Technische Universität München
TUM School of Engineering and Design

Control of a Two-Wheeled Balancing Robot

Klaus Albert

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Dr.-Ing. Florian Holzapfel

Prüfer*innen der Dissertation:

1. Prof. Dr.-Ing. habil. Boris Lohmann
2. Prof. Dr.-Ing. habil. Ansgar Trächtler

Die Dissertation wurde am 13.01.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 07.05.2023 angenommen.

Abstract

Two-wheeled inverted pendulums (TWIPs) offer new possibilities for personal transport and filming, but also for indoor logistics. Especially for the last two tasks, where autonomous driving might be desired, stable and precise setpoint and tracking control for TWIPs is essential. Besides the given commercial applications, TWIPs continue to be used in research to test and verify new algorithms as they are a challenging test bench for control algorithms. In the last decades, several contributions have proposed numerous prototypes and models of TWIPs and used them to develop and test different control schemes.

This thesis presents a holistic concept where a TWIP is built and modeled, and algorithms to perform setpoint and trajectory tracking are developed and evaluated in simulations and experiments.

A TWIP prototype concept and its building process are presented with a strong focus on the desired real-time closed-loop control application. This design led to a robust and lightweight robot with an appropriate choice of sensors delivering high-quality measurement.

A comprehensive model of the TWIP is presented using the Lagrangian framework for modeling. The proposed approach is less error-prone and has a more well-rounded modeling procedure, as the electrical and mechanical systems are modeled jointly. Moreover, in the awareness of non-ideal sensors, a model for the onboard sensors is presented to provide the required congruence to the physical system for simulation-based observer tuning. Finally, a versatile analysis of the linearized models is provided, and state transformations are introduced to decouple the dynamics into smaller subsystems.

For trajectory generation, a method to compute an energy optimal trajectory offline is presented. The introduced optimization problem minimizes the input energy of the TWIP and not only the control input. Moreover, additional analyses are presented to compare the standard Runge-Kutta (RK) against variational integrator (VarInt) schemes for discretization.

A novel complete discrete-time control structure for setpoint and trajectory tracking is introduced to perform stable closed-loop control, even in the presence of significant control errors. Therefore, the closed-loop system's limited domain of attraction (DoA) is estimated by a quadratic Lyapunov function (QLF) and a limiting level set. Using linear matrix inequalities (LMIs), the maximum DoA is efficiently calculated via convex optimization.

An innovative state estimation method for the TWIP is proposed, which is able to incorporate delayed remote measurements properly. In particular, the introduced algorithm requires low computational power and little memory and thus can be executed on microcontrollers.

Last but not least, a new algorithm to estimate clock parameters online on a low-power microcontroller with limited memory is presented, which delivers the time information required by the introduced state estimation algorithm.

To conclude, this thesis presents methods, algorithms, and results connected to stable and precise setpoint and tracking control for TWIPs ranging from prototype development, modeling, trajectory generation, and feedback control to state and clock parameter estimation.

Contents

Chapter 1

Introduction

1.1 Motivation

The invention of the wheel around 4000 BCE (see Potts [91]) is clearly the foundation for almost all mobile vehicles in our modern time. Already 2000 BCE two-wheeled chariots used for transportation (see Kuznetsov [58]) eased people's daily life. Astonishingly, in 2001 an article by Heilemann [43] in a newspaper titled "Reinventing the Wheel" and 2003 a book with the same title has been published by Kemper [54]. What happened? Around 1990 Dean Kamen started with the development of the two-wheeled balancing wheelchair iBOT and in 1999 clinical trials began. Shortly thereafter in 2001, he presented the Segway: a two-wheeled, self-balancing personal transporter (see Kemper [54] or Wikipedia [114, 115]). In 2022, more than 20 years later, the enthusiasm faded, and the prognosticated global change in mobility using two-wheeled, self-balancing personal transport did not come. The early adopters, guided city tour providers, switched back from Segways to bicycles. In addition, the police and security staff in many nations realized, that in most cases they are slower than and not as agile as running and after the first enthusiastic trials with the Segway, they stopped their campaigns. Moreover, safety was a clear issue: For instance, famous accidents and falls happened, e.g. with US President George W. Bush or in 2015 with runner Usain Bolt. Usain Bolt has been knocked over by a cameraman who crashed with a Segway after he won gold at the world athletics championships (see ABC News [1]). Together with economic decisions, all this led to the end of production of the Segway in 2020.

Contrary to this trend, in 2019 the next generation of the iBOT has been released to the market (see Wikipedia [114]), showing that there still is a need for self-balancing personal mobility devices, but for a different group of customers. In addition, there are other applications besides wheelchairs, which have not been addressed by commercial products yet. Two-wheeled, self-balancing vehicles, also called two-wheeled inverted pendulums (TWIPs), might be used for autonomous surveillance applications on predefined tracks around or in buildings. Moreover, at movie sets, where the camera has to be at a specific position at a defined time with a particular speed an autonomous TWIP equipped with a camera offers new possibilities. It has to be assumed, that the crash in 2015 with Usain Bold would not have taken place if instead of a Segway an autonomous TWIP equipped with modern sensors would have been used, as the accident was clearly caused by a driving error of the cameraman. In addition, for intralogistics, TWIPs could bring added value. The iBOT can balance on two wheels or move in a stable mode with four

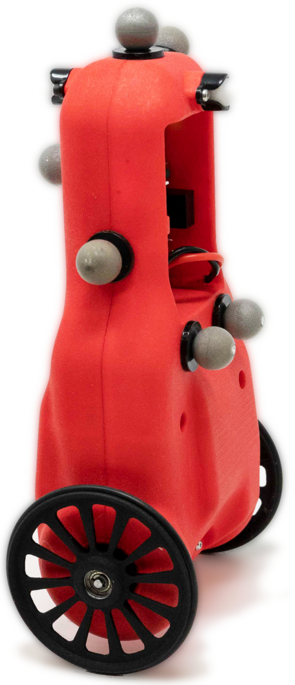


Figure 1.1: TWIP with markers for an optical tracking system

wheels and the same concept can be used in warehouses. Usually, if the goods have to be inserted into a high storage rack, the robot needs to have a heavy basement to avoid toppling over. As the TWIP maintains stability through balancing, such a high mass is not necessary, and thus less energy is required for motion. So it is conceivable, that the future belongs to transportation vehicles that are able to operate in an inherent stable mode on ground with three or four wheels as well as a stabilized, balancing mode on two wheels. Therefore, besides algorithms for stabilization, methods for precise setpoint and trajectory tracking are necessary.

Besides the given commercial applications, TWIPs as shown in Figure 1.1 continues to be used in research to test and verify new algorithms as they are a challenging test bench for control algorithms. Stein [101] stated, that the inverted pendulum on a cart has been motivated by the space race as control engineers can learn how to balance rockets based on this simple experimental setup. The same considerations apply to the TWIP, but in addition, the system is subject to nonholonomic constraints, making the control task even more difficult. Contrary to the TWIP, the inverted pendulum on a cart is a pure research test bench without practical application examples. Finally, the Segway and TWIPs use sensor setups quite similar to rockets. Therefore, the algorithms presented for TWIPs might be applied for the control of rockets and missiles or in other domains.

In the last decades, several contributions have proposed numerous prototypes and models of TWIPs and used them to develop and test different control schemes. Nevertheless, a complete solution that solves the problem of trajectory tracking has not been proposed yet and is a non-trivial task. Hence, developing a TWIP prototype to test novel control structures in real-time would contribute to improving these current solutions. Moreover, there is a demand for new algorithms to solve the aforementioned tasks like trajectory generation, stabilization, setpoint and trajectory tracking as well as state estimation.

1.2 Literature and Prototypes

Ongoing, during the last twenty years several TWIPs have been built and a large number of papers for design and control have been published. The two surveys by Chan et al. [15] from 2013 and Romlay et al. [94] from 2019 provide a great overview of the ongoing research in this area and also categorize the model and control approaches of the different papers. In addition, Delgado [23], as well as Murdock [78] give a comprehensive literature overview. Besides the papers listed in the surveys, up to now the activities in this field did not stop. In the following, publications in relation to this thesis as well as results recently published are introduced.

One of the early publications presenting a TWIP prototype is Grasser et al. [39]. They present the small mobile TWIP 'JOE' together with a modeling approach and experimental results. The system dynamics of the presented prototype are drastically slowed down by a large mass placed on top. In particular, they used the Newtonian method to model the robot and linearized and decoupled the dynamics into two linear models afterwards. Therefore, a 'pendulum' model and a 'rotation' model are received, controlled by two state-space controllers, and interfaced via a decoupling unit to the two DC motors driving the wheels. Hereby, the state-space controllers were designed by pole placement. Grasser et al. [39] concluded, that the backlash of the gearboxes and the non-ideal placement of the encoders as well as the maximum torque that can be transmitted to the ground (grip) prevented that the poles could be moved past a certain limit.

Pathak et al. [84] proposed one of the first models using the Lagrangian approach for modeling and they incorporated the nonholonomic constraints into the dynamic model. Thereupon, a two-level controller structure has been proposed. Based on this, two different controller types for this structure were designed and evaluated. Firstly, a two-level controller which makes use of the partial feedback linearization was designed. The controller stabilizes the TWIP and drives the robot to a given orientation and heading speed set-points. Afterwards, another two-level controller is proposed which not only stabilizes the TWIP but also provides setpoint tracking to a point, starting from any initial configuration. Unfortunately, only simulation results are presented. Moreover, the authors mention that 'further work needs to be done to make the controller design robust with respect to parameter uncertainties'.

Later, Kim and Kwon [56] provided a comparison of the different modeling approaches in the literature. In particular, they compared the 'Newtonian method', the 'Lagrangian approach' and 'Kane's method' (see Kane and Levinson [51]), which is based on the Newtonian method. Herein the authors reveal modeling errors included in many articles. Nevertheless, the model proposed by Delgado et al. [22], using the Lagrange-d'Alembert principle to derive a model which is consistent with the system constraints for the TWIP does not suffer under such modeling errors. Finally, Gajbhiye et al. [38] revealed symmetries in the model of the TWIP. All mentioned works neglect the current dynamics of the DC motors and assume direct torque control or at least steady-state current dynamics.

Among others, between 2021 and 2022 Jamil et al. [49], Luo et al. [67], Mohamed Gad et al. [74], Velagić et al. [109], Zhang and Cai [121] published papers presenting prototypes, and the development of lumped models for simulation and control. Most of them use small microcontrollers to execute a cascaded PID controller or a linear-quadratic regulator (LQR) to stabilize the TWIP. In addition, almost all of these controllers are experimentally tuned and the benchmark experiments are mostly small output steps, driving at slow constant speed or only stabilization. Moreover, there are also research activities applying nonlinear control based on correct dynamic models, e.g. Kim and Kwon [55] in 2017. Unfortunately, Kim and Kwon [55] presented only experimental results changing the heading angle and a torque disturbance rejection. Dengler [25] presented adaptive controllers in the form of function approximators which are applied on a TWIP and presents experiments for setpoint and trajectory tracking but with large control errors. Similar to Grasser et al. [39], Dengler [25] placed a large mass (the battery) on top of his robot to slow down the system dynamics and to ease the control task. Contrary to the works mentioned, Delgado [23] uses a sophisticated dynamic model, a nonlinear controller by applying the presented total energy shaping theory for underactuated systems to the TWIP and presents trajectory tracking in experiments[1].

Finally, there is a long list of student theses as well, where TWIPs were built and controlled. Already in 2003, Ooi [81] built a small TWIP and used pole placement to stabilize the system, and designed a model-free Kalman filter for sensor fusion. Moreover, he designed an LQR and compared it with a pole placement controller. One of the last works is Brandt [11] in 2019 who used a cascaded structure with two PID controllers to stabilize the system.

Recently, algorithms have been presented to ensure a stable operation during setpoint and trajectory tracking, which can be applied to the TWIP. Thereby, the stability of the closed loop system is ensured by the use of a QLF inside the command governor with a known level set, bounding the estimated stable region. Originally, Buhl and Lohmann [13] proposed a setpoint command governor for continuous-time systems with one control input. Based on this, a modification for trajectory tracking and systems with multiple inputs has been presented by Dessort [27], Pieczona [89], Diepold and Pieczona [28] and Diepold [31]. In Diepold [31], a command governor for trajectory tracking has been applied on the single-input-single-output (SISO)-system ‘Inverted pendulum on a cart’ in experiments. All introduced algorithms in these publications consider continuous-time models, trajectories, and control and hence, neglect the time-discrete execution on real-time systems in experiments.

While many papers cover the modeling and control of the TWIP, a much smaller number discuss the sensor modeling, fusion, filtering, and state estimation. Lupian and Avila [69] presented an Kalman-Bucy estimator to observe all state variables required for stabilization and developed an LQR feedback strategy. Unfortunately, the resulting linear-quadratic-Gaussian (LQG) controller has only been tested in simulations. Model uncertainties and external disturbances are regarded as a lumped perturbation term in Zhao et al. [122] and are estimated by a fixed-time extended state observer. By the use of an active disturbance rejection control, the TWIP is stabilized and the estimated perturbation is compensated. The proposed approach has been tested in experiments. A method filtering the measurements for the TWIP running on a low–cost measurement system is presented by Laddach et al. [59]. Hereby, deterministic disturbances are corrected and filtered with Kalman and complementary filters. The performance of the proposed filtering approach is evaluated experimentally.

[1]Delgado [23] used the TWIP prototype for experiments which has been developed and built in the scope of this thesis.

1.3 Outline of the Thesis

In the author's opinion, besides Delgado [23] almost all presented results seem to suffer under at least one of the listed issues:

- Experiments are performed with prototypes having hardware issues, e.g. large gearbox backlash, disadvantageous sensor placement, insufficient encoder resolution, faulty encoder evaluation, algorithms are not executed in real-time, wrong motor drive configuration, highly and strong nonlinear friction.
- Prototypes are used with a large extra mass to slow down the dynamics and thus to ease the control task such that the proposed controllers are able to stabilize the system afterwards.
- Models are based on lumped or even wrong modeling approaches. Furthermore, parameters and friction curves heavily deviate from the prototype ones.
- Only a cascaded PID controller is used, mostly tuned in experiments without any theoretical stability considerations.
- Presented algorithms are not tested experimentally.
- Algorithms are evaluated with trivial tasks in experiments, e.g. only changing orientation.
- No model-based state estimation is used to reduce the error of the estimated state used for control.

In this thesis, a TWIP is built, modeled and algorithms are presented as well as evaluated to alleviate the conflicts introduced. The presented TWIP can be used to educate control engineer students and as a research test bench. In particular, it is suited to develop, implement and evaluate novel control algorithms required to use TWIPs in autonomous applications.

In the next chapters, construction aspects and thorough mathematical modeling methods are presented, and approaches to generate energy optimal trajectories are derived. Based on the TWIP prototype and its mathematical model, new algorithms for setpoint and tracking control as well as state and parameter estimation are developed, simulated, implemented, and finally evaluated in experiments. Figure 1.2 depicts the different topics covered in this thesis, besides the content of the compulsory chapters 'Introduction' as well as 'Outlook and Achievements'.

Chapter 2 presents the development of a TWIP demonstrator and introduces the experimental setup used in the consecutive chapters. At first, a brief overview of previous and recently developed TWIP prototypes is given, revealing issues that affect the functionality of TWIPs. Based on these insights, the design aspects, which have been in focus during the development of our TWIP are introduced and a description of the components as well as the firmware is provided. Finally, the experimental setup with a tracking system and a communication software as well as the structure of the simulation environment is presented.

The mathematical modeling of the TWIP required for simulation, controller, and observer design is the subject of Chapter 3. Firstly, nonlinear time-invariant models of

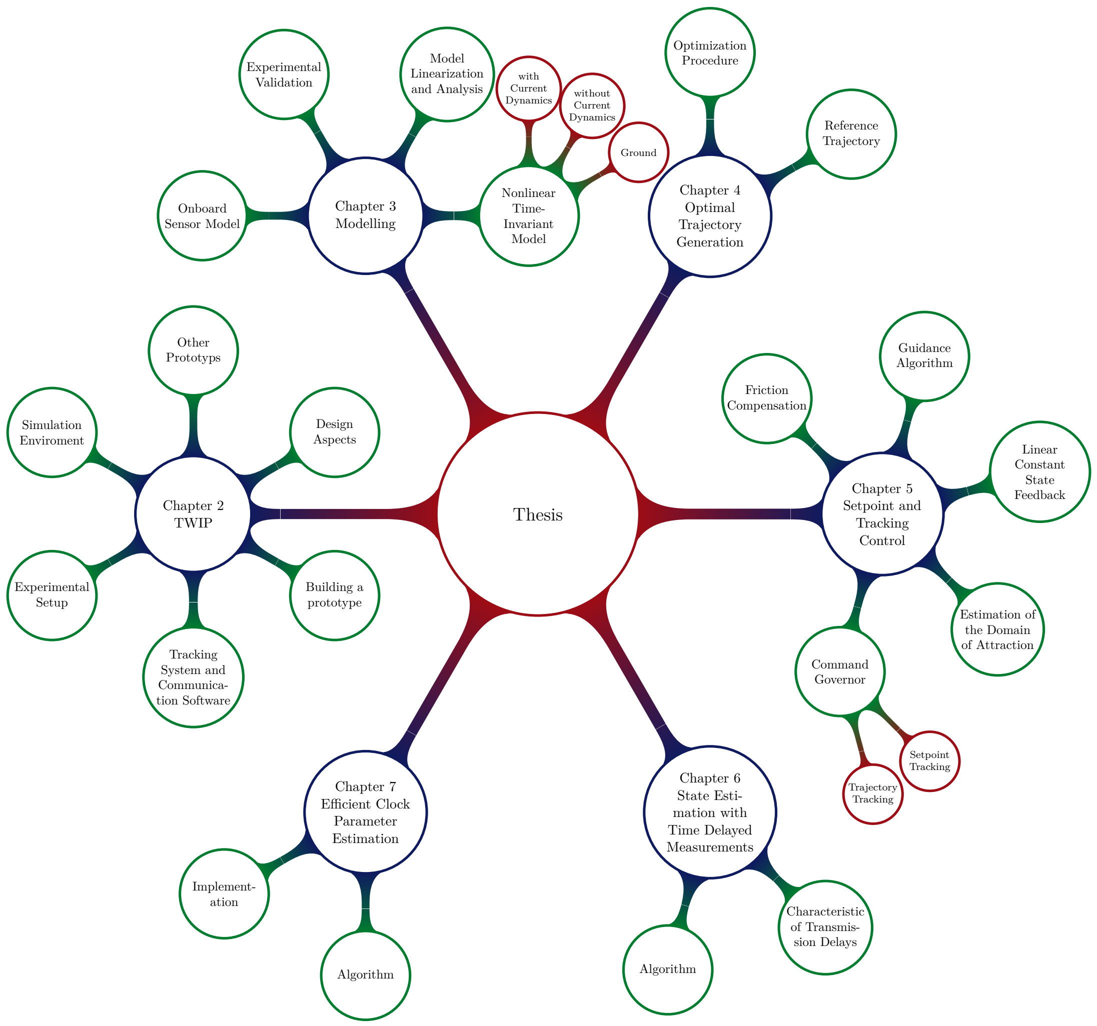


Figure 1.2: Outline of the thesis mindmap

the TWIP in the *two-wheeled inverted pendulum mode* with and without motor currents and the *wheelchair mode* are derived. In addition, alternative system representations are discussed and the energy contribution and power exchange of the different components during trajectory tracking are presented. A model of the onboard sensors is also derived. Moreover, the presented model is evaluated in experiments and the results are reviewed. Thereupon, linear time-invariant (LTI) models are derived and their properties are discussed.

In Chapter 4, a procedure is presented to generate an energy optimal trajectory for the TWIP offline. Moreover, a two-stage initialization is introduced to speed up optimization and to improve the convergence of the nonlinear optimization problem. Based on the proposed method, a reference trajectory is generated and used in the following chapters to evaluate (online) closed-loop control and state estimation algorithms.

A novel discrete-time feedback method is presented in Chapter 5 to stabilize the TWIP and to perform setpoint or trajectory tracking. Thereby, a full discrete-time treatment of all elements of the control structure is derived. Firstly, a friction compensation that reduces the effects of nonlinear mechanical friction is introduced. Afterwards, a guidance algorithm is introduced to overcome the restrictions arising from the fact, that the system underlies nonholonomic constraints. Thereupon, a stabilizing linear constant feedback controller is designed and the stability of the closed-loop system is ensured by a command governor utilizing a QLF to estimate the DoA. Besides the theoretical derivation of the control algorithms, experimental results proof the applicability of the presented approaches.

A novel state estimation algorithm for the TWIP is introduced in Chapter 6. The proposed method minimizes the deterioration of the estimated state due to the non-constant transmission delays of the remote measurements received. In particular, the presented state estimation is a mixture of a time-variant, nonlinear extended Kalman filter (EKF) and a time-invariant, linear, optimal state estimator (or stationary Kalman filter). Contrary to an EKF for the TWIP, the presented algorithm is able to run on a low-budget and low-power microcontroller. At the beginning, the key problem is sketched and a deeper look into the nature of the remote measurement delays is taken. This is done to reveal the need for a novel state estimation algorithm. In consequence, the developed cascaded quasi-linear Kalman filter (CQLKF) method is presented and simulation, as well as experimental results, are provided.

To calculate the non-constant transmission delay of each remote measurement arriving, the clock parameters have to be available for time conversion. Therefore, in Chapter 7 a novel method is proposed to estimate online clock parameters online and efficiently on a microcontroller. In particular, the introduced algorithm is capable of running with limited resources.

All algorithms presented in Chapters 4 to 7 are developed to work together like cogs in a gearbox. Thereby, each is trimmed to be computationally efficient to run on small, low-power microcontrollers used in mobile robots as the TWIP build in Chapter 2. The foundation for a high congruence between simulation and experimental results is laid in Chapter 3 by thoroughly modeling the TWIP.

Finally, in Chapter 8 a summary of the achievements is given together with an outlook, motivating further research in this area.

Chapter 2

Two-Wheeled Inverted Pendulum

This chapter presents the TWIP which has been built and used in this thesis in simulation and experiments to evaluate the proposed control algorithms. The chapter is structured as follows: first, a short insight into the motivation for the development of a TWIP at the Chair of Automatic Control is given. In the second section, to put the contribution of this thesis into context, a brief overview of previous and recently developed TWIP prototypes is provided, revealing minor and major issues that affect the functionality of the TWIP. Afterwards, the design aspects, which have been particularly in focus during the development of our TWIP are introduced. Moreover, a detailed description of the physical components that are used to meet the design requirements is provided. In the subsequent section, the structure of the programmed firmware is introduced. During the development and configuration process of the TWIP, possible pitfalls with regard to the onboard sensors and H-bridges have been identified and are circumvented. Thus, two sections with remarks regarding these pitfalls are provided to raise awareness of them. Subsequently, the overall experimental setup with a tracking system and a communication software used to exchange data with the robot via Bluetooth, is presented. Finally, the simulation environment is introduced and the chapter ends with concluding remarks.

2.1 Motivation

At the Chair of Automatic Control research focusing on real-time control of mechatronic systems has been carried out over the past years, whose content has been offered to the students in the several lectures and lab courses, including lab course 'Controller Implementation on Microcontrollers'. The idea to build a TWIP has been ignited by the urgent need for at least six new robots for this lab course, as the robots used were reaching their end of life. To replace these robots, a new robot was needed, to teach the students different control concepts and its implementation. In addition, another central requirement for the new robot was, that the existing control design challenge, the implementation, and tuning of a flatness-based feedback controller to follow a trajectory on ground in a *wheelchair mode*, is also possible with the new robot. Moreover, an additional and more challenging control task for the students was intended. Due to the available restricted funding and paired with the requirement to offer each of the lab course participants a robot, the price of each robot was limited to a couple of hundred euros.

In parallel, the need for a benchmark system to test recently developed novel control methods rose at the Chair of Automatic Control. To be more precise, the desired system has been required to have an input saturation, to be underactuated as well as unstable but easy to operate. Therefore, the idea was born to combine the development and build a TWIP, which can be used in a lab course for teaching purposes and research in a balancing *two-wheeled inverted pendulum mode.*

As introduced in Section 1.1, several contributions have proposed TWIPs but the authors reported issues with their prototypes. Hence, developing an improved TWIP prototype also contributes to revising these solutions.

Based on the intended use in lab courses and to rapidly run experiments with novel control algorithms, three central requirements stick out. First, since the robot should be used for lab courses, research experiments as well as in lectures, *universal applicability* is specified as a requirement. Secondly, the robot should be suitable for extensive and repeatable experiments which includes the demand for a lightweight setup. This lightweight setup is needed to reduce the risk of injury to the operator and the robot itself if stabilization fails or the robot drives against an obstacle at full speed. Moreover, the robot has to be robust to withstand crashes and ideally can be operated on a desk table. This requirement is condensed as *suitability and robustness for scientific experiments.* Last, but not least, the robot has to show a high *usability for reliable real-time closed-loop control applications.* This means, a sophisticated choice and placement of actors and sensors as well as their connection to a real-time capable microcontroller unit (MCU) is required. Moreover, a software testing and code generation toolchain is necessary to be able to rapidly test new algorithms in real-time.

2.2 Previously and Recently Developed Prototypes

Existing prototypes previously and recently published, rendered to be unsuitable for the application motivated in the previous section and the introduced requirements. The different disadvantages and issues of the presented prototypes are discussed in the following.

Due to the introduced requirements, the robot has to be lightweight with a relatively small design to conduct experiments rapidly and without any danger. Kim and Kwon [55] presented a prototype with a weight of 49 kg, followed by Grasser et al. [39] with 12 kg. The recently developed robot from Dengler [25] has already a drastically reduced weight of only 2 kg but is still considerably heavy. One reason is the solid frame and housing components out of aluminum, which drives up the total weight and requires more powerful and heavy motors to drive the robot. Powerful motors, drive electronics and large batteries thereby contribute an additional danger besides the weight. Another reason for the high weight is, that some of the robots (e.g. Dengler [25]) are equipped with powerful computational units, which are energy-hungry and in consequence, require a large thus heavy battery.

Secondly, most setups placed additional masses as close as possible to the top, to increase inertia and to slow down the system dynamics but the motivated application requires a fast and challenging setup. Placing weights to the top might ease the control task extensively but again at the expense of agility or the required power. This mass placement is done extensively by Grasser et al. [39] by placing large extra weights on the top.

In consequence, the total robot weight increases as well, which leads to the problems mentioned in the previous paragraph. Dengler [25] also placed the large battery on top. Let us derive an estimate of the unstable pole position, based on the assumption that the robot is an inverted pendulum rotating on the wheel axis. If m is the body mass, l is the length from the wheel axis to the center of mass, I the inertia of the body, and g the gravity constant. Then the unstable pole can be calculated by $p = \sqrt{\frac{mgl}{I}}$. The resulting values are $p = 6.1\,\mathrm{rad/s}$ for the robot introduced by Kim and Kwon [55], $p = 8.7\,\mathrm{rad/s}$ for Dengler [25] and $p = 16.5\,\mathrm{rad/s}$ for the robot developed in this thesis with the parameters shown in Table A.1, respectively. The rough estimate given illustrates how different the dynamics are already without damping and no interaction with other components. Moreover, considering damping will only slow down the dynamics and push the poles closer to the origin.

Finally, Grasser et al. [39] mentioned real-time control performance drawbacks arising from gearbox backlash. In addition, the results of Dengler [25] show a large limit cycle, which might be caused by an insufficient encoder resolution and high stick-friction. Grasser et al. [39] claimed that the backlash of the gearbox affects the encoder measurements since the encoders are mounted on the motor shaft and not on the wheels. Therefore, a similar conclusion can be stated for the robot presented by Dengler [25] which has the same encoder configuration. Sadly, Kim and Kwon [55] do not provide any information about the encoders and presented a high-speed spinning motion only, where stick-slip effects cannot be observed. In consequence, no statement can be made about possible hardware problems compared to the prototypes discussed above.

To conclude, these robots most likely fulfilled the purposes they have been built for but do not satisfy the requirements for our applications. For a quick experiment on the desk table, these robots are far too heavy and if the operation fails they cause even a risk of crushing fingers or feet[1]. In addition, hardware issues might degenerate the validity of the gained results in experiments compared to the results gained in simulations and might conceal the improvements reached by a novel control method that has been tested.

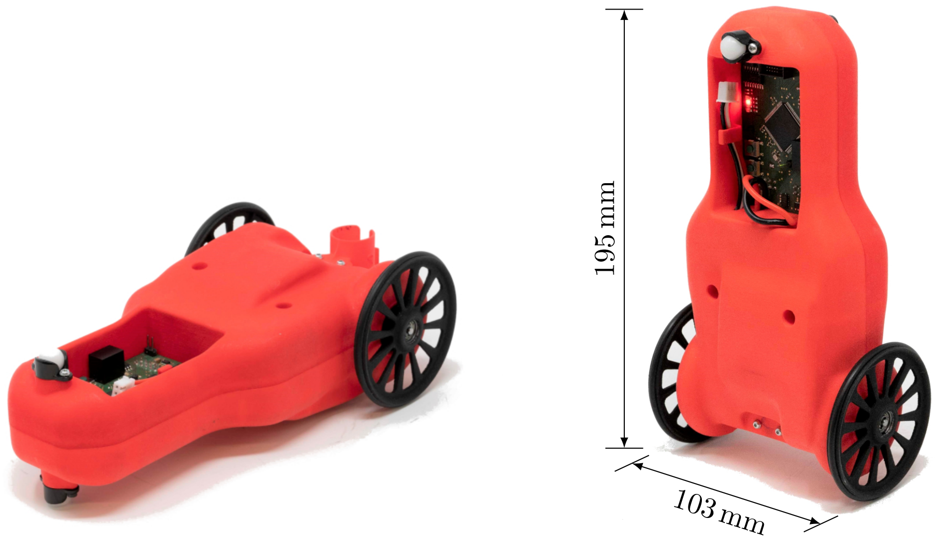
Therefore, a TWIP has been developed in this thesis, well suited for the motivated applications and presented in the next section.

2.3 Building the TWIP Prototype

The TWIP build is shown in Figure 2.1. In the following, the development process is presented accompanied by a discussion of the considered design aspects which were in focus and how they were realized during construction. Moreover, a detailed description of the components is provided examining their most important specifications. In addition, some remarks on hardware details are given, which were found to extensively influence the system's performance or are considered to be quite unpleasant pitfalls.

The development process has been supported by several student theses supervised by the author. Firstly, Hölzle [48] designed the first PCB prototype and contributed an initial version of the firmware. Leonhardt [62] designed the nicely shaped housing shells and was able to balance the robot for the first time, even though he used a manually tuned PID controller and a rod with extra weights to slow down the system dynam-

[1]Kim and Kwon [55] even motivate the spinning experiment instead of a traveling experiment by claiming that "high speed and long travel experiments require a large space and carry the risk of accident"



(a) Wheelchair mode with pen holder (b) Two-wheeled inverted pendulum mode

Figure 2.1: Operation modes of the TWIP

ics. Afterwards, Kaufmann [53] set up the code generation toolchain, identified model parameters, implemented a model inspired by Pathak et al. [84], and presented first stabilization results using an LQR.

2.3.1 Design Aspects

The three requirements of *universal applicability*, *suitability and robustness for scientific experiments*, and *usability for reliable real-time closed-loop control applications* have been defined and were particularly in the center of attention during the development of the TWIP. In the following, their influence on the design concept is introduced.

First, the focus was set on the *universal applicability* of the robot. In consequence, easily modifiable lightweight parts have been designed with the ability to use the robot in multiple applications. Thus, most parts are constructed as rapid prototyping parts, which are easy to change and manufacture, especially the casing shells. All printed parts are manufactured with selective laser sintering (SLS) and are made out of Polyamide/Nylon 12. This design decision led to lightweight but stiff elements due to the thin shell thickness with ribs. The circuit board is easily exchangeable with plugged connections to the motors and encoders. Moreover, two different electronic modules have been designed: a PCB with a 32 bit-MCU with a floating-point unit (FPU) for computationally intensive control tasks and a simpler, easy to program 8 bit-MCU version for teaching and education. Two different operation modes of the robot are considered. In the first mode, the robot drives on ground in a *wheelchair mode*, supported by a ball caster on the robot's top acting as a third omnidirectional wheel. In this mode, a pen holder can be mounted on the robot, such that a whiteboard marker can be attached to record the driven path of the robot on ground as shown in Figure 2.1a. On the other hand, the robot can be used in a balancing *two-wheeled inverted pendulum mode* as presented in Figure 2.1b, which is more challenging, since the open-loop systems equilibrium point is unstable in this configuration.

Secondly, the requirement *suitability and robustness for scientific experiments* has been regarded during the design process. Due to its lightweight and small size, the robot can be operated in reduced spaces like a desk table without any risk of danger to the operator. Moreover, the robot has been constructed and tested to withstand crashes like driving against a wall at full speed or hitting the ground if balancing fails. Even a drop from a table with a height of 1 m typically leads to no serious damage, even though clearly this kind of crash should be avoided. The wheels are mounted via two bearings inside the wheels on a solid wheel axis, which is fixed in the drive mount unit with the motor and the encoders. The chosen lithium-ion polymer battery (LiPo) provides enough energy for more than 2 h of continuous operation. Since 20 identical robots[2] have been produced, artifacts in experiments, which may arise from a single robot, can be identified.

Last, but not least, a central design aspect has been the *usability for reliable real-time closed-loop control applications.* Related to this requirement, an H-bridge is selected, which allows one to choose different operation modes, drive forward, drive backward, brake and coast. On the sensor side, high-resolution codewheels for both wheels have been selected and directly attached to the wheels, to be able to measure the motion between the robot body and the wheel without the gearbox backlash, for odometry purposes. For inertial measurements, a gyroscope and accelerometer are mounted directly on the PCB of the MCU and are accessible via the serial peripheral interface (SPI) bus and the inter-integrated circuit (I^2C) bus with high data rates and low latency. An important aspect has been, that the configurations of the sensor chips are accessible and transparent, such that the chosen filter configurations in the inertial sensor chips are known and can be modeled in the TWIP simulation and may be considered during state observer tuning if required. Furthermore, the position of the acceleration sensor on the PCB has been set onto the middle axis of the robot and as close as possible to the robot's wheel axis, to avoid centripetal forces during pure heading motion and to reduce them at tilt motion. To be able to rapidly test control algorithms, a code generation toolchain has been set up, which enables the user to generate C code out of Simulink for the 32 bit MCU. The source code is then compiled together with the firmware source code and transferred to the robot, where the algorithms are executed in real-time. Due to this, the implemented novel algorithms can be tested in a realistic simulation environment in Matlab/Simulink first and then rapidly converted, compiled, and run on the robot afterwards. The algorithms on the robot are executed with a fixed sample time of 5 ms, triggered by a peripheral timer interrupt. Commands like the desired trajectory or a new set point can be transmitted via Bluetooth from the personal computer (PC). In addition, measurement data from external sensors (like an optical tracking system) can be transferred to the robot. Thus, the robot can deal with local and remote measurements allowing to reduce the error of the estimated states. In the other direction, a Bluetooth connection can be used to simultaneously send local measurements and data from the robot to a PC for signal logging. The firmware itself has been programmed bare-metal in C without an intervening operating system which might disturb the real-time execution of the control algorithms or lead to performance drawbacks. Based on the chosen bare-metal implementation, a fixed and low jitter sampling time of 5 ms is received as well as a known, deterministic run-time behavior of all software components. Finally, during the design process, the masses of the robot are placed to receive fast system dynamics

[2]20 robots of the 2nd generation, 13 of 1st version used by Anhalt [5], Delgado [23], Kaufmann [53], 1 prototype used by Leonhardt [62]

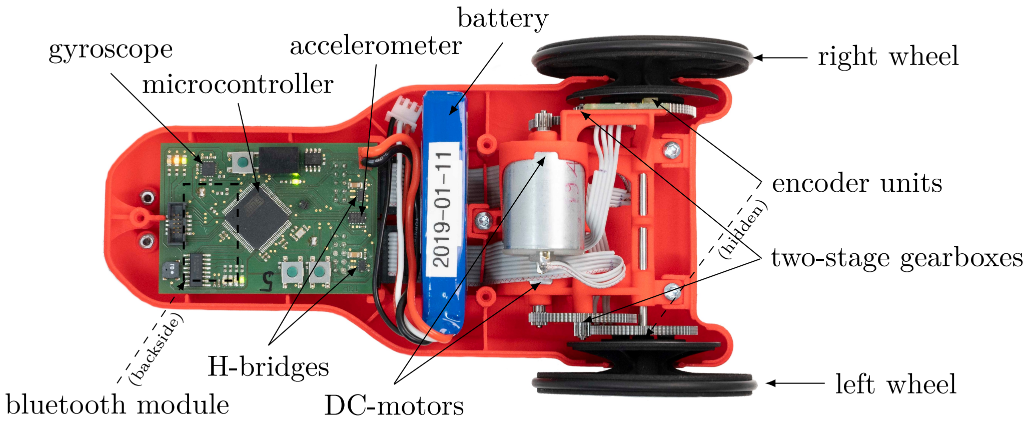


Figure 2.2: TWIP components

in the unstable configuration and thus a challenging but lightweight setup.

In the next section, more details about the outcome of the design process and the chosen components are presented.

2.3.2 Components

The TWIP has a mass of 336 g (battery included), a height of 195 mm, a width of 103 mm and a depth of 33 mm. The center of gravity of the body is at 49.5 mm above the wheel axis, which leads to a highly dynamic and unstable system. Two thin housing shells are used as a case for the robot. The motors, the two-stage gearboxes, the encoder units as well as the wheels are assembled in the drive unit which is mounted in the lower housing shell. The PCB with the MCU and other peripheral chips are clamped between the two housing shells. For the wheelchair operation mode, a ball caster is placed close to the top of each shell. The components labeled in Figure 2.2 are presented in the following.

The two brushed 6 W DC motors accelerate the robot to a forward speed of up to 1.1 m/s or a heading rate of 23.2 rad/s. The motor windings have a resistance of 1.5 Ω and an inductance of $4 \cdot 10^{-4}$ H and the motor has a motor torque constant of $3.76 \cdot 10^{-3}$ N m/A. One two-stage gearbox uses four gears with a module of 0.4. The smaller gears have 11 teeth and the larger gears 78 teeth which leads to the total gear ratio of $(78/11)^2 \approx 50.28$. The required energy for the motors and the electronics is provided by a 7.4 V, 2 cell, 1000 mAh LiPo. The motors are wired up to two H-bridge motor drivers *DRV8835* from Texas Instruments, which limit the motor current to a maximum of 3 A for overload protection. A pulse-width modulation (PWM) with a frequency of 6 kHz and a duty cycle command from 0 to 100 percent is used to emulate an analog motor terminal voltage command. The desired duty cycle is quantized into 1000 increments on the MCU side.

The embedded software for control, state estimation, and communication is executed on a 32 bit MCU (*AT32UC3C1512C* from Atmel/Microchip) which runs at 66 MHz. An integrated FPU in the MCU allows fast execution of single-precision floating point arithmetic. The MCU has 64 kB static random-access memory (SRAM) for variable data and 512 kB flash memory to store the firmware. For the MCU clock generation, an oscillator with a nominal frequency of 20 MHz with a clock frequency error of ±30 ppm

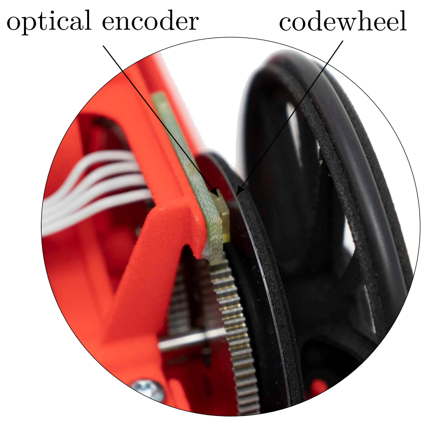


Figure 2.3: TWIP encoder unit

is used. Thus, an error in time of ±108 ms/h is possible due to deviations from the nominal oscillator frequency used for time keeping[3].

For odometry, two optical encoders (*AEDR-8300* from Agilent Technologies) with reflective codewheels of 900 CPR (counts per revolution (CPR)) are built in to measure the relative angular motion between the body and the left and right wheel as shown in Figure 2.3. The two 90° degrees out-of-phase pulses of each encoder are evaluated by a peripheral quadrature decoder (QDEC) on the MCU which leads to an effective resolution of 3600 CPRs.

Two additional sensors are placed on the electronic circuit board for onboard inertial measurements. The 3-axis accelerometer (*ADXL345* from Analog Devices) to measure the body acceleration and the acceleration due to gravity as well as a 3-axis gyroscope (*ITG-3050* from InvenSense) to measure the angular rates of the robot body. At every sample step of the MCU program, the measurement data is queried before the control and estimation algorithms are executed.

A short overview of the sequence of software routines called every sample step is given in the next section. As the choice and configuration of the onboard sensors have a deep impact on the achievable performance of the state estimation and thus control, additional remarks are given afterwards.

2.3.3 MCU Firmware

The firmware running on the MCU is programmed in C and is designed to operate in real-time. This is done by the use of a hardware timer interrupt, which triggers cyclically the execution of the software routines to perform a sample step. As the exchange of serial data with the Bluetooth module, the generation of the PWM signals, and the evaluation of the encoders are done by peripheral components automatically, the execution of the routines during one sample step is never interrupted. A flowchart of the software routines running during one sample step is shown in Figure 2.4.

If the interrupt has triggered the execution of a sample step, the local onboard sensors

[3]This detail turns out to be important, as the onboard time is required to incorporate time-stamped remote measurements received for state estimation.

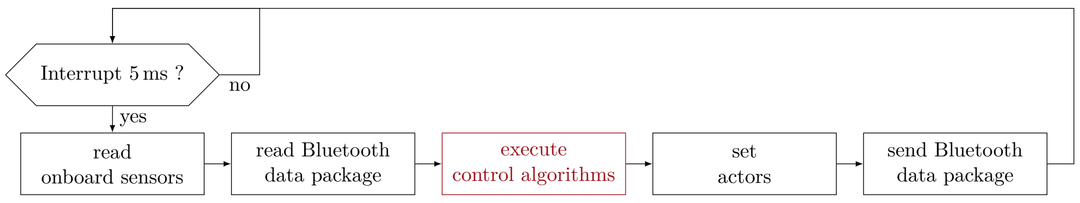

Interrupt 5 ms ?
no
yes
read onboard sensors
read Bluetooth data package
execute control algorithms
set actors
send Bluetooth data package

Figure 2.4: Firmware sample step flowchart

are read first. This is done to assure, that the time between consecutive sensor reads is as precise as possible and thus velocities derived through numerical differentiation of these measurements are not degenerated. Afterwards, the serial data buffer is read. A time-stamp is provided to the last valid data package received from the PC via Bluetooth which is used by the subsequent routines. As all required input data is available now, the control algorithms can be executed. The control algorithm binaries originate from compiled C code auto-generated from a Simulink model as well as Matlab code. After the control algorithms are finished, the actors (PWM, LEDs, etc.) are set corresponding to the control algorithms output. Finally, the last routine is called, which sends out the data specified for signal logging to the PC via Bluetooth.

The firmware, including all control algorithms presented in this thesis (controller, command governor, state estimator, clock parameter estimator) require around 51 kB flash memory and 29 kB are allocated in the SRAM. Moreover, the time required to run the cyclic algorithms, require always less than the available 5 ms with an average execution time of 1 ms. To conclude, the chosen MCU is able to handle the algorithms in real-time but it has also to be said, that the used algorithms have been optimized to run on the available MCU.

2.3.4 Remarks on Onboard Sensors

Typically, inertial measurement units (IMUs) based on microelectromechanical systems (MEMS) technology and consist of at least an acceleration sensor and a gyroscope. Hereby, the quality of the sensors measurements depends on the sensor sensitivity and resolution and gets degenerated by bias, noise, inter-axis misalignment, cross-axis sensitivity, and temperature sensitivity to mention some of the most imported effects among others. In general, there is a simple rule: the more money is spent, the better the measurements are. Prices for MEMS IMUs range from a few euros for customer products up to above thousand Euros for IMUs with ‘tactical grade precision’. Expensive ‘tactical grade precision’ IMUs include a gyroscope and an accelerometer in a single chip, are calibrated and effects like temperature and linear acceleration sensitivity are internally compensated. Due to a limited budget, cheap sensors ($<$ 20 Euro) were bought which limits the reachable performance of the state estimation and complicates the sensor modeling, due to missing specification values in the datasheets. Last but not least, typically all IMUs have internal filters to which special attention should be paid, too. Therefore, it is a good advice to take a close look at the sensor configuration and to choose an appropriate sampling rate and filter cut-off frequency. If the filter bandwidth frequency is chosen too small, this might cause bad overall closed-loop performance or even the stabilization of the robot fails.

Besides the IMU, two encoders are used for velocity measurements and odometry. The choice of the codewheel resolution and its evaluation method is crucial for the later achievable quality of the state estimation and thus closed-loop performance. Thus, it is quite worth to spend some time on this topic during the design procedure.

Let us focus on the evaluation methods first. A common approach in control applications is, to count the pulses during a fixed time interval which is typically the sample time used for the control algorithm. This method is referred to as the M method. It will give a good approximation of the speed as long as more than a few encoder pulses are counted per time interval. In the case of the TWIP, this might not always be the case, since the angular velocity of the wheels is close to zero during balancing at a position and thus only a few encoder pulses are counted per interval. To conclude, the measurement error increases with the M method, as the wheels' speed decreases.

Another approach is, to use a hardware timer and evaluate the timer increments between two encoder pulses. This approach is called the T method. It has a small error if more than a few time increments are counted between two encoder pulses. The higher the timer frequency is, the smaller the error gets. Nevertheless, the error increases with higher velocities, as the time between two encoder pulses decreases, and thus the error increases.

A third approach, proposed by Ohmae et al. [80], combines both methods by counting encoder pulses per time interval and counting timer increments between two consecutive encoder pulses, called the M/T method. Moreover, Ohmae et al. [80] provided a thorough introduction to the three introduced evaluation methods and provides an accuracy analysis of encoder-based speed measurement. Nevertheless, even if the combined method is used, at low speeds, with less than two encoder pulses per sample step of the control algorithm, the speed measurement shows a large relative error.

Clearly, the third method is the desirable one but most MCUs offer only pure QDEC as a peripheral device. QDECs decode the quadrature signals from encoders (two 90 degrees phase shifted pulses) into a total count number which can be interpreted as a discrete angle measurement of the wheel. The benefit of an MCU with a QDEC peripheral device is, that no computation time is required on the MCU to detect the direction and to count the encoder pulses. Unfortunately, an implementation of the M/T method is not straightforward, as long as the required computational cost has to be limited. In the focus on the desired real-time operation of the control algorithms, the design decision has been made to only use the M method to evaluate the wheel speeds. In consequence, expensive high-resolution codewheels were required. But even with the high codewheel resolution, the TWIP presented has at forward velocities below 11.5 mm/s less than one increment at a sample step of 5 ms and the relative measurement error goes up to 100 %. This issue can be reduced by increasing the sampling time or the codewheel resolution. Experimental results of other prototypes have shown, that even worse values (e.g. Dengler [25] 36.5 mm/s with 10 ms sample time) can still be used for control purposes. However, the actual reason why speed control does not completely fail at low speed is, that the average of the calculated velocities approximates the real velocity in a window of several sample steps.

To conclude, a selection of a high resolution codewheel combined with an appropriate choice of the sampling frequency increases the quality of the measurements of the wheel speed. If available, the M/T evaluation method will reduce the measurement error.

Finally, as already mentioned, to avoid the measurement of gearbox backlash by the encoders, a placement of the encoder between the body and the wheels is superior to

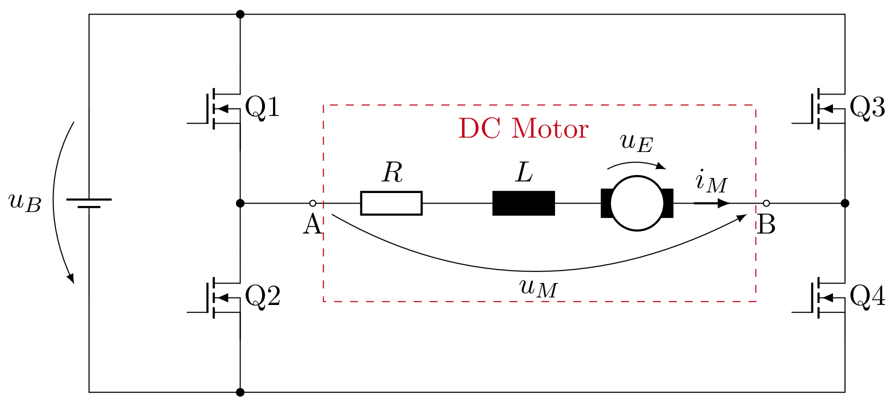


Figure 2.5: H-bridge circuit

encoders mounted on the motor shafts.

2.3.5 Remarks on H-Bridge Operation Modes

In the following, the crucial importance of the correct choice of the operation mode of the H-bridges is highlighted, as this has a large influence on the control properties of the system.

At first, the basic setup and functionality of an H-bridge will be introduced. Figure 2.5 shows a typical H-bridge structure, with four switching elements (Q1-Q4) and the motor wired up to the bridge terminals A and B. In most cases Q1-Q4 are MOSFETs. The H-bridge itself is connected to a power supply, the robot's battery with the voltage u_B. DC motors are commonly modeled by three components. An ohmic resistor R, an inductance L for the rotor windings, and a voltage source u_E for the back electromotive force (back-EMF). u_E is usually assumed to be proportional to the rotor speed ω. To drive the motor in one direction, Q1 and Q4 are driven to low impedance and for the reverse direction, Q3 and Q2 are activated. If Q1 and Q3 or Q2 and Q4 are enabled, the motor terminals are shorted and thus the motor is decelerated. The combinations Q1 and Q2 or Q3 and Q4 should be never active since they will shorten the power source and cause serious damage to the components.

For speed control, in almost all applications a PWM signal is used. There are basically two different PWM operation modes for an H-bridge. To simplify the following explanation of the modes, let us focus on one direction and assume Q4 is always enabled.

The first mode is called *free-wheeling* or *coast* mode. In this mode, at the high level of the PWM signal, Q1 is closed and the motor is driven with the battery voltage. Thus, in the high state, we get $u_M = u_B$ and a motor current can flow. Contrary, during the low level of the PWM signal, Q1 is opened and thus we have an open circuit since the terminal A is high impedance or 'floating'. In consequence, no motor current can flow ($i_M = 0$) and the motor terminal voltage is equal to the back-EMF voltage ($u_M = u_E$).

The second mode is the *break* mode. Similar to the *coast* mode, at the high level of the PWM signal Q1 is closed and during the low level Q1 is opened. But now, in addition, Q2 is closed if Q1 is opened. Thus the motor is driven with $u_M = u_B$ during the high level of the PWM signal and shorted during the low level forcing $u_M = 0$. As the motor terminals are shorted, i_M can be different from zero as it is driven by u_E and limited by the windings resistance R.

Let us now discuss the impact of using the different modes in control applications. In general, the controller output is a desired value $\mathring{u}_M$ for u_M between $\pm u_B$. Based on the sign, the direction is chosen and the PWM duty cycle δ is calculated by $\delta = \frac{\mathring{u}_M}{u_B}$. If we use the *break* mode the average motor voltage $\bar{u}_M$ over one PWM period T is then given by

$$\begin{aligned}\bar{u}_M &= \frac{1}{T}\int_0^T u_M(t)\,\mathrm{d}\tau \\ &= \frac{1}{T}\left(\int_0^{T\delta} u_B\,\mathrm{d}\tau + \int_{T\delta}^{T} 0\,\mathrm{d}\tau\right) = \delta u_B = \mathring{u}_M \end{aligned} \tag{2.1}$$

and thus, $\bar{u}_M$ is equal to the desired value $\mathring{u}_M$.

But what is the result of (2.1) if we use the *coast* mode instead? In this case, during the low level of the PWM signal we have $u_M = u_E(\omega)$ and thus u_M depends on the rotor speed of the motor during this phase. The average motor voltage over one period is then

$$\bar{u}_M = \frac{1}{T}\left(\int_0^{T\delta} u_B\,\mathrm{d}\tau + \int_{T\delta}^{T} u_E(\omega)\,\mathrm{d}\tau\right) = \delta u_B + (1-\delta)u_E(\omega) \neq \mathring{u}_M \tag{2.2}$$

which is not equal to the desired value. To solve this issue, the duty cycle δ has to be calculated based on the measured motor speed and the desired value $\mathring{u}_M$ with $\delta = \frac{\mathring{u}_M - u_E(\omega)}{u_B - u_E(\omega)}$. As δ might get negative, the H-bridge direction might have to be inverted to reach the desired result. Moreover, as during the low level $u_M = u_E(\omega)$ holds, i_M has to be zero and thus the applied torque to the robot is also zero since it is proportional to the motor current. In consequence, the *coast* mode is not desirable in most cases.

Kaufmann [53] evaluated the influence of the different modes and PWM frequencies on the feedforward speed control driving on ground with the TWIP built and his experimental data is plotted in Figure 2.6. Hereby, Kaufmann [53] calculated δ as introduced for the *break* mode and used it for both modes. The *break* mode shows an almost linear relation between PWM duty cycle in percent and the measured wheel velocity. Furthermore, the PWM frequency seems to have no almost influence on the result. Contrary to this, the *coast* mode leads to a high dead-zone and the results depend on the choice of the PWM frequency. With a frequency of 6 kHz the wheels start to turn at a duty cycle of $25 - 30\,\%$. This undesirable dead-zone even increases with 60 kHz, where the motion starts around a duty cycle of $55 - 60\,\%$. Moreover, especially the curve with 6 kHz PWM frequency is clearly nonlinear.

In conclusion, the choice of the operation mode might have a huge influence on the plant input. In the case of the TWIP, the *coast* mode leads to a nonlinear input, which is a function of the motor speed, whereas the *break* mode shows a linear input characteristic. This is an unpleasant pitfall, especially if an 'off-the-shelf' H-bridge breakout board is taken to build up a TWIP and no closer look at the operation mode is taken. In particular, the mismatch of the calculation of the duty cycle and the operation mode might then be misinterpreted as a highly nonlinear mechanical friction with a large stick-friction component. In consequence, awareness of the introduced details on the different H-bride operation modes and their consequences for closed-loop control performance should be raised.

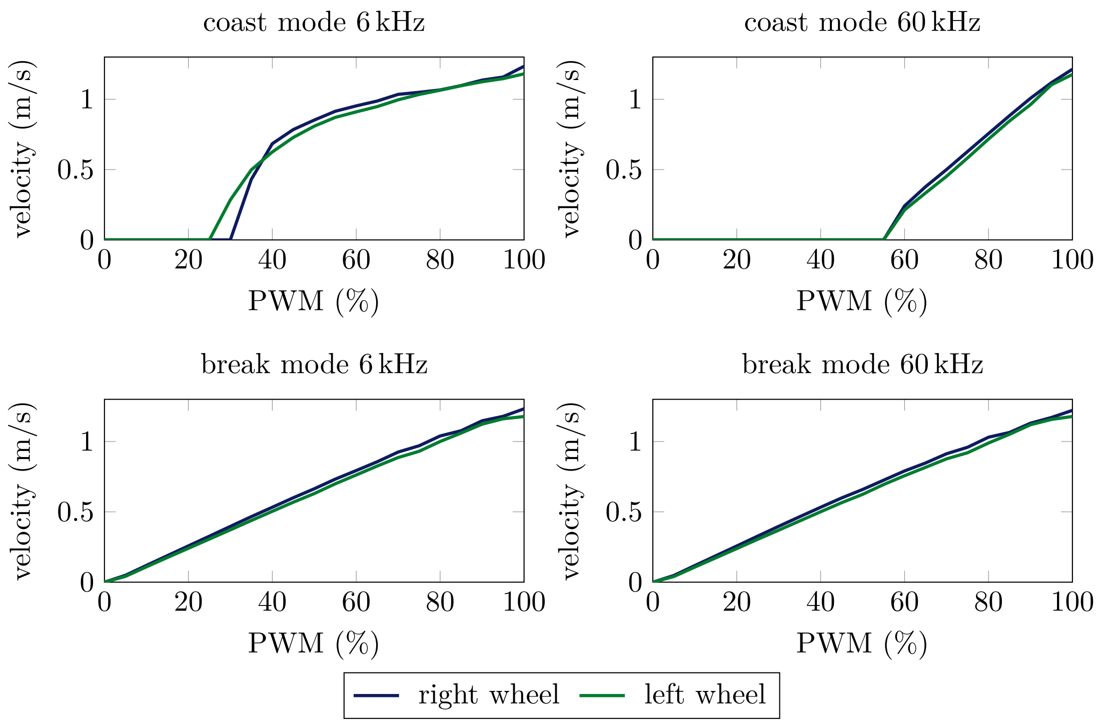


Figure 2.6: Velocity over PWM with different H-bridge operation modes and PWM frequencies (experimental data from Kaufmann [53])

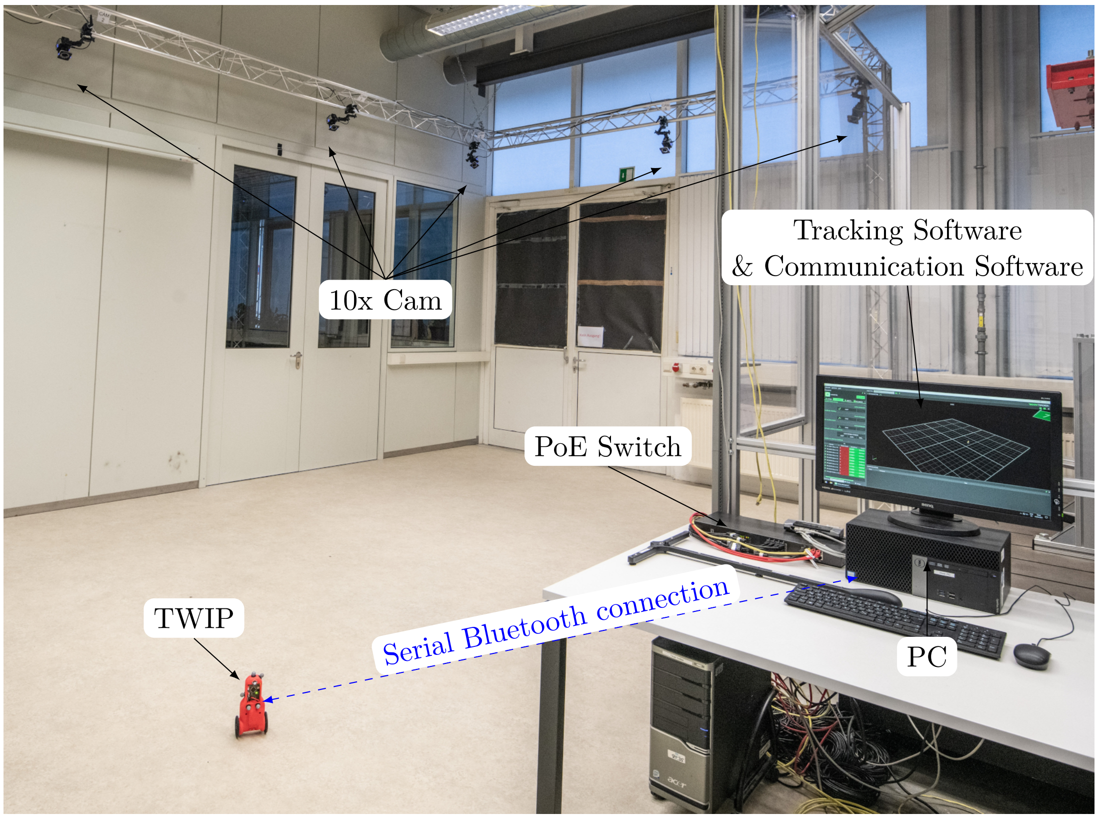


Figure 2.7: TWIP tracking system with TWIP

2.4 Tracking System and Communication Software

An optical tracking system, Vicon Tracker with 10 Vera v1.3 cameras covering a tracking area of 4 m × 6.5 m, provides measurements of the position x_{B} and y_{B} as well as the orientation θ of the robot. In addition, the lab with the tracking system, the cameras, the PC, and the robot is shown in Figure 2.7.

The image data processing, to get the position and orientation of the robot, is done by the tracking software from Vicon running on a PC with Microsoft Windows 10 operating system which has to be considered not to be real-time. Contrary, the triggering of the images is done by the Vera cameras in real-time via Ethernet with a sample time of 20 ms. Hereby the images get an incremented index ${}_{T}k$, which can be interpreted as a timestamp.

In parallel, a communication software written in C♯ is executed on the PC, which queries the last set of tracking data supplied by the tracking software and sends this data set to the robot via Bluetooth. Besides, the communication software reads the desired trajectory from an input file and sends it to the robot. Moreover, data received from the robot for logging is stored in an output file.

As already mentioned, the serial communication between the PC and the microcontroller on the robot is done via Bluetooth. In particular, the Windows[4] serial port profile (SPP) driver has to be used for this, which introduces additional delays due to the buffering done by the Bluetooth/SPP driver. Thus, the measurements from the tracking system received by the robot via Bluetooth inhibit a time delay, and typically every transmitted data set has a different time delay. As already said, the time delay is mainly caused by the scheduling of the operating system and the implementation of its native serial port driver.

2.5 Overview on the Experimental Setup

Figure 2.8 shows the full setup and illustrates the interconnections of the components of the robot introduced in Section 2.3 as well as the tracking system and the communication software presented in Section 2.4.

First, the tracking system including the PC with the tracking software, the communication software as well as the cameras are shown in the upper part of Figure 2.8. A clock triggers the recording of a new image every ${}_{T}m = 20\,\mathrm{ms}$ and is illustrated inside the *Cam 1*. This trigger adds a timestamp (or sample step) ${}_{T}k_{T}$, which is included in the measurement output of the *Tracking Software*. The *Communication Software* reads the measurements from the *Tracking Software*, which are the computed positions $y_{T,x}$ and $y_{T,y}$ as well as the orientation $y_{T,\theta}$ of the robot. Moreover, the *Communication Software* reads the desired trajectory from an input file and transmits the trajectory data as well as a data package D_T including the measurements from the tracking system to the robot via Bluetooth. In addition, the data received via Bluetooth from the robot as well as the measurement data from the tracking system is written into log files by the *Communication Software*. With the data stored in the log files, experiments can be post-processed and analyzed.

In the lower part of Figure 2.8, the robot and the interconnection of the different software and hardware components is shown. All routines running on the robot's MCU are started at every sample step with a sample time of ${}_{R}m = 5\,\mathrm{ms}$. The received trajectory data is buffered in the *Trajectory Buffer*. The *Trajectory Buffer* supplies the current desired state $\mathring{x}$ and feed-forward input $\mathring{u}$ to the *Controller & Command Governor* subsystem. Based on $\mathring{x}$ and $\mathring{u}$ as well as the estimated state $\hat{x}$ the *Controller & Command Governor* calculates the control output u to be applied to the drives of the robot. The robot's state is estimated inside the *State Estimation* block using measurements from local sensors (encoder, accelerometer, and gyroscope) as well as the remote measurements received from the tracking system. If a data package D_T is received, a timestamp ${}_{R}k_{R}$ of the current local time is added and altogether stored in the data package D_P. Since the local and the tracking system clock differ, the measurement timestamps ${}_{T}k_{T}$ from the tracking system are converted in the *Time Conversion* block to the local time ${}_{T}k_{R}$ before they are used by the *State Estimation*. As the offset of the clock is different in every experiment and the clock sample time are also might vary, the clock parameters ${}_{T}m$ and ${}_{T}o$ of the tracking system are estimated inside the *Clock Parameter Estimation*[5] block and

[4]Similar experiment indicated, that the serial Bluetooth communication driver on a standard Ubuntu Linux distribution introduces comparable delays.

[5]To be precise, the clock parameters of the local robot clock ${}_{R}m$ and ${}_{R}o$ as well as the tracking system's clock parameters ${}_{T}m$ and ${}_{T}o$ vary. Nevertheless, to convert the clock time only one clock parameter set in respect to the chosen reference clock has to be determined.

