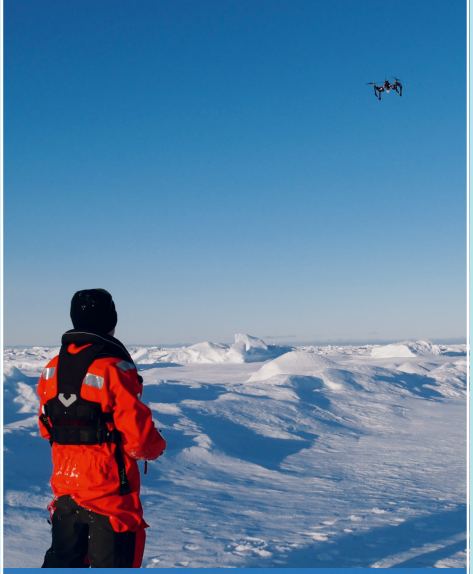
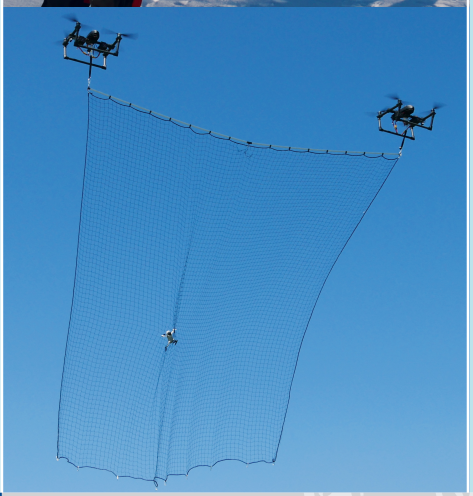




Institut für Informatik
Lehrstuhl für Informationstechnik
für Luft- und Raumfahrt
Prof. Dr. Sergio Montenegro

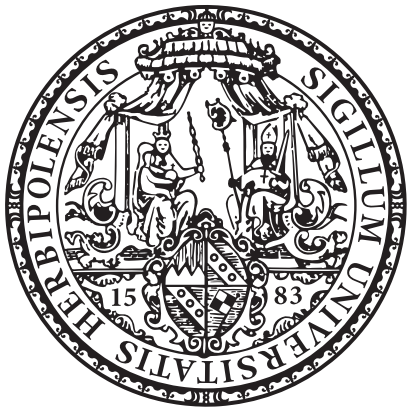
Research in Aerospace Information Technology

Julius-Maximilians-
UNIVERSITÄT WÜRZBURG

RAIT 1

Michael Strohmeier

FARN
A Novel UAV Flight Controller for Highly Accurate and Reliable Navigation



JULIUS-MAXIMILIANS-UNIVERSITÄT WÜRZBURG
CHAIR OF COMPUTER SCIENCE VIII
AEROSPACE COMPUTER SCIENCE

Dissertation

FARN – A Novel UAV Flight Controller for Highly Accurate and Reliable Navigation

---

submitted to the Faculty of
Mathematics and Computer Science
of the University of Würzburg
in fulfillment of the requirements for the degree of
Doctor Rerum Naturalium (Dr. rer. nat.)
by

MICHAEL STROHMEIER

December 2020

Supervisor and first reviewer:	Prof. Dr. Sergio Montenegro
Second reviewer:	Prof. Dr. Andreas Nüchter
Third reviewer:	Prof. Dr. Thomas Gustafsson

"If you do not know where you come from, then you don't know where you are, and if you don't know where you are, then you don't know where you're going. And if you don't know where you're going, you're probably going wrong."

Sir Terence David John Pratchett
I Shall Wear Midnight

Abstract

FARN – *A Novel UAV Flight Controller for Highly Accurate and Reliable Navigation*

This thesis describes the functional principle of FARN, a novel flight controller for Unmanned Aerial Vehicles (UAVs) designed for mission scenarios that require highly accurate and reliable navigation. The required precision is achieved by combining low-cost inertial sensors and Ultra-Wide Band (UWB) radio ranging with raw and carrier phase observations from the Global Navigation Satellite System (GNSS). The flight controller is developed within the scope of this work regarding the mission requirements of two research projects, and successfully applied under real conditions.

FARN includes a GNSS compass that allows a precise heading estimation even in environments where the conventional heading estimation based on a magnetic compass is not reliable. The GNSS compass combines the raw observations of two GNSS receivers with FARN's real-time capable attitude determination. Thus, especially the deployment of UAVs in Arctic environments within the project for ROBEX is possible despite the weak horizontal component of the Earth's magnetic field.

Additionally, FARN allows centimeter-accurate relative positioning of multiple UAVs in real-time. This enables precise flight maneuvers within a swarm, but also the execution of cooperative tasks in which several UAVs have a common goal or are physically coupled. A drone defense system based on two cooperative drones that act in a coordinated manner and carry a commonly suspended net to capture a potentially dangerous drone in mid-air was developed in conjunction with the project MIDRAS.

Within this thesis, both theoretical and practical aspects are covered regarding UAV development with an emphasis on the fields of signal processing, guidance and control, electrical engineering, robotics, computer science, and programming of embedded systems. Furthermore, this work aims to provide a condensed reference for further research in the field of UAVs.

The work describes and models the utilized UAV platform, the propulsion system, the electronic design, and the utilized sensors. After establishing mathematical conventions for attitude representation, the actual core of the flight controller, namely the embedded ego-motion estimation and the principle control architecture are outlined. Subsequently, based on basic GNSS navigation algorithms, advanced carrier phase-based methods and their coupling to the ego-motion estimation framework are derived. Additionally, various implementation details and optimization steps of the system are described. The system is successfully deployed and tested within the two projects. After a critical examination and evaluation of the developed system, existing limitations and possible improvements are outlined.

Zusammenfassung

FARN – *Eine neue UAV-Flugsteuerung für hochpräzise und zuverlässige Navigation*

Diese Arbeit beschreibt das Funktionsprinzip von FARN, einer neuartigen Flugsteuerung für unbemannte Luftfahrzeuge (UAVs), die für Missionsszenarien entwickelt wurde, die eine hochgenaue und zuverlässige Navigation erfordern. Die erforderliche Präzision wird erreicht, indem kostengünstige Inertialsensoren und Ultra-Breitband (UWB) basierte Funkreichweitenmessungen mit Roh- und Trägerphasenbeobachtungen des globalen Navigationssatellitensystems (GNSS) kombiniert werden. Die Flugsteuerung wird im Rahmen dieser Arbeit unter Berücksichtigung der Missionsanforderungen zweier Forschungsprojekte entwickelt und unter realen Bedingungen erfolgreich eingesetzt.

FARN verfügt über einen GNSS-Kompass, der eine präzise Schätzung des Steuerkurses auch in Umgebungen erlaubt, in denen eine konventionelle Schätzung mit Hilfe eines Magnetkompasses nicht zuverlässig ist. Der GNSS-Kompass kombiniert die Messungen von zwei GNSS-Empfängern mit der echtzeitfähigen Lagebestimmung von FARN. Damit ist insbesondere der Einsatz von UAVs in arktischen Umgebungen im Rahmen des Projektes ROBEX trotz der schwachen horizontalen Komponente des Erdmagnetfeldes möglich.

Zusätzlich erlaubt FARN eine zentimetergenaue relative Positionierung mehrerer UAVs in Echtzeit. Dies ermöglicht präzise Flugmanöver innerhalb eines Schwarms, aber auch die Ausführung kooperativer Aufgaben, bei denen mehrere UAVs ein gemeinsames Ziel haben oder physikalisch gekoppelt sind. In Verbindung mit dem Projekt MIDRAS wurde ein Drohnenabwehrsystem entwickelt, das auf zwei kooperativen Drohnen basiert, die koordiniert agieren und ein gemeinsam aufgehängtes Netz tragen, um eine potenziell gefährliche Drohne in der Luft einzufangen.

Im Rahmen dieser Arbeit werden sowohl theoretische als auch praktische Aspekte der UAV-Entwicklung behandelt, wobei der Schwerpunkt auf den Bereichen der Signalverarbeitung, der Navigation und der Steuerung, der Elektrotechnik, der Robotik sowie der Informatik und der Programmierung eingebetteter Systeme liegt. Darüber hinaus soll diese Arbeit eine zusammengefasste Referenz für die weitere Drohnenforschung darstellen.

