motorized turning hubs in order to maintain a line of sight. The two laser trackers were synchronized using a common trigger signal generated using a Raspberry Pi which in turn was synchronized with the Husky's onboard computer by synchronizing their respective system clocks using GNSS receivers. The different system components were integrated into measurement setup using ROS. The resulting measurement setup was capable of collected the needed data with adequate accuracy for successful system identification of the driving dynamics. The shortcomings of the measurement setup were an eccentricity of the motorized turning hubs and a configuration error pertaining to the time synchronisation which could be compensated during the analysis. Possible improvements are among others the use of a RS-422 signal for interference robust transmission of the laser tracker trigger cable and the use of 360° laser tracker prisms for eliminating the need for the motorized turning hubs.

In order to obtain the input data for the system identification the time series of prism coordinates were time synchronised using a linear clock model for the internal laser tracker clocks. Subsequently the poses of the Husky A200 were calculated in a two-step progress utilising an overparameterisation to calculate the 2D pose which was used for the linearization in the context of a least square estimation for a pose with three translation and two rotation parameters. Those pose components were interpolated using cubic splines while being weighted by the inverse of their variance obtained from the adjustment. A smoothing of the interpolation was shown to be beneficial for the performance of the identified model. For this different values for the smoothing parameter were tested. The spline interpolation was analytically differentiated to obtain the linear and angular velocities needed for the state vector representation of the system state.

The system identification of the driving dynamics was successfully conducted using the SINDY algorithm utilizing the integral notation which eliminated the need of differentiating the pose interpolation a second time. The state vector of the system consisting of the pose and its first derivative was truncated under the sensible assumption that the driving dynamics are homogeneous and isotropic, eliminating the possibility of a dependence on the position or heading to be included in the models. The driving dynamics were then estimated with regard to the vehicle's body frame, greatly simplifying the complexity of the model to be estimated and aiding the sparse identification as the dynamics to be identified are only sparse with regard to appropriate state vector bases. The tuning of the hyperparameters proved to be of critical importance for generating well performing yet interpretable and compact models that avoid overfitting. In this thesis this was accomplished using a grid search with 5-fold cross-validation.

The best performing model resulted in a point position estimation uncertainty of 14 cm and a heading estimation uncertainty of 4.2° after a five second integration window. This is a promising result, suggesting beneficial uses of the model in the context of state estimation using the Kalman filter, especially when utilizing low-cost GNSS receivers. Furthermore, two models have been interpreted, trying to ascribe meaning to identified model terms, as the core idea and strength of the SINDY algorithm is identifying interpretable and parsimonious models.

The generalizability of the final model resulting from this thesis is strongly limited. As the tire-ground interaction is of critical importance for the driving dynamics of skid-steered robots, the predictive power of the model in all likelihood is greatly diminished for different kind of surfaces. Furthermore, the model assumes driving on horizontal ground. The different weight distribution on the tires in slanted driving positions would also likely influence the dynamics and therefore reduce the validity of the model and its predictions.

An attempted least squares adjustment for improving the coefficients of the identified model terms revealed strong correlation between parameter, highlighting the benefit of the employed Ridge regression during the SINDY process. Potential solutions in the form of targeted subsampling of the training data as well as generating control input signals for intelligent sampling of the phase-input-space have been proposed.

Overall the application of SINDY for the identification of the driving dynamics of the Husky A200 using geodetic data shows promising results with many areas of potential improvements. Besides improvements in the measurement setup they pertain to more extensive hyperparameter tuning, testing different feature libraries, better numerical evaluation of the resulting model and the aforementioned methods for sampling training data for improved parameter separability.

8.1 Outlook

A logical next step is the utilization of the estimated model in a Kalman filter state estimation, possibly by simulating GNSS data based on the laser tracker measurements with different measurement frequencies and precision levels. A direct comparison of the chosen approach using SINDY for the system identification of the drivings dynamics of the Husky A200 with other approaches such as neural networks in this context would be very insightful.

Another very exciting prospect for further development is the extension to more surface types with different friction coefficients. Data from multiple different driving surfaces could be collected and used for system identification using a state vector augmented by one or more non-changing state variables describing the surface characteristics. Here, the enhanced generalizability of the SINDY approach, when compared to black box methods such as neural networks (Brunton et al., 2016b), could lead to better performance on a wide range of surfaces based on measurements on only a few surface types. When used in a state estimation algorithm such as the Kalman Filter the variables describing the surface could be updated using measurement data based on the contradiction of the model predictions. Similarly, the model could be extended to non-level driving surfaces.