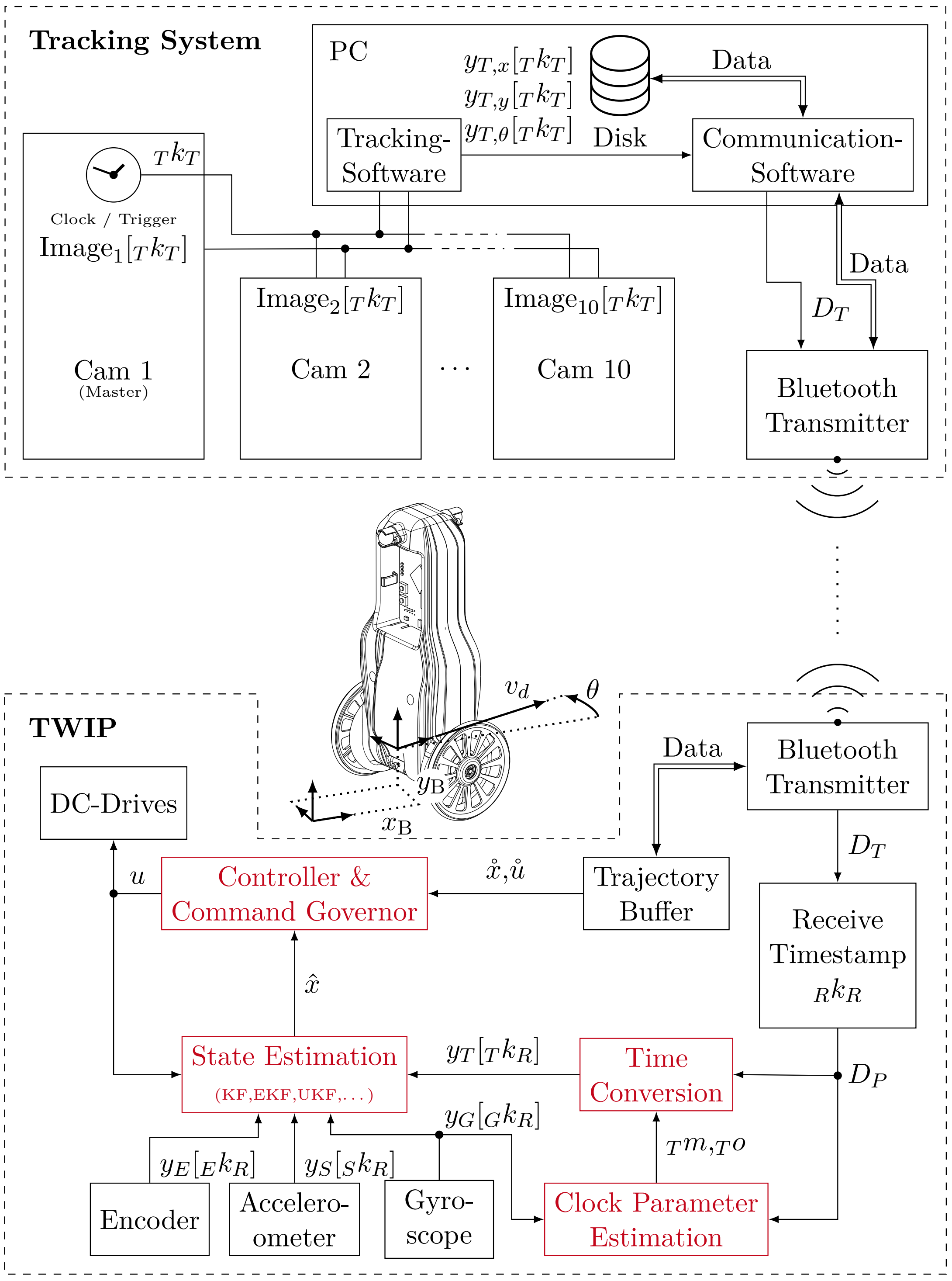


Figure 2.8: Structure of the experimental setup

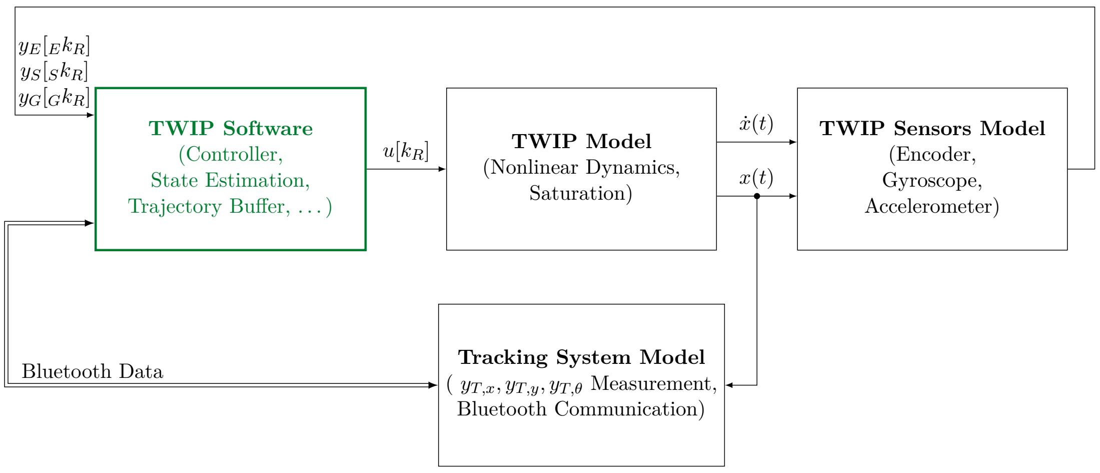

$y_E[{}_Ek_R]$
$y_S[{}_Sk_R]$
$y_G[{}_Gk_R]$
TWIP Software
(Controller, State Estimation, Trajectory Buffer, ...)
$u[k_R]$
TWIP Model
(Nonlinear Dynamics, Saturation)
$\dot{x}(t)$
$x(t)$
TWIP Sensors Model
(Encoder, Gyroscope, Accelerometer)
Bluetooth Data
Tracking System Model
($y_{T,x}, y_{T,y}, y_{T,\theta}$ Measurement, Bluetooth Communication)

Figure 2.9: Structure of the simulation model

are supplied to the *Time Conversion*. Herein, ${}_Tm$ is the sample time (or clock skew), and the ${}_To$ clock offset (or initial time offset). The clock parameters are estimated by the use of the timestamped measurements $y_T[{}_Tk_R]$ from the tracking system, the gyroscope measurements $y_G[{}_Gk_R]$ as well as the receive timestamps ${}_Rk_R$ from the robot. As the data signals of the TWIP which should be logged may change from experiment to experiment, these signals are not shown. In summary, almost all signals could be sent via the *Bluetooth Transmitter* to the PC for logging if required. The blocks colored red, include the algorithms which are presented in this thesis and are tested on the TWIP in experiments.

2.6 Simulation Environment

Last but not least, a simulation environment has been set up in Matlab and Simulink, to test the developed algorithms on a model covering all relevant properties of the physical system. Only if the simulation model is 'close' to the physical system, the simulation results will give a reliable statement about the control performance on the real system.

The structure of the simulation environment is shown in Figure 2.9. The dynamical model of the robot, including the mechanical system and the motors as proposed in Subsection 3.1.6, are included in the *TWIP Model* block. In addition state and input constraints, as presented in Subsection 3.1.9, are incorporated.

As sensors never provide ideal measurements, a model of the onboard sensors are included in the *TWIP Sensors Model* block. The sensor model accounts for noise, quantization, and sampling as well as the digital filters included in the sensors as presented in Section 3.2.

A model of the tracking systems, including the PC and the communication software is labeled as *Tracking System Model*. This model includes the sampling of the tracking systems measurements and a sloppy transmission model, which approximates the transmission delays of the data packages found on the real system.

The simulation environment above will be used to tune the controller and the observer offline. In particular, it is used to optimize the performance of the state estimator in

Chapter 7. Besides, the simulation is used for fast bug fixing and to investigate the influence of parameter deviations.

2.7 Concluding Remarks

In this chapter, the conceptual and building process of the TWIP during this thesis has been presented which is used in the following chapters to evaluate control algorithms proposed for mobile robots. A strong focus during the design process on the desired sophisticated real-time closed-loop control application and experiments lead to a robust and lightweight robot. Furthermore, the appropriate choice of the sensors deliver high-quality measurement signals required for precise state estimation, and due to the connection of a tracking system as a remote sensor to the robot, integration errors in position and heading angle can be compensated. Based on the bare-metal firmware on the MCU all algorithms can be executed in real-time. A code generation framework provides fast experimental testing and the simulation environment a valid offline evaluation of new algorithms. Due to the improved applicability of the presented TWIP for control, it has been used for several other research projects and semester, master and Ph.D. theses, eg. Delgado [23], Anhalt [5], Albert et al. [2]. Moreover, it has been presented in the 150 year Technical University of Munich (TUM) exhibition and is used as an illustration example for control in the lectures of the institute frequently. Besides, the robot has shown its robustness in several lab courses. The developed TWIP can be considered as one of the most sophisticated TWIP for research and teaching presented up to now.

Chapter 3

Modeling

For simulation, controller, and observer design, a model showing a high congruence with the physical system is required. In Section 2.6 a simulation environment has been proposed and in this chapter, the different blocks shown in Figure 2.9 are 'filled' with equations. Thus, nonlinear time-invariant models of the TWIP in the *two-wheeled inverted pendulum mode* with and without motor currents and the *wheelchair mode* are derived first. In addition, alternative system representations are introduced, which turn out to ease the controller and estimator design procedures in the following chapters. Moreover, a short excursion into the energy contribution and power exchange of the different components is given to evaluate if terms could be neglected which would allow to simplify the derived models. Also, a model of the onboard sensors, required to tune state estimators with simulations is presented. Thereupon, LTI models are derived for linear control and observer synthesis. Moreover, the properties of the different linear models are discussed. Afterwards, experimental results are compared with simulation results obtained with the introduced model. Finally, the chapter is closed with concluding remarks.

The derivation of the different models has been supported by several student theses supervised by the author. Kaufmann [53] implemented the first model based on Pathak et al. [84] but in addition, he incorporated a model of the motors and a linear description of the friction between the wheels and the body. In addition, Kaufmann [53] identified the motor parameters and mechanical properties based on experiments as well as the CAD model. Anhalt [5] replaced the linear terms with a nonlinear friction curve, identified experimentally which increased the model accuracy even further. Even though the presented sensor model is quite different, the first version of an onboard sensor model has been proposed by Wunderlich [117].

3.1 Nonlinear Time-Invariant Model

In this section, three different dynamical models of the TWIP are derived. In particular, a closed treatment of the current dynamics and the mechanical dynamics within the Lagrangian framework is presented which is novel compared to Delgado [23], Kaufmann [53], Anhalt [5] and less error-prone. Furthermore, the Lagrangian of the system is derived which can be used to derive a variational integrator to get a discrete-time model which preserves system invariants like momentum and energy as presented in Albert et al. [2].

All models in this section will cover the dynamics of rigid bodies and consider the two DC motors. Hereby, the voltages of the electric motor's terminals are defined as model input and the H-bridge modeling is excluded. In particular, the presented models for the *two-wheeled inverted pendulum mode* with and without motor currents and the *wheelchair mode* will differ in the choice of configuration and state variables as well as the treatment of the current dynamics.

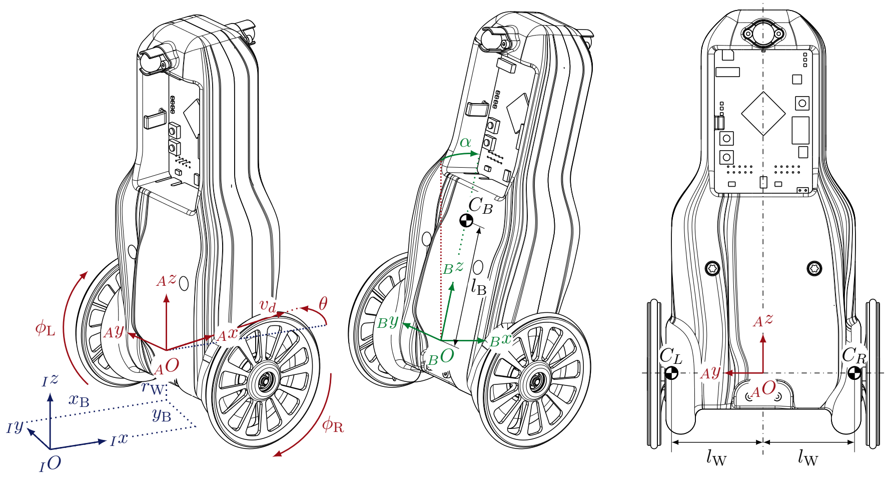


Figure 3.1: TWIP coordinate systems and geometric parameters

A simplified model of the TWIP, as shown in Figure 3.1, consists of three[1] rigid bodies: the robot's body, the right, and the left wheel. Additionally, to gain a high model accuracy, terms for the gear's and motor rotor's rotational energy are considered which have not been included in most other publications e.g. Kim and Kwon [56], Pathak et al. [84], Ha and Yuta [40].

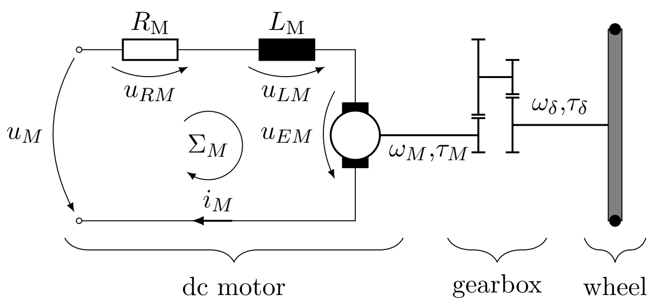


Figure 3.2: Diagram of the electronic motor circuit with an illustration of the connection to a gearbox and a wheel (inspired by Delgado [23])

Furthermore, the motor circuit electronics, shown in Figure 3.2, are incorporated in the model, instead of assuming a pure torque or current input, as commonly done e.g. Kim and Kwon [56], Pathak et al. [84], Ha and Yuta [40]. Delgado [23] and Anhalt [5] applied Kirchhoff's second law for the mesh Σ_M in the electric circuit. While Delgado

[1]Actually, there are significantly more rigid bodies if the motor rotors and gears are considered as individual parts. This is discussed in Subsection 3.1.3 in detail.

[23] neglected the current dynamics afterwards, Kaufmann [53] and Anhalt [5] included them in the final model. To conclude, both derived the mechanical and electrical model separately and joined them in consecutive steps by replacing the torque input of the mechanical system with the torque equations gained by the modeling of the electrical system. Contrary to this approach, in the following the current dynamics as well as the mechanical dynamics will be treated within the Lagrangian framework.

Kim and Kwon [56] compared different modeling approaches published and pointed out errors. Unfortunately, they removed the spatial positions x_B and y_B by distance driven d and thus completely lost track of the capability for position tracking in their nonlinear model. To gain that property back, they will need to extend their system with (3.96) afterwards.

Finally, the system is subject to nonholonomic constraints that arise due to the assumption of no slipping and pure rolling of the wheels. To account for these nonholonomic constraints, constraint equations are formulated during the modeling process and incorporated into the model as proposed by Pathak et al. [84].

3.1.1 Parameters and Lagrange Equations

Firstly, the parameters and properties of the three rigid bodies are introduced. The robot's body has a mass m_B with the center of gravity C_B at a height l_B, measured from the body-fixed frame origin ${}_BO$ along the ${}_Bz$ axis. The two wheels of radius r_W have a mass of m_W each and are mounted on the wheel axis which is aligned with the coordinate frame axis ${}_Ay$. Their centers of mass C_R and C_L are located on the wheel axis, with a distance of l_W from the middle of the robot. Each wheel is able to rotate independently and is driven by an electric motor via a gearbox stage, assembled in the robot's body.

Secondly, a look is taken at the drive electronics and the connection of the motors to the wheels via the gearboxes. Since both wheels, motors and gearboxes are similar, only one set of them are discussed. The motor armature is modeled with an ohmic resistance of the rotor windings R_M and their inductance L_M connected in series as shown in Figure 3.2. If the armature rotates with the speed ω_M in the magnetic field of the permanent magnets, the voltage $u_{EM} = k_\mathrm{E}\omega_M$ is induced in the armature windings. k_E is hereby the back-EMF constant of the motor. In addition, the current i_M flowing through the armature coil causes a torque $\tau_M = k_\mathrm{M} i_M$ where k_M is the motor torque constant. The motor armature shaft is connected to the gearbox. Each stage has a ratio of n_WG and the two build-in stages deliver a total gear ratio of $n_\mathrm{WM} = n_\mathrm{WG}^2$ from the motor shaft to the wheels.

Now, as the required parameters and the robot's structure are defined, let us proceed with the modeling. Herein, the goal is to compute a state-space representation of the model with a desired choice of state variables. In the first step, let us formulate the Lagrangian L of the TWIP

$$L(\mathfrak{q}, \dot{\mathfrak{q}}) = T(\mathfrak{q}, \dot{\mathfrak{q}}) - V(\mathfrak{q}, \dot{\mathfrak{q}}) \ , \tag{3.1}$$

which is the total kinetic energy T minus the potential energy V where both terms depend on the configuration variable vector $\mathfrak{q}$ and its time-derivative $\dot{\mathfrak{q}}$. Then, based on the Lagrangian, the Euler–Lagrange equations (or Lagrange's equations of the second kind)

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\mathfrak{q}}}\right) - \frac{\partial L}{\partial \mathfrak{q}} =$$

$$\underbrace{\frac{\partial^2 T}{\partial \dot{\mathfrak{q}}^2}}_{M}\ddot{\mathfrak{q}} + \underbrace{\frac{\partial^2 T}{\partial \dot{\mathfrak{q}}\partial \mathfrak{q}}\dot{\mathfrak{q}} - \frac{\partial T}{\partial \mathfrak{q}}}_{K} \underbrace{- \frac{\partial^2 V}{\partial \dot{\mathfrak{q}}^2}\ddot{\mathfrak{q}} - \frac{\partial^2 V}{\partial \dot{\mathfrak{q}}\partial \mathfrak{q}}\dot{\mathfrak{q}} + \frac{\partial V}{\partial \mathfrak{q}}}_{P} = \mathcal{F}_{\mathrm{dis}} + \mathcal{F}_{\mathrm{ext}} + A^{\mathrm{T}}\lambda\ , \tag{3.2a}$$

$$A\dot{\mathfrak{q}} = 0, \tag{3.2b}$$

are calculated which gives a constrained second-order representation of the system. For this, the dissipative $\mathcal{F}_{\mathrm{dis}}$ and external forces $\mathcal{F}_{\mathrm{ext}}$ acting on the system are needed. In contrast to holonomic systems, additional equations are required for the TWIP, which enforce the velocity constraints. Therefore, they are included into the model in form of a matrix-vector product $A(\mathfrak{q})\dot{\mathfrak{q}} = 0$, also known as Pfaffian constraint (see Choset [20]). The vector λ includes the Lagrange multipliers. In this second-order system, M is the mass matrix, K assembles the terms arising from the kinetic energy and P includes the terms from the potential energy.

For mechanical systems usually, the potential energy term only depends on $\mathfrak{q}$ and thus the terms $\frac{\partial^2 V}{\partial \dot{\mathfrak{q}}^2}$, $\frac{\partial^2 V}{\partial \dot{\mathfrak{q}}\partial \mathfrak{q}}$ in (3.2a) are zero and dropped. Knowing that this, at least for the second term, is not necessarily the case if the electronic circuit is also modeled within the Lagrangian-Euler approach, these terms are explicitly included in P. Wellstead [113] offers a great overview of how to model systems with different, connected physical domains with the Euler-Lagrangian approach whereas Wells [112] especially introduces the modeling of coupled electro-mechanical systems.

The motion of the TWIP is restricted by nonholonomic constraints and the admissible velocities have to fulfill our condition $A(\mathfrak{q})\dot{\mathfrak{q}} = 0$. Let us consider

$$\dot{\mathfrak{q}} = S(\mathfrak{q})\nu \tag{3.3}$$

as a mapping between the velocities $\dot{\mathfrak{q}}$ in generalized coordinates and the velocities ν in permissible directions. Moreover, let us choose the matrix S such that it lies in the null space of the matrix A and thus $AS = 0$ holds. Now the Lagrange multipliers λ are eliminated by pre-multiplying both sides in (3.2a) by S^{T}. Then substituting (3.3) and $\ddot{\mathfrak{q}} = S\dot{\nu} + \dot{S}\nu$ leads to the second-order system

$$S^{\mathrm{T}}\left(MS\dot{\nu} + M\dot{S}\nu + K + P - \mathcal{F}_{\mathrm{dis}} - \mathcal{F}_{\mathrm{ext}}\right) = 0 \tag{3.4}$$

without Lagrange multipliers and in minimal coordinates. Finally, (3.4) is rewritten to

$$\dot{\nu} = -\left(S^{\mathrm{T}}MS\right)^{-1}S^{\mathrm{T}}\left(M\dot{S}\nu + K + P - \mathcal{F}_{\mathrm{dis}} - \mathcal{F}_{\mathrm{ext}}\right) \tag{3.5}$$

and a first order, nonlinear state space model with the state vector $x = (*, \nu)^{\mathrm{T}}$

$$\dot{x} := \begin{pmatrix} S_r\nu \\ -\left(S^{\mathrm{T}}MS\right)^{-1}S^{\mathrm{T}}\left(M\dot{S}\nu + K + P - \mathcal{F}_{\mathrm{dis}} - \mathcal{F}_{\mathrm{ext}}\right)\end{pmatrix}\ , \tag{3.6}$$

can be set up where the matrix $S_r(\mathfrak{q})$ defines the integration terms of the state variables labeled with the placeholder $(*)$ and is chosen, based on the use case for which the model should be used as well as the desired control task. The introduced method to remove the term $A^{\mathrm{T}}\lambda$ in (3.2a) by the use of a mapping $S(q)$ between generalized velocities and admissible velocities in minimal coordinates, has been proposed by Pathak et al. [84].

The presented procedure from (3.1) to (3.6) is the same for all presented models of the TWIP in the subsequent subsections.

3.1.2 Positions and Velocities

To derive (3.1), the kinetic and potential energy terms for each considered component are needed. The kinetic energy terms of the mechanical components consist of a rotational and a translational velocity energy term. To formulate those, the rotational and transitional velocity of the center of mass for each rigid body with respect to the inertial frame are required. A common approach is to describe the transitional velocities of a body in the inertial frame and the rotational (or angular) velocities in the body-fixed frame. Let us follow this approach since it eases the formulation of the energy terms and the inertia matrix with respect to the center of mass is constant. Thus we start with the modeling of the mechanical system by calculating the absolute velocities of the centers of mass as well as the body-fixed rotational velocities of the robot's body, the left and the right wheel. A rotation around the ${}_{I}z$ axis with the heading angle θ results in the rotation matrix

$$ {}^{I}R_{A} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.7} $$

which transforms a position vector defined in the axis-fixed frame A of the robot to the inertial frame I. A consequent rotation around the ${}_{A}y$ axis with the tilt angle α brings us with the rotation matrix

$$ {}^{A}R_{B} = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \tag{3.8} $$

to the orientation of the body-fixed frame B of the body. Multiplying both principal rotations results in the total rotation matrix

$$ {}^{I}R_{B} = {}^{I}R_{A}{}^{A}R_{B} = \begin{pmatrix} \cos(\alpha)\cos(\theta) & -\sin(\theta) & \sin(\alpha)\cos(\theta) \\ \cos(\alpha)\sin(\theta) & \cos(\theta) & \sin(\alpha)\sin(\theta) \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \tag{3.9} $$

from frame B to frame I . Let us define the position vector from the inertial frame origin ${}_{I}O$ to the center of gravity C_B of the body

$$ {}_{I}p_{C_B} = \begin{pmatrix} x_{\mathrm{B}} \\ y_{\mathrm{B}} \\ r_{\mathrm{W}} \end{pmatrix} + {}^{I}R_{B} \begin{pmatrix} 0 \\ 0 \\ l_{\mathrm{B}} \end{pmatrix} = \begin{pmatrix} x_{\mathrm{B}} + l_{\mathrm{B}}\sin(\alpha)\cos(\theta) \\ y_{\mathrm{B}} + l_{\mathrm{B}}\sin(\alpha)\sin(\theta) \\ r_{\mathrm{W}} + l_{\mathrm{B}}\cos(\alpha) \end{pmatrix} . \tag{3.10} $$

This vector is the sum of the vector from ${}_I O$ to ${}_A O = {}_B O$ and the vector pointing from ${}_B O$ to C_B, transformed with ${}^I R_B$ from the frame B to the frame I. Based on (3.10) we calculate the absolute velocity of C_B

$$
\begin{aligned}
{}_I v_{C_B} &= \frac{\mathrm{d}}{\mathrm{d}t} \left({}_I p_{C_B}\right) && (3.11)\\
&= \begin{pmatrix} \dot{x}_\mathrm{B} + l_\mathrm{B}\dot{\alpha}\cos(\alpha)\cos(\theta) - l_\mathrm{B}\dot{\theta}\sin(\alpha)\sin(\theta) \\ \dot{y}_\mathrm{B} + l_\mathrm{B}\dot{\alpha}\cos(\alpha)\sin(\theta) + l_\mathrm{B}\dot{\theta}\sin(\alpha)\cos(\theta) \\ -l_\mathrm{B}\dot{\alpha}\sin(\alpha) \end{pmatrix} && (3.12)
\end{aligned}
$$

in the frame I. The rotational velocity vector of the robot's body-fixed frame B in respect to the inertial frame I, given in frame B, is calculated by

$$
\begin{aligned}
{}^I_B\tilde{\omega}_B &= {}^I R_B^\mathrm{T} \frac{\mathrm{d}}{\mathrm{d}t}\left({}^I R_B\right) && (3.13)\\
{}^I_B\omega_B &= \begin{pmatrix} -\dot{\theta}\sin(\alpha) \\ \dot{\alpha} \\ \dot{\theta}\cos(\alpha) \end{pmatrix}. && (3.14)
\end{aligned}
$$

Since there are three bodies in total, the same steps for the left and right wheel have to be performed also. Consequently, for each wheel a body-fixed frame has to be introduced. These frames R and L are rotated by the wheel angles ϕ_R and ϕ_L around the axis ${}_A y$ and thus we define the rotation matrices

$$
{}^A R_R = \begin{pmatrix} \cos(\phi_\mathrm{R}) & 0 & \sin(\phi_\mathrm{R}) \\ 0 & 1 & 0 \\ -\sin(\phi_\mathrm{R}) & 0 & \cos(\phi_\mathrm{R}) \end{pmatrix}, \quad {}^A R_L = \begin{pmatrix} \cos(\phi_\mathrm{L}) & 0 & \sin(\phi_\mathrm{L}) \\ 0 & 1 & 0 \\ -\sin(\phi_\mathrm{L}) & 0 & \cos(\phi_\mathrm{L}) \end{pmatrix} \tag{3.15}
$$

from the axis-fixed frame A to the body-fixed frames for the right wheel R and left wheel L. Again, by the multiplication of the principal rotations, a total rotation matrix from the inertial frame I to the corresponding frames of right wheel

$$
{}^I R_R = {}^I R_A {}^A R_R = \begin{pmatrix} \cos(\phi_\mathrm{R})\cos(\theta) & -\sin(\theta) & \sin(\phi_\mathrm{R})\cos(\theta) \\ \cos(\phi_\mathrm{R})\sin(\theta) & \cos(\theta) & \sin(\phi_\mathrm{R})\sin(\theta) \\ -\sin(\phi_\mathrm{R}) & 0 & \cos(\phi_\mathrm{R}) \end{pmatrix} \tag{3.16}
$$

and the left wheel

$$
{}^I R_L = {}^I R_A {}^A R_L = \begin{pmatrix} \cos(\phi_\mathrm{L})\cos(\theta) & -\sin(\theta) & \sin(\phi_\mathrm{L})\cos(\theta) \\ \cos(\phi_\mathrm{L})\sin(\theta) & \cos(\theta) & \sin(\phi_\mathrm{L})\sin(\theta) \\ -\sin(\phi_\mathrm{L}) & 0 & \cos(\phi_\mathrm{L}) \end{pmatrix} \tag{3.17}
$$

is obtained. Since the centers of gravity C_R and C_L of the right and left wheel are aligned on the ${}_A y$ axis, only the rotation matrix for frame A is needed to define the position vectors from ${}_I O$ to C_R and ${}_I O$ to C_L. Adding the vector from ${}_I O$ to ${}_A O$ to the vectors from ${}_AO$ to C_R and ${}_AO$ to C_L, which are transformed with ${}^IR_A$ or ${}^IR_L$ respectively, the position vectors

$$ {}_Ip_{C_R} = \begin{pmatrix} x_\mathrm{B} \\ y_\mathrm{B} \\ r_\mathrm{W} \end{pmatrix} + {}^IR_A \begin{pmatrix} 0 \\ l_\mathrm{W} \\ 0 \end{pmatrix} = \begin{pmatrix} x_\mathrm{B} + l_\mathrm{W}\sin(\theta) \\ y_\mathrm{B} - l_\mathrm{W}\cos(\theta) \\ r_\mathrm{W} \end{pmatrix} \tag{3.18} $$

and

$$ {}_Ip_{C_L} = \begin{pmatrix} x_\mathrm{B} \\ y_\mathrm{B} \\ r_\mathrm{W} \end{pmatrix} + {}^IR_A \begin{pmatrix} 0 \\ -l_\mathrm{W} \\ 0 \end{pmatrix} = \begin{pmatrix} x_\mathrm{B} - l_\mathrm{W}\sin(\theta) \\ y_\mathrm{B} + l_\mathrm{W}\cos(\theta) \\ r_\mathrm{W} \end{pmatrix} \tag{3.19} $$

are calculated for the right and left wheel's center of gravity. Having defined the position vectors, we are able to calculate the velocity of the wheel's centers of gravity given in the inertial frame with

$$ {}_Iv_{C_R} = \frac{\mathrm{d}}{\mathrm{d}t}\left({}_Ip_{C_R}\right) = \begin{pmatrix} \dot{x}_\mathrm{B} + l_\mathrm{W}\dot{\theta}\cos(\theta) \\ \dot{y}_\mathrm{B} + l_\mathrm{W}\dot{\theta}\sin(\theta) \\ 0 \end{pmatrix} \tag{3.20} $$

for the right wheel and

$$ {}_Iv_{C_L} = \frac{\mathrm{d}}{\mathrm{d}t}\left({}_Ir_{C_L}\right) = \begin{pmatrix} \dot{x}_\mathrm{B} - l_\mathrm{W}\dot{\theta}\cos(\theta) \\ \dot{y}_\mathrm{B} - l_\mathrm{W}\dot{\theta}\sin(\theta) \\ 0 \end{pmatrix}. \tag{3.21} $$

for the left wheel. The rotational velocity vector of the robot's wheels are calculated in the corresponding body-fixed frame by

$$ {}_R^I\tilde{\omega}_R = {}^IR_R^\mathrm{T}\frac{\mathrm{d}}{\mathrm{d}t}\left({}^IR_R\right) \tag{3.22} $$

$$ {}_R^I\omega_R = \begin{pmatrix} -\dot{\theta}\sin(\phi_\mathrm{R}) \\ \dot{\phi}_\mathrm{R} \\ \dot{\theta}\cos(\phi_\mathrm{R}) \end{pmatrix}. \tag{3.23} $$

for the right wheel and

$$ {}_L^I\tilde{\omega}_L = {}^IR_L^\mathrm{T}\frac{\mathrm{d}}{\mathrm{d}t}\left({}^IR_L\right) \tag{3.24} $$

$$ {}_L^I\omega_L = \begin{pmatrix} -\dot{\theta}\sin(\phi_\mathrm{L}) \\ \dot{\phi}_\mathrm{L} \\ \dot{\theta}\cos(\phi_\mathrm{L}) \end{pmatrix}. \tag{3.25} $$

for the left wheel.

3.1.3 Kinetic and Potential Energy Terms

Let us define the energy terms for the rigid bodies in the next steps. Starting with the robot's body, we use the translational velocity and the body mass as well as the angular velocity of the center of mass of the body together with $I_{\mathrm{B}} := \mathrm{diag}(I_{\mathrm{Bxx}}, I_{\mathrm{Byy}}, I_{\mathrm{Bzz}})$ as the inertia of the body with respect to its center of mass in the body-fixed frame B. Then, the kinetic energy of the body is defined by

$$T_B = \frac{1}{2}(m_{\mathrm{B}}\ {}_{I}v_{C_B}^{\mathrm{T}}\ {}_{I}v_{C_B} + {}_{B}^{I}\omega_B^{\mathrm{T}}\ I_{\mathrm{B}}\ {}_{B}^{I}\omega_B)\ . \tag{3.26}$$

Analogously, the kinetic energy of the wheels

$$T_W = \frac{1}{2}(m_{\mathrm{W}}\ {}_{I}v_{C_R}^{\mathrm{T}}\ {}_{I}v_{C_R} + {}_{R}^{I}\omega_R^{\mathrm{T}}\ I_{\mathrm{W}}\ {}_{R}^{I}\omega_R) \tag{3.27}$$
$$+ \frac{1}{2}(m_{\mathrm{W}}\ {}_{I}v_{C_L}^{\mathrm{T}}\ {}_{I}v_{C_L} + {}_{L}^{I}\omega_L^{\mathrm{T}}\ I_{\mathrm{W}}\ {}_{L}^{I}\omega_L)\ , \tag{3.28}$$

is calculated where $I_{\mathrm{W}} := \mathrm{diag}(I_{\mathrm{Wxx}}, I_{\mathrm{Wyy}}, I_{\mathrm{Wzz}})$ is the inertia of the wheels with respect to their center of mass in the body-fixed frame R or L respectively. Notice from Figure 3.2, that the rotor of the electric motor and the gears rotate at different angular speeds compared to the wheels and the body. Therefore, the rotational energy arising from this has to be calculated in addition. Let us define auxiliary variables for the relative motion between the body and the wheels as follows:

$$\omega_{\delta\mathrm{R}} = \dot{\phi}_{\mathrm{R}} - \dot{\alpha} \tag{3.29}$$

for the relative angular velocity between the body and the right wheel as well as

$$\omega_{\delta\mathrm{L}} = \dot{\phi}_{\mathrm{L}} - \dot{\alpha} \tag{3.30}$$

for the relative angular velocity between the body and the left wheel. The kinetic energy terms of the gears and the rotor then are given by

$$T_G = \frac{1}{2}I_{\mathrm{M}}(\dot{\alpha} + n_{\mathrm{WM}}\omega_{\delta\mathrm{R}})^2 + \frac{1}{2}I_{\mathrm{G}}(\dot{\alpha} - n_{\mathrm{WG}}\omega_{\delta\mathrm{R}})^2 \tag{3.31}$$
$$+ \frac{1}{2}I_{\mathrm{M}}(\dot{\alpha} + n_{\mathrm{WM}}\omega_{\delta\mathrm{L}})^2 + \frac{1}{2}I_{\mathrm{G}}(\dot{\alpha} - n_{\mathrm{WG}}\omega_{\delta\mathrm{L}})^2, \tag{3.32}$$

where I_{M} and I_{G} are the inertia of the rotor and the gear about their rotation axis respectively. One remark regarding the modeling of the gears and the rotor: This is a simplification of the kinetic energy terms of these four bodies since it has been decided to avoid the definition of position vectors of the centers of gravity of the gears and the rotors as well as the calculation of the translational and angular velocities as it has been done for the body and the wheels. As the mass m_{B} of the body includes the mass of the gears and the rotor, the corresponding translational energy terms can be dropped. Furthermore, the inertia I_{B} of the body includes the gears and the rotors as well. Thus, it has only to been taken care of the energy terms which arise from the difference between the body angular velocity and the gear and wheel angular velocity. Due to this, a correction term

$$T_{\mathrm{Corr}} = -I_{\mathrm{M}}\dot{\alpha}^2 - I_{\mathrm{G}}\dot{\alpha}^2 \tag{3.33}$$

is defined and added to the total kinetic energy. This treatment of the motor and gear energy has also been used in Anhalt [5], Albert et al. [2] and Delgado [23] but in Anhalt

[5] the coupling terms $\dot{\alpha} n_{\mathrm{WG}} \omega_{\delta}$ arising, if T_G is expanded, were missed[2]. This term compensates the energy terms in (3.31), which are already included by the robot's body kinetic energy term in (3.26). The potential energies of the mechanical parts are due to the gravitational potential of the body and is given by

$$V_B = \mathrm{g} m_{\mathrm{B}} \left(r_{\mathrm{W}} + l_{\mathrm{B}} \cos(\alpha) \right) \tag{3.34}$$

where g is the earth's gravity. As the wheels have a constant potential energy, their derivatives are zero in (3.2a) and thus are not required in the Lagrangian.

To incorporate the DC motors, which drive the wheels via the gearbox, the electric circuit and the interaction with the mechanical system, shown in Figure 3.2 for each motor, has to be modeled, too. The Euler-Lagrange approach was mostly developed to understand the dynamics of mechanical systems but these results can be generalized and applied to other physical systems as well. Wellstead [113] offers a great overview of how to model systems in different domains within the Lagrangian framework. Wells [112] presented, how the Lagrangian equations can be applied to electrical circuits interacting with mechanical systems and especially in the case of the TWIP, the Lagrangian formulation allows us to connect both subsystems easily and thus let us follow the steps as proposed by Wells [112]. In electric circuits, capacitors are considered as the potential energy where as inductors are handled in the kinetic energy terms. Resistors are treated as dissipative forces and sources can be treated in the potential terms or as external forces acting on the system. The configuration variables are the charges flowing through the different branches of the circuit over time. As there is only one branch in each motor circuit let us define the charges as q_{R} and q_{L} for the right and left motor respectively. The derivatives of the charges are the motor currents $\dot{q}_{\mathrm{R}} = i_{\mathrm{R}}$ and $\dot{q}_{\mathrm{L}} = i_{\mathrm{L}}$. Therefore, the kinetic energy terms of motor circuits is determined as

$$T_M = \frac{1}{2} L_{\mathrm{M}} (\dot{q}_{\mathrm{R}}^2 + \dot{q}_{\mathrm{L}}^2), \tag{3.35}$$

where L_{M} is the armature winding inductance. In addition, the potential energy due to the back-EMF in the motor circuits is given by

$$V_M = k_{\mathrm{E}} n_{\mathrm{WM}} \left(\omega_{\delta \mathrm{R}} q_{\mathrm{R}} + \omega_{\delta \mathrm{L}} q_{\mathrm{L}} \right) \tag{3.36}$$

where k_{E} is the motor back-EMF constant. To ease in the presentation of how the inputs act on the dynamical system and the transformation from a second-order model into an input-affine, first-order state space model, it has been decided to treat the external voltage source at the motor terminals not with $-u_{\mathrm{R}} q_{\mathrm{R}} - u_{\mathrm{L}} q_{\mathrm{L}}$ in the potential term but including them as external forces in the next subsections.

3.1.4 Dissipative and External Forces

The dissipative forces are friction forces between the robot body and the wheels as well as ohmic losses in the motor circuit. These forces arise from the gears, the bearing, and

[2]Based on the trajectory used for energy and power analysis in Subsection 3.1.11, the relative error of T_G due to the missed terms in Anhalt [5] compared to the presented one is calculated. At least for the trajectory used, the relative error is less than 0.1 % and thus the results in Anhalt [5] could be treated as valid.

the losses of the motors due to the resistance of the armature windings. Let us define the mechanical friction force vector as

$$F_{\text{fric}} := \begin{pmatrix} \tau_{FR} \\ \tau_{FL} \end{pmatrix} = \begin{pmatrix} d_{\text{V}}\omega_{\delta\text{R}} + d_{\text{C}} \tanh\left(d_0\omega_{\delta\text{R}}\right) \\ d_{\text{V}}\omega_{\delta\text{L}} + d_{\text{C}} \tanh\left(d_0\omega_{\delta\text{L}}\right) \end{pmatrix} \tag{3.37}$$

with the nonlinear friction torques τ_{FR} between the right wheel and the body as well as the torque τ_{FL} between the left wheel and the body. The friction losses of the gears and bearing are obtained by identifying the damping parameters d_{V}, d_{C} and d_0 of a typical Coulomb and viscous friction curve from experimental data (see Anhalt [5]).

The potential drops in the motor circuits due to the resistance R_{M} are assembled in the dissipative force vector F_{res} and given by

$$F_{\text{res}} := \begin{pmatrix} R_{\text{M}}\dot{q}_{\text{R}} \\ R_{\text{M}}\dot{q}_{\text{L}} \end{pmatrix} . \tag{3.38}$$

for the right and the left motor. The external force F_{ext} applied to the system is the voltage u_M (see Figure 3.2) available to the terminals of the motors. Let $u_{\text{R}}, u_{\text{L}}$ be the voltage supplied to the right and the left motor terminals respectively. Let us collect the external forces applied to the system in

$$F_{\text{ext}} := \begin{pmatrix} u_{\text{R}} \\ u_{\text{L}} \end{pmatrix} \tag{3.39}$$

which is the external forcing vector.

3.1.5 Nonholonomic Constraints

As mentioned before, the mechanical multi-body system is subject to nonholonomic constraints that arise due to the assumption of no slipping of the wheels. These no-slip conditions lead to two constraints for each wheel: no lateral sliding and pure rolling without slipping on the ground. Let $(x_{\text{R}}, y_{\text{R}}) \in \mathbb{R}^2$ be the right wheel's position and $(x_{\text{L}}, y_{\text{L}}) \in \mathbb{R}^2$ be the left wheel's position on the $x - y$ plane in the inertial coordinates where the wheels contact the ground as shown in Figure 3.3. Both positions are defined analogously to the positions and velocities of the centers of mass of the wheels in (3.18), (3.20) and (3.19), (3.21) but with the distance d_{W} instead of l_{W} with

$$_{I}p_R = \begin{pmatrix} x_{\text{B}} + d_{\text{W}}\sin(\theta) \\ y_{\text{B}} - d_{\text{W}}\cos(\theta) \\ r_{\text{W}} \end{pmatrix} , \quad _{I}v_R := \frac{\mathrm{d}}{\mathrm{d}t}\left(_{I}p_R\right) = \begin{pmatrix} \dot{x}_{\text{B}} + d_{\text{W}}\dot{\theta}\cos(\theta) \\ \dot{y}_{\text{B}} + d_{\text{W}}\dot{\theta}\sin(\theta) \\ 0 \end{pmatrix} \tag{3.40}$$

for the right wheel and

$$_{I}p_L = \begin{pmatrix} x_{\text{B}} - d_{\text{W}}\sin(\theta) \\ y_{\text{B}} + d_{\text{W}}\cos(\theta) \\ r_{\text{W}} \end{pmatrix} , \quad _{I}v_L = \frac{\mathrm{d}}{\mathrm{d}t}\left(_{I}p_L\right) = \begin{pmatrix} \dot{x}_{\text{B}} - d_{\text{W}}\dot{\theta}\cos(\theta) \\ \dot{y}_{\text{B}} - d_{\text{W}}\dot{\theta}\sin(\theta) \\ 0 \end{pmatrix} \tag{3.41}$$

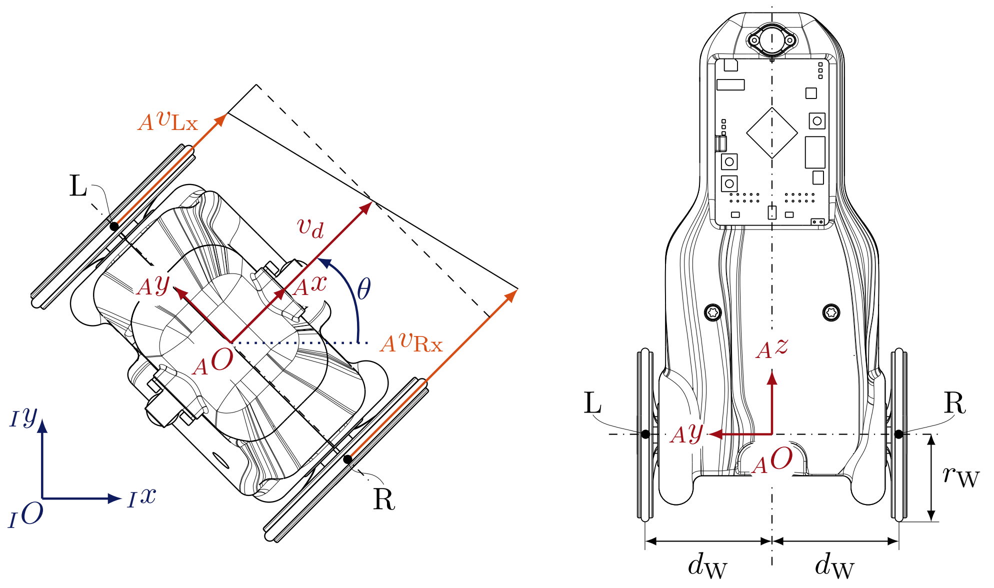


Figure 3.3: Nonholonomic constrains of the TWIP

for the left wheel. This differs from all other published approaches (Pathak et al. [84], Kim and Kwon [56], Delgado [23], Anhalt [5], Albert et al. [2]): In general the points R and L with the distance d_{W} where the wheels contact the ground do not have to be equal to the distance l_{W} of the centers of gravity C_R and C_L of the wheels. In the wheel axis-fixed frame A, the direction of the permissible as well as the impermissible velocities are colinear with the coordinate axes ${}_A x$ and ${}_A y$. Thus, let us transform the velocities of the points R and L from the inertial frame to the axis-fixed frame with

$$\begin{pmatrix} {}_A v_{\mathrm{Rx}} \\ {}_A v_{\mathrm{Ry}} \\ {}_A v_{\mathrm{Rz}} \end{pmatrix} = {}^I R_{AI}^{\mathrm{T}} v_R = \begin{pmatrix} \dot{x}_{\mathrm{B}} \cos(\theta) + \dot{y}_{\mathrm{B}} \sin(\theta) + d_{\mathrm{W}} \dot{\theta} \\ -\dot{x}_{\mathrm{B}} \sin(\theta) + \dot{y}_{\mathrm{B}} \cos(\theta) \\ 0 \end{pmatrix} \tag{3.42}$$

for the right wheel and

$$\begin{pmatrix} {}_A v_{\mathrm{Lx}} \\ {}_A v_{\mathrm{Ly}} \\ {}_A v_{\mathrm{Lz}} \end{pmatrix} = {}^I R_{AI}^{\mathrm{T}} v_L = \begin{pmatrix} \dot{x}_{\mathrm{B}} \cos(\theta) + \dot{y}_{\mathrm{B}} \sin(\theta) - d_{\mathrm{W}} \dot{\theta} \\ -\dot{x}_{\mathrm{B}} \sin(\theta) + \dot{y}_{\mathrm{B}} \cos(\theta) \\ 0 \end{pmatrix} \tag{3.43}$$

for the left wheel. As illustrated in Figure 3.3 the pure rolling motions of the wheels take place in the direction of the ${}_A x$ axis and is thus given by

$$ {}_A v_{\mathrm{Rx}} = \dot{x}_{\mathrm{B}} \cos(\theta) + \dot{y}_{\mathrm{B}} \sin(\theta) + d_{\mathrm{W}} \dot{\theta} = r_{\mathrm{W}} \dot{\phi}_{\mathrm{R}}, \tag{3.44a}$$

$$ {}_A v_{\mathrm{Lx}} = \dot{x}_{\mathrm{B}} \cos(\theta) + \dot{y}_{\mathrm{B}} \sin(\theta) - d_{\mathrm{W}} \dot{\theta} = r_{\mathrm{W}} \dot{\phi}_{\mathrm{L}}. \tag{3.44b}$$

Furthermore, without slip, no motion takes place along the direction of the ${}_A y$ axis. Thus, the no side-slip constraints for both wheels are given by

$$\begin{aligned} {}_A v_{\mathrm{Ry}} &= -\dot{x}_{\mathrm{B}} \sin(\theta) + \dot{y}_{\mathrm{B}} \cos(\theta) = 0, \\ {}_A v_{\mathrm{Ly}} &= -\dot{x}_{\mathrm{B}} \sin(\theta) + \dot{y}_{\mathrm{B}} \cos(\theta) = 0, \end{aligned} \tag{3.45}$$

which are two equal constraint equations. The no side-slip constraints (3.45) and pure rolling constraints (3.44) can now be written as a matrix-vector product (3.2b) to include the nonholonomic restrictions in (3.2a).

3.1.6 Model with Current Dynamics

Based on the results of the previous sections, where the kinetic and potential energy terms, as well as the nonholonomic constraint equations, have been defined, the model with current dynamics of the TWIP is derived in the following.

This model will include the full current dynamics and the configuration and state variables will be chosen in a way, which is commonly used. Since the TWIP stands on the ground, two of the six degrees of freedom (DoFs) are lost. As the wheels are able to rotate independently around the wheel axis, two DoFs are gained. Thus, six configuration variables for the mechanical system are required. By including the full current dynamics of the right and the left motor, two additional variables for them have to be considered. Altogether, this leads to eight configuration variables in total. For the calculation of the Lagrange-Euler equation the configuration variables of the TWIP are introduced, in probably the most intuitive way, as follows:

- $(x_\mathrm{B}, y_\mathrm{B}) \in \mathbb{R}^2$ for the distance to the origin ${}_AO$ of the axis-fixed frame and the origin ${}_BO$ of the body-fixed frame in the horizontal plane with respect to the inertial frame origin ${}_IO$. The lower right index B is added to avoid confusion with the state vector x of the robot.
- $\theta \in \mathbb{S}$ for the heading (or orientation) angle, defined as the angle between the ${}_Ix$-axis of the inertial frame I and the ${}_Ax$ axis of the axis-fixed frame A. This corresponds to a rotation around the axis ${}_Iz$.
- $\alpha \in \mathbb{S}$ for the tilt (or pitch) angle of the body, defined as the angle between the ${}_Az$ axis of the axis-fixed frame A and the axis ${}_Bz$ of the body-fixed frame B. This corresponds to a rotation around the axis ${}_Ay$.
- $\phi_\mathrm{R} \in \mathbb{S}$ and $\phi_\mathrm{L} \in \mathbb{S}$ for the rotation angle of the left and right wheel around the ${}_Ay$ axis.
- $q_\mathrm{R} \in \mathbb{R}$ and $q_\mathrm{L} \in \mathbb{R}$ for the charges on the right and left motor terminals. Their time derivatives are the currents i_R, i_L flowing through the right and left electric motors in the robot's body to generate torque between the wheels and the body.

Based on this choice, the configuration space of the system is $Q_\mathrm{C} := \mathbb{R}^2 \times \mathbb{S} \times \mathbb{S} \times \mathbb{S} \times \mathbb{S} \times \mathbb{R} \times \mathbb{R}$, with

$$\mathfrak{q}_c = (x_\mathrm{B}, y_\mathrm{B}, \theta, \alpha, \phi_\mathrm{R}, \phi_\mathrm{L}, q_\mathrm{R}, q_\mathrm{L})^\mathrm{T} \in Q_\mathrm{C} \ , \tag{3.46}$$

as the configuration variables vector. The time derivative of the configuration vector is

$$\dot{\mathfrak{q}}_c = \left(\dot{x}_\mathrm{B}, \dot{y}_\mathrm{B}, \dot{\theta}, \dot{\alpha}, \dot{\phi}_\mathrm{R}, \dot{\phi}_\mathrm{L}, \dot{q}_\mathrm{R}, \dot{q}_\mathrm{L}\right)^\mathrm{T} \ . \tag{3.47}$$

As all dynamic components are considered, the total kinetic energy of the system is given by

$$T(\mathfrak{q}_c, \dot{\mathfrak{q}}_c) = T_B + T_W + T_G + T_M + T_\mathrm{Corr}. \tag{3.48}$$

In the same way, the potential energy of the system

$$V(\mathfrak{q}_c, \dot{\mathfrak{q}}_c) = V_B + V_M \tag{3.49}$$

includes the gravitational potential energy of the body and the potential energy terms due to the back-EMF in the motor circuits. The dissipative forces are the friction torques in F_{fric} between the robot body and the wheels as well as ohmic losses collected in F_{res} arising in the motor circuits. As the friction torques are acting on the body and as counter torque on the wheels also, they enter the α-direction as well as the ϕ_{R}- and ϕ_{L}-directions but with different signs. The total dissipative force applied to the system along the $\mathfrak{q}_c$-coordinates is then given by

$$\mathcal{F}_{\text{dis}}(\dot{\mathfrak{q}}_c) = \begin{pmatrix} 0_{3\times1} \\ \tau_{FR} + \tau_{FL} \\ -\tau_{FR} \\ -\tau_{FL} \\ -R_{\text{M}}\dot{q}_{\text{R}} \\ -R_{\text{M}}\dot{q}_{\text{L}} \end{pmatrix} = \begin{pmatrix} 0_{3\times2} & 0_{3\times2} \\ 1_{1\times2} & 0_{1\times2} \\ -I_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & -I_{2\times2} \end{pmatrix} \begin{pmatrix} F_{\text{fric}} \\ F_{\text{res}} \end{pmatrix}. \tag{3.50}$$

The external force $\mathcal{F}_{\text{ext}}$ applied to the system is the voltage available to terminals of the motors. Let $u_{\text{R}}, u_{\text{L}}$ be the voltage supplied to right and left motors respectively. The external force applied along the chosen $\mathfrak{q}_c$-coordinates is given by

$$\mathcal{F}_{\text{ext}}(u_{\text{R}}, u_{\text{L}}) = \underbrace{\begin{pmatrix} 0_{6\times2} \\ I_{2\times2} \end{pmatrix}}_{\mathcal{E}} \underbrace{\begin{pmatrix} u_{\text{R}} \\ u_{\text{L}} \end{pmatrix}}_{u} \tag{3.51}$$

with $\mathcal{E}$ as the external forcing matrix and u as the system input.

As already mentioned before and can be seen in $\mathcal{F}_{\text{dis}}$, the friction torques act on the body as well as on the wheels but with the opposite sign. Kim and Kwon [56] pointed out, that this principle of action and reaction between the body and the wheels has been disregarded in many other publications, which resulted in bad model accuracy and false statements about the motion properties.

The no side-slip constraints (3.45) and pure rolling constraints (3.44) can now be as written as a matrix-vector product

$$\underbrace{\begin{pmatrix} -\sin(\theta) & \cos(\theta) & 0 & 0 & 0 & 0 & 0 & 0 \\ \cos(\theta) & \sin(\theta) & d_{\text{W}} & 0 & -r_{\text{W}} & 0 & 0 & 0 \\ \cos(\theta) & \sin(\theta) & -d_{\text{W}} & 0 & 0 & -r_{\text{W}} & 0 & 0 \end{pmatrix}}_{A(\mathfrak{q}_c)} \dot{\mathfrak{q}}_c = 0 \tag{3.52}$$

based on the choice of the configuration vector $\mathfrak{q}_c$ and its time derivative $\dot{\mathfrak{q}}_c$.

As the system dynamics is subject to nonholonomic constraints, a mapping is applied to the model from generalized coordinates into minimal coordinates without nonholonomic constraints. For this purpose, let us define two new variables:

$$v_\theta = \frac{r_{\text{W}}}{2d_{\text{W}}} \left(\dot{\phi}_{\text{R}} - \dot{\phi}_{\text{L}}\right) \in \mathbb{R} \tag{3.53}$$

for the heading (or orientation) angle velocity around the ${}_{I}z$ axis. This equation is a result of the subtraction of (3.44a) from (3.44a). Furthermore, let us define

$$v_d = \frac{r_{\mathrm{W}}}{2}\left(\dot{\phi}_{\mathrm{R}} + \dot{\phi}_{\mathrm{L}}\right) \in \mathbb{R} \tag{3.54}$$

for the forward velocity v_d along the ${}_{A}x$ axis. v_d is shown in Figure 3.3 and is derived by taking the average forward speed between the two wheels, calculated with (3.44a) and (3.44a). Furthermore, to ease the notation and for insight let us define additional variables:

- $v_\alpha := \dot{\alpha} \in \mathbb{R}$ for the tilt (or pitch) angle velocity
- $i_{\mathrm{R}} := \dot{q}_{\mathrm{R}} \in \mathbb{R}$ and $i_{\mathrm{L}} := \dot{q}_{\mathrm{L}} \in \mathbb{R}$ for the right and left currents, flowing through the respective electric motors.

To define the equations of motion of the system in minimal coordinates without nonholonomic constraints in (3.2a) the method presented by Pathak et al. [84] is used. Let us choose the matrix

$$S(\mathfrak{q}_c) = \begin{pmatrix} 0 & \cos(\theta) & 0 & 0 & 0 \\ 0 & \sin(\theta) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{r_{\mathrm{W}}} & \frac{d_{\mathrm{W}}}{r_{\mathrm{W}}} & 0 & 0 \\ 0 & \frac{1}{r_{\mathrm{W}}} & -\frac{d_{\mathrm{W}}}{r_{\mathrm{W}}} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.55}$$

such that it lies in the null space of the matrix $A(\mathfrak{q}_c)$ and thus $A(\mathfrak{q}_c)S(\mathfrak{q}_c) = 0$ holds. Moreover, (3.55) is a mapping from generalized coordinates into minimal coordinates and based on our choice of $S(q)$ is given with

$$\nu_c := \left(v_\alpha, v_d, v_\theta, i_{\mathrm{R}}, i_{\mathrm{L}}\right)^{\mathrm{T}} \tag{3.56}$$

the velocity vector with the permissible motions. Notice, the choice of $S(q)$ is not unique and thus there is an infinite number of possible ν. In Subsection 3.1.10 alternative combinations of S and ν are introduced which also fulfill the condition $A(\mathfrak{q}_c)S(\mathfrak{q}_c) = 0$. Finally, let us neglect the kinematics for the states of angles of the wheels $(\phi_{\mathrm{L}}, \phi_{\mathrm{R}})$ as well as the motor terminal charges $(q_{\mathrm{L}}, q_{\mathrm{R}})$ in S, since they are commonly not from interest, at least for control purposes. Thus, let us set up the reduced matrix S_r by removing the last four rows in S. Using the introduced steps from (3.1) to (3.6) with (3.48) to (3.55), we get the system dynamics on the reduced space in terms of the new states

$$x_c := (\mathfrak{q}_{\mathrm{r,c}}^{\mathrm{T}}, \nu_c^{\mathrm{T}})^{\mathrm{T}} = ((x_{\mathrm{B}}, y_{\mathrm{B}}, \theta, \alpha), (v_\alpha, v_d, v_\theta, i_{\mathrm{R}}, i_{\mathrm{L}}))^{\mathrm{T}} \tag{3.57}$$

as

$$\dot{x}_c := \begin{pmatrix} S_r \nu_c \\ -\left(S^{\mathrm{T}} M S\right)^{-1} S^{\mathrm{T}} \left(M\dot{S}\nu_c + K + P - \mathcal{F}_{\mathrm{dis}} - \underbrace{\mathcal{E}u}_{\mathcal{F}_{\mathrm{ext}}}\right) \end{pmatrix}. \tag{3.58}$$

Since

$$-\left(S^{\mathrm{T}}MS\right)^{-1}S^{\mathrm{T}}\left(-\mathcal{E}\right)=\frac{1}{L_{\mathrm{M}}}\begin{pmatrix}0_{3x2}\\I_{2x2}\end{pmatrix}\tag{3.59}$$

is constant, the system dynamics can be split into a state-dependent, nonlinear drift vector field $f(x)$ and a constant input matrix G:

$$\dot{x}_c=\underbrace{\begin{pmatrix}S_r\nu_c\\-\left(S^{\mathrm{T}}MS\right)^{-1}S^{\mathrm{T}}\left(M\dot{S}\nu_c+K+P-\mathcal{F}_{\mathrm{dis}}\right)\end{pmatrix}}_{f_c(x_c)}+\underbrace{\frac{1}{L_{\mathrm{M}}}\begin{pmatrix}0_{4x2}\\0_{3x2}\\I_{2x2}\end{pmatrix}u}_{G_cu}\ .\tag{3.60}$$

This model will be called c-model in the following.

3.1.7 Model without Current Dynamics

As could be seen in the previous section, the current dynamics of the motors are treated by two states and differential equations in the state space model, if they are considered in the model of the TWIP. Since the current dynamics are quite fast compared to the other dynamics of the TWIP, as shown later based on the linearized model in Section 3.3, a common approach is to neglect the current dynamics in the model. Thus, a simplified derivation of the TWIP model without current dynamics is introduced in the following.

Let us start with the definition of the configuration space, the configuration variables and the Lagrangian of the TWIP model without the charge variables. Therefore, the configuration space $Q_n := \mathbb{R}^2\times\mathbb{S}\times\mathbb{S}\times\mathbb{S}\times\mathbb{S}$ for the system is defined with

$$\mathfrak{q}_n=\left(x_{\mathrm{B}},y_{\mathrm{B}},\theta,\alpha,\phi_{\mathrm{R}},\phi_{\mathrm{L}}\right)^{\mathrm{T}}\in Q_n\tag{3.61}$$

as the configuration variables vector. The time derivative of the configuration vector is then given as

$$\dot{\mathfrak{q}}_n=\left(\dot{x}_{\mathrm{B}},\dot{y}_{\mathrm{B}},\dot{\theta},\dot{\alpha},\dot{\phi}_{\mathrm{R}},\dot{\phi}_{\mathrm{L}}\right)^{\mathrm{T}}\tag{3.62}$$

and does not include the currents as the derivative of the charges anymore.

Moreover, let us drop the term T_M for the kinetic electric energy of the motor to get the total kinetic energy

$$T(\mathfrak{q}_n,\dot{\mathfrak{q}}_n)=T_B+T_W+T_G+T_{\mathrm{Corr}}\tag{3.63}$$

as well as total potential energy

$$V(\mathfrak{q}_n)=\mathrm{g}m_{\mathrm{B}}\left(r_{\mathrm{W}}+l_{\mathrm{B}}\cos(\alpha)\right)\ ,\tag{3.64}$$

where the electric potential energy term V_M of the motor is also not included.

One problem arises now: we lose the input $(u_{\mathrm{R}},u_{\mathrm{L}})$ in the Euler-Lagrange equations as well as the torques applied by the motors to the robot. Thus, we have to derive the static current and torque equations and include them as external forces to the Euler-Lagrange equations. The straightforward approach is, to apply Kirchhoff's second law for the mesh Σ_M in the electric circuit of the motors, shown in Figure 3.2. Then, in the next step $\frac{\mathrm{d}}{\mathrm{d}t} i_M$ is set to zero and the static motor current equation is gained. Since the motor torque is proportional to the motor current i_M with the motor torque constant k_M, a equation from voltage input to motor torque can be formulated which can be included in the Euler-Lagrange equations as an external force.

Contrary, a different approach is chosen directly based on the Lagrangian of the motor current and in the same breath it can be shown, that the definition of the electric kinetic and potential energy in (3.35) and (3.36) as well as the friction forces (3.38) and external forces (3.39) have been correct. As the equations are similar for both motors, let us use $u_M, i_M, \omega_M, \tau_M, \omega_\delta, \tau_\delta$ and drop the specific incidences R and L in the following derivations.

The Lagrangian of one motor circuit based on (3.35) and (3.36), is given by

$$L_I = \underbrace{\frac{1}{2} L_\mathrm{M} \dot{q}_M^2}_{T} - \underbrace{k_\mathrm{E} n_\mathrm{WM} \omega_\delta q_M}_{V} \ . \tag{3.65}$$

Let us calculate the Euler-Lagrange equations and include the right-hand side with the friction forces and external forces based on (3.38) and (3.39) for one motor:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L_I}{\partial \dot{q}_M}\right) - \frac{\partial L_I}{\partial q_M} = \frac{\mathrm{d}}{\mathrm{d}t}\left(L_\mathrm{M} \dot{q}_M\right) + k_\mathrm{E} n_\mathrm{WM} \omega_\delta = -R_\mathrm{M} \dot{q}_M + u_M \ . \tag{3.66}$$

In the next step, (3.66) is rearranged and $\dot{q}_M$ replaced with i_M. Therefore, the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t} i_M = -\frac{R_\mathrm{M}}{L_\mathrm{M}} i_M - \frac{k_\mathrm{E} n_\mathrm{WM} \omega_\delta}{L_\mathrm{M}} + \frac{1}{L_\mathrm{M}} u_M \tag{3.67}$$

is derived for the motor current dependent on ω_δ, i_M and u_M. This differential equation is equal to the one we would get if we apply Kirchhoff's second law for the mesh Σ_M in the electric circuit of the motors, shown in Figure 3.2. In the next step, to neglect the current dynamics, we set $\frac{\mathrm{d}}{\mathrm{d}t} i_M = 0$ and get the static motor current equation

$$i_M = -\frac{k_\mathrm{E} n_\mathrm{WM}}{R_\mathrm{M}} \omega_\delta + \frac{1}{R_\mathrm{M}} u_M \tag{3.68}$$

which describes the steady state current of the motor in dependence of the relative angular velocity between the body and the wheel as well as the terminal voltage. To include the torques applied by the motors to the body and the wheels, the motor torques as a function of u_M and ω_δ are derived in the next steps. The motor torque τ_M is the product of the motor torque constant k_M and the motor current i_M. Let us include the gear stage ratio n_WM to get the torque τ_δ between the body and the wheel and replace i_M with (3.68), to derive the motor torque formula

$$\begin{aligned} \tau_\delta &= n_\mathrm{WM} \tau_M = n_\mathrm{WM} k_\mathrm{M} i_M \\ &= \underbrace{-\frac{k_\mathrm{E} k_\mathrm{M} n_\mathrm{WM}^2}{R_\mathrm{M}} \omega_\delta}_{\text{dissipative}} + \underbrace{\frac{n_\mathrm{WM} k_\mathrm{M}}{R_\mathrm{M}} u_M}_{\text{external}} \end{aligned} \tag{3.69}$$

Inspecting the gained motor torque equation, two terms can be recognized: one term which is dissipative and depends on ω_δ as well as an externally forced term by the terminal voltage u_M of the motor. In the following, u_M is replaced by u_R and u_L as well as ω_δ with $\omega_{\delta\mathrm{R}}$ and $\omega_{\delta\mathrm{L}}$ to gain torque equations for the right and the left motor.

Similar to the model with current dynamics, the dissipative forces are again friction forces between the robot body and the wheels as well as ohmic losses in the motor circuit. Let us sum up the torque terms from (3.69) and the nonlinear friction torque introduced in (3.37) to get the dissipative torques τ_{DR} and τ_{DL} on each side of the robot. Finally, the dissipative torques are used to get the total dissipative force

$$\mathcal{F}_\mathrm{dis}(\mathfrak{q}_n, \dot{\mathfrak{q}}_n) = \begin{pmatrix} 0_{3\times 2} \\ 1_{1\times 2} \\ -I_{2\times 2} \end{pmatrix} \underbrace{\left(\begin{pmatrix} \tau_{FR} \\ \tau_{FL} \end{pmatrix} + \frac{k_\mathrm{E} k_\mathrm{M} n_\mathrm{WM}^2}{R_\mathrm{M}} \begin{pmatrix} \omega_{\delta\mathrm{R}} \\ \omega_{\delta\mathrm{L}} \end{pmatrix} \right)}_{\begin{pmatrix} \tau_{DR} \\ \tau_{DL} \end{pmatrix}} . \tag{3.70}$$

along the $\mathfrak{q}_n$-coordinates applied to the system. The total external force required for the Euler-Lagrange equations is given by

$$\mathcal{F}_\mathrm{ext}\,(u_\mathrm{R}, u_\mathrm{L}) = \underbrace{-\frac{n_\mathrm{WM} k_\mathrm{M}}{R_\mathrm{M}} \begin{pmatrix} 0_{3\times 2} \\ 1_{1\times 2} \\ -I_{2\times 2} \end{pmatrix}}_{\mathcal{E}} \underbrace{\begin{pmatrix} u_\mathrm{R} \\ u_\mathrm{L} \end{pmatrix}}_{u} \tag{3.71}$$

if the second term in (3.69) is applied along the $\mathfrak{q}_n$-coordinates for each motor. Herein $\mathcal{E}$ is the external forcing matrix and u the system input. By inspecting the dissipative force in (3.70) and the external force in (3.71), it can be recognized that the dissipative terms can be compensated by the external forces through the input u. This offers the possibility of a full friction compensation as used in Delgado [23] if the current dynamics of the TWIP are neglected. In Section 5.2 a friction compensation which compensates the nonlinear elements of (3.37) is introduced to increase the domain with a high congruence between the linear model and the nonlinear model.