Die Arbeit erläutert und modelliert die verwendete UAV-Plattform, das Antriebssystem, das elektronische Design und die eingesetzten Sensoren. Nach der Ausarbeitung mathematischer Konventionen zur Lagedarstellung, wird der eigentliche Kern des Flugreglers erläutert, nämlich die eingebettete Schätzung der Eigenbewegung und die prinzipielle Regelungsarchitektur. Anschließend werden, basierend auf grundlegenden Navigationsalgorithmen, fortgeschrittene trägerphasenbasierte Methoden und deren Zusammenhang mit der Schätzung der Eigenbewegung abgeleitet. Zusätzlich werden verschiedene Implementierungsdetails und Optimierungsschritte des Systems beschrieben. Das System wird innerhalb der beiden Projekte erfolgreich verwendet und getestet. Nach einer kritischen Untersuchung und Bewertung des entwickelten Systems werden bestehende Einschränkungen und mögliche Verbesserungen aufgezeigt.

Acknowledgements

At this point, I would like to thank everyone who supported me with words and deeds throughout this work. First of all, I would like to express my uttermost gratitude to my dissertation adviser Prof. Dr. Sergio Montenegro for guiding and supporting me with fruitful discussions, plenty of ideas as well as his valuable insights and critical feedback. Secondly, I would also like to express my deepest thanks to my second reviewer, Prof. Dr. Andreas Nüchter and to my third reviewer, Prof. Dr. Thomas Gustafsson, for the quick and careful review of my dissertation and their support. I would also like to thank Prof. Dr. Reiner Kolla and Prof. Dr. Hakan Kayal for their contribution to my disputation.

Furthermore, I want to extend my gratitude towards all my colleagues, student assistants and final thesis candidates, without whose help the work in its present form would not have been possible. I would like to thank especially Tobias Mikschl for his support during the first Polar expedition and Julian Rothe for his efforts in preparation for and during the second expedition as well as his outstanding collaboration during MIDRAS. The tireless support of Alexander Hilgarth, Thomas Walter and Michael Ruffer in the development of the flight controller hardware deserves special thanks and hence a special mention, too. I would also like to thank Anna Gonel, without whose help not only organizational tasks would have been impossible for me. My very special thanks go to Lennart Werner, Cedric Liman, Jasper Zevering and Stefan Beck for their excellent assistance as well as their enthusiastic and inspiring support. In addition to my current colleagues, I would also like to thank my former colleague and supervisor Dr. Nils Gageik for his preliminary work and for his recommendation for this doctoral position.

I would also like to give my gratitude to my parents, Dr. Helmut and Ursula Strohmeier, and my brother Alexander, who have supported and motivated me all my life, as well as my grandparents Edgar and Hildegard Nagel, who have always been a place of peace and recreation when desperately needed. Thank you, Lisa.

Michael Strohmeier
Würzburg, 2020

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
AUV	Autonomous Underwater Vehicle
CDMA	Code Division Multiple Access
CNN	Convolutional Neural Network
DDR3	Double Data Rate 3
DLR	German Aerospace Center
DS-TWR	Double-Sided Two-Way Ranging
EKF	Extended Kalman Filter
EGNOS	European Geostationary Navigation Overlay System
ESA	European Space Agency
ESC	Electronic Speed Controller
ESKF	Error-State Kalman Filter
FDMA	Frequency Division Multiple Access
FPU	Floating Point Unit
FPV	First Person View
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GLONASS	Global'naya Navigatsionnaya Sputnikovaya Sistema
GNC	Guidance, Navigation and Control
I2C	Inter-Integrated Circuit
IGS	International GNSS Service
ILS	Integer Least Square
ILSQE	Integer Least Squares with Quadratic Equality
IMU	Inertial Measurement Unit
IRNSS	Indian Regional Navigation Satellite System
JPL	Jet Propulsion Laboratory
LAMBDA	Least-Square Ambiguity Decorrelation Adjustment
LOS	Line-of-Sight
LPS	Local Positioning System
LSE	Least Square Estimation
LSQE	Least Square with Quadratic Equality
LTE/4G	Long Term Evolution/4G

MDS	Multi-Dimensional Scaling
MEMS	Micro-Electro-Mechanical System
MQX	Message Queue eXecutive
NAVSTAR GPS	Navigation Satellite Timing and Ranging GPS
NMEA	National Marine Electronics Association
NTRIP	Networked Transport of RTCM via Internet Protocol
NTP	Network Time Protocol
OCRAM	On-Chip Random-Access Memory
PCB	Printed Circuit Board
PID	Proportional-Integral-Derivative Controller
PPP	Precise Point Positioning
PPS	Pulse Per Second
PWM	Pulse-Width-Modulation
QZSS	Quasi-Zenith Satellite System
RDC	Resource Domain Controller
LED	Light-Emitting Diode
RTCM	Radio Technical Commission for Maritime Services
RTOS	Real-time Operating System
RTK	Real-time Kinematic
RMSE	Root-Mean-Square Error
ROS	Robot Operating System
RODOS	Real-time Onboard Dependable Operating System
RPMSG	Remote Processor Messaging
SBAS	Space-Based Augmentation System
SDRAM	Synchronous Dynamic Random-Access Memory
SISRE	Signal-in-space Ranging Error
SNR	Signal-to-Noise-Ratio
SPI	Serial Peripheral Interface
TCM	Tightly-Coupled Memory
TDOA	Time Difference of Arrival
TOA	Time of Arrival
UAV	Unmanned Aerial Vehicle
UWB	Ultra-Wide Band
VSLAM	Visual Simultaneous Localization and Mapping
VTOL	Vertical Take-Off and Landing
WAAS	Wide Area Augmentation System

Für Opa Ed

Chapter 1

Introduction

The main purpose and goal of FARN is to implement a versatile and configurable flight controller for Unmanned Aerial Vehicles (UAVs) that can be easily adapted to match different requirements of complex and advanced mission scenarios thus improving the overall capabilities of autonomous UAVs. Additionally, this thesis aims to provide a detailed reference work to other drone researchers and enthusiasts on advanced UAV algorithms, including low-cost sensor modeling, conditioning and calibration, precise navigation and ego-motion estimation together with robust and reliable UAV control. Both objectives are mainly achieved by directing advanced satellite navigation methods towards UAV applications, implementing a multi-sensor fusion framework for reliable pose and twist estimation in hard real-time as well as by providing detailed insights into the development of different software and hardware components of the flight controller.

A distinctive feature of the developed flight controller is hereby the precise real-time navigation using Global Navigation Satellite System (GNSS) raw and carrier phase measurements. Centimeter positioning accuracy and precise heading estimates are obtained by combining Real-time Kinematic (RTK) algorithms with other UAV sensor information, such as Ultra-Wide Band (UWB) radio ranges or Inertial Measurement Unit (IMU) observations. Therefore, a multi-sensor fusion and ego-motion estimation framework which is based on an embedded Error-State Kalman Filter (ESKF) is implemented and deployed at the very core of the flight controller. Although the flight controller is mainly targeted towards outdoor applications, its framework allows the easy integration of different positioning systems such as radio-based Local Positioning Systems (LPSs), Visual Simultaneous Localization and Mapping (VSLAM) sensors or optical tracking systems in order to navigate reliably indoor or in other GNSS denied environments, too, which was already demonstrated by the author.

The flight controller hardware itself is based on a heterogeneous dual-core architecture allowing the simultaneous use of the Real-time Onboard Dependable Operating System (RODOS) and the Robot Operating System (ROS). While RODOS is used on the smaller core with hard real-time capabilities for timing critical tasks such as the attitude determination and control of the UAV, ROS runs on top of Ubuntu on the more powerful core and is used for high level tasks, wireless communication and interfacing complex sensors. Thereby, the modular approach of both operating systems allows a software design based on building blocks thus providing a maximum degree of flexibility and adaptability to different mission specific requirements and various hardware configurations. Given the publish/subscribe communication paradigm for asynchronous messaging from both middle-wares, inter-core communication, and furthermore, the communication between a heterogeneous cluster of agents within a distributed system of UAVs are possible, too.

1.1 Motivation

Over the last decade, UAV systems have emerged and established themselves as a key technology for modern transportation systems, aerial photography as well as surveillance and surveying tasks. In particular, multi-rotors with Vertical Take-Off and Landing (VTOL) capabilities, commonly referred to as *drones*, have been in the focus of a considerable amount of research and numerous development efforts. As of today, commercial drone solutions are especially widespread in aerial photography and surveying tasks.