Bibliography

Brandstätter, M. (2022). *Development of an autonomously operating stop-and-go multi-sensor system and pose estimation evaluation based on 3D laser scans.*

Brunton, S. L., & Kutz, J. N. (2019). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press. https://doi.org/10.1017/9781108380690

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016a). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences - PNAS*, *113*(15), 3932–3937. https://doi.org/10.1073/pnas.1517384113

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016b). Sparse Identification of Nonlinear Dynamics with Control (SINDYc)**SLB acknowledges support from the U.S. Air Force Center of Excellence on Nature Inspired Flight Technologies and Ideas (FA9550-14-1-0398). JLP thanks Bill and Melinda Gates for their active support of the Institute of Disease Modeling and their sponsorship through the Global Good Fund. JNK acknowledges support from the U.S. Air Force Office of Scientific Research (FA9550-09-0174). *IFAC-PapersOnLine*, *49*(18), 710–715. https://doi.org/10.1016/j.ifacol.2016.10.249

Chartrand, R. (2011). Numerical Differentiation of Noisy, Nonsmooth Data. *ISRN Applied Mathematics*, *2011*, e164564. https://doi.org/10.5402/2011/164564

Clearpath Robotics. (2016). Husky A200 data sheet.

Croston, B. (n.d.). *RPi.GPIO: A module to control Raspberry Pi GPIO channels* (Version 0.7.1). Retrieved March 19, 2023, from http://sourceforge.net/projects/raspberry-gpio-python/

de Silva, B., Champion, K., Quade, M., Loiseau, J.-C., Kutz, J., & Brunton, S. (2020a). PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, *5*(49), 2104. https://doi.org/10.21105/joss.02104

Dogru, S., & Marques, L. (2021). An improved kinematic model for skid-steered wheeled platforms. *Autonomous robots*, *45*(2), 229–243. https://doi.org/10.1007/s10514-020-09959-0

Dormand, J. R., & Prince, P. J. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, *6*(1), 19–26. https://doi.org/10.1016/0771-050X(80)90013-3

Fasel, U., Kaiser, E., Kutz, J. N., Brunton, B. W., & Brunton, S. L. (2021, August 30). *SINDy with Control: A Tutorial*. Retrieved April 10, 2022, from http://arxiv.org/abs/2108.13404

Ferguson, M. (2022, May). Rosserial. https://github.com/ros-drivers/rosserial/tree/noetic-devel

Groves, P. (2013). *Principles of GNSS, inertial, and multisensor integrated navigation systems, second edition*. Artech House. https://books.google.at/books?id=t94fAgAAQBAJ

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., . . . Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hexagon Metrology. (n.d.). Absolute Tracker systems and accessories. url:%20https://go%20.hexagonmi.com/l/49752/2017-07-05/5t7pmw/49752/1654666941ogPvWeyz/Hexagon_MI_Tra%20cker_Catalogue_3.0_210x210_en_WEB.pdf

Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, *12*(1), 55–67. https://doi.org/10.2307/1267351

Hui, C. K., Luo, T. X., Lai, W. W., & Chang, R. K. (2022). GPR mapping with mobile mapping sensing and tracking technologies. *Tunnelling and Underground Space Technology*, *122*, 104362. https://doi.org/10.1016/j.tust.2022.104362

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Joeckel, R., Stober, M., & Huep, W. (2008). *Elektronische Entfernungs- und Richtungsmessung und ihre Integration in aktuelle Positionierungsverfahren* (5., neubearb. und erw. Aufl). Wichmann.

Kaiser, E., Kutz, J. N., & Brunton, S. L. (2018). Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society. A, Mathematical, physical, and engineering sciences*, *474*(2219), 20180335–20180335. https://doi.org/10.1098/rspa.2018.0335

Kaptanoglu, A. A., de Silva, B. M., Fasel, U., Kaheman, K., Goldschmidt, A. J., Callaham, J., Delahunt, C. B., Nicolaou, Z. G., Champion, K., Loiseau, J.-C., Kutz, J. N., & Brunton, S. L. (2022). PySINDy: A comprehensive Python package for robust sparse system identification. *Journal of Open Source Software*, *7*(69), 3994. https://doi.org/10.21105/joss.03994

Krishna, N., George, A., John, A., & Sudheer, A. (2017). Controller Design for a Skid-Steered Robot and Mapping for Surveillance Applications. *Proceedings of the Advances in Robotics*, 1–7. https://doi.org/10.1145/3132446.3134887

Lehtola, V., Nüchter, A., & Goulette, F. (Eds.). (2022). *Advances in Mobile Mapping Technologies*. MDPI - Multidisciplinary Digital Publishing Institute. https://doi.org/10.3390/books978-3-0365-3489-3
Accepted: 2022-05-06T11:33:02Z

Leica Geosystems. (2003). Leica Laser Tracker LT D800 Manual.