Finally, the no side-slip constraints (3.45) and pure rolling constraints (3.44) can be formulated with

$$\underbrace{\begin{pmatrix} -\sin(\theta) & \cos(\theta) & 0 & 0 & 0 & 0 \\ \cos(\theta) & \sin(\theta) & d_\mathrm{W} & 0 & -r_\mathrm{W} & 0 \\ \cos(\theta) & \sin(\theta) & -d_\mathrm{W} & 0 & 0 & -r_\mathrm{W} \end{pmatrix}}_{A(\mathfrak{q}_n)} \dot{\mathfrak{q}}_n = 0 \tag{3.72}$$

similar as it has been done modeling the TWIP with current dynamics.

The mapping (3.3) from generalized coordinates in minimal coordinates is then defined as

$$\nu_n := \left(v_\alpha, v_d, v_\theta \right)^\mathrm{T} . \tag{3.73}$$

and the matrix

$$S(\mathfrak{q}_n) = \begin{pmatrix} 0 & \cos(\theta) & 0 \\ 0 & \sin(\theta) & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & \frac{1}{r_\mathrm{W}} & \frac{d_\mathrm{W}}{r_\mathrm{W}} \\ 0 & \frac{1}{r_\mathrm{W}} & -\frac{d_\mathrm{W}}{r_\mathrm{W}} \end{pmatrix} \tag{3.74}$$

is chosen again, such that it lies in the null space of the matrix $A(\mathfrak{q}_n)$. For the model without current dynamics, the reduced matrix S_r is gained by removing the last two rows in S for the absolute wheel angles.

Using the introduced steps from (3.1) to (3.6) with (3.63) to (3.64) and (3.70) to (3.74), the system dynamics on the reduced space in terms of the new state vector

$$x_n := (\mathfrak{q}_{r,n}^\mathrm{T}, \nu_n^\mathrm{T})^\mathrm{T} = ((x_\mathrm{B}, y_\mathrm{B}, \theta, \alpha), (v_\alpha, v_d, v_\theta))^\mathrm{T} \tag{3.75}$$

are derived as

$$\dot{x}_n := \begin{pmatrix} S\nu_n \\ -\left(S^\mathrm{T} M S\right)^{-1} S^\mathrm{T} \left(M\dot{S}\nu_n + K + P - \mathcal{F}_\mathrm{dis} - \mathcal{E}u\right) \end{pmatrix} . \tag{3.76}$$

Moreover, the system dynamics can be split into a drift and an input term. Contrary to term (3.59) of the model with current dynamics, now the term

$$\left(S^\mathrm{T} M S\right)^{-1} S^\mathrm{T} \mathcal{E} = g_n(x_n) \tag{3.77}$$

is not constant since it depends on the tilt angle α. Finally, the control-affine, nonlinear state space model is given as

$$\begin{aligned} \dot{x}_n = & \underbrace{\begin{pmatrix} S_\mathrm{r}\nu_n \\ -\left(S^\mathrm{T} M S\right)^{-1} S^\mathrm{T} \left(M\dot{S}\nu_n + K + P - \mathcal{F}_\mathrm{dis}\right) \end{pmatrix}}_{f_n(x_n)} \\ & + \underbrace{\left(S^\mathrm{T} M S\right)^{-1} S^\mathrm{T} \mathcal{E}}_{g_n(x_n)} u \, . \end{aligned} \tag{3.78}$$

but in this case a state-depended, nonlinear drift vector field $f_n(x_n)$ as well as a state depended, nonlinear input field $g_n(x_n)$ is received. This model is called n-model in the following.

3.1.8 Model on Ground without Current Dynamics

Besides the mode, where the TWIP is balancing, it is also possible to operate the robot in a lying state on ground. In this case, the robot dynamics are stable and further simplified, since the state for α and $\dot{\alpha}$ are not required. Let us derive the TWIP model on ground without current dynamics, in the following.

Let us start with the definition of the configuration space, the configuration variables, and the Lagrangian of the TWIP model without the tilt and charge variables. Therefore, the configuration space is defined by $Q_\mathrm{g} := \mathbb{R}^2 \times \mathbb{S} \times \mathbb{S} \times \mathbb{S}$ for the system with

$$\mathfrak{q}_g = (x_\mathrm{B}, y_\mathrm{B}, \theta, \phi_\mathrm{R}, \phi_\mathrm{L})^\mathrm{T} \in Q_g \tag{3.79}$$

as the configuration variables vector. The time derivative of the configuration vector is then given as

$$\dot{\mathfrak{q}}_g = \left(\dot{x}_\mathrm{B}, \dot{y}_\mathrm{B}, \dot{\theta}, \dot{\phi}_\mathrm{R}, \dot{\phi}_\mathrm{L}\right)^\mathrm{T} . \tag{3.80}$$

Let us use the kinetic and potential energy terms as defined in Eqs. (3.26), (3.27) and (3.31), but we set $\alpha = \frac{\pi}{2}$ and $\dot{\alpha} = 0$ in all terms and get

$$T(\mathfrak{q}_g, \dot{\mathfrak{q}}_g) = T_B + T_W + T_G \tag{3.81}$$

as well as total potential energy

$$V(\mathfrak{q}_g) = \mathrm{g} m_\mathrm{B} r_\mathrm{W} = \text{const.} . \tag{3.82}$$

The dissipative forces are set up in the same fashion as in (3.70) for the model without current, but without the entries for $\dot{\alpha}$, since we assume the robot to drive on ground with $\alpha = \frac{\pi}{2}$. They are given with

$$\mathcal{F}_\mathrm{dis}(\mathfrak{q}_g, \dot{\mathfrak{q}}_g) = \begin{pmatrix} 0_{3\times2} \\ -I_{2\times2} \end{pmatrix} \left(\begin{pmatrix} \tau_{FR} \\ \tau_{FL} \end{pmatrix} + \frac{k_\mathrm{E} k_\mathrm{M} n_\mathrm{WM}^2}{R_\mathrm{M}} \begin{pmatrix} \omega_{\delta\mathrm{R}} \\ \omega_{\delta\mathrm{L}} \end{pmatrix} \right) . \tag{3.83}$$

along the $\mathfrak{q}_g$-coordinates. Analogously to the dissipative force, the external force is given by

$$\mathcal{F}_\mathrm{ext}\left(u_\mathrm{R}, u_\mathrm{L}\right) = \underbrace{-\frac{n_\mathrm{WM} k_\mathrm{M}}{R_\mathrm{M}} \begin{pmatrix} 0_{3\times2} \\ -I_{2\times2} \end{pmatrix}}_{\mathcal{E}} \underbrace{\begin{pmatrix} u_\mathrm{R} \\ u_\mathrm{L} \end{pmatrix}}_{u} . \tag{3.84}$$

Finally, the no side-slip constraints (3.45) and pure rolling constraints (3.44) can be formulated with

$$\underbrace{\begin{pmatrix} -\sin(\theta) & \cos(\theta) & 0 & 0 & 0 \\ \cos(\theta) & \sin(\theta) & d_\mathrm{W} & -r_\mathrm{W} & 0 \\ \cos(\theta) & \sin(\theta) & -d_\mathrm{W} & 0 & -r_\mathrm{W} \end{pmatrix}}_{A(\mathfrak{q}_g)} \dot{\mathfrak{q}}_g = 0 \tag{3.85}$$

similar to what has been done for the balancing TWIP models.

The mapping (3.3) from generalized coordinates in minimal coordinates is defined with

$$\nu_g := \left(v_d, v_\theta\right)^\mathrm{T} . \tag{3.86}$$

and the matrix

$$S(\mathfrak{q}_g) = \begin{pmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \\ \frac{1}{r_\mathrm{W}} & \frac{d_\mathrm{W}}{r_\mathrm{W}} \\ \frac{1}{r_\mathrm{W}} & -\frac{d_\mathrm{W}}{r_\mathrm{W}} \end{pmatrix} \tag{3.87}$$

is chosen, such that it lies in the null space of the matrix $A(\mathfrak{q}_g)$ and the reduced matrix S_r is gained by removing the last two rows in S for the absolute wheel angles.

Using the introduced steps from (3.1) to (3.6) with (3.81) to (3.82) and (3.83) to (3.87), the system dynamics on the reduced space in terms of the new state vector

$$x_\mathrm{g} := (\mathfrak{q}_{\mathrm{r,c}}^\mathrm{T}, \nu_\mathrm{g}^\mathrm{T})^\mathrm{T} = ((x_\mathrm{B}, y_\mathrm{B}, \theta), (v_d, v_\theta))^\mathrm{T} \tag{3.88}$$

are defined as

$$\dot{x}_\mathrm{g} := \begin{pmatrix} S_r \nu_\mathrm{g} \\ -\left(S^\mathrm{T} M S\right)^{-1} S^\mathrm{T} \left(M \dot{S} \nu_\mathrm{g} + K + P - \mathcal{F}_\mathrm{dis} - \underbrace{\mathcal{E} u}_{\mathcal{F}_\mathrm{ext}}\right) \end{pmatrix} . \tag{3.89}$$

Since

$$-\left(S^\mathrm{T} M S\right)^{-1} S^\mathrm{T} \left(-\mathcal{E}\right) = \text{const.} \tag{3.90}$$

is constant, the system dynamics can be split into state-dependent, nonlinear drift vector field $f(x)$ and a constant input matrix G:

$$\dot{x}_\mathrm{g} = \underbrace{\begin{pmatrix} S_r \nu_\mathrm{g} \\ -\left(S^\mathrm{T} M S\right)^{-1} S^\mathrm{T} \left(M \dot{S} \nu_\mathrm{c} + K + P - \mathcal{F}_\mathrm{dis}\right) \end{pmatrix}}_{f_\mathrm{c}(x_\mathrm{g})} + \underbrace{\begin{pmatrix} 0_{3x2} \\ \left(S^\mathrm{T} M S\right)^{-1} S^\mathrm{T} \mathcal{E} \end{pmatrix} u}_{G_\mathrm{g} u} . \tag{3.91}$$

This model is called g-model in the following.

3.1.9 State and Input Limitations

The models of the TWIP presented in Subsections 3.1.6 to 3.1.8 cover the nonlinear dynamics, but physical state and input limits have not been incorporated into the model yet. This gap will be closed in this section.

Firstly, the tilt angle α is limited by the floor to $\pm 90°$. Furthermore, the motor currents i_R, i_L saturate at $\pm 3\,\mathrm{A}$ due to the overload protection of the H-bridges. Finally, the input voltages u_L, u_R are limited by the battery used. Let us consider, that a voltage of at least $7.4\,\mathrm{V}$ is always available and thus the inputs saturate at $\pm 7.4\,\mathrm{V}$.

These limits are part of the nonlinear model and part of the physics of the TWIP. Consequently, these limits are also included in the simulation environment introduced in Section 2.6. In addition, if the limits are reached, they have a drastic effect on the system dynamics and thus they have to be considered during control design. Moreover, they also have to be considered during stability analysis in Section 5.5, to avoid trajectories in the estimated DoA that will violate these constraints.

3.1.10 Alternative System Representations

The models of the TWIP presented in Subsections 3.1.6 to 3.1.8, use the probably most intuitive choice of state variables for the nonlinear state space model. Since there is an infinite number of possibilities of choices of the state variables in the state vector, one more state representation for the TWIP is presented, which might ease the control and observer synthesis by unfolding the structural details of the model. For relative motion between the body and the wheels, the variables $\omega_{\delta\mathrm{R}}$ and $\omega_{\delta\mathrm{L}}$ are defined in (3.29) and (3.30). Let us use the equations for v_d and v_θ, given in (3.53) and (3.54) as well as (3.29) and (3.30) and propose the state transformation

$$x_{\omega,c} = \underbrace{\begin{pmatrix} I_{4\times4} & 0_{4\times1} & 0_{4\times1} & 0_{4\times1} & 0_{4\times2} \\ 0_{1\times4} & 1 & 0 & 0 & 0_{1\times2} \\ 0_{1\times4} & -1 & \frac{1}{r_\mathrm{W}} & \frac{d_\mathrm{W}}{r_\mathrm{W}} & 0_{1\times2} \\ 0_{1\times4} & -1 & \frac{1}{r_\mathrm{W}} & -\frac{d_\mathrm{W}}{r_\mathrm{W}} & 0_{1\times2} \\ 0_{2\times4} & 0_{2\times1} & 0_{2\times1} & 0_{2\times1} & I_{2\times2} \end{pmatrix}}_{T_{\omega,c}} x_c \tag{3.92}$$

$$x_{\omega,n} = \underbrace{\begin{pmatrix} I_{4\times4} & 0_{4\times1} & 0_{4\times1} & 0_{4\times1} \\ 0_{1\times4} & 1 & 0 & 0 \\ 0_{1\times4} & -1 & \frac{1}{r_\mathrm{W}} & \frac{d_\mathrm{W}}{r_\mathrm{W}} \\ 0_{1\times4} & -1 & \frac{1}{r_\mathrm{W}} & -\frac{d_\mathrm{W}}{r_\mathrm{W}} \end{pmatrix}}_{T_{\omega,n}} x_n \tag{3.93}$$

$$x_{\omega,g} = \underbrace{\begin{pmatrix} I_{3\times3} & 0_{3\times1} & 0_{3\times1} \\ 0_{1\times3} & \frac{1}{r_\mathrm{W}} & \frac{d_\mathrm{W}}{r_\mathrm{W}} \\ 0_{1\times3} & \frac{1}{r_\mathrm{W}} & -\frac{d_\mathrm{W}}{r_\mathrm{W}} \end{pmatrix}}_{T_{\omega,g}} x_g \tag{3.94}$$

to replace the forward velocity as well as the heading rate by $\omega_{\delta\mathrm{R}}$ and $\omega_{\delta\mathrm{L}}$. By applying the transformation (3.92) to the state space model (3.60), a new state space model with the state vector

$$x_{\omega,c} := \left((x_\mathrm{B}, y_\mathrm{B}, \theta, \alpha), (v_\alpha, \omega_{\delta\mathrm{R}}, \omega_{\delta\mathrm{L}}, i_\mathrm{R}, i_\mathrm{L})\right)^\mathrm{T} \ , \tag{3.95}$$

is derived which is called the ω-c-model. The n-model as well as the g-model can be transformed analogously and the matrices $T_{\omega,n}$ and $T_{\omega,g}$. Delgado et al. [22] also uses $\omega_{\delta\mathrm{R}}$ and $\omega_{\delta\mathrm{L}}$ during modeling but applies a transformation into forward and heading velocities at the end of his derivation. Furthermore, Ha and Yuta [40] uses the relative angle between the body and the wheels during modeling but then they simplify and linearize their model and dismiss the former structure. During observer synthesis, the presented transformation might ease the process, as the encoders directly measure the differential motion of the wheels in respect to the body and if identical properties of the wheels are assumed, the number of design parameters can be reduced. This property will be used in Section 6.3. In addition, the ω-representations may ease the design of nonlinear control laws. The transformation reveals the symmetries of the right and the left wheel and the system inputs. Moreover, the linearized ω-c-model, has an integration chain $u_{\mathrm{R}} \rightarrow i_{\mathrm{R}} \rightarrow \omega_{\delta\mathrm{R}}$ which is (almost) decoupled from the chain $u_{\mathrm{L}} \rightarrow i_{\mathrm{L}} \rightarrow \omega_{\delta\mathrm{L}}$ and both chains show symmetries. The existing weak coupling between the two integrator chains might be neglected and treated as a disturbance. Based on this, new opportunities may arise in the application of partial feedback linearization or backstepping. The exploration of the revealed structure in the ω-representations and how one could benefit from this representation opens up new opportunities for further research.

Another model can be gained by a simplification of the presented models. The control design is usually challenging, due to the nonholonomic constraints and the nonlinear differential equations

$$\dot{x}_{\mathrm{B}} = v_d \cos(\theta) \tag{3.96}$$

$$\dot{y}_{\mathrm{B}} = v_d \sin(\theta) \tag{3.97}$$

for the position of the robot. By removing these equations in the state space model and introducing a new state $d = \int_0^t v_d \, \mathrm{d}\tau \in \mathbb{R}$ for the distance driven along the ${}_{A}x$ axis, a simplified model is obtained. Based on the c-model the new state vector is defined as

$$x_{d,c} := \left((d, \theta, \alpha), (v_\alpha, v_d, v_\theta, i_{\mathrm{R}}, i_{\mathrm{L}})\right)^{\mathrm{T}} \ . \tag{3.98}$$

This model will be labeled as d-c-model in the following. The n-model as well as the g-model can be modified in the same way and are called d-n model and d-g-model respectively.

Contrary to the ω-models the d-models are gained by a simplification and not by similarity transformation of the state space model. Thus, the d-models have one state less and ease the control task, since the $x - y$ plane position information has been removed. This approach is used by e.g. Kim and Kwon [56] and Kim and Kwon [55]. In the case of the application of linear control, a linearized model of the robot is required. If the model (3.60) or (3.78) is linearized, the first two differential equations are linear dependent from each other and the model loses validity if the heading angle changes. Thus, the first two states x_{B} and y_{B} are replaced after linearization by d commonly. Let's use the d-models in the following for linearization to avoid the manual replacement of the first two states afterwards.

3.1.11 Kinetic Energy and Energy Dissipation

The state space model of the TWIP has been derived using the Lagrange framework. Therefore, kinetic and potential energy terms, which define the energy-storages of the system, have been formulated in Subsection 3.1.1. In addition, external forces for dissipation and system input have been considered in Subsection 3.1.4. Let us utilize these terms to get a physical insight into the interconnection of the different energy-storages and the dissipation terms in the following, by the use of the trajectory '8-Knot and Snail' introduced in Section 4.3 and shown in Figures 4.1, 4.2, 4.4 and 4.5.

Firstly, let us focus on the input power applied through the motor terminals to the TWIP and the power dissipation. The input power is defined by

$$P_{\mathrm{in}} = u_{\mathrm{R}} i_{\mathrm{R}} + u_{\mathrm{L}} i_{\mathrm{L}} \tag{3.99}$$

and the power dissipated by the mechanical friction

$$P_{\text{diss,fric}} = \tau_{FR}(\omega_{\delta\text{R}})\omega_{\delta\text{R}} + \tau_{FL}(\omega_{\delta\text{L}})\omega_{\delta\text{L}} \tag{3.100}$$

based on (3.37) as well as the power dissipated by the motor armature resistance

$$P_{\text{diss,res}} = R_{\text{M}}(i_{\text{R}}^2 + i_{\text{L}}^2) \tag{3.101}$$

according to(3.38).

In addition, the difference between the input power and the dissipated power, which is 'converted' and then stored or emitted in kinetic and/or potential energy, is calculated by

$$P_{\text{conv}} = P_{\text{in}} - P_{\text{diss,fric}} - P_{\text{diss,res}} \,. \tag{3.102}$$

The power values over time is shown in Figure 3.4 for the chosen trajectory. It can be

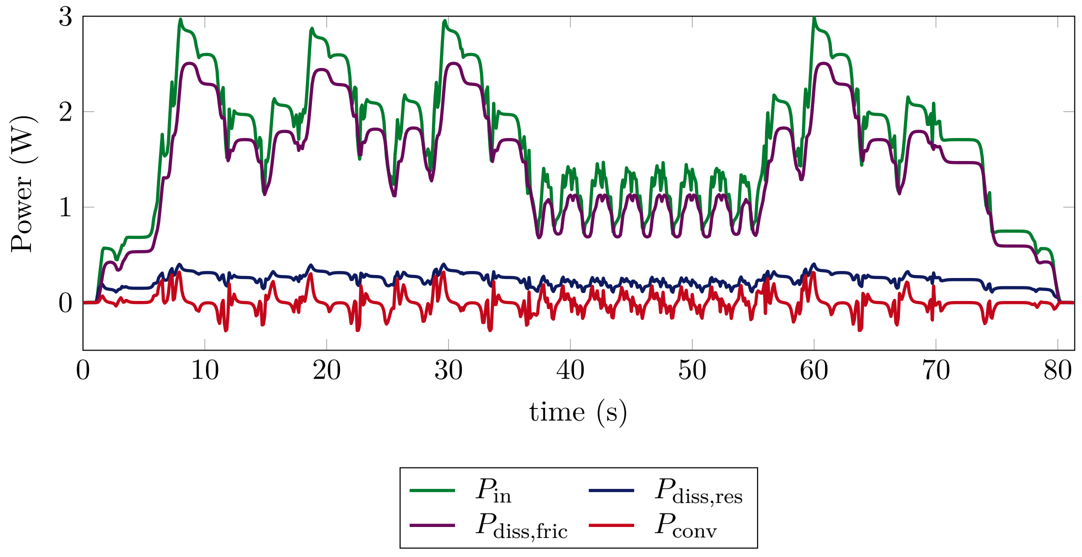


Figure 3.4: Power flows inside the TWIP

observed in the plot, that the energy exchange over time (which is the work or power P_{conv}) between the system input and the kinetic as well as potential energy storages is quite small compared to the energy dissipated by the mechanical friction and the motor armature resistance. In particular, the vast amount of the input power is dissipated by the mechanical friction of the system. Sloppily speaking, our TWIP is nothing else than a large, nonlinear mechanical damper, which should be balanced. More seriously it can be concluded, that increased attention has to be paid during modeling and parameter identification of the mechanical friction since these terms are dominating based on the power point of view. Surprisingly, the majority of publications (e.g. Pathak et al. [84], Kim and Kwon [56]) neglect friction completely in their model or assume simple viscous friction and concentrate on perfectly accurate modeling of the mechanical kinetic system instead. Other authors like Delgado [23] propose a full friction compensation and assume that perfect compensation takes place. As a consequence, all friction terms where removed from the model used for control synthesis. This is a good approach as long as an accurate friction model and/or a controller, robust enough to handle uncertainties, is available. This discussion is based on an exemplary trajectory with the parameters of our

TWIP. In consequence, if the TWIP is just balanced or a TWIP with a total different friction parameter is used, the results might not be directly transferable. To conclude, the author likes to emphasize that the identification and modeling of the mechanical friction is one of the important keys to gain a model with a high degree of congruence with the real system, which is the fundament of a successful controller synthesis.

In addition to Figure 3.4, the time derivatives of the terms of the energy storages (kinetic and potential energy) given in Subsection 3.1.3 are calculated based on the same trajectory used in the paragraph above. Only for the derivative of the electrical potential term a slightly modified power term

$$\dot{V}_{M,el} = k_{\mathrm{E}} n_{\mathrm{WM}} \left(\omega_{\delta \mathrm{R}} \dot{q}_{\mathrm{R}} + \omega_{\delta \mathrm{L}} \dot{q}_{\mathrm{L}} \right) \tag{3.103}$$

is defined and used for better interpretation. As a result $\dot{V}_{M,el}$ is the electrical power converted from the motor into mechanical work. In Table 3.1 the root mean square (RMS)-values and the maximum values of the time derivatives of the different energy terms in the Lagrange equation are sorted by their RMS-values.

Table 3.1: RMS and maximum power values of the time derivates of the different energy terms in the Lagrange equation, evaluated on a trajectory

Term	Equation	Component	Type	RMS in W	max in W
P_{in}	(3.99)	Input	–	1.764	2.991
$\dot{V}_{M,el}$	(3.103)	Motor	–	1.525	2.589
$P_{\mathrm{diss,fric}}$	(3.100)	Friction	–	1.522	2.507
$P_{\mathrm{diss,res}}$	(3.101)	Resistance	–	$2.427 \cdot 10^{-1}$	$4.052 \cdot 10^{-1}$
$\dot{T}_W$	(3.27)	Wheels	rotational	$8.725 \cdot 10^{-2}$	$2.932 \cdot 10^{-1}$
$\dot{T}_G$	(3.31)	Motor	rotational	$6.578 \cdot 10^{-2}$	$2.172 \cdot 10^{-1}$
$\dot{T}_B$	(3.26)	Body	translational	$1.186 \cdot 10^{-2}$	$3.952 \cdot 10^{-2}$
$\dot{T}_G$	(3.31)	Gears	rotational	$8.949 \cdot 10^{-3}$	$2.945 \cdot 10^{-2}$
$\dot{T}_W$	(3.27)	Wheels	translational	$2.679 \cdot 10^{-3}$	$8.996 \cdot 10^{-3}$
$\dot{T}_B$	(3.26)	Body	rotational	$5.699 \cdot 10^{-4}$	$1.985 \cdot 10^{-3}$
$\dot{V}_B$	(3.34)	Body	–	$1.829 \cdot 10^{-4}$	$8.080 \cdot 10^{-4}$
$\dot{T}_M$	(3.35)	Inductance	–	$4.589 \cdot 10^{-5}$	$5.445 \cdot 10^{-4}$

First of all, the RMS of P_{in} as well as the maximum is far below the rating of 6 W of each motor. Thus, based on the power rating of the motors and the trajectory analyzed, there are enough reserves left for the feedback controller. In addition, it can be recognized, that the power RMS of the mechanical dissipation dominates all other terms, besides the total input power.

Secondly, it can be observed that the power RMS value connected to the energy storage of the motor inductance T_M is the smallest. This is a strong indication that the neglection of the current dynamics is a valid approach to reduce the system order and complexity. Let us draw a quick connection to port-Hamiltonian systems and model order reduction. The Lagrangian of a dynamical system can be transformed with the Legendre transformation into a Hamiltonian system and thus can also be brought into port-Hamiltonian representation. Polyuga and van der Schaft [90] presented a model order reduction method for port-Hamiltionian systems, which takes the power flow between different energy storages into account. In particular, Polyuga and van der Schaft [90] describe this procedure as "replacing the interconnections to the energy storage, which carry little power, by zero-power constraints". In the same spirit, the neglection of the current dynamics in Subsection 3.1.7 may be justified due to the small power values linked to the magnetic field of the motor inductance corresponding to Table 3.1.

Contrary to the current dynamics, it is not admissible to simplify the model of the balancing TWIP by removing the rotational components of $\dot{T}_B$ as well as $\dot{V}_B$. Both power components linked to the body motion are also relatively small but the connected energy terms include the instabilities of the TWIP and thus cannot be removed. Nevertheless, if this is done one will end up with the model of the TWIP on ground as introduced in Subsection 3.1.8.

3.2 Onboard Sensors Model

As introduced in Section 2.3, the TWIP has four onboard sensors: the accelerometer, the gyroscope, and two optical encoders. In the following subsection, the modeling of these is presented.

There is always a trade-off between neglecting the effects and dynamics of the sensors completely or the attempt to capture almost all properties and dynamics. A precise model strongly depends on the information given in the sensor's datasheet or requires additional experiments to identify the properties and parameters. Since the accelerometer and gyroscope are manufactured for low-end customer products, unfortunately only limited information is available for modeling. Thus, the focus is laid on the most important inaccuracies and dynamics and uses rough estimates for the uncertainties of the sensor parameters. For the encoder, also a simple model is proposed.

For control design, it has been decided to neglect the sensor model and thus assume a perfect state estimate. Also, the state estimation design is based on the TWIP model without the sensor model but for the tuning procedure of the observer, presented in Chapter 6, the simulation environment for the TWIP which includes the sensor models is used. Furthermore, for all simulation studies presented, the model of the TWIP as well as the sensor models are included.

3.2.1 Accelerometer

The 3-axis accelerometer is soldered on the main PCB as shown in Figure 2.2 and measures the body acceleration as well as gravity. In the following, an acceleration sensor model is derived, as illustrated in Figure 3.5.

As shown in Figure 3.6, the accelerometer coordinate system S is rotated around the ${}_{B}y$ axis by $-90°$. This fixed rotation

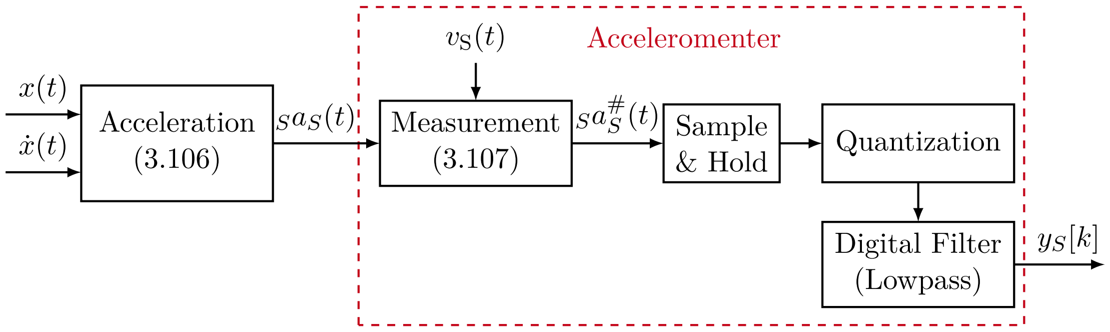


Figure 3.5: Structure of accelerometer model

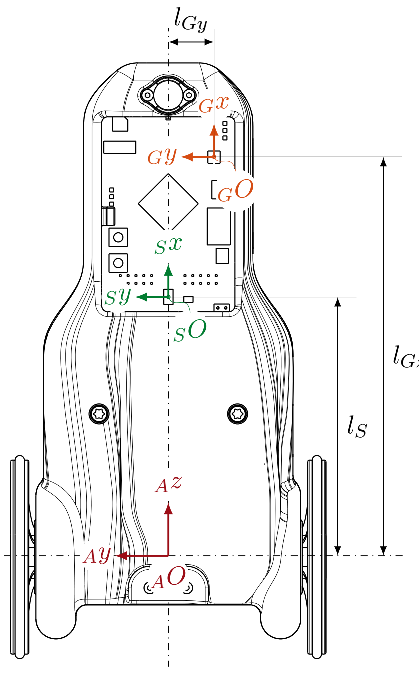


Figure 3.6: Coordinate systems and geometric parameters of the accelerometer and gyroscope

$$
{}^B R_S = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{3.104}
$$

is used, to align the sensor's coordinate system modeled along with the coordinate system given in its datasheet. As a result, measurements read from the sensor's output registers directly match with the axes for S defined in Figure 3.6. The acceleration 'felt' by the sensors, is the sum of the acceleration of the body and through gravity. Based on the measurement principle of the accelerometer, gravity is measured with the opposite sign.

As a result, the acceleration

$$
{}_I a_A = \frac{\mathrm{d}^2}{\mathrm{d}t}\begin{pmatrix} x_\mathrm{B} \\ y_\mathrm{B} \\ r_\mathrm{W} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \mathrm{g} \end{pmatrix} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} v_d\cos(\theta) \\ v_d\sin(\theta) \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \mathrm{g} \end{pmatrix} = \begin{pmatrix} \dot{v}_d\cos(\theta) - v_d\sin(\theta)\dot{\theta} \\ \dot{v}_d\sin(\theta) + v_d\cos(\theta)\dot{\theta} \\ \mathrm{g} \end{pmatrix} \tag{3.105}
$$

would be measured if the sensor is placed in the origin ${}_A O$ of the body and the axis of S is aligned with A. Due to design limitations, the sensor chip is not placed in the origin of A but has a fixed position in B with the distance l_S along the ${}_B z$ axis. Thus a rotation around α to ${}_I a_A$ has to be applied and due to this, additional acceleration terms arise which have to be considered. In the last step, the sum of all accelerations have to be rotated to the sensor's coordinate system and we get

$$
\begin{aligned}
\underbrace{\begin{pmatrix} {}_S a_{Sx} \\ {}_S a_{Sy} \\ {}_S a_{Sz} \end{pmatrix}}_{{}_S a_S} &= {}^B R_S^\mathrm{T}\left({}^I R_{BI}^\mathrm{T} a_A + {}_B^I\dot{\omega}_B \times \begin{pmatrix} 0 \\ 0 \\ l_S \end{pmatrix} + {}_B^I\omega_B \times \left({}_B^I\omega_B \times \begin{pmatrix} 0 \\ 0 \\ l_S \end{pmatrix}\right)\right) \\
&= \begin{pmatrix} -l_S(v_\theta^2\sin(\alpha)^2 + v_\alpha^2) + \dot{v}_d\sin(\alpha) + \mathrm{g}\cos(\alpha) \\ l_S(\dot{v}_\theta\sin(\alpha) + 2v_\alpha v_\theta\cos(\alpha)) + v_\theta v_d \\ l_S(v_\theta^2\sin(\alpha)\cos(\alpha) - \dot{v}_\alpha) - \dot{v}_d\cos(\alpha) + \mathrm{g}\sin(\alpha) \end{pmatrix},
\end{aligned} \tag{3.106}
$$

which is the acceleration acting on the assigned axes of S. Our result (3.106) is hereby a function of the state x and it's derivative $\dot{x}$ and is thus connected to the TWIP model, as shown in Figure 2.9.

In the ideal case, the calculated accelerations are the ones received from the sensor's measurement output. As always with real devices, this is not the case and measurement errors, noise, and additional effects, which degenerate the measurements, have to be considered. Thus, for the measured acceleration the equation

$$
{}_S a_S^\# = c_S\left\{ \underbrace{\begin{pmatrix} 1+\epsilon_x & m_{xy} & m_{xz} \\ m_{yx} & 1+\epsilon_y & m_{yz} \\ m_{zx} & m_{zy} & 1+\epsilon_z \end{pmatrix}}_{M_S} {}_S a_S + \underbrace{\begin{pmatrix} v_{Sx} \\ v_{Sy} \\ v_{Sz} \end{pmatrix}}_{v_S} \right\} \tag{3.107}
$$

is proposed, where c_S is the conversion factor from m/s^2 to LSB. The measurements matrix M_S is used to model the effects of scale error, sensor misalignment, and cross-axis sensitivity and v_S for noise. Inside M_S, the terms ϵ_i take the scale error into account while the m_{ij} values represent the misalignment and cross-axis sensitivity. A possible measurement bias is neglected, as experiments have shown that they are negligible in our case.

Unfortunately, the precise values of M_S are unknown and any characteristics as well as values of v_S either. Moreover, some of these values might even change during operation (e.g. with temperature). Nevertheless, let us try to approximate the effects described as follows:

- A maximum nonlinearity of $\pm 0.5\,\%$ (percentage of full scale) is specified in the product datasheet. Thus, let us choose a random value for each ϵ_i in this bound for every simulation run.
- The cross-axis sensitivity is specified to be less than $\pm 0.5\,\%$ and the inter-axis alignment error less than $\pm 0.1°$. But since the chip is soldered on the main PCB and the PCB itself cannot be precisely mounted in the body, this misalignment is supposed to overlay all the chip imprecisions. Thus let us assume a small rotation of the matrix $\mathrm{diag}(1+\epsilon_x, 1+\epsilon_y, 1+\epsilon_z)$ along all three axes of S. The value of each of the three rotation angles is chosen as a random value in the range of $\pm 1°$ in each simulation. As a result, we get the off-diagonal values m_{ij} in M_S.
- For the term v_S let us consider an additive zero-mean white Gaussian noise, where the power spectrum is adjusted manually such that the measurement output $y_S[k]$ of the accelerometer model shows qualitatively the same noise amplitude.

In the last modeling step, let us focus on the data processing inside the sensor. Firstly, the analog measurement data is sampled and converted into digital values. Unfortunately, the native sampling frequency of the implemented filter is not given in the datasheet. To gain a high resolution, the accelerometer is configured to use the full resolution of 3.9 mg/LSB. Thus, the conversion factor is defined by $c_S = \frac{1}{(3.9 \cdot 10^{-3} \cdot 9.81)}$. Finally, the measurement data is processed by a digital lowpass filter inside the accelerometer as shown in Figure 3.5 with a configured bandwidth (-3 dB) of 400 Hz. To account for this, a first-order FIR filter with the same bandwidth in the 'Direct form II Transposed' (see Smith III [100]) is implemented in the sensor model and a sampling frequency of 2 kHz is assumed.

As a result, a sampled measurement output of the accelerometer $y_S[k]$ is received, which is somewhat closer to the real system than just ignoring the sensor effects and dynamics.

3.2.2 Gyroscope

Beside the accelerometer, a 3-axis gyroscope is soldered on the main PCB as shown in Figure 2.2 and Figure 3.6. Analogously to the accelerometer, a model for the gyroscope is presented in the following with the structure shown in Figure 3.7.

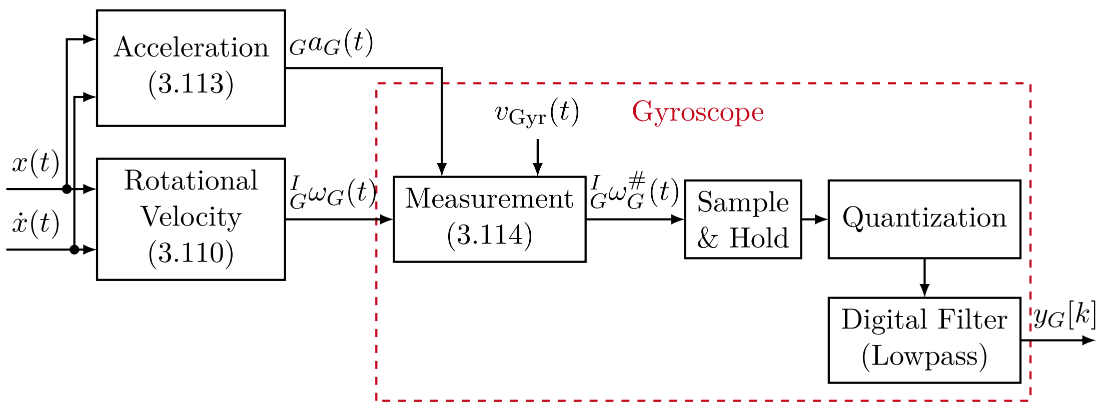


Figure 3.7: Structure of gyroscope model

The coordinate system G for the gyroscope is rotated around the ${}_By$ axis by $-90°$, as shown in Figure 3.6. By introducing the rotation matrix

$$
{}^BR_G = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{3.108}
$$

from the coordinate system G to B, the total rotation from G to I

$$
{}^IR_G = {}^IR_B{}^BR_G = \begin{pmatrix} \sin(\alpha)\cos(\theta) & -\sin(\theta) & -\cos(\alpha)\cos(\theta) \\ \sin(\alpha)\sin(\theta) & \cos(\theta) & -\cos(\alpha)\sin(\theta) \\ \cos(\alpha) & 0 & \sin(\alpha) \end{pmatrix} \tag{3.109}
$$

can be derived. The gyroscope thus undergoes the relative rotational motion

$$
{}^I_G\tilde{\omega}_G = {}^IR_G^{\mathrm{T}}\frac{\mathrm{d}}{\mathrm{d}t}\left({}^IR_G\right) \tag{3.110}
$$

$$
{}^I_G\omega_G = \begin{pmatrix} \dot{\theta}\cos(\alpha) \\ \dot{\alpha} \\ \dot{\theta}\sin(\alpha) \end{pmatrix} \tag{3.111}
$$

in respect to the inertial system, which is measured solely in the ideal case.

Unfortunately, based on the gyroscope measurement principle, the sensor is sensitive to linear acceleration. Thus, let us derive the acceleration of the gyroscope analogously to the acceleration equations of the accelerometer in the next steps. The acceleration the gyroscope would be exposed to, if it is located at ${}_AO$, is given with

$$
{}_Ia_A = \frac{\mathrm{d}^2}{\mathrm{d}t}\begin{pmatrix} x_{\mathrm{B}} \\ y_{\mathrm{B}} \\ r_{\mathrm{W}} \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ \mathrm{g} \end{pmatrix} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} v_d\cos(\theta) \\ v_d\sin(\theta) \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ \mathrm{g} \end{pmatrix} = \begin{pmatrix} \dot{v}_d\cos(\theta) - v_d\sin(\theta)\dot{\theta} \\ \dot{v}_d\sin(\theta) + v_d\cos(\theta)\dot{\theta} \\ -\mathrm{g} \end{pmatrix}, \tag{3.112}
$$

where the gravity has to be treated with the negative sign, contrary to the accelerometer. The origin G of the gyroscope coordinate system is located with a distance of l_{Gz} along the ${}_Bz$ axis and l_{Gy} along the ${}_By$ from the origin of B. Including the additional acceleration terms, arising from the rotation and translation of G in respect to B the acceleration acting on the gyroscope axis is given by

$$
\begin{aligned}
\underbrace{\begin{pmatrix} {}_Ga_{Gx} \\ {}_Ga_{Gy} \\ {}_Ga_{Gz} \end{pmatrix}}_{{}_Ga_G} &= {}^BR_G^{\mathrm{T}}\left({}^IR_B^{\mathrm{T}}{}_Ia_A + {}^I_B\dot{\omega}_B \times \begin{pmatrix} 0 \\ l_{Gy} \\ l_{Gz} \end{pmatrix} + {}^I_B\omega_B \times \left({}^I_B\omega_B \times \begin{pmatrix} 0 \\ l_{Gy} \\ l_{Gz} \end{pmatrix}\right)\right) \\
&= \begin{pmatrix} -l_{Gy}\dot{v}_\theta\sin(\alpha) - l_{Gz}(v_\theta^2\sin(\alpha)^2 + v_\alpha^2) + \dot{v}_d\sin(\alpha) - \mathrm{g}\cos(\alpha) \\ -l_{Gy}v_\theta^2 + l_{Gz}(\dot{v}_\theta\sin(\alpha) + 2v_\alpha v_\theta\cos(\alpha)) + v_\theta v_d \\ l_{Gy}\dot{v}_\theta\cos(\alpha) + l_{Gz}(v_\theta^2\sin(\alpha)\cos(\alpha) - \dot{v}_\alpha) - \dot{v}_d\cos(\alpha) - \mathrm{g}\sin(\alpha) \end{pmatrix}.
\end{aligned} \tag{3.113}
$$

Based on (3.110) and (3.113) the measurement equation

$$
{}^{I}_{G}\omega^{\#}_{G} = c_G \left\{ \underbrace{\begin{pmatrix} 1+\epsilon & m_{xy} & m_{xz} \\ m_{yx} & 1+\epsilon & m_{yz} \\ m_{zx} & m_{zy} & 1+\epsilon \end{pmatrix}}_{M_G} {}^{I}_{G}\omega_G + \underbrace{\begin{pmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{pmatrix}}_{K_G} {}_{G}a_G + \underbrace{\begin{pmatrix} v_{Gx} \\ v_{Gy} \\ v_{Gz} \end{pmatrix}}_{v_G} \right\} \tag{3.114}
$$

is proposed for the gyroscope, where c_G is the conversion factor to from rad/s to LSB. Similar to the accelerometer measurement equation, the scale error and axis misalignment are accumulated in the measurement matrix M_G. Rate measurements due to linear acceleration are incorporated through K_G and noise is included by v_G. Let us choose the parameters as follows:

- The maximum nonlinearity is specified with $\pm 0.2\,\%$ (percentage of full scale) in the gyroscope data sheet and thus we calculate a random value for each ϵ_i in this bound for each simulation run.

- The cross-axis sensitivity is specified to be less than $\pm 2\,\%$, the inter-axis alignment error is not specified. Since the chip is soldered on the main PCB and the PCB itself cannot be precisely mounted in the body, this misalignment is supposed to overlay all the chip imprecisions, similar to the accelerometer. Thus, M_S is calculated analogously to the accelerometer with the same range of $\pm 1°$ for the random rotations.

- The entries of K_G are unknown and thus random values for each k_{ij} are generated for each simulation run. Based on the product specification, the values are chosen in a range of $\pm 0.1\,(°/\mathrm{s})/\mathrm{g}$.

- For the term v_G we considered additive zero-mean white Gaussian noise, where the power spectrum is adjusted manually such that the measurement output $y_G[k]$ show qualitative the same noise amplitude.

The measurement data processing in the gyroscope is similar to the accelerometer. In the firmware of the MCU a resolution of $16.4\,\mathrm{LSB}/(°/\mathrm{s})$ is configured for quantization and thus $c_G = 16.4\frac{180}{\pi}$. Furthermore, the sampling and FIR filter frequency is configured to 2 kHz. According to the selected bandwidth in the firmware, the model filter has been designed with a bandwidth of 256 Hz.

Finally, a sampled measurement output of the gyroscope $y_G[k]$ is derived, which includes the most important effects and dynamics.

3.2.3 Encoder

The TWIP contains two encoders units. Each unit consists of a reflective codewheel glued on each wheel with 900 CPR and an optical encoder mounted on the opposite side on the body, as shown in Figure 2.3 for one side. Thus, the encoders measure the angles between the body and the wheels

$$\phi_\delta = \begin{pmatrix} \phi_\mathrm{R} - \alpha \\ \phi_\mathrm{L} - \alpha \end{pmatrix} = \int_0^t \omega_\delta \mathrm{d}\tau \ , \tag{3.115}$$

which is the integrated relative motion ω_δ. Based on its definition in (3.29),(3.30) the term can be calculated by

$$\omega_\delta = \begin{pmatrix} \omega_{\delta\mathrm{R}} \\ \omega_{\delta\mathrm{L}} \end{pmatrix} = \begin{pmatrix} \dot{\phi}_\mathrm{R} - v_\alpha \\ \dot{\phi}_\mathrm{L} - v_\alpha \end{pmatrix} = \begin{pmatrix} \frac{1}{r_\mathrm{W}} v_d + \frac{d_\mathrm{W}}{r_\mathrm{W}} v_\theta - v_\alpha \\ \frac{1}{r_\mathrm{W}} v_d - \frac{d_\mathrm{W}}{r_\mathrm{W}} v_\theta - v_\alpha \end{pmatrix} \ , \tag{3.116}$$

which is a function of the robot's state x. The model structure of the encoders is shown in Figure 3.8. Based on the measurement principle of counting reflective lines

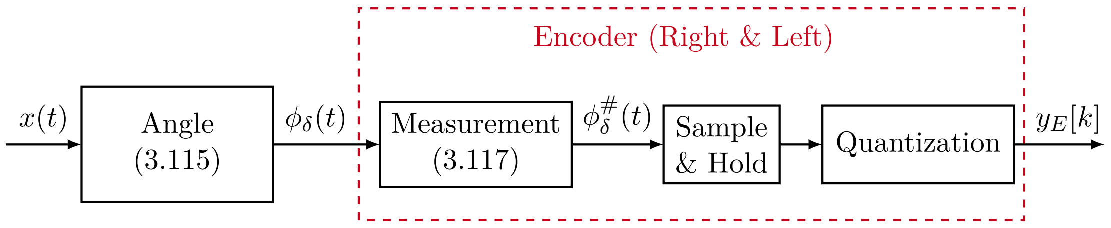


Figure 3.8: Structure of encoder model

passing by the optical encoder chip, the measured values are already discrete. This is emphasized by the 'Quantization' block in the structure. The 'Sample & Hold' stands for the cyclic reading of the MCU register, with the current counter value. However, the implementation in the simulation model is similar to the gyroscope and accelerometer model.

The measurements based on the encoder's counts degenerated by errors from several sources, e.g. from a slight variation of the thickness in the reflective lines of the codewheels or a misalignment of the center of the codewheel to the rotation axis. Let us assume, that these errors are negligible in our application but account in the measurement equation for another error, which indirectly affects the measurements, the deviation of the wheel radius from its nominal value r_W. Therefore, the encoder measurement equation

$$\phi_\delta^\# = c_E \underbrace{\begin{pmatrix} 1+\epsilon_r & 0 \\ 0 & 1+\epsilon_l \end{pmatrix}}_{M_E} \phi_\delta \ , \tag{3.117}$$

is introduced where c_E is the conversion factor to from rad to LSB and M_E accounts for the measurement errors. Hereby, ϵ_r and ϵ_l are randomly chosen in a range of $\pm 1\,\%$ for every simulation run. As the codewheel has 900 CPR and is evaluated by a $4\times$ quadrature decoding module of the MCU, the conversion factor is defined by $c_E = \frac{3600}{2\pi}$.

3.3 Model Linearization and Analysis

So far, the nonlinear models describing the dynamics of the TWIP have been derived. While the nonlinear state space models offer a high accuracy through the whole state space, system analysis, as well as control synthesis, are commonly quite challenging. On the other side, there are strong tools to analyze LTI models as well as for control synthesis. 'There is no free lunch' and thus all results gained with the LTI models may rapidly lose validity if the region is left around the model has been linearized. Having this in mind, LTI models are derived by linearizing the nonlinear models from Section 3.1, analyzed, and compared to get a close insight into the properties of the linearized dynamics of the TWIP around the equilibrium. This is of particular interest, as the linear models are used for control and observer synthesis in the next chapters.

3.3.1 Linearization

All TWIP models introduced have a nonlinear friction torque curve with a high $d_D = \frac{\mathrm{d}\tau_D(\omega_\delta)}{\mathrm{d}\omega_\delta}\big|_{\omega_\delta^*}$ damping value around $\omega_\delta^* = 0$ as can be seen in Figure 3.9. The torque arises

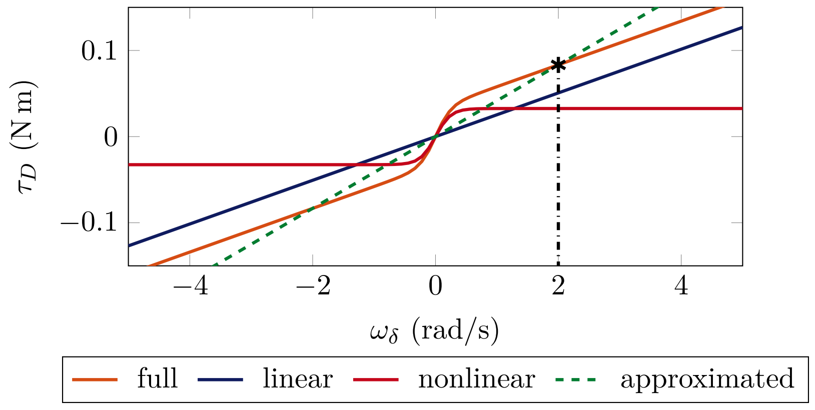


Figure 3.9: Dissipative torques forcing the n-model

from the mechanical friction as well as the motor back-EMF and resistance but the nonlinearity is due to the Coulomb-friction terms. Coulomb-friction or stick-friction is modeled with an sgn-function. This results in a torque curve that is not smooth around zero and causes problems, e.g. during numerical integration. Therefore, this term is commonly replaced by a steep tanh-function. As a result, if the models are linearized at the origin, the friction will directly be related to the product $d_\mathrm{C} d_0$ as a result of the tanh-function in the mechanical friction equation in (3.37). The choice of d_0 has no physical connection to the real system and is a trade-off: On the one hand, it should be as high as possible to closely approximate the sgn-function and thus the stick-friction. On the other hand, too high values of d_0 will lead to tiny time steps during numerical integration. Thus d_0 should not be too high, ensuring that the differential equations are not getting too stiff. Concluding, a linearization without modifying the friction torque term, will result in linear models which include too high damping terms. As a result, these models will lose rapidly their validity as ω_δ leaves the origin.

This problem has been solved in Anhalt [5] and Albert et al. [2] by evaluating the torque τ_D at a nominal operating speed ω_δ^* and calculating a viscous friction coefficient d_V which leads to the same torque value. This is illustrated in Figure 3.9 for a value of $\omega_\delta^* = 2\,\mathrm{rad/s}$. Afterwards the parameters d_C in (3.37) is set to zero. The drawback of this method is, that around the linearization point the gained damping always differs from the real damping $d_D = \frac{\mathrm{d}\tau_D(\omega_\delta)}{\mathrm{d}\omega_\delta}\big|_{\omega_\delta^*}$.

In this thesis, a different approach for linearization is pursued: It is assumed that the nonlinear components of the friction torque can be compensated through the plant input. At least for the models without current dynamics, as could be seen in (3.71) and (3.70) as well as (3.84) and (3.83) this is possible. Contrary, for the model with current dynamics it has to be assumed that the current dynamics play a negligible role to apply a friction compensation. Therefore, a friction compensation is introduced in Section 5.2 as presented by Delgado [23]. However, herein only the nonlinear components are compensated since this will ease the allocation of the limited control input between the controller and the friction compensation. Based on the assumption, that the nonlinear components can be compensated, the linear friction torque curve shown in Figure 3.9 is derived. This leads to a constant friction viscous coefficient d_V for all ω_δ and in consequence, the Coulomb-friction coefficient can be set to $d_\mathrm{C} = 0$ in (3.37) for linearization.

To avoid two linearly dependent differential equations for the positions x_B and y_B, we will use the *d*-models with the forward-driven distance for linearization. Based on the models with only linear friction curves, we are able to linearize the systems. The resulting A, B matrices for the *c*-*d*-model, *n*-*d*-model and *g*-*d*-model with the parameters listed in Table A.1 is given in Appendices A.1 to A.3.

The eigenvalues of the linear systems, different choices of the system output, state transformations and a discussion is provided in the next subsections.

3.3.2 Linear Decoupling

The linear models of the TWIP can be decoupled into two separate dynamic systems: one system describing the forward motion along A_x-axis and a second system describing the heading motion around the A_z axis. Thus the goal is, to find a state transformation T_x and input transformation T_u, to get the decoupled structure

$$\begin{pmatrix} \dot{x}_{Fw} \\ \dot{x}_{He} \end{pmatrix} = \underbrace{\begin{pmatrix} A_{Fw} & 0 \\ 0 & A_{He} \end{pmatrix}}_{T_x A T_x^{-1}} \begin{pmatrix} x_{Fw} \\ x_{He} \end{pmatrix} + \underbrace{\begin{pmatrix} b_{Fw} & 0 \\ 0 & b_{He} \end{pmatrix}}_{+T_x B T_u^{-1}} \begin{pmatrix} u_{Fw} \\ u_{He} \end{pmatrix} \quad \begin{matrix} \textit{forward dynamics} \\ \textit{heading dynamics} \end{matrix} \tag{3.118}$$

of the system, with

$$\begin{pmatrix} x_{Fw} \\ x_{He} \end{pmatrix} = T_x x \tag{3.119}$$

where x_{Fw} is the state vector of the forward dynamics subsystem and x_{He} is the state vector of the heading dynamics subsystem. With T_u, given in (A.7), the input can be transformed to

$$u_{Fw} = u_\mathrm{R} + u_\mathrm{L} \tag{3.120}$$

for the forward dynamics system and

$$u_{He} = u_\mathrm{R} - u_\mathrm{L} \tag{3.121}$$

for the heading dynamics. Since the *c-d-*, *n-d-* and *g-d*-models differ in the number of states, different matrices T_x are needed, and in consequence, different transformed state vectors x_{Fw} are received. The T_x matrices are given in (A.8), (A.9) and (A.10) for the *c-d-*, *n-d-* and *g-d*-models respectively. Finally, based on $T_{c\text{-}d}$ given in (A.8), the state vectors

$$x_{c\text{-}Fw} = \begin{pmatrix} d, & \alpha, & v_d, & v_\alpha, & \underbrace{i_\mathrm{R} + i_\mathrm{L}}_{i_{Fw}} \end{pmatrix}^\mathrm{T} \quad x_{c\text{-}He} = \begin{pmatrix} \theta, & v_\theta, & \underbrace{i_\mathrm{R} - i_\mathrm{L}}_{i_{He}} \end{pmatrix}^\mathrm{T} \tag{3.122}$$

are derived for the feedforward and head dynamics respectively. In the same fashion with $T_{n\text{-}d}$ given in (A.9) the state vectors

$$x_{n\text{-}Fw} = \begin{pmatrix} d, & \alpha, & v_d, & v_\alpha \end{pmatrix}^\mathrm{T}, \quad x_{n\text{-}He} = \begin{pmatrix} \theta, & v_\theta \end{pmatrix}^\mathrm{T} \tag{3.123}$$

are defined for the decoupled *n-d*-model as well as with $T_{g\text{-}d}$ given in (A.10) the state vectors

$$x_{g\text{-}Fw} = \begin{pmatrix} d, & v_d \end{pmatrix}^\mathrm{T}, \quad x_{g\text{-}He} = \begin{pmatrix} \theta, & v_\theta \end{pmatrix}^\mathrm{T} \tag{3.124}$$

for the decoupled *g-d*-model. The gained decoupled representations offer several advantages: First of all, we can use each subsystem to design and analyze a separate controller for it, e.g. designing a standalone LQR for the feedforward as well for the heading system. Additionally, the dynamic properties like the eigenvalues are directly allocated to the corresponding subsystem and the motion it represents. Even though Delgado [23] solely used the nonlinear (non-decoupled) system he used the presented input transformation to get inputs in “more natural quantities”. Finally in the case of trajectory generation, the decoupled system representation can be beneficial, too. Diepold et al. [30] used a quite similar procedure to decouple the linear state space model of a robot balancing on a ball into three separate systems for trajectory generation. Therefore, the decoupled models are used for several purposes in the following sections and chapters: First, to reveal the connection and interpretation of the different eigenvalues of the models to the different system motions. Secondly, to choose a convenient system output, and finally to ease the controller and observer synthesis.

3.3.3 System Output

In the previous subsection, the system has been split into two subsystems with a single input. Now we are facing the question: What are *good* choices for the (single) output of each system for stabilization, setpoint and trajectory tracking?

$y_{Fw} = d$ for the forward dynamics and $y_{He} = \theta$ for the heading dynamics are probably the most natural choices. Nevertheless, for models including the tilt angle in the forward dynamics, there exists also another quite interesting choice

$$y_{Fw} = d + l_O\alpha \tag{3.125}$$

for the output with $l_O \in \mathbb{R}$. This output could be interpreted as a point aligned with the robot's body axis ${}_Bz$ at a distance l_O from the origin ${}_BO$. For example, by an appropriate choice of l_O the output can be shifted to the robot's top.

For trajectory tracking and feedforward control, a flat output (see Lèvine [70], Fliess et al. [34]) is advantageous. If an appropriate reference trajectory $y_{ref}(t)$ is given for a flat output, the feedforward control inputs can be computed easily. Finding a flat output for a controllable LTI-system (see Zak [119], Dorf and Bishop [33]) is a trivial task. All forward dynamics systems as well as their companion heading dynamics systems are controllable. This can be shown through the evaluation of the rank condition

Theorem 3.1. *A LTI system is controllable if the controllability matrix $Q_{\mathcal{C}}$ has full row rank* $\operatorname{rank}(Q_{\mathcal{C}}) = n$ *with n as the number of states.*

with the controllability matrix

$$Q_{\mathcal{C}} = [B,\, AB,\, A^2B,\, \cdots,\, A^{n-1}B] \tag{3.126}$$

for each model. In consequence, the models can be transformed into the controllable canonical form with

$$z = T_{\mathcal{C}}x \tag{3.127}$$

and

$$T_{\mathcal{C}} = \begin{pmatrix} t^{\mathrm{T}} \\ t^{\mathrm{T}}A \\ \vdots \\ t^{\mathrm{T}}A^{n-1} \end{pmatrix} \quad \text{with} \quad t^{\mathrm{T}} = \Big(0,\, \cdots,\, 0,\, 1\Big)\, Q_{\mathcal{C}}^{-1}\,. \tag{3.128}$$

For forward dynamics models of the balancing robot, the flat output is derived by

$$\begin{aligned} y_{\mathcal{F}} &= t^{\mathrm{T}}x_{Fw} \\ &= \Big(t(1),\, t(2),\, 0,\, \cdots,\, 0\Big)\, x_{Fw}\,, \end{aligned} \tag{3.129}$$

where only the elements for d and α are non-zero. Let us scale the flat output by $\frac{1}{t(1)}$ with

$$l_O = \frac{t(2)}{t(1)} \approx 0.0644 \tag{3.130}$$

to gain the form of (3.125). As a result, the calculated flat output could be interpreted as the distance traveled of a point, placed 64.4 mm over the wheel axis on the ${}_Bz$ axis.

3.3.4 Eigenvalues, Poles, Zeros and Performance Limitation

Let us take a closer look at the properties of our linear models. In Table 3.2 the eigenvalues of all three proposed model types are listed. Due to the decoupling of the models, we are able to allocate different eigenvalues to the forward and heading motions. In general, an assignment of eigenvalues to certain states is not possible, besides the ones where the corresponding eigenvectors have only one non-zero entry. This is the case for the eigenvalues 1 and 6, which are the integration of v_d to d and v_θ to θ respectively. Nevertheless, since we removed the current dynamics to gain the n-model as well as the tilt motion in the g-model, we can at least relate the eigenvalues to the different motions of the TWIP. The eigenvalues 2 and 7 are most likely assigned with v_d and v_θ. The duo

Table 3.2: Eigenvalues of the linearized models

Index	c-model	$\leftarrow$ Deviation $\rightarrow$	n-model	$\leftarrow$ Deviation $\rightarrow$	g-model
		Forward dynamics			
1	0.00	-	0.00	0.00	0.00
2	-31.57	0.77 %	-31.33	15.68 %	-26.41
3	-7.21	0.01 %	-7.21		
4	7.53	0.00 %	7.53		
5	-3720.62				
		Heading dynamics			
6	0.00	-	0.00	0.00	0.00
7	-30.70	0.77 %	-30.48	23.02 %	-23.45
8	-3721.14				

of the eigenvalue 3 and 4 are connected α and v_α and describe the balancing motions. Finally, we see the fast eigenvalues 5 and 8, which are only present in the c-model. Thus, we relate them to the motor currents of the TWIP. We recognize, that there is a factor of ≈ 118 between the eigenvalues related to the currents and the other eigenvalues. This justifies the neglection of the current dynamics for control and observer design as done in e.g. Delgado [23]. But if this is done, one has to be aware of the neglected dynamics and ensure, that placed eigenvalues in the closed-loop do not interfere with them.

Based on the choice of the output with (3.125) for the forward dynamics and $y_{He} = \theta$ for the heading dynamics as well as the decoupled linear models in (3.118), we are able to derive a SISO transfer function for each subsystem. As long as we do not calculate the minimal realization where pole-zero cancellation might happen, all eigenvalues listed in Table 3.2 are poles of these transfer functions. It can be immediately recognized, that the transfer functions of the forward dynamics (c- and n-models) have one pole in the right half-plane (RHP) with $p_4 = 7.53$. Let us assume we use output feedback control to stabilize the robot and obtain the open-loop transfer function $L(s) = G(s)K(s)$, with $G(s)$ for the TWIP forward dynamics transfer function and $K(s)$ for an unknown compensator. Recalling the Bode-Integral

$$\int_0^\infty \ln|S(\mathrm{j}\omega)|\,\mathrm{d}\omega = \pi \sum_{p\in\mathcal{P}} \mathrm{Re}(p) \quad \text{with} \quad \mathcal{P} = \{p\,|\,\mathrm{Re}(p) > 0\} \tag{3.131}$$

where

$$S(\mathrm{j}\omega) = \frac{1}{1 + L(s)} \tag{3.132}$$

is the sensitivity transfer function. We see that the area of the log integral where disturbances are amplified is $\pi \cdot p_4$ larger than the area where disturbances are rejected. This will get even worse if the compensator will add additional RHP-poles to $L(s)$. Furthermore, let us follow the argumentation done by Stein [101] for ‘The X-29 Airplane

Story' and take into account, that the shape of the sensitivity function can only be influenced up to certain frequencies. Stein [101] called this bandwidth up to the limiting frequency 'available bandwidth' Ω_a, which should not be mixed up with the cross-over frequency of L or the (closed-loop) bandwidths for $S(\mathrm{j}\omega)$ or $T(\mathrm{j}\omega) = \frac{L(s)}{1+L(s)}$ as defined in e.g. Skogestad [98]. In our case, the 5 ms sampling-rate of the controller as well as the internal filter bandwidth with 256 Hz of the gyroscope will limit Ω_a to $\approx 66 - 100\,\mathrm{rad/s}$[3]. As a result, the upper limit of the Bode-Integral changes to

$$\int_0^{\Omega_a} \ln |S(\mathrm{j}\omega)|\, \mathrm{d}\omega = \pi \sum_{p\in\mathcal{P}} \mathrm{Re}(p) \quad \text{with} \quad \mathcal{P} = \{p\,|\,\mathrm{Re}(p) > 0\} \tag{3.133}$$

and imposes the physical performance limitation for the closed-loop control, at least if output feedback is used. Unfortunately, for multiple-input-multiple-output (MIMO)-systems and state feedback there is not such a clear barrier for the performance limits (see Skogestad [98] and Morari [76]). Nevertheless, it is probably a piece of good advice to keep the Ω_a limit in mind during the design of a state feedback controller. In this spirit, the author recommends not pushing eigenvalues more to the left than Ω_a or moving eigenvalues of the systems which are left from this value, especially the eigenvalues related to the current dynamics.

Moreover, Stein [101] gives a 'rule of thumb' for the bandwidth required, based on the unstable pole: "The available bandwidth should exceed the airplane's unstable pole by at least a factor of ten". Applying this rule of thumb to the TWIP, the unstable pole $p_4 = 7.53$ demands an available bandwidth of at least 75.3 rad/s. This shows, that the chosen fast sample rate of the controller as well as the fastest available filter configuration for the gyroscope is *necessary* to achieve a good control performance.

While it is widely known and accepted, that eigenvalues on the RHP significantly complicate the controller design, the influence of the zeros sometimes is left unattended. Nevertheless, this topic has been in focus in several publications and books. Freudenberg [37] gives a great overview over frequency domain properties of SISO and MIMO feedback systems and Seron [96] gives an introduction to the Bode-Integral formulae, the water-bed effect and reveals the effects of RHP-zeros and poles on the closed-loop performance. In Freudenberg and Looze [35, 36], Looze and Freudenberg [66] the limitations of feedback properties imposed by RHP -poles and RHP-zeros have been discussed as well as the related design trade-offs in feedback systems. Chen et al. [19, 18], Chen [16] focused on limitations on maximal tracking accuracy and Chen [16, 17] discusses sensitivity integral relations and design trade-offs caused by RHP-poles and RHP-zeros. Thus, let us take a closer look at the zeros in $G(s)$ of our system, to draw conclusions to their influence on the achievable closed-loop performance in the case of the TWIP.