Typical applications for commercial UAVs include wild life monitoring [1], crops analysis and agricultural surveying [2, 3], the inspection of buildings and infrastructure [4, 5], delivery logistics [6], search and rescue operations [7] as well as the aerial support of firefighters [8]. A common denominator within the applications described above is that commercial UAVs are utilized as flying sensor platforms and are equipped with mission specific payloads. If required, a dedicated single board computer for payload computations might be used, too. Relying on commercial flight controllers, the respective scenarios are mostly limited to autonomous GNSS waypoint flight or manual control with a direct line of sight or a real-time video link providing a First Person View (FPV) for the UAV operator.

Despite the wide availability of commercial UAV platforms and flight controllers, highly customized UAVs are still being researched and developed with great eagerness. Comparing publications regarding the development of autonomous UAVs within the last five years, the following three main research areas can be identified among others:

UAVs in challenging environments Within this context, challenging UAV environments are typically characterized by the fact that a part of the built-in sensors are unreliable or can not be used for navigation purposes at all. Hereby, the majority of publications is limited to the UAV deployment in GNSS denied environments although other sensors can be affected by the environment, too. Originally, the research of UAVs in GNSS denied environments addressed alternative positioning methods for indoor applications such as the combination of stereo vision and laser scans [9], Local Positioning Systems (LPSs) based on UWB radio ranging [10] or navigation approaches using optical flow and a reactive collision avoidance system [11]. More recent studies, however, aim to develop UAVs for underground missions in humid and dark environments, e.g. in mining applications [12], sewage tunnels [13] and even for the autonomous inspection of the Fukushima containment vessel [14]. Additionally, there are outdoor scenarios where the GNSS signal reception is heavily disturbed by surrounding infrastructure or buildings, e.g. in the vicinity of windmills [15] or when flying close to artificial and natural canyons [16]. Surely, the most extreme and simultaneously most impressive example is NASA's recently launched Mars Helicopter *Ingenuity* which is going to be the first UAV to be deployed on another planet [17]. Comparing the on-board computers for the different navigation solutions within GNSS denied environments, it can be observed that all approaches combine a real-time capable flight controller with a dedicated navigation computer. While the flight controller can be either, commercial or custom-built, dedicated and application specific algorithms are required for a reliable navigation. The main challenge within this area is certainly to provide a precise and robust navigation solution despite the degradation of primary navigation sensors.

UAVs for aerial manipulation The second area of increased research interest is the deployment of UAVs for aerial manipulation. Within this context, aerial manipulation is defined as the interaction of a UAV with its environment. Hereby, the UAV platforms are typically equipped with one or more manipulators, that can be either actively controlled or passively suspended from a drone. To the authors knowledge, except for large area spraying, UAVs for aerial manipulation are not deployed in commercial applications yet, but are limited to research scenarios. In research, different ways of manipulation are considered: The UAVs can be equipped with robotic arms for grasping tasks [18–21], with spraying mechanisms for precision agriculture or spray painting [22, 23] or just carry a cable suspended load [24]. A drone carrying a net gun in order to intercept other drones in mid-air is described in [25]. For more examples, the interested reader is referred to very comprehensive reviews of aerial manipulation in [26] and [27]. The main challenge within aerial manipulation is the change of the UAV dynamics when interacting with its environment. The applied control algorithms need to adapt to this change in order to remain stable in flight. Hence, low-level access and a modification of the applied control algorithms are required. Additionally, in order to interact with the environment, a precise localization and environmental perception are required.

Multi-agent systems The third area covers multi-agent systems which includes the deployment of cooperative UAVs or swarms to complete certain tasks but also communication methods for distributed systems. Regarding the latter, an emerging trend are interconnected UAVs using the mobile network, e.g. LTE and 5G [28]. Considering cooperative UAVs and swarms, free moving agents and physically coupled systems can be distinguished. The most famous commercial application of a UAV swarm with free moving agents is Intel's light show. Using up to 500 custom drones with ultra bright LEDs, colorful animations can be displayed at the night sky. In research, however, the fields of application for cooperative UAVs overlap with those described above. For example, in order to navigate through a canyon, two cooperative drones are used: One drone hovers above the canyon using GNSS, while the second drone is located in the canyon with respect to the first drone using optical markers [16]. Similarly, a popular application for physically coupled systems is the aerial manipulation. A system of cooperative UAVs that carries a commonly suspended load is presented in [29]. Despite homogeneous UAV swarms, heterogeneous multi agent systems are researched, too, like the autonomous landing of a UAV on a moving platform [30]. Focusing on the same applications, the challenges within this area are similar to the two research areas described above. Furthermore, robust and reliable communication between agents is a key component within multi-agent systems.

Conclusion To sum up, although there is great amount of commercial flight controllers available, research and development of proprietary navigation and perception components as well as complex low-level control strategies are still mandatory in order to deploy UAVs within complex mission scenarios and advanced applications. Depending on the target application, the augmentation of existing open-source autopilots might be sufficient while in other cases an adaption of critical components like the ego-motion estimation and low-level control are required. In case of the latter, the development of an own flight controller can be beneficial since it grants full access but even more important a complete understanding of the flight controller's core components and their interaction.

Mission Scenarios Two different mission scenarios are considered in this work. The first scenario is addressed by the ROBEX project with focus on robots in extreme environments. The goal of ROBEX is to develop an autonomous UAV for the deployment in an Arctic environment. The second scenario is developed as a part of the MIDRAS project which aims to develop a micro drone defense system. Within MIDRAS, two cooperative UAVs carry a net by means of formation flight in order to safely capture a potentially harmful drone in mid-air. An early vision for each mission scenario is depicted in Figure 1.1.

(A) UAV in Arctic Environment. (B) Cooperative UAVs.

FIGURE 1.1: Early project concepts.

1.1.1 Robotic Exploration of Extreme Environments

Within the Helmholtz Alliance "Robotic Exploration of Extreme Environments – ROBEX", project partners from several institutes involved in space and maritime research aimed to jointly develop technologies for the exploration of highly inaccessible terrain, such as the deep sea and Arctic regions, as well as the Moon and other planets.

Among other deployments, a Autonomous Underwater Vehicle (AUV) of the Alfred Wegener Institute is operated to investigate physical, biological and chemical processes in the marginal ice zone [31]. Therefore, the AUV is launched from the research vessel Polarstern close to the ice edge and follows a pre-programmed trajectory. In order to safely deploy the AUV, various characteristics of the ice edge need to be identified beforehand. This also includes the drift velocity of the ice edge to due wind, currents and waves. The tracking of the ice edge movement can be done using GNSS receivers, however, the manual deployment of those tracking devices is time consuming and dangerous, since for deployment typically a Zodiac needs to be operated close to the ice edge. Within ROBEX, the manual deployment of the GNSS trackers was replaced by a semi-autonomous UAV. Apart from tracking the ice movement, additional light intensity measurements should be carried out on ice as a reference for the measurements conducted underwater by the AUV. The mission scenario is illustrated in Figure 1.2.

The complete mission scenario can be divided into three operating phases. First, the UAV is manually started, flown above a suitable ice floe and landed semi autonomously on ice providing a real-time video link. The semi-autonomous landing allows an automatic velocity controlled descent, while the pilot keeps control over the horizontal movement. In the second phase, the UAV rests on the ice floe and records its current position and velocity as well as the light intensity measurements. After a certain time, the UAV is commanded to fly back to the ship autonomously.

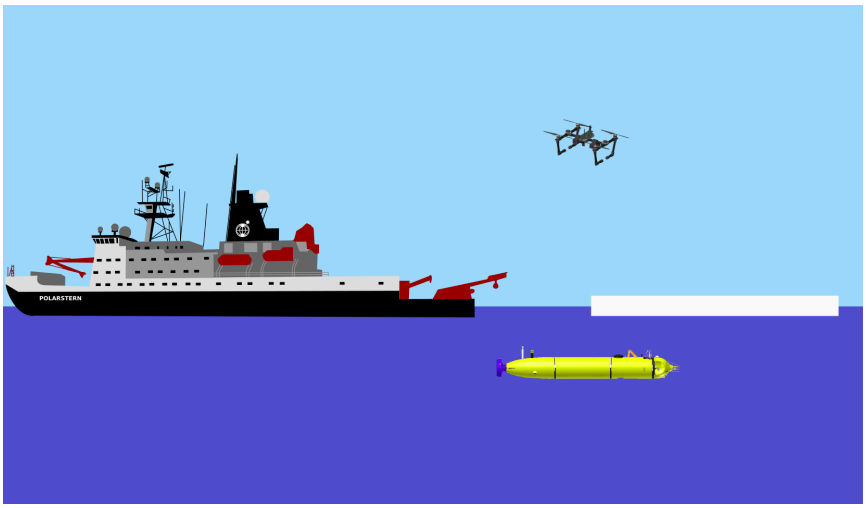


FIGURE 1.2: ROBEX mission scenario.

