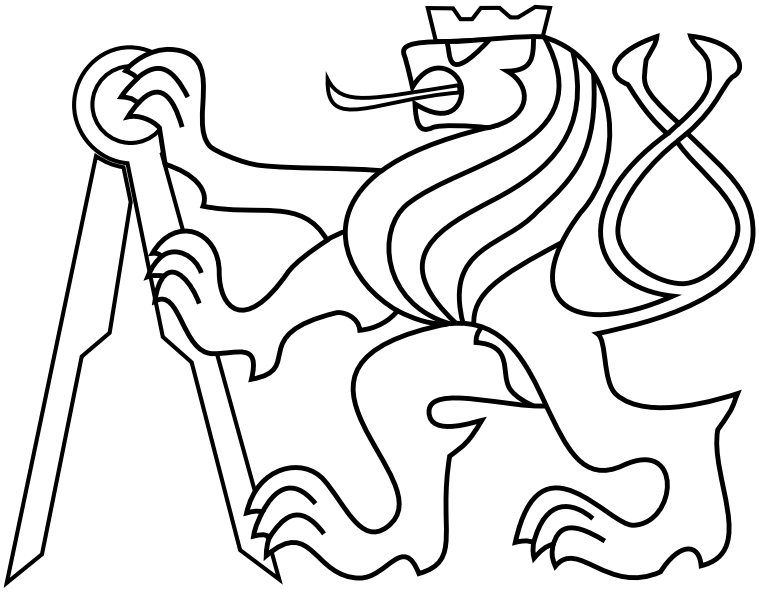
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER'S THESIS



Daniel Smrčka

Physical human-machine interaction with a small multirotor helicopter

Department of Control Engineering

Thesis supervisor: **Ing. Tomáš Báča**

Declaration:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....

signature

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Smrčka Daniel** Personal ID number: **466509**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Physical human-machine interaction with a small multirotor helicopter

Master's thesis title in Czech:

Fyzická interakce malé vícerotorové bezpilotní helikoptéry s člověkem

Guidelines:

The thesis aims to design, implement, and verify a system that would allow physical human interaction with Unmanned Aerial Vehicles (UAVs). The motivation arises from the Aerial Core project, where safe human-UAV cooperation is needed. Human-UAV interaction is usually disregarded in the feedback control loops of UAVs and any external forces on the UAV are typically interpreted as a disturbance. However, such interaction could be also interpreted as a signal to the UAV for, e.g., a change of the UAV's relative position. The aim of this thesis is to explore the potential of manipulation with a UAV by direct physical interaction. The following main tasks will be solved:

1. Design, construct and identify the dynamics of a small multirotor helicopter that is suitable for human-vehicle interaction.
2. Design and implement a state observer for estimating an external force acting on a UAV.
3. Conduct experiments (and/or simulations) for establishing the performance of the designed state observer.
4. Implement an admittance control [1] approach that will allow controlling the UAV based on the estimated external force.
5. If the hardware of the UAV allows, test the system behavior in the Gazebo simulator and conduct the real-world experiments, if possible.

Bibliography / sources:

[1] Augugliaro, Federico, and Raffaello D'Andrea. 'Admittance control for physical human-quadrocopter interaction.', IEEE 2013 European Control Conference (ECC), 2013.
[2] T. Báča, M. Petrlík, M. Vrba, V. Spurný, R. Pěnička, D. Hert, and M. Saska, "The MRS UAV System: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles,", https://arxiv.org/abs/2008.08050, 2020.
[3] S. Rajappa, H. Bülthoff, and P. Stegagno, "Design and implementation of a novel architecture for physical human-uav interaction,", The International Journal of Robotics Research, vol. 36, no. 5-7, pp. 800–819,2017.

Name and workplace of master's thesis supervisor:

Ing. Tomáš Báča, Department of Cybernetics, FEE

Name and workplace of second master's thesis supervisor or consultant:

doc. Ing. Martin Saska, Dr. rer. nat., Multi-robot Systems, FEE

Date of master's thesis assignment: **28.01.2021** Deadline for master's thesis submission: **13.08.2021**

Assignment valid until: **30.09.2022**

Ing. Tomáš Báča	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to thank my thesis adviser Ing. Tomáš Báča, for his guidance, patience with me and his support which allowed me to complete this thesis. Furthermore, I would like to thank all the members of the Multi-robot Systems group who helped me whenever I needed it. Special thanks goes to Ing. Jiří Horyna, who spent a lot of time with me discussing the problems I encountered and also for spending time reviewing this work.

I would also like to thank my family for their encouragement and support during my whole studies. Finally, my deepest thanks go to my beloved girlfriend Marcela, who stood behind my back all the time and always believed in me.

Thank you all. This work would not be possible without you.

Abstract

This thesis deals with a physical interaction of a human and an unmanned multirotor vehicle. A specialized platform that allows a safe interaction is designed and built. The dynamical model the designed platform is presented, and its parameters are identified. The main focus of this work lies in the design of a state observer capable of an external force estimation. Further, a control design suitable for physical interaction is proposed, using the Admittance control. The functionality of the developed system is verified in the series of simulation experiments.

Keywords: unmanned aerial vehicles, physical human-UAV interaction, Admittance control

Abstrakt

Tato práce se zabývá fyzickou interakcí člověka s bezpilotní vícerotorovou helikoptérou. Pro účely této práce byla navržena a postavena specializovaná platforma, která je vhodná pro danou aplikaci. Byl odvozen dynamický model této helikoptéry a jeho parametry byly identifikovány. Hlavní cílem této práce je návrh stavového pozorovatele, který odhaduje sílu pusobící na helikoptéru. Dále je představen řídící systém vhodný pro fyzickou interakci. Funkčnost navrženého systému je ověřena pomocí několika experimentů provedených v simulaci.

Klíčová slova: bezpilotní letouny, interakce s člověkem, Admittance control

Contents

List of Tables

List of Figures

List of abbreviations

GPS	Global Positioning System
CTU	Czech Technical University
SLAM	Simultaneous Localization And Mapping
GPS	Global Positioning System
ROS	Robot Operating System
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
MBZIRC	Mohamed Bin Zayed International Robotics Challenge
DARPA	Defense Advanced Research Projects Agency
IMU	Inertial Measurement Unit
MPC	model predictive control
KF	Kalman filter
LKF	Linear Kalman filter
UKF	Unscented Kalman Filter
EKF	Extended Kalman Filter
MRS	Multi-robot Systems group
VTOL	vertical takeoff and landing
FCU	flight control unit
ESC	electronic speed controller
FPV	first-person view
VIO	visual odometry
VINS	visual-inertial system
pdf	probability density function
RMSE	root mean square error
ALS	Autocovariance least squares
CAD	computer-aided design

1 Introduction

Unmanned Aerial Vehicles (UAVs), also known as drones, have gained much popularity in recent years. Such vehicles can have different configurations and looks, but a common thing to all is the absence of a pilot. The most popular UAV type is called a vertical takeoff and landing (VTOL) vehicle (figure 1.1). The popularity of VTOL vehicles is enormous, and they can be found in many different areas including the film industry, military, or hobby helicopter usage. Multirotor UAV usually consists of a frame from light materials (e.g., plastic or carbon fibre), a flight control unit (FCU) (low-level stabilization), brushless DC motors controlled via electronic speed controllers (ESCs), propellers, remote control receiver, and battery(e.g., Lithium-Polymer ones). Quadcopters, hexacopters, or octocopters are commonly used types of UAVs. The cheapest, most basic quadcopters are possible to get from tens of dollars. However, their deployment within research and industry is impractical regarding the limited equipment. On the other hand, costly vehicles can be equipped with different cameras, a GPS locator, or a robotic arm, which increases the range of potential applications.

In most cases, a human operator on the ground controls the UAV either by observing the vehicle or using a first-person view (FPV) camera. A possible way of improving the vehicle is by adding a level of autonomous assistance. An essential autonomy is provided by equipping the UAV with a computer to run necessary computations and adding sensors such as a Global Positioning System (GPS) receiver Such configuration can provide most basic functions like following objects or trajectories, which is crucial for performing more complex tasks. Many research laboratories nowadays deal with tasks like package delivery [5], fire extinguishing [6], or autonomous aerial monitoring [7].

The increasing range of applications also increases a need for a possible interaction of UAVs with the environment, including interaction with people. For example, the UAV system has to handle the interaction without harming the nearby people, during a package delivery.

Figure 1.1: An example of a general purpose quadcopter [1].

Another possible application might be direct cooperation with people, e.g., to help them carry and hand objects. However, UAVs are very dangerous vehicles. Their propellers usually spin tens of thousands of times per minute which makes contact with such propeller might be critical and, in the worst cases, imply fatal injury. Therefore providing a protective frame around the UAV and implementing an adequate control design is necessary.

The interaction force must be estimated first to allow the UAV to behave appropriately in physical contact with its surroundings. This task can be approached in many ways. One of the methods is the implementation of a state observer. It can be either linear or nonlinear. We expect to have a linearized linear model at some equilibrium, if the linear observer is used. Either way, the external force has to be included in the model. This approach is very convenient as it relies only on data from localization and does not require additional sensors.

The estimated external force can be used to control the UAV appropriately. Most controllers use the estimation and compensate for it to reach the best position tracking. Nevertheless, it is desirable to require the exact opposite reaction during interaction with people. Once there is a contact, the vehicle is tasked to move in the direction of that estimated interaction force to prevent an unwanted collision. Such an approach is well studied, and the most frequent methods are Impedance control and Admittance control [4].

The thesis is structured as follows. Chapter 1 introduces the concerned problem, an overview of state of the art, the contribution of presented work and used mathematical notation. Chapter 2 describes the necessary terms and tools used in the thesis. The description of dynamical designed model of a quadcopter is in chapter 3. Chapter 4 then shows a derivation of a dynamical model, which is needed to provide a Kalman filter implementation. Identification of this model is discussed in chapter 5. Theoretical description of state estimation methods is introduced in chapter 6, together with verification methods. Chapter 7 describes the theoretical details of the Admittance control. Implementation details are presented in chapter 8. Complete results of the thesis are presented in chapter 9. The conclusion of the thesis, together with discussion over the results, is in chapter 10.

1.1 Problem Definition

This thesis aims to design, build, and identify a small Unmanned Aerial Vehicle that is suitable for human-machine interaction. This vehicle is expected to have propeller protection to prevent harming the surrounding objects, including people. No motion capture system is used to provide localization and visual localization is used instead. The UAV will be designed for the usage primarily in indoor conditions. Furthermore, two-state estimation methods are designed to estimate an external force acting on the vehicle. Many sources of disturbances can be found around the vehicle. Thus, it is important to differentiate each element to provide solid estimates for the controller. To achieve that, we expect that disturbances driven by wind are omitted. Our goal is to separate only the parasitic disturbances and applied external force. The last part of the thesis focuses on force control. Such a topic is well studied [8], mainly in industrial robotics, typically connected with robotic arms. Therefore this thesis deals with extending human-machine interaction to the UAVs. The force control approach used in this strategy is called Admittance control. This approach only controls the reference position and velocity in the world based on the force acting on the controlled system. Hence, a position controller is further needed to complete the whole control pipeline. The design of the whole

control system is a complex and challenging task and it is beyond the proposed thesis. Thus, focus is put on design and implementation of Admittance control and its integration into the MRS UAV system.

1.2 State of the art

This thesis can be divided into two separate subtasks, force estimation and interaction control. Force estimation is well studied across the robotic field, and therefore multiple solutions are available. The topic of interaction control is based on the previous topic. Knowing the interacting force together with its direction and magnitude is crucial to provide reasonable control action. Combinations of both tasks are then called physical interaction. This topic is also well studied in industrial robotics, mainly in combination with robotic arms. Although there is also a contribution in the UAV research area, it is not nearly that significant.

1.2.1 Force estimation

The force that is to be estimated in the thesis represents disturbance acting on a vehicle. If there were no other sources of other disturbances, it would be enough to estimate disturbance as one and call it external force. Unfortunately, it is not usually like that. The estimated disturbance can be typically divided into two main components: internal and external disturbance. The internal disturbance is caused by the errors on the electronics, unbalanced platform, modelling error, and all things within the UAV itself. On the other hand, external represent all the external factors such as wind, ground or wall effect, or the interaction force caused by contact. However, whether the application requires getting information of specific components or not, disturbances are first estimated as one and then parsed based on additional information. Therefore the initial approach is typically similar.

Problems without the need for knowledge of disturbance source appear typically from precision flying problems or position control tasks. The control within such problems compensates for all the disturbances to be as precise as possible. California Institute of Technology, together with the ETH Zurich, published disturbance estimation and rejection for high precision control [9]. They presented a solution for rejecting strong wind gusts up to $12\,\mathrm{m\,s^{-1}}$ and ground effect by implementing and comparing augmented model predictive control (MPC) and PID control. Non-linear Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) were used to estimate disturbances. Based on several tests, the UKF did not provided significant improvement over the EKF. Furthermore, the EKF showed as more effective, because of its computational load. They tested their work in real-world experiments using a motion capture system.

A different approach was shown in [10]. The main focus was to design model predictive control for a UAV running onboard embedded controller. To provide a disturbance rejection, they presented disturbance estimation as a subtask. By adding the external force elements to the linear model as a component of acceleration, they were able to estimate the force by the Linear Kalman filter (LKF). This approach was tested using a platform localized with an optic-flow sensor in indoor conditions.

Another approach was presented by the Institute of robotics and Mechatronics in Germany [11]. They presented a model-based approach and divided the solution into momentum-based and acceleration-based approaches. They were able to estimate the external force purely based on Inertial measurement unit data and known input commands, by defining a model correctly and using the mentioned technique.

1.2.2 Interaction control

Background on the topic of interaction control is vast and was implemented in many applications. The most frequent approaches are called Impedance and Admittance control.

Christian Ott et al. [4] presented an article concerning the topic of these two control approaches. As mentioned in the article, "Impedance and Admittance control are two distinct implementations of the same control goal." Both methods provide control action that directly works with an external force. The difference between them is in the structure of the control pipeline. The Impedance control behaves as mechanical impedance and directly feeds out the estimated force to the controlled plant. Whereas in the Admittance control, the external force is converted to the position command, fed to the position controller. The article itself also provides pros and cons to both the mentioned approaches. Impedance control better behaves in more stiff conditions, whereas Admittance control in soft conditions.

University of Twente, Netherlands, elaborated more on the Admittance control [12] used in the physical interaction between humans and machines. They provided a framework where many concerning effects and their influence on the controller performance can be studied. Force signal filtering, internal robot flexibility, or influence of feed-forward control were effects taken into account. At last, they provided seven design guidelines to achieve the best performance.

An example of an application of Admittance control that is not UAV-control related is controlling dual-arm robots used in industry. Sonny Tarbouriech et al. presented their work in their study [13].

1.2.3 Physical human-UAV interaction

The application of both mentioned topics was also studied within the UAV research.

Federico Augugliaro and Raffaello D'Andrea from ETH in Zurich presented a work [14] that strongly motivated this thesis. Their work focuses exactly on the topic of Admittance control for the human-UAV interaction. They used position and attitude information in combination with the UKF to estimate the external force. Other than external interacting forces were not considered, and their experiments were presented under a motion capture system.

Sujit Rajappa et al. had a different approach to this task. They presented a mechanical construction around the UAV equipped with contact sensors. It provided a reliable solution to differentiate the interacting contact force from other disturbances. Their approach was to use an unscented Kalman filter to obtain all disturbances acting on the vehicle. Based on the information from the contact sensors, the disturbance components were separated from each other [8].

1.3 Contribution

This thesis presents a solution to the physical interaction of humans with the small quadcopter in real conditions without additional sensory infrastructure, such as a motion capture system. We provide an interaction-safe design of a small (800 g, 200 mm) quadcopter made of carbon fiber parts. The mathematical model of proposed vehicle was identified and studied in detail to allow external force estimation based on linear and nonlinear estimation. Linear Kalman filter and nonlinear Unscented Kalman Filter were implemented and verified based on the multiple tests. An Admittance force control was implemented, to allow a reasonable behaviour of the quadcopter around humans. This approach allowed controlling desired velocities and position based on the external force. All the work was integrated to the MRS UAV control system that was recently made open-source.

1.4 Mathematical notation

This section denotes information about mathematical symbols used in this thesis (table 1.1).

Symbol	**Description**
Upper and lowercase letter - x, X	scalar
Bold lowercase letter - $\mathbf{x}$	column vector
Bold uppercase letter - $\mathbf{X}$	matrix
$\mathbf{x}^T$, $\mathbf{X}^T$	vector and matrix transpose
$\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$, $\dddot{\mathbf{x}}$	first, second and third time derivative
$\mathbf{x_t}$	vector $\mathbf{x}$ at time t
$\|\mathbf{x}\|$	Euclidean norm of a vector $\mathbf{x}$
$\mathbf{x}^{(\mathcal{W})}$	vector $\mathbf{x}$ in coordinate system $\mathcal{W}$
$\mathbb{R}, \mathbb{Z}$	Real number, Integer numbers

Table 1.1: Used mathematical notation.

2 Preliminaries

This chapter describes the software tools that are used in this thesis. First, Robot Operating System, which is used to control a UAV, is introduced together with the Gazebo simulator. Then, the structure of an MRS UAV system is shown, and the main parts included in this work are depicted.

2.1 Robot Operating System

Robot Operating System (ROS) is an open-source middleware framework. As mentioned on the official website "*It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms*"[15]. ROS has a great variety of libraries that simplify the user's work thanks to the fact that many people can contribute to the system. It allows for a different and robust background in the process of robot development. Many recent and trending research tasks might look trivial to humans, but it needs a huge effort to achieve the goal from the robot's perspective. It would be difficult to deal with all the aspects of the research without the robotic community and software support.

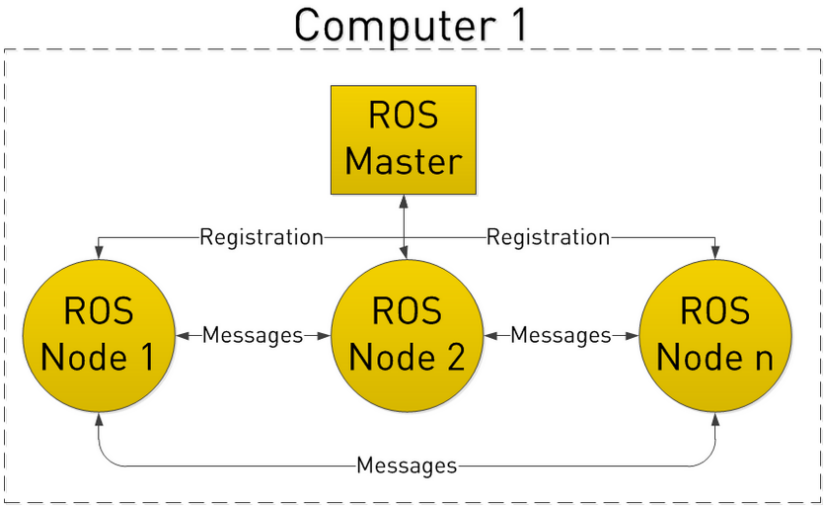


Figure 2.1: ROS structure

The basic working structure of ROS is depicted in figure 2.1. The ROS Master keeps track of all separated nodes and allows different features between them. The most common way of communication between nodes is via messages. Each node can publish data to topics and simultaneously subscribe data from any other topic. Another communication feature is

called service. Services can trigger any node and pass information from the user while the system is already running. Sharing the configuration parameters is allowed thanks to the global parameter server that keeps track of each parameter and allows for its changes. All features described above make the system built upon ROS reliable and robust against the failure of individual node, which is crucial for robot development.

2.2 Gazebo robotic simulator

Gazebo [16] is a realistic robotic simulator that allows for creating custom platforms and models of hardware sensors. Gazebo is compatible with ROS, which perfectly fits the needs of the MRS group and the purpose of this thesis. Native support of Pixhawk autopilot is another essential feature of the simulator since Pixhawk is used as one of the few out of the box components in the MRS system.

2.3 MRS system

This thesis is built upon the MRS UAV system [2]. This system provides fully autonomous control over the various types of multirotor vehicles. It is prepared for a wide range of possible applications and allows user to work separately on multiple levels. From high-level planning and navigation, through adding and testing different sensors to lower-level localization and control pipeline. flight control unit (FCU) is the only part of the system that is restricted and used externally. It is a low-level autopilot that provides basic attitude stabilization and thrust control of the vehicle. Pixhawk autopilot [3] is the most commonly used autopilot in the MRS. More information about it is can be found in chapter 4. The MRS system is implemented in ROS, which provides great extensibility. MRS system can be divided into three main subsystems: control, odometry, and planning subsystems. Its overall architecture is depicted in figure 2.2.

The odometry subsystem is created in a way that it usage of mixing many possible localization methods. GPS, optic-flow, Simultaneous Localization And Mapping (SLAM) algorithms or a Visual odometry (VIO) are all options that are currently implemented within the pipeline. The type of localization used is based on the user's choice and the sensor equipment on a UAV. The odometry subsystem covers blocks *State estimator* and *Odometry & Localization* in figure 2.2.

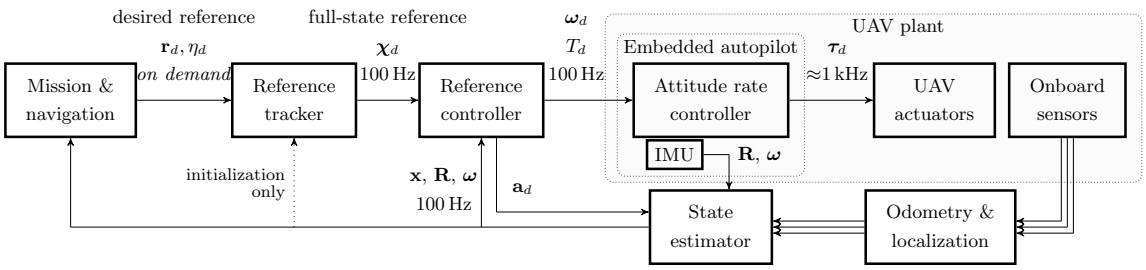


Figure 2.2: Structure of the MRS UAV system pipeline [2].