With the choice of $y_{He} = \theta$ as output, the heading dynamics do not have any zeros. Thus, the subsystem for the heading dynamics is quite well suited since it has no RHP-zeros nor RHP-poles. Contrary to the heading dynamics, the forward dynamics could have zeros depending on the choice of parameter l_O in the output (3.125). In Table 3.3 the zeros of the transfer function are given with different values for l_O. Unfortunately, there are output configurations with one zero in the RHP, which leads to a non-minimum-phase system behavior. Together with the RHP-pole, this makes the control task extremely challenging: The forward dynamics cannot be stabilized by a stable output feedback

[3]Based on the assumption, analogously to Stein [101], that a minimum of $2 - 3\,\mathrm{sample/rad}$ for a proper control is necessary.

Table 3.3: Zeros of the forward dynamics model with current

l_O	z_1	z_2	min. ISE
0.0000	9.1091	−9.1091	0.2196
0.0643	288.05	−288.05	0.0069
0.0644	–	–	0
0.0645	0.0+j288.05	0.0−j288.05	0
0.1288	0.0+j9.1091	0.0−j9.1091	0

controller[4]. Similar to the example of the inverted pendulum, discussed in Hoagg and Bernstein [44], the controller will need an odd number of RHP-poles to the right of the RHP-zero. These controller poles in the RHP are required to prevent the unstable open-loop pole from being attracted by the RHP-zero if we close the loop. As stated in Hoagg and Bernstein [44] and the references provided herein, "RHP-zeros limit the achievable performance of fixed-gain controllers [. . .] as well as adaptive controllers[. . .]". Moreover, Skogestad et al. [99] conclude, that if an RHP-pole and RHP-zero are close to each other in the complex plane (the distance between the zero and the pole is small), then the peaks in the transfer functions $|S(\mathrm{j}\omega)|$ and $|T(\mathrm{j}\omega)|$ will be large, and stabilization is in practice impossible. As we can see, if we chose $l_O = 0$, then the distance between the RHP-zero and the RHP-pole of the forward dynamics transfer functions is quite small, where the distance increase with larger values of l_O. Finally, Skogestad [98] relates restrictions on the 'gain crossover frequency' ω_c, which is defined as the frequency where $|L(\mathrm{j}\omega)|$ first crosses the 0 dB from above. According to Skogestad [98], only up to ω_c a tight control of frequencies is possible but in practice, ω_c should be chosen to hold approximately $\omega_c < z/2$ for the smallest RHP-zero. Thus, an RHP-zero z close to the origin degenerates the control performance. This encourages even more that one should choose $l_O \gg 0$ for output feedback control[5].

Finally, Qiu and Davison [93] presented "a quantitative measure of the degree of difficulty in solving the servomechanism problem for a non-minimum phase systems with constant disturbances". Let us employ the same performance limit for the TWIP, to get a quantitative measure to judge about different choices for our output parameter l_O. Qiu and Davison [93] introduced the 'cheap LQR' with the cost function

$$J_\epsilon = \min_u \int_0^\infty \left(y'y + \epsilon^2 u'u \right) \mathrm{d}t \tag{3.134}$$

for the system output y and control input u. The minimum value of the cost function is found by reducing the weight of the control input to zero: $J_0 = \lim_{\epsilon \to 0} J_\epsilon$. Qiu and Davison [93] proved, that J_0 is bounded from below by $2 \sum_{i=1}^{N_z} \frac{1}{z_i}$ with N_z for the number of real RHP- zeros.

[4]A 'stable output feedback controller' is a controller, whose transfer function has only poles in the left half-plane.

[5]In the lab course 'Controller Implementation on Microcontrollers' a pseudo-cascade control architecture with output feedback is used. It turned out in experiments, that stabilizing controllers can be easily found and tuned manually with $l_O \gg 0$, where as the same task is almost impossible for $l_O = 0$. The given discussion reveals the reason.

In the next step, let us use the Matlab command *lqry* with $Q_y = 1$ and $R_u = \epsilon^2$ with $\epsilon = 1 \cdot 10^{-5}$ to design a linear-quadratic state-feedback regulator with output weighting. The system matrices used are given in (A.11) and (A.12) for feedforward dynamics with current. This is done for all parameters l_O listed in Table 3.3 for the output equation (3.125) together with the calculated lower bound for the integral square error (ISE) value. The bound from below J_0 is larger than zero for $l_O < 0.0644$, which means, that any response to a unit step or disturbance will cause a higher ISE value than the bound. Contrary, the lower bound for $l_O \geq 0.0644$ is zero since the transfer functions of these systems have no RHP-zeros.

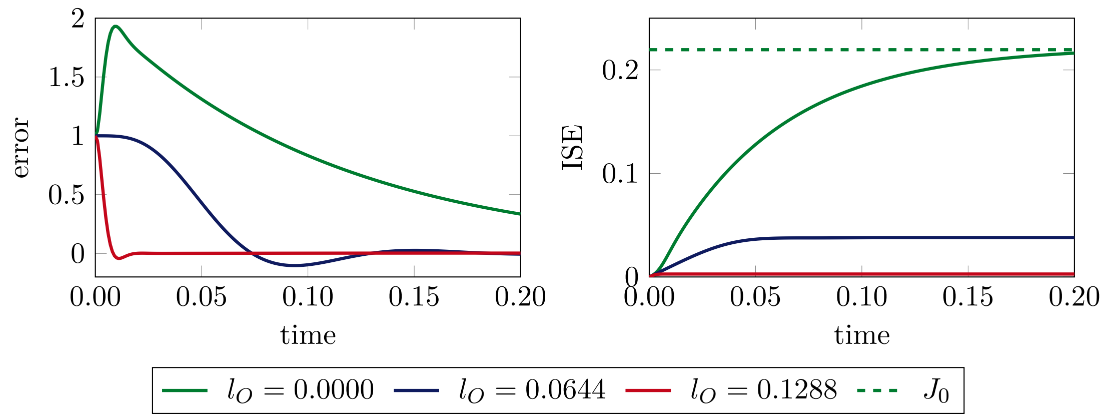


Figure 3.10: Error and ISE over time for different output

In Figure 3.10 the error response to a unit step as well as the ISE is shown. For sure, these results only hold if the control input is unbounded, which is never the case in physical systems. The fastest response shows the system with the output parameter $l_O = 0.0128$ and a tiny ISE of 0.0028. Furthermore, the system with the output parameter $l_O = 0.0644$ has no zeros at all, since it is a flat output. Its error response does not show any undershoot and smoothly decreases to zero with a strongly damped oscillation and a final ISE value of 0.0380. Finally, the response of the system with $l_O = 0.0$ is by far the slowest and shows a huge undershoot. Moreover, its final ISE has a value of 0.2224. As expected, the final ISE is slightly above the lower bound of $J_0 = 0.2196$. Concluding the results, for output feedback control it is recommendable to choose $l_O > 0.0644$ since the achievable control performance is not bounded by the excitement of RHP-zeros.

3.4 Experimental Validation

Only if the model shows a high degree of concurrence with the dynamics of the physical TWIP, model-based control and state estimation design will lead to satisfying results in practice. Thus, in this section, measurements from experiments are evaluated with simulation data calculated with the proposed model.

Typically for linear systems, the open-loop response to a step or impulse input of the dynamical model and the real system is compared. As the TWIP is nonlinear and has an unstable open-loop dynamic, such a simple comparison is not useful. Instead, the TWIP is balanced with a controller and then the control loop is opened by setting the inputs to zero at $t \approx 1.6\,\mathrm{s}$. The estimated states x_{B}, v_d, and α of the experiment as well

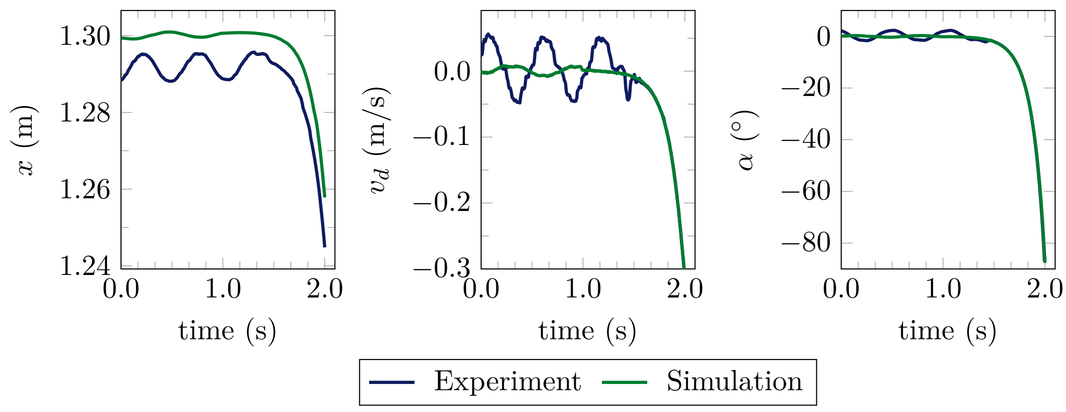


Figure 3.11: Switch off experiment (x, v_d, α)

as the simulation are presented in Figure 3.11. Firstly, it can be recognized that the limit cycle during the balancing is larger in the experiment than in the simulation. This is due to the better performance of the friction compensation, the controller as well as the state estimation in the simulation compared to the experiment as well as a friction parameter deviation around $v_d = 0$. Nevertheless, as soon as the loop is opened, the tip-over trajectory of the model perfectly fits with the experimental data.

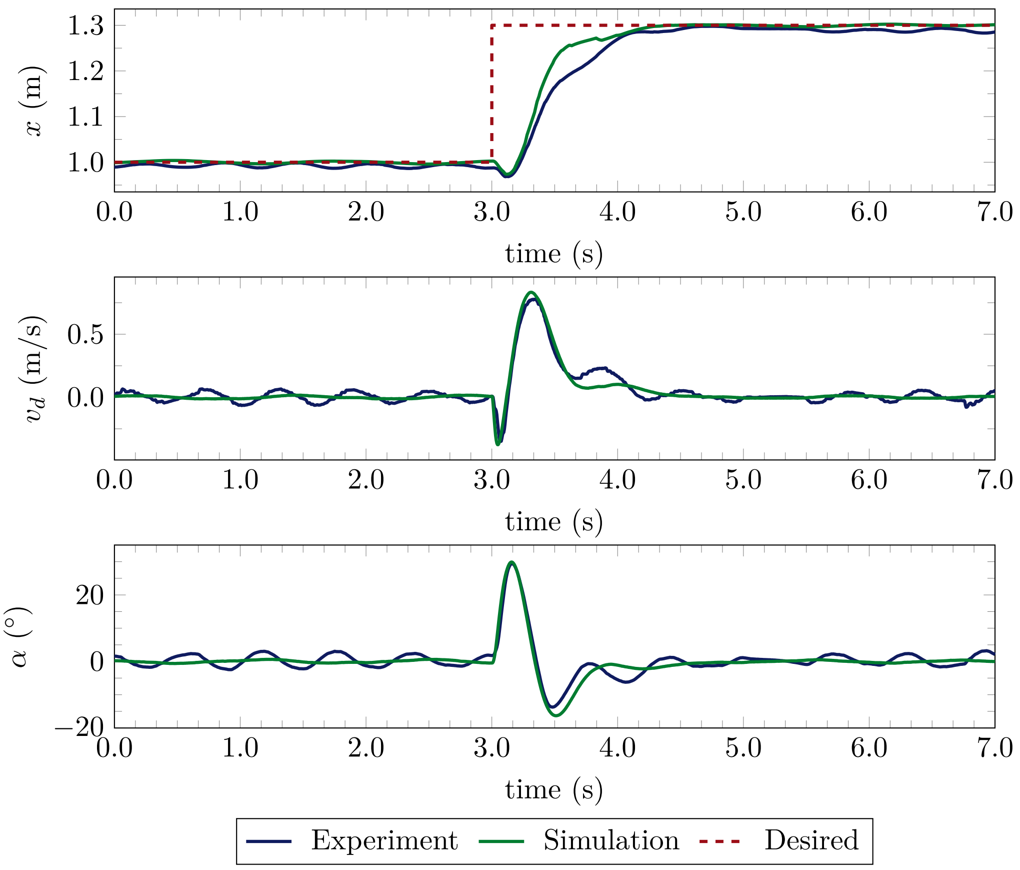


Figure 3.12: Step experiment (x, v_d, α)

As already mentioned, a comparison of the open-loop step response is not applicable.

Therefore, a step of 0.3 m in the forward direction d is commanded to the closed-loop simulation and experiment. The results, plotted in Figure 3.12, show a high degree of concurrence. Especially, the inverted initial reaction due to the non-minimum phase properties of the system can be recognized and simulation and experimental data show an almost perfect fit. In addition, the minimum and maximum values of v_d and α during the step in Figure 3.12 are quite similar. This indicates an excellent mapping between the physical system behavior and the simulation model. Compared to other published results, this congruence is outstanding.

Finally, a small comment on alternative evaluations presented in the literature is given, where the robot is rotated around θ to compare simulation and experimental results. The more difficult part is balancing the robot in its d direction as pure rotation is supported by the gyroscopic effects and thus, a high heading rate stabilizes the system (like a toy gyroscope) almost without any controller. Thus, in this case, the easier task compared to balancing and driving is presented since the linear dynamics (see Subsection 3.3.4) of the heading motions have no right half-plane (RHP) poles or zeros in their transfer function.

3.5 Concluding Remarks

In this chapter, a comprehensive model of the TWIP model has been presented. Compared to existing literature (e.g. Pathak et al. [84], Kim and Kwon [56], Delgado [23]) the dynamics of the motor currents are included directly in the formulation of the Lagrangian. In literature, the current dynamics are neglected or modeled as a separate subsystem and connected to the mechanical subsystem afterwards. In contrast, the proposed method is less error-prone and more well-rounded, as the electrical and mechanical systems are modeled jointly inside the Lagrangian framework. Based on the presented approach, the Lagrangian of the TWIP can be directly used to set up a variational integrator to receive a discrete model of the TWIP as presented in Albert et al. [2]. Another advantage of including the current dynamics is, that the nonlinear state space model has a constant input matrix which is beneficial for several nonlinear control design methods. Last but not least, the physical current limits can easily be incorporated in the simulation model and the model offers the possibility to consider them as state limits during control design and the estimation of the DoA. This enables a more sophisticated control synthesis.

Secondly, an insight into the system properties of the presented models with different levels of details is given, namely one model with current dynamics as well as one without current dynamics for the balancing *two-wheeled inverted pendulum mode* and a non-balancing model of the TWIP on ground for the *wheelchair mode*. Utilizing an exemplary trajectory, the interconnection and power exchange of the different energy storages of the system, included in the Lagrangian, has been analyzed. Based on this, negligible terms are detected as well as parts of the model which require particular attention during modeling. The provided information allows a comparison of different models and a model selection based on the requirements for the considered application. For example, the nonlinear current dynamics model for the closed-loop simulation, the linearized model with currents for controller and observer design, and the model without current dynamics to design a friction compensation. The model on ground may be used to speed up trajectory generation by using the solution gained with this simplified model as an initial guess for the trajectory generation with the current dynamics model.

In the consciousness of non-ideal sensors, a model for each of the three onboard sensors is presented, which is used in the simulation environment of the TWIP. The introduced sensors model provides the required congruence to the physical system for simulation-based observer tuning and includes sampling, quantization, and digital on-chip filters. Even though, the sensors show non-negligible dynamics due to their filters in the presented application, such a treatment of the sensors have not been presented before for the TWIP.

In addition, a versatile treatment of the linearized models has been provided. State transformations were introduced to decouple the dynamics into two subsystems, one for the heading dynamics and one for the forward dynamics of the robot. These models could ease the controller design and reveal additional system insights. Moreover, the eigenvalues of the LTI-models as well as the poles and zeros of the subsystems transfer functions are presented and an interpretation to physical system properties is given. Furthermore, the connection between the choice of the system's output to performance limitations have been analyzed thoroughly. As a result, possible pitfalls in the design of feedback controllers are revealed, especially due to unmodeled system dynamics, limited bandwidth as well as state limits and input saturation. For the first time, such a comprehensive treatment has been published which offers novel insights for modeling and control of the TWIP.

Finally, experimental results confirm a high degree of concurrence of the model with the physical TWIP. In particular, the closed-loop response to a position step in the experiments shows a remarkable match to the simulation results. Based on the proper and control-orientated system design presented in Chapter 2, coupled with the thorough modeling of the TWIP introduced in this chapter, an excellent mapping between the physical system behavior and the simulation is gained. Compared to results published by other authors, this congruence is outstanding and builds the foundation for a successful controller and observer design in the later chapters.

Chapter 4

Optimal Trajectory Generation

An offline procedure to generate an optimized trajectory for the TWIP is presented in this chapter. In particular, the objective is to follow through pre-specified positions and orientations at pre-defined times (so-called checkpoints) while fulfilling state and control constraints. Moreover, energy optimal trajectory generation is performed to conserve resources and not only a simple control input minimization. For this, a two-stage initialization is introduced to speed up optimization and to improve the convergence of the nonlinear optimization problem. Additionally, the trajectory is computed such that sequences of it can be cut and repeated to generate a longer trajectory to perform long-term test runs. The resulting trajectory will be used in the following chapters for (online) closed-loop control and to evaluate the proposed control and state estimation algorithms.

The chapter is structured as follows: at the beginning, a quick motivation for trajectory optimization is given. In the subsequent section, the optimization procedure is presented. Afterwards, the optimized reference trajectory, used in the following chapters, is presented. Finally, concluding remarks on the chapter are given.

This chapter complements the publication Albert et al. [2] of the author by adding more detailed information about the implementation of the optimization problem and the resulting trajectory. Whereas Albert et al. [2] focuses on the application of the variational integrator (VarInt) discretization method and discrete mechanics, first published in Phogat et al. [88], in this chapter the aspects of the Runge-Kutta (RK) discretization schemes are discussed. Moreover, some fine-tuning on the optimization algorithms and the checkpoints, compared to Albert et al. [2], led to a reduction of the computation time, especially for the VarInt discretization method, which is also discussed. In addition, so-called equalitypoints are included to be able to slice and repeat the optimized trajectory. To conclude, there is an unavoidable overlap of the contributions and content of this chapter and Albert et al. [2] but this chapter adds necessary details about the optimization procedure. Moreover, the resulting reference trajectory is required to make this work self-contained.

4.1 Motivation

Let us pick up the use-case of the TWIP for surveillance applications as introduced in Section 1.1 and consider a robot that has to drive through different checkpoints to observe the surrounding. If more than one robot is in use, a collision has to be avoided and if e.g. doors have to be passed at a specific time, a simple path-following is not possible, and instead, a trajectory is required. In the case of safety-critical systems, like the unstable TWIP, state and control constraints have to be respected during the generation of the trajectory. In addition, mobile robots require an energy supply, which is typically a battery. Thus, the trajectory has to drive from a base (e.g. charging station) to the desired area of operation, and then a sliced part of the trajectory is repeated e.g. a surveillance application as described. Afterwards, if the battery is low, the robot has to drive from the operating area back to the charging station. To be able to operate as long as possible and to reduce the time in the charging station, an energy-efficient trajectory is desirable.

One option to generate such an energy-optimal trajectory, subject to nonlinear system dynamics as well as state and control constraints, is to set up a nonlinear optimization. In the next section, such a nonlinear optimization procedure is presented and used to optimize a trajectory, which might be used for a surveillance task as described.

4.2 Optimization Procedure

Firstly, we define the performance measure for energy optimal control of the TWIP which should be minimized. The energy $e(u, x_c)$ consumed from the battery by the robot is calculated by

$$e(u, x_c) = \int_{t_0}^{t_e} (u_\mathrm{L} i_\mathrm{L} + u_\mathrm{R} i_\mathrm{R})\,\mathrm{d}\tau \; . \tag{4.1}$$

This is the time-integral of the electrical input power of the two DC-motors as defined in (3.99) and is used as a performance measure to be minimized. Besides, the trajectory has to fulfill the system dynamics as well as the state and control constraints. Thus, let us cast (4.1) used for the performance measure and the dynamic model of the TWIP with current dynamics (c-model) (3.60) into one ordinary differential equation (ODE)

$$\begin{bmatrix} \dot{x}_c \\ \dot{e} \end{bmatrix} = \mathcal{F}_{ode}(x_c, u_c) = \begin{bmatrix} f_c(x_c) + G_c u \\ u_\mathrm{L} i_\mathrm{L} + u_\mathrm{R} i_\mathrm{R} \end{bmatrix} \tag{4.2}$$

for optimization.

As the TWIP controller is executed in discrete-time, a discrete-time trajectory is required and the natural choice is to choose the sample time of the controller of 5 ms for the step length for the discrete optimization. Thus, a scheme to discretize (4.2) is needed. Moreover, typically explicit schemes with a fixed time step length are required for efficient optimization algorithms. Therefore, different schemes are compared in the following to find the best-suited scheme for the intended purpose. Firstly, the Runge-Kutta schemes are considered and compared as they are the standard schemes for numerical discretization. In particular, RK1, RK2 and RK4 are covered, where RKn is the n^{th} order of the scheme. Secondly, the VarInt scheme is evaluated, as presented in Albert et al. [2], as it has the advantage of preserving system invariants like momentum and energy and is more accurate than conventional techniques as discussed in Marsden and Ratiu [71]. In Section 4.3, the required computation time of the different discretization schemes will be compared based on an optimized reference trajectory.

We start with the RK4 scheme with which (4.2) is discretized as follows

$$[\mathcal{K}_{x,1}, \mathcal{K}_{e,1}]^{\mathrm{T}} = \mathcal{F}_{ode}(x, u) \tag{4.3a}$$

$$[\mathcal{K}_{x,2}, \mathcal{K}_{e,2}]^{\mathrm{T}} = \mathcal{F}_{ode}(x + \frac{\Delta_T}{2}\mathcal{K}_{x,1}, u) \tag{4.3b}$$

$$[\mathcal{K}_{x,3}, \mathcal{K}_{e,3}]^{\mathrm{T}} = \mathcal{F}_{ode}(x + \frac{\Delta_T}{2}\mathcal{K}_{x,2}, u) \tag{4.3c}$$

$$[\mathcal{K}_{x,4}, \mathcal{K}_{e,4}]^{\mathrm{T}} = \mathcal{F}_{ode}(x + \Delta_T\mathcal{K}_{x,3}, u) \tag{4.3d}$$

$$x_c = x_c + \frac{\Delta_T}{6}(\mathcal{K}_{x,1} + 2\mathcal{K}_{x,2} + 2\mathcal{K}_{x,3} + \mathcal{K}_{x,4}) \tag{4.3e}$$

$$e = e + \frac{\Delta_T}{6}(\mathcal{K}_{e,1} + 2\mathcal{K}_{e,2} + 2\mathcal{K}_{e,3} + \mathcal{K}_{e,4}) \ . \tag{4.3f}$$

As can be seen, the energy is also calculated within the scheme in (4.3f) to receive a precise result of the energy integral (4.1). The schemes RK1, RK2 have the same structure but with fewer function evaluations per discretization step length Δ_T. For proper discretization, the stability of the scheme has to be ensured and thus the step length Δ_T has to be appropriately selected. In particular, for the Dahlquist test equation $\dot{y} = \lambda y$ the stability region of the RK1 and RK2 schemes is known to be $-\lambda\Delta_T < 2.0$ and for the RK4 scheme it is $-\lambda\Delta_T < 2.8$ as shown in Hairer and Wanner [41]. Due to the fastest eigenvalue of the c-model $\lambda_8 = -3721.14$, as listed in Table 3.2 for the linearized model, 10 RK1 or RK2 substeps with $\Delta_T = 0.5\,\mathrm{ms}$ and 7 substeps for the RK4 are chosen with $\Delta_T = 0.714\,\mathrm{ms}$ and for time discretization of (4.3). As a result, for one sample step of ${}_{R}m = 0.5\,\mathrm{ms}$ in the optimization, the RK1 and RK2 schemes are repeated 10 times and the RK4 scheme 7 times.

In addition, the trajectory has to be optimized with respect to additional state and control constraints, which are chosen as follows:

(c-i) Input voltage ($u_{\mathrm{L}}, u_{\mathrm{R}}$): [-5, 5]V,

(c-ii) Input voltage rate ($\dot{u}_{\mathrm{R}}, \dot{u}_{\mathrm{L}}$): [-2, 2]V/s,

(c-iii) Motor current ($i_{\mathrm{L}}, i_{\mathrm{R}}$): [-3, 3]A,

(c-iv) Tilt angle (α): [-15, 15]°,

(c-v) Heading angle rate (v_θ): [-120, 120]°/s.

These constraints are introduced in Subsection 3.1.9 but are chosen tighter than the physical limits. This is necessary, as the optimized trajectory is used for feedforward control and has to share the available control input with the feedback controller. Moreover, if the optimal trajectory states and inputs are close to the physical state limits, in case of small disturbances, the control algorithm might fail to stabilize the system.

In addition, besides the fixed initial state $x_c[0]$ and the final state $x_c[N]$ for the optimal trajectory, the TWIP should pass through $N_m \leq N$ pre-specified checkpoints $\{\bar{x}_{k_j}\}_{j=1}^{N_m}$, where N is the number of discretization steps for the optimization.

To be able to slice and repeat the trajectory, $N_e \leq N$ equalitypoints $\mathcal{P}_e = \{\bar{k}_i, \bar{k}_{e,i}\}_{i=1}^{N_e}$ are enforced, where the state vectors $x_c[\bar{k}_i] = x_c[\bar{k}_{e,i}]$ have to be identical.

Finally, the discrete-time optimal control problem can be formulated with the integration scheme (4.3) and the constraints (c-i)-(c-v) as well as the checkpoints and equalitypoints by

$$
\begin{aligned}
&\underset{\{u[k],x_c[k]\}_{k=0}^{N-1}}{\text{minimize}} \quad \mathbb{J} := \sum_{k=0}^{N-1} e[k+1](u[k], x_c[k]) \text{ (discrete energy)} \\
&\text{subject to} \\
&\begin{cases} x_c[k+1] = \mathcal{F}_{dis}(x_c[k], u[k]) \text{ (discrete dynamics)} \\ -5 \leq u_{\mathrm{R}}[k], u_{\mathrm{L}}[k] \leq 5 \end{cases} \\
&\qquad \text{for } k = 0, \ldots, N-1, \\
&\begin{cases} -3 \leq i_{\mathrm{R}}[k], i_{\mathrm{L}}[k] \leq 3 \\ -\frac{\pi}{12} \leq \alpha[k] \leq \frac{\pi}{12} \\ -\frac{2\pi}{3} \leq v_\theta[k] \leq \frac{2\pi}{3} \\ -2_R m \leq u_{\mathrm{R}}[k-1] - u_{\mathrm{R}}[k] \leq 2_R m \\ -2_R m \leq u_{\mathrm{L}}[k-1] - u_{\mathrm{L}}[k] \leq 2_R m \\ x_c[k] = \bar{x}_c[k_j] \ \text{ if } \ k = k_j \text{ for any } \ j = 1, \ldots, N_m \\ x_c[k] = x_c[\bar{k}_{e,i}] \ \text{ if } \ k = \bar{k}_i \ \in \mathcal{P}_e \end{cases} \\
&\qquad \text{for } k = 1, \ldots, N-1, \\
&x_c[0] = \bar{x}_c[0], \\
&x_c[N] = \bar{x}_c[N].
\end{aligned} \tag{4.4}
$$

The minimization problem (4.4) is implemented in MATLAB 2020b by the use of CasADi v3.5.5 from Andersson et al. [4] and solved with IPOpt v3.12.3 from Wächter and Biegler [118].

Unfortunately, as introduced, the nonlinear optimization takes a large computation time to solve or does even not converge at all if it is not properly initialized. The IPOpt solver requires an initial guess for the optimization values and the closer they are to the optimal solution, the more likely the algorithm converges and the time to compute the solution can be reduced. In experiments, in many cases, the optimization failed to solve (4.4) with the default initialization (zero vector). Thus, a simplified optimization problem with the g-model (3.91), as presented in Subsection 3.1.8, is set up. As the nonlinear dynamics are much slower, a RK1 scheme with a single step for the time step length of 5 ms has been found to be sufficient. Moreover, the reduced number of states eases the optimization problems in magnitudes. As the motor currents are not included in the g-model, the energy is calculated by evaluating the static motor current equation (3.68) and the value of the input. Then, the solution of this first stage optimization will be used to initialize the optimizations of the second stage, solving (4.4) with different discretization schemes for comparison purposes, namely RK1, RK2 and RK4 and VarInt.

To conclude, a two-stage optimization is used to compute the optimal discrete-time trajectory. Firstly, the g-model is discretized and used to generate a good initial guess for the optimization with the c-model. Secondly, the c-model is discretized with different schemes and the optimization problem (4.4), initialized with a solution from the g-model, is solved. In the next section, the optimization of the reference trajectory is presented, based on the optimization procedure introduced.

4.3 Reference Trajectory

To test the proposed optimization procedure and to compare the required computation time of the different discretization schemes, a reference trajectory has been set up. In addition, the RKn schemes are also tested with 1 substep to ease the comparison with the results presented in Albert et al. [2] as well as the VarInt scheme. This trajectory includes a path from an initial start point to the desired area, then an *eight-knot* and a *zig-zag* path as well as a path back to the initial start point. The x-y phase plot of the optimized trajectory is plotted in Figure 4.1 and could be the required trajectory of a surveillance application as introduced. In particular, the optimized trajectory is

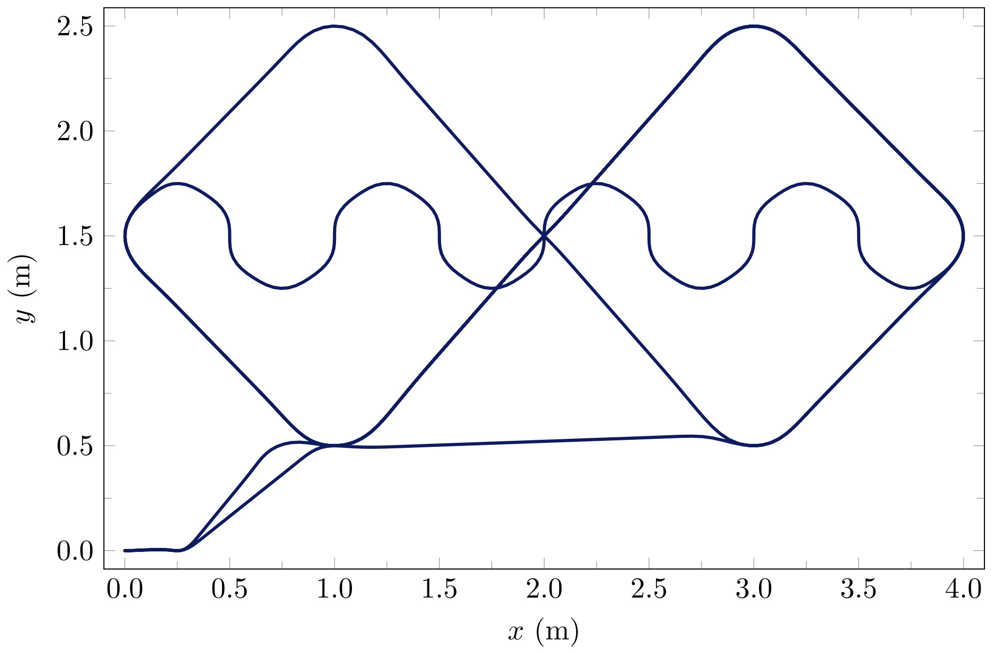


Figure 4.1: Trajectory x-y phase

parameterized and partitioned into different parts as follows:

- Step length ($_Rm$): 5 ms,
- Final time (T):
  1. The *eight-knot*: 21.68 s,
  2. The *zig-zag*: 19.52 s,
  3. Complete trajectory: 81.38 s
- Number of Steps ($N = T/_Rm$)
  1. The *eight-knot*: 4336,
  2. The *zig-zag*: 3904,
  3. Complete trajectory : 16 276.

To specify the trajectory, 37 checkpoints are defined and shown in Figure 4.2 as red crosses. Herein, checkpoints define that the trajectory has to fulfill the $x_\text{B}, y_\text{B}, \theta$ requirement at a defined time instance. Moreover, 16 equalitypoints are specified and marked

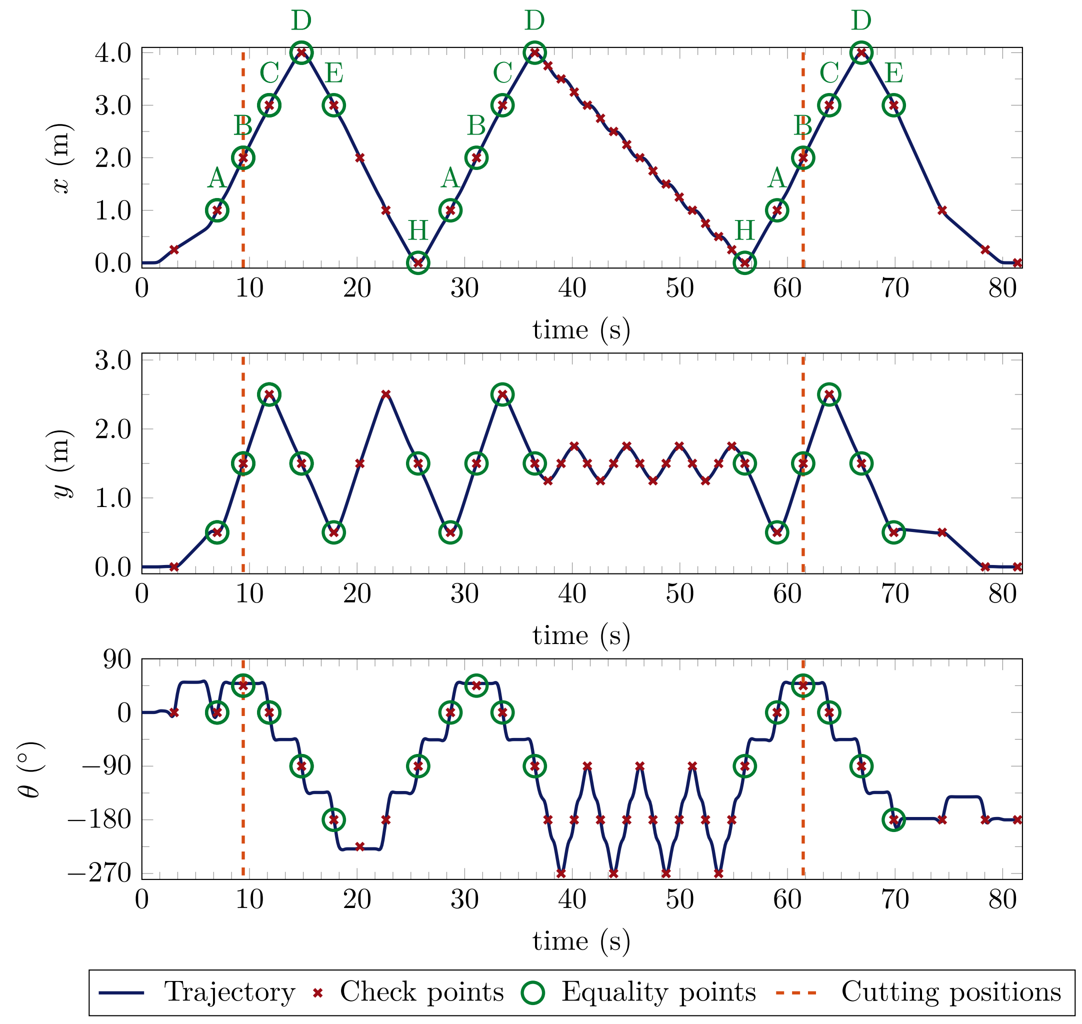


Figure 4.2: Optimal trajectory states (x_B,y_B,θ)

in Figure 4.2 with green circles and labeled with the letters ‘A’ up to ‘H’. The equality-points labeled with the same letter, are forced to have the same state vector x_c which leads to 10 equality constraints in (4.4). Due to this, the resulting trajectory can be cut and repeated after the optimization. The position, between which the trajectory is cut and repeated is marked with dashed red lines at $t_1 = 9.42\,\mathrm{s}$ and $t_2 = 61.46\,\mathrm{s}$ in Figures 4.2 and 4.4 as well as the plot of the control inputs in Figure 4.5. For long-term benchmark runs, the sequence between t_1 and t_2 is repeated 80 times, leading to an optimal trajectory with a total length of $\approx 70\,\mathrm{min}$.

The introduced reference trajectory is optimized on a machine with CPU - Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, RAM - 8 GB, OS - Windows 10 64-Bit. On this machine in sum 77 s are required to assemble and solve the optimization problem with the g-model. Herein, the time for CasADi to assemble the problem is around 16 s. The resulting trajectory is then used to initialize the c-model in the next step.

Depending on the discretization, the trajectory optimization problem with the c-model is solved between 206 s and 2241 s. Besides small numerical differences, the solutions of the optimizations with different schemes were identical. The total computation time is the sum of the time required to step up the problem and the time needed to solve it. In particular, CasADi requires between 6 s for RK1with 1 substep and 17 s for RK4 with 7 substeps to set up the problem. The time required to set up the task using the VarInt discretization scheme requires 17 s and is thus comparable with the RK4 with 7 substeps.

In Figure 4.3 the total computation time to assemble and solve the optimal trajectory problem with the c-model is plotted. In sum, the time required by MATLAB and

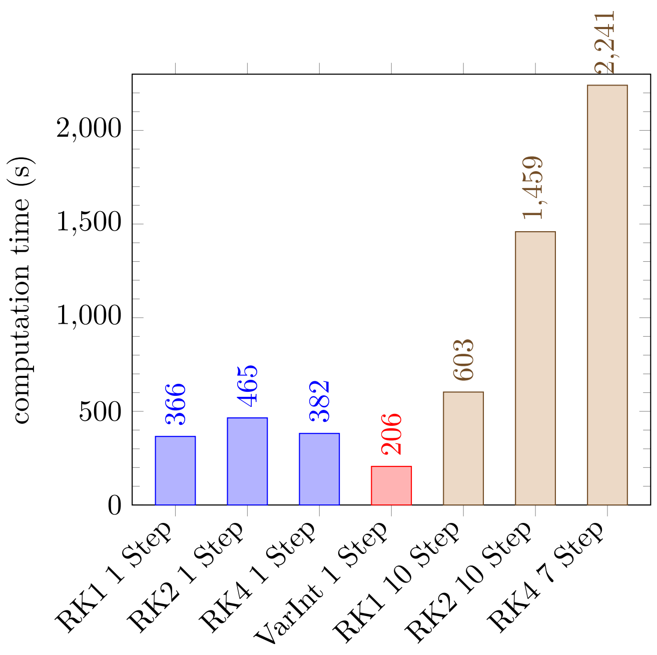


Figure 4.3: Computation time comparison

CasADi to set up the optimization task is negligible compared to the time required to solve the problem with the IPOpt solver. Notice, that the required computation times are between 14 % for RK2 and 60 % for the VarInt reduced compared to reported values in Albert et al. [2]. This is the result of choosing a different, non-default option for the treatment of fixed variables[1] in the IPOpt solver, which leads to the same optimal trajectory but requires less computation power. Especially the VarInt discretization scheme outperforms the classical RKn schemes even more than it has been presented in Albert et al. [2].

Due to the fast current dynamics considered in the model, a proper discretization using the RKn schemes demands 10 or 7 substeps. As visualized in Figure 4.3, in this case, the computation time rises up to 603 s for RK1, 1459 s for RK2 and 2241 s for RK4. Thus, the traditional RKn schemes require for this benchmark optimization between 3 and 11 times more computation time than the VarInt scheme.

Finally, an optimization run with the c-model and VarInt without the initial solution from the g-model has been started, to evaluate the advantage of the proposed two-stage optimization procedure. In this case, IPOpt stopped after 2000 iterations, which took over 10 h, without providing an optimal trajectory. This highlights, that the proposed procedure reduced the required optimization time or even enables the IPOpt solver to solve the problem at all.

[1]In particular, the option 'fixed_variable_treatment' has been set to 'relax_bounds'.

4.4 Concluding Remarks

A method to compute an energy optimal trajectory offline, subject to the nonlinear TWIP model with current dynamics and state and input constraints has been presented. Herein, the introduced optimization problem minimizes the input energy of the TWIP and not only control input. In addition, by the use of pre-specified checkpoints and so-called equalitypoints, the desired trajectory can be precisely specified. Moreover, an optimized reference trajectory is presented which can be cut and repeated for long-term test runs. Also, this chapter includes improvements in the setup of the optimization task and adds detailed information about the implementation and optimization compared to Albert et al. [2].

A major contribution is the introduced two-stage optimization scheme using the simpler g-model in the wheel-chair mode to compute an initial guess for the optimization task using the c-model of the balancing TWIP with current dynamics. Moreover, additional analyses are presented to compare the standard RKn with VarInt discretization scheme as presented in Albert et al. [2]. All tested schemes were applicable for the optimization and all results were similar besides small numerical inaccuracies. Using VarInt scheme for discretization, the optimization required about 44 % less computational power compared to the fastest RKn scheme. In addition, the VarInt scheme preserves system invariants which makes the VarInt discretization scheme superior compared to the standard RKn schemes. This result coincides with Albert et al. [2] but in this chapter, it has been revealed that VarInt outperforms the standard schemes even more than it has been presented by Albert et al. [2].

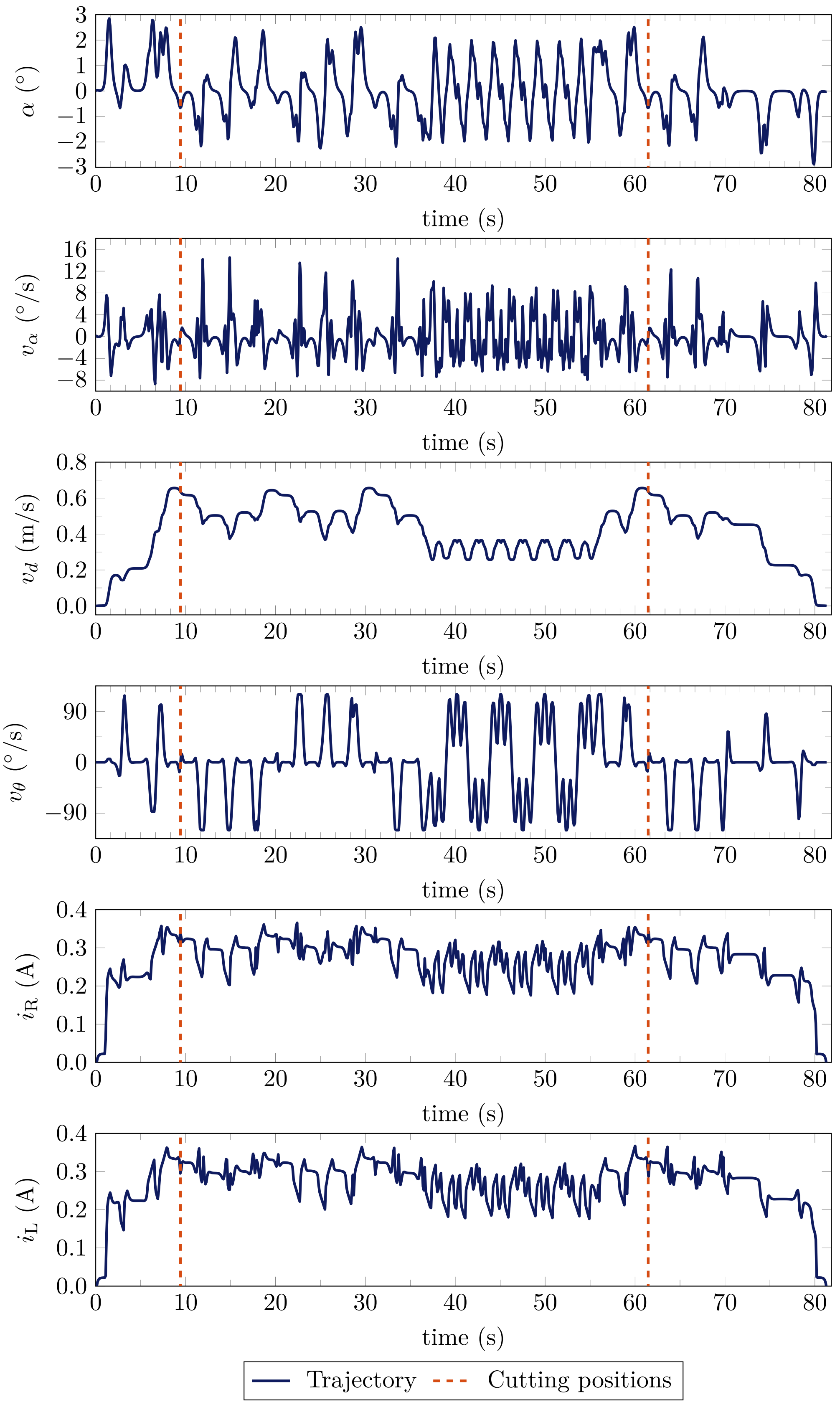


Figure 4.4: Optimal trajectory states (α,v_α,v_d,v_θ,i_R,i_L)

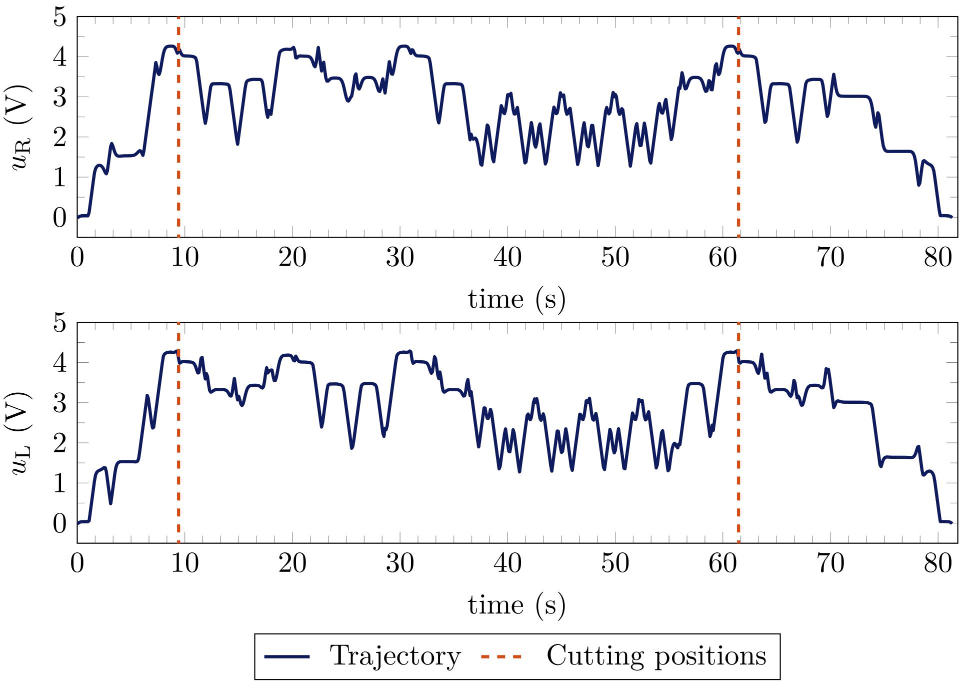


Figure 4.5: Optimal trajectory input

Chapter 5

Setpoint and Trajectory Tracking

In this chapter a novel discrete-time feedback structure is presented as shown in Figure 5.1, to stabilize the TWIP and to perform setpoint or trajectory tracking. A major contribution is the derivation of a full discrete-time treatment of all elements of the control structure as well as results for a command governor for trajectory tracking applied for the first time on a MIMO-system in experiments.

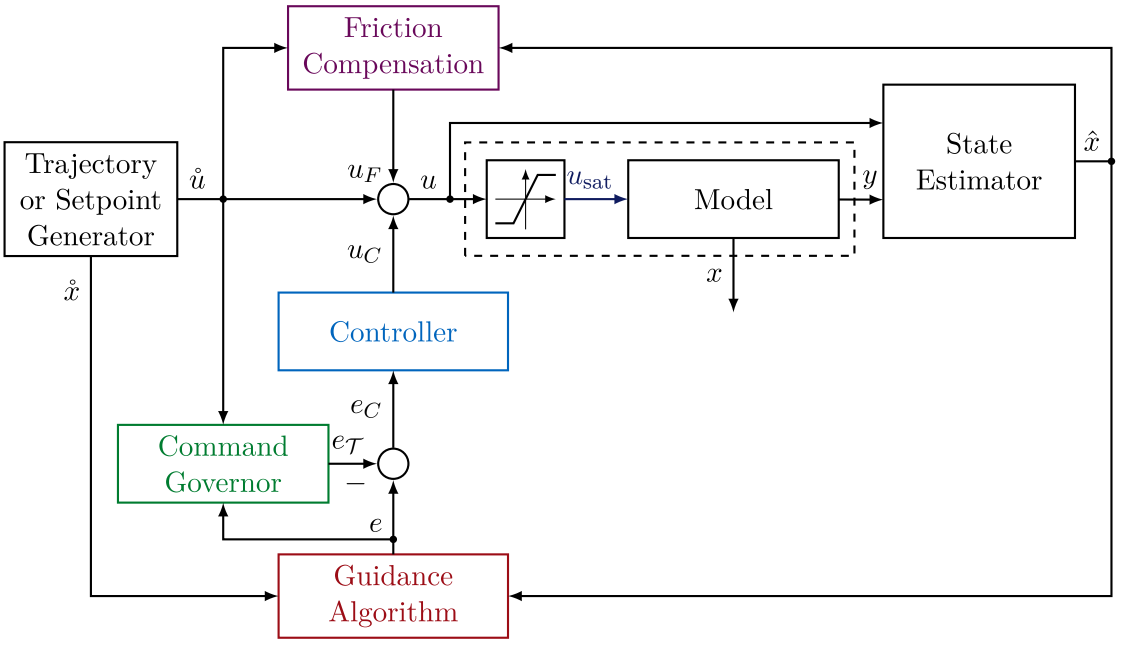


Figure 5.1: Control structure with command governor

Due to the state limits and the input saturation of the physical system, introduced in Subsection 3.1.9, the DoA is limited and stabilization will fail if the control error e_C is outside the DoA. To ensure a stable operation during setpoint tracking and trajectory tracking, a command governor algorithm for each of the two tracking modes is presented. Thereby, the stability of the closed-loop system is ensured by the use of a QLF inside the command governor with a known level set, bounding the estimated stable region. Originally, Buhl and Lohmann [13] proposed a setpoint command governor for continuous-time systems with one control input. Based on this, modifications for trajectory tracking and systems with multiple inputs have been presented by Dessort [27], Pieczona [89], Diepold and Pieczona [28] and Diepold [31]. All introduced algorithms in these publications consider continuous-time models, trajectories and control. Contrary in this chapter, a novel discrete-time realization is introduced and its applicability is proven in simulations as well as experiments. Moreover, in Diepold [31] the command governor for trajectory tracking has only been applied on the SISO-system 'Inverted pendulum on a cart' in experiments. Contrary in this chapter, the algorithms are applied on the TWIP and thus experimental results for a MIMO-system are presented which additionally underlies nonholonomic constraints.

The chapter is structured as follows: In the first section, it is motivated why the control structure, as illustrated in Figure 5.1, with a command governor is proposed. In the subsequent section, friction compensation, which reduces the effects of nonlinear mechanical friction, is introduced. This approach ensures the congruence of the LTI-model proposed in Section 3.3 for all velocities. Afterwards, a guidance algorithm is presented to ensure, that the desired trajectory can be reached even though the TWIP underlies nonholonomic constraints. Furthermore, a stabilizing linear constant feedback controller is designed based on the discrete-time LTI-model of the robot with motor currents as presented in Section 3.3. All control algorithms are considered to be discrete-time, as they will be executed cyclically on the robots MCU, which is novel. Based on the stabilizing linear constant feedback controller, the DoA is estimated in Section 5.5 and used in the consecutive sections by the therein introduced command governors. Besides the theoretical derivation of the command governor algorithms, experimental results are presented. Finally, concluding remarks on the chapter are given.

The development, implementation, and first experimental testing of the guidance algorithm, the setpoint tracking command governor, as well as the trajectory tracking command governor, has been intensively supported by the master's thesis of Anhalt [5], supervised by the author. In consequence, the algorithms and first experimental results are also presented there. The introduced algorithms are the outcome of almost daily meetings, creative discussions, and troubleshooting and debugging done jointly. In consequence, this chapter includes the non-separable contribution of Anhalt [5] and the author. While the underlying principles are the same as already presented in Anhalt [5], some notations and derivations as well as experimental results differ in the following sections.

5.1 Motivation

Typically, a 'standalone' controller for unstable systems has a very limited DoA in presence of input saturation and state limits. In consequence, stabilization will fail if the control error is outside the DoA of the designed stabilizing controller.

Such a large control error might happen due to one of the three sources introduced, among others:

- a large disturbance acting on the robot increasing the control error,
- a large step in the desired state $\mathring{x}$,
- a measurement update causing a large change in estimated state $\hat{x}$ used to calculate the control error.

The 1st source is commonly mentioned but not the 2nd and 3rd and thus an explanation is given.

Performing setpoint tracking, it might be the case, that the operator or an upper-level process control has no knowledge about the DoA of the robot. If a new desired position and orientation are then commanded to the robot, a large control error outside the stable region could occur. Assuming trajectory tracking, it might happen that a new trajectory is planned due to a mission change while the robot follows another. For sure, the ideal case is, that the new trajectory starts exactly at the current state of the robot. But if the planning of the trajectory requires some amount of time, the robot's state might already differ from the state assumed as the initial state of the new trajectory which results in a large control error after switching to the new trajectory.

Another issue for a large control error may be linked to the state estimation. Commonly, local sensors are used for odometry in combination with a remote global measurement to estimate the position and orientation of the robot. If the remote measurement has delays or dropouts, the position and orientation have to be calculated by integration temporarily which accumulates measurement errors. Thus, the estimated position and orientation will differ more and more from the real one over time as long as no remote measurements are available. If a new measurement is received after a while, this might cause a large correction step and thus a large change in the estimated state. As the control error is calculated with the estimated state $\hat{x}$, this might cause an increase in control error and an abandonment of the DoA.

Since all three introduced issues might happen during operation, not only a friction compensation and a stabilizing controller is presented in the next sections but also a command governor to drastically increase the DoA of the pure standalone controller and prevent failing stabilization, if possible.

5.2 Friction Compensation

Recently, Dai et al. [21] presented a sliding mode controller applied to a TWIP to avoid the negative effects of nonlinear friction. As the control algorithms proposed in this chapter, are designed with the linear state space model of the robot as introduced in Section 3.3, this sliding mode approach does not apply. Moreover, a good control performance with a controller designed for the linearized system may only be reached, if the congruence of the linear and nonlinear model is high in the domain of operation. In consequence, the nonlinear components of the friction torque are compensated through the plant input by the approach presented by Delgado [23]. Contrary to Delgado [23] and as already mentioned in Section 3.3, only the nonlinear components will be compensated, since this will ease the allocation of the limited control input between the controller and the friction compensation. Thus the desired goal is to remove the $tanh()$ terms modeling the stick-friction in (3.37).

At least for the models without current dynamics (n-model) this can be achieved. In the equation of the reduced system dynamics (3.83) the terms (3.70) with the nonlinear friction as well as (3.71) for the plant input are summed up. Therefore, a full friction compensation through the input u has to fulfill

$$\mathcal{F}_{\text{dis}} + \underbrace{\mathcal{E}u}_{\mathcal{F}_{\text{ext}}} = 0 \; . \tag{5.1}$$

As Delgado [23] argued, this is only possible if the condition

$$\mathcal{E}_\perp \mathcal{F}_{\text{dis}} = 0 \tag{5.2}$$

is fulfilled, where $\mathcal{E}_\perp$ is the left annihilator of $\mathcal{E}$. The left annihilator,

$$\mathcal{E}_\perp = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{5.3}$$

leads to $\mathcal{E}_\perp \mathcal{E} = 0$ and also fulfills (5.2) in the case of the n-model. In other words, the friction terms in $\mathcal{F}_{\text{dis}}$ lie in the image of the plant input matrix $\mathcal{E}$.

Let us assume, $\mathcal{F}_{\text{dis}}$ includes only the nonlinear friction, then (5.1) is given by

$$0 = \begin{pmatrix} 0_{3\times 2} \\ 1_{1\times 2} \\ -I_{2\times 2} \end{pmatrix} \begin{pmatrix} d_{\text{C}} \tanh\left(d_0 \omega_{\delta\text{R}}\right) \\ d_{\text{C}} \tanh\left(d_0 \omega_{\delta\text{L}}\right) \end{pmatrix} \underbrace{- \frac{n_{\text{WM}} k_{\text{M}}}{R_{\text{M}}} \begin{pmatrix} 0_{3\times 2} \\ 1_{1\times 2} \\ -I_{2\times 2} \end{pmatrix}}_{\mathcal{E}} \underbrace{\begin{pmatrix} u_{\text{R}} \\ u_{\text{L}} \end{pmatrix}}_{u} . \tag{5.4}$$

Based on this, the plant input can be calculated by

$$u_F = - \left(\mathcal{E}^{\text{T}} \mathcal{E}\right)^{-1} \mathcal{E}^{\text{T}} \begin{pmatrix} 0_{3\times 2} \\ 1_{1\times 2} \\ -I_{2\times 2} \end{pmatrix} \begin{pmatrix} d_{\text{C}} \tanh\left(d_0 \omega_{\delta\text{R}}\right) \\ d_{\text{C}} \tanh\left(d_0 \omega_{\delta\text{L}}\right) \end{pmatrix} \tag{5.5}$$

$$= \frac{R_{\text{M}}}{n_{\text{WM}} k_{\text{M}}} \begin{pmatrix} d_{\text{C}} \tanh\left(d_0 \omega_{\delta\text{R}}\right) \\ d_{\text{C}} \tanh\left(d_0 \omega_{\delta\text{L}}\right) \end{pmatrix} , \tag{5.6}$$

which compensates the nonlinear friction terms. A nice detail is, that u_F only depends on $\omega_{\delta\text{R}}$ and $\omega_{\delta\text{L}}$ which are directly measured by the encoders. In consequence, a friction compensation can even be used if no state estimation is available.

As only the nonlinear friction components are compensated, the maximum input demanded is about $u_{F,max} = 0.26\,\text{V}$ whereas a full friction compensation would require about 4.3 V evaluated at a forward speed of 0.65 m/s.

Unfortunately, condition (5.2) cannot be fulfilled in the *c*-model, and in consequence, no 'perfect' friction compensation can be designed if the current dynamics are considered. Nevertheless, let us assume that if we use the compensation (5.6) on our physical robot, the nonlinear friction will be compensated sufficiently as the current dynamics are considerably faster than the mechanical dynamics. Based on this, let us assume that the nonlinear friction terms can also be dropped in the *c*-model before linearization, which is then used for linear control synthesis in Section 5.4.

5.3 Guidance Algorithm

As the system underlies nonholonomic constraints the reduction of the control error is a nontrivial task in the $x - y$ plane. For visualization, let us just consider the case with an error in the y coordinate and heading angle of zero and thus an error in the lateral direction of the robot. To reduce the error we may turn the robot about 90°, drive to the desired position in y and turn 90° back. Another option is, to drive slightly forward and turn a few degrees and then drive backwards and turn back again. In particular, there are infinite possible trajectories to reach the desired position even though they will differ in the time needed and energy consumption. To solve this problem in respect to a cost function (e.g. minimal time or energy) and additional constraints (e.g. $v_d \geq 0$), a nonlinear optimization will be needed. But even with a nonlinear optimization, several solutions might exist leading to the same minimal cost function value. Moreover, nonlinear optimizations usually are computationally intensive, require a long time to solve and might even fail to converge in some cases due to numerical issues.

Dengler and Lohmann [26] and Dengler [25] presented a controller in the form of a recurrent neural network trained with imitation learning using trajectories from nonlinear optimizations. In consequence, the controller has a kind of guidance algorithm included which imitates the learned trajectories to drive to the desired point. This method has two major drawbacks. First, to cover almost all possible trajectories to the desired point, the recurrent neural network has to be quite large or performance drawbacks have to be accepted. In particular, Dengler [25] used a fully connected neural network with 64 and 32 neurons in two hidden layers as well as 32 neurons in the output layer. In consequence, a powerful MCU is required to compute the neural network at each sample step. Secondly, the path planning or guidance is combined with the controller design and thus both, stabilization as well as setpoint tracking, have to be trained together. This increases the overall complexity of the neural network as well as the training procedure. This could have been avoided if the recurrent neural network had been split into two, one used for guidance and one for stabilization. In addition, the presented experimental results show a relatively large stationary offset but it is hard to judge about the source of this offset as in Dengler and Lohmann [26], Dengler [25] no error analysis is provided. It might result from the relatively small neural network, the control method itself, or could be caused by hardware issues. Moreover, Dengler [25] considered setpoint tracking only.

Another "extremely simple" approach for setpoint tracking has been proposed by Astolfi [6] with a nonlinear control law based on a transformed coordinate system. In particular, the presented paths where the robot is initially in different positions on a unit circle around the desired point as well as the published parking maneuver are quite promising and seem to be superior to other presented algorithms. Unfortunately, this method also combines controller and guidance design and has only been developed for setpoint tracking.

For setpoint tracking, there is one more quite simple approach, used by e.g. Delgado [23] and Strohm [102]. In this case, a line is considered between the $x-y$ coordinates of the robot and the desired point. The desired orientation $\mathring{\theta}$ is then chosen such that the robot will orientate to the angle of this line and the desired distance $\mathring{d}$ is set to the length of the line. To avoid helical trajectories close to the desired point, additional smoothing parameters and a dead-zone have to be included as discussed in Strohm [102].

In this section, a guidance algorithm for setpoint tracking as well as trajectory tracking is proposed, separating the guidance or path planning from the controller design. Moreover, to be able to use linear control laws, the error in the spatial space (x_B, y_B, and θ) is 'translated' into an error in forward direction d and orientation θ. In addition, the algorithms for setpoint tracking discussed above present stumbling blocks if they are applied to trajectory tracking. Let us simply consider, that we are moving quite fast in the forward direction but with a small offset in the lateral direction. The intuitive control action would be, to slightly change the orientation of the robot towards the desired point during driving and then slightly turn back, similar to changing lanes on a highway. Contrary to this intuitive control action, the above-mentioned algorithms try to directly drive to the desired point which leads to a command to turn 90° at full speed. Clearly, this is not the desired control action as we like to have a smooth approach to the trajectory. In the following, a guidance algorithm is introduced, which leads to a smooth approach if the robot is close to the desired trajectory such that the tracking error smoothly decreases. The first presentation of the guidance algorithm as well as the application on the TWIP can be found in Anhalt [5], supervised by the author.

Let us introduce the tracking error

$$\tilde{e} = \mathring{x} - \hat{x} = [\tilde{e}_{x_B},\ \tilde{e}_{y_B},\ \tilde{e}_{\theta},\ e_{\alpha},\ e_{v_\alpha},\ e_{v_d},\ e_{v_\theta},\ e_{i_R},\ e_{i_L}]^{\mathrm{T}} \tag{5.7}$$

where $\mathring{x}$ is the desired state and $\hat{x}$ is the estimated state of the robot. As we need an error in d and θ instead of $x_{\mathrm{B}}, y_{\mathrm{B}}, \theta$ coordinates, we introduce the tracking error

$$e = [e_d,\ e_\theta,\ e_\alpha,\ e_{v_\alpha},\ e_{v_d},\ e_{v_\theta},\ e_{i_R},\ e_{i_L}]^{\mathrm{T}} \tag{5.8}$$

transformed into the coordinates of the *c*-*d*-model including the distance error e_d and the orientation error e_θ, which are derived in the following.

First, based on $\tilde{e}$ the distance

$$\Delta_d = \sqrt{\tilde{e}_{x_B}^2 + \tilde{e}_{y_B}^2} \tag{5.9}$$

is calculated between the estimated position $(\hat{x}_B, \hat{y}_B)$ of the robot and the desired position $(\mathring{x}, \mathring{y})$ in the $x-y$ plane. In addition, the angle of the vector, pointing from $(\hat{x}_B, \hat{y}_B)$ to $(\mathring{x}, \mathring{y})$ can be calculated by

$$\Delta_\theta = \arctan\left(\frac{\tilde{e}_{y_B}}{\tilde{e}_{x_B}}\right) . \tag{5.10}$$

Up to this point, the method is quite similar to the approach presented by Strohm [102] and also used by other authors.

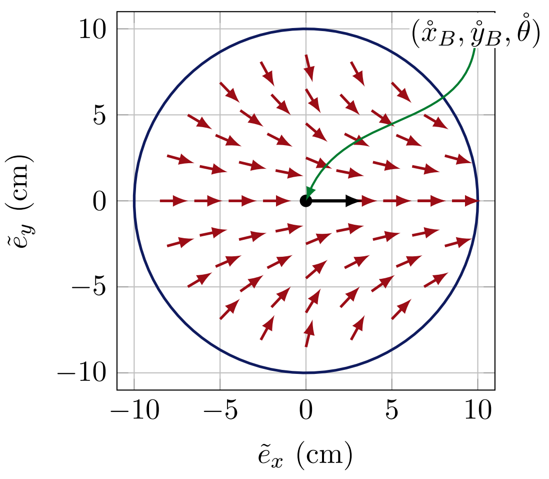
To provide a smooth approach to the desired trajectory, a radius R of 100 mm is considered around the desired position and the algorithm switches between two methods depending if e_d is larger or smaller than R. If $e_d \geq R$, the robot is outside the considered area and we intend to drive directly to the desired point. In consequence, the orientation error is calculated with

$$e_\theta = \Delta_\theta - \hat{\theta} . \tag{5.11}$$

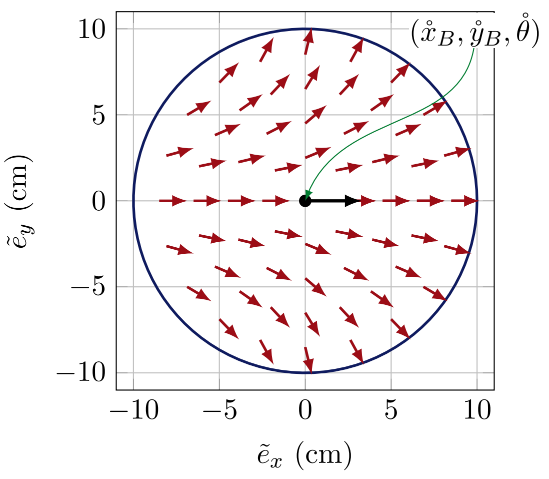
The calculation of the error in distance depends on the value of e_θ and is given by

$$e_d = \begin{cases} \Delta_d & \text{if} \quad |e_\theta| \leq \frac{\pi}{2} \\ 0 & \text{if} \quad |e_\theta| > \frac{\pi}{2} \end{cases} . \tag{5.12}$$

Thus, if the desired point is outside the 'view range' of the robot of $\pm\frac{\pi}{2}$ the distance error is set to zero to first turn the robot towards the desired point $(\mathring{x}_B, \mathring{y}_B)$.