However, due to the shape of the magnetic field in Polar regions, the weak horizontal component as well as the low resolution of Micro-Electro-Mechanical System (MEMS) magnetometers and magnetic field disturbances in ship vicinity, it is impossible to rely on a magnetic compass for heading estimation only.

The main challenge within this mission is therefore the reliable navigation in an Arctic environment which is mandatory for an autonomous flight. Hence, in a challenging environment like this suffering from a non-uniform magnetic field and a weak horizontal component of the magnetic field, additional heading information is required. Within this context, it is important to note that for a flying vehicle in general, course and heading refer to different directions. Following aviation conventions, the course is the direction of movement as a result of the own propulsion and superposed environmental effects such as wind. Conversely, the heading refers to the direction the aerial vehicle is pointing to with respect to the geographical North. While the UAV course can be estimated from its velocity, the heading is not observable using a sequence of reference positions only.

Six other methods to obtain a reliable heading estimation apart from using a magnetic compass are described in [32]. One of them suggests the use of two or more GNSS receivers rigidly mounted on a single platform. If the GNSS receiver constellation can be reconstructed accurately enough, it can be used as attitude information. In case that two single-frequency GNSS receivers are used, the heading accuracy depends on the distance between the receiver pair, the so called baseline length [33, 34]. Throughout literature, the baselines range from 1 m for cars up to 40 m or more for aircraft and ships [35–38]. Nevertheless, according to [39], smaller baselines down to 2–3 times the L1 carrier wavelength ($\lambda_{L1} = 19.05$cm) are possible, too, allowing the use of a GNSS compass on-board of a small UAV. Regarding the heading accuracy, an upper boundary for the Root-Mean-Square Error (RMSE) is:

$$\psi_{RMSE} \leq \frac{0.5^\circ}{\text{baseline in [m]}} \tag{1.1}$$

In conclusion, this mission scenario requires autonomous flight capabilities such as automatic landing and waypoint flight, a long range communication without external infrastructure and a GNSS compass instead of a magnetic heading reference in order to allow a reliable navigation.

1.1.2 Micro-Drone-Defense System

As part of the Federal Ministry of Education and Research funded project "Mikro-Drohnen-Abwehr-System – MIDRAS" (engl. Micro-Drone-Defense System), the University of Wuerzburg and its partners had the goal to enhance existing micro-drone-defense systems by developing innovative techniques for the detection and the defense of micro drones. The different subsystems developed within this context should enable both, the detection and the classification of drones, as well as the use of situation-specific countermeasures.

The MIDRAS mission scenario is sketched in Figure 1.3. Within this scenario, a certain perimeter around critical infrastructure, e.g. around an airport, must be protected from unauthorized access by drones. After the initial detection of an approaching intruder, the exact position and speed is estimated by a combination of optical, acoustic and radio-based positioning systems which are developed by associated project partners. Depending on the risk assessment, various defensive measures can be initiated. For example, the intruding drone can be disturbed by using GNSS spoofing techniques or by jamming the remote control radio channel which typically causes an uncontrolled crash of the intruder. A less invasive countermeasure that is developed and described within the context of this work. It is based on two cooperative drones carrying a net in order to catch a potentially harmful drone in mid-air and thus safely eliminate the threat.

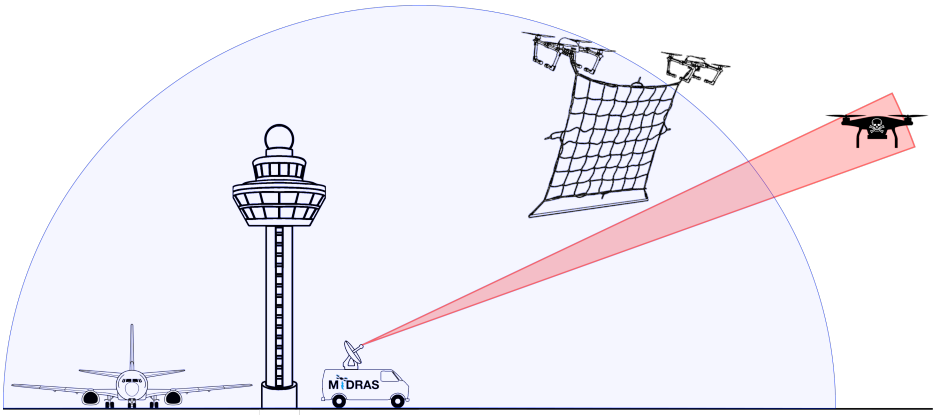


FIGURE 1.3: MIDRAS mission scenario.

The development of such a system has to face the challenges which are typical for cooperative UAVs that perform some kind of aerial manipulation. First, since both UAVs carry a net and thus a commonly suspended load, the relative positioning between both UAVs must be very accurate to prevent the individual drones and hence the entire system from oscillating. Additionally, having two physically coupled drones, the control scheme of the overall system has to be adapted. Second, the respective control algorithms have to compensate the drone impact into the net and subsequently the increased weight of the overall system. Therefore, an adaptation of the control parameters is required online in real-time.

The required centimeter positioning accuracy can be obtained using carrier phase-based differential GNSS [40]. In order to adapt the respective control algorithms, however, critical low-level components of the respective flight controller need to be modified. Hence, a realization of this mission scenario is only possible if an open-source flight controller is heavily modified or a custom one developed.

1.2 Concept

To meet the challenges of the two mission scenarios described above, a suitable flight controller concept needs to be developed. Within ROBEX, a reliable magnetic heading estimation is not possible. Therefore, the magnetic compass has to be replaced by a GNSS compass. The biggest challenge within MIDRAS is to provide very accurate and precise relative position estimates for each drone in real-time in order to avoid system oscillations.

GNSS Compass In order to estimate the UAV heading in the presence of a non-uniform and heavily disturbed magnetic field with a weak horizontal component, two GNSS antennas should be mounted rigidly on a UAV frame in H-configuration. The H-configuration allows to keep a maximum distance between both antennas without relying on additional beams. However, due to the selected configuration, the estimated heading needs to be corrected by 90 °, the offset between the nominal UAV heading ψ and the GNSS compass direction ψ_{GNSS}. The GNSS compass concept is depicted in Figure 1.4.

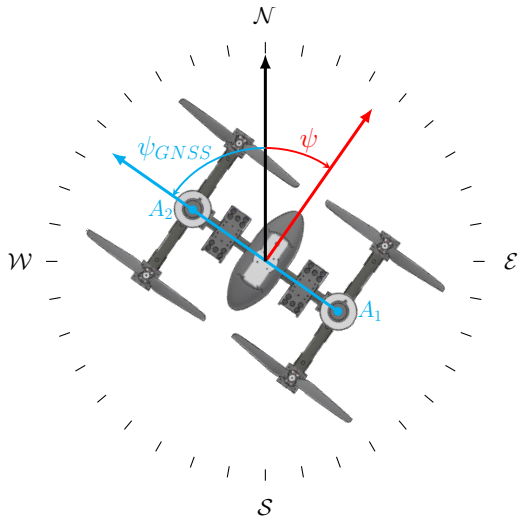


FIGURE 1.4: Concept for the GNSS compass.

If the relative position between the two antennas A_1 and A_2 is estimated accurately enough, a reliable heading solution can be computed based on the baseline orientation. Since the accuracy of single frequency GNSS receivers and standard navigation techniques is not high enough, a differential GNSS technique based on raw and carrier phase observations is applied. This approach is commonly referred to as Real-time Kinematic (RTK) positioning. Using carrier phase observations and RTK positioning in the UAV concept shown above with a baseline of 50 cm, a heading determination with a theoretical RMSE of about 1 ° is possible according to Equation 1.1. However, pure RTK approaches applied to UAVs suffer from GNSS signal lock losses or cycle slips caused by poor receiving conditions, aggressive flight maneuvers or electromagnetic interference from the UAV propulsion system. Therefore, in this thesis the traditional RTK positioning is coupled with the UAV's attitude determination. The GNSS compass provides absolute heading observations that allow to correct for gyroscope drift, while the inertial navigation system of the UAV compensates for poor receiving conditions and stabilizes the system between two consecutive valid GNSS heading observations.