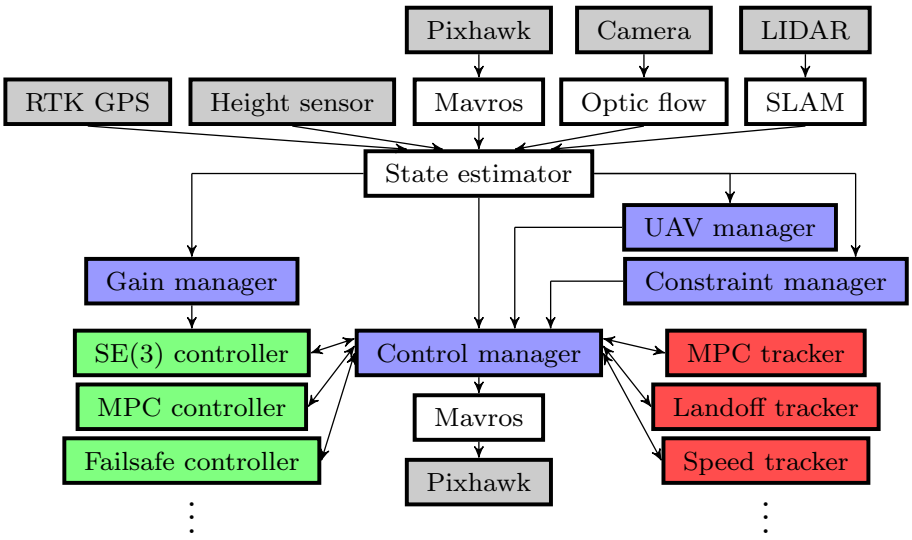


Figure 2.3: Structure of the managers implemented within the MRS system [2].

The control subsystem provides the necessary conversion between the desired state of the UAV and the attitude rate and thrust which is needed for the FCU. The control subsystem is based on two controllers, model predictive control and a *SE3* controller [17]. This combination shows as adequate and enough for various applications, and the MRS team builds upon these two controllers. To provide a better control performance, the underlying controller is preceded by the reference tracker that parses the user commands from the planning algorithm to feasible states. The structure of the control subsystem allows possible multi-level safety as both the trackers and the controllers have separated constraints.

The combination of the odometry subsystem and the control subsystem provides the main core of the MRS system. The implementation of high-level planning and navigation is build upon these two subsystems.

The whole MRS system is wrapped in several managers that fit well in the distributed architecture (figure 2.3). These managers provide control over separated parts of the pipeline.

- *Control manger* manages changes between controllers and trackers during the flight. It keeps track of control errors, the feasibility of inputs passing to the autopilot and triggers emergency landing safety features. Furthermore, it enables using the RC transmitter (real flights) and ROS-compatible joystick (simulation flights).
- *Gain manager* responds to the gains passing to the controllers. Each controller expects different sets of parameters. It is convenient to allow dynamic changes during the flight, which *Gain manager* does.
- *Constraint manager* stores and provides sets of constraints used within the trackers.
- *UAV manger* takes care of the procedure around the takeoff and landing. Desired takeoff height, maximal thrust, or maximum altitude are included here.

This presented architecture is the basis of the MRS group success in recent years. Its contribution in the world of UAVs is significant. The success in the Mohamed Bin Zayed

International Robotics Challenge (MBZIRC) 2020 and 2017 competitions proved the MRS system's capability in real-world conditions against the best universities in the world. The contribution was undoubtedly shown in the Defense Advanced Research Projects Agency (DARPA) challenge, where UAVs with the presented system competed together with the team of Unmanned Ground Vehicles (UGVs) from the Czech Technical University (CTU). Apart from these most significant world-known competitions, the MRS system is also actively used and developed in multi-robotic radiation sensing [18], historical buildings documentation [19] or AERIAL-core project [20].

3 Platform design

Physical human-machine interaction is a challenging task that involves special demands on the control level. Providing capable force estimation and robust control approach is indeed the core of the task. However, a significant part of this thesis is also to provide a safe platform for the experiments. There should always be safety bounds to protect interacting humans from danger.

In the case of the UGVs or robotic arms, there is usually an important red button that shuts down all the systems and makes the robot stop the movement. Most of the time, this is the safest approach. Nevertheless, when it comes to the UAVs, implementation of such a button is not straightforward. Because it is a flying vehicle, shutting down all systems would lead to a crash. Moreover, even the smallest UAVs are very dangerous to the surroundings. As already mentioned in chapter 1, contact with their propellers can harm a human very seriously. In combination with the fact that UAV can often be operating at a significant height, it presents another challenge of dealing with such critical situations and preventing unwanted injuries.

One possible approach to this is to replicate an idea of the red button, but instead of shutting down the system, make the UAV perform some manoeuvre. However, what kind of manoeuvre to use is not straightforward, and there is not a universal solution that one might say to be the best. The UAV can be already in a complex space to perform any manoeuvre or too close to objects which may affect the control abilities. An existing example of such a solution can be found within the MRS UAV system. A manoeuvre that is performed in their implementation is called *Emergency landing*. Once the safety pilot that holds the RC controller spots anything odd or the system itself discovers a problem, e.g. large control error, the pilot presses the switch, and the UAV slowly lands. Based on the current state, if flying or hovering, it either lands directly or lands on a trajectory that keeps the direction of the original trajectory.

Although such a manoeuvre usually safes an aircraft, it does not protect the UAV from harming the surroundings. The easiest solution to prevent from this potential problem is to equip the aircraft with a safety protective frame. With such protective equipment, even if the UAV would be landing, and there was an object in a way, the chance of hurting anybody or damaging UAV and objects would be significantly lowered.

The motivation behind this work is not to develop an intelligent system that would prevent collisions when needed. It is expected that the experiments are performed at heights up to 2 m, and an environment is without unnecessary objects. Hence, the only possible danger comes from performing real physical interaction, and a sufficient level of safety can be reached by making the UAV in a way that propellers are guarded from possible contact. It is also assumed that a small aircraft is more suitable for the thesis than a large vehicle. This assumption comes from the fact that necessary equipment is lightweight and the solution

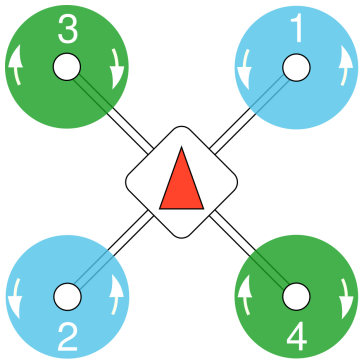


Figure 3.1: Configuration of a quadcopter setup [3]. Numbers and arrows in the circles symbolize a direction of a rotation and order of a propeller in the Pixhawk autopilot. The center red arrow shows the orientation of the vehicle.

should be easily scalable to perform with any platform.

Although there are many commercial frames and complete UAVs on the market, not many of them include propeller protection. If they do, it is usually lightweight plastic protection only, which does not work on a more significant impact and tends to bend. There are certainly exceptions. Complete vehicles with protection can be found. However, extending such a vehicle to fit all necessary equipment to allow needed functionality would be complicated. Therefore such an option is not very suitable for the purpose of this work.

There is research on building a platform that is suitable for human-UAV interaction. In [21], authors presented their solution on this topic. They presented a protective shroud built around a general quadrotor and demonstrated a capability of continuing in-flight after a collision. Their design was mounted at the centre of the vehicle and had a shape of a block. It proved to be working well, and therefore it makes an option how to proceed with the design of a protective cage.

Another very robust solution that is already applied in the industry is presented by a company *FlyAbility* [22]. Their solution consists of a spherical cage build around the vehicle made of tubes connected multiple spots. This makes it a optimal solution to fly in a confined space. They offer several types of solutions, add-on drone cages that can be customized and mounted on any UAV. Fixed-caged solution similar to the add-on cage with the difference that fixed-cage is built from the ground to sustain higher impact collision. The last offer is a decoupled drone cage which works similarly as a gimbal mechanism. Its structure allows making the aircraft stable while the cage around is rotating. Their solutions are robust and ideal even for human-machine interaction. However, it is an excessive bulk for our purpose.

Based on the matters mentioned above, it was decided that the best option is to design a custom UAV. Designing a custom UAV consists of mechanical design, selecting adequate drive components and adding sensory equipment. The mechanical design includes creating a base frame to which all the components are mounted. Drive components consist of a selection of motors with all electronic components and batteries. The last part includes all optional sensors that a user want to use.

3.1 Mechanical design

The design of the custom UAV was modeled using a 3D computer-aided design (CAD) software [1]. The structure of the aircraft is separated into three parts

- Base frame - A part that represents the main building piece of the aircraft. All the components are mounted here.
- Extension of the base - To create more space for mounting, another layer is created on top of the base frame. It is a flat rectangle that is placed over the centre and is separated via 3 cm plastic spacers.
- Propeller guards - Propeller protections are independent from each other, and each propeller has its guard. Each guard is mounted on the base frame.

It is sufficient to design the most common type of UAV, the quadcopter (basic structure is on figure 3.1). There is no added value of more complicated platforms for this thesis. The size of the frame was chosen to have a diagonal (from one motor diagonally to the opposite one) of 200 mm. Such dimension was selected based on different reasons. The idea was to build a vehicle that is as small as possible and explore how small can be an autonomous helicopter that still fit into the MRS UAV laboratory. So far, the smallest UAV within the MRS laboratory has a diagonal size of 330 mm which carries a large computer (Intel NUC) and numerous extra equipment. It is expected to use a much smaller Raspberry Pi computer and only a lightweight camera for localization. Therefore the dimensions could be much smaller.

Next, the size of the propellers was set between 4 - 5 inches. This was selected based on the common knowledge and parameters of similarly sized vehicles on the market. Furthermore, an estimate of the total weight was created. Similarly, as in table 3.1 with only approximated frame weights. The final estimate of the total weight was compared to an average maximal thrust of motors with 4-inch propellers. This provided theoretical maximal thrust force generate by the motors and proved that it is capable of carrying the whole vehicle.

The final design of the frame is in figure 3.2. The main focus was put on the arm sections, where the space for propeller guards is needed. The center section of the frame was modelled to provide mounting holes for the second layer of the frame and sensor equipment. The frame was skeletonized to mak it as light as possible. More detailed view is on figure 3.4. The realisation of designed model is then depicted on figure 3.5 during an experimental flight.

Carbon fibre was chosen as a construction material. Its properties such as high stiffness, low weight to strength ratio or high tensile strength make it a suitable material. Additionally, based on the guideline [23] the thickness of the base frame was selected to 4 mm. The thickness of the propeller guards and top part of the frame was picked to 2 mm.

The UAV was designed in such a way that it allows for two configurations. The propellers can be placed either under the base frame or on the top. A more conventional way is to equip the vehicle with 4-inch propellers and put the motors to face upwards. This configuration is sufficient to fly. However, the potential is not maximized as a lot of space is consumed by the second layer of the frame, which occurs between the propellers and the space under the

[1] Freecad 3D free open-source parametric modelling application

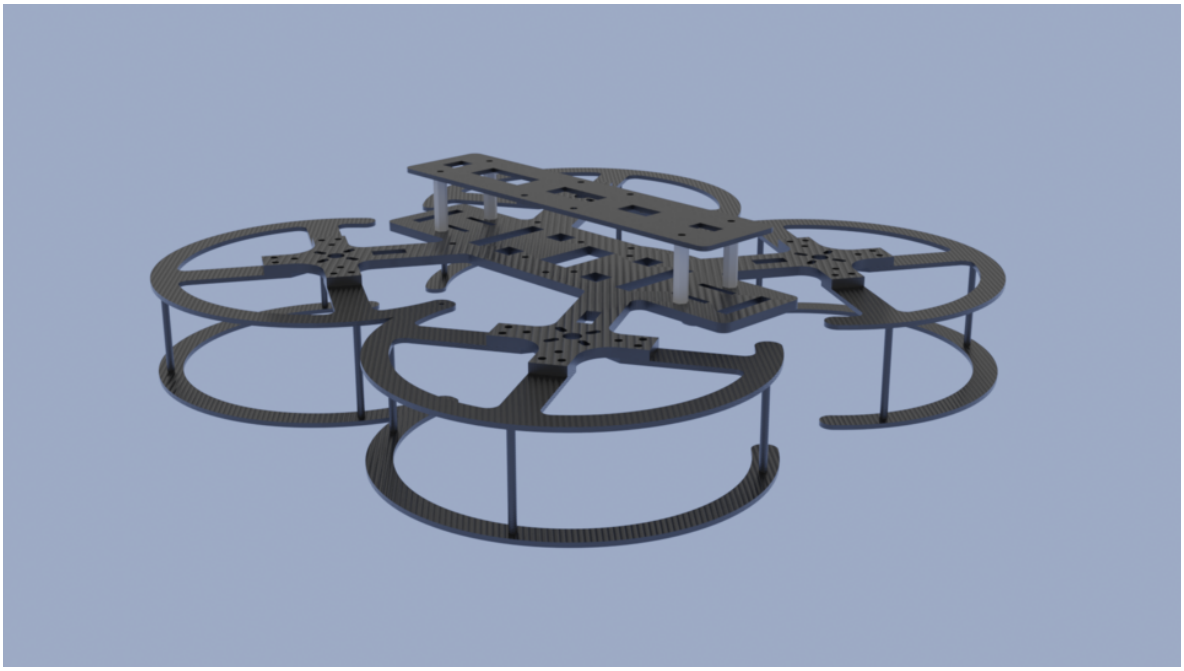


Figure 3.2: The full design of a custom UAV that is prepared to fit 5-inch propellers under the base frame.

vehicle is free. More innovative configuration uses the 5-inch propellers that are placed under the vehicle. This requires rotating the motors to face downwards. Another profit comes from the special use of the protective guards, which can serve as legs.

3.2 Components

To choose all the propulsion components that are suitable, it is convenient to use the online tool *eCalc*. This tool allows for a selection of different propulsion subsystems and finds the optimal components. The only assumptions made before were about the expected weight of the aircraft, size of the frame and size of the propellers. A table that sums up the weight of each component and total weight is in table 3.1. The resulting parameters provided by the online tool are shown in figure 3.3.

As mentioned in [24], an important parameter for designing a multi-rotor vehicle is maneuverability. Maneuverability is a key parameter that defines how easily can the vehicle maneuver in a space. However, this variable is immeasurable, thus a relation with a thrust-to-weight ratio is proposed. Thrust-to-weight ratio is defined as a ratio between a maximal thrust generated by a vehicle to a vehicle's total weight. As the author justifies, the ratio should not be lower than 2:1, which is an optimal solution. Our design provided theoretical a ratio of 2.7. It is over the optimal value, however, it is not wrong, as the vehicle will have higher manoeuvrability than is necessary.

Following list briefly summarizes the final components that were chosen

- Motors - *Graupner Ultra 2806-2300* kV motors are great combination of low weight

Figure 3.3: Result of *eCalc* design of the drive components and its flight characteristics.

and great power. They come from a category of brushless DC motors that work on switching DC current on several windings. Such switching creates a magnetic field that then makes the motor rotate. The size of the motor can be deduced from the number 2806 in the name. It means that the motor has diameter 28 mm and its inside height is 6 mm. Another parameter that specifies the motor is the revolutions per minute per volt of an unloaded motor usually denoted as kV. In our case it is 2300 kV. And because used voltage is 14.8 kV, the theoretical maximal RPM can be calculated as

$$maxRPM = 2300 \cdot 14.8 = 34080 \text{ RPM}. \tag{3.1}$$

- electronic speed controllers (ESCs) - *Turningy Multistar BlHeli32 ARM* is electronic that provides a connection between autopilot and motors. In particular, it typically converts the desired motor speed scaled to the range [0, 1] that comes from the autopilot to 3-phase signal that excites the motors. The ESCs can be additionally programmed to allow braking, acceleration or change direction of spinning. Our choice of the controller can work with current up to 30 A.

- Propellers - *Graupner 5030* are two-blade plastic props that are compatible with the chosen motors. They are 5-inches long and have a 3-inch pitch. Propeller's pitch is a measurement of how far the propeller would move for one revolution in an ideal fluid. The lower-pitched propellers are more efficient, but the torque they produced is lower. On the other hand, higher-pitched props are less efficient, because they cause more turbulence which lowers the lift. In our case, the middle pitched propellers were selected.

Component	**Specifications**	**Weight** [g]
Frame	Custom	157
Prop. guards	Custom	40
2nd frame layer	Custom	4 x 43
Motors	4 x Grauptner 2206-2300 KV	4 x 33
ESCs	4 x BlHeli32	4 x 3
Propeller	4 x Graputner 5030	4 x 4
Battery	Tattu R-line 1800 mAh	200
Autopilot	Pixhawk 4 mini w/o case	18
Transmitter	Optima Sl 2.4 GHz	22
Camera	Bluefox	45
Rangefinder	Garmin Lidar-Lite	23
Total weight		818

Table 3.1: Summary of used components with their weights.

- Battery - *TATTU R-Line* - 4S 1800 mAh 14.8 V is a small Lithium-Polymer battery that provides a power source to all motors, ESCs, an autopilot, an onboard computer and possibly to sensory equipment. Its 1800 mAh capacity is enough power to all the components while the flight is still reasonably long.

- Onboard computer - *Raspberry Pi 4B* is a single board computer that provides a computational capability to allow for necessary computations. It has 8 GB RAM memory. Its dimensions, together with a quad-core processor's, make it suitable for small autonomous vehicles. This computer is capable of running any UNIX-based operating system. Hence it can be connected to autopilot and provide communication via MAVlink protocol [25]. It also provides several USB ports, an RJ-45 ethernet connector, general-purpose input-output (GPIO) pins, therefore many external components can be added.

- Autopilot - *Pixhawk 4 mini* is a lightweight open-source autopilot that provides attitude stabilization and controls the vehicle. It includes two accelerometers, two gyroscopes, a barometer and a magnetometer to provide feedback to the control loop. The 32-bit ARM processor allows for the implementation of multiple Kalman filters that filters the sensed data. The Pixhawk 4 Mini is a smaller version of the Pixhawk 4, which is widely known.

- Radio receiver - *Optima SL* 2.4 GHz is a single line radio receiver that passes information from the RC controller with a compatible radio transmitter to the autopilot. It works on public 2.4 GHz frequency and provides eight channels for communication. In practice, four channels are taken by the sticks operating roll, pitch, yaw and thrust, and the other four channels are user-specified to fit the application.

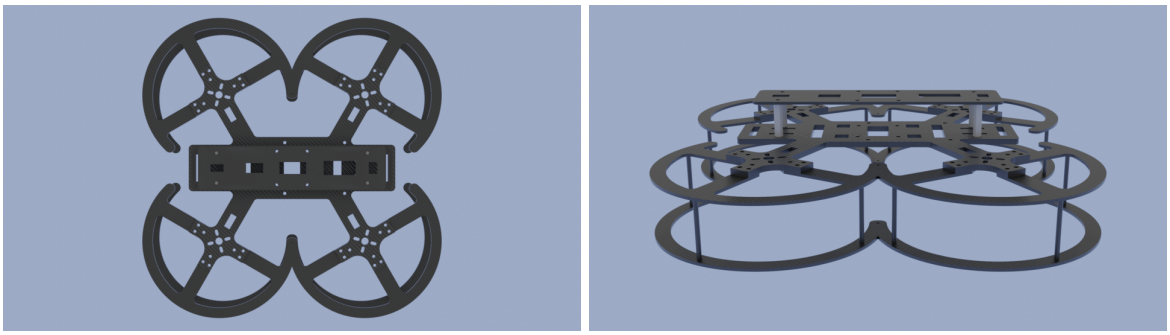


Figure 3.4: Detail on designed model.

3.3 Sensory equipment

This section provides information about the sensors that were used on the designed vehicle.

- Visual odometry camera - Miniature camera *mvBlueFOX* together with high resolution miniature SuperFisheye lens provides fast and quality image information for a visual based state estimation. The lens provides view greater than 180° with a resolution up to 5 MP. Camera itself can be connected to the computer via USB. Camera is further equipped with a Inertial Measurement Unit *ICM - 42688 - P*. The IMU is a 6-axis high precision motion tracking device, that includes 3-axis gyroscope and 3-axis accelerometer. With this combination, a monocular visual-inertial system (VINS) can be used.
- Rangefinder - The vehicle can be optionally equipped with a *Garmin Lidar-Lite*. It is a laser rangefinder that works up to 40 m with a resolution of 1 cm. The accuracy of the measurement is ± 2.5 cm in the height up to 5 m and ± 10 cm in the heights over 5 m.

3.4 Localization

This section provides information about visual odometry (VIO) localization algorithm that is used on the designed platform. Self-localization of UAVs using visual odometry is a popular topic as it can provide state information based only on one or two cameras. Author in [26] summarized and compared multiple algorithms. A localization algorithm VINS-Mono was chosen based on his work.

3.4.1 VINS-Mono

This algorithm was developed by authors from Hong Kong University [27]. It provides a state estimation based on a single monocular camera and IMU. They provide a robust procedure for estimator initialization. It then uses a non-linear optimization-based method to obtain odometry by fusing IMU measurements and feature observations. The algorithm

can also reuse a map by saving and loading it efficiently. The maps can be further merged thanks to a global pose graph optimization. Their work proved that their system is complete, versatile and reliable and therefore fits the purpose of this thesis.