(a) Velocity $v_d > 0.1\,\mathrm{m/s}$



(b) Velocity $v_d < -0.1\,\mathrm{m/s}$

Figure 5.2: Operation of the guidance algorithm inside R (inspired by Anhalt [5])

Contrary to the introduced first case, a smoother approaching algorithm is applied if $e_d < R$ to reduce the tracking error. In Figure 5.2 the behavior of the guidance algorithm in the close range ($e_d < R$) is illustrated for $\mathring{\theta} = 0$ for positive and negative velocities v_d of the robot. The red arrows indicate the desired direction in which the TWIP should move. For $\mathring{\theta} \neq 0$ the vector field shown will be rotated around the origin with $\mathring{\theta}$. In the first step, the error in distance is calculated by

$$e_d = \cos(\Delta_\theta - \mathring{\theta})\Delta_d \tag{5.13}$$

for this case, such that a large difference between Δ_θ and $\mathring{\theta}$ reduces the distance error. Subsequently, the auxiliary angle

$$\Delta_Z = \begin{cases} +(\Delta_\theta - \mathring{\theta}) & \text{if} \quad |\Delta_\theta - \mathring{\theta}| \leq \frac{\pi}{2} \wedge \hat{v}_d \geq 0 \\ -(\Delta_\theta - \mathring{\theta}) & \text{if} \quad |\Delta_\theta - \mathring{\theta}| \leq \frac{\pi}{2} \wedge \hat{v}_d < 0 \\ -(\Delta_\theta - \mathring{\theta}) + \pi & \text{if} \quad |\Delta_\theta - \mathring{\theta}| > \frac{\pi}{2} \wedge \hat{v}_d \geq 0 \\ +(\Delta_\theta - \mathring{\theta}) + \pi & \text{if} \quad |\Delta_\theta - \mathring{\theta}| > \frac{\pi}{2} \wedge \hat{v}_d < 0 \end{cases} \tag{5.14}$$

is calculated, pointing from the TWIP to the desired trajectory in relation to the velocity v_d.

For a smooth approach, e_θ is scaled with the distance Δ_d, such that the angle gets smaller as the closer the robot is to the trajectory. In addition, to gain a smooth approach even with slow velocities below $v_{\text{lim}} = 0.1\,\mathrm{m/s}$, an additional scaling is introduced. e_θ is finally given by

$$e_\theta = \begin{cases} \mathring{\theta} - \hat{\theta} + \Delta_Z \frac{\Delta_d}{R} & \text{if} \quad |\hat{v}_d| \geq v_{\text{lim}} \\ \mathring{\theta} - \hat{\theta} + \Delta_Z \frac{\Delta_d}{R} \frac{|\hat{v}_d|}{v_{\text{lim}}} & \text{if} \quad |\hat{v}_d| < v_{\text{lim}} \end{cases} . \tag{5.15}$$

To summarize, the introduced guidance algorithm uses two different approaches depending on the distance between the robot and the desired point on the trajectory to calculate a pseudo-control error e. Based on e, a linear state space controller is able to stabilize the system as well as follow a trajectory smoothly.

5.4 Linear Constant State Feedback

For stabilization and tracking a linear-quadratic regulator (LQR) is proposed. In Subsection 3.3.2 it has been shown, how the linear state space model can be split into two separate models, one for the forward dynamics and one for the heading dynamics with the state vectors as defined in (3.122) and new inputs given in (3.120) and (3.121). In the following, the decoupled models are used to ease the choice of the control parameters. Moreover, to reduce the number of parameters even further, two linear-quadratic state-feedback regulators with output weighting (LQRYs) are used and the outputs are chosen considering the performance limitations discussed in Subsection 3.3.4. The input saturation and state limits are neglected for the LQR design procedure.

As all algorithms are executed on the MCU with a sample time of 5 ms, both linear state space models are discretized in time under the assumption of constant plant inputs during one sample step. In consequence, a discrete-time linear state space model in error coordinates is calculated in the form of

$$e_C[k+1] = A^{\circlearrowright} e_C[k] + B^{\circlearrowright} u_C[k] \tag{5.16}$$

for the forward as well as the heading state space model. Hereby, the 'circle arrow'-symbol on the right top of A and B indicates that these are the dynamic and input matrices of a discrete-time state space model and k is the sample step. The matrices for the forward dynamics are given in (B.1) and (B.2) as well as (B.3) and (B.4) for the heading dynamics. In consequence, the controllers are also designed in discrete-time.

Firstly, let us define the control error vectors for each submodel based on (3.122) with

$$e_{C,Fw} = \begin{pmatrix} e_d, & e_\alpha, & e_{v_d}, & e_{v_\alpha}, & e_{i_{Fw}} \end{pmatrix}^{\mathrm{T}} \text{ and } e_{C,He} = \begin{pmatrix} e_\theta, & e_{v_\theta}, & e_{i_{He}} \end{pmatrix}^{\mathrm{T}} . \tag{5.17}$$

Then, the control error e_C as depicted in Figure 5.1 can be calculated with

$$e_C = T_{c\text{-}d}^{-1} \begin{pmatrix} e_{C,Fw} \\ e_{C,He} \end{pmatrix}^{\mathrm{T}} . \tag{5.18}$$

using the state transformation matrix $T_{c\text{-}d}$ given in Appendix A.4. For closed-loop control, a feedback matrix K is required, such that the control law

$$u_C[k] = K e_C[k] \tag{5.19}$$

stabilizes the TWIP. To reduce the number of parameters, a LQRY is used for each of the submodels. Hereby, the output for the forward model is chosen to be flat and is defined by (3.125) and (3.130) and thus the output matrix is given by

$$C_{Fw} = \begin{pmatrix} 1, & 0.0644, & 0, & 0, & 0 \end{pmatrix} . \tag{5.20}$$

In addition, the output matrix for the heading model is chosen as

$$C_{He} = \begin{pmatrix} 1, & 0, & 0 \end{pmatrix} \tag{5.21}$$

and thus the output is $y_{He} = \theta$. Finally, let us define the cost function used in the LQRY design synthesis for the forward model with

$$J = \sum_{k=0}^{\infty} (e_{C,Fw}[k]^{\mathrm{T}} C_{Fw}^{\mathrm{T}} \boldsymbol{\mathcal{Q}}_{Fw} C_{Fw} e_{C,Fw}[k] + u_{Fw}[k]^{\mathrm{T}} \boldsymbol{\mathcal{R}}_{Fw} u_{Fw}[k]) \tag{5.22}$$

and for the heading model with

$$J = \sum_{k=0}^{\infty} (e_{C,He}[k]^{\mathrm{T}} C_{He}^{\mathrm{T}} \boldsymbol{\mathcal{Q}}_{He} C_{He} e_{C,He}[k] + u_{He}[k]^{\mathrm{T}} \boldsymbol{\mathcal{R}}_{He} u_{He}[k]) \ . \tag{5.23}$$

In consequence, the derived feedback gains K_{Fw} and K_{He} minimize the respective cost function. Using the input transformation T_u and the state transformation $T_{c\text{-}d}$ defined in (A.7) and (A.8) the feedback gain

$$K = T_u^{-1} \begin{pmatrix} K_{Fw} & 0_{1\times 3} \\ 0_{1\times 4} & K_{He} \end{pmatrix} T_{c\text{-}d} \tag{5.24}$$

can be calculated which is used in (5.19) and stabilizes (5.16). As each model only has one output, the weighting matrices reduce to scalar factors. They are tuned in simulations and verified in experiments with $\boldsymbol{\mathcal{Q}}_{Fw} = 3000$, $\boldsymbol{\mathcal{Q}}_{He} = 150$ for the outputs and $\boldsymbol{\mathcal{R}}_{Fw} = \boldsymbol{\mathcal{R}}_{He} = 1$ for the control inputs. The resulting matrix K is given in (B.5).

For sure, the same feedback gain can also be derived by assembling and transforming $\boldsymbol{\mathcal{Q}}_{Fw}$, $\boldsymbol{\mathcal{Q}}_{He}$ as well as $\boldsymbol{\mathcal{R}}_{Fw}$, $\boldsymbol{\mathcal{R}}_{He}$ and solving the LQR problem for the full (not decoupled) state space model.

The proposed design procedure eases the control synthesis, as only two scalars ($\boldsymbol{\mathcal{Q}}_{Fw}$ and $\boldsymbol{\mathcal{Q}}_{He}$) have to be tuned. In addition, the approach offers a clear relation between the parameters to tune and the effect on the error compensation related to the longitudinal and heading motions of the robot. Moreover, if one decides to neglect the current dynamics for closed-loop control and uses the model without currents for control synthesis, the procedure stays the same and the tuned parameters can be reused as they weight the output and not directly the states.

5.4.1 Experimental Results

The performance of the controller and guidance algorithm proposed has been evaluated in experiments with the optimal trajectory as presented in Chapter 4. To be precise, one loop of the trajectory with a duration of about 53 s as shown in Figures 4.1, 4.2, 4.4 and 4.5 is sliced out of the experimental data for analysis. In Tables 5.1 and 5.2 the maximum absolute errors and RMS values of all states are listed. In addition, Figures 5.3 and B.1[1] show the error plot of the states.

Table 5.1: Experimental control errors (x_{B}, y_{B}, θ, α)

state	x_{B}	y_{B}	θ	α
unit	m	m	°	°
$\max \lvert e_C \rvert$	$1.7 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$	$5.4 \cdot 10^{0}$	$2.3 \cdot 10^{0}$
$\mathrm{RMS}(e_C)$	$7.3 \cdot 10^{-3}$	$7.0 \cdot 10^{-3}$	$2.1 \cdot 10^{0}$	$6.2 \cdot 10^{-1}$

[1]Note that tiny error peaks due to disturbances and noise in the estimated state, especially at the currents i_{R} and i_{R}, are included in the calculation for the maximum absolute and RMS error values but cannot be seen in the plots, as they have to be drawn with down-sampled data.

Table 5.2: Experimental control errors (v_α, v_d, v_α, i_R, i_L)

state	v_α	v_d	v_α	i_R	i_L
unit	°/s	m/s	°/s	A	A
$\max\lvert e_C\rvert$	$4.9 \cdot 10^{-1}$	$4.9 \cdot 10^{-2}$	$8.8 \cdot 10^{-1}$	$4.9 \cdot 10^{-1}$	$4.9 \cdot 10^{-1}$
$\mathrm{RMS}(e_C)$	$1.1 \cdot 10^{-1}$	$1.5 \cdot 10^{-2}$	$1.4 \cdot 10^{-1}$	$10 \cdot 10^{-2}$	$9.4 \cdot 10^{-2}$

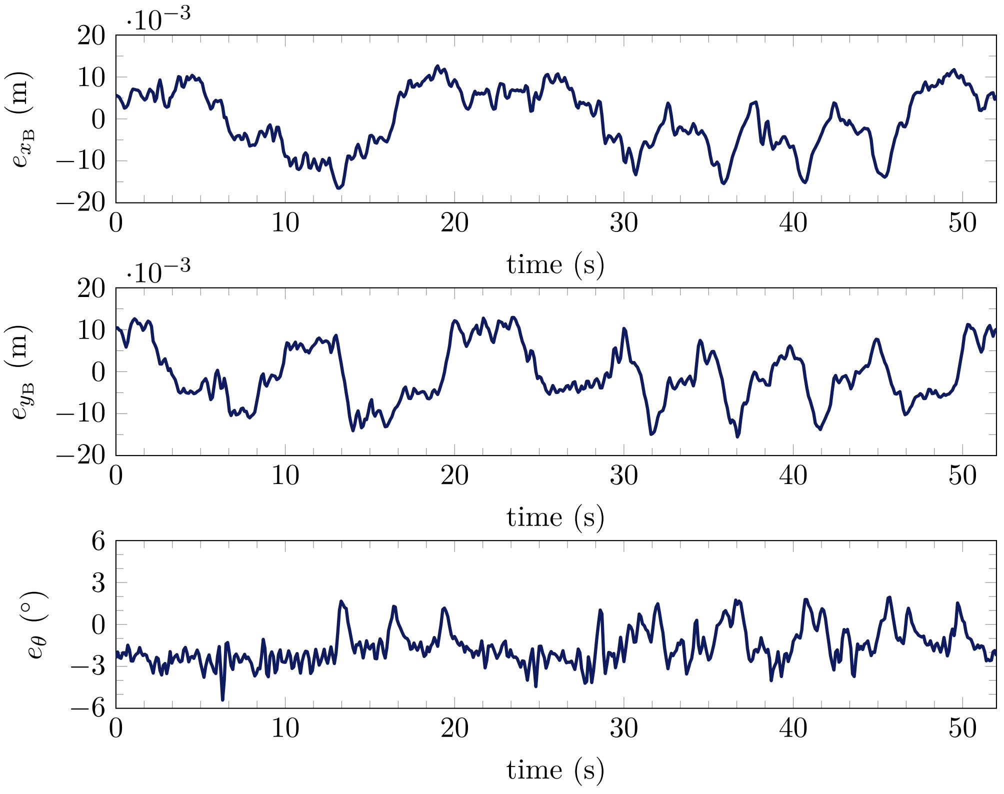


Figure 5.3: Experimental control errors (e_{x_B},e_{y_B},e_θ)

For stability, precise control of α and high disturbance rejection is required. This property is fulfilled as can be inferred by the values for α in Table 5.1. Moreover, the TWIP is able to follow the trajectory precisely with less than 2 cm maximum offset in position and less than 6° in orientation with the proposed algorithm. Note that the typical offsets are much smaller, as the RMS values show. Moreover, a controller tuned to be more aggressive is able to even further reduce the influence of disturbances but will lead to a smaller DoA due to the limited motor terminal voltages. Thus, the controller parameters were chosen to get a good compromise between error minimization as well as disturbance rejection and the resulting size of the DoA.

5.5 Estimation of the Domain of Attraction

In this section, the DoA is estimated by the use of a quadratic Lyapunov function (QLF)

$$V[k] = e_C[k]^T P e_C[k] \quad \text{with} \quad P = P^{\mathrm{T}} > 0 \;, \tag{5.25}$$

the linear state space model and the controller derived in the previous section. Hereby, the calculated estimated domain of attraction (EDoA) is limited by a defined level set $V(e_C[k]) = \eta_0$ and the state limits and input saturation of the systems are incorporated by the derivation of the matrix P.

The set of matrix inequalities defining the constraints on the QLF as well as an optimization problem to find the largest EDoA is introduced in the following. As they can be converted into LMIs as presented in Boyd et al. [10] the optimization problem can be solved via convex optimization efficiently as introduced by Boyd and Vandenberghe [9].

To ensure a minimum control performance inside the EDoA, a minimum decay rate of the Lyapunov function can be considered. Moreover, such a minimum decay rate is even required to ensure stability during trajectory tracking. The requirement for a minimum decay rate $\beta > 0$ can be formulated with the difference of the Lyapunov function between two consecutive sample steps by

$$\Delta V[k] = V[k+1] - V[k] \leq -\beta V[k] \text{ with } 0 < \beta \leq 1 \;. \tag{5.26}$$

In the next step, $e_C[k+1]$ can be replaced by the linear closed-loop system dynamic

$$e_C[k+1] = \underbrace{(A^{\circlearrowright} + B^{\circlearrowright} K)}_{A_{cl}^{\circlearrowright}} e_C[k] \tag{5.27}$$

in $V[k+1]$ and included in (5.26). Based on the resulting difference equation

$$\begin{aligned} \Delta V[k] &= e_C[k+1]^{\mathrm{T}} P e_C[k+1] - e_C[k]^{\mathrm{T}} P e_C[k] \\ &= e_C[k]^{\mathrm{T}} \left(A_{cl}^{\circlearrowright\,\mathrm{T}} P A_{cl}^{\circlearrowright} - P \right) e_C[k] \leq -\beta e_C[k]^T P e_C[k] \;. \end{aligned} \tag{5.28}$$

the matrix inequalities

$$P = P^{\mathrm{T}} > 0 \tag{5.29a}$$

$$A_{cl}^{\circlearrowright\,\mathrm{T}} P A_{cl}^{\circlearrowright} - (1-\beta) P \leq 0 \tag{5.29b}$$

can be formulated. In consequence, if a positive definite matrix P can be found such that (5.29) is fulfilled, a QLF for the closed-loop system has been found. As no constraints have been incorporated yet, the EDoA covers the full state space of the linear model.

Unfortunately, due to the state and input limits of the TWIP, the real DoA is limited. In consequence, these limits have to be included in the estimation of the DoA. This is possible, by adding additional LMIs to the problem such that the largest level set η_0 of the QLF is known, in which all trajectories starting with a value $V(e_C) \leq \eta_0$ will not violate the state and input limits. In consequence, the goal is to find an EDoA

$$\mathcal{S}_0 = \{\; e_C \mid V(e_C) \leq \eta_0 \} \tag{5.30}$$

as large as possible, with η_0 inside all introduced restrictions. This is nothing else than maximizing the size of an ellipsoid inside a polytope as shown in Boyd et al. [10]. Hu and Lin [46] as well as Hu et al. [47] introduced how state and input limits could be included in the optimization of P through the definition of a polytope. This approach has also been used by Diepold et al. [30], Diepold [31] and Anhalt [5]. Defining $\eta_0 = 1$ the conditions

$$k_i^{\mathrm{T}} P^{-1} k_i - u_{C,i,\eta_0}^2 \leq 0 \ , \tag{5.31a}$$
$$g_j^{\mathrm{T}} P^{-1} g_j - e_{C,j,\eta_0}^2 \leq 0 \tag{5.31b}$$

can be formulated to consider the limits. Hereby, the vector k_i is the i row of the constant feedback gain K (5.24) and enforces the input limit of the ith input. Analogously, the vector g_j accounts for the jth state limit and is defined as a vector with the size of the state vector, with all entries zero beside the jth entry. For the TWIP, the inputs u_{R} and u_{L} are limited as well as the states α, i_{R} and i_{L}. The values u_{C,i,η_0} and e_{C,j,η_0} are the smallest absolute values of the available control input or the acceptable error in the limited state which are inside the level set η_0.

The minimization of the determinant of P in respect to the conditions (5.29) and (5.31) is similar to the maximization of the volume of the ellipsoid of the QLF limited by the conditions on η_0 (see Boyd et al. [10], Hannah [42]). Thus, the complete optimization problem assembles to

$$\min \det P \tag{5.32a}$$
$$\text{s. t.} \quad P > 0 \tag{5.32b}$$
$$A_{cl}^{\circlearrowleft\,\mathrm{T}} P A_{cl}^{\circlearrowleft} - (1-\beta) P \leq 0 \ , \tag{5.32c}$$
$$k_i^{\mathrm{T}} P^{-1} k_i - u_{C,i,\eta_0}^2 \leq 0 \quad \forall i \in \{R, L\} \ , \tag{5.32d}$$
$$g_j^{\mathrm{T}} P^{-1} g_j - e_{C,j,\eta_0}^2 \leq 0 \quad \forall j \in \{\alpha, i_{\mathrm{R}}, i_{\mathrm{L}}\}. \tag{5.32e}$$

To solve this problem efficiently, the introduced inequality equations and cost function have to be converted to LMIs and a convex cost function. Using the substitution $Q = P^{-1}$ and applying the Schur complement on (5.32c) LMIs are received. Furthermore, if the logarithmic function is applied to (5.32a), the optimization problem

$$\min(-\log \det Q) \tag{5.33a}$$
$$\text{s. t.} \quad Q \geq 0 \ , \tag{5.33b}$$
$$\begin{pmatrix} (1-\beta)Q & Q A_{cl}^{\circlearrowleft\,\mathrm{T}} \\ A_{cl}^{\circlearrowleft} Q & Q \end{pmatrix} \geq 0 \ , \tag{5.33c}$$
$$k_i^{\mathrm{T}} Q k_i - u_{C,i,\eta_0}^2 \leq 0 \quad \forall i \in \{R, L\} \ , \tag{5.33d}$$
$$g_j^{\mathrm{T}} Q g_j - e_{C,j,\eta_0}^2 \leq 0 \quad \forall i \in \{\alpha, i_{\mathrm{R}}, i_{\mathrm{L}}\} \ . \tag{5.33e}$$

is convex and can be solved through convex optimization and a EDoA based on a QLF is derived. The introduced optimization problem is solved by the use of the toolbox YALMIP created by Löfberg [65] and with the solver SDPT3 provided by Toh et al. [104] and Tütüncü et al. [105]. As setpoint tracking requires different parameters compared to trajectory tracking, the numerical values of the parameters used in (5.33) and their choice are discussed in the corresponding sections of the tracking method.

5.6 Command Governor

In Section 5.4 the design of a linear control law is presented which stabilizes the TWIP. Based on this controller, a QLF is derived in Section 5.5 with a level set η_0, limiting the EDoA. In consequence, only if the control error e_C is inside the EDoA, stabilization can be guaranteed. As motivated in Section 5.1 this might not always be the case even during 'normal' operation.

To extend the EDoA, Buhl [12], Buhl and Lohmann [13], Diepold et al. [30] and Diepold [31] presented algorithms for setpoint and trajectory tracking. In particular, Buhl [12] and Buhl and Lohmann [13] presented an algorithm for setpoint tracking with SISO-systems and Diepold et al. [30] and Diepold [31] extended the algorithm for MIMO-systems and developed additional requirements to perform trajectory tracking. Finally, Anhalt [5] and the author adapted the concepts to be used with discrete-time systems. In consequence, the first presentation of the discrete-time variants of the above-mentioned algorithms as well as the application on the TWIP can be found in Anhalt [5], supervised by the author.

Buhl [12], Buhl and Lohmann [13] and Diepold [31] used a control structure for setpoint tracking, where the command governor modifies the desired state before the calculation of the control error. However, in this thesis, the command governors are presented directly in error coordinates for both, setpoint tracking as well as trajectory tracking. For a more detailed derivation and thorough theoretical treatment it is referred to Buhl [12], Buhl and Lohmann [13] for the SISO setpoint tracking case and to Diepold et al. [30], Diepold [31] for the MIMO and trajectory tracking case.

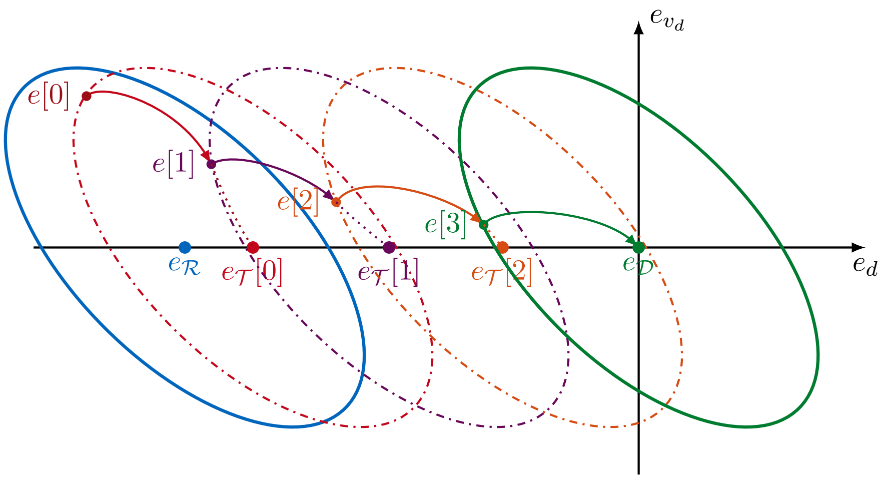


Figure 5.4: Illustration of the command governor principle

In Figure 5.4 a projection of a EDoA to a 2D-plane with the distance error e_d and forward velocity error e_{v_d} for the TWIP is illustrated. Herein, the green solid line marks the limiting level set η_0 of the EDoA in respect to the desired equilibrium $e_{\mathcal{D}} = 0$ at the origin. The control error e_C, as shown in Figure 5.1, is calculated by

$$e_C = e - e_{\mathcal{T}} \,, \tag{5.34}$$

where $e_{\mathcal{T}}$ is a temporary equilibrium set by the command governor algorithm. Let us first assume, that e is inside the EDoA and thus $V(e) \leq \eta_0$. Then, the command governor will set $e_{\mathcal{T}} = e_{\mathcal{D}} = 0$ and due to the asymptotically stabilizing controller, the control error $e_C = e - e_{\mathcal{T}} = e - e_{\mathcal{D}} = e$ should converge to zero over time.

However, a couple of issues have been discussed in Section 5.1, causing a large control error outside the EDoA. For an initial time step $k = 0$ such a large error $e[0]$ is plotted in Figure 5.4, which is clearly outside the EDoA with the origin $e_{\mathcal{D}}$. But, if a stationary offset in the distance d is accepted, another equilibrium $e_{\mathcal{R}}$ ('R' for reference) may be found, in which EDoA includes $e[0]$, as shown in Figure 5.4 with the blue limiting level set. In consequence, if the control error is calculated by $e_C = e - e_{\mathcal{T}} = e - e_{\mathcal{R}}$ the control error is inside the EDoA of $e_{\mathcal{R}}$ and thus stabilization is ensured but with a stationary offset in d. In particular, e_C will converge to zero and thus e to $e_{\mathcal{T}} = e_{\mathcal{R}}$ over time. However, the goal has to be that e converges to zero and thus $e_{\mathcal{T}}$ is set to $e_{\mathcal{D}}$.

In consequence, the idea is to look for a temporary equilibrium $e_{\mathcal{T}}$, which lies between $e_{\mathcal{R}}$ and $e_{\mathcal{D}}$ with e inside the EDoA, but as close as possible to $e_{\mathcal{D}}$. With discrete-time control, the derivation of $e_{\mathcal{T}}$ can be repeated in every sample step and in consequence, the control error will be guided by $e_{\mathcal{T}}$, set by the command governor, to $e_{\mathcal{D}}$ over time. Thereby, stabilization can be guaranteed as long as a reference equilibrium with $V(e - e_{\mathcal{R}}) \leq \eta_0$ can be found.

Such a shift of $e_{\mathcal{T}}$ by the command governor is visualized in Figure 5.4. At first, a reference equilibrium $e_{\mathcal{R}}$ is calculated to ensure, that $e[0]$ is inside any EDoA at all. Afterwards, a temporary equilibrium $e_{\mathcal{T}}[0]$ is calculated and used to calculate $e_C[0] = e[0] - e_{\mathcal{T}}[0]$. Due to the asymptotically stabilizing control law, designed in Section 5.4, the error will converge slightly towards $e_{\mathcal{T}}[0]$ during the sample step. In the next sample step $k = 1$, a new temporary equilibrium $e_{\mathcal{T}}[1]$ can be derived which is closer to $e_{\mathcal{D}}$ and can be used to calculate $e_C[1]$. As shown in Figure 5.4, this procedure is repeated until e is inside the EDoA of the origin $e_{\mathcal{D}}$.

Let us now introduce the equations to derive the temporary equilibrium $e_{\mathcal{T}}$. At first, the reference equilibrium has to be found. As linear dynamics are considered for control synthesis and analysis, the QLF found for $e_{\mathcal{D}}$ is also valid for all other equilibria and thus all points share the same Lyapunov matrix P. Buhl [12] and Buhl and Lohmann [13] noted for linear systems, that the reference points lie in the nullspace N of the continuous-time dynamic matrix A, if the system has eigenvalues in the origin. As the TWIP has two eigenvalues in zero, we calculate

$$e_{\mathcal{R}} = Nw \quad \text{with} \quad N = \begin{pmatrix} 1 & 0 & 0_{1\times 6} \\ 0 & 1 & 0_{1\times 6} \end{pmatrix}^{\mathrm{T}}, \tag{5.35}$$

which defines the reference equilibrium via the vector $w = [e_d, e_\theta]^{\mathrm{T}}$. In the next step, let us derive w, such that the corresponding value of the QLF of $e_{\mathcal{R}}$ is as small as possible. In consequence, we formulate the minimization problem

$$e_{\mathcal{R}} = \underset{e_{\mathcal{R}}}{\arg\min}\, V(e - e_{\mathcal{R}}) = \underset{e_{\mathcal{R}}}{\arg\min} \Big((e - e_{\mathcal{R}})^{\mathrm{T}} P (e - e_{\mathcal{R}}) \Big) \tag{5.36}$$

to find the desired $e_{\mathcal{R}}$. As $e_{\mathcal{R}}$ is defined in (5.35) as a function of w, we include this in

(5.36) and get

$$w = \arg\min_{w} V(e - Nw) \tag{5.37}$$

$$= \arg\min_{w} \Big(e^{\mathrm{T}} Pe - e^{\mathrm{T}} PNw - (Nw)^{\mathrm{T}} Pe + (Nw)^{\mathrm{T}} P(Nw) \Big) \tag{5.38}$$

$$= \arg\min_{w} \underbrace{\Big(e^{\mathrm{T}} Pe + w^{\mathrm{T}} N^{\mathrm{T}} PNw - 2w^{\mathrm{T}} N^{\mathrm{T}} Pe \Big)}_{f_w} \tag{5.39}$$

which has to be solved for w. The extrema of the function f_w are at the zeros of its derivative

$$\frac{\partial f_w}{\partial w} = 2N^{\mathrm{T}} PNw - 2N^{\mathrm{T}} Pe = 0 \; . \tag{5.40}$$

In our case, only a single solution for the minimum with

$$w = (N^{\mathrm{T}} PN)^{-1} N^{\mathrm{T}} Pe \tag{5.41}$$

is derived, depending on e. By including the solution of w into (5.35) the reference equilibrium is given by

$$e_{\mathcal{R}} = Nw = \underbrace{N(N^{\mathrm{T}} PN)^{-1} N^{\mathrm{T}} P}_{L} \, e \; , \tag{5.42}$$

which is the equilibrium with the smallest value of the QLF $V(e)$. As the matrix L can be calculated offline in advance it does not have to be computed online at each sample step to derive $e_{\mathcal{R}}$.

In the following, the derivation of a temporary equilibrium is introduced. If e is already inside the EDoA of the origin $V(e - e_{\mathcal{D}}) = V(e) \leq \eta_0$ then the intuitive choice is to set $e_{\mathcal{T}} = e_{\mathcal{D}} = 0$. In all other cases let us look for a temporary equilibrium point between $e_{\mathcal{R}}$ and $e_{\mathcal{D}}$, defined by

$$e_{\mathcal{T}} = e_{\mathcal{R}} + c\,(e_{\mathcal{D}} - e_{\mathcal{R}}) \qquad c \in [0, 1) \tag{5.43}$$

with c as large as possible, to derive $e_{\mathcal{T}}$ to be as close as possible to $e_{\mathcal{D}}$. Consequently, the desired $e_{\mathcal{T}}$ leads to $V(e - e_{\mathcal{T}}) = \eta_0$ and thus to

$$\left(e - e_{\mathcal{R}} - c(e_{\mathcal{D}} - e_{\mathcal{R}})\right)^{\mathrm{T}} P \left(e - e_{\mathcal{R}} - c(e_{\mathcal{D}} - e_{\mathcal{R}})\right) = \eta_0 \; . \tag{5.44}$$

This condition can be rearranged to

$$c^2 \Big((e_{\mathcal{R}} - e_{\mathcal{D}})^{\mathrm{T}} P (e_{\mathcal{R}} - e_{\mathcal{D}}) \Big) + c \Big(2(e_{\mathcal{R}} - e_{\mathcal{D}})^{\mathrm{T}} P (e - e_{\mathcal{R}}) \Big) + \Big((e - e_{\mathcal{R}})^{\mathrm{T}} P (e - e_{\mathcal{R}}) \Big) = \eta_0 \tag{5.45}$$

and is a scalar quadratic equation in c. Due to this, c can be calculated by solving the quadratic equation and the selection of the larger value of c in $[0, 1)$. Finally, we are able to calculate $e_{\mathcal{T}}$ with (5.43) and the calculated value of c such that e_C is inside the EDoA.

For visualization purposes, a fictive temporary value for the desired distance and heading angle can be calculated by

$$\mathring{d}_{\mathcal{T}} = \hat{d} + e_{C,d} = \hat{d} + (e_d - e_{\mathcal{T},d}) \tag{5.46}$$

and

$$\mathring{\theta}_{\mathcal{T}} = \hat{\theta} + e_{C,\theta} = \hat{\theta} + (e_{\theta} - e_{\mathcal{T},\theta}) \, , \tag{5.47}$$

with $e_{\mathcal{T}}$ and e. These two temporary values will be used to visualize the experimental results in the later sections of this chapter.

Even though the EDoA typically underestimates the DoA drastically, it is not a piece of good advice to choose a $e_{\mathcal{T}}$ such that e is exactly at the border of the EDoA with $V(e) = \eta_0$. During experiments, measurement noise might increase the error of the state estimation and also model uncertainties have to be considered. Thus, the command governors are typically parameterized to calculate $e_{\mathcal{T}}$, such that e is on a smaller, safer level set $V(e) = \eta_S = k_S \eta_0$ with $0 > k_S \geq 1$.

Last but not least, there might be the case that e leads to a value $\eta_0 > V(e) > \eta_S$ and no $e_{\mathcal{T}}$ can be calculated but a stable $e_{\mathcal{R}}$ can be found. Even worse, unexpected large disturbances might lead to $V(e) > \eta_0$, which means e is not inside any EDoA. As a fallback solution, $e_{\mathcal{R}}$ is calculated by (5.42) and used in these cases as $e_{\mathcal{T}}$, even though the value of V might be even larger than η_0 and stabilization cannot be guaranteed. The case $V(e) > \eta_0$ should never occur during operation, but it is implemented to at least attempt to stabilize the system if it occurs. On the other side, the case $\eta_0 > V(e) > \eta_S$ occurs frequently since $e_{\mathcal{T}}$ is set to be on the level set η_S and the estimated states often slightly differ from the real state values due to measurement noise.

The charm of the presented procedure is that almost no computational power is required to compute $e_{\mathcal{R}}$ and to solve the quadratic equation (5.45). Thus the derivation of $e_{\mathcal{T}}$ is incredibly cheap and can be calculated on literally every MCU in real-time.

At the end, a remark to a small modification to the introduced command governor method is given to improve the performance: Experiments conducted by Anhalt [5] have shown that the command governor leads to S-curved trajectories, even if only an initial error in distance is considered. This is caused by an interaction of the stable heading dynamics as introduced in Subsection 3.3.2, the degree of freedom in (5.42) in θ as well as the guidance algorithm to handle the nonholonomic constraints of the TWIP. In consequence, to force the TWIP to a more forward-directed motion, the degree of freedom in (5.35) for θ is removed by default and only if no $e_{\mathcal{R}}$ could be found which leads to an e_C inside the EDoA, the introduced equation (5.42) with two degrees of freedom is used.

In the next section, the setpoint tracking control algorithm will be presented, using the command governor introduced, and its applicability is shown by experiments.

5.7 Setpoint Tracking Control

To change the robot's position and orientation a setpoint tracking control algorithm is required, which ensures a stable transmission from the initial point to the new desired setpoint. Such setpoint changes may be commanded by a simple upper-level navigation scheme, leading the robot with waypoints through a building or logistic center. To ensure stable operation, the command governor presented in Section 5.6 is used and thus a QLF that estimates the DoA is needed.

Table 5.3: Parameters to estimate the DoA for setpoint tracking

Parameter	Value
$u_{C,R,\eta 0}$	$7.14\,\mathrm{V}$
$u_{C,L,\eta 0}$	$7.14\,\mathrm{V}$
$e_{C,\alpha,\eta 0}$	$\pm\frac{1}{6}\pi\,\mathrm{rad}$
$e_{C,i_\mathrm{R},\eta 0}$	$2.82\,\mathrm{A}$
$e_{C,i_\mathrm{L},\eta 0}$	$2.82\,\mathrm{A}$
β	$4 \cdot 10^{-3}$
k_S	$\frac{2}{3}$

5.7.1 Parameters

The parameters used to calculate the EDoA, as presented in Section 5.5, are listed in Table 5.3 and introduced in the following.

Firstly, the available input for control has to be determined. The friction compensation, presented in Section 5.2 requires the input u_F and depends on the robot's state. In consequence, the maximum input which is available for control is calculated by $u_{C,R,\eta 0} = u_{C,L,\eta 0} = u_\mathrm{bat} - |u_{F,max}|$, which is the difference between the available battery voltage and the maximum value of the input demanded by the friction compensation. As only the nonlinear friction components are compensated, the amount of input that has to be statically allocated to the friction compensation is drastically reduced. This leads to a larger available input $u_{C,R,\eta 0}$ and $u_{C,L,\eta 0}$ for control, which can be considered in finding the EDoA. Moreover, the current limit has only to be reduced from $\pm 3\,\mathrm{A}$ to $\pm 2.82\,\mathrm{A}$ due to the friction compensation.

Furthermore, to force all trajectories to stay inside the region, where the linear model shows a high congruence with the real nonlinear system, the physical tilt angle limit of $\pm\frac{1}{2}\pi\,\mathrm{rad}$ is reduced to $\pm\frac{1}{6}\pi\,\mathrm{rad}$. Due to this, a setpoint change will only show small tilt angles.

For robustness, the command governor is parameterized with $k_S = \frac{2}{3}$.

Last but not least, to ensure a minimum performance on the EDoA, a minimum decay rate of $\beta = 4 \cdot 10^{-3}$ is considered.

5.7.2 Experimental Results

The proposed control structure has been tested experimentally and the results are presented in the following. In the experiment, a setpoint change of $1.5\,\mathrm{m}$ in x and y has been commanded. This leads to an error e_d in d of $\frac{3}{\sqrt{2}}\,\mathrm{m}$ which is outside the DoA and thus a stable setpoint change will fail without the use of the command governor.

Experimental results using the proposed command governor are shown in Figure 5.5 and Figure 5.6. In Figure 5.5, the dotted blue lines are the desired values given and the setpoint changes for x and y at $t = 3.5\,\mathrm{s}$ can be recognized. The setpoint change requires

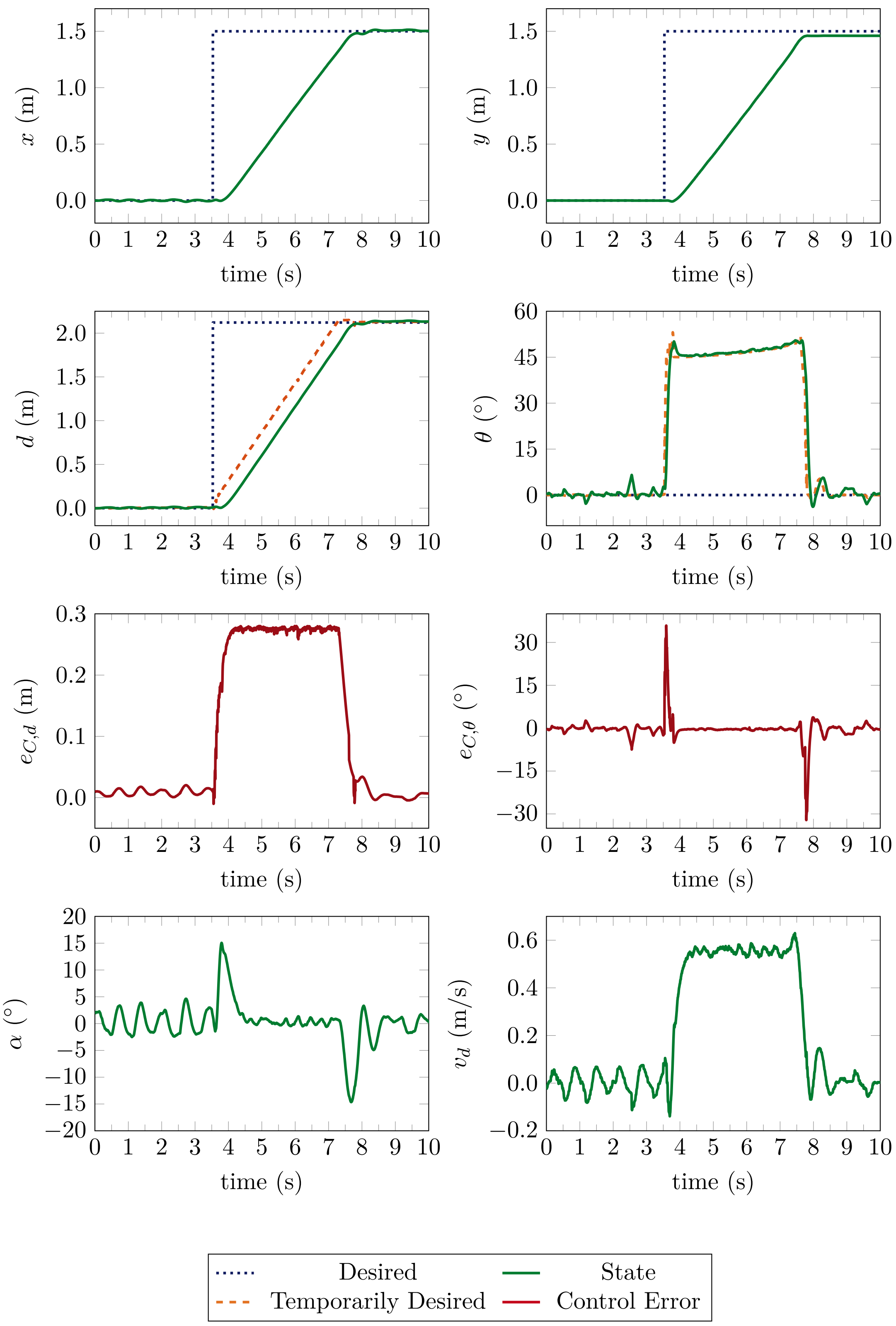


Figure 5.5: Experimental results setpoint tracking

about 4.5 s where the forward speed is about 0.5 m/s. In addition, the non-minimum phase behavior of the system can be observed in the plots, as the system response first

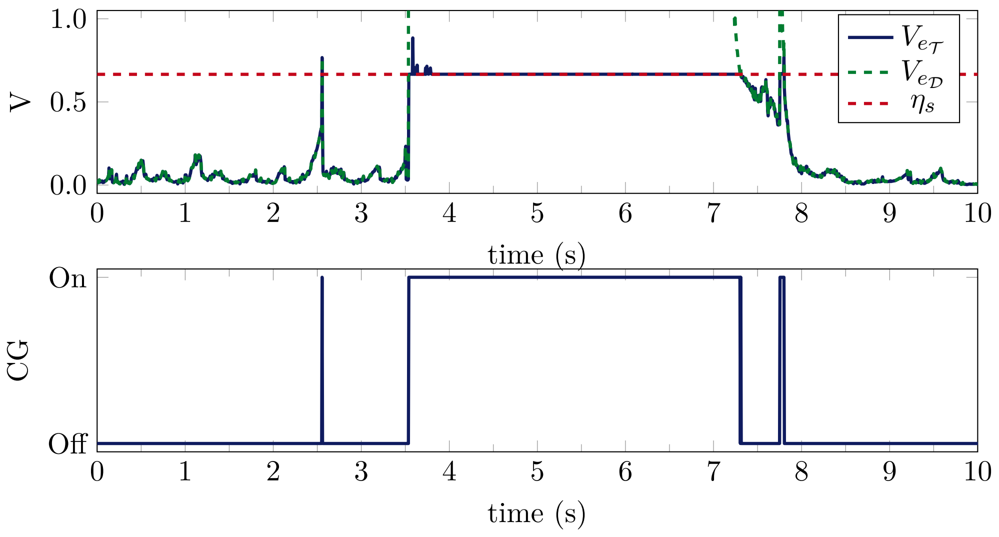


Figure 5.6: Experimental results command governor setpoint tracking

goes into the reverse directions for x, y, d and v_d. Also, the robot tilts up to 15° in α to accelerate and decelerate at the beginning and the end of the maneuver. Moreover, the temporary values for the desired distance d and heading angle θ that are calculated by the command governor are shown in the plots of d and θ by dashed orange lines. Hereby, using temporarily desired values, the control error e_C is limited to be inside the DoA as can be seen in the plots of $e_{C,d}$ and $e_{C,\theta}$

To perform the setpoint change, the orientation of the robot has to be modified from its final desired value, as can be seen in the plot of θ. This is done by the guidance algorithm by manipulating the desired orientation of θ, such that the robot drives to the new setpoint and then recovers the desired heading. In particular, the reference value for θ is changed from 0° to 45° at the beginning of the setpoint change and then back to the desired value of 0° as the new position is approached.

The upper plot of Figure 5.6 shows $V_{e_\mathcal{D}}$, which is the value of the QLF evaluated with the control error e_C using the equilibrium $e_\mathcal{D}$ of the new desired setpoint. In addition, $V_{e_\mathcal{T}}$ is drawn, which is the value of the QLF calculated with e_C using the temporary equilibrium $e_\mathcal{T}$. Also, the limiting level set η_S is drawn, marking the values for a safe command governor operation, including a safety margin to $\eta_0 = 1$. In addition, the lower plot of Figure 5.6 shows where the command governor is active and manipulates the value of e_C by $e_\mathcal{T} \neq 0$.

At $t = 2.6\,\text{s}$, as well as $t = 7.7\,\text{s}$, short disturbances push the value of the QLF above the limit of η_S and the command governor is activated for a few milliseconds to ensure safe operation. As the value of $V_{e_\mathcal{D}}$ exceeds η_S as well as η_0 at the beginning of the setpoint change at $t = 3.5\,\text{s}$, the command governor is activated as expected. During the active phase of the command governor, the value of $V_{e_\mathcal{T}}$ stays equal to η_S as $e_\mathcal{T}$ is recalculated every sample step to fulfill $V_{e_\mathcal{T}} = \eta_S$ in order to reach $e_\mathcal{T} = e_\mathcal{D}$ as quickly as possible. As soon as $V_{e_\mathcal{D}}$ is below η_S, the command governor is deactivated and $e_\mathcal{T}$ is set to $e_\mathcal{T} = e_\mathcal{D} = 0$. This can be observed in the plots at $t = 7.3\,\text{s}$.

The experimental results show that the proposed control structure for setpoint tracking is able to provide a stable transition of the robot to a new setpoint, even if the new setpoint is far outside the DoA of the controller. Moreover, the applicability and robustness of the proposed and implemented algorithms are verified.

5.8 Trajectory Tracking Control

Besides setpoint tracking, modern applications for mobile robots also require accurate trajectory tracking. For example, this is required if several robots operate in the same area of a logistic center and a collision has to be avoided, even if their paths are crossing. Moreover, if a robot is used for surveillance applications, trajectory tracking is required to accurately follow the desired target in motion.

The major difference between setpoint tracking and trajectory tracking is that the limits for the control input u_C and error e_C depend on the feedforward values of the trajectory $\mathring{u}$ and $\mathring{x}$. In particular, the plant input $u = u_C + u_F + \mathring{u}$, shown in Figure 5.1, has to be inside the physical limits of the TWIP and thus the condition $\|u\|_\infty \leq u_{max}$ always has to be fulfilled[2]. The same principles apply for the state limits: the desired trajectories superposed with possible errors, have always to be inside the state limits of the system. Let us assume a desired value $\mathring{i}_\mathrm{R}[k^*] = 1\,\mathrm{A}$ at the sample step k^* and a state limit of $i_{R,max} = 3\,\mathrm{A}$, for example. In this case, the value of the control error e_{C,i_R} has to be $\leq 2\,\mathrm{A}$. In consequence, there are two possibilities to estimate the DoA.

The first option is an a priori *static allocation of input and state limits (SAL)*. In this case, the limits for u_C and e_C are calculated in advance based on the maximum values of the desired trajectory. In consequence, similar to the setpoint tracking application, a constant limiting level set $\eta_L = \eta_0$ of the calculated QLF is received, which can be used by the command governor. The operation principle is then similar to the introduced setpoint tracking. The disadvantage of this static input and state allocation is, that if the desired input and states are close to the system's limits for a single moment, the input and state limits on u_C and e_C might get tight and the EDoA drastically shrinks.

Contrary to the static allocation, a *dynamic allocation of input and state limits (DAL)* can be considered. In this case, the control input and control error limits are calculated based on the values $\mathring{u}$ and $\mathring{x}$ of the desired trajectory in each sample step k, which leads to a changing limiting level set $\eta_L(\mathring{x}[k], \mathring{u}[k], \eta_0)$ of the QLF in every sample step. In consequence, additional requirements on the desired trajectory and the Lyapunov function have to be fulfilled to ensure stability. The EDoA based on the dynamic allocation may change over time, as the limiting level set depends on the desired input and state of the trajectory. Moreover, it can be larger but, due to the additional requirements mentioned above, also smaller than with the static allocation.

To use the benefits of both approaches, the EDoA is based on SAL and DAL and in each sample step the approach is selected which guarantees stability and reduces the control error faster.

In the following, the static allocation as well as the dynamic allocation with additional conditions for the desired trajectory and the Lyapunov function are introduced. Afterwards, an algorithm to choose the better approach (SAL or DAL) at each sample step is introduced. Finally, experimental results are presented and discussed.

[2]Diepold [31] presented an approach with additional conditions inside the LMIs to estimate the DoA to guarantee stability even if the control limits are violated (also called oversaturation).

5.8.1 Static Allocation of Input and State Limits

If the static allocation of input and state limits (SAL) between the desired trajectory and the controller is chosen, the maximum values of the entire desired trajectory have to be used to calculate the effective input and state limits for the controller. This is contrary to the setpoint tracking application, where $\mathring{u} = 0$ and $\mathring{x} = 0$.

For the trajectory, introduced in Chapter 4, the maximum feedforward values are 4.28 V for the input, 0.37 A for the motor currents and 0.05 rad for the tilt angle. The maximum values allocated to the friction compensation are similar to the setpoint tracking application with 0.26 V for the input and 0.18 A for the motor currents.

To obtain the available input for the controller, the maximum input which may be allocated to the friction compensation as well as the feedforward input of the desired trajectory has to be subtracted from the available battery voltage. In consequence, the input limits considered to estimate the limiting level set η_0 of the DoA have to be calculated with

$$u_{C,i,\eta_0} = u_{\text{bat}} - |u_{F,i,max}| - \max_k |\mathring{u}_i[k]| \quad \forall i \in \{R, L\} \; . \tag{5.48}$$

In addition, the limit for the current errors are given by

$$e_{C,i,\eta_0} = x_{i,\text{lim}} - |x_{F,i,max}| - \max_k |\mathring{x}_i[k]| \quad \forall i \in \{i_\text{R}, i_\text{L}\} \tag{5.49}$$

and the error limit for the tilt angle is defined by

$$e_{C,i,\eta_0} = x_{i,\text{lim}} - \max_k |\mathring{x}_i[k]| \quad \forall i \in \{\alpha\} \; . \tag{5.50}$$

Since no state limits have been defined for the other states, they are not considered here. In accordance to the setpoint tracking application, the same decay rate constraint is used for the estimation of the DoA.

Finally, all limits and parameters used to find the EDoA as introduced in Section 5.5 with a static allocation of the input and state limits are listed in Table 5.4.

Table 5.4: Parameters to estimate the DoA for trajectory tracking

Parameter	Static	Dynamic
u_{C,R,η_0}	2.85 V	7.14 V
u_{C,L,η_0}	2.85 V	7.14 V
e_{C,α,η_0}	0.34 rad	0.39 rad
e_{C,i_R,η_0}	2.45 A	2.82 A
e_{C,i_L,η_0}	2.45 A	2.82 A
β	$4 \cdot 10^{-3}$	$6.6 \cdot 10^{-3}$
k_S	$\frac{1}{2}$	$\frac{1}{2}$

5.8.2 Dynamical Allocation of Input and State Limits

Contrary to a static allocation, dynamic allocation of input and state limits (DAL) is only possible if an additional condition linking the desired trajectory and the decay rate of the Lyapunov function is fulfilled. In this case, at each sample step the controller can use all resources which are currently not allocated to the desired trajectory and friction compensation. In consequence, the limiting level set may change at every sample step and thus the EDoA may increase or shrink.

Thus, in the following, the calculation of the limiting level set with DAL is introduced and the required minimum decay rate for the Lyapunov function is derived. More precisely, the required decay rate can be calculated based on a given desired trajectory or vice versa, based on a given decay rate, the maximum rate constraints for the input and states of the desired trajectory can be calculated. Finally, additional information is given on how a ‘duplicate’ allocation of input and state resources can be avoided if a friction compensation is used together with a desired trajectory.

Now, let us consider a dynamical allocation of the state and input limits available for the controller to stabilize the system. In this case, the limiting level set η_L of the EDoA changes at each sample step, depending on the desired trajectory with $\mathring{u}[k]$ and $\mathring{x}[k]$. Firstly, the available input for the controller is defined with

$$u_{C,i,\mathrm{avail}}[k] = \underbrace{u_{\mathrm{bat}} - |u_{F,i,max}|}_{u_{i,\eta_0}} - |\mathring{u}_i[k]| \quad \forall i \in \{R, L\} \;, \tag{5.51}$$

the admissible state error for the currents, limiting the control error trajectories are defined as

$$e_{C,i,\mathrm{avail}}[k] = \underbrace{x_{i,lim} - |x_{F,i,\mathrm{max}}|}_{e_{C,i,\eta_0}} - |\mathring{x}_i[k]| \quad \forall i \in \{i_{\mathrm{R}}, i_{\mathrm{L}}\} \tag{5.52}$$

and for the tilt angle control error with

$$e_{C,i,\mathrm{avail}}[k] = \underbrace{x_{i,\mathrm{lim}}}_{e_{C,i,\eta_0}} - |\mathring{x}_i[k]| = \quad \forall i \in \{\alpha\} \tag{5.53}$$

for every sample step k. Contrary to the static allocation, in (5.51), (5.52) and (5.53) the values at the current sample step k of the desired trajectory are considered instead of their maximum over all sample steps. In consequence, the available control input and control error changes for every sample step k as $\mathring{u}[k]$ and $\mathring{x}_i[k]$ of the desired trajectory change. Thus, the EDoA and the limiting level set η_L will also change. Similar to Subsection 5.8.1, the DoA can be estimated as presented in Section 5.5. Here, the limits u_{C,i,η_0} and e_{C,i,η_0} as labeled in (5.51), (5.52) and (5.53) with the values listed in Table 5.4 are used. As a result, the EDoA is calculated for $\mathring{u}[k] = 0$ and $\mathring{x}_i[k] = 0$ with the limiting level set η_0. In presence of $\mathring{u}[k] \neq 0$ or $\mathring{x}_i[k] \neq 0$, the limiting level set is reduced by the squared ratio of the available limits $u_{C,i,\mathrm{avail}}$, $e_{C,i,\mathrm{avail}}$ and the limits u_{C,i,η_0}, e_{C,i,η_0}, used to calculate the QLF with η_0. In particular, the limiting level for the sample step k can be calculated set based on the desired control input by

$$\eta_{L,u}[k] = \eta_0 \min_i \left(\frac{(u_{C,i,\eta_0} - |\mathring{u}_i[k]|)^2}{u^2_{C,i,\eta_0}} \right) \quad \forall i \in \{R, L\} \tag{5.54}$$

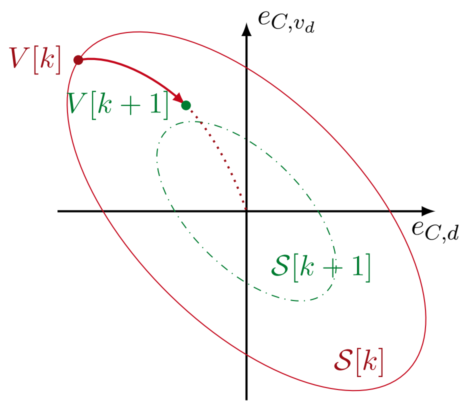


Figure 5.7: Reduction of the EDoA due to changed state and input limits

and based on the desired states by

$$\eta_{L,x}[k] = \eta_0 \min_i \left(\frac{(e_{C,i,\eta_0} - |\mathring{x}_i[k]|)^2}{e^2_{C,i,\eta_0}} \right) \quad \forall i \in \{\alpha, i_{\mathrm{R}}, i_{\mathrm{L}}\} \ . \tag{5.55}$$

As both, the control input and the state error limits have to be fulfilled, the effective limiting level set is given by

$$\eta_L[k] = \min\left(\eta_{L,u}[k], \eta_{L,x}[k]\right) \ . \tag{5.56}$$

Based on the desired trajectory, increasing $\mathring{u}$ or $\mathring{x}$ from one sample step to the next can lower the limiting level set $\eta_L[k]$ and the EDoA might shrink. To ensure stability, the value of the Lyapunov function $V(e_C)$ needs to decrease faster than the limiting level set $\eta_L[k]$ shrinks the EDoA. This is visualized in Figure 5.7, where the EDoA for sample steps k and $k+1$ are shown. Here the EDoA is labeled with $\mathcal{S}[k]$ and the ellipsoid drawn is the corresponding limiting level set $\eta_L[k]$ of the QLF. Let us assume that increasing values of $\mathring{u}$ and $\mathring{x}$ cause a reduction of the limiting level set from k to $k+1$. In the case shown, $e_C[k]$ is inside the EDoA but the value of V does not decrease the same amount as η_L does from sample step k to $k+1$. In consequence, $e_C[k+1]$ is outside the EDoA $\mathcal{S}[k+1]$, while $e_C[k]$ has been inside $\mathcal{S}[k]$. Clearly, this should never happen and thus, to use a dynamical allocation of the input and state limits, a minimal decay rate constraint to the Lyapunov function is required. This minimal decay rate constraint ensures that an increasing allocation to the desired trajectory never reduces the limiting level set faster than the Lyapunov function decreases. Luckily, we are able to enforce such a minimal decay rate of the QLF in Section 5.5 and moreover, the required decay rate can be calculated in advance based on the desired trajectory, as is introduced in the following.

Let us first change the expressions of the limiting ratio for the control input in (5.54) to

$$\xi_{u_C} = \left(1 - \underbrace{\max_i \left| \frac{\mathring{u}_i[k]}{u_{C,i,\eta_0}} \right|}_{\nu_{u,i}[k]} \right) \quad \forall i \in \{R, L\} \tag{5.57}$$

as well as the ratio for the states in error coordinates in (5.55) to

$$\xi_{e_C} = \left(1 - \underbrace{\max_i \left| \frac{\mathring{x}_i[k]}{e_{C,i,\eta_0}} \right|}_{\nu_{x,i}[k]} \right) \quad \forall i \in \{\alpha, i_\mathrm{R}, i_\mathrm{L}\} \tag{5.58}$$

such that $\nu_{u,i}[k]$ and $\nu_{x,i}[k]$ are scaled values between 0 and 1. As result, a value of $\nu = 1$ indicates that the full amount of the corresponding quantity is allocated to the desired trajectory and nothing is left for the controller and vise versa, $\nu = 0$ indicates that the full amount of the quantity can be used by the controller. Thus, the largest value

$$\nu[k] = \max\left\{\nu_{u,i}[k], \nu_{x,j}[k]\right\} \quad \forall i \in \{R, L\},\ \forall j \in \{\alpha, i_\mathrm{R}, i_\mathrm{L}\} \tag{5.59}$$

of all $\nu_{u,i}[k]$ and $\nu_{x,i}[k]$ is the one which will lead to the smallest level set η_L. In consequence, the level set limiting the EDoA can be calculated with

$$\eta_L[k] = \eta_0(1 - \nu[k])^2 \ . \tag{5.60}$$

The most critical situation occurs, if the value of the Lyapunov function for the control error is equal to the limiting level set

$$V(e_C[k]) = \eta_L[k] \tag{5.61}$$

and thus 'on the border' of the estimated stable region.

To guarantee stability in this situation, the difference $\Delta V[k] = V[k+1] - V[k]$ of the Lyapunov function has to decrease faster than the difference $\Delta\eta_L = \eta_L[k+1] - \eta_L[k]$ of the limiting level set, which leads to the condition

$$\Delta V[k] \leq \Delta\eta[k] \ . \tag{5.62}$$

Using the Lyapunov candidate function (5.25) with the minimal decay rate β, this condition is fulfilled if and only if

$$-\beta V[k] \leq \Delta\eta[k] \tag{5.63}$$

holds.

Let us now derive the connection between the minimal decay rate β and the rate

$$\Delta\nu[k] = \nu[k+1] - \nu[k] \tag{5.64}$$

calculated with (5.57), (5.58) and (5.59) using $\mathring{u}$ and $\mathring{x}$ from the desired trajectory.

Substituting (5.61) in (5.63) gives us the condition

$$-\beta\eta_L[k] \leq \Delta\eta[k] \tag{5.65}$$

where the rate of the limiting level set is given by

$$\begin{aligned} \Delta\eta_L[k] &= \eta_L[k+1] - \eta_L[k] \\ &= \eta_0 \left[(1 - \nu[k+1])^2 - (1 - \nu[k])^2 \right] \ . \end{aligned} \tag{5.66}$$

Afterwards, $\nu[k+1]$ can be substituted in (5.66) with (5.64) such that we derive

$$\begin{aligned}\Delta\eta_L[k] &= \eta_0\left[(1-\Delta\nu[k]-\nu[k])^2-(1-\nu[k])^2\right]\\ &= \eta_0\left[\Delta\nu[k]^2-2\Delta\nu[k]+2\Delta\nu[k]\nu[k]\right]\\ &= \eta_0\left[\Delta\nu[k]^2-2(1-\nu[k])\Delta\nu[k]\right]\ ,\end{aligned} \tag{5.67}$$

which plugged into (5.65) gives us the inequality

$$\Delta\nu[k]^2-2(1-\nu[k])\Delta\nu[k]+\beta(1-\nu[k])^2 \geq 0\ . \tag{5.68}$$

relating ν with β.

The left side of this inequality is a parabola opened upwards with respect to $\Delta\nu[k]$, whose zeros are calculated by

$$\Delta\nu_{1/2}[k] = (1-\nu[k])(1\pm\sqrt{1-\beta})\ . \tag{5.69}$$

In consequence, the inequality (5.68) is only fullfiled, if $\Delta\nu[k]$ lies outside the region restricted by the zeros $\Delta\nu_{1/2}$ of the quadratic equation. Considering the conditions $0 \leq \nu \leq 1$ and $0 < \beta < 1$ and in consequence $-1 \leq \Delta\nu \leq 1$, only the solution

$$\Delta\nu[k] \leq (1-\nu[k])\left(1-\sqrt{1-\beta}\right) \quad \forall k \tag{5.70}$$

is valid. Thus, the inequality (5.70) has to be fulfilled to guarantee stable operation with the proposed command governor and a dynamic allocation of the input and state limits between the controller and the desired trajectory.

On the other hand, with (5.68), the required decay rate β can be calculated with

$$1 > \beta \geq \frac{2(1-\nu[k])\Delta\nu[k]-\Delta\nu[k]^2}{(1-\nu[k])^2} \quad \forall k. \tag{5.71}$$

for a given desired trajectory. If (5.71) is not be fulfilled with the desired trajectory, a regeneration of the trajectory with rate constraints on the input and states or a QLF with a larger decay rate β is required.

5.8.3 Combining SAL and DAL

In the previous sections, two different methods to estimate the DoA have been presented. While the SAL has a constant limiting level set, the DAL allocates the limits depending on the desired trajectory at each sample step and thus the EDoA changes over time.

The EDoAs of both methods can be nested, overlapping or even identical. In consequence, it might happen that e_C is inside both EDoAs (DAL and SAL) or only one of them. If e_C is only inside of one EDoA, the corresponding allocation method is used to calculate a temporary point $e_\mathcal{T}$. Contrary, if e_C is inside both EDoAs, both methods provide a value for $e_\mathcal{T}$. As DAL and SAL use the same controller, the method is selected based on the values of the temporary distance $e_{\mathcal{T},d}$ and the temporary orientation $e_{\mathcal{T},\theta}$. The smaller $e_{\mathcal{T},d}$ and $e_{\mathcal{T},\theta}$, the larger the resulting error e_C and thus the higher the outputs of the controller u_C. To prioritize the error reduction in orientation, at first values of $e_{\mathcal{T},\theta}$ are compared and only if SAL and DAL provide the same values, the temporary distance $e_{\mathcal{T},d}$ is used to select the desired method.

As a result of combining both methods, both EDoAs are joined and the overall command governor and controller performance can be increased.

5.8.4 Remarks to Modifications to the Friction Compensation

In Chapter 4, a trajectory has been optimized using the full nonlinear model, including the nonlinear friction. In consequence, if the TWIP exactly follows the desired trajectory $\mathring{x}$, no additional friction compensation is required. If the friction compensation introduced in Section 5.2 will be used together with the trajectory from Chapter 4 without any modifications, the nonlinear friction will be compensated twice. To avoid this, the input $u_{F,\mathring{x}}$ to compensate the nonlinear friction based on $\mathring{x}$ as well as the input $u_{F,\hat{x}}$ based on the estimated state $\hat{x}$ is calculated with (5.6). Afterwards, instead of $u_{F,\hat{x}}$ only the difference

$$u_F = u_{F,\hat{x}} - u_{F,\mathring{x}} \tag{5.72}$$

is applied to the plant input to compensate the nonlinear friction properly during trajectory tracking.