Relative Positioning In order to estimate the relative position between two or more UAVs with centimeter accuracy at a high rate, carrier phase-based RTK positioning is combined with the inertial navigation system. In contrast to the GNSS compass setup, the baseline between moving receivers is not fixed and hence unknown. This can be compensated for by integrating UWB radio ranging into FARN which allows to measure the distance between two UAVs in real-time. Consequently, the radio range observations can be used to augment the RTK positioning. Additionally, a multi-sensor fusion and ego-motion estimation framework is deployed. The ego-motion framework estimates the UAV's pose and twist by merging typical UAV sensors, like a IMU, a barometric pressure sensor or different proximity sensors, with position and velocity observations from various sensor systems, including RTK systems, optical tracking setups or VSLAM sensors.

Within the context of carrier phase-based RTK positioning, two different setups can be considered, namely fixed and moving base scenarios. Both setups can be additionally augmented by auxiliary radio range measurements. Figure 1.5 illustrates the two different scenarios. For applications that require accurate absolute positioning, a fixed base station located at a well known position can be used. If only the relative position between two moving vehicles is required, e.g. between two cooperative drones, or no additional infrastructure is available, one of the vehicles can act as a moving base.

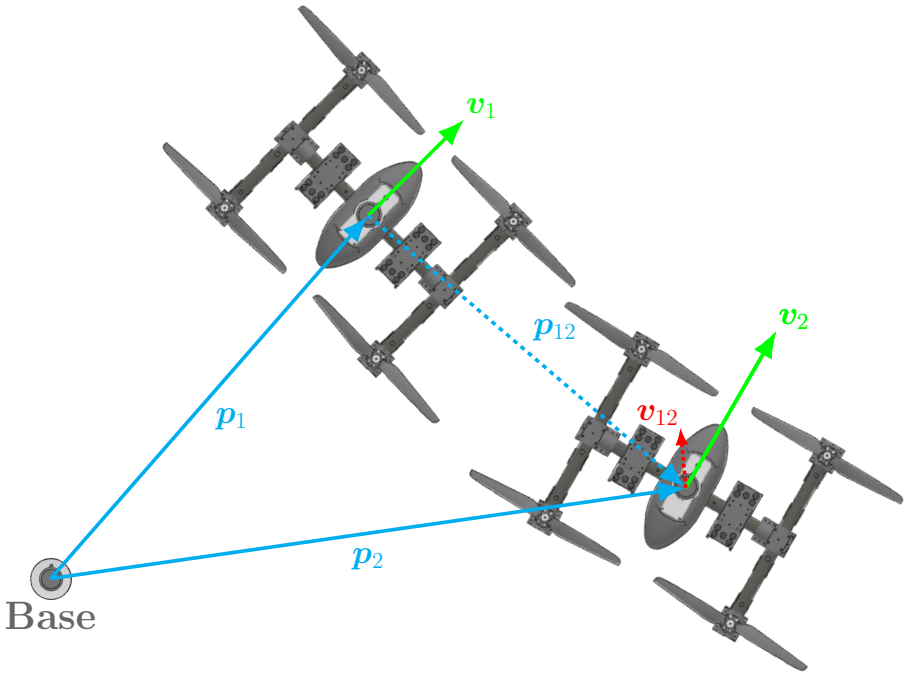


FIGURE 1.5: Different RTK base setups. The fixed base observables are indicated with solid lines, the moving base setup with dashed lines. Only relative positions and velocities are observable.

Irrespective of this, the developed positioning system should also function for the case that UWB range information is not available. In this case, conventional carrier phase-based RTK positioning is used. The system is realized using a master-slave architecture, where a single master is used as reference for several slaves. The master can be a stationary base with a known position or a moving platform that localizes itself using standard GNSS positioning.

System Architecture Both navigation concepts should be closely integrated into the attitude and ego-motion estimation of a suitable flight controller. Therefore, at the beginning of this work, a suitable platform had to be selected which allowed an easy integration of the advanced navigation methods as well as a sufficient amount of autonomous functionalities that are required within the previously described mission scenarios. Since a custom designed flight controller was already available at the Chair of Aerospace Information Technology at the University of Wuerzburg, its functionality was compared to open-source solutions first. The available flight controller was developed within the AQopterI8 project and the doctorate of Dr. Nils Gageik [41]. It is targeted towards autonomous indoor applications providing a reactive collision avoidance system. Comparing the available flight controller to open source projects like ArduPilot [42], the latter provide a wider range of functionality at the expense of a more complex software. However, since neither the custom developed flight controller nor the open-source solution provided an easy way to integrate the components and algorithms required for a successful mission at the time, the development of a new flight controller was considered. Decisive for the development of an own flight controller was ultimately the desire to use the Real-time Onboard Dependable Operating System (RODOS) in a terrestrial application. RODOS is a Real-time Operating System (RTOS) that was originally developed at the German Aerospace Center (DLR) and is currently maintained by the Chair of Aerospace Computer Science at the University of Wuerzburg.

A conceptual overview of the different tasks and hardware components of FARN is shown in Figure 1.6. The ego-motion estimation is at the core of the developed flight controller. It receives observations from different sensor systems and combines them in a single pose and twist state estimate. This estimate is subsequently utilized by the low-level control which regulates the UAV motion. Depending on the real-time requirements, the computational expenses and complexity as well as the utilized hardware interfaces, all tasks can be grouped into two categories, namely high- and low-level tasks.

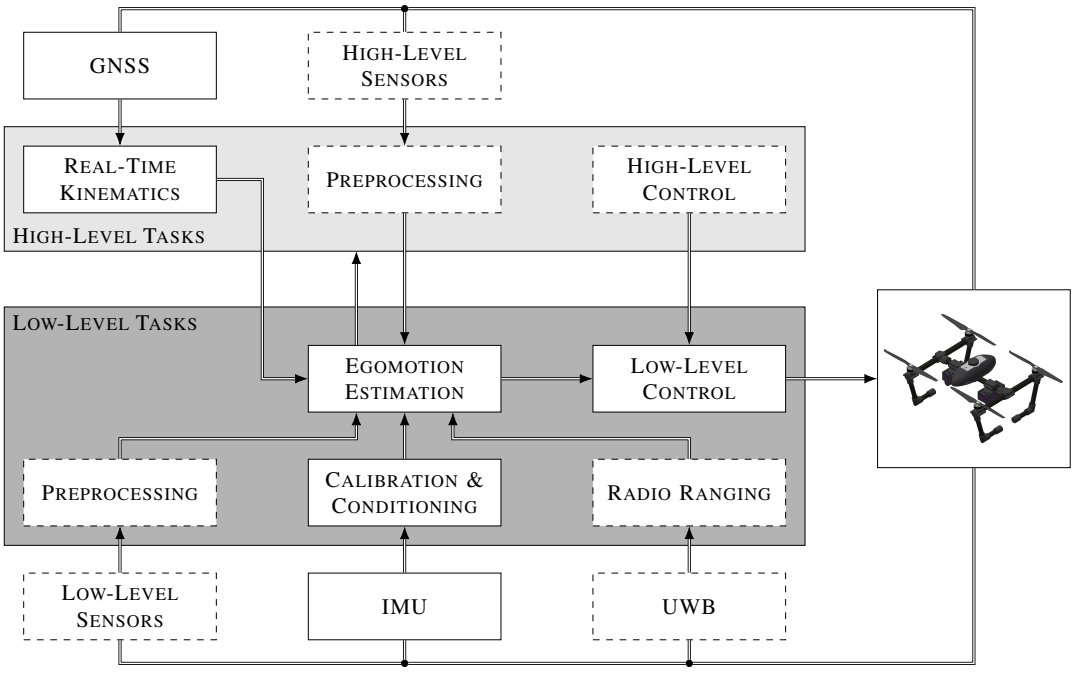


FIGURE 1.6: FARN system architecture. Optional sensors and tasks are indicated with dashed lines.

High-level tasks are tasks that can be described by soft real-time requirements, high computational effort or sensors that rely on complex hardware interfaces (e.g. USB) with a high amount of data per measurement. They include high-level control such as the flight trajectory planning and mission control as well as the processing of sensor data from sensors with comparatively low sample frequencies. High-level sensor processing includes the computation of RTK GNSS navigation solutions based on carrier phase observations and raw data as well as high-level communication or image processing tasks. Both, the GNSS compass as well as the relative positioning of several UAVs are considered as high-level tasks.

In contrast, low-level tasks need to meet hard real-time requirements in order to allow the UAV ego-motion estimation and the low-level actuator control at a high rate. The ego-motion estimation combines measurements obtained in both, hard and soft real-time, with calibrated and conditioned IMU measurements. The estimates are provided to the low-level control and different high-level tasks. Low-level sensors are characterized by relatively low data amounts per measurement and comparatively high update rates that utilize rather simple interfaces (e.g. SPI, I2C, UART). Low-level sensor processing with hard real-time constraints includes the processing of IMU measurements, proximity sensors and UWB radio ranging tasks which require very precise timings.