Leica Geosystems. (2008). emScon 3 Manual.

Leica Geosystems. (2013). emScon 3 Programmers Manual.

Leica Geosystems. (2020). Leica Absolute Tracker AT930-AT960 User Manual v2.2.0.

Lettner, R. (2022). *Uncertainty assessment of TLS distance deviations evoked by material and incidence angle*. https://doi.org/10.34726/hss.2023.107104

Li, R. (2011). Mobile Mapping: An Emerging Technology for Spatial Data Acquisition. In *The Map Reader* (pp. 170–177). https://doi.org/10.1002/9780470979587.ch24

Ljung, L. (1998). System Identification. In A. Procházka, J. Uhlíř, P. W. J. Rayner, & N. G. Kingsbury (Eds.), *Signal Analysis and Prediction* (pp. 163–173). Birkhäuser. https://doi.org/10.1007/978-1-4612-1768-8_11

Ltd, R. P. (n.d.). *Raspberry Pi 4 Model B specifications*. Raspberry Pi. Retrieved March 21, 2023, from https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, Am., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., . . . Scopatz, A. (2017). SymPy: Symbolic computing in Python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

New River Kinematics. (2020). *SpatialAnalyzer User Manual*. Retrieved March 23, 2023, from :%20https://spatialanaly%20zer.com/ftp/SA/Install/Documentation/SA%5C%20User%5C%20Manual_2020.07.20.pdf

Niemeier, W. (2008, September 25). *Ausgleichungsrechnung: Statistische Auswertemethoden*. De Gruyter. https://doi.org/10.1515/9783110206784

Ordonez, C., Gupta, N., Reese, B., Seegmiller, N., Kelly, A., & Collins, E. G. (2017). Learning of skid-steered kinematic and dynamic models for motion planning. *Robotics and Autonomous Systems*, *95*, 207–221. https://doi.org/10.1016/j.robot.2017.05.014

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Schaeffer, H. (2017). Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *473*(2197), 20160446. https://doi.org/10.1098/rspa.2016.0446

Schaeffer, H., & McCalla, S. G. (2017). Sparse model selection via integral terms. *Physical review. E*, *96*(2-1), 023302–023302. https://doi.org/10.1103/PhysRevE.96.023302

Schmidt, M., & Lipson, H. (2009). Distilling Free-Form Natural Laws from Experimental Data. *Science*, *324*(5923), 81–85. https://doi.org/10.1126/science.1165893

Shamah, B., Wagner, M. D., Moorehead, S., Teza, J., Wettergreen, D., & Whittaker, W. L. (2001). Steering and control of a passively articulated robot. *Sensor Fusion and Decentralized Control in Robotic Systems IV*, *4571*, 96–107. https://doi.org/10.1117/12.444150

Silva, B. M. de, Champion, K., Quade, M., Loiseau, J.-C., Kutz, J. N., & Brunton, S. L. (2020b). PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, *5*(49), 2104. https://doi.org/10.21105/joss.02104

Srikonda, S., Norris, W. R., Nottage, D., & Soylemezoglu, A. (2022). Deep Reinforcement Learning for Autonomous Dynamic Skid Steer Vehicle Trajectory Tracking. *Robotics (Basel)*, *11*(5), 95–. https://doi.org/10.3390/robotics11050095

Stanford Artificial Intelligence Laboratory et al. (2018, May 23). *Robotic operating system* (Version ROS Noetic Ninjemys). https://www.ros.org

Thalmann, T., & Neuner, H.-B. (2016). In-Field Calibrated Odometry for Skid-Steered Mobile Robots, 9. Retrieved March 21, 2023, from https://repositum.tuwien.at/handle/20.500.12708/43622

Accepted: 2022-08-02T14:43:26Z

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: A retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *73*(3), 273–282. https://doi.org/10.1111/j.1467-9868.2011.00771.x

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., . . . SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wang, H., Zhang, J., Yi, J., Song, D., Jayasuriya, S., & Liu, J. (2009). Modeling and motion stability analysis of skid-steered mobile robots. *2009 IEEE International Conference on Robotics and Automation*, 4112–4117. https://doi.org/10.1109/ROBOT.2009.5152342

Yi, J., Wang, H., Zhang, J., Song, D., Jayasuriya, S., & Liu, J. (2009). Kinematic Modeling and Analysis of Skid-Steered Mobile Robots With Applications to Low-Cost Inertial-Measurement-Unit-Based Motion Estimation. *IEEE Transactions on Robotics*, *25*(5), 1087–1097. https://doi.org/10.1109/TRO.2009.2026506

List of Figures

List of Tables