5.8.5 Experimental Results

The proposed control structure to perform trajectory tracking has been tested experimentally and the results are presented in the following. To test the command governor, an offset of 1.5 m in x and y has been added at $t = 1\,\mathrm{s}$ to the trajectory presented in Chapter 4. This leads to an error e_d in d of $\frac{3}{\sqrt{2}}$ m which is outside the DoA and thus stable tracking will fail without the use of the command governor.

In Figure 5.8 and Figure 5.9 the results of the experiment, using the proposed command governor for trajectory tracking are shown. The dotted blue lines in Figure 5.8 are the values of the desired trajectory and the applied offset for x and y at $t = 1\,\mathrm{s}$ can be recognized. The robot requires about 5 s to catch up to the shifted (or new) trajectory. For this, the forward speed v_d is increased from the desired value up to 0.85 m/s. To accelerate and decelerate at the beginning and the end of the maneuver, the robot tilts up to 6.5° in α. The temporary values for the desired distance d and heading angle θ calculated by the command governor are shown in the plots of d and θ by dashed orange lines. To ensure stable operation, temporarily desired values are used to calculate the control error which can be recognized in the plots of $e_{C,d}$ and $e_{C,\theta}$. Doing so, the control error e_C is limited to be inside the DoA by the command governor, as can be seen in the plot $e_{C,d}$, where the control error in distance is less than 0.15 m whereas the distance error e_d is about $\frac{3}{\sqrt{2}}\,\mathrm{m} \approx 2.12\,\mathrm{m}$ after the shift of the trajectory.

Similar to the command governor used for setpoint tracking, the orientation of the robot has to be modified from the desired value of the trajectory, as can be seen in the plot of θ. Again, this is done by the guidance algorithm by manipulating the desired orientation of θ, such that the robot drives to the desired trajectory.

Figure 5.9 visualizes the operation of the command governor during the maneuver with five plots. The upper diagram, labeled with *CG status* shows if the command governor is in use, which corresponds to $e_\mathcal{T} \neq 0$. At $t = 1\,\mathrm{s}$, where the offset is added to the desired trajectory, the command governor is activated. After reaching the new desired trajectory at $t \approx 5.4\,\mathrm{s}$, the command governor is deactivated again.

The second and third plot show the values of $V_{e_\mathcal{T}}$ of the QLF, calculated with $e_\mathcal{T}$, the limiting level set η_L as well as the safe level set η_S derived with the SAL and DAL

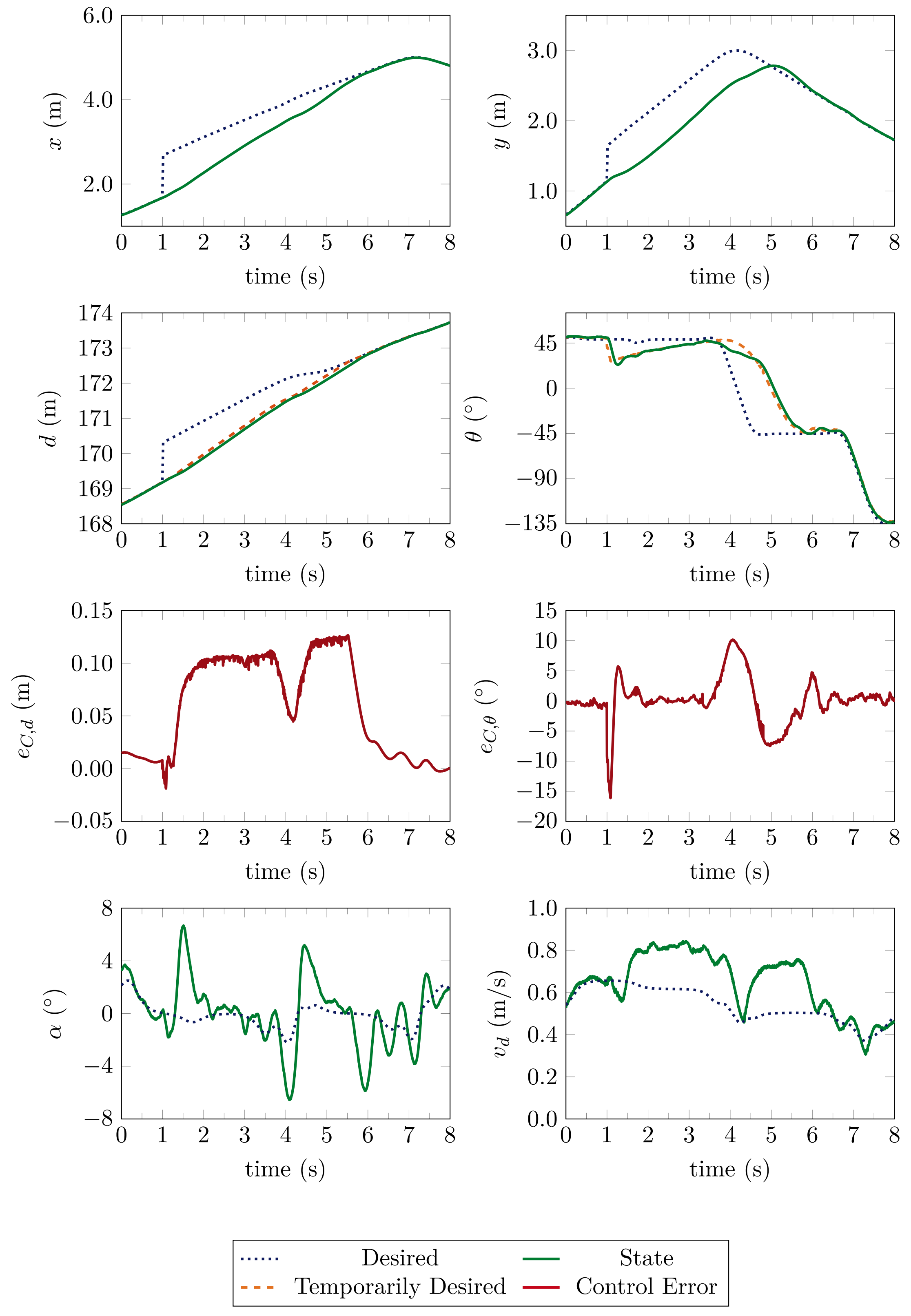


Figure 5.8: Experimental results trajectory tracking

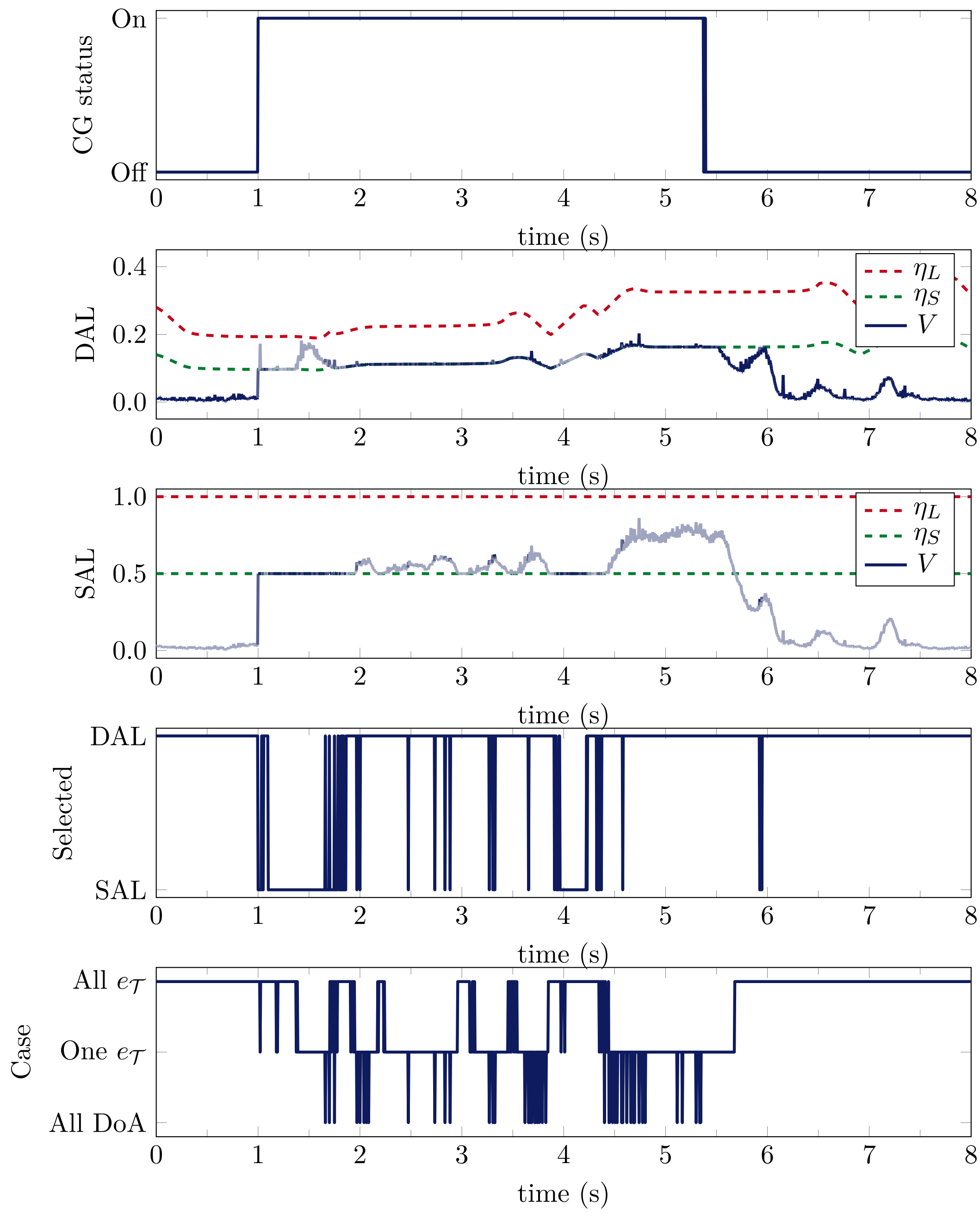


Figure 5.9: Experimental results command governor trajectory tracking

methods. SAL has a fixed limiting level set $\eta_L = 1$ and safe level set $\eta_S = 0.5$. Contrary, the plot for DAL shows time variant limits η_L and η_S.

The fourth plot, labeled with *Selected*, indicates if the temporary equilibrium $e_\mathcal{T}$ from SAL or DAL is used to manipulate the control error e_C. In addition, in the plots for DAL and SAL the line color of $V_{e_\mathcal{T}}$ is faded if the method is not selected.

$V_{e_\mathcal{T}}$ of both methods never exceeds the limiting level set but often the safe level is violated if the method is not active. On the other side, during the active phase of the command governor, $V_{e_\mathcal{T}}$ is equal to η_S of the selected method. This is the result of the derivation of $e_\mathcal{T}$ inside the command governor, which ensures that $V_{e_\mathcal{T}}(e_C)$ is equal to

η_S, as introduced in Section 5.6.

The lowest plot of Figure 5.9 reveals the internal status of the DAL and SAL during the maneuver. The y-label 'All $e_{\mathcal{T}}$' indicates that both methods are able to provide an $e_{\mathcal{T}}$ such that $V_{e_{\mathcal{T}}}(e_C) \leq \eta_S$ is fulfilled. In this case, the control distance error and control orientation error is used to select between DAL and SAL. 'One $e_{\mathcal{T}}$' labels the case where only one method is able to provide a temporary equilibrium and in this situation, the corresponding method is used. The label 'All DoA' marks the case, where both methods are able to calculate an $e_{\mathcal{R}}$ leading to $V_{e_{\mathcal{R}}}(e_C) \leq \eta_L$ but both are not able to provide an $e_{\mathcal{T}}$ that fulfills $V_{e_{\mathcal{T}}}(e_C) \leq \eta_S$. Similar to the setpoint tracking case, the last case happens quite often during the operation of the command governor, as the states used to calculate $e_{\mathcal{T}}$ such that $V_{e_{\mathcal{T}}}(e_C) = \eta_S$ are estimated and disturbed by measurement noise.

The experimental results show that the proposed structure for trajectory tracking combining SAL and DAL is able to ensure a stable transition from one trajectory to another, even if large tracking errors occur. Again, the applicability and robustness of the proposed and implemented algorithms could be demonstrated.

5.9 Concluding Remarks

In this chapter, a novel control structure to perform setpoint and trajectory tracking in the presence of large control errors has been presented. A major contribution is the derivation of a complete discrete-time treatment of all elements of the control structure. Moreover, the presented guidance algorithm enables that a desired setpoint or trajectory can be reached even though the TWIP underlies nonholonomic constraints. In addition, a friction compensation is proposed to reduce the effects of nonlinear mechanical friction and to increase the congruence of the LTI-model used for control synthesis.

For stabilization and tracking a discrete-time linear-quadratic regulator (LQR) has been designed. This is done using the decoupled linear state space models for the forward dynamics and heading dynamics, which eases the control synthesis, as only two scalars have to be tuned in simulations and experiments. In addition, the approach offers a clear relation between the parameters to tune and the effect on the error compensation related to the longitudinal and heading motions of the robot. Moreover, the design procedure stays the same if the motor currents are neglected and the tuned parameters can be reused.

In the next step, the domain of attraction (DoA) of closed-loop system is estimated using a quadratic Lyapunov function (QLF). As the system is subject to state and input constraints, the DoA is limited. Using the introduced LMIs the matrix P of the QLF can be efficiently calculated by solving a convex optimization. Moreover, with the introduced procedure, the limiting level set of the QLF is known which defines the estimated domain of attraction (EDoA) limited by all state and input constraints. This approach is easy to implement and can be efficiently solved.

Based on the calculated EDoA and the limiting level set, a command governor algorithm is introduced to extend the DoA of the closed-loop system of the TWIP. This is done by the derivation of a reference equilibrium $\mathcal{R}$, which is inside the EDoA. Then a temporary equilibrium $\mathcal{T}$ is calculated such that the current state is still inside the EDoA. In particular, $\mathcal{T}$ is derived to lead the TWIP into the direction of the desired equilibrium $\mathcal{D}$. In the case of the proposed setpoint tracking algorithm, $\mathcal{D}$ is the new set point. In the case of the introduced trajectory tracking, $\mathcal{D}$ is reached, if the robot exactly follows the desired trajectory. In both cases, the control error then is zero. To enhance the performance of the algorithm for trajectory tracking, static and dynamic allocations of the input and state limits between the desired trajectory and the stabilizing controller are presented.

As another major contribution experiments on a MIMO-system for setpoint and trajectory tracking are presented to prove the applicability of the algorithms in practice. The problems arising from nonholonomic constraints stay for the TWIP, thus stability can only be guaranteed for the linear system but not for the nonlinear system with the guidance algorithm. This problem is unsolved in the literature up to now and remains. Nevertheless, the presented experimental results highlight that in practice good results for setpoint tracking as well as trajectory tracking can be achieved. In particular, large setpoint changes and trajectory offsets could be handled in experiments without the loss of stability. Finally, the proposed control algorithms showed impressive performance, even in presence of disturbances, measurement noise and state estimation errors, which are present in every real system.

Chapter 6

State Estimation with Time Delayed Measurements

In the preceding chapter control algorithms for the TWIP have been presented. To calculate the control error e, besides the desired state $\mathring{x}$, the estimated state $\hat{x}$ of the TWIP calculated by a state estimator as shown in Figure 5.1, is required. In consequence, a state estimation algorithm is developed and presented in this chapter.

A novel state estimation for the TWIP is introduced, which uses the local sensor measurements from the accelerometer, the gyroscope and the encoders as well as the remote measurements from the remote tracking system introduced in Section 2.4. In particular, the presented state estimation is a mixture of a time-variant, nonlinear extended Kalman filter (EKF) and a time-invariant, linear, optimal state estimator (or stationary Kalman filter). Contrary to an EKF for the TWIP, the presented algorithm is able to run on the microcontroller used with limited computational capacity. Moreover, the deterioration of the estimated state due to the non-constant transmission delays of the remote measurements received is minimized by the proposed technique. In addition, the introduced algorithm incorporates the property, that the motion of the TWIP underlies nonholonomic constraints, in a novel way to reduce the estimation error.

At the beginning of this chapter, a brief motivation is given and preliminaries are introduced. Afterwards, the key problem is sketched and a deeper look into the nature of the remote measurement delays is taken. This is done to reveal the need for a novel state estimation algorithm as presented in this chapter. In addition, it is explained why a moving horizon estimation (MHE) or an EKF is not applicable for the intended application on the TWIP. Afterwards, the developed cascaded quasi-linear Kalman filter (CQLKF) method is presented and simulation and experimental results are given to prove its applicability. Finally, concluding remarks on the chapter are given.

The first version for an estimator for the TWIP but without the tracking system and thus measurement delays has been presented in Wunderlich [117]. Based on this, Priesack [92] added the tracking system measurements to the estimator. Afterwards, Bürchner [14] added the clock parameter estimation such that time-delays of the remote measurements could be calculated. Note, all three student theses have been supervised and supported by the author. Moreover, the algorithm and thus the results presented in this chapter differ from the preliminary results presented in the student theses.

6.1 Motivation

6.1.1 Problem Description

Typically mobile robots, used for indoor applications, are linked via wireless connections (e.g. Bluetooth, Wi-Fi) to other robots, upper-level command governors or remote sensors. As introduced in Section 2.4 and Section 2.5, a similar experimental setup for the TWIP is used in this thesis. Thereby, an optical tracking system is connected via Bluetooth to the robot as a remote sensor. To improve the state estimation, the tracking system's measurements are used to update the predicted position and orientation of the robot. Contrary, the velocities as well as the tilt angle can be updated with the local sensor measurements from the IMU and the odometry[1].

Unfortunately, the tracking system's measurements are received by the robot with a varying time delay. This is caused by the nature of image processing on a PC as well as the tasks involved in transferring data wirelessly to the robot, namely, output buffering, sending, receiving and input buffering which leads to packet delay variation (PDV) or sometimes called jitter (see Demichelis and Chimento [24]). Commonly, this time delay varies between every data set received by the robot. Especially, if the robot moves fast, the transmission delays mentioned above can deteriorate the estimation of the current position and orientation of the robot as has been pointed out in van der Merwe et al. [107].

6.1.2 Estimation Error due to Delayed Measurements

Let us quickly take a look at the reason for the deterioration. Typically, mobile robots as the TWIP, use measurements from encoders and an IMU to estimate the forward velocity v_d and heading rate v_θ of the robot. By means of integration, the position (x, y) and orientation θ of the robot can be calculated. Based on this dead reckoning navigation, errors due to gyroscope drift and wheel slip are also integrated and lead to cumulated errors in position and orientation. These errors can be bounded and reduced if a tracking system provides absolute measurements of (x, y, θ) to the robot with a fixed sample rate. If the position and orientation calculated by the dead reckoning algorithm are updated with a data set ($y_{T,x}$, $y_{T,y}$, $y_{T,\theta}$) from the tracking system just at the time instant when the data set is received by the robot, neglecting the time delay, a considerable estimation error might occur. Figure 6.1 depicts this phenomenon with experimental data. In the upper plot, the points marked by asterisks show the position of the robot at the time instant when the tracking system gathered the image data, whereas the circles denote the time instant when the position and orientation have been finally received by the robot. The lower plot shows the time delay calculated offline for each transmitted data set from the tracking system. As already mentioned, in this experiment an observer is used which uses the received data immediately to update the estimated values at the current time instant, neglecting the delay. For instance, notice that at time $t = 76.345\,\mathrm{s}$ a large delay of $112\,\mathrm{ms}$ occurs. Instead of reducing the estimation error of the position x, the error has increased through the measurement update. Moreover, the faster the robot moves and the larger the delay of a data set is, the more this error may increase.

[1]The local sensors of the TWIP are introduced in Subsection 2.3.2 and remarks are given in Subsection 2.3.4.

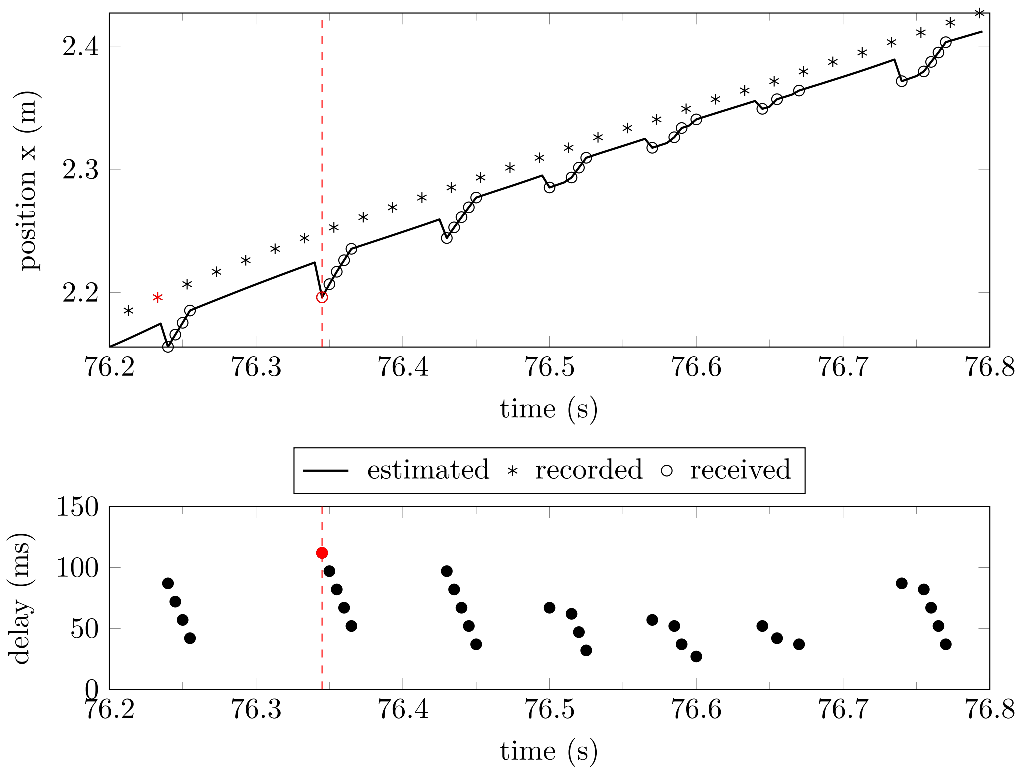


Figure 6.1: Position x_B estimated neglecting the time delays

To solve this issue, state estimators have been presented in literature to handle fixed finite measurement delays e.g. Kaszkurewicz and Bhaya [52]. Moreover, Shi et al. [97] presented a Kalman filter to handle time-varying measurement delays, caused by an packet-delaying network, using a probabilistic approach. Nevertheless, the method requires that the measurement delays can be modeled as random variable which is independent and identically distributed in time. Therefore, in the next subsection the properties of the random process behind the PDV are analyzed for presented experimental setup. This is required to verify, if the methods proposed in literature using an probabilistic model of the delays, can be applied.

6.1.3 Characteristic of Transmission Delays

The properties of the random process of the experimental setup are analyzed by looking at the setup as well as the experimental data. In the lower plot of Figure 6.1, the delays of the data packages are plotted over time. It can be recognized, that the packages are typically received in clumped groups of three to five received packages. Due to this, the delay of the first package received is typically much larger than the delays of the consecutive packages of the same group. This already indicates a (time-)dependence of the individual time delays to each other.

The probability as well as the cumulative distribution function (CDF) of the delays, evaluated with experimental data, are plotted in Figure 6.2. In addition, a log-normal distribution and a Γ distribution (see Hogg et al. [45]) fitted to the experimental data are shown. Both distributions fit quite well with the experimental data. This result coincides with Mukherjee [77], who found that Γ distributions are well suited to model transmis-

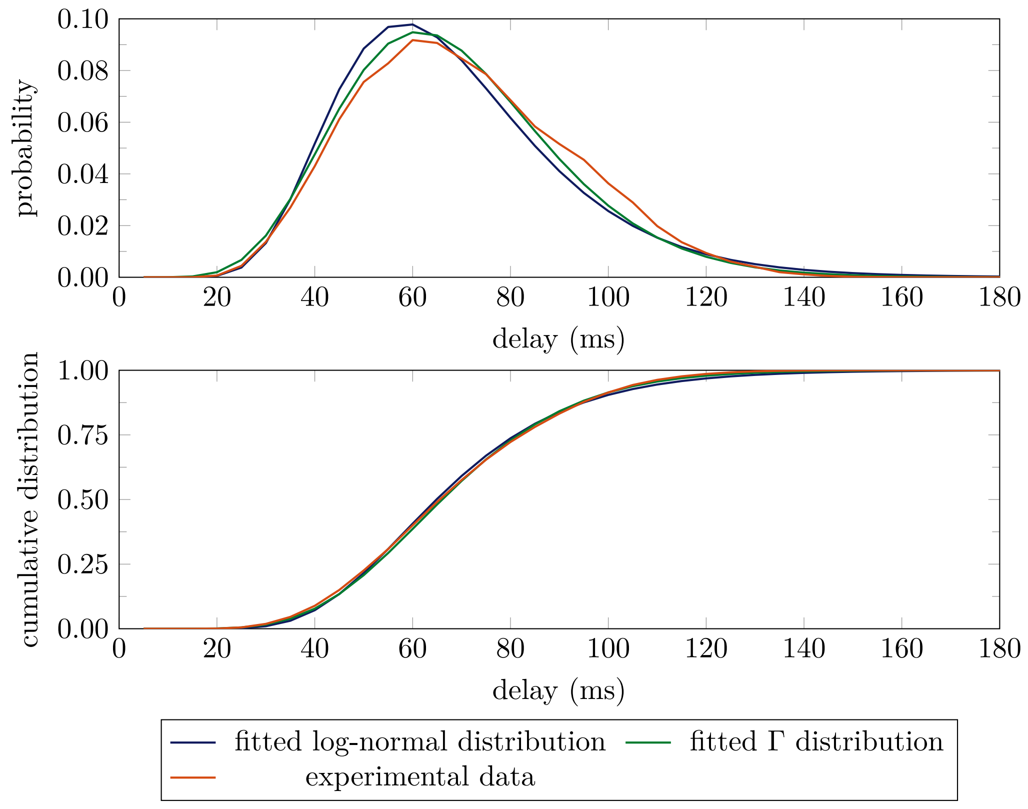


Figure 6.2: Statistical delay time analysis

sion delays on networks while a normal distribution does not. Due to the typically assumed identical distribution in time the delays might be modeled with a log-normal or Γ distribution function.

To check the time-independence of the random process, the Ljung–Box test presented in Ljung and Box [64] is applied to the delays calculated offline with measurement data from experiments. Unfortunately, the result of the Ljung–Box test clearly indicates that the delays are strongly autocorrelated in time. This denies the use of the methods introduced above from literature to handle time delays with an observer which require that the delays are independent and identically distributed in time. It has to be assumed, that all methods based on a Kalman filter, which model the delays as random variable, will show bad performance as it is mandatory that the random variable is independent in time. This is clearly not the case in the presented application.

Moreover, a data package transmission model for simulation purposes using a simple log-normal or Γ random number generator, will show a comparable distribution as the transmission delays retrieved in experiments, but the transmission delays in simulation will be time-independed. This is contrary to the real transmission delays. In consequence, this demands a more sophisticated time delay model for a simulation-based observer testing, tuning and evaluation. Thus, in this thesis, a model which imitates the buffering done by the PC and MCU software combined with a fitted Γ distribution is used to simulate transmission delays which are also time-depended.

As already introduced, an observer algorithm who assumes the delays to be constant or just uses the measurements as they arrive will show avoidable large estimation errors.

An improvement can be reached by a simple ordering of the packages, buffering up to the largest expected delay and then using the constant maximum delay in the filter synthesis. However, if the delays are calculated individually for each measurement data set online with the provided time stamps and this information is used for the measurement update and state prediction, the state estimation error can be reduced even further as van der Merwe et al. [107] shows.

6.1.4 State Estimation Concepts for Delayed Measurements

Different concepts for a state estimation are proposed in Alexander [3], Bak et al. [8], Larsen et al. [61], Pachner et al. [83], van der Merwe et al. [107] which use time stamps (e.g. sample step indexes) of the received measurements to compute the time delay. The calculated time delay of each measurement is then used to improve the state estimation. In particular, in van der Merwe et al. [107] a Sigma-point Kalman Filter with and without sensor delay compensation is presented. Herein, experimental results highlight a considerable reduction of the estimation error if the delay of each measurement received is available and are incorporated for the measurement update. As already mentioned, this is done by calculating the measurement errors at the time instant the measurements have been recorded together with stored measurements. Afterwards, with the stored data (states, measurements, inputs, error-covariance matrix, etc.) the state is re-predicted and corrected up to the current sample step.

All introduced methods from literature are computationally intensive, at least if the nonlinear dynamics of the TWIP are used, and require a computation power far beyond the capacities of the MCU used. In particular, an EKF as proposed in Maybeck [72] requires the evaluation of the Jacobian of nonlinear system dynamics and measurement functions at each sample step. But the most computational cost is caused by the required calculation of a time-integral involving a matrix exponential. The evaluated method from van Loan [108] to compute the integral, as well as the required calculation of a matrix exponential itself with methods as presented in Moler and van Loan [75] showed, that an EKF cannot be used in real-time on the TWIP. Moreover, Sigma-point Kalman Filters (SPKF) as presented in van der Merwe [106] and used for time-stamped transmission delays in van der Merwe et al. [107] also exceeds the available computational resources in magnitudes. Herein, the integration of the nonlinear system dynamics function for each sigma point over one sample step as well as the calculation of the Cholesky factors are violating the real-time limits with the available computational power.

Finally, methods based on a moving horizon estimation (MHE) as presented in Philipp [86], Philipp and Altmannshofer [85] and Philipp and Lohmann [87] require nonlinear optimizations to be performed in real-time on the TWIP and thus cannot be used as well.

Due to this, a novel state estimation algorithm using time-stamped delayed measurements which is able to run on the MCU of the TWIP is presented in the following.

6.2 Preliminaries

6.2.1 Clock Model and Time Conversion

Clocks in electronic circuits utilize crystal oscillators that generate square wave signals with a (almost) stable frequency. The local time of a clock is then calculated by counting the periods of the oscillator signal. Based on such a local time, the tracking system as well as the TWIP generates a trigger for each sample step. Each sample step ${}_Tk \in \mathbb{N}$ of the tracking system is triggered with a sample time (or clock skew) ${}_Tm \in \mathbb{R}^+$. If the clock offset (or initial time offset) ${}_To \in \mathbb{R}$ related to the ideal (or reference) time t is known, then the time instant is defined as

$$t = {}_Tm \cdot ({}_Tk - {}_To) \tag{6.1}$$

for each sample step ${}_Tk$. In the same manner, based on the robot's sample step ${}_Rk$ and its clock offset ${}_Ro$, the time instant

$$t = {}_Rm \cdot ({}_Rk - {}_Ro) \tag{6.2}$$

is defined. The leading lower index R is used to highlight that ${}_Rk$ is related to the sample time ${}_Rm$ and offset ${}_Ro$ of the robot and the lower index T for the tracking system's sample time ${}_Tm$ and offset ${}_To$ respectively.

If both clocks are not running with exactly the same offsets and sample times, a function to derive the concurrent sample step ${}_Rk$ using ${}_Tk$ cannot be formulated. Instead, only the sample step ${}_Rk$ with the smallest difference in time to the sample step ${}_Tk$ is derived. As we are free to choose the time t origin let us assume ${}_Ro$ to be zero and based on (6.1) and (6.2) define the function

$${}_Rk_T = \text{round}\left(\frac{{}_Tm \cdot ({}_Tk - {}_To)}{{}_Rm}\right) \in \mathbb{N} \tag{6.3}$$

to calculate the closest robot sample step ${}_Rk_T$ for a given sample step ${}_Tk$ of the tracking system.

6.2.2 Data Sets

As previously introduced and shown in Figure 2.8, the TWIP receives data packages in an irregular pattern from the tracking system with unknown time delays. Lets us define a data set (or tuple)

$$D_P = ({}_Rk_R, {}_Tk_T, y_{T,x}, y_{T,y}, y_{T,\theta}) \ , \tag{6.4}$$

which contains the sample step ${}_Rk_R$[2] of the robot when the data package has been received by the robot as well as the data from the tracking system included in the package. As introduced in Section 2.5, each package contains the position $y_{T,x}$, $y_{T,y}$ and the orientation $y_{T,\theta}$ of the robot measured by the tracking system as well as the sample step ${}_Tk_T$ of the tracking system at the time instant the measurement has been done. The received data sets can be converted with (6.3) into data sets

$${}^cD_P = ({}_Rk_R, {}_Rk_T, y_{T,x}, y_{T,y}, y_{T,\theta}) \ . \tag{6.5}$$

[2]The trailing lower index R in ${}_Rk_R$ stands for *received*.

Note that, the clock parameters ${}_To$, ${}_Tm$ and ${}_Rm$ have to be known for this operation.

Based on ${}^cD_P$ an observer is able to utilize the modified data to improve the state estimation, by calculating a measurement update at sample step ${}_Rk_T$ with stored local measurement data from the sample step (e.g. from the IMU and odometry) and then predicting the updated estimated states together with stored data (e.g. plant input) till the current sample step ${}_Rk$. Notice that the prediction and update technique does not need absolute but relative time information.

Since both sample times (${}_Rm$,${}_Tm$) deviate from their nominal values and the offset (${}_To$) depends on the time instance when the sample process has been started on the tracking system and the TWIP, they cannot be implemented as fixed parameters in the estimator. This corroborates the demand for an online estimation of the clock parameters as presented in Chapter 7.

6.3 Cascaded Quasi-Linear Kalman Filter

The methods discussed in Section 6.1 are able to reduce the estimation error caused by the varying time-delay of the data package D_P with the ($y_{T,x}$, $y_{T,y}$, $y_{T,\theta}$) measurements from the optical tracking system. This is done, by the use of timestamps provided along the measurements. All algorithms presented above from the literature are not able to run on low-power and cost hardware. Thus, a novel approach for the TWIP is presented in the following which reduces computation costs for the state estimation.

6.3.1 Concept

The concept of the algorithm is to receive a good state estimation with reduced computational requirements through in-cooperation of the timestamps of the delayed measurements to perform the measurement updates at the correct time instances in the past. For this, a buffer is used to store the required past states, inputs and measurements. Then, if a delayed measurement arrives, the update can be calculated at the corresponding sample step in the past. Afterwards, the state is re-predicted and updated with stored data step by step up to the current sample step.

To reduce the complexity and thus computational requirements, the model of the TWIP as well as the observer are split into two parts: a linearized discrete-time submodel

$$\hat{x}_l[k] = A_l^{\circ}\hat{x}_l[k-1] + B_l^{\circ}u_O[k-1] \tag{6.6}$$

with the state vector $\hat{x}_l$, the observer input u_O and a nonlinear discrete-time submodel

$$\hat{x}_u[k] = f_u^{\circ}(\hat{x}_u[k-1], \hat{x}_l[k-1]) \tag{6.7}$$

with the state vector $\hat{x}_u$. Herein, the u and l indices classify the upper, nonlinear and the lower, linear submodel respectively. The union of both state vectors assembles to

$$\hat{x} := (\underbrace{(\hat{x}_\mathrm{B}, \hat{y}_\mathrm{B}, \hat{\theta})}_{\hat{x}_u^\mathrm{T}}, \underbrace{(\hat{\alpha}, \hat{v}_\alpha, \hat{v}_d, \hat{v}_\theta, \hat{i}_\mathrm{R}, \hat{i}_\mathrm{L})}_{\hat{x}_l^\mathrm{T}})^\mathrm{T} \tag{6.8}$$

which is the full estimator state vector $\hat{x}$ of the c-model presented in Subsection 3.1.6.

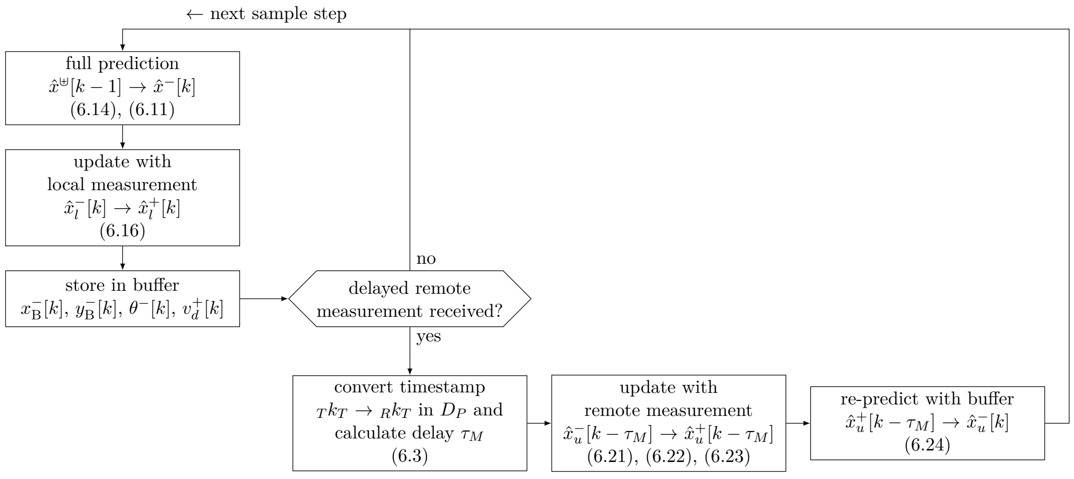

← next sample step
full prediction
$\hat{x}^{\uplus}[k-1] \to \hat{x}^{-}[k]$
(6.14), (6.11)
update with local measurement
$\hat{x}_l^{-}[k] \to \hat{x}_l^{+}[k]$
(6.16)
store in buffer
$x_{\mathrm{B}}^{-}[k]$, $y_{\mathrm{B}}^{-}[k]$, $\theta^{-}[k]$, $v_d^{+}[k]$
delayed remote measurement received?
no
yes
convert timestamp
${}_{T}k_T \to {}_{R}k_T$ in D_P and calculate delay τ_M
(6.3)
update with remote measurement
$\hat{x}_u^{-}[k-\tau_M] \to \hat{x}_u^{+}[k-\tau_M]$
(6.21), (6.22), (6.23)
re-predict with buffer
$\hat{x}_u^{+}[k-\tau_M] \to \hat{x}_u^{-}[k]$
(6.24)

Figure 6.3: CQLKF flowchart

As will be shown later, the upper model only requires the states $\hat{v}_d$ and $\hat{v}_\theta$ out of $\hat{x}_l$ to predict $\hat{x}_u$ and the dynamic equations of $\hat{x}_l$ do not depend on $\hat{x}_u$. In consequence, the structure of the proposed method can be interpreted as cascading two state estimators. In addition, the update gains will be chosen to minimize the sum of the weighted state estimation error and measurements in a cost function in the following. Thus, the proposed state estimation algorithm is called the cascaded quasi-linear Kalman filter (CQLKF).

At every sample step, $\hat{x}_l$ is predicted from $k-1$ to k. As the time-discrete dynamic equations for $\hat{x}_l$ are chosen to be linear, the prediction can be done with minimal computational costs. Similarly, $\hat{x}_u$ has to be predicted at every sample step from $k-1$ to k, which is done by the use of forward Euler integration as the underlying continuous-time dynamic equations are nonlinear. The described state prediction from $k-1$ to k at the beginning of a sample step is shown in a flowchart of the algorithm in Figure 6.3 and is explained further in Subsection 6.3.2.

After prediction, $\hat{x}_l$ can be directly updated by the use of the local sensor measurements, namely the gyroscope, accelerometer and encoders. As all local measurements are available without time-delay, the update of $\hat{x}_l$ can be done directly after prediction. Contrary, to update the upper part $\hat{x}_u$ of the estimated state, a remote measurement from the optical tracking system is required. As the measurements usually arrive with a large and varying time-delay, $\hat{x}_u$ cannot be updated instantaneously. In consequence, the estimated state vector $\hat{x}$ of sample step k always includes not updated (a priori) as well as updated (a posteriori) states. To indicate the update status, a '$-$' for predicted states and a '$+$' for updated states are added as upper right index. Moreover, $\hat{x}_l[k-1]$ of $\hat{x}[k-1]$ is typically updated but $\hat{x}_u[k-1]$ can but does not have to be updated by the proposed algorithm. Thus, the index '$\uplus$' is used to indicate the two possible update statuses of $\hat{x}_u$.

After the prediction of $\hat{x}$ and the local measurement update, the values of $\hat{x}_{\mathrm{B}}^{-}[k]$, $\hat{y}_{\mathrm{B}}^{-}[k]$, $\hat{\theta}^{-}[k]$ and $\hat{v}_d^{+}[k]$ of the current sample step k are stored in a buffer. This reduction of variables to be stored is a result of splitting the state estimator model into a linear and a nonlinear part and the proposed estimator structure, as will become more clear in the next sections. In particular, the reduced set of variables to be stored is an advantage in contrast to the algorithms discussed in Subsection 6.1.4 which typically require at least the full state vector, the input and the local measurement vector to be buffered. Therefore, the proposed estimation algorithm for the TWIP requires only a fraction of the memory for buffering compared to other methods.

After these first steps, the algorithm checks if a new delayed remote measurement has arrived, as illustrated in Figure 6.3. In case of a new measurement, the delay has to be calculated first. For this, the timestamp in the received data package D_P has to be converted with (6.3) into a package ${}^{c}D_P$ with a local timestamp as presented in Subsection 6.2.1 and Subsection 6.2.2. Afterwards, the delay τ_M can be calculated by the difference between the current sample step ${}_{R}k$ of the robot and the tracking system timestamp ${}_{R}k_T$ stored in ${}^{c}D_P$ together with the measurements. Then, the values of sample step $k - \tau_M$ are read from the buffer and a measurement update of $\hat{x}_u[k-\tau_M]^-$ to $\hat{x}_u[k-\tau_M]^+$ is performed. Subsequently, $\hat{x}_u$ is re-predicted from $\hat{x}_u[k-\tau_M]^+$ up to $\hat{x}_u[k]^-$ using the buffered values. Please note, that all re-predicted values are also stored in the buffer and thus replace the former ones of the corresponding sample steps.

6.3.2 State Prediction

Now, let us derive the discrete-time models used for state estimation. For the lower state estimation vector $\hat{x}_l$, the linearized *c-d*-model presented in Subsection 3.1.10 is used with the state vector

$$x_{d,c} := ((d, \theta, \alpha), (v_\alpha, v_d, v_\theta, i_\mathrm{R}, i_\mathrm{L}))^\mathrm{T} \ . \tag{6.9}$$

as given in (3.98). Similar to the linear control design in Section 5.4, linear friction as discussed in Subsection 3.3.1 is assumed. As the TWIP input u (shown in Figure 5.1) is the sum of the feedforward value $\mathring{u}$, the controller output u_C as well as the friction compensation output $u_F(\hat{x}, \mathring{x})$, the observer input u_O has to be expurgated from the non-linear friction compensation components introduced in Section 5.2 and Subsection 5.8.4. u_O for a model assuming a linear friction curve as shown in Figure 3.9 can be calculated by

$$u_O = u - u_{F,\hat{x}} \ , \tag{6.10}$$

where $u_{F,\hat{x}}$ is the friction compensation output presented in (5.6) evaluated with the estimated state $\hat{x}$.

In the next step, the distance d as well as the orientation θ from the linear, continuous-time model and the corresponding state vector $x_{d,c}$ are removed, as they are not included in $\hat{x}_l$. This can be done, as the differential equations of the remaining states in $\hat{x}_l$ are not a function of d and θ.

Similar to the other presented algorithms in this thesis, the state estimation is executed on the MCU with a sample time of 5 ms and thus the linear state space model is discretized in time and given with

$$\hat{x}_l^-[k] = A_l^{\circlearrowleft} \hat{x}_l^+[k-1] + B_l^{\circlearrowleft} u_O[k-1] \tag{6.11}$$

under the assumption of constant input u_O during one sample step. The numerical values of $A_l^{\circlearrowleft}$ and $B_l^{\circlearrowleft}$ are given in (C.1) and (C.2).

As the guidance algorithm in Section 5.3 requires $\hat{x}_\mathrm{B}$ and $\hat{y}_\mathrm{B}$, these states are required and thus included in the estimated state vector $\hat{x}$. In consequence, to predict $\hat{x}_u$ the two nonlinear differential equations

$$\dot{\hat{x}}_\mathrm{B} = \hat{v}_d \cos(\hat{\theta}) \tag{6.12}$$

$$\dot{\hat{y}}_\mathrm{B} = \hat{v}_d \sin(\hat{\theta}) \tag{6.13}$$

for the positions of the TWIP are required. Together with the orientation $\hat{\theta}$, they are numerically discretized by the use of forward Euler integration:

$$\underbrace{\begin{pmatrix} \hat{x}_\mathrm{B}^-[k] \\ \hat{y}_\mathrm{B}^-[k] \\ \hat{\theta}^-[k] \end{pmatrix}}_{\hat{x}_u^-[k]} = \underbrace{\begin{pmatrix} \hat{x}_\mathrm{B}^\uplus[k-1] \\ \hat{y}_\mathrm{B}^\uplus[k-1] \\ \hat{\theta}^\uplus[k-1] \end{pmatrix} + {}_R m \begin{pmatrix} \hat{v}_d^+[k-1]\cos(\hat{\theta}^\uplus[k-1]) \\ \hat{v}_d^+[k-1]\sin(\hat{\theta}^\uplus[k-1]) \\ \hat{v}_\theta^+[k-1] \end{pmatrix}}_{f_u^\circlearrowleft(\hat{x}^\uplus[k-1])} . \tag{6.14}$$

Notice, that the integrated differential equations in Eqs. (6.12) and (6.13) for positions $\hat{x}_\mathrm{B}$ and $\hat{y}_\mathrm{B}$ are nonlinear. In addition, $f_u^\circlearrowleft$ in (6.14) depends on $\hat{v}_d$, $\hat{v}_\theta$ from the lower state vector $\hat{x}_l$ beside $\hat{x}_u$. Due to this, only two variables from $\hat{x}_l$ have to be stored to predict $\hat{x}_u$ from $k-1$ to k. This property is used, to reduce the number of variables that have to be stored in the buffer, as discussed in Subsection 6.3.1 and shown Figure 6.3. Finally, to predict $\hat{x}$ from $k-1$ to k, (6.11) and (6.14) have to be evaluated.

6.3.3 Local Measurement Update

At every sample step k the measurement

$$\underbrace{\begin{pmatrix} y_{E,\delta} \\ y_{G,x} \\ y_{G,y} \\ y_{S,z} \end{pmatrix}}_{y_l[k]} = \underbrace{\begin{pmatrix} 0 & c_{E,\alpha} & c_{E,v_d} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_G & 0 & 0 \\ 0 & c_G & 0 & 0 & 0 & 0 \\ c_S & 0 & 0 & 0 & 0 & 0 \end{pmatrix}}_{C_l} x_l[k] \tag{6.15}$$

from the local onboard sensors of the TWIP is available. The parameters used in the measurement matrix C_l are given in Table C.1. In consequence, the lower state vector $\hat{x}_l$ can directly be updated with

$$\hat{x}_l^+[k] = \hat{x}_l^-[k] + M_l(y_l[k] - \hat{y}_l[k]) \tag{6.16}$$

after prediction as shown in the flowchart in Figure 6.3. The local update gain M_l is a design parameter that will be chosen in Subsection 6.3.5. In the following, the four components of the lower measurement vector y_l are introduced.

$y_{E,\delta}$ is a virtual measurement, derived with the encoder measurement vector y_E, which is presented in Subsection 2.3.2, modeled in Subsection 3.2.3 and illustrated in Figure 3.8. In particular, the (scalar) measurement $y_{E,\delta}$ is calculated by

$$y_{E,\delta} = \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} y_E[k] - y_E[k-1] \end{pmatrix} \tag{6.17}$$

in advance of the evaluation of the state estimator and uses $y_E[k]$ as well as $y_E[k-1]$ from the preceding sample step. Then, based on (3.116) and (6.17), the measurement equation is derived by

$$\begin{aligned} y_{E,\delta} \approx & {}_R m \, c_E \begin{pmatrix} 1 & 1 \end{pmatrix} \omega_\delta = {}_R m \, c_E \; (\omega_{\delta\mathrm{R}} + \omega_{\delta\mathrm{L}}) \\ = & \underbrace{2 \; {}_R m \, c_E \, \frac{1}{r_\mathrm{W}}}_{c_{E,v_d}} v_d \underbrace{-2 \; {}_R m \, c_E}_{c_{E,\alpha}} v_\alpha \; . \end{aligned} \tag{6.18}$$

The entries $y_{G,x}$ and $y_{G,y}$ of y_l are the angular velocity measurements along the ${}_G x$ and ${}_G y$ axis of the gyroscope as introduced in Subsection 2.3.2, modeled in Subsection 3.2.2 and illustrated in Figure 3.7. Thereby, the parameter c_G is the conversion factor from rad/s to LSB.

Last but not least, $y_{S,z}$ is the acceleration measurement of the accelerometer on its ${}_S z$ axis as introduced in Subsection 2.3.2, modeled in Subsection 3.2.1 and illustrated in Figure 3.5. The parameter $c_{S,\alpha} = \mathrm{g} c_S$ in C_l is the conversion factor to from rad to LSB. Herein, it is assumed that $y_{S,z} \approx \mathrm{g}\sin(\alpha) \approx \mathrm{g}\alpha$ holds for small tilt angles and other terms in (3.106) can be treated as disturbance on the measurement.

6.3.4 Delayed Remote Measurement Update

Contrary to the local measurements, the remote measurements

$$\underbrace{\begin{pmatrix} y_{T,x} \\ y_{T,y} \\ y_{T,\theta} \end{pmatrix}}_{y_u[k]} = \underbrace{\begin{pmatrix} c_{T,xy} & 0 & 0 & 0 \\ 0 & c_{T,xy} & 0 & 0 \\ 0 & 0 & 0 & c_{T,\theta} \end{pmatrix}}_{C_u} x_u[k] \tag{6.19}$$

from the optical tracking system, as introduced in Section 2.4 and discussed in Section 6.1 arrive with a varying time delay τ_M. $c_{T,xy}$ and $c_{T,\theta}$ are the conversion factors between the units of the tracking system and ones of the states and are given in Table C.1.

As presented in Subsection 6.3.1, τ_M can be calculated for each remote measurement received using the supplied timestamp and time conversion introduced in Subsection 6.2.1. Afterwards, together with the buffered data, the measurement update on $\hat{x}_u^-$ can be performed at the sample step $k - \tau_M$ by

$$\hat{x}_u^+[k-\tau_M] = \hat{x}_u^-[k-\tau_M] + M_u(y_u[k-\tau_M] - \hat{y}_u[k-\tau_M]) \; . \tag{6.20}$$

The diagonal structure of the measurement matrix C_u in (6.19) offers the possibility to split up the update into three separate updates steps

$$\hat{x}_\mathrm{B}^+[k-\tau_M] = \hat{x}_\mathrm{B}^-[k-\tau_M] + m_{u,x}(y_{T,x}[k-\tau_M] - \hat{y}_{T,x}[k-\tau_M]) \tag{6.21}$$

$$\hat{y}_\mathrm{B}^+[k-\tau_M] = \hat{y}_\mathrm{B}^-[k-\tau_M] + m_{u,y}(y_{T,y}[k-\tau_M] - \hat{y}_{T,y}[k-\tau_M]) \tag{6.22}$$

$$\hat{\theta}^+[k-\tau_M] = \hat{\theta}^-[k-\tau_M] + \underbrace{m_{u,\theta}(y_{T,\theta}[k-\tau_M] - \hat{y}_{T,\theta}[k-\tau_M])}_{\Delta_M^*} \; . \tag{6.23}$$

The three scalar update gains $m_{u,x}$, $m_{u,y}$ and $m_{u,\theta}$ are design parameters and will be chosen in Subsection 6.3.5. After the measurement update, a re-prediction of $\hat{x}_\mathrm{B}$, $\hat{y}_\mathrm{B}$,

$\hat{\theta}$ from $k - \tau_M$ up to k is required. To reduce the computational cost for re-prediction, the prediction equations $f_u^{\circlearrowleft}$ in (6.14) are modified. In particular, as the orientation $\hat{\theta}$ is a linear integration over time, the update Δ_M^* in (6.23) calculated for $\hat{\theta}^-[k - \tau_M]$ can simply be added to all values in the buffer up to $\hat{\theta}^-[k]$. Therefore, the equations

$$\begin{pmatrix} \hat{x}_\mathrm{B}^-[k] \\ \hat{y}_\mathrm{B}^-[k] \\ \hat{\theta}^-[k] \end{pmatrix} = \begin{pmatrix} \hat{x}_\mathrm{B}^{\uplus}[k-1] \\ \hat{y}_\mathrm{B}^{\uplus}[k-1] \\ \hat{\theta}^{\uplus}[k-1] \end{pmatrix} + \begin{pmatrix} {}_{R}m\,\hat{v}_d^+[k-1]\cos(\hat{\theta}^{\uplus}[k-1]) \\ {}_{R}m\,\hat{v}_d^+[k-1]\sin(\hat{\theta}^{\uplus}[k-1]) \\ \Delta_M^* \end{pmatrix} \tag{6.24}$$

have to be evaluated from $k - \tau_M$ to k to re-predict the upper state. In addition, all buffer values of $\hat{x}_\mathrm{B}$, $\hat{y}_\mathrm{B}$ and $\hat{\theta}$ have to be updated. To sum up, minimal computational power is required to update and re-predict the state vector from $k - \tau_M$ to k. Moreover, only the states $\hat{x}_\mathrm{B}$, $\hat{y}_\mathrm{B}$, $\hat{\theta}$ and $\hat{v}_d$ have to be buffered, which leads to relaxed memory requirements on the MCU. This is a strong advantage of the proposed algorithm with a focus on low-power and low-cost MCUs compared to other algorithms (e.g. Alexander [3], Pachner et al. [83] or van der Merwe et al. [107]).

6.3.5 Synthesis

Up to now, the choice and calculation of the update gains M_l, $m_{u,x}$, $m_{u,y}$ and $m_{u,\theta}$ remained open. Let us start with the observer for $\hat{x}_l$ and use an optimal state estimation approach and define the cost function

$$J_l = \sum_{k=1}^{\infty} \Big((x_l - \hat{x}_l)^\mathrm{T} \mathcal{Q}_l (x_l - \hat{x}_l) + y_l^\mathrm{T} \mathcal{R}_l y_l \Big) \tag{6.25}$$

to be minimized by M_l. This problem leads to a discrete-time algebraic Riccati equation (DARE) and can be easily solved by the use of the duality with the discrete-time LQR synthesis and the available design tools. For this, the symmetric positive semi-definite weighting matrix $\mathcal{Q}_l$ and the symmetric positive definite matrix $\mathcal{R}_l$ have to be chosen properly to gain the desired estimation performance. As the underlying system is nonlinear and the identification, as well as accurate modeling of the process and sensor noise, is non-trivial, an optimization that minimizes the estimation error in the simulation environment is used to tune the parameters $\mathcal{Q}_l$ and $\mathcal{R}_l$ as will be discussed in Subsection 6.3.6. If all entries of $\mathcal{Q}_l$ and $\mathcal{R}_l$ are optimized, the best performance might be found in theory, but the curse of dimensionality renders the optimization problem to get unsolvable. Thus, a reduction of parameters is desired.

Firstly, only the diagonal elements of $\mathcal{Q}_l$ and $\mathcal{R}_l$ are chosen to be non-zero. This step already reduces the number of parameters down to ten. Secondly, to make use of symmetries the state transformation (3.92) given in Subsection 3.1.10 is modified and applied to $\hat{x}_l$. From this, the state transformation

$$x_{\omega,l} = \begin{pmatrix} \alpha \\ v_\alpha \\ \omega_{\delta\mathrm{R}} \\ \omega_{\delta\mathrm{L}} \\ i_\mathrm{R} \\ i_\mathrm{L} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & \frac{1}{r_\mathrm{W}} & \frac{d_\mathrm{W}}{r_\mathrm{W}} & 0 & 0 \\ 0 & -1 & \frac{1}{r_\mathrm{W}} & -\frac{d_\mathrm{W}}{r_\mathrm{W}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{T_{\omega,l}} x_l \tag{6.26}$$

from x_l to $x_{\omega,l}$ can be derived for the continuous-time model. Based on the assumption, that $\omega_{\delta\mathrm{R}}$ and $\omega_{\delta\mathrm{L}}$, as well as i_R and i_L, is subject to the same modeling uncertainties, errors and process noise due to symmetry and by the use of the $T_{\omega,l}$, the weighting matrix for the states is defined as follows

$$\mathcal{Q}_l = T_{\omega,l}^\mathrm{T} \operatorname{diag}\left(q_{l,\alpha}, q_{l,v_\alpha}, q_{l,\omega}, q_{l,\omega}, q_{l,i}, q_{l,i}\right) T_{\omega,l} \tag{6.27}$$

with a reduced number of parameters. In consequence, only four parameters remain for the weighting of the six states in x_l. Unfortunately, such a drastic reduction of parameters is not possible for the weighting of the four measurements

$$\mathcal{R}_l = \operatorname{diag}\left(r_{l,E\delta}, r_{l,Gx}, r_{l,Gy}, r_{l,Sz}\right) \tag{6.28}$$

and thus the four parameters remain. To sum up, eight parameters have to be optimized in the optimal cost function to find the update gain M_l which leads to a minimal state estimation error for x_l in simulations.

Let us focus in the next step on the design of the state estimator for x_u. In Subsection 6.3.4, the estimation problem has already been split down into three separate state estimators with only one state variable. Similar to the design for the lower state estimation, update gains $m_{u,x}$, $m_{u,y}$ and $m_{u,\theta}$ which minimize the cost functions

$$J_{u,x} = \sum_{k=1}^{\infty} \Big((x_\mathrm{B} - \hat{x}_\mathrm{B})^\mathrm{T} q_{u,x} (x_\mathrm{B} - \hat{x}_\mathrm{B}) + y_{T,x}^\mathrm{T} r_{u,xy} y_{T,x} \Big) \tag{6.29}$$

$$J_{u,y} = \sum_{k=1}^{\infty} \Big((y_\mathrm{B} - \hat{y}_\mathrm{B})^\mathrm{T} q_{u,y} (y_\mathrm{B} - \hat{y}_\mathrm{B}) + y_{T,y}^\mathrm{T} r_{u,xy} y_{T,y} \Big) \tag{6.30}$$

$$J_{u,\theta} = \sum_{k=1}^{\infty} \Big((\theta - \hat{\theta})^\mathrm{T} q_{u,\theta} (\theta - \hat{\theta}) + y_{T,\theta}^\mathrm{T} r_{u,\theta} y_{T,\theta} \Big) \tag{6.31}$$

for the positions and orientation are required. As can be noticed, the weighting entries for the $y_{T,x}$ and $y_{T,y}$ measurements are chosen to be equal with $r_{u,xy}$, as it is assumed that the tracking system measurements of both quantities underlie the same uncertainties and noise.

Contrarily, the state weights for the position estimation errors are chosen differently as a function of $\hat{\theta}$. If the robot moves along the ${}_Ix$ axis as shown in Figure 3.1 and thus $\theta = 0°$ or $\pm 180°$, the y_B value stays constant while the velocity v_d is integrated and added to x_B. In consequence, estimation errors of v_d mainly accumulate in $\hat{x}_\mathrm{B}$. The opposite happens, if the orientation is along the ${}_{I}y$ axis with $\theta = \pm 90°$. In addition, estimation errors in $\hat{\theta}$ lead to a wrong partition of the driven distance between $\hat{x}_\mathrm{B}$ and $\hat{y}_\mathrm{B}$. This observation is used to define the weighting functions as follows

$$q_{u,x} = q_{u,xy} + |q_{u,v_d} \cos(\hat{\theta})| \tag{6.32}$$

$$q_{u,y} = q_{u,xy} + |q_{u,v_d} \sin(\hat{\theta})| \; . \tag{6.33}$$

Herein, the parameter $q_{u,xy}$ is used to take errors, raised by the estimation of $\hat{\theta}$ as well as from discrete-time integration where $\hat{\theta}$ has to be assumed to be constant over one sample step, into account. In addition, q_{u,v_d} is introduced to consider integration and estimate errors related to $\hat{v}_d$. Due to this, the underling nonholonomic constraints of the movement of the TWIP can be treated inside the state estimation to improve the performance of the estimation of the positions.

The update gains which minimize the cost functions (6.29) and (6.30) can be found by solving the scalar DAREs

$$0 = apa^\mathrm{T} - p - (apc^\mathrm{T})(cpc^\mathrm{T} + r)^{-1}(cpa^\mathrm{T}) + q \; . \tag{6.34}$$

In the introduced case, the parameters are $a = 1$, $c = c_{T,xy}$, $r = r_{u,xy}$ and $q = q_{u,x}$ for $\hat{x}_\mathrm{B}$ or $q = q_{u,y}$ for $\hat{y}_\mathrm{B}$ respectively. As (6.34) is scalar, it can be solved analytically and an equation for $p_x(\hat{\theta})$ with $q_{u,x}$ and $p_y(\hat{\theta})$ with $q_{u,y}$ can be derived. Based on these two equations for p, the scalar functions for the gains

$$m_{u,x}(\hat{\theta}) = p_x(\hat{\theta})c_{xy}(c_{xy}p_x(\hat{\theta})c_{xy} + r_{u,xy})^{-1} \tag{6.35}$$

$$m_{u,y}(\hat{\theta}) = p_y(\hat{\theta})c_{xy}(c_{xy}p_y(\hat{\theta})c_{xy} + r_{u,xy})^{-1} \tag{6.36}$$

can be formulated to update the predictions $\hat{x}_\mathrm{B}^-[k-\tau_M]$, $\hat{y}_\mathrm{B}^-[k-\tau_M]$ using the new measurements $y_{T,x}[k-\tau_M]$ and $y_{T,y}[k-\tau_M]$.

The presented approach incorporates the nonholonomic constraints of the dynamics of the TWIP and thus offers an improved state estimation for the positions compared to solutions using fixed gains. Based on the parameters found in Subsection 6.3.6, the resulting update gains $m_{u,x}(\hat{\theta})$ and $m_{u,y}(\hat{\theta})$ are plotted over $\hat{\theta}$ in Figure 6.4. It can be seen, that the optimized parameters lead to a high update gain for $m_{u,x}$ at $\hat{\theta} = 0°$ or $\pm 180°$ and a quite small gain for $\hat{\theta} = \pm 90°$. The gain $m_{u,y}$ for y_B shows the opposite behavior. This coincides with the intuition, that the prediction error along the direction of the movement is larger than to the perpendicular direction. Finally, as $m_{u,x}(\hat{\theta})$ and $m_{u,y}(\hat{\theta})$ are scalar and only contain $\sin(\hat{\theta})$ and $\cos(\hat{\theta})$ terms, they can be evaluated online at every sample step on the MCU with low computational cost. The three parameters q_{xy}, q_{v_d}, r_{xy} in total for the remote measurement update have to be optimized in the next section to tune the estimation performance of the positions.

Finally, the update gain $m_{u,\theta}$ for the orientation has to be calculated and constant values for $q_{u,\theta}$ and $r_{u,\theta}$ in (6.31) are used. Similar to the position estimation, the cost function (6.31) leads to a scalar DARE and $m_{u,\theta}$ can be calculated analytically or numerically. The parameters $q_{u,\theta}$ and $r_{u,\theta}$ have also be tuned in the next section to minimize the estimation error evaluated by simulations.

6.3.6 Parameter Optimization

To optimize the observer parameters, the simulation environment presented in Section 2.6 and shown in Figure 2.9 is used. Herein, the full nonlinear model with cur-

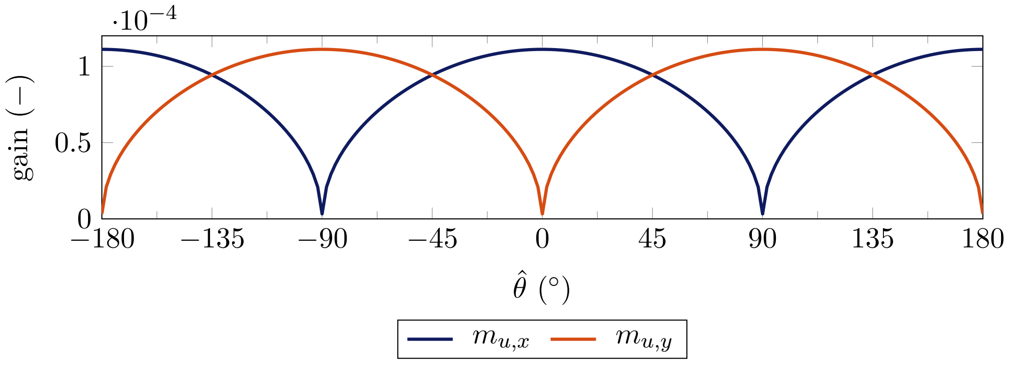


Figure 6.4: Update gains $m_{u,x}$, $m_{u,y}$ over orientation $\hat{\theta}$ for $\hat{x}_{\mathrm{B}}$, $\hat{y}_{\mathrm{B}}$

rent dynamics of the TWIP given in Subsection 3.1.6 and sensor models as presented in Section 3.2 are included as well as an algorithm to imitate the transmission delays of the tracking system measurements. In addition, the friction compensation, guidance algorithm and the LQR presented in Sections 5.2 to 5.4 are used to follow the reference trajectory generated in Section 4.3.

If the observer parameters are optimized based on simulations using the nominal nonlinear model of the TWIP and the linearized version is used for synthesis, the accuracy of the model used inside the estimator will typically be overestimated. In particular, this leads to small values of the corresponding q-parameters in the cost function and small update gains. Also, the same considerations will apply to the sensor models. Unfortunately, the resulting estimator will show bad performance in experiments due to model parameter uncertainties. To avoid this, 168 models with randomly varied parameters are generated, differing from the nominal ones and used to tune the estimator parameters with an optimization algorithm. For sure, in the best case, the random parameter and thus the models differ from the nominal model in a similar manner then the real TWIPs deviate from the nominal model.