A heterogeneous dual-core platform is selected as hardware platform as shown in Figure 1.7. An application core is responsible for high-level tasks, while a real-time core handles all low-level tasks. The application core runs Ubuntu and the Robot Operating System (ROS) while the real-time core relies on RODOS. Compared to a standard approach with two separate processors, this design offers several advantages despite a small form factor. First, both processors share a common clock source which simplifies time synchronization between both cores since they are subject to the same clock drift and thermal changes. Second, both processors have access to shared memory and peripherals. The memory access is controlled during run-time by a messaging unit while the resource domain controller assigns different interfaces during boot. Using shared memory a high bandwidth communication between both cores is possible in real-time. Depending on the UAV configuration and the attached payload, different interfaces can be assigned to either core.

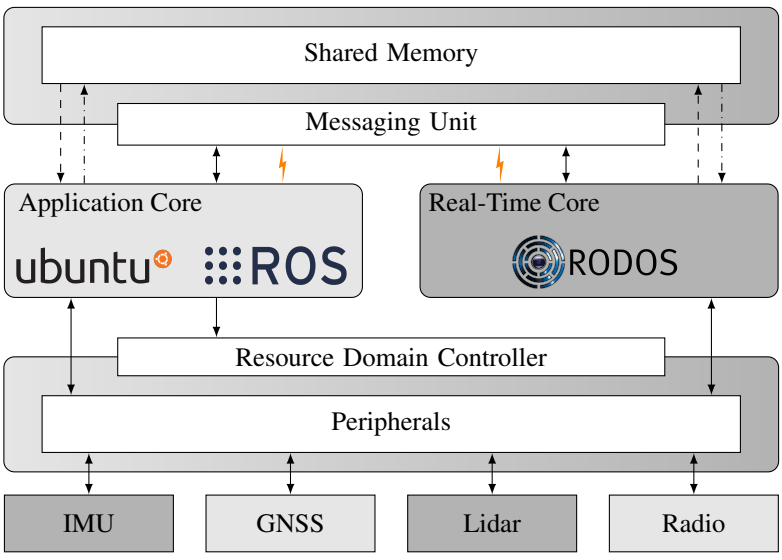


FIGURE 1.7: Dual-core with shared memory and peripherals.

1.3 My Contribution

The most important scientific contribution of this work are the different methods and technologies that are developed for FARN. The RODOS-based flight controller FARN allows to perform autonomous tasks that require highly accurate and reliable navigation using carrier phase-based GNSS positioning. In particular, FARN enables the autonomous flight in Polar regions where the deployment of a MEMS magnetic compass is not feasibly and, additionally, the relative navigation of several UAVs within a swarm. In the context of FARN, the following technologies are developed:

Versatile Ego-motion Estimation Framework A reliable ego-motion estimation is the core and therefore the most important component of every flight controller. Throughout this work, different state estimation algorithms were developed, implemented, evaluated and continuously improved. Within this development, relatively simple complementary filters [41], highly efficient quaternion-based attitude estimators [43] and complex state estimators based on Extended Kalman Filters (EKFs) [44] were investigated. The final estimation framework is based on an Error-State Kalman Filter (ESKF) and runs in real-time at up to 1 kHz. It allows an easy integration of various sensor systems like a GNSS compass, different inertial sensors as well as various external and vehicle mounted positioning systems. The framework estimates the pose and the twist of the UAV. Furthermore, inertial sensor biases and magnetic disturbances are estimated and thus can be efficiently compensated for. Measurement delays of different sensors or sensor systems can be compensated for, too, by providing a mechanism to integrate time delayed observations. Regarding the complexity, functionality and performance, the developed ego-motion estimation is on the same level as the state-of-the-art of current open source flight controllers. Within this context, extensive sensor models for low-cost MEMS sensors are derived and their respective noise characteristics empirically determined.

Modular Control Architecture for Multi-rotors Since the flight controller that was previously developed at the Chair of Aerospace Information Technology was limited to position and attitude control only [41], the control architecture is expanded within this work to allow different levels of control. Therefore, a cascaded control architecture is proposed, developed and implemented that allows a dedicated control of the UAV's position, velocity, attitude and its angular rates. The proposed control architecture is targeted towards fixed multi-rotor UAVs and can be easily adapted to match different motor configurations. The implementation of the control architecture has a modular design so that individual levels within the control cascade can be easily modified or exchanged. The basic principle of the cascaded control approach can be compared to the state-of-the-art of current open source flight controllers.

GNSS Compass for UAVs Within the ROBEX project, a GNSS compass is developed and integrated into the FARN's navigation system to allow a reliable heading determination in Arctic environment. A commercial low-cost Global Positioning System (GPS) heading system developed by ANAVS was evaluated within this project but could not be deployed successfully in flight due to the high dynamic flight characteristics of the UAV [45]. Since the development of this feature exceeded the state-of-the-art of available flight controllers at the time, existing approaches in literature are briefly discussed, before the solution developed within the context of this thesis is outlined.

Already in the early 1990s, research on GPS-based attitude determination for aircraft was successfully deployed in flight [46, 47] and established expensive multi-frequency GNSS receivers as the state-of-the-art solution. However, the upcoming of inexpensive MEMS sensors and the growing demand for smaller and lightweight platforms for various tasks encouraged the development of less expensive GNSS attitude determination systems.

A popular method combining low-cost MEMS IMUs and multiple GNSS receivers employs tight-coupling, meaning that a single Extended Kalman Filter (EKF) is utilized for attitude estimation based on double differenced carrier phase observations [48–51]. Using the double differences technique, an unknown integer ambiguity needs to be determined which can be done using different methods. While approaches with a two antenna configuration rely mostly on the Least-squares AMBiguity Decorrelation Adjustment (LAMBDA) [52], additional angular constraints are introduced on platforms with three or four antennas in order to reduce the integer search space. Using the latter constellation, a complete attitude estimation is possible. While the tightly-coupled approach promises RMSEs of less than 0.25°/baseline in [m] [49], existing solutions are limited to update-rates of 100 Hz in real-time implementations or post-processing using MATLAB due to the high computational load.

A promising loosely-coupled approach is presented in [53], where a-priori baseline information is used as a so called soft baseline constraint in order to improve the float approximation of the integer ambiguity. The baseline estimate based on LAMBDA fixed integer ambiguities is subsequently used to update the heading estimation of the UAV. One advantage of the loosely-coupled approach is the possibility of running two separate state estimators simultaneously. A hard-real time state estimator can keep track of the attitude throughout aggressive flight maneuvers , while the GNSS observations can be processed using synchronized attitude estimates at a much lower frequency.

Within this work, an attitude estimation framework based on single-frequency GNSS observations and double differences is loosely-coupled to the developed ESKF for ego-motion estimation. In contrast to solution presented in [53], in addition to soft baseline constraints so-called hard baseline constraints are added in order to improve and validate the integer fixing step using LAMBDA. Additionally, the proposed solution requires low-cost components only. The developed GNSS compass can estimate the current UAV heading at up to 10 Hz which is fast enough to estimate the gyroscope drift within the ESKF reliably. The ESKF provides the current UAV heading at up to 1 kHz.

UWB Augmented Relative GNSS Positioning Within MIDRAS, a precise relative navigation scheme for physically coupled UAVs using carrier phase-based positioning is developed and integrated into FARN. Similar to the GNSS compass, precise positioning using UWB augmented carrier phase-based GNSS positioning exceeds the current state-of-the-art of UAV flight controllers. Hence, existing research approaches are briefly discussed before the developed solution is described, too.

The basic setup for a simple carrier phase-based relative positioning system is described in [40]. The main required components are GNSS receivers that provide raw and carrier phase observations from the base and the moving platform, a communication link and a system that runs the carrier phase-based positioning algorithm, e.g. a ground station PC. The vast majority of studies on carrier phase-based

UAV relative positioning rely on fixed base stations and ambiguity resolution using LAMBDA [54–58]. A carrier phase-based positioning system that does not rely on integer ambiguity resolution is presented in [59]. Here, the authors propose a method applying a real-time sliding-window estimator that tightly integrates differential GPS and IMU observations.