Figure 3.5: Detail photos of the real model during experimental flight. The model is equipped with all necessary sensors, including VIO camera.

4 System description

Knowledge of the UAV's dynamics is important when predicting a system's behaviour and providing feedback control. If the dynamics is not complicated and the system itself is simple, it is often sufficient to omit the system's description and empirically tune necessary parameters. This, however, might be quite dangerous as the behaviour of the system is unpredictable. Moreover, as the number of parameters included in the system grows, it becomes more and more challenging to find the right values. Therefore, when dealing with complicated systems, providing a mathematical model is necessary. Such a model provides information about the system states evolution and responses to the different inputs. This technique is called a Model-Based Design.

The approaches to the modelling of the dynamical system differ based on how much about the system is known. It can be structured as follows,

- Whitebox model - This term means that all dynamics of the system can be described using mathematical equations, based on, e.g., Hamiltonian or Lagrangian mechanics. This model is based purely theoretically, and no data are needed to tune any parameters.
- Greybox model - This model is used when only partial information about the system is known. It contains unknown parameters, and experimental data are necessary to get the unknown parameters.
- Blackbox model - This expects no information about the inner workings of the model. The only thing that is known is the response to inputs.

In our case, mathematical description is known, and multiple assumptions are made to provide a sufficient model, yet still not over-complicated. Therefore, the system is modelled as a Greybox.

4.1 Coordinate systems

Multiple coordinate systems are presented to provide a clear description of the dynamics of the vehicle. Figure 4.1 depicts how the systems are oriented. World coordinate system $\mathcal{W}$ with axis $\{\mathbf{e}_1,\ \mathbf{e}_2,\ \mathbf{e}_3\}$ is a fixed frame in the world. It provides the base point for further orientation in the world. The position and orientation of the vehicle are expressed w.r.t this system. Next system is the body frame $\mathcal{B}$, this frame is aligned with a mechanical frame of the UAV. Last system is an untilted body frame $\mathcal{U}$, it has an origin in the centre of the UAV but is rotated only around the z-axis.

To allow for the transformation between frames, the rotation matrix $\mathbf{R}$ is used to represent a rotation from a child frame to a parent frame. To achieve this, the rotation matrix is

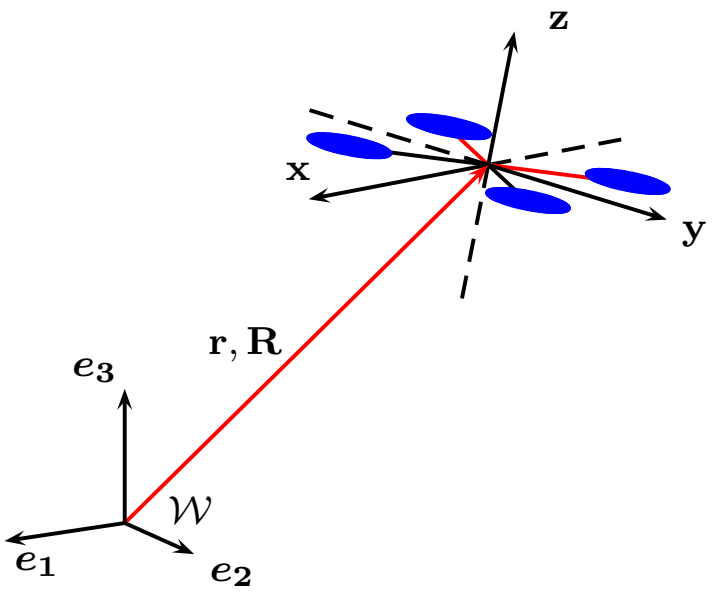


Figure 4.1: UAV's coordinates frames. Position and orientation of a UAV in the world frame $\mathcal{W}$ is given by a position vector $\mathbf{r}$ and rotational matrix $\mathbf{R}(\phi, \theta, \psi)$. The rotation matrix is given by the intrinsic Tait-Bryan angles, also known as roll, pitch and yaw.

expressed by intrinsic Tait-Bryan angles. It means that the matrix is obtained by continuous rotation around all three axes in the order z-y'-x". Intrinsic represents that the next rotation is done around an already rotated system. These rotation angles are further known as ϕ, θ and ψ and the corresponding matrix is obtained by (4.1).

$$\mathbf{R} = \mathbf{R}_z(\psi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_x(\phi), \tag{4.1}$$

where each $\mathbf{R}_i$ represents the rotation around i-axis. This results in the final rotation matrix,

$$\mathbf{R} = \begin{pmatrix} c(\psi)\, c(\theta) & c(\psi)\, s(\phi)\, s(\theta) - c(\phi)\, s(\psi) & s(\phi)\, s(\psi) + c(\phi)\, c(\psi)\, s(\theta) \\ c(\theta)\, s(\psi) & c(\phi)\, c(\psi) + s(\phi)\, s(\psi)\, s(\theta) & c(\phi)\, s(\psi)\, s(\theta) - c(\psi)\, s(\phi) \\ -s(\theta) & c(\theta)\, s(\phi) & c(\phi)\, c(\theta) \end{pmatrix}. \tag{4.2}$$

The rotation matrices are square matrices and belong to a special orthogonal group $\mathcal{SO}_3$. This gives to the matrices the following important attribute

$$\mathbf{R}^{-1} = \mathbf{R}^T, \tag{4.3}$$

which simplifies the transformations. Instead of computing an inverse of a matrix, it is possible to simply transpose the matrix and get transformation in the opposite direction.

4.2 UAV dynamics

A study of UAV's dynamics had already attention of many researchers and is considered as done. For example, the author in [28] describes multiple ways how to get the differential

equations of the quadcopter dynamics. He provides a unified basis for future research. Its description of final differential equations are following,

$$m\ddot{\mathbf{r}} = mg\mathbf{e}_3 + \mathbf{R}\mathbf{f}_d + \mathbf{R}\mathbf{f}_e \tag{4.4}$$

$$\mathcal{I}\dot{\boldsymbol{\omega}} = mgS(\mathbf{r}_g)\mathbf{R}^T\mathbf{e}_3 + S(\mathcal{I}\boldsymbol{\omega})\boldsymbol{\omega} + \mathbf{m} + \mathbf{m}_e \tag{4.5}$$

$$\dot{\mathbf{R}} = \mathbf{R}S(\boldsymbol{\omega}), \tag{4.6}$$

where m is a robot mass, $\mathbf{r} = [x, y, z]^T$ is a position vector, $\mathbf{R} \in \mathcal{SO}_3$ represents a rotational matrix from world frame to body frame, $\mathbf{f}_d = [0, 0, f_d]^T$ is a desired output collective thrust force, $\mathbf{f}_e = [f_{ex}, f_{ey}, f_{ez}]^T$ presents an external disturbance force, $9.81\,\mathrm{m\,s^{-2}}$ is a magnitude of gravitational acceleration, $\mathbf{e}_3$ is a z-axis unit vector, $\mathcal{I} \in \mathbb{R}^{3\times3}$ is vehicle's moment of inertia, $S(\mathbf{x})\mathbf{y}$ is a skew symmetric operator and it equals $\mathbf{x} \times \mathbf{y}$, $\mathbf{r}_g = [x_g, y_g, z_g]^T$ is a position on center of in the body frame, $\boldsymbol{\omega}$ is the vector of angular rates $[p, q, r]^T$ in the body frame, $\mathbf{m} = [m_e, m_e, m_e]^T$ is the vector of controlled torque and $\mathbf{m}_e = [m_{ex}, m_{ey}, m_{ez}]^T$ is the external torque in the body frame.

The equation (4.4) describes translational dynamics and comes from the 2nd Newton's law. The rotational dynamics are described by (4.5) and (4.6) describes the rotational matrix $\mathbf{R}$ evolution. Based on the [29], dynamics of the vehicle are simplified to a single rigid-body description, omitting equation (4.5). It is mostly due to a fact that quadcopter uses smaller fixed-pitched propellers. Omitting part of the dynamic lowers the accuracy of the model, but it was already proved to provide sufficiently. Furthermore, as mentioned in [14], the system is treated as decoupled.

The next assumption is made about the inner stabilization loop. Using an autopilot, such as Pixhawk, allows simplifying the model as the autopilot provides an attitude stabilization. Hence, instead of providing the desired thrust to all four propellers, the inputs can be a desired roll rate (p_d), pitch rate (q_d), yaw rate (r_d) and a collective thrust (U_d). If the autopilot is properly tuned, it converts input to the output as a first-order transfer function [10]. Equations 4.7-4.10 describing this behavior:

$$\frac{\mathcal{L}\{q\}}{\mathcal{L}\{q_d\}} = \frac{K_1}{T_1 s + 1}, \tag{4.7}$$

$$\frac{\mathcal{L}\{p\}}{\mathcal{L}\{p_d\}} = \frac{K_2}{T_2 s + 1}, \tag{4.8}$$

$$\frac{\mathcal{L}\{U\}}{\mathcal{L}\{U_d\}} = \frac{K_3}{T_3 s + 1}, \tag{4.9}$$

$$\frac{\mathcal{L}\{r\}}{\mathcal{L}\{r_d\}} = \frac{K_4}{T_4 s + 1}. \tag{4.10}$$

To include this dynamics to the model, it was converted back to the time domain via the inverse Laplace transformation. Differential equation for pitch rate q evolution is then

$$\dot{q} = -\frac{q}{T_1} + \frac{K_1}{T_1} q_d. \tag{4.11}$$

The dynamics of the collective thrust U(4.9) is not well defined. The thrust itself is not a measured variable. An important variable that is needed in the model is collective thrust force f_d, which represents an actual force exerted on the vehicle. As mentioned in [29], collective thrust force can be simplified to the quadratic function of the angular speed of the propellers. The conversion of the thrust U to collective thrust force f_d can be then approximated as

$$f_d = \left(\frac{U - b_t}{a_t}\right)^2. \tag{4.12}$$

This approximation is necessary to provide the correct model in all circumstances, including takeoff and landing [10]. However, for the purpose of the linear model, it is expected that also the thrust works only around an equilibrium point. The dynamic is therefore simplified to conversion from desired thrust U_d to desired acceleration $\ddot{z}_d$,

$$\frac{\mathcal{L}\{\ddot{z}_d\}}{\mathcal{L}\{U_d\}} = \frac{K_3}{T_3 s + 1}. \tag{4.13}$$

Following sections further focus on a linear model which includes already the linearized form of the equations. Complete list of the nonlinear equations is depicted in the section 8.2.2. The linear model is necessary to allow implementation of LKF. The equilibrium point was selected as hover point, where

$$\phi = 0,\ p = 0,\ \theta = 0, q = 0,\ \psi = 0,\ r = 0,\ f_d = mg,\ f_x = 0,\ f_y = 0,\ f_z = 0. \tag{4.14}$$

It is expected that the absolute values of the roll and pitch will not be large.

4.2.1 Attitude system

The attitude system describes dynamics on the horizontal plane. It concerns the dynamics connected with roll and pitch. Complete states of the subsystem around x-axis are $\mathbf{x}_x = (x, \dot{x}, \theta, q, f_x)$ with the input $\mathbf{u}_x = q_d$. Similarly, y-axis - $\mathbf{x}_y = (y, \dot{y}, \phi, p, f_y)$ with the input $\mathbf{u}_y = p_d$. By calculating the 4.4, the equations describing the attitude dynamics are

$$\ddot{x}^{(\mathcal{W})} = g\theta + \frac{1}{m} f_x, \tag{4.15}$$

$$\ddot{y}^{(\mathcal{W})} = -g\phi + \frac{1}{m} f_y. \tag{4.16}$$

By combining these equations with the differential equations of the attitude (4.7, 4.8), the continuous state space representation of linearized subsystem is then following

$$\mathbf{A}_x = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & g & 0 & \frac{1}{m} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T_1} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{B}_x = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{K_1}{T_1} \\ 0 \end{pmatrix}, \mathbf{A}_y = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & \frac{1}{m} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T_2} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{B}_y = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{K_2}{T_2} \\ 0 \end{pmatrix}. \tag{4.17}$$

4.2.2 Altitude system

The equations are similar to the attitude system. States of the z-axis subsystem are $\mathbf{x}_z = (z, \dot{z}, \ddot{z}_d, f_z)$ with the input $\mathbf{u}_x = U_D$. Differential equation given by 4.4 is

$$\dddot{z}^{(\mathcal{W})} = \dddot{z}_d + \frac{1}{m} f_z. \tag{4.18}$$

Notice, that the equation does not include gravitational acceleration g. It is given by the choice of the equilibrium point, which is hovering and the vehicle already counteracts the gravity. The conversion of desired thrust U_d was simplified as described above by equation 4.13 and its differential equation is

$$\dddot{z}_d = -\frac{\ddot{z}_d}{T_3} + \frac{K_3}{T_3} U_d, \tag{4.19}$$

where the $\ddot{z}_d$ is desired acceleration in the body frame. Here it also represents only the divergence from the necessary acceleration to hover the vehicle. By combining above equations, the state space representation is

$$\mathbf{A}_z = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{m} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{B}_z = \begin{pmatrix} 0 \\ 0 \\ \frac{K_3}{T_3} \\ 0 \end{pmatrix}. \tag{4.20}$$

4.2.3 Yaw subsystem

The last system describes the behaviour of the yaw (ψ) angle. It is simpler compared to the previous systems and contains only two states $\mathbf{x}_\psi = (\psi, r)$ with input of $\mathbf{u}_\psi = r_d$. Corresponding state space representation is

$$\mathbf{A}_\psi = \begin{pmatrix} 0 & 1 \\ 0 & \frac{1}{T_4} \end{pmatrix}, \mathbf{B}_\psi = \begin{pmatrix} 0 \\ \frac{K_4}{T_4} \end{pmatrix}. \tag{4.21}$$

4.2.4 Overall system

Connection of described subsystems creates complete model of the UAV. It is described as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_x & 0 & 0 & 0 \\ 0 & \mathbf{A}_y & 0 & 0 \\ 0 & 0 & \mathbf{A}_z & 0 \\ 0 & 0 & 0 & \mathbf{A}_\psi \end{pmatrix}, \mathbf{B} = \begin{pmatrix} \mathbf{B}_x & 0 & 0 & 0 \\ 0 & \mathbf{B}_y & 0 & 0 \\ 0 & 0 & \mathbf{B}_z & 0 \\ 0 & 0 & 0 & \mathbf{B}_\psi \end{pmatrix}. \tag{4.22}$$

4.3 Discretization

For the purpose of this thesis, the system needs to be discretized. Following equation describes the necessary conversion continuous and discrete system,

$$\dot{\mathbf{x}} = \mathbf{A}_c \mathbf{x} + \mathbf{B}_c \mathbf{u} \quad \rightarrow \quad \mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k, \quad k \in \mathbb{Z}, \tag{4.23}$$

where subindex **c** denotes continuous and **d** discrete. Based on the nature of the problem and the computational sources of the Raspberry Pi, the sampling rate is set to $\Delta t = 0.01\,\mathrm{s}$. The discretization method that is used is called Forward-Euler method and the continuous system can be converted by following

$$\mathbf{A}_d = (\mathbf{I} + \Delta t \cdot \mathbf{A}_c), \quad \mathbf{B}_d = \Delta t \cdot \mathbf{B}_c. \tag{4.24}$$

5 System identification

In order to find the unknown model parameters, the system has to be identified. This requires performing test flights to obtain real data of the system. There are several methods how to obtain identification parameters. The classical methods divide into two categories: based on the time and frequency domain. The principal behind the identification is to excite all possible poles and zeros, which defines the dynamics of the system. Step response or frequency response is the most common identification method. However, to use them, it is expected that the system is stable. This condition does not hold for the purpose of UAV and using such methods might cause damage to the system and its surroundings. Therefore, using another method is preferable. Another option is to use an optimization-based method, where the input data are fitted to the resulting output.

Linear model that is described in previous chapter 4 has eight unknown parameters $(K_1, T_1, \ldots, K_4, T_4)$. Parameters are coefficients of first-order transfer functions, which makes it four pairs and thanks to the complete decoupling of the system, each pair of the parameters can be identified separately. Test flights that were performed for the purpose of identification were operated by a human operator, and data were recorded from an onboard Pixhawk autopilot. In order to excite all dynamics of the system, it would be ideal for adding a white noise (zero-mean signal) to the reference input. Unfortunately, it is not possible in the case of manual flying.

5.1 Optimization method

The optimization method based on least squares method was used instead as an identification method. Main advantage of this method is its universality. Any data can be used for the purpose of the identification and no given maneuver, such as step, is not necessary. Although the system might not be excited enough, it is expected that it will be sufficient for the purpose of the LKF. Because the equations that are to be fitted are first order transfer functions, it can be rewritten to following form

$$\mathbf{x}_{k+1} = P\mathbf{x}_k + Q\mathbf{u}_k, \tag{5.1}$$

where P, Q are unknown parameters, that are to be find.

A method that can solve similar problems is called *Least squares method*. It is a type of regression method which solves overdetermined system of equations, where is more equations than variables. It works by minimizing the sum of squares of the residuals $\mathbf{r}$. Residuals are defined as

$$\mathbf{r} = \mathbf{A}\mathbf{q} - \mathbf{b}, \tag{5.2}$$

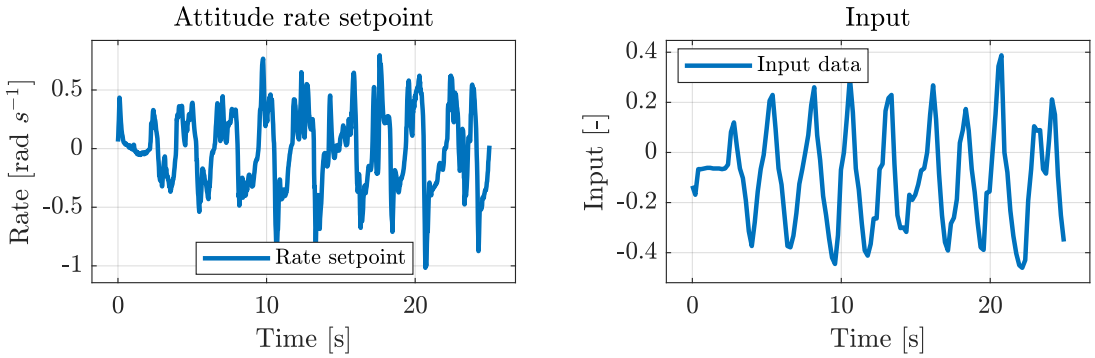


Figure 5.1: Input data for the y-axis identification. The graph on the right depicts the RC input that corresponds to the roll angle and left shows a generated attitude rate setpoint.

where $\mathbf{A} \in \mathbb{R}^{(n-1)\times 2}$, $\mathbf{B} \in \mathbb{R}^{(n-1)\times 1}, \mathbf{q} \in \mathbb{R}^{2\times 1}$ for n equals number of measurements. More precisely the parameters P, Q are obtained as

$$\mathbf{q} = \begin{pmatrix} P, & Q \end{pmatrix}^T, \quad \mathbf{A} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{u}_1 \\ \mathbf{x}_2 & \mathbf{u}_2 \\ \vdots & \vdots \\ \mathbf{x}_{n-2} & \mathbf{u}_{n-2} \\ \mathbf{x}_{n-1} & \mathbf{u}_{n-1} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_{n-1} \\ \mathbf{x}_n \end{pmatrix}. \tag{5.3}$$

5.1.1 Attitude system

The attitude system describes dynamics along the horizontal plane. If the mechanical design was symmetrical in all directions, it would be possible to assume that the attitude subsystems are identical. However, it is not satisfied in this work, and both attitude subsystems were identified separately.

Several test flights were made to gather data. Although the reference coming from the RC controller is a desired attitude, not attitude rate, the Pixhawk autopilot solves this internally. It includes an own attitude controller that takes this attitude reference on input and controls attitude by producing an attitude rate setpoint, that is further internally process to control the vehicle. Therefore, input data that were used to identify the y-axis subsystem (figure 5.1) depicts both the RC-command and attitude rate setpoint. In order to obtain the parameters, both the attitude rate and attitude rate input signal had to be filtered first as it included a large noise component. Then, the least squares method was used to obtain first estimate of parameters which was then empirically tuned to find the best fit:

$$K_2 = 0.9255, T_2 = 0.0580. \tag{5.4}$$

The subsystem was further tested by comparing an open-loop estimate with measured data. The result is in figure 5.2. Data obtained in open-loop fit very well the attitude rate and the attitude. But a comparison of velocity and position does not fit very well. It is caused by the bias in the attitude angle, which then propagates to the velocity and position. Notice that the first integration of this bias results in ramp drift, and another integration

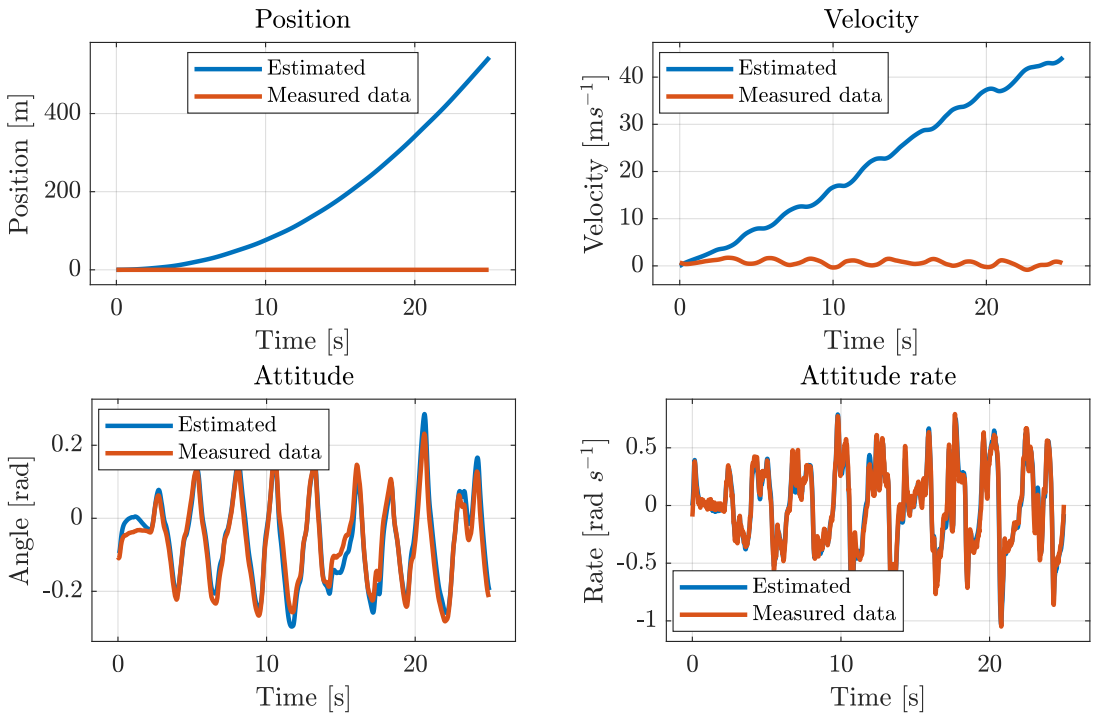


Figure 5.2: Comparison of the estimated values and measurements on the y-axis.

into position makes the drift squared. This, however, is not a problem as the proper filtering method (chapter 6) can work with bias.