Additionally, to ensure a sufficient fast decay of the estimation error

$$\hat{e} = x - \hat{x} \tag{6.37}$$

after initialization, all nonlinear simulation models are initialized with random non-zero state vectors as aspected in experiments. Contrary, the initial state $\hat{x}[k = 0]$ of the estimator is set as a zero vector. Finally, the timespan of the simulations used has to be long enough such that accumulated drift effects due to sensor noise and small offsets will influence the estimation result sufficiently and thus the tuned estimator is able to cope with these effects in experiments afterwards. A timespan of 300 s has been found as a good compromise between incorporating drift effects and minimizing computational cost to perform the simulation-based optimizations.

For the parameter optimization, the cost function

$$J_{opt} = \sum_{k=1}^{\infty} \Big((x - \hat{x})^{\mathrm{T}} \mathcal{Q}_{opt} (x - \hat{x}) \Big) \tag{6.38}$$

is defined to judge the estimator's performance. Hereby, $\mathcal{Q}_{opt}$ is a diagonal weighting matrix used to scale the states based on the minimum and maximum values of the trajectory used. The matrix $\mathcal{Q}_{opt}$ is given in (C.3). Therefore, a minimization of the cost function correlates with an improvement of the estimator performance and thus minimizes state estimation error.

To reduce the number of parameters to be optimized at once, firstly the parameters $q_{l,\alpha}, q_{l,v_\alpha}, q_{l,\omega}, q_{l,i}$ and $r_{l,E\delta}, r_{l,Gx}, r_{l,Gy}, r_{l,Sz}$ of the state estimator for the lower state vector $\hat{x}_l$ are tuned. To avoid a non-unique solution, $q_{l,\alpha} = 1$ has been fixed. Thus, the resulting optimization problem has seven free parameters.

Afterwards, the parameters $q_{u,xy}, q_{u,v_d}, r_{u,xy}$ and $q_{u,\theta}, r_{u,\theta}$ of the estimator for the upper state vector $\hat{x}_u$ are tuned via optimization. Again, to get a unique solution the parameters $q_{u,xy} = q_{u,\theta} = 1$ are fixed. This results in an optimization with three remaining parameters.

The resulting parameters found by simulation-based optimizations using the pattern search algorithm (see Audet and Dennis [7]), where able to give a proper state estimation in simulation and also showed good performance in experiments as well. The parameters found by the optimization are listed in Table C.2. Simulation and experimental results are presented and discussed in the next two sections.

6.4 Simulation Results

In the following, the proposed state estimation algorithm with the optimized parameters is presented with simulation results. Therefore, the TWIP is following the trajectory presented in Section 4.3. In simulations, the real state is known and thus the estimation error can be calculated by (6.37) which is the difference between the state of the nonlinear model and the estimated state. Note, this difference cannot be calculated with the data presented in the next section as the real state of the TWIP is unknown in experiments.

The presented simulation results use accurate clock parameters such that the performance of the pure state estimator can be seen. Also, the nominal model parameters for the TWIP are used. Contrarily, random sensor misalignment, sensor quantization, digital filtering and noise, as introduced in Section 3.2, have been included. Moreover, a random initialization error between the estimator state and model state is set, to evaluate the convergence of the estimated state to the model state.

Table 6.1: Simulation estimation errors ($\hat{e}_{x_\mathrm{B}}$, $\hat{e}_{y_\mathrm{B}}$, $\hat{e}_\theta$)

state	x_B	y_B	θ	α
unit	m	m	°	°
$\max\lvert\hat{e}\rvert$	$3.0 \cdot 10^{-3}$	$4.0 \cdot 10^{-3}$	$1.4 \cdot 10^{0}$	$3.7 \cdot 10^{-1}$
$\mathrm{RMS}(\hat{e})$	$8.0 \cdot 10^{-4}$	$1.0 \cdot 10^{-3}$	$4.4 \cdot 10^{-1}$	$9.5 \cdot 10^{-2}$

Figure 6.5 and Figure 6.6 show the estimation error during trajectory tracking. Herein, the first three seconds are plotted separately with a larger y-axis scale to show the decay of the initial estimation error. Two things can be observed. First, the estimator is able to reduce the initial error of all state variables in less than three seconds. Second, the estimation error is kept small by the algorithm which highlights its ability to reduce the excitation effects from the sensor models and the differences between the partly linear estimator model and the nonlinear model used to simulate the TWIP. Moreover, it can

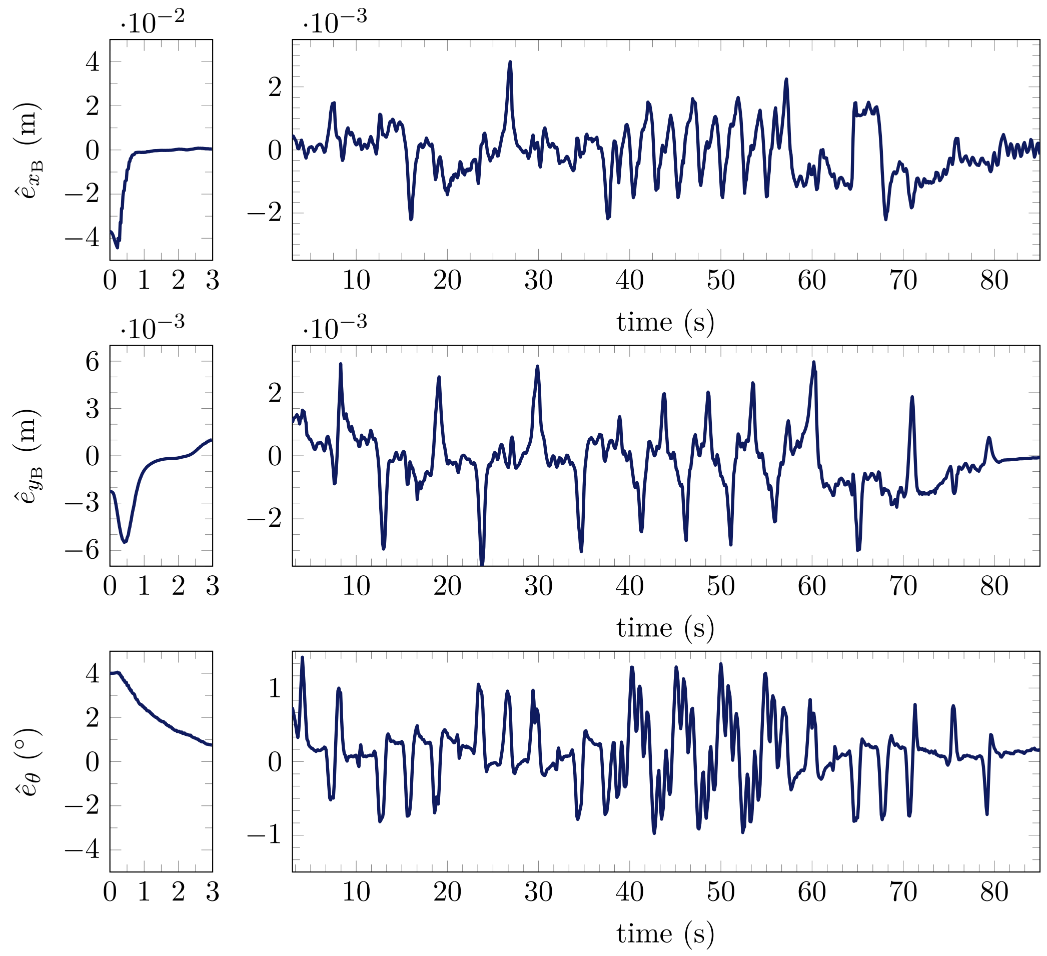


Figure 6.5: Simulation state estimation errors ($\hat{e}_{x_{\mathrm{B}}}$,$\hat{e}_{y_{\mathrm{B}}}$,$\hat{e}_{\theta}$)

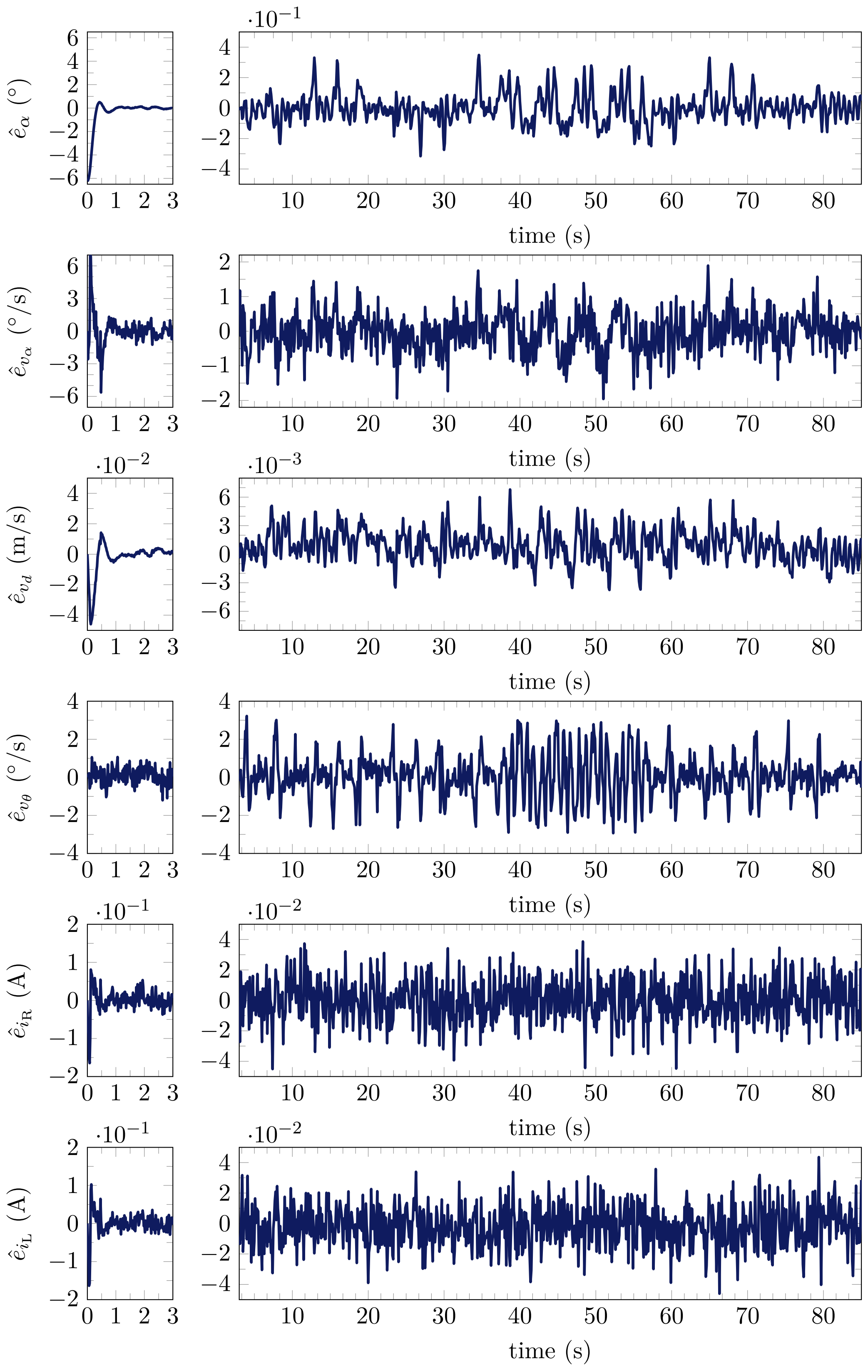


Figure 6.6: Simulation state estimation errors ($\hat{e}_\alpha$,$\hat{e}_{v_\alpha}$,$\hat{e}_{v_d}$,$\hat{e}_{v_\theta}$,$\hat{e}_{i_\mathrm{R}}$,$\hat{e}_{i_\mathrm{L}}$)

be seen in Figure 6.5, that the position errors $\hat{e}_{x_\mathrm{B}}$ and $\hat{e}_{y_\mathrm{B}}$ show peaks, if the $\hat{e}_\theta$ error is large. This is expected, as $\hat{\theta}$ is used to predict the positions $\hat{x}_\mathrm{B}$ and $\hat{y}_\mathrm{B}$. Finally, Table 6.1

Table 6.2: Simulation estimation errors ($\hat{e}_\alpha$,$\hat{e}_{v_\alpha}$,$\hat{e}_{v_d}$,$\hat{e}_{v_\theta}$,$\hat{e}_{i_\mathrm{R}}$,$\hat{e}_{i_\mathrm{L}}$)

state	v_α	v_d	v_α	i_R	i_L
unit	°/s	m/s	°/s	A	A
$\max\lvert\hat{e}\rvert$	$2.5 \cdot 10^{0}$	$6.9 \cdot 10^{-3}$	$3.9 \cdot 10^{0}$	$5.7 \cdot 10^{-2}$	$5.5 \cdot 10^{-2}$
$\mathrm{RMS}(\hat{e})$	$5.8 \cdot 10^{-1}$	$1.9 \cdot 10^{-3}$	$1.1 \cdot 10^{0}$	$1.4 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$

and Table 6.2 list the $\mathrm{RMS}(\hat{e})$ as well as $\max\lvert\hat{e}\rvert$ values of the presented simulation run.

6.5 Experimental Results

In addition to simulations, the performance of the CQLKF has been tested in experiments. Since in experiments the real state of the TWIP is unknown, the difference between the tracking system measurements $y_{T,x}$, $y_{T,y}$ and $y_{T,\theta}$ and the estimated positions $\hat{x}_\mathrm{B}$, $\hat{y}_\mathrm{B}$ and orientation $\hat{\theta}$

$$\Delta_{x_\mathrm{B}} = y_{T,x} - \hat{x}_\mathrm{B} \tag{6.39a}$$
$$\Delta_{y_\mathrm{B}} = y_{T,y} - \hat{y}_\mathrm{B} \tag{6.39b}$$
$$\Delta_\theta = y_{T,\theta} - \hat{\theta} \tag{6.39c}$$

is used to evaluate the estimation performance in experiments. For sure, the real position and orientation of the TWIP will differ from the tracking systems measurement as well. In particular, the tracking system measurements suffer from noise and there are always small differences between the origin $_IB$ assumed during modeling in Section 3.1 and the origin of the TWIP defined by the tracking system software. Moreover, the clock parameters have to be estimated offline as well to convert the measurement data to the same time scale to evaluate (6.39). Thus, clock estimation errors will add an additional error to the calculated difference in (6.39).

Nevertheless, due to a lack of alternatives for comparison, the differences (6.39) are used to evaluate the estimator performance and are plotted in Figure 6.7. In addition, the $\mathrm{RMS}(e_o)$ as well as $\max\lvert e_o\rvert$ values are given in Table 6.3. Based on the results, it can be

Table 6.3: Experimental position and orientation differences (x_B, y_B, θ)

state	x_B	y_B	θ
unit	m	m	°
$\max\lvert e_o\rvert$	$8.3 \cdot 10^{-3}$	$9.2 \cdot 10^{-3}$	$2.6 \cdot 10^{0}$
$\mathrm{RMS}(e_o)$	$3.8 \cdot 10^{-3}$	$3.9 \cdot 10^{-3}$	$1.1 \cdot 10^{0}$

concluded that the proposed estimation algorithm with the optimized parameters offers a good state estimation such that the control algorithms presented in Chapter 5 are not

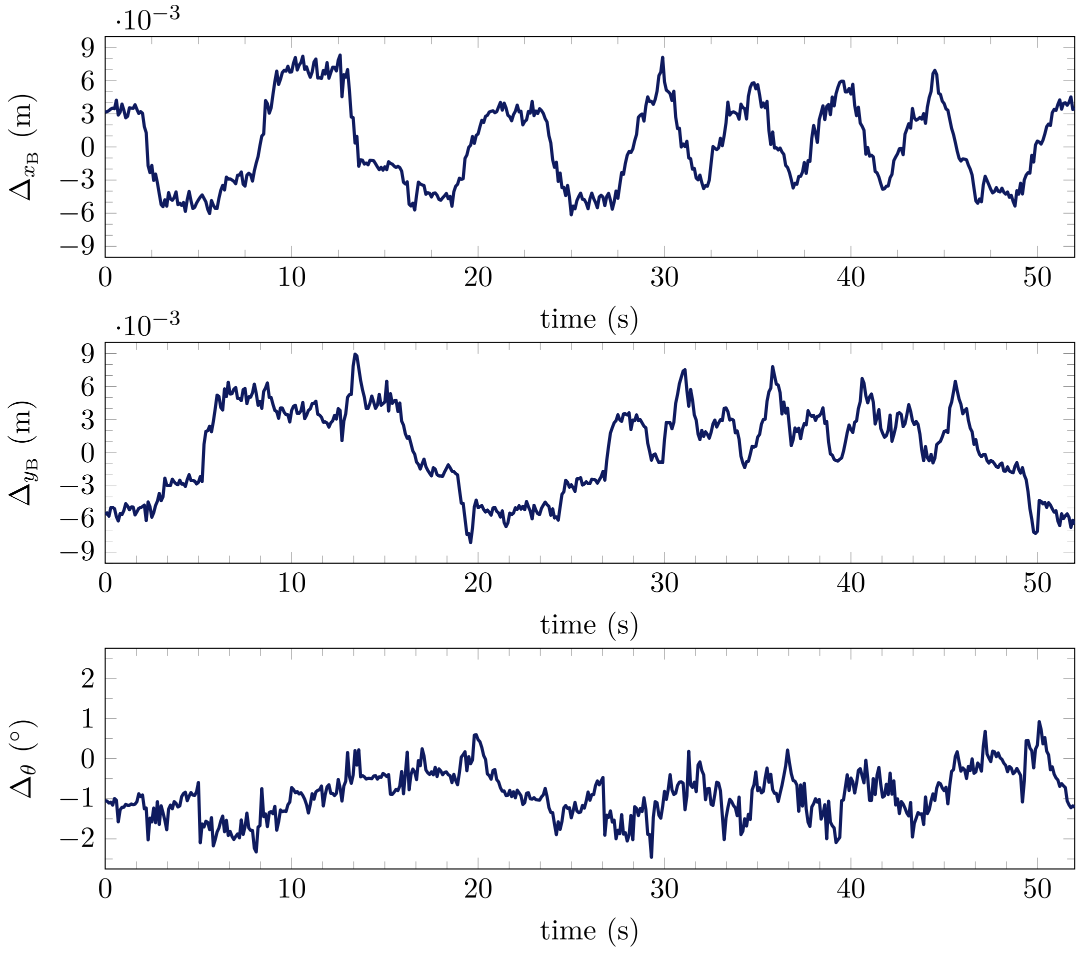


Figure 6.7: Experimental position and orientation differences (Δ_{x_B},Δ_{y_B},Δ_θ)

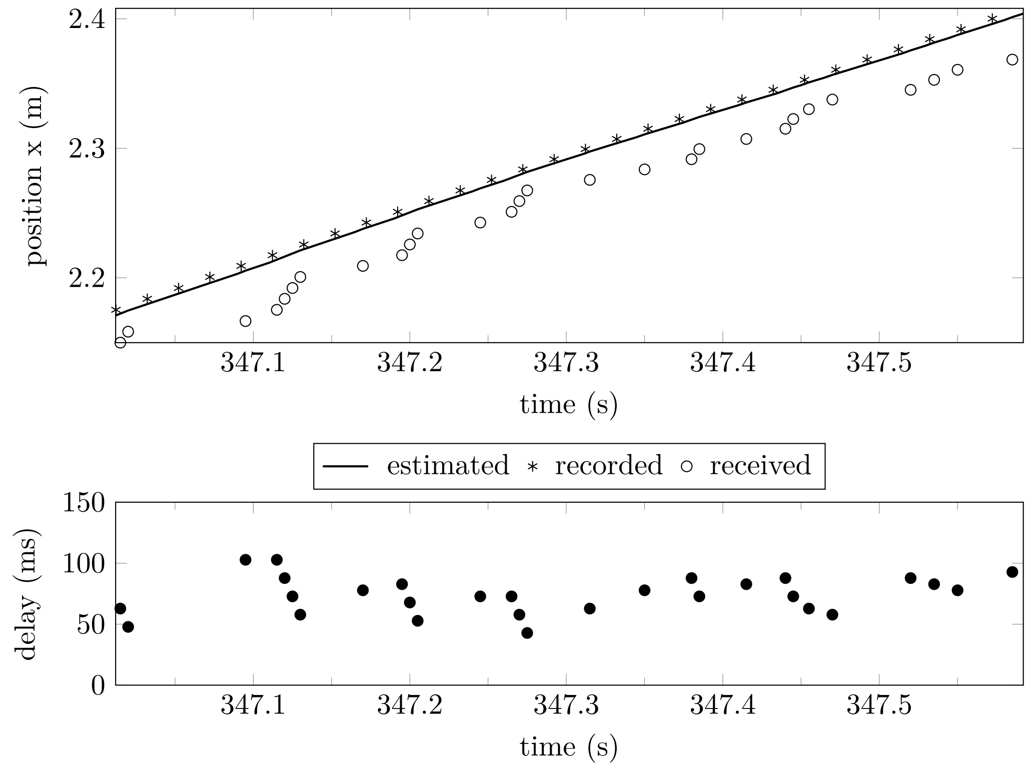


Figure 6.8: Position x_{B} estimated with consideration of the time delays

only able to stabilize the TWIP but also are able to follow the trajectory precisely as shown in the experimental results of Chapter 5. In addition, based on the evaluations with the tracking system measurements in Figure 6.7 and Table 6.3 the estimator shows good performance and small estimation errors for the positions and orientation. Thus, it has been shown that the proposed algorithm is able to mitigate the effects of delayed remote measurements.

At the beginning of this chapter, in Subsection 6.1.1, experimental results from an estimator which directly uses the delayed measurements as they arrive are given in Figure 6.1 for the x_{B} of the TWIP. Based on estimated clock parameters (see Chapter 7) as well as the proposed estimator, an estimation of the position x_{B} is provided in Figure 6.8. Hereby, the algorithm uses the online estimated clock parameters to calculate the delay of each received measurement and uses them for the measurement update as proposed. Similar to Figure 6.1, the points marked by asterisks show the position of the robot at the time instant when the tracking system gathered the image data, whereas the circles mark the time instant when the position and orientation data sets have been finally received by the robot. Compared to the results presented in Figure 6.1, an impressive reduction of the error of the estimated position x_{B} is achieved in Figure 6.8 with the novel algorithm.

6.6 Concluding Remarks

In this chapter, a new state estimation method for the TWIP has been presented which is able to incorporate delayed remote measurements properly. Moreover, the proposed algorithm requires low computational power and small memory requirements.

In addition, a statistical analysis of the non-constant transmission delays has been performed, based on experimental data. Thereby, an important outcome is that the identical distribution in time can be approximated by a Γ distribution function but the random process is not time-independent. This result is contrary to the assumptions usually defined during the estimator design in literature, as most methods rely on the time-independence of the random process to yield a good estimation performance.

The presented method uses the timestamps included in the delayed measurements and incorporates them during the measurement update, such that no assumptions on the random process of the delays are required to design the estimator. To reduce the computational power requirements the proposed estimator structure of the presented cascaded quasi-linear Kalman filter (CQLKF) splits the TWIP model into a linear and a nonlinear part. This results in an algorithm that is able to run on low-cost and low-power hardware while still offering good state estimation results. In addition, the underlying nonholonomic constraints of the movement of the TWIP are used by the presented algorithm to improve the estimator's performance. Finally, the applicability and accurate state estimation performance, even in presence of large measurement delays, have been demonstrated in experiments.

To conclude, the presented cascaded quasi-linear Kalman filter (CQLKF) is a mixture of a time-variant, nonlinear EKF and a time-invariant, linear, optimal state estimator and thus is able to run on a microcontroller with limited computational power and memory. Thereby, the deterioration of the estimation state due to the non-constant time-delays of the transmitted remote measurements is minimized by the proposed algorithm. Hence, this chapter contributes a novel solution to reduce the algorithm complexity for state estimation for the TWIP and to decrease the required computational power while still offering an accurate state estimation performance in presence of large time-varying measurement delays.

Chapter 7

Efficient Clock Parameter Estimation

In Subsection 6.2.1 a clock model and time conversion have been introduced, which is used in Chapter 6 to convert the timestamps of the tracking system into local time of the TWIP. For this, the clock parameters of the tracking system and the TWIP have to be available for time conversion. In this chapter, a novel method is proposed to estimate clock parameters online and efficiently on a microcontroller of a small mobile robot. Particularly, the presented algorithm requires low computational power and has small memory requirements. The introduced algorithm estimates the clock parameters by repeated online optimizations which compare remote and local sensor measurements mapped to a common local time. No additional software (e.g. a Network Time Protocol (NTP) client) is required on the tracking system or the robot since the incremented sample step index of the tracking system and the robot are used. Based on the estimated clock parameters provided by the algorithm and the sample step index the time delay of each measurement received can be calculated and used by an observer as presented in Chapter 7.

The remainder of the chapter is organized as follows. Firstly, a brief motivation is given. Afterwards Section 7.2 presents the proposed method to estimate online clock parameters on a microcontroller as used by the TWIP. Thereupon, Section 7.3 presents the algorithm and Section 7.4 discusses the advantages and disadvantages of the proposed method. In Section 7.5 additional implementation details of the algorithm are given. Experimental and simulation results are discussed in Section 7.6. Finally, Section 7.7 concludes the contribution presented in this chapter with remarks.

An early version but with some differences from the proposed algorithm has been presented in Bürchner [14], a student thesis supported and supervised by the author.

7.1 Motivation

In general, tracking systems trigger image acquisition properly in real-time with a high sample rate and provide for every recorded set of images a unique, incremented sample step index. This index can be interpreted as a local time stamp of the data set. The gathered images are analyzed by the tracking system's software, running on a PC, which computes the position and orientation of the robot. Commonly, the incremented sample step index of the images is appended to the resulting position and orientation data set. This also applies to the tracking system used for the TWIP as presented in Section 2.4. Based on this index, the delay of each received data set can be calculated and utilized to improve the estimate by the robot's observer as has been shown in Chapter 7. However, this requires the clock sample times and offsets and thus the clock time of the tracking system and the robot to be known.

One solution to compute the time delays of the received measurements is, to synchronize the clocks on the tracking system's PC as well as on the robot. This synchronization can be done by a Network Time Protocol (NTP) client Mills [73] which ensures a synchronized and precise system clock on the tracking system's PC as well as on the robot. The trigger signal for the tracking system's cameras to capture a new set of images has then to be used to capture a time stamp of the NTP synchronized time on the PC. This time stamp is then transmitted together with the position and orientation data to the robot, instead of the sample step index of the tracking system. Based on this time stamp, the robot is able to calculate the time delay of each measurement based on its local NTP synchronized clock. An extended review of possible methods of clock synchronization over wireless networks is offered by Sundararaman [103], Wu et al. [116], Djenouri and Bagaa [32]. Nevertheless, this solution to calculate the time delays has drawbacks in a wide number of applications: Firstly, the trigger of the tracking system has to be accessible as a signal to a software program to record a time stamp and a separate program might have to be written for that purpose. To avoid a missed trigger signal and to minimize a delay between the trigger signal and the record of the time stamp, this software has to be executed in a real-time environment. Secondly, running an NTP client demands computational power, memory and communication bandwidth on the mobile robot. This is contrary to the fact, that small robots typically use a low-cost microcontroller with limited computational power, memory and energy consumption. Furthermore, on several mobile robots, no operating system is used and no NTP client software implementation is available. Last but not least, the precision of the time synchronization with NTP strongly depends on the delay time of the network connection and it is assumed that the delay in both directions (sending and receiving) is equal. Additionally, with large communication delays, clock errors of up to 100 ms are not unusual (see Wang et al. [110], Mills [73]). These clock errors will cause an error in the calculation of the time delay of a received measurement and might deteriorate the estimation of the position and orientation of the TWIP.

Another possible solution is to run a moving horizon estimation (MHE) Philipp [86], Philipp and Altmannshofer [85], Philipp and Lohmann [87] on the TWIP. If the clocks of the tracking system and the robot are not synchronized and have different sample times and offsets, the computation of the time delays requires the knowledge of the clock parameters, namely, clock sample time and clock offset. An MHE is able to estimate the states (including position and orientation) as well as the clock parameters by solving an optimization problem in real-time at every sample step of the robot. Hence an MHE is perfectly suited to provide a model-based state estimation, on one hand, and to estimate simultaneously the clock parameters on the other. Running an MHE in real-time with a small sample time (e.g. 5 ms) on a small mobile robot like the TWIP requires a vast amount of computational power, memory and energy, since the data for the whole horizon has to be stored and a large optimization problem has to be solved every sample step. This demand on computational resources and energy denies the application on the TWIP.

In conclusion, the aforementioned methods, the clock synchronization over the network or the use of an MHE, are not suitable to run on a microcontroller with low computational power and limited memory and thus on small mobile robots. Therefore in this chapter a new algorithm to estimate online the clock parameters on the mobile robot is presented which utilizes the properties of typical indoor applications of mobile robots and thus the TWIP.

7.2 Concept

Two sensor outputs that measure the same state (e.g. orientation angle velocity of the robot v_θ) with time stamps in the form of sample steps (e.g. ${}_Rk_G, {}_Tk_T$) are used to perform a clock parameter optimization. One measurement comes from a remote sensor (e.g. tracking system) and the other from a local robot's sensor (e.g. gyroscope). Both measurements are stored over a chosen horizon length together with their sample steps.

The stored timestamped measurements are used to evaluate a cost function to be optimized. In the cost function J, the sample steps ${}_Tk_T$ of the remote sensor's measurements, as well as the sample steps ${}_Rk_G$ of the local sensor's measurements, are converted with (6.1) respectively (6.2) into time. Afterwards the absolute error between the two measurement signals (interpolated in time) is summed up over the chosen measurement horizon and the mean error is calculated.

The optimization parameters can be reduced to two parameters if the robot's time is used as a reference. In this case, let us assume that the sample time ${}_Rm$ is fixed at its nominal value and the clock offset ${}_Ro$ is zero. The described cost function is then used to derive the tracking system's sample time ${}_Tm$ and clock offset ${}_To$ via optimization.

It is important to set up a cost function that leads to robust optimization results, especially in presence of measurement noise. This means that a sequence of two simultaneously recorded measurement signals have to differ from each other if the estimated clock parameters differ from the real ones and thus the converted signal is shifted and warped in time.

This is visualized in Figure 7.1 which shows an exemplary plot of the heading rate of the robot versus time. Besides the reference signal, a curve with a clock offset deviation and a curve with a sample time deviation is drawn in the figure. The lower subplot in the figure shows the absolute error between the reference signal and the curves with a parameter deviation. It can be observed, that the signals have to show a certain amount of excitation, such that a parameter deviation causes a noticeable difference from the reference signal. In the time intervals, 0.2 s to 1.5 s and 2.5 s to 4.0 s the heading rate curves show a high rate of change and the parameter mismatch causes an error between the signals. Contrary, in the time interval from 1.5 s to 2.5 s the heading rate curves show no difference even though the clock parameters differ.

Also, another observation can be done based on Figure 7.1. While a clock offset, even with a short horizon (e.g. from 0 s to 1 s), causes an error between the reference signal and the shifted signal, the error generated by a deviation of the sample time remains small at the beginning and increases with the length of the horizon. Thus the horizon length influences the sensitivity of the cost function. In summary, the cost function will be sensitive to the clock offset ${}_To$ even with a short horizon, the sensitivity of the cost function with respect to the sample time ${}_Tm$ increases with the length of the horizon.

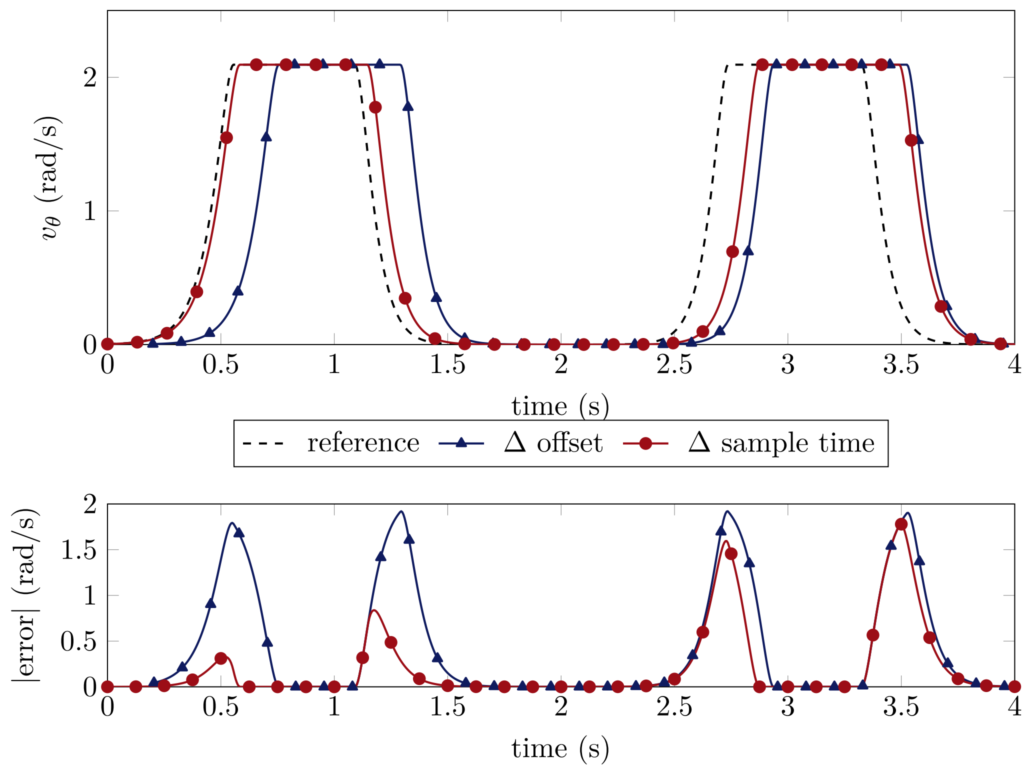


Figure 7.1: Change of clock parameters and absolute error in heading rate to reference

Consequently, the idea of the method is to acquire signals from a remote and a local sensor which are timestamped and measure the same state. To minimize memory consumption, power and computational cost only sequences of the signals which are sensitive to a misalignment in time are stored to run optimizations. To maintain a cost function, which is sufficiently sensitive to the sample time ${}_Tm$, a finite number of sequences of the signals over time are stored to obtain a long horizon. Every time a new signal sequence has been recorded, an optimization will be executed. The optimization minimizes the cost function $J({}_Tm, {}_To)$ which calculates the absolute error between the remote and local signal sequences stored. The outputs of the optimization are optimal values of ${}_Tm$ and ${}_To$ which can be used by an observer subsequently.

The choice of the measurement of the state v_θ is based on the observation that mobile robots used indoor are operated in a very similar way typically: In most cases, robots follow way-points through a building and turn at predefined points to drive around corners or objects Jose and Antony [50], Zhang et al. [120], Lim et al. [63]. Even if they follow more smooth, optimized trajectories, they usually still show segments of straight movements and sequences where the robot turns. The measured heading rate slopes of the robot can thus be used in this case to determine the clock parameters with an optimization.

Hence, the following design decisions are proposed to run the clock parameter estimation online on a microcontroller with reduced memory, computational power and energy requirements:

C.1 Since the absolute time is not typically required by an observer but the time information in terms of the sample step, the sample time ${}_Rm$ of the robot is assumed to match its nominal value and its offset ${}_{R}o$ to be zero. Thus only the sample time ${}_{T}m$ and clock offset ${}_{T}o$ of the remote sensor are used as optimization parameters in the cost function $J({}_{T}m, {}_{T}o)$.

C.2 Only one timestamped measurement signal from a local and one from a remote sensor are used for the optimization, namely the heading rate of the robot.

C.3 To reduce the amount of data to be stored, only sequences of measurements are recorded that exhibit at least a certain amount of excitation. The variance of a sequence of measurements is hereby used as an indicator for the sensitivity of the cost function which will use the recorded data sequence for evaluation. This leads to a long horizon with less required memory, since only isolated data sequences are stored, at which the cost function $J({}_{T}m, {}_{T}o)$ is sensitive to the clock parameters ${}_{T}m$ and ${}_{T}o$.

C.4 An optimization method has to be used which can be easily split into a sequence of small tasks and can be executed over several sample steps on a microcontroller. This is required to ensure that a microcontroller is able to calculate also other tasks in real-time (e.g. observer and controller) without a task scheduling operating system while running an optimization.

7.3 Algorithm

For the TWIP, an algorithm is presented which utilizes the proposed concept. The algorithm running on the robot's microcontroller can be split into three parts namely filter, storage and optimization. The structure and the purpose of each part will be discussed in the following. A sketch of the structure of the algorithm is shown in Figure 7.2.

7.3.1 Filter

The filter part is visualized at the top of Figure 7.2 and can be split into two sections: the processing of the local measurements on the left side and the processing of the remote measurements on the right side. The local section processes the data sets

$$D_R = ({}_{R}k_G, y_{G,x})$$

which includes the measured heading rate $y_{G,x}$ of the gyroscope as well as the sample step ${}_{R}k_G$ of the robot when data from the IMU has been read and the remote section processes the data sets D_P, given in (6.4), containing the measured orientation $\theta_T = y_{T,\theta}$ from the tracking system. Both measurements are subject to noise which should be smoothed out for optimization.

For both measurements ($y_{G,x}$, $y_{T,\theta}$) a symmetric Savitzky-Golay-Filter (SG-Filter) proposed by Savitzky and Golay [95] is used to smooth out noise. Moreover, the SG-Filter for $y_{T,\theta}$ is designed to output the first derivate of the input signal and thus $\tilde{v}_{\theta,T}$. As a digital differentiator, it minimizes the noise-amplification factor and attunes high frequencies (see Luo and Ying [68]). An SG-Filter fits a low-degree polynomial to a subset of the data points by minimizing the least-square error (see Orfanidis [82]). Filtering is done by means of convolution, which is easy to implement and utilizes fast multiply-accumulate instructions on microcontrollers with digital signal processing capabilities.

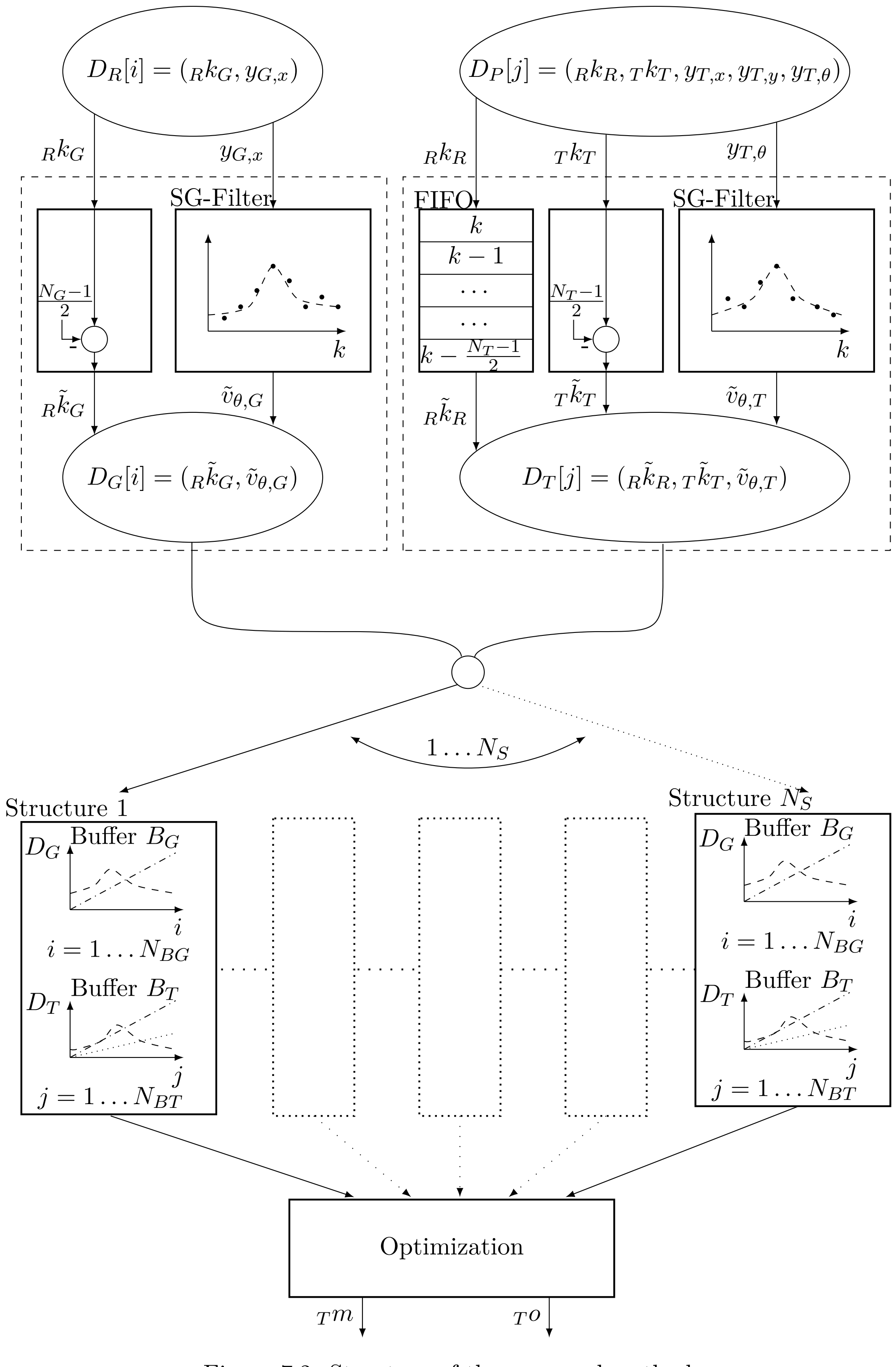


Figure 7.2: Structure of the proposed method

The measurement values used for convolution can be stored in a simple First-In-First-Out (FIFO)-buffer.

It has to be noted that both measurements are part of a data set. The measurement of the gyroscope $y_{G,x}$ is related to the robot's sample step ${}_Rk_G$ when the IMU has been read. Also the measured orientation $y_{T,\theta}$ from the tracking system and its derivative are related to the sample steps ${}_Tk_T$ and ${}_Rk_R$ included in the data set D_P. Since the filter output introduces a time delay of $N_D = \frac{1}{2}(N_F - 1)$ sample steps from its input (N_F is the filter order), the related data set items have to be equally delayed. This is necessary to ensure that the sample steps stored in the data sets D_G and D_T still correspond to the filtered measurements. This delay can be implemented by the use of a simple FIFO-buffer. To derive the values ${}_T\tilde{k}_T$ and ${}_R\tilde{k}_G$ a FIFO is not required since they can be calculated by subtracting N_D from the current sample steps ${}_Tk_T$ and ${}_Rk_G$ at the filter input.

7.3.2 Storage

The outputs of the filter part are fed into the storage part of the algorithm, shown in the middle of Figure 7.2. To record the data sets from the filter part which are required for the clock parameter estimation, an array of N_S structures is used. Each structure has two buffers. In the first buffer B_G, N_{BG} data sets

$$D_G = \left({}_R\tilde{k}_G, \tilde{v}_{\theta,G}\right) \ , \tag{7.1}$$

can be stored. To record N_{BT} data sets

$$D_T = \left({}_R\tilde{k}_R, {}_T\tilde{k}_T, \tilde{v}_{\theta,T}\right) \ , \tag{7.2}$$

a second buffer B_T is used.

During the recording of data sets into one of the structures, the online variance of $\tilde{v}_{\theta,G}$ is calculated using Welford's Online algorithm (see Welford [111]). For each data set stored, the algorithm updates the mean and variance of all stored data sets in the structure. This is done by recurrence formulas which ensure a numerically stable calculation of mean and variance (see Knuth [57]). Due to the recurrent updates, a peak in required computational power is avoided when the buffer has been filled up.

If both buffers in a structure are full, it is checked, if the variance exceeds the defined threshold $\sigma^2_{\min}$. If the variance is above $\sigma^2_{\min}$ the data is kept and used for optimization. If not, both buffers are deleted and the structure will be used again to record new data. As soon as all structures are filled up with recorded data and always, if one structure has been filled up with new data, the parameter optimization is started. When the optimization is finished, the structure with the oldest data is deleted and used to record new data.

7.3.3 Optimization

The last part of the algorithm corresponds to the parameter optimization, visualized at the bottom of Figure 7.2. It uses the data sequences stored in all structures and their buffers.

The Downhill Simplex method (DSM) proposed by Nelder and Mead [79], which is also known as Nelder-Mead method is used to run the optimization. It is a direct search method and one of the most popular methods for nonlinear optimization (see Lagarias et al. [60]), e.g. used in Matlab. The algorithm has been selected since it is easy to implement and minimizes a scalar nonlinear cost function using only function values without the need for gradients. This is an important property for our application since interpolation is required during the evaluation of the cost function and the online calculation of the gradients causes high computational costs. Moreover, an analytical solution of the gradients might be complicated to calculate and has to be changed, if the interpolation function is modified. The calculation of gradients with single precision numeric and noisy measurement data is also avoided by the chosen optimization method.

For each evaluation of the cost function $J({}_Tm, {}_To)$ by the DSM all data sets stored are used. Firstly, the time instants

$$t_G(i) = {}_Rm \cdot {}_R\tilde{k}_G(i) \text{ for } i = \{1 \dots N_{BG}\} \tag{7.3}$$

for each sample step ${}_R\tilde{k}_G$ in D_G as well as the time instants

$$t_T(i) = {}_Tm \cdot ({}_T\tilde{k}_T(i) - {}_To) \text{ for } i = \{1 \dots N_{BT}\} \tag{7.4}$$

for all sample steps ${}_T\tilde{k}_T$ in D_T are calculated. Based on t_G and t_T the values of $\tilde{v}_{\theta,T}$ in D_T are linearly interpolated at the time instants t_G:

$$\bar{v}_{\theta,T} = \text{interpolate}(\tilde{v}_{\theta,T}, t_T, t_G)\ . \tag{7.5}$$

The set Ω_v contains all indices $i \in \{1 \dots N_{BT}\}$ where the time vectors t_G and t_T overlap and thus interpolation is possible and N_v is the number of elements in Ω_v. In the next step the difference

$$\epsilon_v(i) = \tilde{v}_{\theta,G}(i) - \bar{v}_{\theta,T}(i) \quad \forall\, i \in \Omega_v \tag{7.6}$$

is calculated. Afterwards the cost function value

$$J_\epsilon = \frac{1}{N_v} \sum_i \text{abs}\left(\epsilon_v(i)\right) \quad \forall\, i \in \Omega_v \tag{7.7}$$

can be defined. Finally, for all data packages received, the transmission times

$$\Delta_t(i) = {}_Rm \cdot {}_R\tilde{k}_R(i) - t_T(i) \text{ for } i = \{1 \dots N_{BT}\} \tag{7.8}$$

are calculated. The transmission of a data set from the tracking system to the robot needs at least one sample step of the robot and thus a minimum time $\Delta_t \geq {}_Rm$. To ensure that only clock parameters which fulfill this condition lead to a small cost function value, the final cost function value with the penalty cost for not feasible (too small or even negative) transmission delays

$$J = J_\epsilon + \alpha_p \sum_i \text{abs}(\Delta_t(i) - {}_Rm) \tag{7.9}$$
$$\text{for } i \in \{j = \{1 \dots N_{BT}\} : (\Delta_t(j) < {}_Rm)\}$$

is computed where α_p weights the penalty term.

The optimal clock parameter ${}_Tm$ and ${}_To$ are returned when the optimization is finished.

7.4 Advantages and Disadvantages

The method sketched reduces computational costs, required memory and thus energy and the algorithm can be implemented and executed in real-time on a microcontroller of a small mobile robot. To summarize, the proposed method has the following advantages:

- A SG-Filter can be implemented as a convolution of FIFO-buffered input data with the SG-Filter-Parameters which is easy to code and fast to execute on a microcontroller and has a small and fixed memory demand.
- The delay of the sample step indices in the data sets can be implemented through a FIFO-buffer or calculated in the case of ${}_Tk_T$ and ${}_Rk_G$.
- By the use of Welford's Online algorithm, the calculation of the variance is split into small steps during the recording of the data sets into a buffer. This avoids a peak in required computational power.
- The variance of a sequence of measurements is a simple indicator of the sensitivity of the cost function.
- Since only sequences of data sets, which will lead to good optimization results, are stored and used, the algorithm has a lower memory consumption compared to an algorithm that stores all data at a similar horizon length. Furthermore, the number of data points that have to be interpolated, evaluated and summed up by the cost function is reduced which also reduces the computational cost.
- By the use of the DSM a well-known and quite common optimization algorithm is used which is easy to implement. Furthermore, the calculation of gradients with single precision numeric and noisy measurement data is avoided.

The major disadvantages of the proposed method are also listed:

- If the robot's trajectory has no turn or turns with small heading rates, the variance threshold may never be reached and thus no data will be stored and no optimization will be performed. The method relies on recurring intervals with changing heading rates.
- The algorithm requires some time to fill up all structures before the first optimization can be started. During this period no optimized clock parameters are available for subsequent algorithms like an observer.
- Offsets like gyroscope drift and measurement noise will limit the accuracy of the results of the parameter optimization since the measurement data is used in the cost function. The higher the disturbances of the measurements are the less accurate the estimated parameters (${}_Tm$ and ${}_To$) will be and the more often an optimization is required to keep the time error small.

Implementation details and algorithm parameters used for the presented simulations and experiments are given in the following section. Readers which are not interested in implementation details and do not intend to apply the proposed algorithm themself might skip over Section 7.5. In Section 7.6 simulation and experimental results are given.

7.5 Implementation Details

The algorithm has been implemented in Matlab and the code generation toolboxes are used to generate C-Code which can be compiled and run on the microcontroller of the TWIP.

A hardware timer triggers an interrupt every 5 ms to perform a sample step on the robot. Firstly, the routine to read the local sensors and evaluate the received Bluetooth data is executed. Afterwards the clock estimation method, the observer, and the controller functions are called. Finally, the actuator outputs are updated and the data which should be logged is sent out via Bluetooth.

As already mentioned, the IMU is read first and a new data set D_R is created. If a new data set has been received via Bluetooth by the robot during the last microcontroller sample step, a data set D_P is also created including the received information. If no data has been received from the tracking system an empty data set is generated. Subsequently the algorithm function Clock Parameter Estimation (CPE) (see Algorithm 1) is executed and both packages D_R and D_P are passed. If the function has finished, the subsequent functions, for the observer and controller, etc., are executed. It is imported to ensure, that the sum of the execution time of all functions during a sample step never exceeds the sample time of 5 ms.

The most important CPE functions of the algorithm are listed in Algorithm 1 and Algorithm 2 as pseudo code based on the structure visualized in Figure 7.2. Line 3 in Algorithm 1 calls the function to record data sets and Line 4 to Line 13 cover the optimization part. The introduced filter part is presented in Algorithm 2 from Line 2 to Line 5. Finally, Line 6 to Line 18 in Algorithm 2 introduces the storage part. The implementation details of the filtering, data storage, and optimization parts are explained separately in the following subsequent sections.

7.5.1 Filtering

The IMU with the gyroscope is read at each sample step of the microcontroller and thus the filter for the data sets D_R is executed at every sample step too. However, the tracking data set filter and the corresponding FIFO is only executed, if a package D_P with new data has been received by the microcontroller during the last sample step.

The gyroscope measurement $y_{G,x}$ of the heading rate in the dataset D_R as well as the tracking system's measurement $y_{T,\theta}$ of the heading angle is fed into a SG-Filter of $N_G = N_T =$ 15th order, fitting a 5th degree polynomial. Both SG-Filters are set up symmetrically and the output is evaluated at the middle of the fitted polynomial. Contrary to the SG-Filter for the gyroscope heading rate, the SG-Filter for the tracker measurement is designed to give the filtered first-order derivative $\tilde{v}_{\theta,T}$ as output. In addition, it is assumed that the deviation of the sample time ${}_{T}m$ from its nominal value has a negligible influence on the time derivative calculated by the filter.

To keep up the connection between the robot's sample step index ${}_{R}k_{R}$ and the filter output $\tilde{v}_{\theta,T}$ a FIFO-buffer with a length of $(N_T - 1)/2$ is used. At every evaluation of the filter, the sample step index ${}_{R}k_{R}$ from the current data set D_T is enqueued and the oldest entry in the buffer is dequeued. The dequeued sample index ${}_{R}\tilde{k}_{R}$ then corresponds to the filter output $\tilde{v}_{\theta,T}$ since it has been delayed in the same amount.

Algorithm 1 Clock Parameter Estimation Function

```
Input:
    $D_P$   ▷ Tracking data set of current robot sample step (empty if no data has been received)
    $D_R$   ▷ Gyro data set of current robot sample step
    $_Tm$   ▷ Clock sample time from last optimization
    $_To$   ▷ Clock offset from last optimization
Output:
    $_Tm$   ▷ Current estimate of the clock sample time
    $_To$   ▷ Current estimate of the clock offset
Static:
    $A$   ▷ Array of $N_S$ structures to recorde data sets $D_T$ and $D_G$ in buffer
    $i$ (init=0)   ▷ Current buffer index
    $f_b$ (init=true)   ▷ Flag which is set if data recording is active
    $\sigma^2$ (init=0)   ▷ Variance of current gyro buffer
 1: function CPE($D_R$,$D_P$)
    ▷ Check if data recording is active
 2:     if $f_b$ is true then
 3:         ($f_b$,$\sigma^2$) = recDataSet($D_P$,$D_R$,$A$,$\sigma^2$,$i$)
 4:     else
 5:         $f_O$ = runOptimizationStep($A$,$_Tm$,$_To$)
            ▷ Check if optimization is finished
 6:         if $f_O$ is true then
            ▷ Reactivate data recording
 7:             $i$ = getOldestStructureIndex()
 8:             cleanStorage($A(i)$)
 9:             cleanVariance()
10:             $f_b$ = true
                ▷ Update new estimated parameter
11:             ($_Tm$,$_To$) = getOptimizationResults()
12:         end if
13:     end if
14:     return ($_Tm$,$_To$)
15: end function
```

Algorithm 2 Data Set Record Function

```
Input:
    $D_P$   ▷ Tracking data set of current robot sample step (empty if no data has been received)
    $D_R$   ▷ Gyro data set of current robot sample step
    $A$   ▷ Array of $N_S$ structures to record data sets $D_T$ and $D_G$ in buffer
    $\sigma^2$   ▷ Variance of current gyro buffer
    $i$   ▷ Current buffer index
Output:
    $f_b$   ▷ Flag which is set if data recording is active
    $\sigma^2$   ▷ Variance of current gyro buffer
 1: function recDataSet($D_P$,$D_R$,$A$,$\sigma^2$,$i$)
    ▷ Call SG-Filter
 2:     $D_G$ = SGFGyro($D_R$)
 3:     if $D_P$ is not empty then
 4:         $D_T$ = SGFTracking($D_P$)
 5:     end if
 6:     storeGyroDataSet($D_G$, $A(i).B_G$)
 7:     $\sigma^2$ = updateVariance($D_G$, $\sigma^2$)
 8:     if $D_P$ is not empty then
 9:         storeTrackingDataSet($D_T$, $A(i).B_T$)
10:     end if
11:     if $A(i)$ is full then
12:         if $\sigma^2 > \sigma^2_{\min}$ then
            ▷ Deactivate recording and start optimization
13:             $f_b$ = false
14:         else
            ▷ Restart data recording in same buffer
15:             cleanStructure($A(i)$)
16:             cleanVariance()
17:         end if
18:     end if
19:     return ($f_b$,$\sigma^2$)
20: end function
```

The sample step index ${}_Rk_G$ can be calculated with ${}_R\tilde{k}_G = {}_Rk_G - (N_G - 1)/2$. In the same way, the corresponding sample index ${}_Tk_T$ can be calculated with ${}_T\tilde{k}_T = {}_Tk_T - (N_T - 1)/2$. Due to this, a FIFO for ${}_Rk_G$ and ${}_Tk_T$ is not needed as shown in Figure 7.2.

7.5.2 Data Storage

A total number of $N_S = 8$ structures are used, each with a buffer B_G and a buffer B_T. All variables are saved as single precision floats or 32 bit-integers.

The size of the buffer B_G is chosen to store $N_{BG} = 100$ data sets D_R. Only every second sample of the gyroscope filter output is stored in the buffer B_G. This is done to double the horizon length without increasing the required buffer memory and the number of points that have to be interpolated and evaluated in the cost function during optimization. A period of 1 s of data can be recorded in the buffer B_G of one structure in this case.

Based on the nominal sample time ${}_Tm$ of the tracking system with 20 ms, a size of $N_{BT} = 50$ has been chosen, to record the same time period of tracking system data sets D_T into the buffer B_T.

Each structure requires 2.95 kBytes to store the given number of data sets and variables which makes in total 23.6 kBytes for all structures.

Based on preliminary experimental results an average delay of approximately 65 ms of the received data packages from the tracking system has been evaluated. The start of the recording of the tracking system data sets is delayed about the average transmission delay time of 65 ms, to prevent that the first data sets of the tracking system cannot be used for interpolation and for the calculation of the cost function because there is no matching gyroscope data.

The variance threshold has been experimentally tuned to a value of $\sigma^2_{\text{min}} = 1\,(\text{rad/s})^2$. If the variance of the stored gyro data in the buffer B_G is below σ^2_{min} when the buffer is full, the data in both buffers is deleted and the recording is restarted into the same structure.

To ensure that the stored data in the structures cover a long period of time which is required to get a sensitive cost function, a long horizon is required, as discussed in Section 7.2. Thus, a break is implemented which ensures a minimum time gap between the last recorded data set in the last filled-up structure and the trigger to start the storage of data into a new structure. A break of 12 s has been experimentally chosen and is a compromise between minimum horizon length and time which is to required renew the data sets in the structure. To avoid an overload with details in the listing of Algorithm 1, this feature is not sketched but has been implemented in the code.

The number of structures and the length of the buffers has been chosen such that the memory demand of the algorithm does not exceed the available resources of the microcontroller. The more memory and the more computational power a microcontroller offers, the longer the horizon can be made and the more data sets can be evaluated. Offline experiments with measurement data show, that this improves the estimation of ${}_Tm$, ${}_To$. Unfortunately, a higher number of structures or longer buffer lengths leads to a sample time violation in the online implementation.

7.5.3 Optimization

As already introduced, the Downhill Simplex method (DSM) introduced by Nelder and Mead [79], Lagarias et al. [60] is used to run the optimization online on the microcontroller. In each iteration, the DSM performs different tasks to vary three points of (${}_Tm$, ${}_To$) in order to find the optimal values. To split the tasks of each iteration into small steps, a state machine is implemented. In each stage of the state machine at least for one point, the cost function has to be evaluated. At every call of the optimization (Algorithm 1 Line 5) the cost function is evaluated for only one structure and one point. Thus, to evaluate the cost function, N_S calls of the optimization function are required. This separation into small tasks is done to limit the maximum computation time of DSM at each sample step of the microcontroller.

To ensure that the optimization time is limited, two stop criteria are introduced. The optimization is either stopped after 40 iterations or if the area covered by the three points is below $1 \cdot 10^{-7}$.

Algorithm 3 sketches the procedure to calculate the cost function value of one storage. First, from Line 2 to Line 7 the sample steps stored in the buffer B_T and B_G of the current structure are converted into time variables with the current set of clock parameters (${}_Tm$, ${}_To$). Afterwards, for each data set in B_G an interpolated value at the time t_G of the tracking system's $\tilde{v}_{\theta,T}$ is calculated (Line 8). The interpolation in time is done linearly between the two nearest points $\tilde{v}_{\theta,T}$ in time t_T. If a value t_G is out of the interpolation region, the value $\bar{v}_{\theta,T}$ is marked as invalid.

For all valid interpolation points $\bar{v}_{\theta,T}$, Line 12 calculates the difference between $\tilde{v}_{\theta,G}$ and $\bar{v}_{\theta,T}$ which is added to the cost function value in Line 13.

As introduced, the transmission of a data set from the tracking system to the robot needs at least one sample step of the robot and thus a minimum time ${}_Rm$. Thus, Line 17 to Line 22 add a penalty cost to the function value as given in (7.9). The penalty parameter $\alpha_p = 0.05$ is used to weight the constraint violation influence.

7.6 Results

7.6.1 Simulation

Firstly, the proposed method is evaluated via simulations. A trajectory optimization using a model of the TWIP to compute a trajectory of over 4700 s is utilized to generate measurement data sets. The clock parameters of the tracking system are set to ${}_To = 1000$ samples and ${}_Tm = 20.006$ ms such that they differ from their nominal parameters. Contrary, the clock parameters of the robot are defined to their nominal values.

Two different cases are considered. In the first case, the algorithm is fed with data sets with no noise. The second test case is set up to investigate how noise affects the parameter estimation. Especially noise on the tracking system measurement θ_T may decrease the accuracy of the parameter estimation, since in the cost function of the optimization its numerical derivative v_θ is used, which will amplify the noise. Thus additive zero-mean white Gaussian noise with a standard deviation of $\sigma = 0.5°$ is added to the tracking system measurements during the generation of the test data sets.

In Figure 7.3 and Figure 7.4 the results for the estimated parameters ${}_To$ and ${}_Tm$ are

Algorithm 3 Cost Function

```
Input:
    $_Tm$, $_To$    ▷ Clock parameter
    $B_G = \{D_G(1), D_G(2), \ldots, D_G(N_{BG})\}$    ▷ Buffer with gyro data sets
    $B_T = \{D_T(1), D_T(2), \ldots, D_T(N_{BT})\}$    ▷ Buffer with tracking data sets
Output:
    $J$    ▷ Cost function value
 1: function CALCULATEJ($_Tm$, $_To$, $B_G$, $B_T$)
    ▷ Calculate local time of gyro data
 2:     for $i = 1$ to length($B_G$) do
 3:         $t_G(i) = {}_Rm \cdot B_G(i).{}_R\tilde{k}_G$
 4:     end for
        ▷ Calculate local time of tracking data
 5:     for $i = 1$ to length($B_T$) do
 6:         $t_T(i) = {}_Tm \cdot (B_T(i).{}_T\tilde{k}_T - {}_To)$
 7:     end for
        ▷ Interpolate tracking heading rate on gyro time points
 8:     $\bar{v}_{\theta,T}$ = interpolate( $B_T(:).\tilde{v}_{\theta,T}$,$t_T$,$t_G$)
        ▷ Calculate cost function (mean error)
 9:     $J = 0, j = 1$
10:     for $i = 1$ to length($B_G$) do
        ▷ Check $\bar{v}_{\theta,T}$ is valid (no extrapolation)
11:         if $\bar{v}_{\theta,T}(i)$ is valid then
            ▷ Calculate error between tracking and gyro heading rate
12:             $\epsilon_v = B_G(i).\tilde{v}_{\theta,G} - \bar{v}_{\theta,T}(i)$
                ▷ Add absolute error of current point to cost function
13:             $J = \frac{j-1}{j} \cdot J + \frac{1}{j} \cdot \text{abs}(\epsilon_v)$
                ▷ Increment mean value counter
14:             $j = j + 1$
15:         end if
16:     end for
        ▷ Add penalty to cost function for transmission delays below the time of one
    sample step of the robot
17:     for $i = 1$ to length($B_T$) do
18:         $\Delta_t = {}_Rm \cdot B_T(i).{}_R\tilde{k}_R - t_T$
19:         if $\Delta_t < {}_Rm$ then
20:             $J = J + \alpha \cdot ({}_Rm - \Delta_t)$
21:         end if
22:     end for
        ▷ Return cost function value
23: return $J$
24: end function
```

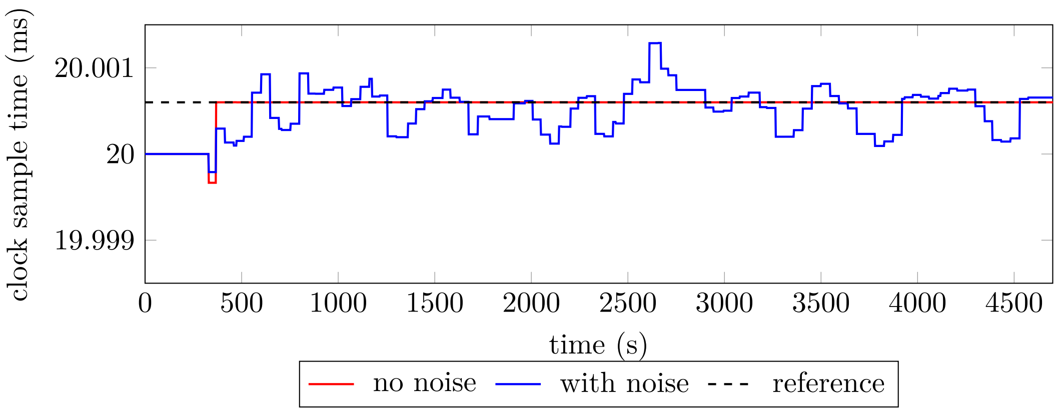


Figure 7.3: Estimated clock sample time $_Tm$ (simulation)

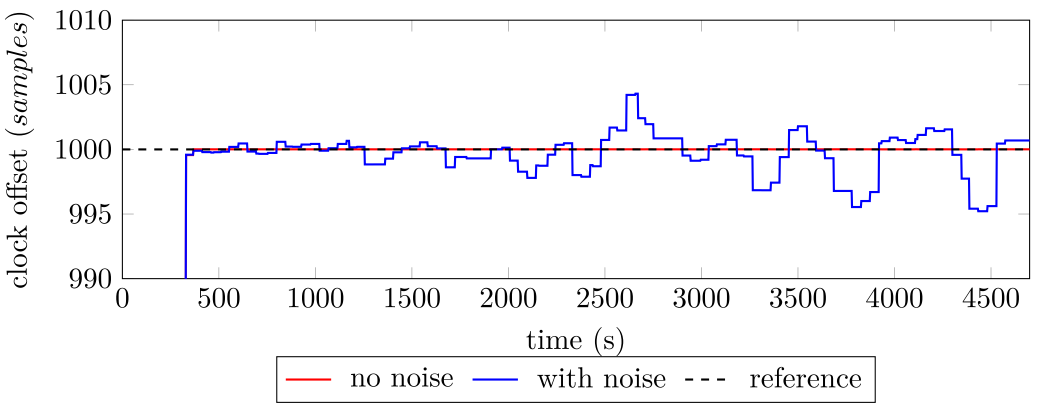


Figure 7.4: Estimated clock time offset $_To$ (simulation)

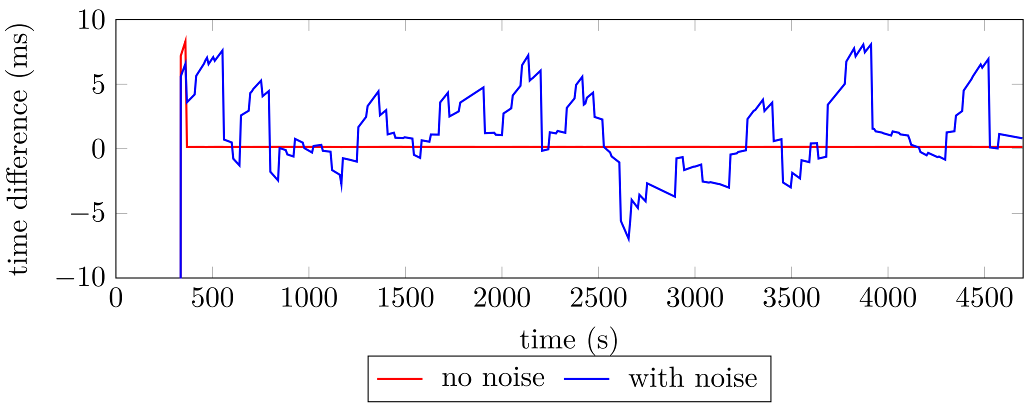


Figure 7.5: Clock time difference (simulation)

plotted. In the first test case without noise the method delivers the accurate clock parameter estimation with negligible small numerical errors after the second run of the optimization. This leads to a perfect time estimation without any time difference, as could be seen in Figure 7.5.