However, with the increasing popularity of low-cost UWB modules, several authors have considered possible ways of combining UWB and GNSS on small flying platforms. In [60] a complementary architecture for a multi agent system consisting of UWB and GNSS in case of GNSS outages is proposed. GNSS code observations and IMU measurements are fused in a tightly-coupled approach, while carrier phase observations are used to estimate the relative distance between UAVs and to calibrate the range radios. If a GNSS outage occurs, partial outages can be recovered using the GNSS enabled UAVs as tie-down points/anchors and allowing to estimate the position of the UAVs with GNSS outage using UWB radio ranging.

Another approach for the relative positioning of two aircraft flying in a close formation is based on a three stage filter algorithm. The first of the three stages is a tightly-coupled absolute positioning filter using GPS and IMU observations. Next, UWB radio ranging distances and double differences of carrier phase observations are collected from both aircraft. In the final step, the integer ambiguities are fixed using LAMBDA [61, 62]. The actual UWB/GPS fusion is done by applying soft-baseline constraints using the UWB distance estimate and a pseudo observation based on the difference between the absolute position estimates from stage one. Other non-UAV targeted applications, implement UWB augmented GNSS navigation systems in a similar manner [63–65]. In these publications, the principle idea is to utilize a network of UWB transceivers and gather additional range information that can be used as complementary filter data or RTK baseline constraint, too.

The approach developed within this work aims to be as modular as possible in order to be deployed within different scenarios. All agents that are part of the relative positioning scheme as well as the fixed base station are equipped with UWB transceivers. Using Double-Sided Two-Way Ranging (DS-TWR) the relative distances between the transceivers are estimated. The gathered UWB range information is used as baseline constraint for the ambiguity resolution during the float estimation as well as during the integer fixing step. In contrast to [61, 62], additional pseudo measurements are renounced, due to low precision and the high inaccuracies of absolute code based position estimates.

RODOS Framework Extensions Within this work, additional functionalities are added to the RODOS software framework. The mathematical library of RODOS was extended to include arbitrary matrix and vector sizes. Using generic data types, platform specific features such as an integrated single precision Floating Point Unit (FPU) as well as operations requiring higher accuracy can be efficiently implemented. Moreover, a reliable message interface is developed that allows high bandwidth real-time communication between the two system cores using shared memory. Therefore, a common data layer based on the publish/subscribe paradigm which is used by both operating systems is implemented. Furthermore, the embedded real-time operating system RODOS is ported to run as a guest on top of the Message Queue eXecutive (MQX) operating system. Additionally, several low-level drivers required for the various sensors used within this work are added. The implemented drivers range from simple interface drivers to complex measurement schemes like the UWB ranging that requires precise timing and an efficient hardware usage.

1.4 Thesis Outline

This thesis is divided into seven chapters. Chapter 2 describes the UAV hardware, including its frame, the propulsion system as well as the avionic system that was developed within this thesis. Moreover, calibration schemes and extended sensor models are derived for the different sensor types that can be interfaced with the flight controller. The described UAV frame is utilized within the ROBEX and MIDRAS projects and the respective payload configurations are described at the end of this chapter. The ego-motion estimation framework is introduced in Chapter 3. After establishing the mathematical framework for rotations and reviewing common practice on concepts for attitude representation, the Error-State Kalman Filter equations are derived. Subsequently, the further utilization of the estimated system states within the proposed control architecture is described. Chapter 4 provides the theoretical background required to develop the GNSS compass and the UWB augmented carrier phase-based positioning scheme. After a short overview and introduction to GNSS in general, measurement models, error sources and basic navigation techniques are described. Based on the mathematical concepts introduced within the previous sections, advanced GNSS navigation methods are outlined with emphasis on carrier phase-based positioning using short baselines and respective constraints. Chapter 5 provides insights into various implementation aspects, like the advanced GNSS navigation applications, different software frameworks, the utilized operating systems, implemented scheduling schemes, algorithmic optimizations and the communication mechanism between different system levels. In Chapter 6, the developed system is evaluated with focus on the ego-motion estimation framework, the control architecture as well as the advanced navigation methods developed within the context of ROBEX and MIDRAS. Finally in Chapter 7, comprehensive conclusions and an outlook for future system improvements are presented.

1.5 Nomenclature

Throughout this thesis scalars are written as non-bold characters, e.g. s or S. Vectors are represented by bold lowercase letters, such as $\boldsymbol{v}$, while matrices are bold uppercase letters, like a rotation matrix $\boldsymbol{R}$. Coordinate frames are defined with capital cursive letters, e.g. the body frame $\mathcal{B}$ and the navigation frame $\mathcal{N}$. A vector in the body frame is written as $\boldsymbol{v}_{\mathcal{B}}$, while an active rotation from the navigation to the body frame is expressed by $\boldsymbol{R}_{\mathcal{NB}}$.

Chapter 2

Mathematical Modeling of the Physical System

The development process of the flight controller software and avionics requires a robust UAV platform for testing and evaluation. Furthermore, the UAV platform should be suited for the application scenarios described in Section 1.1. The UAV used within this work is shown in Figure 2.1. In general, the proposed UAV can be divided into three subsystems: The UAV platform, the avionics and the payload.

FIGURE 2.1: UAV platform.

The UAV platform described in Section 2.1 comprises the UAV frame, the power source as well as the propulsion system and should be designed with respect to mission-specific requirements. Payload requirements are a key factor for the platform design. The avionic subsystem is described in Section 2.2 and includes the electronic components and software that are required to control the UAV. The avionic subsystem follows a generic and modular design that allows a quick and easy adaption for various missions. In addition to a minimal and mandatory sensor suite, optional sensors can be added to the avionic system. These sensors can be mounted on the drone platform itself or are set up externally. The payload can be self-sufficient or linked to the system avionics and exposes significant variations in size and weight. Two payload configurations are given in Section 2.3.

2.1 UAV Platform

The platform was originally designed in cooperation with the Alfred Wegener Institute for Polar and Marine Research for the use in Arctic environments together with a GNSS compass [66–68]. Because of its large payload capabilities, it was later adapted for and used within other missions and applications [69, 70]. The platform itself can be divided into the mechanical frame and the propulsion subsystem which are explained below.

2.1.1 Frame

A quad-rotor frame with a footprint of $50\,\text{cm} \times 50\,\text{cm}$ in H-configuration is constructed out of carbon fiber tubes and 3D printed clamps. The H-configuration allows mounting the two GNSS antennas required for a GNSS compass at the point of intersection of the transverse beam and each of the two motor arms. Hence, no additional beams are required while still having a sufficiently large distance between the two antennas. The computer models of the UAV platform in two different configurations are shown in Figure 2.2.

(A) Dual antenna config. (B) Single antenna config.

FIGURE 2.2: CAD model of the UAV platforms.

The dual antenna configuration is used if a GNSS compass is required, while otherwise the single antenna configuration is sufficient. In both configurations, a cover may be used to protect the required avionics and the electronic speed controllers, allowing to fly at moderate conditions and light rain.

The clamps are 3D printed using a Markforged MARK TWO [71] that allows reinforcing 3D printed objects with carbon fiber inlays. The additional carbon fiber inlays for a motor mounting clamp are exemplified in Figure 2.3.

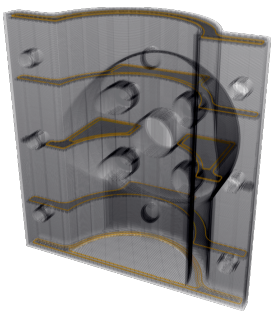


FIGURE 2.3: Carbon fiber inlays for a motor clamp.

For system modeling and simulation purposes, the mass properties and the moment of inertia for the single antenna configuration depicted in Figure 2.2 with two batteries can be estimated using the 3D computer model as:

$$\boldsymbol{I}_{cm} \approx \begin{bmatrix} 0.11 & 0 & 0 \\ 0 & 0.18 & 0 \\ 0 & 0 & 0.26 \end{bmatrix} \left[\mathrm{kg} \cdot \mathrm{m}^2\right] \quad \text{and} \quad m \approx 4.1\,\mathrm{kg} \tag{2.1}$$

2.1.2 Propulsion

The propulsion subsystem can be divided into four components as shown in Figure 2.4: The power supply, the Electronic Speed Controllers (ESCs) and the brushless motor with a propeller. The selection of each component is optimized for a long flight time and non-aggressive flight maneuvers.

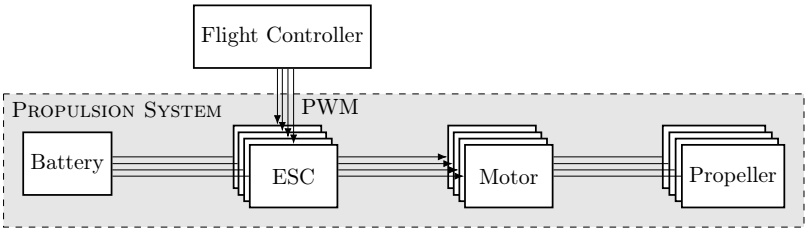


FIGURE 2.4: Complete propulsion system.