The subsystem of the x-axis was identified in the same way. The input data can be seen in figure 5.3. Parameters that we obtained from this identification are

$$K_1 = 0.2255, T_1 = 0.0580. \tag{5.5}$$

Figure 5.4 then depicts the comparison of estimated open-loop values with the measured values. In this case, the result fits worse than the y-axis subsystem. Although the attitude rate fits very well, the attitude already shows a slight drift. This error further propagates to the velocity and position.

5.1.2 Altitude system

For the purpose of the altitude subsystem, the UAV was equipped with a laser range finder. Figure 5.5 shows data that were used to identify the altitude subsystem parameters. In this situation, the input signal was shifted due to the thrust necessary to compensate the gravity acceleration. The obtained values are:

$$K_3 = 14.0100, T_3 = 0.1545. \tag{5.6}$$

It can be seen in figure 5.6, that the acceleration fits very well. However, the position and the velocity integrate again some bias included in the acceleration, which makes the estimated value drift significantly.

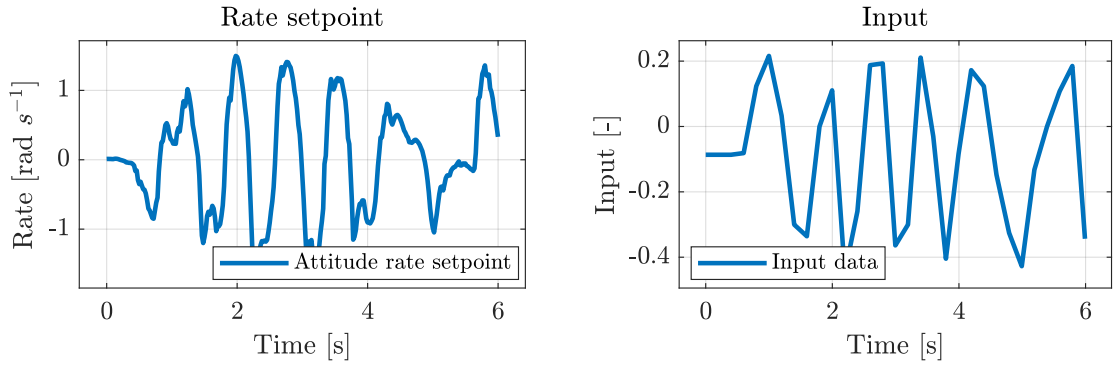


Figure 5.3: Input data for the purpose of x-axis identification. RC input is depicted in the graph on the right and generated attitude rate command is on the left.

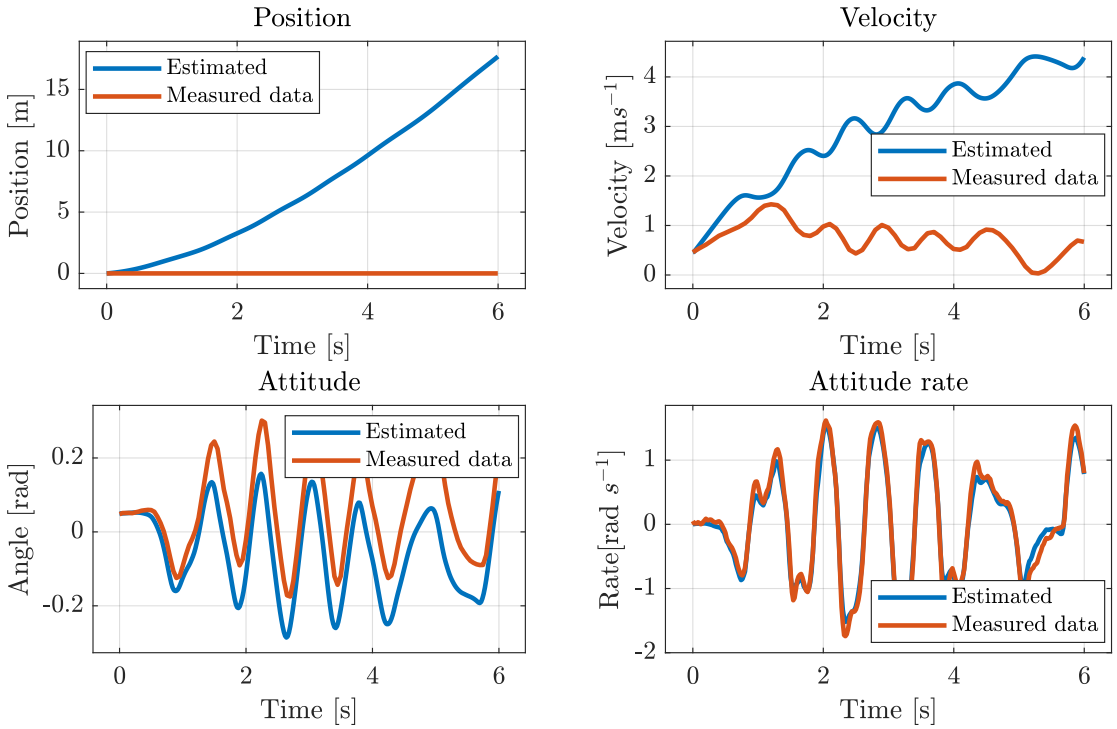


Figure 5.4: Graphs comparing estimations of the identified system and the measured data provided by Pixhawk autopilot.

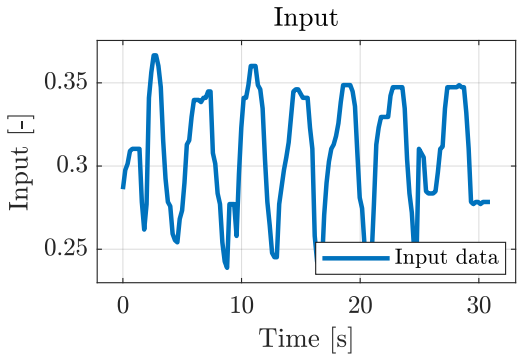


Figure 5.5: Identification data for the altitude subsystem. Graph shows the RC input data that corresponds to acceleration in UAV's z-axis.

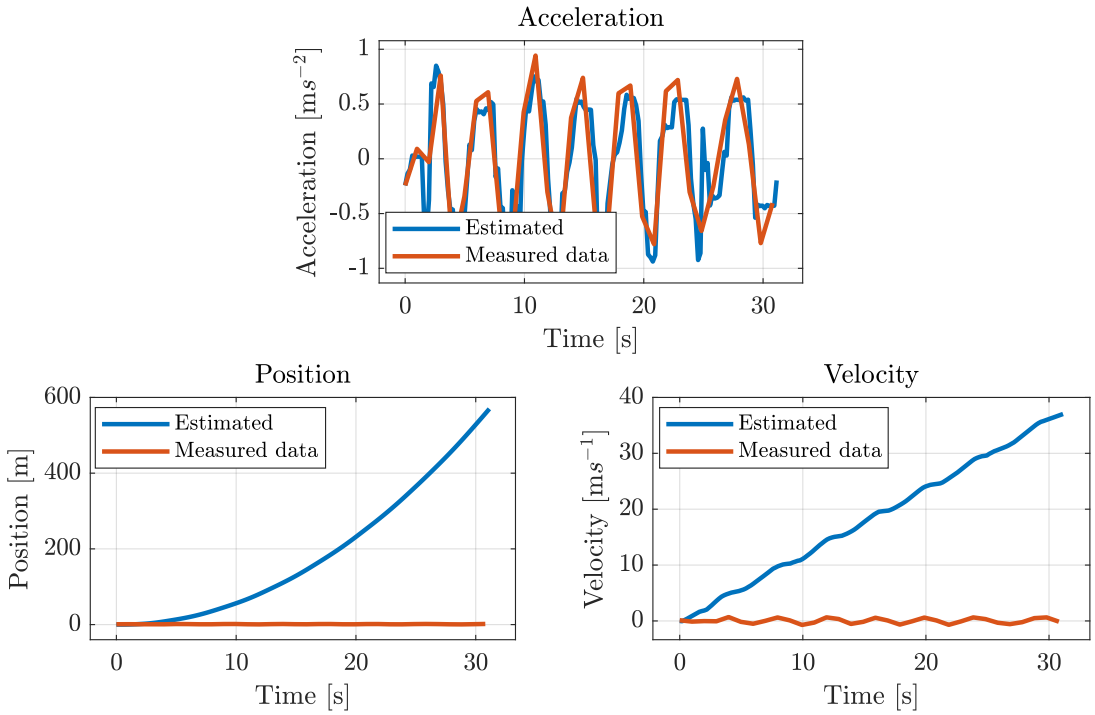


Figure 5.6: Resulting comparison of the estimations and the measured data of the altitude subsystem.

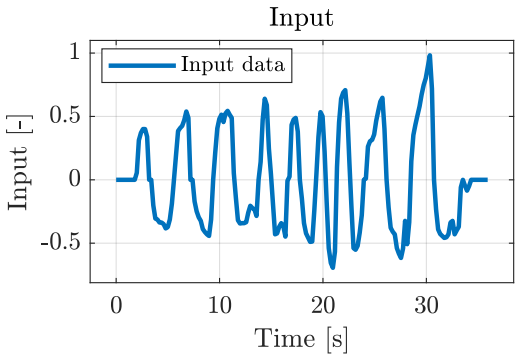


Figure 5.7: RC input data for the yaw subsystem identification, the input directly corresponds to the yaw rate r_d command.

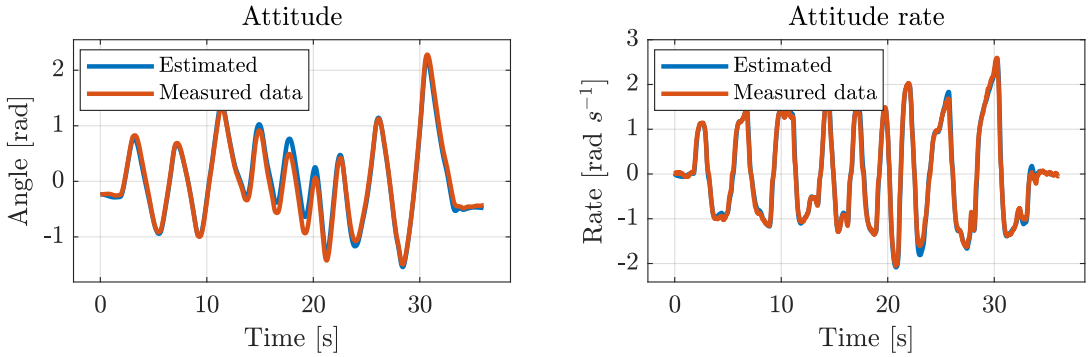


Figure 5.8: Comparison of estimates generated by identified model and measured data of the yaw subsystem.

5.1.3 Yaw subsystem

The last system describes the behaviour of the yaw (ψ) angle. This subsystem was identified in the exact same way as the previous subsystem. Input data that were gathered for the purpose of identification are in figure 5.7. Parameters K_4, T_4 were identified to be

$$K_4 = 1.0011, T_4 = 0.1089. \tag{5.7}$$

Comparison of the open-loop estimated data can be seen on figure 5.8. The attitude rate fits very well. The integrated attitude angle holds the shape of the curve nicely, but it has a significantly different magnitude.

5.1.4 Conclusion

This chapter presented an identification of the dynamics of the UAV. Several test flights were conducted to gather the necessary data. Data that were used were obtained by the onboard Pixhawk autopilot. Pixhawk itself provides all the state information, but only the accelerations, attitude, attitude rate and height are measured. The rest of the states is only estimated. The estimated states were included for the purpose of verifying the identified

model. All the parameters were identified, and the comparison of the open-loop estimation with Pixhawk data was provided. The comparison of the velocity and the position did not fit very well, which is given by the fact that it was estimated by the integration of attitude rate. Therefore, all the noise and bias components included in the measurement were amplified and propagated further. This behaviour is normal and will be solved by closing the control loop and using the proper state estimation method (chapter 6).

6 State estimation

Providing precise state information is critical for reliable feedforward control of fast unstable dynamical systems, such as the multirotor UAV. Data from sensors are often not directly usable as they can be noisy, or the value can be biased. Using the raw sensor data directly for a control might conclude in crashing the vehicle. There are also scenarios when a certain part of the information is not directly measured (i.e. cannot be measured) and also cannot be easily derived from other data. At such a moment, there is a need for the implementation of a state estimation method. Observation methods provide state estimates based on input data and measurements coming from the sensors. These methods can be divided based on the approach to the system, as deterministic or stochastic, and also based on the system model form, as linear or nonlinear.

This chapter focuses on methods of state estimation. It describes classical deterministic state observer and then moves to the well-known Kalman filter (KF). KF has many variants. Therefore, first is described the general concept of this filter and then presented linear version, Linear Kalman filter (LKF) and nonlinear solution, Unscented Kalman Filter (UKF).

6.1 State observer

If the linear system is considered deterministic, the observation method is called *State observer*. It means that there is no randomness in the development of the system states, and therefore certain input sequences will always result in the same output. Let us have a LTI discrete system,

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{6.1}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k. \tag{6.2}$$

If there is no feedback connected to this system, equation 6.1 can be considered as an open-loop state observer. Observed state values are usually denoted as $\hat{\mathbf{x}}_k$. Hence, to be precise, state vectors in 6.1 should be changed to this notation to present the state observer. This observer expects a perfectly identified model. Otherwise, it will not work. Any disturbance in the measurements moves the prediction of the system away from the real state. It may then happen that the observation returns completely wrong data.

This can be improved by closing the loop and connecting the data back. Such a system is called a closed-loop state observer and is depicted in 6.3. By creating the feedback loop, the original system 6.1 extends for the correction part given by multiplication of difference of real system output $\mathbf{y}$ and the observed output $\hat{\mathbf{y}}$. The closed-loop observer has following form

$$\begin{aligned} \hat{\mathbf{x}}_{k+1} &= \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}\mathbf{u}_k + \mathbf{L}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}), \\ \hat{\mathbf{y}}_k &= \mathbf{C}\hat{\mathbf{x}}_k + \mathbf{D}\mathbf{u}_k, \end{aligned} \tag{6.3}$$

where the matrix **L** is so-called Luenberger's observer. If the system matrices **A** and **C** observable, the eigenvalues of the new system matrix $(\mathbf{A} - \mathbf{KC})$ can be shifted anywhere and make system asymptotically stable. There are methods for placing the desired eigenvalues:

- Direct method - The desired characteristic polynomial is compared with a characteristic polynomial of the closed loop system, parameters of **K** are then derived from system of equations,
- Transformation to the observable form - If the system can be converted into a observable canonical form, it can be used to place poles to desired location,
- Systems duality - Placing poles of the observer represents a dual problem to placing poles of the state feedback. Therefore duality principle can be used and match system matrices as transposed matrices of the state observer.

A necessary condition to place all the eigenvalues arbitrarily is that the system is observable. However, asymptotic stability is possible to reach still if the system is only detectable. It means that the system includes non-observable poles, but these poles are stable. State observer at this form does not work for nonlinear systems.

6.2 Kalman Filter

Although the state observer might be useful, most real-world systems are not deterministic. Even a simple system, such as a resistor, has only part of its range that can be approximated linearly but then becomes highly nonlinear. Therefore different observation methods have to be used. Every measurement variable entering the system usually includes a certain amount of noise, which makes tracking such a system further difficult. A very frequent method that provides a solution for stochastic systems is *Kalman filter*. The Kalman filter is a recursive algorithm that provides information about state estimates $\hat{\mathbf{x}}_{t+\tau}$ in a way that minimises a squared error. Given the application, KF can be used to smooth past data ($\tau < 0$), filter present data ($\tau = 0$) or also for future state prediction ($\tau < 0$). It is widely used in engineering or economics and can be found in trajectory tracking, navigation, feature tracking or control systems. KF has multiple different variants and can work even with nonlinear systems.

The conceptual solution of the filter is common for all further variants, and it is based on probability theory. Assuming that the noises in the system are Gaussian, the system state can be treated as random variable X and can be described by normal (Gaussian) distribution. This is described by notation

$$X = \mathcal{N}(\hat{x}, \sigma^2) \tag{6.4}$$

which indicates that system state X is Gaussian random variable with mean value $\hat{x}$ and variance of σ^2. The idea behind the KF is to treat the system states as random variables and represent them by probability density function (pdf) given by 6.4. Then propagate it as the system evolves from the model and correct the development by data from sensors.

In the context of a stochastic system, the mean value is then given by a deterministic part of the model and uncertainty of the variables is given by covariance matrices. The

deterministic part of the equation is the same as in the state observer (equation 6.1). If the condition of Gaussian white noise holds, state estimate can be represented by a conditional pdf 6.5, where $\mathbf{x}$ is a data vector and $\mathcal{D}^{t-1}$ represents past data information. The conditional mean value equals the mean square (MS) estimate [30], which minimises means square error and it represents all the necessary state information. The value given by following equation is often called *apriori* information,

$$p(\mathbf{x}(t)|\mathcal{D}^{t-1}) = \mathcal{N}(\hat{\mathbf{x}}, \mathbf{P}). \tag{6.5}$$

The noise in the system is divided into process noise and measurement noise, and the condition of white, zero mean-value and uncorrelated noise must hold to reach an optimal performance of the filter.

The KF algorithm itself can be divided into two steps; data update, also called filtering step and time update, known as prediction step. Assuming that the data are initiated by the equation 6.5. Filtering step represents an update of predicted the state information given the data to the current step. Its probabilistic definition can be described as

$$p(\mathbf{x}(t)|\mathcal{D}^{t}) = p(\mathbf{y}(t)|\mathbf{x}(t), \mathbf{u}(t)) \cdot p(\mathbf{x}(t)|\mathcal{D}^{t-1}), \tag{6.6}$$

where $\mathbf{y}(t), \mathbf{u}(t)$ are measurements and input data respectively. The product combines an output equation density function and apriori state information given by past data.

The next step makes a shift in the time step. It predicts next state estimate and is given by equation following equation

$$p(\mathbf{x}(t+1)|\mathbf{x}(t), \mathcal{D}^{t}) = p(\mathbf{x}(t+1)|\mathbf{x}(t), \mathbf{u}(t), \mathbf{y}(t)). \tag{6.7}$$

Note that next step prediction does not need past data $\mathcal{D}^t$ and simplifies to using only $\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t)$. This allows for the recursive character of the filter as only the last system state estimate is needed to remember. The prediction step is given by the differential equations of the system.

6.2.1 Linear Kalman filter

Consider a discrete linear stochastic system,

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{v}_k, \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k + \mathbf{e}_k. \end{aligned} \tag{6.8}$$

where $\mathbf{v}_k$ denotes process noise and the similarly output includes measurement noise $\mathbf{e}_k$. For the purpose of optimality [31], the noises have to be white, zero-mean, uncorrelated and have known covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$,

$$\begin{aligned} \mathbf{v}_k &\sim (0, \mathbf{Q}_k), \\ \mathbf{e}_k &\sim (0, \mathbf{R}_k), \\ E[\mathbf{v}_k \mathbf{v}_j^T] &= \mathbf{Q}_k \delta_{k-j}, \\ E[\mathbf{e}_k \mathbf{e}_j^T] &= \mathbf{R}_k \delta_{k-j}, \\ E[\mathbf{v}_k \mathbf{e}_j^T] &= 0, \end{aligned} \tag{6.9}$$

where δ_{k-j} is the Kronecker delta. $E[-]$ represents the mean value of a variable. Covariance matrices has to be symmetric positive semi-definite, $\mathbf{Q}_k, \mathbf{R}_k \succeq 0$ [30].