The simulation results with noisy measurements are plotted in blue in Figure 7.3, Figure 7.4 and Figure 7.5. Besides the fact, that the noise clearly affects the accuracy of the results of the clock parameter estimation, the maximum error in time stays below 10 ms after the first optimization run. Since the clock offset parameter ${}_{T}o$ has an increased sensitivity to the accuracy of the estimated clock sample time ${}_{T}m$, its difference to the real parameter increases over time. This increasing mismatch of the offset over time can be seen in Figure 7.4. Nevertheless, the simulation results indicate that the proposed method will be able to give a valid clock time with a small error if optimizations are performed repeatedly in time.

7.6.2 Experiments

To evaluate the proposed clock parameter estimation method online on a mobile robot with real measurement data, an experiment with the TWIP and the tracking system, as introduced, was set up.

During the experiment of over 4700 s the TWIP follows a repeated trajectory covering an area of 2 m × 4 m. For validation purposes, the clock parameters are also obtained offline by an optimization that uses the recorded data of the complete experiment in one optimization horizon. An average delay of 64 ms and a maximum delay of 161 ms between the time instant when the images are gathered by the tracking system and time instant the data sets are received by the robot has been calculated offline with the recorded data from the experiments.

During the experiment, the maximum computation time required by all tasks (communication, controller, observer and clock parameter estimation) has been 3.6 ms. Herein the evaluation of the clock parameter estimation took approximately 2.6 ms. To conclude, 28 % of the 5 ms sample time of the robot is still available for further tasks.

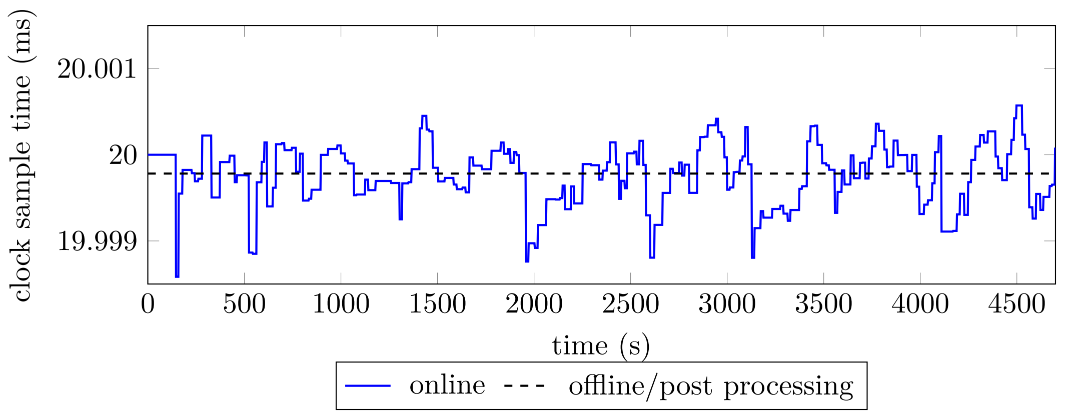


Figure 7.6: Estimated clock sample time ${}_{T}m$ (experiment)

In Figure 7.6 and Figure 7.7 the results of the estimated clock parameters during the experiments are shown. The online estimated clock sample time shows a maximum

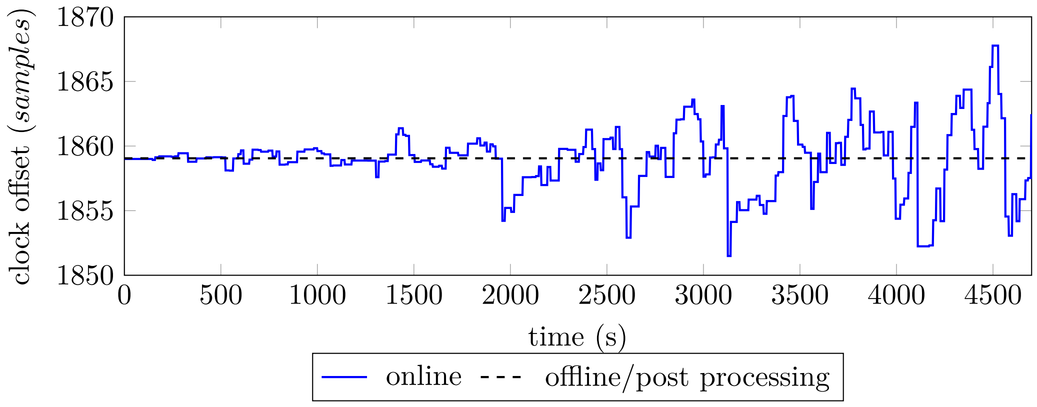


Figure 7.7: Estimated clock time offset ${}_To$ (experiment)

absolute difference from the offline computed sample time ${}_Tm$ of 1.4 µs. Furthermore, the difference between the time calculated online and offline does not increase, as can be seen in Figure 7.8. After the first optimization the maximum absolute error in time stays below 8 ms which corresponds to a delay of less than two sample steps of the microcontroller.

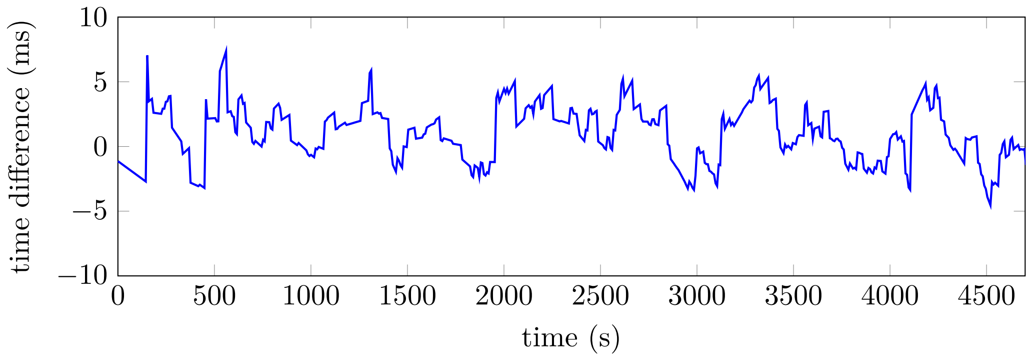


Figure 7.8: Clock time difference (experiment)

One comment to the clock offset ${}_To$ in Figure 7.7 which grows over time: this is not an issue. Therefore, the maximum absolute error in time in Figure 7.8 does not grow over time. In (6.1) the ${}_To$ is multiplied by the clock sample time ${}_Tm$. In consequence, the larger the time t gets as time progresses the larger ${}_To$ changes due a small deviation of ${}_Tm$ from its real value. Thus, this is an expected behavior.

With fixed clock parameters instead of online estimated ones also an error in time has to be considered after 4700 s. This error comes from two sources: on the one hand the deviation of the robot's oscillator frequency from its nominal value, and on the other hand, a possibly wrong initialization of the clock offset parameter. As introduced in Subsection 2.3.2, the robot's oscillator frequency deviation may introduce an error in time of 108 ms/h and thus an error of up to 141 ms at end of the experiment. An additional initialization error is calculated and added based on the assumption, that the clock offset parameter has been defined by the first package received by the robot. Furthermore, it is assumed that this package delay matches exactly the average delay of 64 ms. Thus, with fixed clock parameters an error of up to 205 ms has to be assumed.

However, if the package delay has the maximum delay of 161 ms observed during the experiment, the error in time may grow up to 302 ms. Finally, it has to be mentioned, that the tracking system's clock is not completely accurate either. Since no technical details about the clock parameters of the tracking system's oscillator are available, these deviations are also not considered time error estimate. The presented results highlight, that the method can be used to get an online estimation of the clock parameters, which reduces the maximum error of the calculated time drastically. In the experiment over 4700 s the error in time, compared to the error using fixed clock parameters, is less than 3.9 %.

7.7 Concluding Remarks

In this chapter, a novel method to estimate clock parameters online on small robots with microcontrollers, limited memory, and low-power requirements has been presented. First, the clock parameter estimation problem has been motivated, and afterwards the concept to overcome the issue has been explained. Further implementation details are given to ease the implementation of the algorithm if required. Simulation results as well as experimental results with the TWIP are presented showing the applicability of the proposed method. Finally, an experiment with a duration of 4700 s has been conducted and presented to pove the functionality of the proposed clock parameter estimation algorithm. The experimental results show a reduction of the error in time estimation of up to 96.1 % compared to an approach with fixed clock parameters.

Chapter 8

Summary of Achievements and Outlook

To finalize this thesis, a summary of achievements is presented to give a compact overview of the key insights gained and the major contributions in the field of design, modeling, control as well as state and parameter estimation applied to the two-wheeled inverted pendulum (TWIP). In addition, a brief outlook is given to motivate further research in this challenging area.

8.1 Summary of Achievements

TWIPs offer a new way of personal transport in urban areas, for the recording of movies, filming sport-events, and also for indoor logistics. Especially for the last two tasks, where autonomous driving might be desired, stable and precise setpoint and tracking control for TWIPs is essential. Besides this, small TWIPs offer a challenging benchmark platform to test new control algorithms for researchers. In this thesis, a small TWIP has been built, a comprehensive dynamic model has been derived and control, as well as state and parameter estimation algorithms, have been developed and applied to the mobile robot. In each of the given tasks, contributions were presented which bring the ongoing research in this field forward.

Two-Wheeled Inverted Pendulum In Chapter 2, the concept and building process of the TWIP has been presented with a strong focus on the desired real-time closed-loop control application. This led to a robust and lightweight robot, where an appropriate choice of sensors delivers high-quality measurement signals required for precise state estimation. In addition, the bare-metal firmware on the MCU ensures that all algorithms can be executed in real-time. The contributed insights and details of the design process and the resulting experimental setup enables other researchers to improve their setups and to receive better results.

Modeling A comprehensive model of the TWIP has been presented in Chapter 3. Compared to existing literature the dynamics of the motor currents are included directly in the Lagrangian and are not neglected. Therefore, the proposed novel approach is less error-prone and a more well-rounded modeling procedure, as the electrical and mechanical systems, are modeled jointly inside the Lagrangian framework. In addition, the physical current limits can easily be incorporated into the model derived and it offers the possibility to consider them as state limits during control design and the estimation of the DoA.

In addition, an insight into the system properties of the presented models with different levels of detail is given in Chapter 3, namely one model with current dynamics as well as one without current dynamics for the balancing *two-wheeled inverted pendulum mode* and a non-balancing model of the TWIP on ground for the *wheelchair mode*. Utilizing an exemplary trajectory, the interconnection and power exchange of the different energy-storages of the system, included in the Lagrangian, is revealed. Based on this, negligible terms are detected as well as parts of the model which require particular attention during modeling. To the best of the author's knowledge, such a detailed and comprehensive treatment has not been presented before for TWIPs.

Moreover, in the awareness of non-ideal sensors, a model for each of the three onboard sensors is presented, including sampling, quantization, and the digital on-chip filters to provide the required congruence to the physical system for simulation-based observer tuning. Even though the sensors show non-negligible dynamics due to their filters in the presented application, such a treatment of the sensors has not been presented before for the TWIP.

Furthermore, the presented experimental results confirm a high degree of congruence of the model with the physical TWIP. In particular, the closed-loop response to a position step in the experiments show a remarkable match to the simulation results. Based on the proper and control-orientated system design presented in Chapter 2, coupled with the thorough modeling of the TWIP introduced in Chapter 3, an excellent mapping between the physical system behavior and the simulation is achieved. Compared to other published results, this congruence is outstanding.

Finally, a versatile treatment of the linearized models has been provided. State transformations are introduced to decouple the dynamics into two subsystems, one for the heading dynamics and one for the forward dynamics of the robot. Moreover, the eigenvalues of the LTI-models as well as the poles and zeros of the subsystem's transfer functions are presented and an interpretation of physical system properties is given. Furthermore, the connection of the choice of the system output to performance limitations has been analyzed thoroughly. For the first time, such a comprehensive treatment has been published which offers novel insights for modeling and control of the TWIP.

Optimal Trajectory Generation In Chapter 4, a method has been presented to compute an energy optimal trajectory offline, subject to the nonlinear TWIP model with current dynamics and state and input constraints. The introduced optimization problem minimizes the input energy of the TWIP and not only the control input. By the use of checkpoints and so-called equalitypoints, the desired trajectory can be specified. Moreover, an optimized reference trajectory that can be cut and repeated for long-term test runs is presented. A major contribution is the introduced two-stage optimization scheme using the simpler model on ground in the wheel-chair mode to compute an initial guess for the optimization task using the full model of the balancing TWIP with current dynamics. Moreover, additional analyses are presented to compare the standard Runge-Kutta (RK) against variational integrator (VarInt) schemes for discretization. The presented results coincide with Albert et al. [2] but in this chapter, it has been revealed that VarInt outperforms the standard schemes even more than has been presented by Albert et al. [2].

Setpoint and Trajectory Tracking A novel control structure to perform setpoint and trajectory tracking in the presence of large control errors has been presented in Chapter 5. A major contribution is the derivation of a full discrete-time treatment of all elements of the control structure. Moreover, the presented guidance algorithm makes it possible that a desired setpoint or trajectory can be reached even though the TWIP underlies nonholonomic constraints. In addition, a friction compensation is proposed to reduce the effects of nonlinear mechanical friction and to increase the congruence of the LTI-model used for control synthesis.

For stabilization and tracking, two discrete-time linear-quadratic state-feedback regulators with output weighting (LQRYs) have been designed. This is done by the use of decoupled linear state space models for the forward dynamics and heading dynamics, which eases the control synthesis, as only two scalars have to be tuned in simulations and experiments. In addition, the approach offers a clear relation between the parameters to tune and the effect on the error compensation related to the longitudinal and heading motions of the robot.

In the next step, the limited domain of attraction (DoA) of the closed-loop system is estimated by the use of a quadratic Lyapunov function (QLF). Using linear matrix inequalities (LMIs) the matrix P for the quadratic Lyapunov function (QLF) can be efficiently calculated by solving a convex optimization. Moreover, by the introduced procedure the limiting level set of the QLF is known which defines the estimated domain of attraction (EDoA) limited by all state and input constraints. This approach is easy to implement, can be efficiently solved, and gives an EDoA which can be used by a command governor, as introduced.

Based on the calculated EDoA, a command governor algorithm is introduced to extend the DoA of the closed-loop controller of the TWIP. In addition, to enhance the performance of the discrete-time algorithm for trajectory tracking, a static and dynamic allocation of the input and state limits between the desired trajectory and the stabilizing controller are presented, which are based on the continuous-time algorithm proposed by Diepold [31]. The experiments presented for setpoint and trajectory tracking prove the applicability of the algorithms in practice.

The problems arising from nonholonomic constraints stay for the TWIP, thus stability can only be guaranteed for the linear system but not for the nonlinear system with the guidance algorithm. This problem is an ongoing research topic for trajectory tracking and deserves the attention of future research. Nevertheless, the presented experimental results highlight that in practice good results for setpoint tracking as well as trajectory tracking can be achieved. In particular, large setpoint changes and trajectory offsets could be handled without the loss of stability in experiments.

To sum up, the proposed new control algorithms showed remarkable performance, even in presence of disturbances, measurement noise, and state estimation errors which are present in every real system.

State Estimation with Time Delayed Measurements An innovative state estimation method for the TWIP has been presented in Chapter 6 which is able to incorporate delayed remote measurements properly. In particular, the introduced algorithm requires low computational power and little memory requirements compared to existing methods.

Moreover, a statistical analysis of the non-constant transmission delays has been performed based on experimental data. Most methods in the literature rely on the time-independence of the random process of the delays to yield a good estimation performance, but it turned out that the assumption of time-independence of the random process does not hold. Therefore, the presented novel method uses the timestamps included in the delayed measurements and incorporates them during the measurement update, such that no assumptions on the random process of the delays are required to design the state estimator. Thus, the deterioration of the estimated state due to the non-constant time-delays of the transmitted remote measurements is minimized. In addition, the underlying nonholonomic constraints of the motion of the TWIP are used by the presented algorithm to improve the estimator's performance. Finally, the applicability and remarkable state estimation performance, even in presence of large measurement delays, has been demonstrated in experiments.

Efficient Clock Parameter Estimation In the last Chapter 7, a new algorithm to estimate clock parameters online on a low-power microcontroller with limited memory has been presented. Simulation results show the applicability of the proposed algorithm. In addition, experimental results are presented to prove the functionality of this novel concept to estimate the clock parameters for the proposed state estimation algorithm.

8.2 Outlook

To support and inspire further design and research in this innovative area, ideas and open topics are listed in this last section.

Firstly, the presented platform could be continued as a public open-source project and might be used as a benchmark platform by different research groups. This would ease the comparison of the gained results. Therefore, some parts as the gears might be redesigned to use out-of-the-shelf parts instead of customized ones. The same applies to the PCB and the electronics. Due to the rapid development, the MCU as well as the sensors are already outdated and a redesign with new components would be required. In addition, a more precise mounting for tracking markers should be realized to reduce the misfit between the tracking system and the model coordinate systems, which currently probably introduces some avoidable state estimation errors.

In this thesis, the focus has been laid on nonlinear optimal trajectory generation but other approaches in the literature offer alternatives requiring less computational power. The use of splines to optimize the path first in combination with the use of a flat output, as presented in Subsection 3.3.3, is promising to speed up the generation of an initial guess for the consecutive nonlinear optimization with the model of the balancing TWIP with current dynamics. A quite similar approach has been presented by Diepold et al. [30] for a ball-balancing robot.

In the field of control, a better solution for the guidance algorithm to overcome the stumbling blocks placed by the nonholonomic constraints of the dynamic system will improve the closed-loop performance. Ideally, in the same manner, as Astolfi [6] presented a solution for stable setpoint tracking, a solution for trajectory tracking might be developed. In addition, an investigation into a new method to estimate the DoA with the full nonlinear model (with x_{B} and y_{B}) will increase the trust in the stability region found. Takagi-Sugeno-Models might improve and increase the estimated DoA as proposed by Diepold and Albert [29]. Moreover, the ω-representations of the state space model, as presented in Subsection 3.1.10, may ease the design of nonlinear control laws. As discussed, the model has an integration chain $u_{\mathrm{R}} \rightarrow i_{\mathrm{R}} \rightarrow \omega_{\delta\mathrm{R}}$ which is (almost) decoupled from the chain $u_{\mathrm{L}} \rightarrow i_{\mathrm{L}} \rightarrow \omega_{\delta\mathrm{L}}$. Utilizing this property, new opportunities for control may arise in the application of partial feedback linearization or backstepping.

In the field of state estimation, an estimation error reduction during high velocities might be gained by a new approach to handle delayed measurements. As typically the time instances of the sample steps of the robot and the tracking system do not match exactly, the measurements are currently used in the next or previous sample step of the robot. The faster the robot moves, the larger the resulting error in the measurement update will be. This error might be reduced by interpolating the received measurements and using the interpolated value at the right time instance. Moreover, if a more powerful MCU is available, a comparison of the presented estimation approach with an EKF and SPKF will be interesting.

Finally, the clock parameter estimation suffers from the change of sensitivity of the two parameters over the run-time. Maybe, an approach can be found to reduce or even eliminate this effect.

Appendices

Appendix A

Modeling

A.1 Matrices Model with Current Dynamics (c-d)

$$A_{c\text{-}d} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6.440 \cdot 10^{1} & -6.346 \cdot 10^{-1} & 1.923 \cdot 10^{1} & 0 & -3.916 \cdot 10^{1} & -3.916 \cdot 10^{1} \\ 0 & 0 & 1.196 & 4.087 \cdot 10^{-2} & -1.238 & 0 & 2.522 & 2.522 \\ 0 & 0 & 0 & 0 & 0 & -1.841 & 7.648 \cdot 10^{1} & -7.648 \cdot 10^{1} \\ 0 & 0 & 0 & 4.726 \cdot 10^{2} & -1.432 \cdot 10^{4} & -7.018 \cdot 10^{2} & -3.750 \cdot 10^{3} & 0 \\ 0 & 0 & 0 & 4.726 \cdot 10^{2} & -1.432 \cdot 10^{4} & 7.018 \cdot 10^{2} & 0 & -3.750 \cdot 10^{3} \end{pmatrix} \tag{A.1}$$

$$B_{c\text{-}d} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 2.500 \cdot 10^{3} & 0 \\ 0 & 2.500 \cdot 10^{3} \end{pmatrix} \tag{A.2}$$

A.2 Matrices Model with Current Dynamics (n-d)

$$A_{n\text{-}d} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 6.440 \cdot 10^{1} & -1.051 \cdot 10^{1} & 3.183 \cdot 10^{2} & 0 \\ 0 & 0 & 1.196 & 6.765 \cdot 10^{-1} & -2.050 \cdot 10^{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & -3.047 \cdot 10^{1} \end{pmatrix} \tag{A.3}$$

$$B_{n\text{-}d} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -2.610 \cdot 10^{1} & -2.610 \cdot 10^{1} \\ 1.681 & 1.681 \\ 5.099 \cdot 10^{1} & -5.099 \cdot 10^{1} \end{pmatrix} \tag{A.4}$$

A.3 Matrices Model on Ground (g-d)

$$A_{g\text{-}d} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -2.641 \cdot 10^{1} & 0 \\ 0 & 0 & 0 & -2.345 \cdot 10^{1} \end{pmatrix} \tag{A.5}$$

$$B_{g\text{-}d} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 2.166 & 2.166 \\ 3.925 \cdot 10^{1} & -3.925 \cdot 10^{1} \end{pmatrix} \tag{A.6}$$

A.4 Decoupling Matrices

Input transformation

$$T_u = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{A.7}$$

State transformation model with current

$$T_{c\text{-}d} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \tag{A.8}$$

State transformation model without current

$$T_{n,d} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.9}$$

State transformation model on ground without current

$$T_{g,d} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.10}$$

A.5 Matrices Decoupled Model with Current Dynamics

Forward dynamics

$$A_{c\text{-}Fw} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1.196 & -1.238 & 4.087 \cdot 10^{-2} & 2.522 \\ 0 & 6.440 \cdot 10^{1} & 1.923 \cdot 10^{1} & -6.346 \cdot 10^{-1} & -3.916 \cdot 10^{1} \\ 0 & 0 & -2.864 \cdot 10^{4} & 9.453 \cdot 10^{2} & -3.750 \cdot 10^{3} \end{pmatrix} \tag{A.11}$$

$$b_{c\text{-}Fw} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2.500 \cdot 10^{3} \end{pmatrix} \tag{A.12}$$

Heading dynamics

$$A_{c\text{-}He} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -1.841 & 7.648 \cdot 10^{1} \\ 0 & -1.404 \cdot 10^{3} & -3.750 \cdot 10^{3} \end{pmatrix} \tag{A.13}$$

$$b_{c\text{-}He} = \begin{pmatrix} 0 \\ 0 \\ 2.500 \cdot 10^{3} \end{pmatrix} \tag{A.14}$$

A.6 Matrices Decoupled Model without Current Dynamics

Forward dynamics

$$A_{n\text{-}Fw} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1.196 & -2.050 \cdot 10^{1} & 6.765 \cdot 10^{-1} \\ 0 & 6.440 \cdot 10^{1} & 3.183 \cdot 10^{2} & -1.051 \cdot 10^{1} \end{pmatrix} \tag{A.15}$$

$$b_{n\text{-}Fw} = \begin{pmatrix} 0 \\ 0 \\ 1.681 \\ -2.610 \cdot 10^{1} \end{pmatrix} \tag{A.16}$$

Heading dynamics

$$A_{n\text{-}H} = \begin{pmatrix} 0 & 1 \\ 0 & -3.047 \cdot 10^{1} \end{pmatrix} \tag{A.17}$$

$$b_{n\text{-}He} = \begin{pmatrix} 0 \\ 5.099 \cdot 10^{1} \end{pmatrix} \tag{A.18}$$

A.7 Matrices Decoupled Model on Ground without Current Dynamics

Forward dynamics

$$A_{g\text{-}Fw} = \begin{pmatrix} 0 & 1 \\ 0 & -2.641 \cdot 10^1 \end{pmatrix} \tag{A.19}$$

$$b_{g\text{-}Fw} = \begin{pmatrix} 0 \\ 2.166 \end{pmatrix} \tag{A.20}$$

Heading dynamics

$$A_{g\text{-}He} = \begin{pmatrix} 0 & 1 \\ 0 & -2.345 \cdot 10^1 \end{pmatrix} \tag{A.21}$$

$$b_{g\text{-}He} = \begin{pmatrix} 0 \\ 3.925 \cdot 10^1 \end{pmatrix} \tag{A.22}$$

A.8 Model Parameters

Table A.1: TWIP model parameters

Symbol	Value	Description
g	$9.81\,\mathrm{m/s^2}$	gravity constant
m_B	$280 \cdot 10^{-3}\,\mathrm{kg}$	body mass
l_B	$49.467 \cdot 10^{-3}\,\mathrm{m}$	distance to the body's center of gravity
I_B		body inertia
I_Bxx	$567.7 \cdot 10^{-6}\,\mathrm{kg\,m^2}$	about the x-axis
I_Byy	$496.7 \cdot 10^{-6}\,\mathrm{kg\,m^2}$	about the y-axis
I_Bzz	$155.2 \cdot 10^{-6}\,\mathrm{kg\,m^2}$	about the z-axis
m_W	$28 \cdot 10^{-3}\,\mathrm{kg}$	wheel mass
l_W	$37.396 \cdot 10^{-3}\,\mathrm{m}$	distance to the wheel's center of gravity
I_W		wheel inertia
I_Wxx	$4.998 \cdot 10^{-6}\,\mathrm{kg\,m^2}$	about the x-axis
I_Wyy	$7.404 \cdot 10^{-6}\,\mathrm{kg\,m^2}$	about the y-axis
I_Wzz	$4.998 \cdot 10^{-6}\,\mathrm{kg\,m^2}$	about the z-axis
I_M	$268.528 \cdot 10^{-9}\,\mathrm{kg\,m^2}$	inertia motor shaft
I_G	$1.807 \cdot 10^{-6}\,\mathrm{kg\,m^2}$	inertia gear stage
n_WM	$(78/11)^2\,-$	gear ratio wheel to motor
n_WG	$78/11\,-$	gear ratio wheel to gear
k_E	$3.76 \cdot 10^{-3}\,\mathrm{V/(rads)}$	motor back-EMF constant
k_M	$3.76 \cdot 10^{-3}\,\mathrm{N\,m/A}$	motor torque constant
L_M	$4 \cdot 10^{-4}\,\mathrm{H}$	motor inductance
R_M	$1.5\,\Omega$	motor resistance
r_W	$33 \cdot 10^{-3}\,\mathrm{m}$	wheel radius
$2d_\mathrm{W}$	$2 \times 49 \cdot 10^{-3}\,\mathrm{m}$	distance between wheels
d_V	$1.532 \cdot 10^{-3}\,\mathrm{N\,m/rad}$	viscous damping coefficient
d_C	$32.6 \cdot 10^{-3}\,\mathrm{N\,m}$	coulomb damping coefficient
d_0	$8\,-$	slope of the damping curve

Appendix B

Control

B.1 Matrices Decoupled Discrete-Time Model with Current Dynamics

Forward dynamics

$$A_{Fw}^{\circlearrowright} = \begin{pmatrix} 1 & 1.529 \cdot 10^{-5} & 4.778 \cdot 10^{-3} & 7.363 \cdot 10^{-6} & 2.980 \cdot 10^{-6} \\ 0 & 1.001 & 3.454 \cdot 10^{-3} & 4.887 \cdot 10^{-3} & -4.629 \cdot 10^{-5} \\ 0 & 6.190 \cdot 10^{-3} & 9.089 \cdot 10^{-1} & 3.023 \cdot 10^{-3} & 5.845 \cdot 10^{-4} \\ 0 & 3.189 \cdot 10^{-1} & 1.416 & 9.541 \cdot 10^{-1} & -9.085 \cdot 10^{-3} \\ 0 & 3.145 \cdot 10^{-2} & -6.641 & 2.192 \cdot 10^{-1} & -6.811 \cdot 10^{-3} \end{pmatrix} \tag{B.1}$$

$$B_{Fw}^{\circlearrowright} = \begin{pmatrix} 1.811 \cdot 10^{-5} \\ -2.813 \cdot 10^{-4} \\ 7.450 \cdot 10^{-3} \\ -1.157 \cdot 10^{-1} \\ 5.851 \cdot 10^{-1} \end{pmatrix} \tag{B.2}$$

Heading dynamics

$$A_{He}^{\circlearrowright} = \begin{pmatrix} 1 & 4.669 \cdot 10^{-3} & 9.049 \cdot 10^{-5} \\ 0 & 8.644 \cdot 10^{-1} & 1.778 \cdot 10^{-2} \\ 0 & -3.262 \cdot 10^{-1} & -6.708 \cdot 10^{-3} \end{pmatrix} \tag{B.3}$$

$$B_{He}^{\circlearrowright} = \begin{pmatrix} 5.499 \cdot 10^{-4} \\ 2.262 \cdot 10^{-1} \\ 5.865 \cdot 10^{-1} \end{pmatrix} \tag{B.4}$$

B.2 LQR Feedback Gain

$$K = \begin{pmatrix} -2.616 \cdot 10^{1} & -2.616 \cdot 10^{1} \\ 5.881 & -5.881 \\ -1.548 \cdot 10^{1} & -1.548 \cdot 10^{1} \\ -1.542 & -1.542 \\ -1.859 \cdot 10^{1} & -1.859 \cdot 10^{1} \\ 1.560 \cdot 10^{-1} & -1.560 \cdot 10^{-1} \\ 6.762 \cdot 10^{-3} & 4.122 \cdot 10^{-4} \\ 4.122 \cdot 10^{-4} & 6.762 \cdot 10^{-3} \end{pmatrix}^{\mathrm{T}} \tag{B.5}$$

B.3 LQR Error Plots

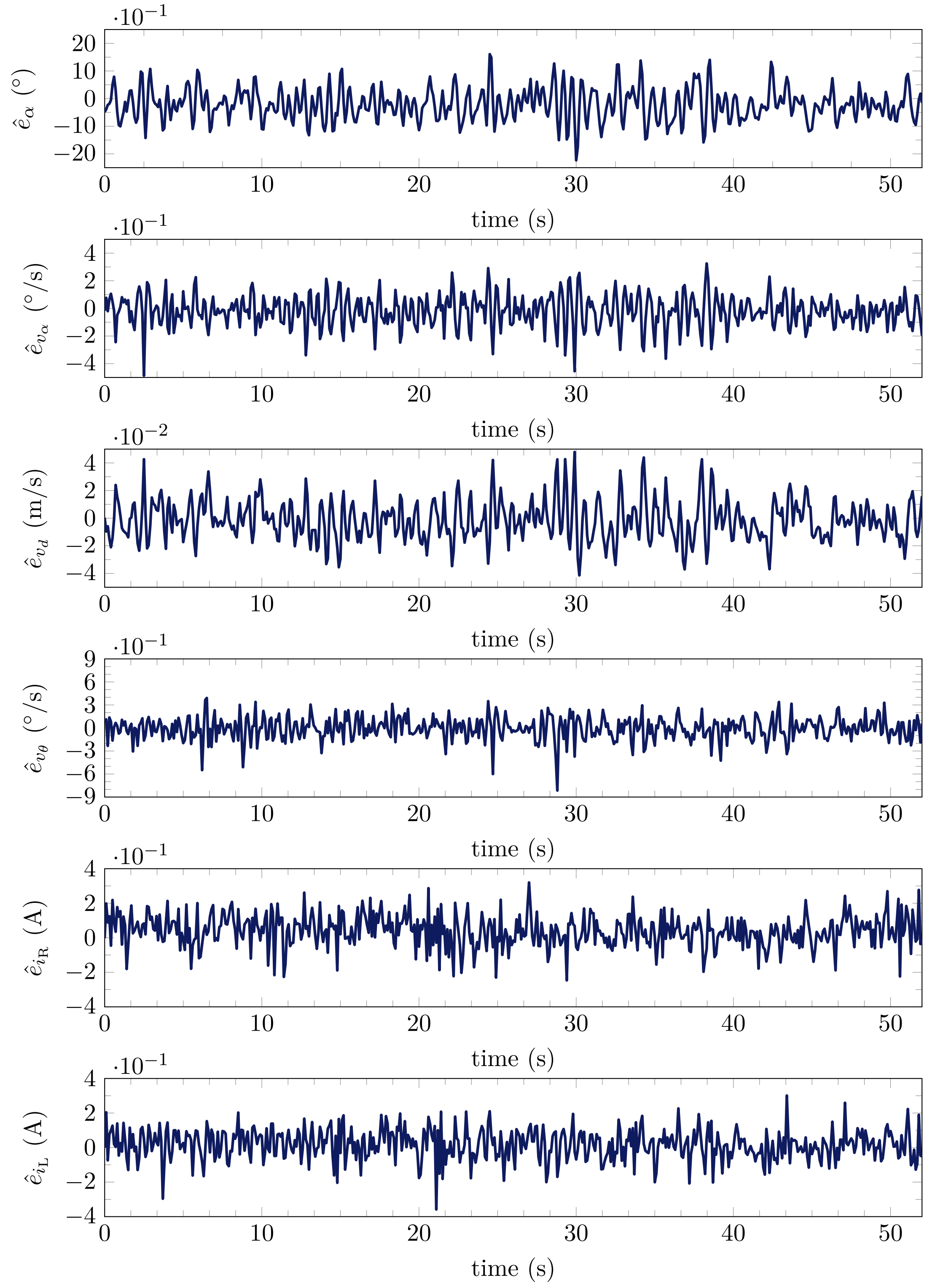


Figure B.1: Experimental control errors (e_α,e_{v_α},e_{v_d},e_{v_θ},e_{i_R},e_{i_L})

Appendix C

State Estimation

C.1 Local Model

$$A_l^{\circlearrowright} = \begin{pmatrix} 1.0 \cdot 10^{0} & 4.9 \cdot 10^{-3} & 3.5 \cdot 10^{-3} & -3.4 \cdot 10^{-20} & -4.6 \cdot 10^{-5} & -4.6 \cdot 10^{-5} \\ 3.2 \cdot 10^{-1} & 9.5 \cdot 10^{-1} & 1.4 \cdot 10^{0} & -1.5 \cdot 10^{-17} & -9.1 \cdot 10^{-3} & -9.1 \cdot 10^{-3} \\ 6.2 \cdot 10^{-3} & 3.0 \cdot 10^{-3} & 9.1 \cdot 10^{-1} & 7.8 \cdot 10^{-19} & 5.8 \cdot 10^{-4} & 5.8 \cdot 10^{-4} \\ -7.4 \cdot 10^{-19} & -8.7 \cdot 10^{-18} & 2.7 \cdot 10^{-16} & 8.6 \cdot 10^{-1} & 1.8 \cdot 10^{-2} & -1.8 \cdot 10^{-2} \\ 1.6 \cdot 10^{-2} & 1.1 \cdot 10^{-1} & -3.3 \cdot 10^{0} & -1.6 \cdot 10^{-1} & -6.8 \cdot 10^{-3} & -5.2 \cdot 10^{-5} \\ 1.6 \cdot 10^{-2} & 1.1 \cdot 10^{-1} & -3.3 \cdot 10^{0} & 1.6 \cdot 10^{-1} & -5.2 \cdot 10^{-5} & -6.8 \cdot 10^{-3} \end{pmatrix} \tag{C.1}$$

$$B_l^{\circlearrowright} = \begin{pmatrix} -2.8 \cdot 10^{-4} & -2.8 \cdot 10^{-4} \\ -1.2 \cdot 10^{-1} & -1.2 \cdot 10^{-1} \\ 7.5 \cdot 10^{-3} & 7.5 \cdot 10^{-3} \\ 2.3 \cdot 10^{-1} & -2.3 \cdot 10^{-1} \\ 5.9 \cdot 10^{-1} & -6.7 \cdot 10^{-4} \\ -6.7 \cdot 10^{-4} & 5.9 \cdot 10^{-1} \end{pmatrix} \tag{C.2}$$

C.2 Measurement Parameter

Table C.1: State estimator measurement matrix parameters

Symbol	Value
$c_{T,xy}$	$1.00 \cdot 10^3$
$c_{T,\theta}$	$1.00 \cdot 10^0$
$c_{E,\alpha}$	$-5.73 \cdot 10^0$
c_{E,v_d}	$1.74 \cdot 10^2$
c_G	$9.40 \cdot 10^2$
c_S	$2.56 \cdot 10^2$

C.3 Cost Function

Table C.2: State estimator cost function parameters

Symbol	Value	Symbol	Value
$q_{l,\alpha}$	1	$r_{l,E\delta}$	$3.4009 \cdot 10^{5}$
q_{l,v_α}	$8.6756 \cdot 10^{-28}$	$r_{l,Gx}$	$8.3330 \cdot 10^{3}$
$q_{l,\omega}$	$1.4464 \cdot 10^{-17}$	$r_{l,Gy}$	$4.3985 \cdot 10^{-27}$
$q_{l,i}$	$5.0793 \cdot 10^{12}$	$r_{l,Sz}$	$6.5133 \cdot 10^{13}$
$q_{u,xy}$	1	$r_{u,xy}$	$9.4493 \cdot 10^{10}$
q_{u,v_d}	$1.3116 \cdot 10^{3}$		
$q_{u,\theta}$	1	$r_{u,\theta}$	2.2439

C.4 Optimization

$$\mathcal{Q}_{opt} = \operatorname{diag}\begin{pmatrix} 1.000 \cdot 10^{2} \\ 1.000 \cdot 10^{2} \\ 1.146 \cdot 10^{1} \\ 2.230 \cdot 10^{1} \\ 4.460 \\ 3.048 \\ 4.775 \cdot 10^{-1} \\ 5.081 \\ 5.081 \end{pmatrix} \tag{C.3}$$

Appendix D

Abbreviations and acronyms

back-EMF back electromotive force

BCE Before the Christian

CAD computer-aided design

CDF cumulative distribution function

CPE Clock Parameter Estimation

CPR counts per revolution

CPU central processing unit

CQLKF cascaded quasi-linear Kalman filter

DAL dynamic allocation of input and state limits

DARE discrete-time algebraic Riccati equation

DC direct current

DoA domain of attraction

DoF degree of freedom

DSM Downhill Simplex method

EDoA estimated domain of attraction

EKF extended Kalman filter

FIFO First-In-First-Out

FIR finite impulse response

FPU floating-point unit

I^{2}C inter-integrated circuit

IMU inertial measurement unit

ISE integral square error

LED light-emitting diode

LiPo lithium-ion polymer battery

LMI linear matrix inequality

LQG linear-quadratic-Gaussian

LQR linear-quadratic regulator

LQRY linear-quadratic state-feedback regulator with output weighting

LSB least significant bit

LTI linear time-invariant

MCU microcontroller unit

MEMS microelectromechanical systems

MHE moving horizon estimation

MIMO multiple-input-multiple-output

MOSFET metal–oxide–semiconductor field-effect transistor

NTP Network Time Protocol

ODE ordinary differential equation

OS operating system

PC personal computer

PCB printed circuit board

PDV packet delay variation

PID proportional-integral-derivative

PWM pulse-width modulation

QDEC quadrature decoder

QLF quadratic Lyapunov function

RAM random access memory

RHP right half-plane

RISC reduced instruction set computer

RK Runge-Kutta

RMS root mean square

SAL static allocation of input and state limits

SDP	semidefinite programming
SG-Filter	Savitzky-Golay-Filter
SISO	single-input-single-output
SLS	selective laser sintering
SPI	serial peripheral interface
SPKF	Sigma-point Kalman Filters
SPP	serial port profile
SRAM	static random-access memory
TUM	Technical University of Munich
TWIP	two-wheeled inverted pendulum
VarInt	variational integrator

List of Figures

List of Tables

References

[1] ABC News. Segway fails: Usain bolt, ian healy, george bush and ellen among the victims of unfortunate incidents, Aug 2015. URL `http://content.time.com/time/business/article/0,8599,186660-5,00.html`.

[2] Klaus Albert, Karmvir Singh Phogat, Felix Anhalt, Ravi N. Banavar, Debasish Chatterjee, and Boris Lohmann. Structure-preserving constrained optimal trajectory planning of a wheeled inverted pendulum. *IEEE Transactions on Robotics*, 36(3):910–923, 2020. doi: 10.1109/TRO.2020.2985579.

[3] Harold L. Alexander. State Estimation for Distributed Systems with Sensing Delay. In *Proceedings SPIE, Data Structures and Target Classification*, volume 1470, 1991. doi: 10.1117/12.44843.

[4] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi: 10.1007/s12532-018-0139-4.

[5] Felix Anhalt. Zeitdiskreter Regler- und Führungsfilterentwurf für ein zweirädriges inverses Pendel. Master's thesis, Technical University of Munich, 2016.

[6] A. Astolfi. On the stabilization of nonholonomic systems. In *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, volume 4, pages 3481–3486 vol.4, 1994. doi: 10.1109/CDC.1994.411685.

[7] Charles Audet and J. E. Dennis. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2002. doi: 10.1137/S1052623400378742.

[8] M. Bak, T.D. Larsen, M. Norgaard, N.A. Andersen, N.K. Poulsen, and O. Ravn. Location estimation using delayed measurements. In *AMC'98 - Coimbra. 1998 5th International Workshop on Advanced Motion Control. Proceedings (Cat. No.98TH8354)*, pages 180–185. IEEE, 1998. ISBN 0-7803-4484-7. doi: 10.1109/AMC.1998.743533.

[9] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004. doi: 10.1017/CBO9780511804441.

[10] Stephen P Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. SIAM, 1994.

[11] Dennis P.J. Brandt. Two-wheeled robot stabilization and motion control. Master's thesis, Eindhoven University of Technology, 2001.

[12] Michael Buhl. *Sättigende strukturvariable Regelungen.* Dissertation, Technische Universität München, München, 2008.

[13] Michael Buhl and Boris Lohmann. Lyapunov-funktionen-basiertes führungsfilter. *at - Automatisierungstechnik*, 57(10):499–504, 2009. doi: 10.1524/auto.2009.0797.

[14] Tim Bürchner. Entwurf, Implementierung und Evaluation von Beobachterkonzepten für einen mobilen Kleinroboter. Technical University of Munich, 2018. Term paper.

[15] Ronald Ping Man Chan, Karl A. Stol, and C. Roger Halkyard. Review of modelling and control of two-wheeled robots. *Annual Reviews in Control*, 37(1):89–103, 2013. ISSN 1367-5788. doi: 10.1016/j.arcontrol.2013.03.004.

[16] Jie Chen. Sensitivity integral relations and design trade-offs in linear multivariable feedback systems. *IEEE Transactions on Automatic Control*, 40(10):1700–1716, 1995. doi: 10.1109/9.467680.

[17] Jie Chen. Multivariable gain-phase and sensitivity integral relations and design trade-offs. *IEEE Transactions on Automatic Control*, 43(3):373–385, 1998. doi: 10.1109/9.661594.

[18] Jie Chen, Li Qiu, and O. Toker. Limitations on maximal tracking accuracy. 2. tracking sinusoidal and ramp signals. In *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)*, volume 3, pages 1757–1761 vol.3, 1997. doi: 10.1109/ACC.1997.610886.

[19] Jie Chen, Li Qiu, and O. Toker. Limitations on maximal tracking accuracy. *IEEE Transactions on Automatic Control*, 45(2):326–331, 2000. doi: 10.1109/9.839960.

[20] Howie M. Choset, editor. *Principles of robot motion.* Intelligent robotics and autonomous agents. A Bradford book. MIT Press, Cambridge, Mass. [u.a.], [3. print.] edition, 2005. ISBN 0262033275 - 9780262033275.

[21] Fuquan Dai, Xueshan Gao, Shigong Jiang, Wenzeng Guo, and Yubai Liu. A two-wheeled inverted pendulum robot with friction compensation. *Mechatronics*, 30: 116–125, 2015. ISSN 0957-4158. doi: doi.org/10.1016/j.mechatronics.2015.06.011.

[22] S. Delgado, S. Gajbhiye, and Banavar R.N. Reduced equations of motion for a wheeled inverted pendulum. In *Proceedings of the 8th Vienna International Conference on Mathematical Modelling (MATHMOD)*, Vienna, Mar 2015. doi: 10.1016/j.ifacol.2015.05.011.

[23] Sergio Delgado. *Total Energy Shaping for Underactuated Mechanical Systems: Dissipation and Nonholonomic Constraints.* PhD thesis, Technical University of Munich, 2016.

[24] C. Demichelis and P. Chimento. Ip packet delay variation metric for ip performance metrics (ippm). RFC 3393, RFC Editor, November 2002. `http://www.rfc-editor.org/rfc/rfc3393.txt`.

[25] Christian Dengler. *Design of Adaptive Nonlinear Controllers using Supervised Learning.* Dissertation, Technische Universität München, München, 2021.

[26] Christian Dengler and Boris Lohmann. Adjustable and adaptive control for an unstable mobile robot using imitation learning with trajectory optimization. *Robotics*, 9(2), 2020. ISSN 2218-6581. doi: 10.3390/robotics9020029.

[27] Ronnie Dessort. 3D-MKS-Modellierung und führungsgrößenadaptive, schaltende LQ-Regelung eines balancierenden Roboters. Technical University of Munich, 2011. Term paper.

[28] K.J. Diepold and S.J. Pieczona. Tracking control with adaptively allocated maximum input amplitudes and enlarged domain of attraction for linear systems. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 2090–2096, Dec 2013. doi: 10.1109/CDC.2013.6760190.

[29] Klaus J. Diepold and Klaus Albert. Lokale stabilitätsanalyse von takagi-sugeno systemen unter berücksichtigung ihres gültigkeitsbereichs. *at - Automatisierungstechnik*, 62(10):687–697, 2014. doi: doi:10.1515/auto-2014-1114.

[30] Klaus J. Diepold, André Albers, and Tobias Guggemos. Flachheitsbasierte trajektorienoptimierung entlang von wegpunkten für ein lineares ballbot-system. *at - Automatisierungstechnik*, 62(12):842–850, 2014. doi: doi:10.1515/auto-2014-1116.

[31] Klaus Jürgen Diepold. *Set Point and Trajectory Tracking of Constrained Systems in Takagi-Sugeno Form.* Dissertation, Technische Universität München, München, 2016.

[32] Djamel Djenouri and Miloud Bagaa. Synchronization protocols and implementation issues in wireless sensor networks: A review. *IEEE Systems Journal*, 10(2): 617–627, Jun. 2016. ISSN 1932-8184. doi: 10.1109/JSYST.2014.2360460.

[33] Richard C. Dorf and Robert H. Bishop. *Modern control systems.* Pearson Education, 12 edition, 2014.

[34] M. Fliess, J. Lévine, P. Martin, and P. Rouchon. *Nonlinear Control System Design.* Pergamon Press, 1992.

[35] J. Freudenberg and D. Looze. Right half plane poles and zeros and design tradeoffs in feedback systems. *IEEE Transactions on Automatic Control*, 30(6):555–565, 1985. doi: 10.1109/TAC.1985.1104004.

[36] J. Freudenberg and D. Looze. A sensitivity tradeoff for plants with time delay. *IEEE Transactions on Automatic Control*, 32(2):99–104, 1987. doi: 10.1109/TAC.1987.1104547.

[37] James S. Freudenberg. *Frequency Domain Properties of Scalar and Multivariable Feedback Systems.* Springer, 1988.

[38] Sneha Gajbhiye, Ravi N. Banavar, and Sergio Delgado. Symmetries in the wheeled inverted pendulum mechanism. *Nonlinear Dynamics*, 90(1):391–403, Oct 2017. ISSN 1573-269X. doi: doi:10.1007/s11071-017-3670-3.

[39] F. Grasser, A. D'Arrigo, S. Colombi, and A.C. Rufer. Joe: a mobile, inverted pendulum. *IEEE Transactions on Industrial Electronics*, 49(1):107–114, 2002. doi: 10.1109/41.982254.

[40] Yun-Su Ha and Shin'ichi Yuta. Trajectory tracking control for navigation of the inverse pendulum type self-contained mobile robot. *Robotics and Autonomous Systems*, 17(1):65–80, 1996. ISSN 0921-8890. doi: 10.1016/0921-8890(95)00062-3.

[41] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II.* Springer Berlin Heidelberg, 2 edition, 1996. ISBN 978-3-540-60452-5. doi: 10.1007/978-3-642-05221-7.

[42] John Hannah. A geometric approach to determinants. *The American Mathematical Monthly*, 103(5):401–409, 1996. ISSN 00029890. doi: 10.2307/2974931.

[43] John Heilemann. Reinventing the wheel. *TIME*, page 1–5, Dec 2001. URL `http://content.time.com/time/business/article/0,8599,186660-5,00.html`.

[44] Jesse B. Hoagg and Dennis S. Bernstein. Nonminimum-phase zeros - much to do about nothing - classical control - revisited part ii. *IEEE Control Systems Magazine*, 27(3):45–57, 2007. doi: 10.1109/MCS.2007.365003.

[45] Robert V. Hogg, Joseph W. McKean, and Allen T. Craig. *Introduction to Mathematical Statistics.* Pearson, Harlow, 8 edition, 2020. ISBN 1292264764.

[46] Tingshu Hu and Zongli Lin. *Control Systems with Actuator Saturation.* Springer Science + Business Media, 2001. doi: 10.1007/978-1-4612-0205-9.

[47] Tingshu Hu, Zongli Lin, and Ben M. Chen. Analysis and design for discrete-time linear systems subject to actuator saturation. *Systems & Control Letters*, 45(2): 97 – 112, 2002. ISSN 0167-6911. doi: 10.1016/S0167-6911(01)00168-2.

[48] Matthias Hölzle. Entwicklung eines multifunktionalen Roboters. Technical University of Munich, 2013. Term paper.

[49] Osama Jamil, Mohsin Jamil, Yasar Ayaz, and Khubab Ahmad. Modeling, control of a two-wheeled self-balancing robot. In *2014 International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE)*, pages 191–199, 2014. doi: 10.1109/iCREATE.2014.6828364.

[50] Supriya Jose and Anil Antony. Mobile robot remote path planning and motion control in a maze environment. In *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, pages 207–209. IEEE, Mar. 2016. ISBN 978-1-4673-9916-6. doi: 10.1109/ICETECH.2016.7569242.

[51] Thomas R. Kane and David A. Levinson. *Dynamics Theory and Applications.* McGraw-Hill Book Company, 1985.

[52] E. Kaszkurewicz and A. Bhaya. Discrete-time state estimation with two counters and measurement delay. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 2, pages 1472–1476 vol.2, 1996. doi: 10.1109/CDC.1996.572723.

[53] Felix Kaufmann. Implementierung und Evaluation verschiedener Regelungskonzepte auf einem zweirädrigen inversen Pendel. Master's thesis, Technical University of Munich, 2015.

[54] Steve Kemper. *Reinventing the Wheel - A Story of Genius, Innovation, and Grand Ambition.* HarperCollins, Heidelberg, 2005. ISBN 978-0-060-76138-7.

[55] S. Kim and S. Kwon. Nonlinear optimal control design for underactuated two-wheeled inverted pendulum mobile platform. *IEEE/ASME Transactions on Mechatronics*, 22(6):2803–2808, Dec 2017. ISSN 1083-4435. doi: 10.1109/TMECH.2017.2767085.

[56] Sangtae Kim and SangJoo Kwon. Dynamic modeling of a two-wheeled inverted pendulum balancing mobile robot. *International Journal of Control, Automation and Systems*, 13(4):926–933, Aug 2015. ISSN 2005-4092. doi: 10.1007/s12555-014-0564-8.

[57] Donald Ervin Knuth. *Volume 2: The Art of Computer Programming.* Addison-Wesley, Upper Saddle River, NJ, 3. ed. edition, 1998. ISBN 0201896842 - 9780201896848.

[58] P.F. Kuznetsov. The emergence of bronze age chariots in eastern europe. *Antiquity*, 80(309):638–645, 2006. doi: doi:10.1017/S0003598X00094096.

[59] Krzysztof Laddach, Rafał Łangowski, and Tomasz Zubowicz. Estimation of the angular position of a two–wheeled balancing robot using a real imu with selected filters. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 2022. doi: 10.24425/BPASTS.2022.140518.

[60] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM J. Optimiz.*, 9 (1):112–147, 1998.

[61] T.D. Larsen, N.A. Andersen, O. Ravn, and N.K. Poulsen. Incorporation of time delayed measurements in a discrete-time kalman filter. In *Proceedings of the 37th IEEE Conference on Decision and Control*, volume 4, pages 3972–3977 vol.4, 1998. doi: 10.1109/CDC.1998.761918.

[62] Christoph Leonhardt. Konstruktion und Regelung eines multifunktionalen Kleinroboters. Technical University of Munich, 2014. Term paper.

[63] Jongil Lim, SeokJu Lee, Girma Tewolde, and Jaerock Kwon. Indoor localization and navigation for a mobile robot equipped with rotating ultrasonic sensors using a smartphone as the robot's brain. In *2015 IEEE International Conference on Electro/Information Technology (EIT)*, pages 621–625. IEEE, May. 2015. ISBN 978-1-4799-8802-0. doi: 10.1109/EIT.2015.7293407.

[64] G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303, Aug. 1978. ISSN 0006-3444. doi: 10.1093/biomet/65.2.297.

[65] J. Löfberg. Yalmip: A toolbox for modeling and optimization in MATLAB. In *Proc. IEEE Symp. on Computer Aided Control Systems Design*, pages 284 – 289, 2004. URL `http://users.isy.liu.se/johanl/yalmip`.

[66] D.P. Looze and J.S. Freudenberg. Limitations of feedback properties imposed by open-loop right half plane poles. *IEEE Transactions on Automatic Control*, 36(6): 736–739, 1991. doi: 10.1109/9.86946.

[67] Cong Luo, Hui Li, Jie Tang, Li Lin, Yuan Wang, Wang Peng, and Lei Zhu. Computer aided design of two-wheel self-balancing vehicle based on stm32. In *2021 IEEE International Conference on Data Science and Computer Application (ICDSCA)*, pages 207–209, 2021. doi: 10.1109/ICDSCA53499.2021.9650274.

[68] P. H. J. B. J. Luo and K. Ying. Properties of savitzky golay digital differentiators. *Digital Signal Process.*, 15:122–136, 2005.

[69] Luis F Lupian and Rodrigo Avila. Stabilization of a wheeled inverted pendulum by a continuous-time infinite-horizon lqg optimal controller. In *2008 IEEE Latin American Robotic Symposium*, pages 65–70, 2008. doi: 10.1109/LARS.2008.33.

[70] Jean Lèvine. *Analysis and control of nonlinear systems.* Mathematical engineering. Springer, Berlin [u.a.], 2009. ISBN 9783642008382 - 9783642008399.

[71] J. E. Marsden and T. S. Ratiu. *Introduction to Mechanics and Symmetry.* Springer-Verlag, New York, 1994.

[72] Peter S. Maybeck. *Stochastic models, estimation and control*, volume 2 of *Mathematics in science and engineering.* Academic Press, New York, 1982. ISBN 012480702X - 0124807011 - 0124807038.

[73] D.L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, 1991. ISSN 00906778. doi: 10.1109/26.103043.

[74] Omar Mohamed Mohamed Gad, Suhaib Ziad Mohammad Saleh, Maleck Ayman Bulbul, and Sofiane Khadraoui. Design and control of two wheeled self balancing robot (twsbr). In *2022 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–6, 2022. doi: 10.1109/ASET53988.2022.9735004.

[75] Cleve Moler and Charles van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003. doi: 10.1137/S00361445024180.

[76] Manfred Morari. *Robust process control.* PTR Prentice Hall, 1989.

[77] Amarnath Mukherjee. On the dynamics and significance of low frequency components of internet load. Technical report, University of Pennsylvania Department of Computer and Information Sciences, 1992.

[78] Daniel Murdock. *Modeling, identification and control of a wheeled balancing system.* PhD thesis, Georgia Institute of Technology, 2016.

[79] J. A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7(4):308–313, 1965.

[80] Tsutomu Ohmae, Toshihiko Matsuda, Kenzo Kamiyama, and Makoto Tachikawa. A microprocessor-controlled high-accuracy wide-range speed regulator for motor drives. *IEEE Transactions on Industrial Electronics*, IE-29(3):207–211, 1982. doi: 10.1109/TIE.1982.356665.

[81] Rich Chi Ooi. Balancing a two-wheeled autonomous robot. Master's thesis, University of Western Australia, 2003.

[82] S.J. Orfanidis. *Introduction to Signal Processing.* Prentice Hall, 1998. ISBN 9780139789335.

[83] Daniel Pachner, Lubomír Baramov, and Vladimir Havlena. Suboptimal State Estimation under Communication Delays. *IFAC Proceedings Volumes*, 43(2):271 – 276, 2010. ISSN 474-6670. doi: 10.3182/20100607-3-CZ-4010.00049. 9th IFAC Workshop on Time Delay Systems.

[84] K. Pathak, J. Franch, and S. K. Agrawal. Velocity and position control of a wheeled inverted pendulum by partial feedback linearization. *IEEE Transactions on Robotics*, 21(3):505–513, June 2005. ISSN 1552-3098. doi: 10.1109/TRO.2004.840905.

[85] P. Philipp and S. Altmannshofer. Experimental validation of a new moving horizon estimator approach for networked control systems with unsynchronized clocks. In *2012 American Control Conference (ACC)*, pages 4939–4944. IEEE, Jun. 2012. ISBN 978-1-4577-1096-4. doi: 10.1109/ACC.2012.6315222.

[86] Peter Philipp. *Centralized and Distributed Moving Horizon Strategies for State Estimation of Networked Control Systems.* PhD thesis, Technical University of Munich, 2014.

[87] Peter Philipp and Boris Lohmann. Moving Horizon Estimation for Nonlinear Networked Control Systems with Unsynchronized Timescales. *IFAC Proceedings Volumes*, 44(1):12457 – 12464, 2011. ISSN 1474-6670. 18th IFAC World Congress.

[88] Karmvir Singh Phogat, Ravi Banavar, and Debasish Chatterjee. Structure-preserving discrete-time optimal maneuvers of a wheeled inverted pendulum. In *6th IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control LHMNC 2018*, volume 51, pages 149–154, 2018. doi: 10.1016/j.ifacol.2018.06.042.

[89] Sebastian Pieczona. Lyapunov-basierte Führungsgrößenadaption für die Bahn- und Trajektorienfolgeregelung unter Stellgrößenbeschränkung. Master's thesis, Technical University of Munich, 2012.

[90] Rostyslav V. Polyuga and Arjan J. van der Schaft. Effort- and flow-constraint reduction methods for structure preserving model reduction of port-hamiltonian systems. *Systems & Control Letters*, 61(3):412–421, 2012. ISSN 0167-6911. doi: 10.1016/j.sysconle.2011.12.008.

[91] Daniel T. Potts. *A companion to the archaeology of the ancient Near East.* Wiley-Blackwell, 2012. ISBN 9781405189880.

[92] Verena Priesack. Beobachterentwurf für ein zweirädriges inverses Pendel unter Verwendung eines Tracking-Systems. Technical University of Munich, 2017. Term paper.

[93] L. Qiu and E.J. Davison. Performance limitations of non-minimum phase systems in the servomechanism problem. *Automatica*, 29(2):337–349, 1993. ISSN 0005-1098. doi: 10.1016/0005-1098(93)90127-F.

[94] MRM Romlay, MI Azhar, SF Toha, and MM Rashid. Two-wheel balancing robot; review on control methods and experiment. *International Journal of Recent Technology and Engineering (IJRTE)*, 7(68):106–12, 2019.

[95] A. Savitzky and M. J. E. Golay. Smoothing differentiation of data by simplified least squares procedures. *Analytical Chem.*, 36:1627, 1964.

[96] Maria Seron. *Fundamental limitations in filtering and control.* Springer, 1997.

[97] Ling Shi, Lihua Xie, and Richard M. Murray. Kalman filtering over a packet-delaying network: A probabilistic approach. *Automatica*, 45(9):2134–2140, 2009. doi: 10.1016/j.automatica.2009.05.018.

[98] Sigurd Skogestad. *Multivariable feedback control.* Wiley, 2. ed. edition, 2007.

[99] Sigurd Skogestad, Kjetil Havre, and Truls Larsson. Control limitations for unstable plants. *IFAC Proceedings Volumes*, 35(1):485–490, 2002. ISSN 1474-6670. doi: 10.3182/20020721-6-ES-1901.00330. 15th IFAC World Congress.

[100] Julius O. Smith III. *Introduction to Digital Filters: with Audio Applications.* W3K Publishing, 2007. ISBN 978-0974560717.

[101] G. Stein. Respect the unstable. *IEEE Control Systems Magazine*, 23(4):12–25, 2003. doi: 10.1109/MCS.2003.1213600.

[102] Johannes Strohm. Festwertregelung eines zweirädrigen inversen Pendels. Technical University of Munich, 2014. Term paper.

[103] B. Sundararaman. Clock synchronization for wireless sensor networks: A survey. *Ad-Hoc Netw.*, 3(3):281–323, Mar. 2005.

[104] K. C. Toh, M. J. Todd, and R. H. Tütüncü. Sdpt3 — a matlab software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11 (1-4):545–581, 1999. doi: 10.1080/10556789908805762.

[105] R. H. Tütüncü, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using sdpt3. *Mathematical Programming*, 95:189–217, 2003. doi: 10. 1007/s10107-002-0347-5.

[106] Rudolph van der Merwe. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models.* Oregon Health & Science University, 2004.

[107] Rudolph van der Merwe, Eric Wan, and Simon Julier. Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion: Applications to Integrated Navigation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004. doi: 10.2514/6.2004-5120.

[108] C. van Loan. Computing integrals involving the matrix exponential. *IEEE Transactions on Automatic Control*, 23(3):395–404, 1978. doi: 10.1109/TAC.1978. 1101743.

[109] Jasmin Velagić, Imran Kovač, Adis Panjević, and Adnan Osmanović. Design and control of two-wheeled and self-balancing mobile robot. In *2021 International Symposium ELMAR*, pages 77–82, 2021. doi: 10.1109/ELMAR52657.2021.9550938.

[110] L. Wang, J. Fernandez, J. Burgett, R. W. Conners, and Y. Liu. An Evaluation of Network Time Protocol for Clock Synchronization in Wide Area Measurements. In *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, pages 1–5, July 2008.

[111] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, Aug. 1962. ISSN 0040-1706. doi: 10.1080/00401706.1962.10490022.

[112] DA Wells. Application of the Lagrangian equations to electrical circuits. *Journal of Applied Physics*, 9(5):312–320, 1938.

[113] P. E. Wellstead. *Introduction to physical system modelling.* Acad. Pr., London, 1979. ISBN 0127443800.

[114] Wikipedia. Segway — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Segway&oldid=1087563139`, 2022. [Online; accessed 25-July-2022].

[115] Wikipedia. Ibot — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=IBOT&oldid=1087743528`, 2022. [Online; accessed 25-July-2022].

[116] Yik-Chung Wu, Qasim Chaudhari, and Erchin Serpedin. Clock synchronization of wireless sensor networks. *IEEE Signal Processing Magazine*, 28(1):124–138, Jan. 2011. ISSN 1053-5888. doi: 10.1109/MSP.2010.938757.

[117] Tim Wunderlich. Entwurf, Implementierung und Evaluation von zeitdiskreten Beobachterkonzepten auf einem zweirädrigen inversen Pendel. Technical University of Munich, 2015. Term paper.

[118] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. doi: 10.1007/s10107-004-0559-y.

[119] Stanislaw H. Zak. *Systems and control.* Oxford University Press, 2003.

[120] Lishuang Zhang, Lei Sun, Sen Zhang, and Jingtai Liu. Trajectory planning for an indoor mobile robot using quintic bezier curves. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 757–762. IEEE, Dec. 2015. ISBN 978-1-4673-9675-2. doi: 10.1109/ROBIO.2015.7418860.

[121] Yan Zhang and Wenjie Cai. Design of self-balancing robot based on esp32. In *2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, pages 1039–1043, 2021. doi: 10.1109/ICCASIT53235.2021.9633673.

[122] Wei Zhao, Qun Lu, Jian Chen, Tiantian Liu, Shulan Xia, and Chun-Yi Su. Design of two-wheeled self-balancing vehicle based on fixed-time observer. In *2021 China Automation Congress (CAC)*, pages 3422–3427, 2021. doi: 10.1109/CAC53003.2021.9727421.