By combining the joint pdf for the system state and output equation with equations for finding conditional values of normally distributed variables, the correction step has the following form,

$$\begin{aligned}\mathbf{K}_t &= \mathbf{P}_{t|t-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{t|t-1}\mathbf{C}^T + \mathbf{R})^{-1},\\ \hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{C}\hat{\mathbf{x}}_{t|t-1} - \mathbf{D}\mathbf{u}_t),\\ \mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{t|t-1}\mathbf{C}^T + \mathbf{R})^{-1}\mathbf{C}\mathbf{P}_{t|t-1}.\end{aligned} \tag{6.10}$$

The matrix $\mathbf{P}$ represents a covariance matrix of the new density function. The prediction step describes the evolution of both the system state and uncertainty in time. System state evolution is based on the systems natural evolution, and covariance update comes from the Lyapunov equation. It is denoted as

$$\begin{aligned}\hat{\mathbf{x}}_{t+1|t} &= \mathbf{A}\hat{\mathbf{x}}_{t|t} + \mathbf{B}\mathbf{u}_t,\\ \hat{\mathbf{P}}_{t+1|t} &= \mathbf{A}\mathbf{P}_{t|t}\mathbf{A}^T + \mathbf{Q}.\end{aligned} \tag{6.11}$$

This closes the LKF iteration and completes the algorithm.

LKF is a very powerful tool and is very frequently applied in the real-world systems. Even though the condition of white Gaussian noises is undoubtedly problematic, LKF is still the best linear state estimator even if the condition does not stand [30].

6.2.2 Nonlinear Kalman Filter

As already mentioned, linear systems do not exist. That is why there is a need to use more advanced filters based directly on the nonlinear systems. The general form of the nonlinear system is described as follows.

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_t), \tag{6.12}$$

$$\mathbf{y}_k = g(\mathbf{x}_k, \mathbf{u}_k, \mathbf{e}_t), \tag{6.13}$$

where the function $f()$ describes state transition function and $g()$ represents output function.

The most widespread nonlinear state estimator is the Extended Kalman Filter [30]. Its principal is similar to LKF and also uses the linearised model to update and predict states. Nevertheless, instead of using one fixed linearised model, EKF constantly creates a new linear approximation based on the estimate of the current state. Although it is not an optimal nonlinear filter, it is usable, as optimal filters are usually on infinite-dimensional [30]. EKF proved to be useful in many applications and still is frequent in systems. But it is difficult to tune this filter, and its estimates can be often unreliable as there is always an inevitable linearisation error in the system. There are also improved filters based on EKF, that are attempting to minimise this error. For example, iterative EKF, which iterates during the data update step to provide the best posteriori estimate, or higher-order EKF, that higher elements from Taylor series when approximating state and output equations. But the problem with linearisation error persists.

A completely different approach is presented with the Unscented Kalman Filter (UKF) algorithm. This algorithm was first presented in [32]. Its key contribution lies in a different approach to propagation of the mean and covariance. Instead of using linear approximation, it uses the unscented transformation for this purpose. This transformation is based on two principles; first, it is easy to transform a single point rather than whole pdf. Second, for a given pdf it is relatively simple to find a sample of points that represents its proper form. Let us have an estimate of $\hat{\mathbf{x}}$ with its covariance $\mathbf{P}$. Given this information, it is possible to find a set of deterministic vectors called σ-points, which represents current probability density function (pdf). To get the transformed density function, each σ-point is transformed separately and from transformed points then restored the transformed pdf. This process is a key principle behind the unscented transformation.

$\sigma - points$ are defined as

$$\begin{aligned} \mathbf{x}_0 &= \hat{\mathbf{x}} \ ; \ w_0 = \frac{k}{n+k}, \\ \mathbf{x}_i &= \hat{\mathbf{x}} + \mathbf{S}_i \ ; \ w_i = \frac{1}{2(n_x+k)} \ ; \ i = 1 \dots n_x, \\ \mathbf{x}_i &= \hat{\mathbf{x}} - \mathbf{S}_i \ ; \ w_i = \frac{1}{2(n_x+k)} \ ; \ i = n+1 \dots 2n_x, \end{aligned} \tag{6.14}$$

where $\mathbf{x}_i$ are the σ-points, n_x is size of state vector, k controls the distance of sigma points from the mean $\hat{\mathbf{x}}$, w_i are weighting coefficients, $\mathbf{S}_i$ are direction vectors of weighted covariance matrix $\mathbf{P}$ and can be obtained as $(n_x + k)\mathbf{P} = \mathbf{S}^T\mathbf{S}$. Matrix $\mathbf{S}$ is also know as Cholesky factor and the direction vectors are its columns. The process of obtaining the transformed parameters of new pdf are

$$\hat{\mathbf{x}} = \sum_{i=0}^{2n_x} w_i \mathbf{x}_i, \quad \mathbf{P}_{xx} = \sum_{i=0}^{2n_x} w_i (\mathbf{x}_i - \hat{\mathbf{x}})(\mathbf{x}_i - \hat{\mathbf{x}})^T. \tag{6.15}$$

The knowledge of this transformation allows to introduce the UKF algorithm. It is again based on the filtration and prediction step. In case of the correction step, lets assume that predicted state $\hat{\mathbf{x}}_{t|t-1}$ and its covariance $\mathbf{P}_{t|t-1}$ are known. σ-points $\mathbf{x}_i^\sigma$ are transformed by equation $\mathbf{y}_i = g(\mathbf{x}_i^\sigma)$ and then used to compute necessary components to update actual state with its covariance,

$$\begin{aligned} \hat{\mathbf{y}}_k &= \sum_{i=0}^{2n_x} w_i \mathbf{y}_i \\ \mathbf{P}_{yy} &= \sum_{i=0}^{2n_x} w_i (\mathbf{y}_i - \hat{\mathbf{y}})(\mathbf{y}_i - \hat{\mathbf{y}})^T + \mathbf{R} \\ \mathbf{P}_{xy} &= \sum_{i=0}^{2n_x} w_i (\mathbf{x}_i^\sigma - \hat{\mathbf{x}})(\mathbf{y}_i - \hat{\mathbf{y}})^T. \end{aligned} \tag{6.16}$$

The prediction step is designed similarly. At this case, assume known corrected state $\hat{\mathbf{x}}_{t|t}$ and its covariance $\mathbf{P}_{t|t}$. Required transformation to new state prediction is done by generating new σ-points and transforming them through the state transition equation (6.12), as $\mathbf{x}_{k+1} =$

$f(\mathbf{x}_i^\sigma)$. Resulting formulas for setting state prediction are

$$\begin{aligned}\hat{\mathbf{x}}_{k+1} &= \sum_{i=0}^{2n_x} w_i \mathbf{x}_i, \\ \mathbf{P}_{k+1} &= \sum_{i=0}^{2n_x} w_i (\mathbf{x}_i^\sigma - \hat{\mathbf{x}})(\mathbf{x}_i^\sigma - \hat{\mathbf{x}})^T + \mathbf{R}.\end{aligned} \tag{6.17}$$

The UKF is a great improvement compared to the more known EKF. Its main advantage is in the transformation of the probability density function, which does not require computing any derivatives. Only computationally difficult is in computing the direction vectors $\mathbf{S}_i$. Overall, the UKF is a great choice for the purpose of this thesis, where the studied model is highly nonlinear. Furthermore, successful implementation of UKF was already presented for the same purpose [14].

6.2.3 KF verification

Verification of the performance of the filter is an important step. The filter itself is sensitive, and many parameters can be tuned to optimise its behaviour. To verify a KF, there are multiple ways how to check the performance. The optimal general method would be to compare the filtered data to a ground truth value via a certain metric, such as Root mean square error (RMSE). Unfortunately, providing a ground truth is difficult. One option would be to use a system like motion capture that would provide very precise data. But even then, only available ground truth data would be position and 3D orientation. Velocity or attitude rates would have to be derived and would not be without error. There is no such system currently presented in the MRS group.

Nevertheless, the filter performance metric can be also defined by innovations and allow checking the consistency of the filter. The consistency of a filter is defined as an attribute that shows that the filter increases the accuracy with an increasing number of samples. Innovation is represented as

$$\boldsymbol{\epsilon}_k = \mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_{k|k-1}, \tag{6.18}$$

and it describes the amount of new information added to the filter. Similarly to the system states, the variance of innovations is represented by matrix $\mathbf{Z}$. It is defined as

$$\mathbf{Z}_k = E[\boldsymbol{\epsilon}_k \boldsymbol{\epsilon}_k^T] = \mathbf{C}\mathbf{P}_{t|t-1}\mathbf{C}^T + \mathbf{R}. \tag{6.19}$$

If the filter performs well, the innovation $\boldsymbol{\epsilon}_k$ should have zero mean value and the covariance $\mathbf{Z}_k$. Multiple tests can be then applied to check the performance of the filter [33],

- *Innovation magnitude bound test* - The first test checks if the innovations are consistent with innovations covariance. Innovations should be bounded by value $\pm 2\sqrt{\sigma_k^2}$. It shows that innovation is unbiased, and the bound itself represents the 95 % confidence level. Only less than 5 % innovations can cross the bound to satisfy the test.

- *Normalised innovations squared χ^2 test* - This test checks the unbiasedness of innovations through χ^2 test. For this purpose, the squared normalised innovations are computed as,

$$q_k(i) = \boldsymbol{\epsilon}_k(i)\mathbf{Z}_k^{-1}(i)\boldsymbol{\epsilon}_k(i), \tag{6.20}$$

  where i is the number of trials of KF. It is necessary to provide an estimate of sample mean $\bar{q}$ to test for the unbiasedness. That is done by averaging values of normalised innovations and creating a form of a moving average. This estimate of sample mean $\bar{q}$ is then tested on the confidence bound $[r_1, r_2]$. This interval comes from the hypothesis H_0, that is defined as

$$P(N\bar{q} \in [r_1, r_2]|H_0) = 1 - \alpha, \tag{6.21}$$

  where α is the 5 % outer bound. If value $N\bar{q}$ fits in this interval, the hypothesis is accepted, and the filter is unbiased.

- *Innovation whiteness test* - Last test, also known as autocorrelation test, is trying to show that time-averaged correlation

$$r(\tau) = \frac{1}{N}\sum_{k=0}^{N-\tau-1} \boldsymbol{\epsilon}_k^T \boldsymbol{\epsilon}_{k+\tau} \tag{6.22}$$

  is zero mean within allowable statistical error. This correlation should be normalised by $r(0)$. Apart from the peak at $\tau = 0$, all other values should be disturbed around zero mean value. The oscillations around zero value should be random. For large enough samples N, the variance can be represented as $1/N$. Hence the values should be within the gate of $\pm 2\sigma$. Again, at least 95 % should fit within this region

Although the theory is clear and should be simple to use these methods to tune and verify the filter, multiple factors make the process difficult. All the above theory expects that both the noise covariance matrices and model are known precisely. However, it is not usually correct in practice, and it is necessary to be aware of the different effects that the modelling imperfections. Following list summarizes the effect of wrong choice of the noise variance parameters,

- Underestimating σ_q or σ_r results in exceeding the 95 % confidence bound given by 2σ. Furthermore, the sample mean does not fit in the confidence bound given by χ^2 test because the normalised squared innovations are more significant than expected. The combined process and measurement noises levels are low.

- Overestimating σ_q or σ_r shows behaviour where the innovations are well within the 95 % confidence bound given by 2σ and do not even exceed the bound. The sample mean again does not fits the confidence bound given by χ^2 test The normalised squared innovations are lower than expected. The combined process and measurement noises levels are too high.

When dealing with a wrong model of the system that does not behave as expected, this model incorporates the filtration with an additional error. Such a problem is called *mis-matched filter problem*. This error might result in a drift of the mean value, which makes the data unreliable. A solution to this problem might be to increase the process noise covariance matrix $\mathbf{Q}$. This

should increase the Kalman gain and result in a more precise following of the measurements. However, this works only in the case of a small model mismatch. If the error were too large, the KF would not be able to compensate at all.

7 Admittance control

The physical interaction of humans and robots requires a specific control design that provides adequate actions. Several methods exist for the interaction of a robot with a mechanical environment, where a mechanical environment usually represents a human. Author in [12] sums the methods to indirect force control [34], Impedance control, Admittance control and full-state interaction [35]. The methods that are more examined in this work are Admittance and Impedance control. These methods are very similar, and they are often summed up as Impedance control [4]. In a certain way, they can be seen as the opposite of each other. The choice of the control depends on the relationship between the controller and the controlled system.

The key attribute that differs between these two control approaches is based on the definitions of *admittance* and *impedance* [4]. A system that takes motion as input and produces force is referred to as *impedance*. On the other hand, a system is defined as *admittance* when it receives force inputs and produces motion outputs. The definition of the control approaches is then simple. In the case of the Impedance control, the controller behaves as *impedance* and the controlled system as *admittance*, and for the Admittance control, it is other way around. Given this definition, Admittance control is suitable for this work. The UAV can be seen as a form of *impedance* that produces force, and the Admittance controller then processes this force to the resulting change of motion.

Admittance and Impedance control aims to establish a necessary action to satisfy a dynamic behaviour between reference trajectory and external force. This behaviour can be described as

$$\mathbf{M}_d(\ddot{\mathbf{r}}_d - \ddot{\mathbf{r}}_r) + \mathbf{D}_d(\dot{\mathbf{r}}_d - \dot{\mathbf{r}}_r) + \mathbf{K}_d(\mathbf{r}_d - \mathbf{r}_r) = -\mathbf{f}_e, \tag{7.1}$$

where $\mathbf{f}_e$ is vector of external force acting on the vehicle, $\mathbf{r}_d$ is a desired position vector, $\mathbf{r}_r$ is a reference position vector. Matrix $\mathbf{M}_d$ is a symmetric positive definite matrix that represents the desired inertia of the system. Matrices $\mathbf{D}_d, \mathbf{K}_d > 0$ are positive matrices that represents damping and stiffness, respectively [4]. A diagram showing the application of the controller is depicted in figure 7.1. In order to use this control technique for an interaction with a human, it is essential to choose the matrices correctly. This also involves defining the type of expected interaction and desired behaviour. For the purpose of physical interaction, it is acceptable to define desired acceleration $\ddot{\mathbf{r}}$ and velocity $\dot{\mathbf{r}_d}$ to zero as the control of the UAV will be purely driven by force. Furthermore, if the position of the UAV is expected to recover after an external push, the stiffness should be set $\mathbf{K} > 0$. On the other hand, setting $\mathbf{K} = \mathbf{0}$ results in entirely omitting a position and makes the control only for appropriate reference in velocity. In that case, the UAV can be freely dragged through space. In order to solve the (7.1), it can be seen as second order differential equation and transformed to

$$\ddot{\mathbf{r}}_r = -\mathbf{M}^{-1}\mathbf{D}\dot{\mathbf{r}}_r + \mathbf{M}^{-1}\mathbf{f}_e, \tag{7.2}$$

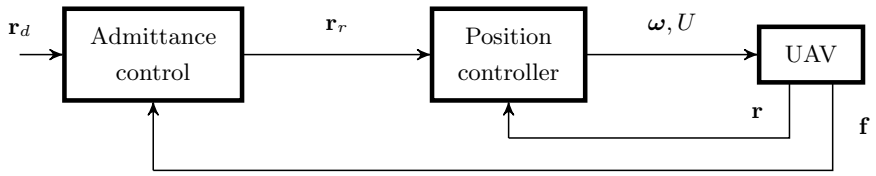


Figure 7.1: Overall pipeline of a general Admittance control [4]. The UAV produces a force vector $\mathbf{F}$ and information about current state $\mathbf{r}$. Admittance controller then processes the desired state $\mathbf{r}_d$ together with force $\mathbf{f}$ and produces a reference state $\mathbf{r}_r$ for a position controller. Controller then combines the reference with the actual state and generates control input the UAV, at this example attitude rate $\boldsymbol{\omega}$ and desired collective thrust U.

where $\dot{\mathbf{r}_r}$ can be seen as a state variable and $\mathbf{f}_e$ as input. This equation can be further differentiated and implemented into the control pipeline.

8 Implementation details

This chapter focuses on the implementation aspects First, the details about the simulation model that was created in the Gazebo simulator are described. Moreover, details about the Kalman filter implementation are presented, in particular the choice of the covariance matrices. The next part focuses on the Admittance control and its implementation into the MRS control system. The last part of the section mentions details about the real hardware model of the UAV.

8.1 Gazebo model

The hardware model of the UAV (described in chapter 3) was implemented the Gazebo simulator chapter 2. This model allowed to test the parts of this work safely in the virtual space and reduce the risk of breaking the real hardware. Resulting model is depicted in the figure 8.1. In order to make the model as realistic as possible, it was necessary to find characteristic parameters of the model. The key attribute is the characteristics of the motors. The simulation requires the knowledge of a real maximal thrust force to provide the best similarity to the real model. The derivation of these parameters is described in section 8.2.2. Other parameters as mass and the inertia matrix could be then either derived or explicitly measured.

Gazebo simulator provides a set of services that allow interaction with a model physically. These services were crucial and allowed for estimation and Admittance control verification. The most relevant service for this work is */gazebo/apply_body_wrench*. This service

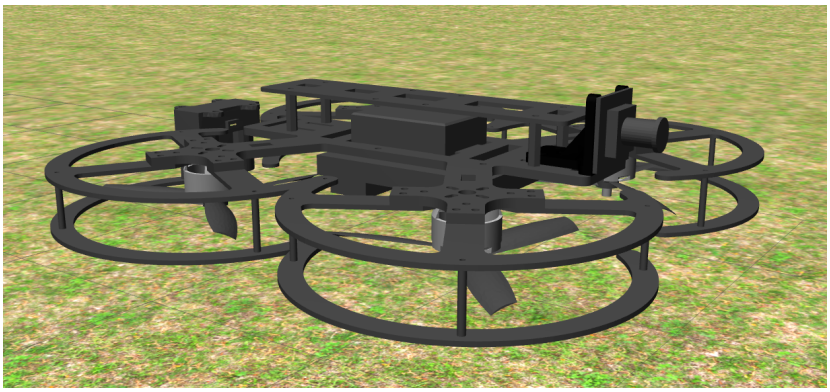


Figure 8.1: Simulation model of the UAV implemented in the Gazebo simulator. The model corresponds to the design described in chapter 3.

allows to apply force for a certain time duration to a specific part of the model and can be therefore used to simulate physical human interaction.

8.2 Kalman filter implementation

The Kalman filter was built based the model of the UAV dynamics described in chapter 4. Multiple data sources are used for the filtration because no sensor provides complete information about all the states. Information about the position and velocity were obtained from VIO, and information about the attitude and attitude rate was provided by Pixhawk autopilot. Furthermore, the estimators were extended to also include data from Global Positioning System (GPS) due to problematic behaviour of the VIO in certain situations. Both the LKF and UKF were implemented in such a way that it is possible to choose the source of the data before a flight.

An essential aspect concerning the KFs is tuning of the apriori information about process noise $\mathbf{Q}$ and the measurement matrices $\mathbf{R}$. The matrices should be positive definite,$\mathbf{Q}, \mathbf{R} \succ 0$, because they represent the covariance between variables. The choice of these matrices gives the filter its characteristics. Both the matrices are closely related, and therefore it is more about finding the right balance. On one hand, it might be desirable to provide well filtered smooth data, then values of $\mathbf{Q}$ should be much lower than $\mathbf{R}$. However, if it is more necessary to follow the measured data, the choice of the matrices should be vice versa. In practice, the matrices should be estimated from the measured sample of data. There are several methods for this purpose, the most known is Autocovariance least squares (ALS) and its derivations. Authors in [36] tested the performance of such methods and were comparing them to the maximum likelihood method. Although the methods give a reasonable estimate of the covariances, it is difficult to perform such an estimation. Therefore the covariance matrices at this work were tuned empirically.

The next significant factor in the KF process is the choice of initial state $\hat{\mathbf{x}}_0$ and state covariance matrix $\mathbf{P}_0$. As mentioned in [33], the initialisation is desirable for the LKF, but it is not essential as the system will take longer to settle. However, initialisation is critical in the case of the nonlinear UKF filter. One of the possibilities is to set the initial state information from the measurements $\mathbf{y}$ and the state covariance as the process noise matrix multiplied by a constant factor k. It can be written as follows

$$\hat{\mathbf{x}}_0 = \mathbf{y}, \quad \mathbf{P}_0 = k\mathbf{Q}, \tag{8.1}$$

where the factor is typically $k = 10$. This method was used to initialize both estimation algorithms in this work.

8.2.1 Linear Kalman filter

Linear Kalman filter was implemented based on the linear model described in chapter 4. This model was then identified in chapter 5. The noise matrices were empirically tuned to provide reliable information for the underlying controller. The process noise matrices are

following,

$$\mathbf{Q}_{x,y} = \begin{pmatrix} 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{Q}_z = \begin{pmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{8.2}$$
$$\mathbf{Q}_\psi = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.1 \end{pmatrix}.$$

The measurement covariance matrices are divided based on the type of measurement and are defined as

$$\mathbf{R}_{pos,vel,att} = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}, \mathbf{R}_{rate} = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix}. \tag{8.3}$$

Subscripts *pos* and *vel* indicates position and velocity measurements, respectively. These matrices were used for both the VIO and GPS measurements. Subscript *att* shows measurement covariances for attitude information coming form Pixhawk autopilot. Similarly, *rate* indicates Pixhawk measurement of attitude rate.

8.2.2 Unscented Kalman filter

Nonlinear equations presented in chapter 4 had to be unified and slightly modified for Unscented Kalman Filter implementation. The thrust dynamics are not simplified as in the LKF and includes now their whole dynamics, which comes from the assumption that desired force $f_t \sim \omega^2$. Instead of using direct conversion from input thrust $T \in [0, 1]$ to acceleration in z-axis $\ddot{z_u}$ (see (4.13)), it first uses (4.12) to convert the input thrust T to desired input acceleration $\ddot{z}_d$. It can be described as

$$\frac{\mathcal{L}\{\ddot{z}_u\}}{\mathcal{L}\{\ddot{z}_d\}} = \frac{K_5}{T_5 s + 1}. \tag{8.4}$$

To find the parameters $a_t,\ b_t$ included in the (4.12), a test flight had to be conducted. By putting multiple different weights to the UAV, making the vehicle hover and then reading the amount of thrust, pairs of (thrust, weight) are obtained. This information is then used to reconstruct the thrust-to-weight curve, which can be approximated, for example, by the least-squares method. The parameters that were obtained form the real experiment are following

$$a_t = 0.57787, b_t = -0.57853. \tag{8.5}$$

This curve is essential for modelling the UAV in the simulator. The necessary parameter, the maximal force produced by a single motor, can be found in the curve as the force with full throttle divided by the number of motors.

The input of the model is therefore different and is defined as $\mathbf{u} = (p_d, q_d, z_d, r_d)$. Complete state vector can be represented as $\mathbf{x} = (\mathbf{r}, \dot{\mathbf{r}}, \mathbf{R}, p, q, \ddot{z_u}, \mathbf{f}_e)$, where $\mathbf{r} = (x, y, z)$ is a

position vector, $\mathbf{R}$ is a rotation matrix, that can be represented by Tait-Bryan angles, ϕ, θ, ψ, the p, q, r are roll-rate, pitch-rate and yaw-rate, respectively. The differential equations are

$$\begin{aligned}
\dot{\mathbf{r}} &= \dot{\mathbf{r}}, \\
\ddot{\mathbf{r}} &= \mathbf{R}\mathbf{f}_d + \mathbf{R}\mathbf{f}_e + mg\mathbf{e}_3, \\
\dot{\mathbf{R}} &= \mathbf{R}S(\boldsymbol{\omega}), \\
\dot{\mathbf{f}}_e &= \mathbf{0}, \\
\dot{q} &= -\frac{1}{T_1}q + \frac{K_1}{T_1}q_d, \\
\dot{p} &= -\frac{1}{T_2}p + \frac{K_2}{T_2}p_d, \\
\dot{r} &= -\frac{1}{T_4}r + \frac{K_4}{T_4}r_d, \\
\dddot{z}_u &= -\frac{1}{T_5}\ddot{z}_u + \frac{K_5}{T_5}\ddot{z}_d.
\end{aligned} \tag{8.6}$$

Notice that parameters $T_1, K_1, T_2, K_2, T_4, K_4$ are the same as in the linear model and its values can be found in chapter 5. However, the change in the thrust dynamics introduced new parameters T_5, K_5, which were identified similarly to the other parameters. The process and measurement covariance matrices were selected the same parameters as described in section 8.2.1.

8.3 Admittance control

Implementation of the Admittance control is based on the details described in the chapter 7. The control itself does not work as the last element that outputs signals directly to the autopilot but instead produces position and velocity reference for the underlying position controller. The position controller that is used in this work is called *SE(3) Controller* and is based on [17]. The controller can handle different combinations of inputs, including position and velocity, which is suitable for this work. Given this structure, the Admittance control implementation is referred to as an Admittance tracker, not the controller.

It is expected that the physical interaction is exerted while the UAV is hovering in the space. Furthermore, the environment's stiffness is omitted, which means that the vehicle can be moved simply in the space. The expected complete situation is following. The vehicle is hovering in the space. When physical contact is sensed, the vehicle undertakes this force and starts to move in the direction of this force. Once the force is released, the UAV will fluently decrease its speed until it stops moving completely.

Detection level f_d is introduced to prevent reacting to even the smallest exerted force, or possibly to a force caused by other effects. Furthermore, the mentioned force level needs to be exceeded for a certain amount of time t_d. The Admittance tracker is designed as a simple state automaton to allow this behaviour. Figure 8.2 depicts graphically its structure. It consists of three states,

- *IDLE* state is active when the vehicle waits for a contact. It detects a present acting force, and once the magnitude of this force $\|\mathbf{f}\|$ overcomes the detection level f_d, it

starts to measure time. If the force is applied longer than the given time t_d, it switches to *FOLLOW* state. When this state is active, the tracker outputs a reference to a constant position.

- *FOLLOW* state includes an implementation of the Admittance control. It is initialised by the current speed of the vehicle and then modifies this velocity based on the applied force $\mathbf{f}$. This relationship is given by differential equation (7.2). At this state, the vehicle is controlled purely by the output reference velocity. It simultaneously keeps track of the magnitude of this force, and when the force is lower than mentioned detection level, it switches to *SLOW DOWN* state.
- *SLOW DOWN* state serve only as a transition between the above states. If the vehicle's speed is still large, switching suddenly to a position control with the current position might cause unwanted huge control action. Therefore, at this state, the velocity is continuously lowered by a factor of 0.99. Once the reference velocity $\dot{\mathbf{r}}_r$ is low enough, the state is switched back to the *IDLE*.

To provide smooth and responsive behaviour, it was necessary to tune correctly all parameters. Parameters concerning the transition between states were set to

$$f_d = 0.3\,\mathrm{N},\ t_d = 0.2\,\mathrm{s}. \tag{8.7}$$

This means that it is necessary to apply force greater than f_d for a time larger than t_d. Next, it was necessary to tune parameters concerning the behaviour of the admittance behaviour itself, it is described by differential equation (7.2). The parameters of inertia $\mathbf{M}$ and damping $\mathbf{D}$ were set to

$$\mathbf{M} = \begin{pmatrix} 0.9 & 0 & 0 \\ 0 & 0.9 & 0 \\ 0 & 0 & 0.9 \end{pmatrix},\ \mathbf{D} = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}. \tag{8.8}$$

It is necessary to compensate parasitic forces in order to control the vehicle solely by the external force. This force might be given by the controller error or possibly by the unbalanced vehicle. Therefore this part of the force is taken as offset and is removed internally in the Admittance tracker.

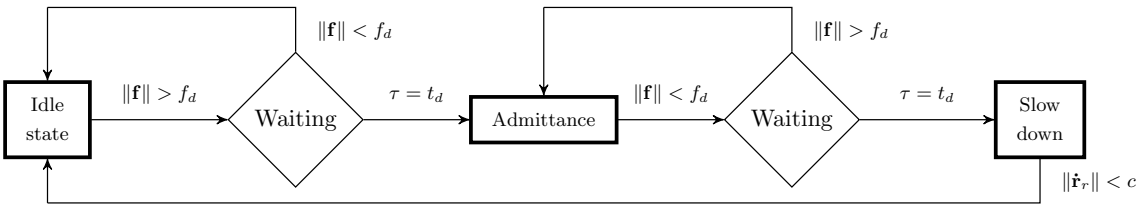


Figure 8.2: A flow chart describing an inner working of the Admittance tracker. It consist of three main states and two condition states. The condition states represent a timer and switch between main states happens only if the condition is satisfied for time t_d. f_d represents condition on force magnitude and c condition on velocity vector magnitude.

9 Experiments

This chapter presents the final experiments that were performed to verify the proposed solution to the given problem. The first part focuses on the verification of the KF, it provides experiments of both the LKF and UKF individually. Then the methods are compared during one experiment to see an immediate difference between them. Further, the estimation of external forces is verified. The last section demonstrates the behaviour of a complete system, where the state estimators are connected with the designed Admittance tracker.

Unfortunately, it was not possible to perform the experiments in the real world. The choice of the computer turned out to be inadequate, and only the VIO processing was consuming almost the whole computational sources of the Raspberry Pi. Together with the computational demands of the MRS system, it was not possible to perform autonomous real-world experiments. Therefore the experiments showing the capabilities of the designed system are performed only in the Gazebo simulator.

Another problem was raised during the simulation of physical interaction while directly pushing the vehicle. The implementation of Visual odometry in the Gazebo is not perfect. Although it worked reliably during normal simulated flight and most of the experiments are performed based on these data sources, the VIO failed when there was an explicit force applied to the model. Once this happened, even a small force of 1 N caused a significant jump in the direction of the push. This problem was compared to data from the GPS data. GPS was stable, and there was no jump in the measurements data. Hence the problem was in the implementation of the VIO. Based on this problem, experiments, where the UAV is physically pushed in the simulation, are performed with odometry based on the GPS and IMU from Pixhawk autopilot.

9.1 Kalman filter verification

The verification of the filters consists of multiple simulated flights that verify their performance. The filters are also compared on the same simulated flight. The last experiment shows the ability of the estimators to obtain the external force that is applied to the UAV. The root mean square error comparison between the estimation and measurements is provided. Together with the innovation magnitude test, it provides a metric to define the filter's performance. Description of this method innovation magnitude test can be found in section 6.2.3.

9.1.1 Linear Kalman filter

The verification of the LKF is depicted in the figure 9.1. It shows the performance of the filter on the x-axis. The conducted experiment for this purpose was flying around a

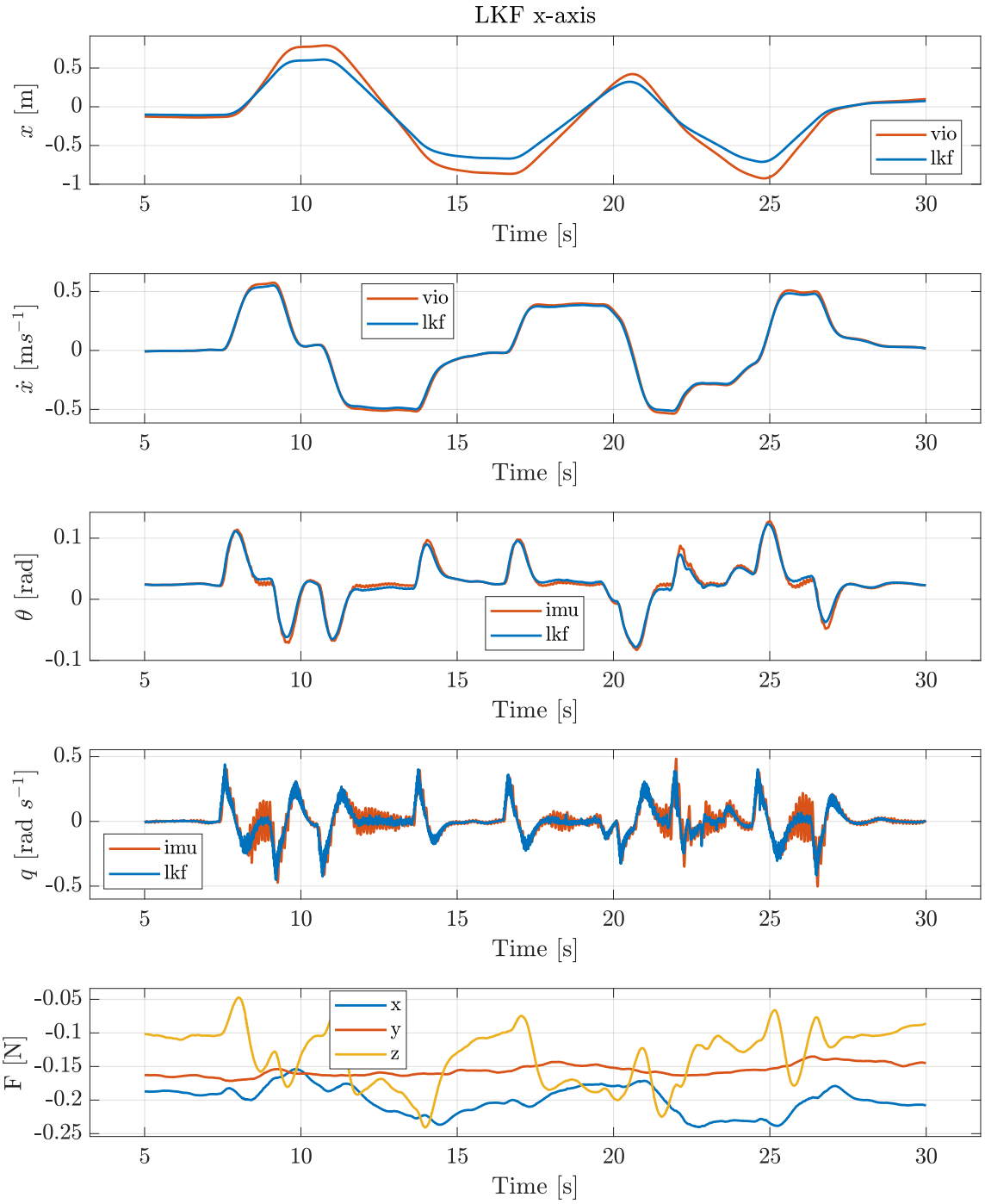


Figure 9.1: Verification of the LKF filter. The graph shows an estimation of the states in the x-axis. Specifically, it shows a position x, velocity $\dot{x}$, pitch angle θ and pitch rate q. The last graph depicts the estimated force at all axis. Notice the constant force offset in the x a y axis. Verification in other axis can be found in the appendix B.

Method	x [m]	$\dot{x}$ [m s^{-1}]	θ [rad]	q [rad s^{-1}]
LKF	0.061	0.0093	0.0039	0.0335

Table 9.1: Table shows a RMSE of the estimated values and measurement data in all states of x-axis subsystem estimated by the LKF. The error values corresponds to the figure 9.1.

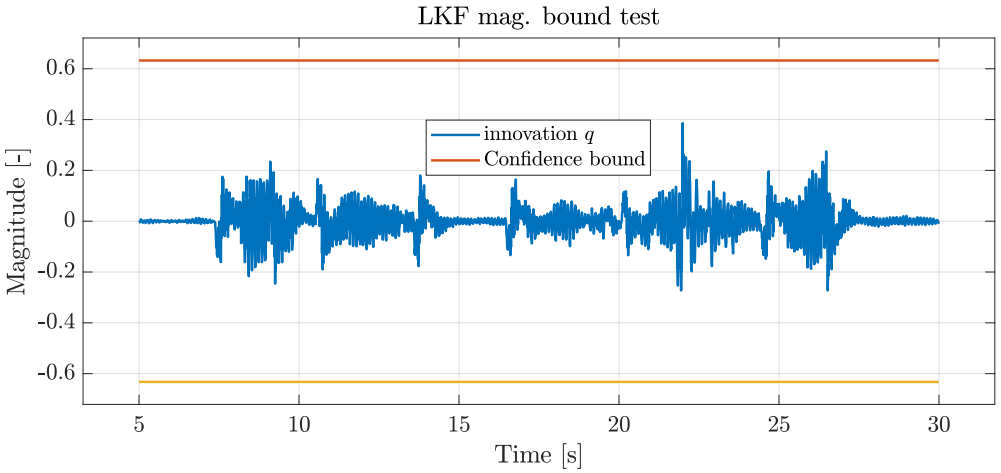


Figure 9.2: Innovation magnitude test of the pitch rate q estimated by the LKF. The graph indicates overestimated noise covariance matrices of the filter.

hovering point in each axis. Verification in other axes than is presented here can be found in appendix B. Notice that the data are not very noisy, and significant filtration can be seen mainly in the attitude rate. Significant deviation from the measured data can be found in the graphs in the position. This deviation becomes significant (larger than 20 cm) during higher pitched manoeuvres ($\theta > 0.08$). This error is expected as the linear model works only around equilibrium, which is the hovering point. The model incorrectly estimates the attitude rate already, and this error is further propagated to other states. Root mean square error between the estimated and measured states is in table 9.1. The last graph shows the estimated forces in all axes. Notice that the UAV has a constant disturbance in the horizontal axes. These disturbances are caused by the vehicle's design, which is not symmetrical or might be unbalanced.

The innovation magnitude test was performed on the pitch rate q data from the figure 9.1 and is depicted on figure 9.2. It can be seen that the innovations do not cross the confidence bound. This behaviour signalizes that the choice of the process noise $\mathbf{Q}$ and measurement noise $\mathbf{R}$ was overestimated.

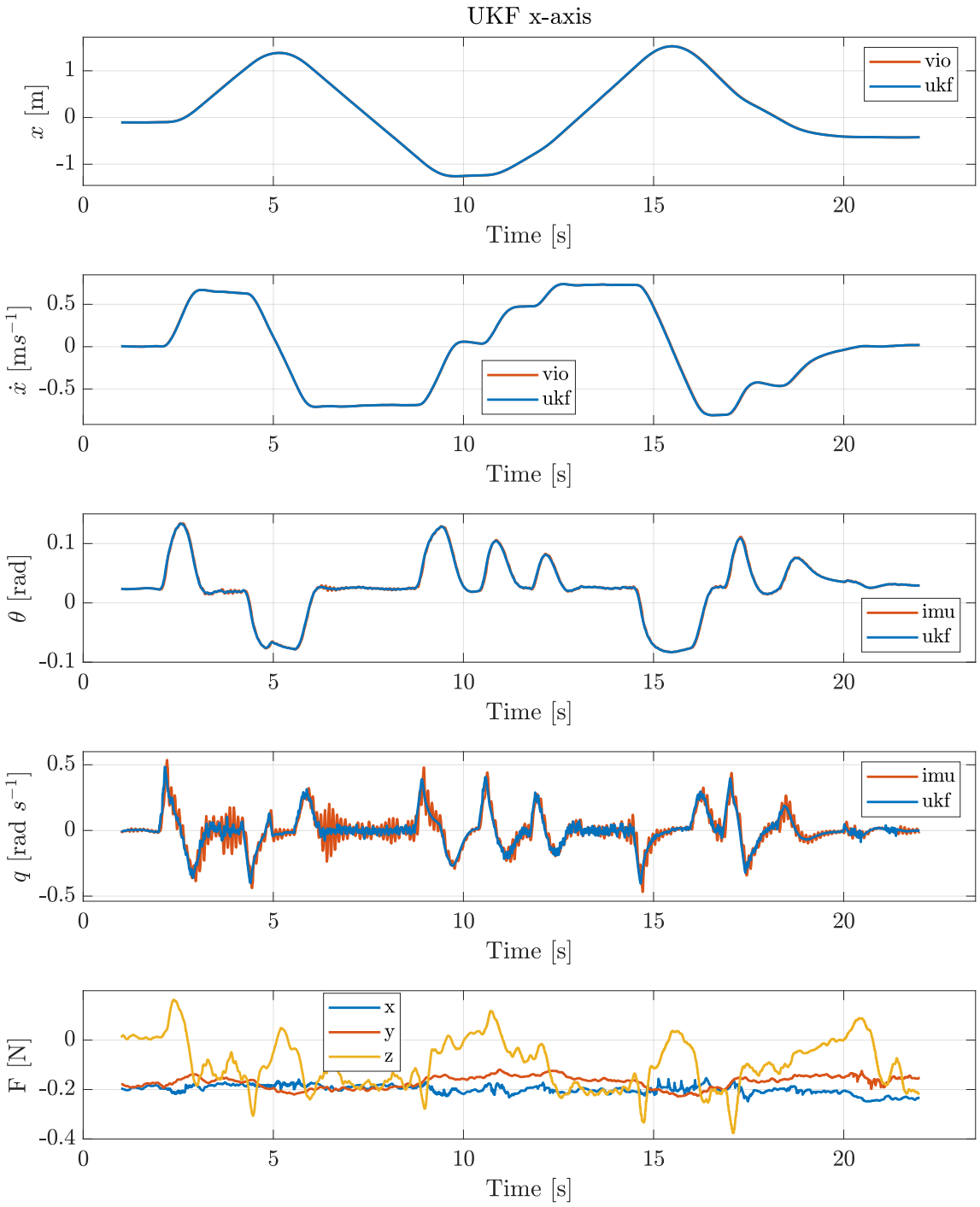


Figure 9.3: Verification of the UKF filter. The graph shows estimation of the states in the x-axis. Last graph depicts the estimated force at all axis, the estimator . Verification in other axis can be found in the appendix B.

Method	x [m]	$\dot{x}$ [m s^{-1}]	θ [rad]	q [rad s^{-1}]
UKF	0.0120	0.0024	0.0008	0.0293

Table 9.2: Table showing a RMSE of the estimated values with measurement data. The error values corresponds to the figure 9.3.

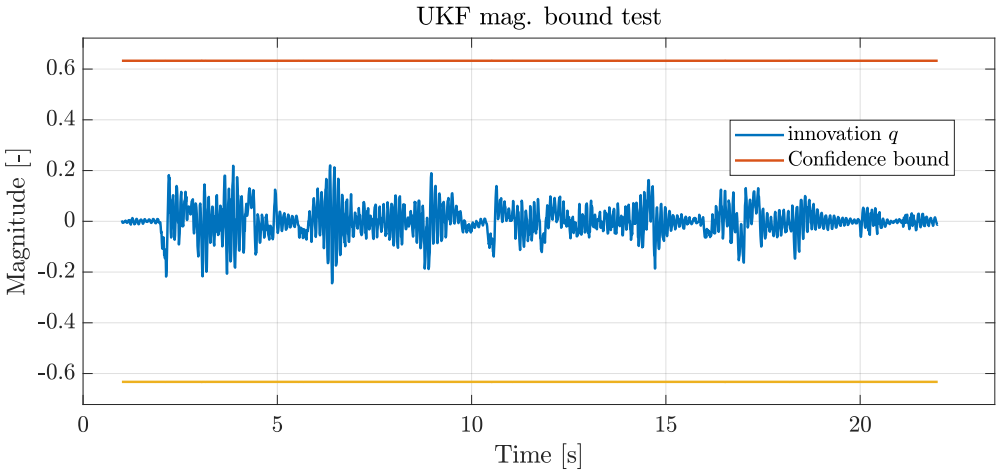


Figure 9.4: Innovation magnitude test of the pitch rate q with the confidence bounds from data estimated by UKF. The result indicates that the combination of the measurement and process noise are overestimated.

9.1.2 Unscented Kalman Filter

The UKF was verified in the same way as the LKF. A test flight was conducted, and the performance of the filter can be seen on the figure 9.3. The picture again depicts only the x-axis states. For complete verification, see appendix B. The UKF tracks the measured data reliably, and there is no deviation visible from the graphs. Also the table 9.2 shows lower error values. The only higher error can be seen in the attitude rate, which is given by the noisy measurements. The innovation test of the pitch rate q was also performed (figure 9.4. Similarly to the LKF innovation test, the innovations do even come close to the confidence bound, which means that the apriori noise parameters are overestimated.

9.1.3 Comparison

Both of the estimators were also tested together simultaneously. Instead of flying around the hovering point, there is a step reference on the position in this experiment. Steps of 0.5 m, 1 m and 5 m were performed and the result can be seen on the figure 9.5. This experiment provides comparison only in the x-axis. From the graph, it is possible to see that both the algorithm were able to track the current state. Only significant divergence (over 1 m) can be seen in the LKF estimation of the position. It is the same effect that occurred in the LKF verification, but here the pitch angle was much higher (up to 0.5 rad). Notice that the position estimated by the LKF corrects to the correct value after the step back. It is given by the fact, that same step in the opposite direction causes same deviation, which in this case corrects

Method	x [m]	$\dot{x}$ [m s^{-1}]	θ [rad]	q [rad s^{-1}]
LKF	0.5049	0.0180	0.0055	0.369
UKF	0.0132	0.0066	0.0023	0.0321

Table 9.3: Table showing a RMSE of the estimated values with measurement data. The error values corresponds to the figure 9.5.

the prediction. Comparison of the filters in the mean of the RMSE can be see in table 9.3. The UKF is had a lower error at every state variable.

The last experiment focused on the ability to estimate the external forces correctly. This experiment is depicted in figure 9.6. The experiment consisted of applying a direct force to the model in the simulation. Three pushes had a duration of 3 s with the magnitude of 1, 2 and 3 N. The graph shows that both estimators quickly tracked the generated force. UKF behaves more aggressively, and oscillations occur on a higher impact. This would require tuning the filter parameters to optimize the transition. LKF shows smooth and fast estimation. Notice the constant offset between reference and the estimation. It is caused by the parasitic force that is present during the whole flight and offsets the measurements. This parasitic force can be also seen in figure 9.1 and figure 9.3. This parasitic force is further recorded and used as an offset to provide correct estimations.

9.1.4 Summary

The experiments proved that both estimators work. A key attribute, the external force estimation, was precisely estimated by both methods. Even though the error of LKF estimates can be significant in the estimation of other variables, the LKF still can be used under certain circumstances. The position controller should be tuned and constrained to prevent significant attitude angles. This manoeuvre moves the linear model further from its equilibrium and makes the LKF prediction unreliable. The controller was slightly constrained even during the above experiments. Notice that the response to the steps in the figure 9.5 is very smooth and slow. Nevertheless, both methods can be used for Admittance control.

9.2 Admittance control

Previous experiments proved that both estimation methods work, and it is expected that they will behave similarly while flying with the Admittance tracker. Therefore to verify the Admittance tracker in the simulation, only the UKF is used. Whether to use UKF or LKF in the real experiments depends on the situation and circumstances. The UKF provides more reliable data. On the other hand, it is prone to encounter a numerical error when calculating direction vectors for the generation of the σ-points. The LKF might not be that precise but is numerically stable and less computationally demanding.

The experiment that verifies the working and shows the capabilities of the Admittance tracker can be seen on the figure 9.7. It shows the situation where the UAV is exposed to three pushes in the x-axis direction. The first impact is represented as 3 s long push with the

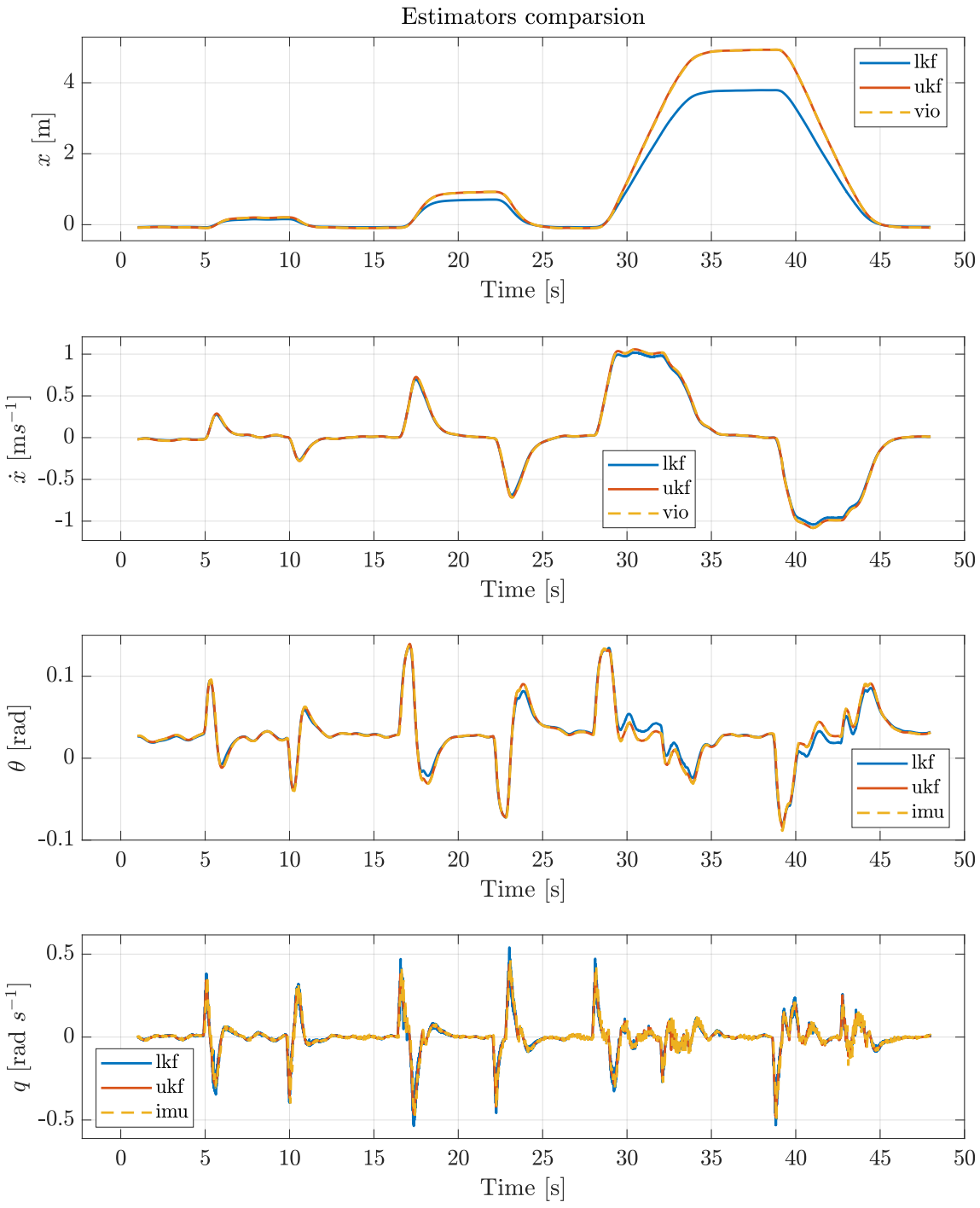


Figure 9.5: Comparison of the state estimation between a reference VIO data, LKF and UKF estimates in the x-axis. The graphs show the position x, velocity $\dot{x}$, pitch angle θ and attitude rate q. The UAV responded to the three continuous changes in the position of magnitude 0.5 m, 1 m and 5 m.

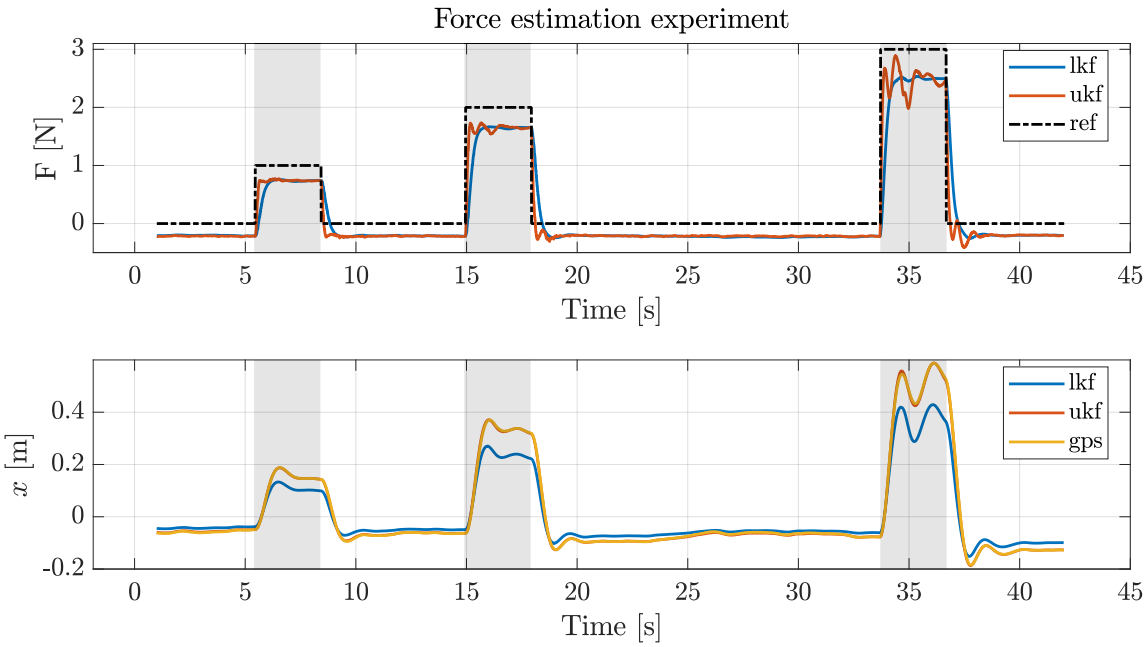


Figure 9.6: Comparison of the external force estimation, where three pushes with different magnitude alongside x-axis were applied for the time of 3 s. The comparison between referenced force, force estimated by UKF and LKF is in the first graph. Second graph shows the position deviation in the x-axis.

magnitude of 1 N. It smoothly initiated the change of the position, and after the force was released, the UAV continuously slowed down. The next two impacts were represented as short impulses to the system, the first one with magnitude 3 N and the second one with 3 N in the opposite direction. The UAV correctly initiated the movement in the direction of the applied force, and after releasing the force, it started to slow down smoothly.

The tracker was tuned by the parameters described in the chapter 8. The parameters were based on the [14]. Another experiment was conducted to examine the effects of these parameters on the behaviour of the UAV. The first experiment focused on the effect of changing inertia matrix $\mathbf{M}$. For this purpose was the damping matrix $\mathbf{D}$ fixed to a value described in the chapter 8. The resulting effect is depicted in the figure 9.8. With increasing values on the diagonal, the UAV becomes less responsive and more force is required to move the vehicle. On the other hand, if the values are lowered, the UAV become much more reactive to the external force, and even slight touch makes the vehicle fly further away. It corresponds to the virtual increasing the moment of inertia of the aircraft and confirms the intended behaviour.

A similar experiment was also performed to simulate the behaviour when the damping matrix changes. The inertia matrix was fixed for this experiment and set to the value presented in chapter 8. The results are in figure 9.9. The effect on the vehicle is similar to changing the inertia matrix for the values that are lower than 1. However, if the damping is higher, it can be seen that the vehicle's velocity is damped very significantly. This behaviour can be helpful in situations where fast manoeuvres are not wanted. Similarly to the inertia matrix, here also, the name damping matrix confirms the behaviour.

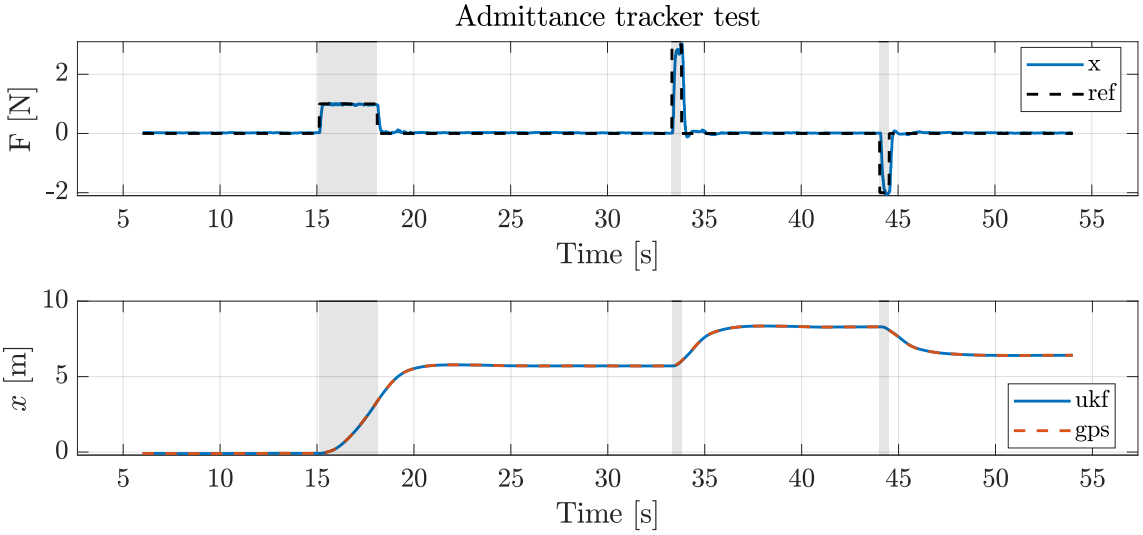


Figure 9.7: Data showing the working of the Admittance tracker in the x-axis. The first graph shows the reference and estimated force. The second graph depicts the change in the position during the physical interaction. Red areas highlight the time when the force was applied. Video showing the functionality in the simulation can be found at `http://mrs.felk.cvut.cz/smrcka2021thesis`

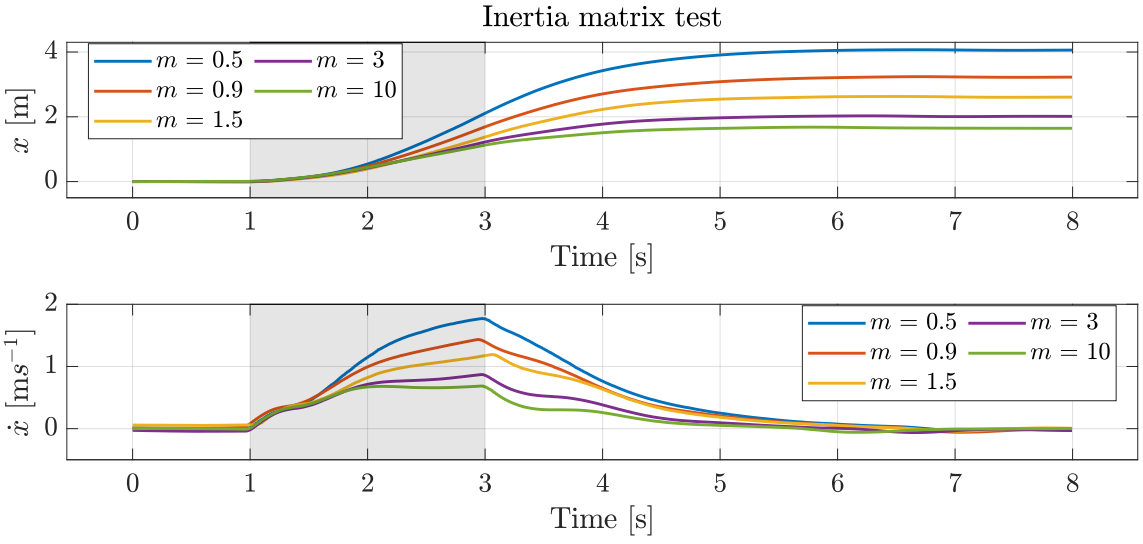


Figure 9.8: Influence of the inertia matrix parameters to the behaviour of the UAV in response to the 2 s long impact of 1 N. Upper graph shows the position x response and lower plot the velocity $\dot{x}$.

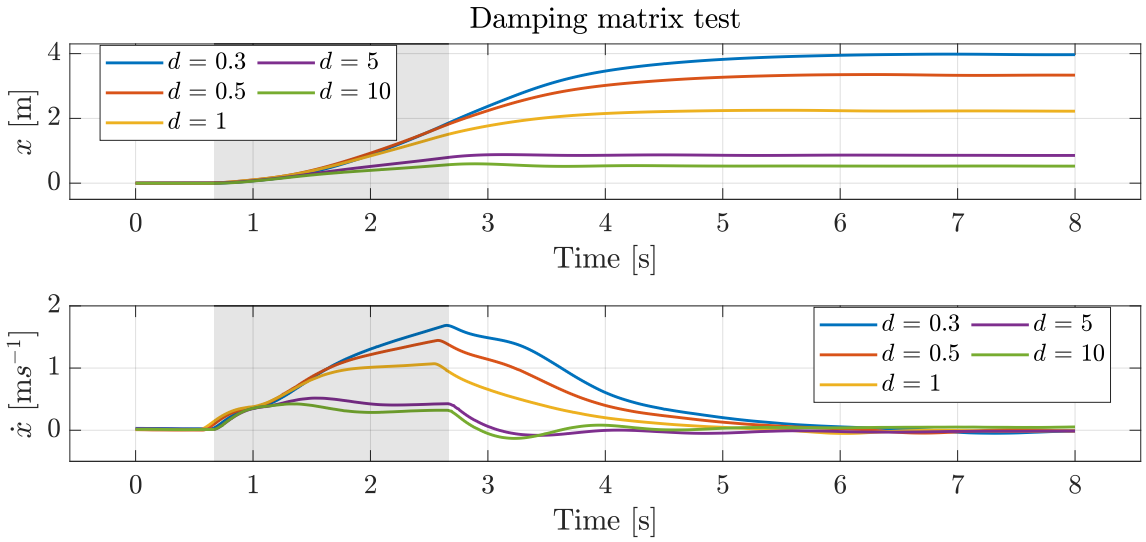


Figure 9.9: Data showing the influence of the different damping parameters to the behaviour of the UAV. The graph shows the response on 2 s long impact of 1 N

10 Conclusion

This thesis presented a solution to the challenge of the physical human-UAV interaction. The solution included the design and build of a small interaction safe aircraft. Implementation of two different state estimation methods that are used to estimate an external force was presented. Last, the thesis dealt with the implementation of the Admittance control. The complete system was integrated into the MRS system pipeline.

Design of the UAV is described in the chapter 3. The vehicle was built as small as possible to fit all the necessary equipment, and its final dimension is 200 mm in diameter, which makes it currently the smallest UAV in the MRS laboratory. The intended localization method is visual odometry (VIO), however as it turned out, the computational resources provided by the Raspberry Pi 4 were not enough. Therefore the vehicle was tested only in a manual flight. Manual flights were sufficient to provide identification of the model chapter 5 and implementation of a realistic model into the Gazebo simulator. Dynamics of the UAV were studied in chapter 4. The external force was added to the model, and both the nonlinear and linear models were presented. State estimation was studied in detail in order to estimate the external forces acting on the vehicle without additional sensory equipment (chapter 6). The Linear Kalman filter and nonlinear Unscented Kalman Filter were implemented. The Admittance control was implemented to enable physical interaction and made the system suitable for such situations. It is described in the chapter 7. Its dynamics were wrapped into the Admittance tracker to allow smooth operation. Implementation details were presented in chapter 8 and described the choice of all parameters that were necessary to be tuned.

Chapter 9 presented experiments that have been conducted. Firstly, the estimation methods were verified and compared with each other. The verification showed that both the LKF and UKF were able to estimate the external forces very well. Both of the methods could also provide reliable information about the UAV's complete state and allow simulation flight. However, the LKF performed less precisely overall, and deviation could be found in all the states. This behaviour is, however, expected as its model expects only small attitude angles. Therefore, both methods are suitable for the task, but the UKF's performance is more consistent over variation of situations and manoeuvres. Then several experiments verified the performance of the implemented Admittance tracker. The tracker worked well and performed a smooth reaction to the external forces. The last experiment focused on different inertia and damping matrices that define the tracker's dynamics.

10.1 Future work

Work presented in this thesis provided good basics into the problem of the physical interaction between humans and UAV. Future work will be to equip the constructed vehicle with a more powerful computer and verify the system in real-world experiments. A great

focus will be put on tuning the parameters of the KFs because the innovation magnitude test showed that the presented values were overestimated. Lastly, more focus will be put on a study of the overall stability of the system. This analysis is crucial for performing real-world interaction experiments with humans.

Bibliography

[1] "Dragonplate carbon fibre uav," https://www.suasnews.com/, online, August 2021.

[2] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The mrs uav system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 26, pp. 1–28, May 2021.

[3] "Pixhawk documentation," http://docs.px4.io/master/en, online, July 2021.

[4] C. Ott, R. Mukherjee, and Y. Nakamura, "Unified impedance and admittance control," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 554–561.

[5] M. Khosravi and H. Pishro-Nik, "Unmanned aerial vehicles for package delivery and network coverage," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–5.

[6] V. Spurny, V. Pritzl, V. Walter, M. Petrlik, T. Baca, P. Stepan, D. Zaitlik, and M. Saska, "Autonomous firefighting inside buildings by an unmanned aerial vehicle," *IEEE Access*, vol. 9, pp. 15 872–15 890, January 2021.

[7] N. A. Khan, N. Jhanjhi, S. N. Brohi, R. S. A. Usmani, and A. Nayyar, "Smart traffic monitoring system using unmanned aerial vehicles (uavs)," *Computer Communications*, vol. 157, pp. 434–443, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366420300189

[8] S. Rajappa, H. Bülthoff, and P. Stegagno, "Design and implementation of a novel architecture for physical human-uav interaction," *The International Journal of Robotics Research*, vol. 36, pp. 800 – 819, 2017.

[9] D. Hentzen, T. Stastny, R. Siegwart, and R. Brockers, "Disturbance estimation and rejection for high-precision multirotor position control," 2019.

[10] T. Báča, "Model predictive control of micro aerial vehicle using onboard microcontroller," Ph.D. dissertation, Faculty of Electrical Engineering, CTU in Prague, 2015, thesis.

[11] T. Tomić, C. Ott, and S. Haddadin, "External wrench estimation, collision detection, and reflex reaction for flying robots," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1467–1482, 2017.

[12] A. Keemink, H. van der Kooij, and A. Stienen, "Admittance control for physical human-robot interaction," *International journal of robotics research*, vol. 37, no. 11, pp. 1421–1444, Sep. 2018, sage deal.

[13] S. Tarbouriech, B. Navarro, P. Fraisse, A. Crosnier, A. Cherubini, and D. Sallé, "Admittance control for collaborative dual-arm manipulation," in *2019 19th International Conference on Advanced Robotics (ICAR)*, 2019, pp. 198–204.

[14] F. Augugliaro and R. D'Andrea, "Admittance control for physical human-quadrocopter interaction," in *2013 European Control Conference (ECC)*, July 2013, pp. 1805–1810.

[15] "Robot operating system," http://wiki.ros.org/ROS/Introduction, online, June 2021.

[16] "Gazebo simulator," http://gazebosim.org/, online, July 2021.

[17] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *49th IEEE Conference on Decision and Control (CDC)*, Dec 2010, pp. 5420–5425.

[18] T. Baca, P. Stibinger, D. Doubravova, D. Turecek, J. Solc, J. Rusnak, M. Saska, and J. Jakubek, "Gamma Radiation Source Localization for Micro Aerial Vehicles with a Miniature Single-Detector Compton Event Camera," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, June 2021, pp. 338–346.

[19] P. Petráček, V. Krátký, and M. Saska, "Dronument: System for Reliable Deployment of Micro Aerial Vehicles in Dark Areas of Large Historical Monuments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2078–2085, April 2020.

[20] G. Silano, T. Baca, R. Penicka, D. Liuzza, and M. Saska, "Power line inspection tasks with multi-aerial robot systems via signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4169–4176, 2021.

[21] C. Patel, S. Rao, and B. Driessen, "A testbed for mini quadrotor unmanned aerial vehicle with protective shroud," *Robotics & Automation*, 01 2005.

[22] "Flyability," https://www.flyability.com/, online, July 2021.

[23] J. Baichtal, *Building Your Own Drones: A Beginners' Guide to Drones, UAVs, and ROVs*. Que Publishing, 2015.

[24] P. Petráček, "Design, localization and position control of a specialized uav platform for documentation of historical monuments," Ph.D. dissertation, Faculty of Electrical Engineering, CTU in Prague, 2019, thesis.

[25] "Mavlink documentation," https://mavlink.io/en/, online, July 2021.

[26] J. Bednář, "Self-localization of unmanned aerial vehicles using visual inertial odometry," Ph.D. dissertation, Faculty of Electrical Engineering, CTU in Prague, 2020, thesis.

[27] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[28] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, p. 22, 2011.

[29] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[30] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006. [Online]. Available: https://books.google.cz/books?id=UiMVoP_7TZkC

[31] I. Reid, "Lecture notes in estimation i," February 2001.

[32] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158.

[33] I. Reid, "Lecture notes in estimation ii," February 2001.

[34] T. Winiarski and A. Woźniak, "Indirect force control development procedure," *Robotica*, vol. 31, 05 2013.

[35] A. Albu-Schäffer, C. Ott, and G. Hirzinger, "A unified passivity-based control framework for position, torque and impedance control of flexible joint robots," *The International Journal of Robotics Research*, vol. 26, no. 1, pp. 23–39, 2007. [Online]. Available: https://doi.org/10.1177/0278364907073776

[36] P. Matisko and V. Havlena, "Noise covariances estimation for kalman filter tuning," *IFAC Proceedings Volumes*, vol. 43, no. 10, pp. 31–36, 2010, 10th IFAC Workshop on the Adaptation and Learning in Control and Signal Processing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667015323351

Appendices

A CD Content

In Table A.1 are listed names of all root directories on CD.

Directory name	**Description**
thesis	the thesis in pdf format
src/thesis	latex source code
src/odometry	sources for the estimation task
src/model	sources for the simulation model
src/control	sources of the admittance tracker implementation
src/cad	complete cad model with stl files

Table A.1: CD Content

B Additional material for the verification of LKF and UKF

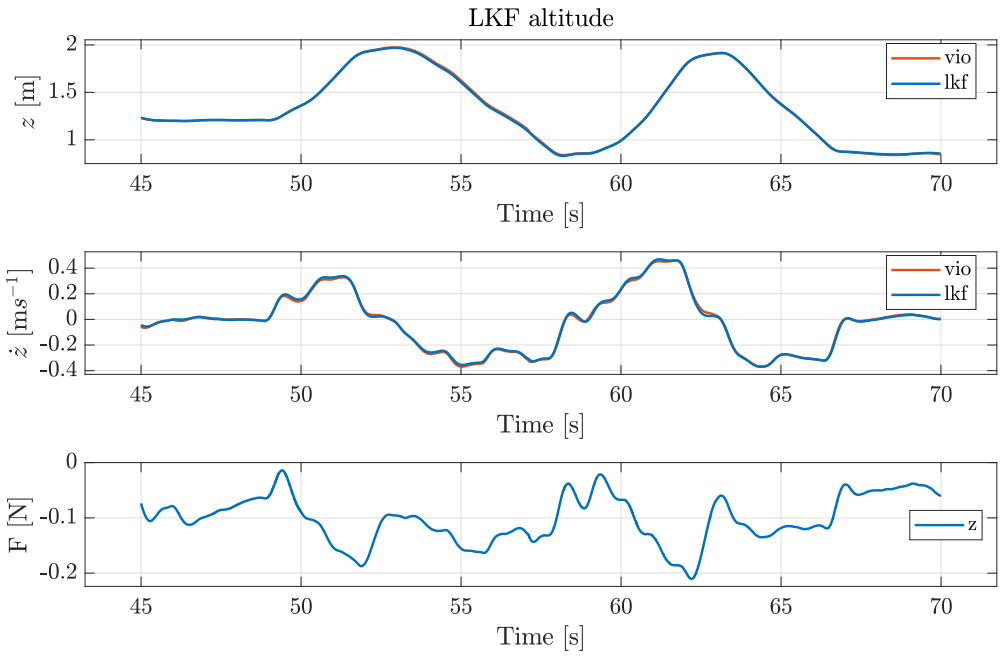


Figure B.1: Additional data of the LKF verification, graphs show the altitude data.

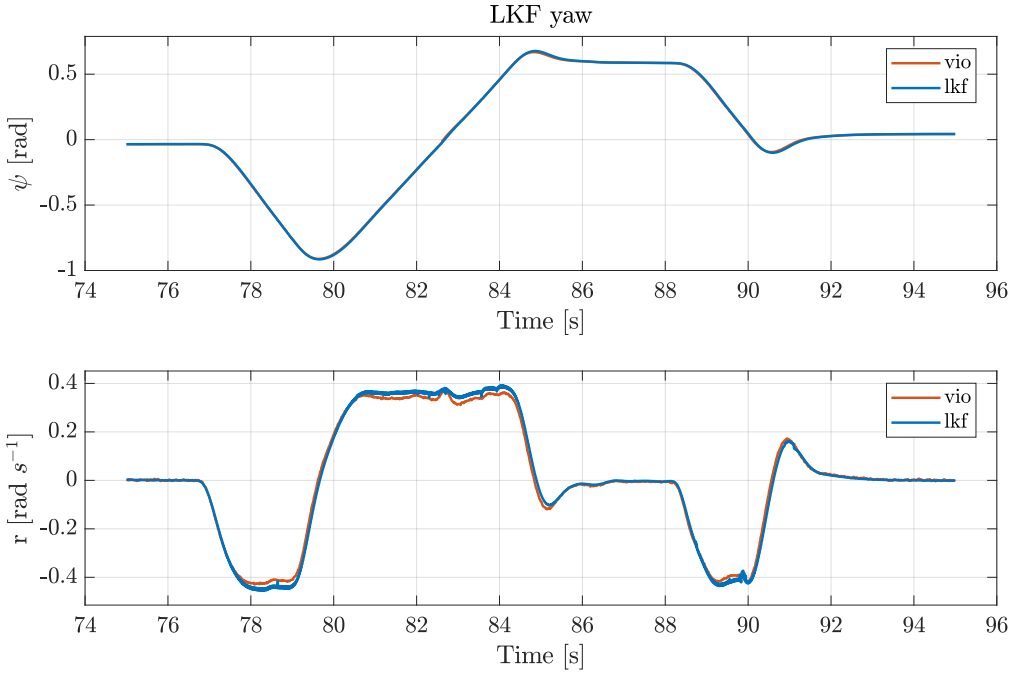


Figure B.2: Additional data of the LKF verification, graphs show the yaw subsystem data.

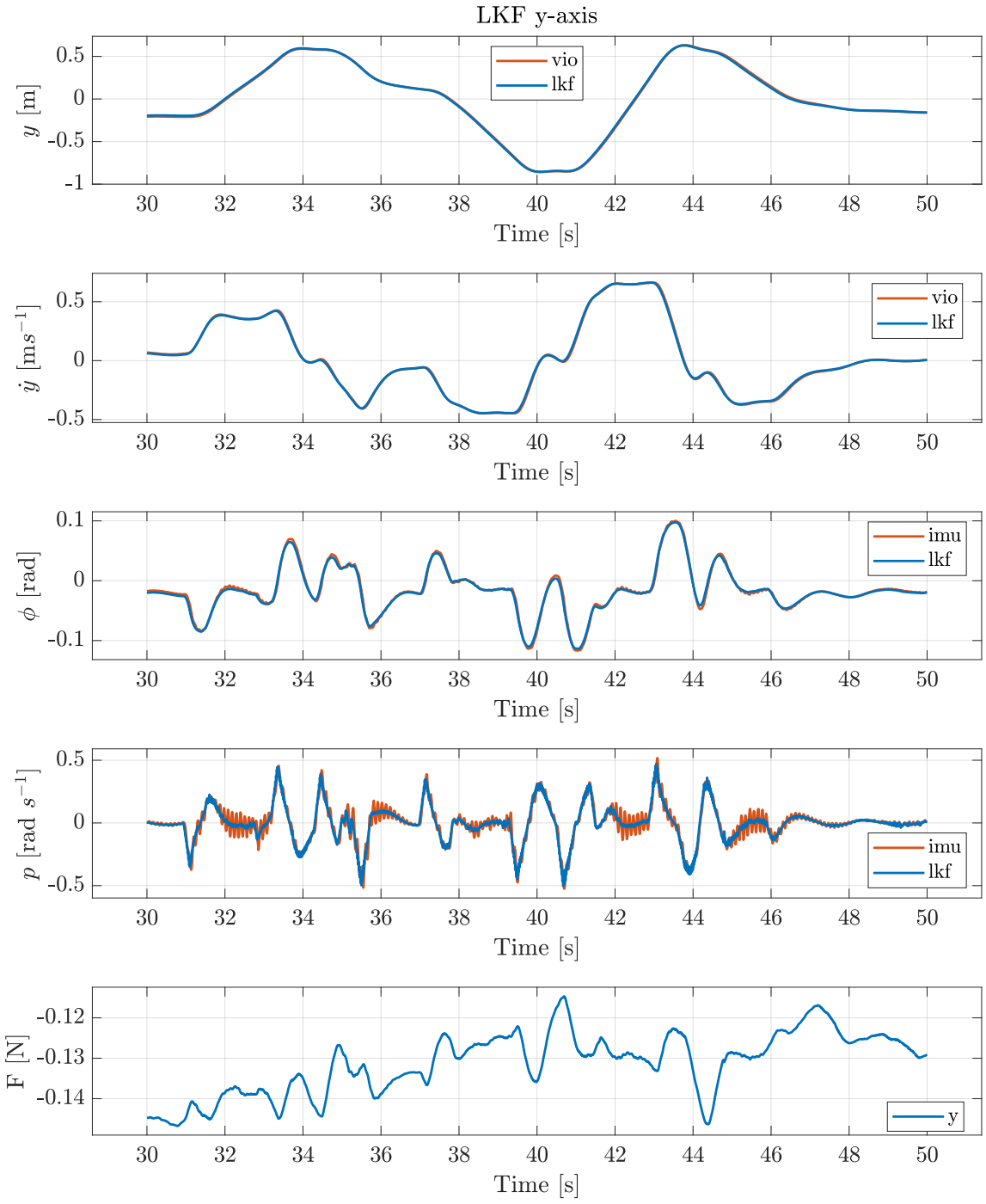


Figure B.3: Additional data of the LKF verification, graphs show the y-axis data.

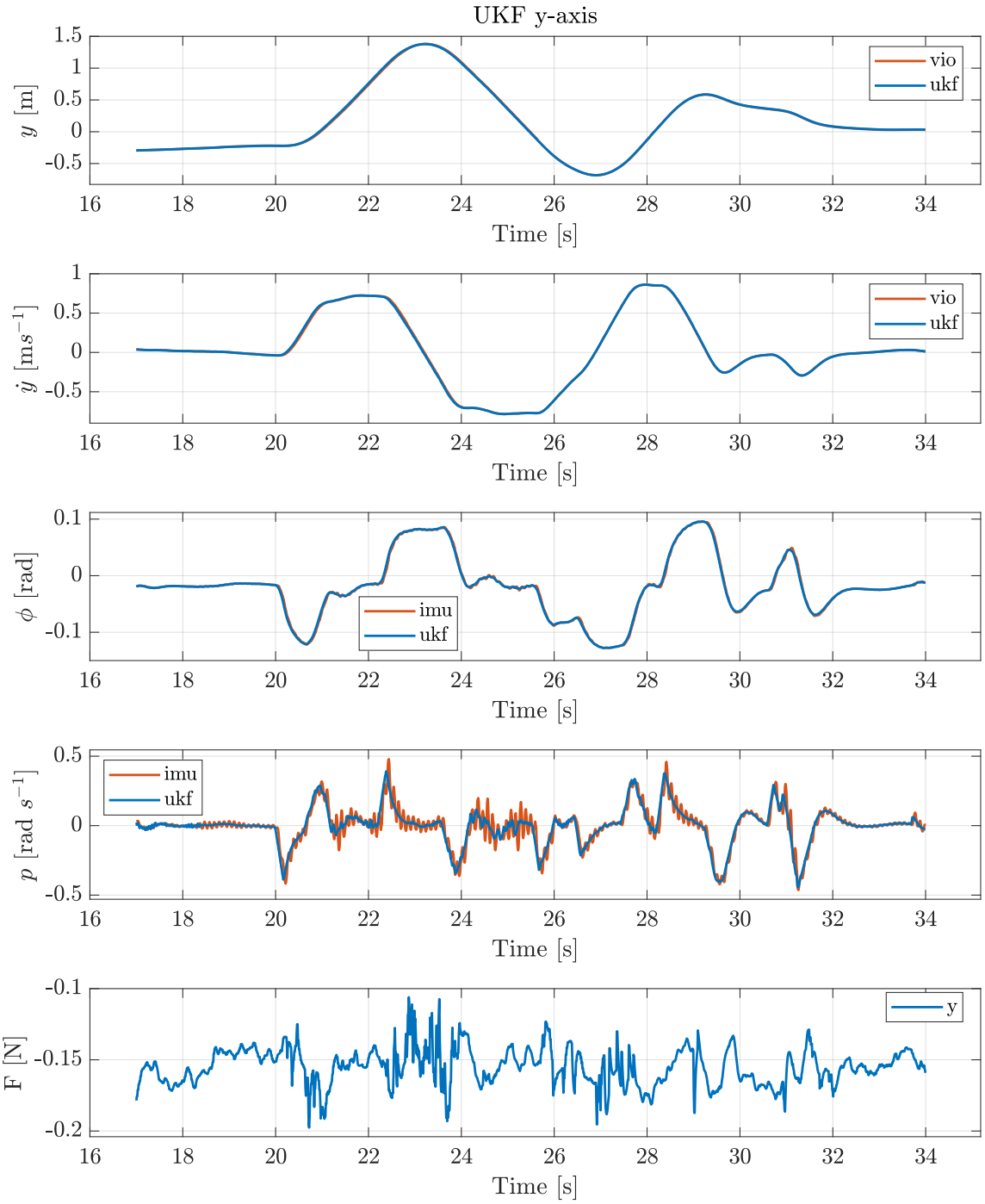


Figure B.4: Additional data of the UKF verification, graphs show the y-axis subsystem data.

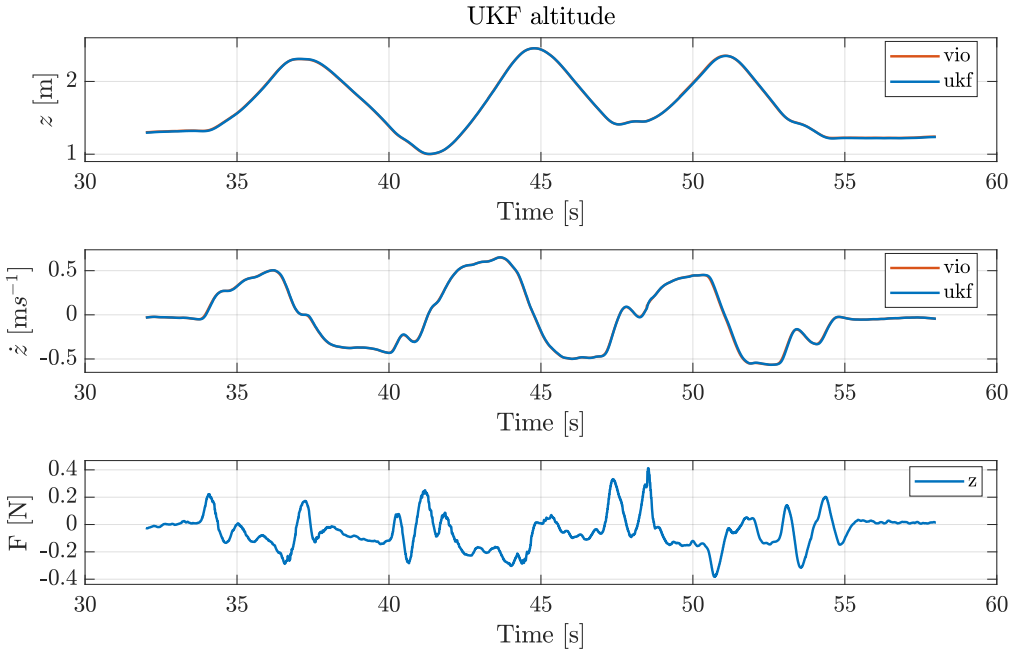


Figure B.5: Additional data of the UKF verification, graphs show the attitude subsystem data.

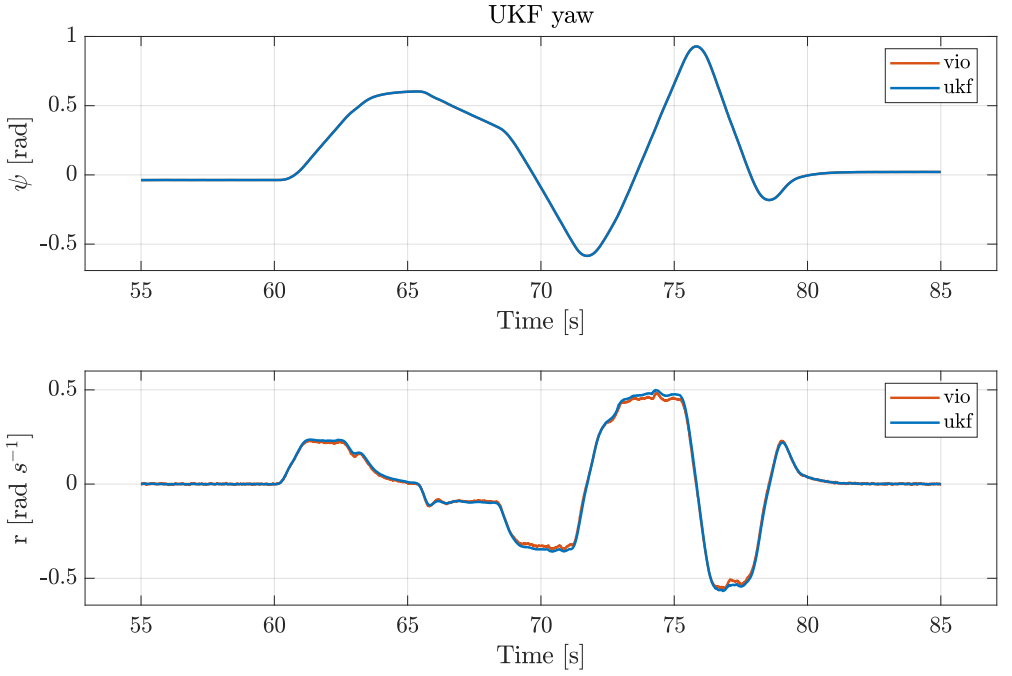


Figure B.6: Additional data of the UKF verification, graphs show the yaw subsystem data.