Power Supply For power supply, two 6S Hacker TopFuel Eco-X LiPo batteries [72] with a capacity of 5000 mAh and a maximum discharging C-rate of 20 are used in parallel. A single battery with its 3D printed housing weighs $m_B = 750\,\mathrm{g}$. The batteries are used to power the propulsion system, the avionics as well as payload if it does not have an independent power supply. Voltage converters are integrated into the avionics or added accordingly if required by the payload.

Electronic Speed Controllers The T-Motor 45 A 600 Hz controllers are used as ESCs [73]. They run the SimonK firmware allowing Pulse-Width-Modulation (PWM) control signals with a low time of at least 5 microseconds [74]. The maximum load is 60 A for not more than 10 s.

Brushless Motors T-Motor MN4014 400KV brushless motors are used together with 15-inch carbon fiber propellers [73]. This motor propeller combination allows a theoretical combined thrust of 10 kg for four motors. Optionally, 17-inch propellers can be mounted for higher payloads at a combined thrust of 12 kg, however the flight time is reduced.

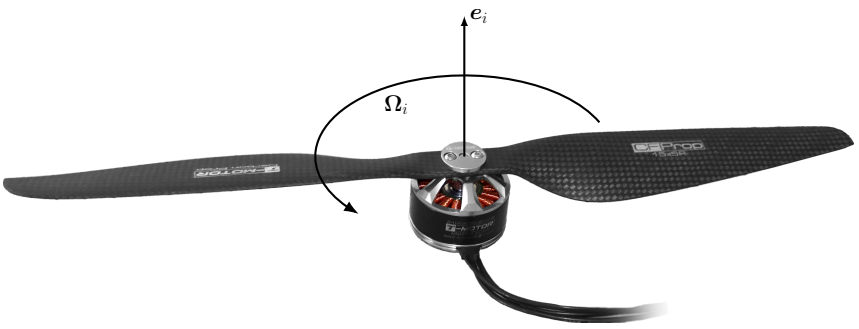


FIGURE 2.5: A motor and propeller pair.

For simulation purposes, the propulsion system needs to be modeled. Model identification is done with MATLAB. Given a certain control PWM signal $u_{PWM,i}$, the system model should produce the same thrust $\boldsymbol{f}_i(u_i)$ and torque output $\boldsymbol{\tau}_i(u_i)$ as the real combination of ESC, motor and propeller. For simplification, the power supply is assumed to be constant at 22.2 V. A more general model should include battery discharging, too. The relation between the PWM control signal and the generated thrust as well as the relation between the control signal and the generated torque are approximated using second order polynomials. The time delay caused by the ESC control loop and the motor inertia are approximated by combining a PT1-element and a dead-band, where the PT1-element is characterized by a proportional output with a first order signal delay. The propulsion system model for a single motor can be written as:

$$\boldsymbol{f}_i(u_i) = f_T(u_i) \cdot \boldsymbol{e}_i \tag{2.2}$$

$$\boldsymbol{\tau}_i(u_i) = -\text{sgn}(\Omega_i) f_\tau(u_i) \cdot \boldsymbol{e}_i \tag{2.3}$$

where $\boldsymbol{e}_i$ is the vector of thrust and Ω_i is the rotational rate generated from the motor i as can be seen in Figure 2.5. The functions $f_T(u_i)$ and $f_\tau(u_i)$ are second order polynomials describing the thrust and torque generated for a certain ESC motor command u_i, respectively, and are given by:

$$f_\bullet(u_i) = a_\bullet \cdot u_i^2 + b_\bullet \cdot u_i \tag{2.4}$$

with unknown parameters $a_\bullet$ and $b_\bullet$ that need to be identified for the thrust and torque model, respectively, e.g. $\bullet \in \{T, \tau\}$. The relation between u_i and $u_{PWM,i}$ in the time domain is described by:

$$T_\tau \dot{u}_i(t) + u_i(t) = u_{PWM,i}(t - T_d) \tag{2.5}$$

where T_d and T_τ are time constants that need to be identified. Having established the principle model, a series of measurements are conducted using the thrust and torque measurement setup shown in Figure 2.6.

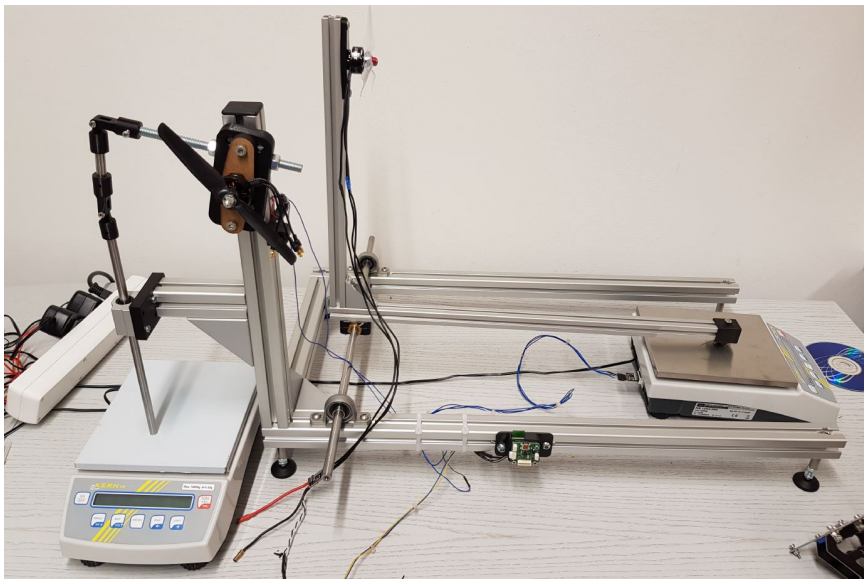


FIGURE 2.6: Thrust and torque measurement setup [75].

Thrust Two different experiments are conducted to obtain the propulsion thrust model. In both experiments, the motor is mounted vertically to reduce ground effects. Over a rotary joint, two crossbeams and a scale, the generated thrust can be measured.

For the first experiment, the motor is commanded from resting state to a certain PWM duty cycle. The maximum thrust at the steady state for each PWM duty cycle is recorded and used for the second order polynomial fit. The different step responses as well as the polynomial fit are shown in Figure 2.7. The error bars consider crossbeam and scale errors of the setup.

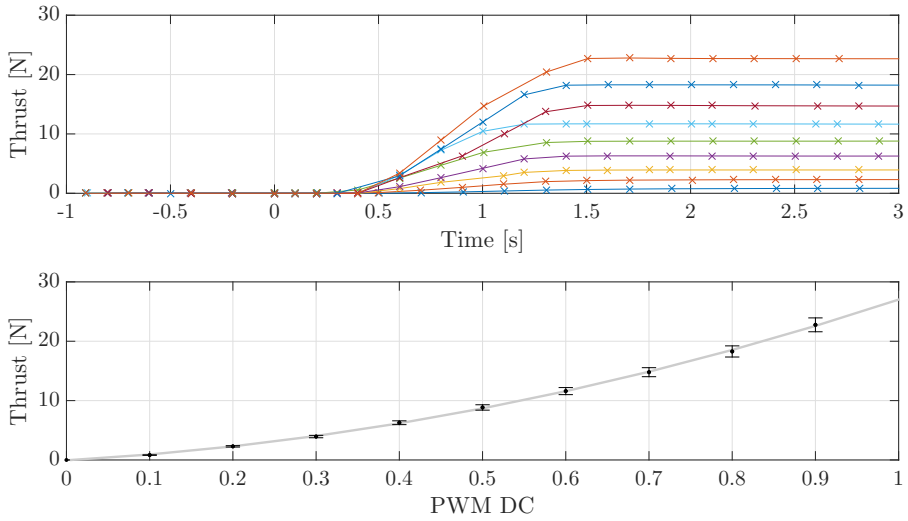


FIGURE 2.7: Step responses and steady state thrust for different PWM duty cycles.

In the second experiment, the propulsion system dynamics are analyzed. Therefore, the motor is commanded ten times to a PWM duty cycle of 75%. The measurements are combined to an average step response. The step response is modeled according to Equation (2.5) where the time constants are identified using MATLAB and the `ident` toolbox. The raw, average and modeled step response are shown in Figure 2.8. The estimated model parameters are summarized in Table 2.1.

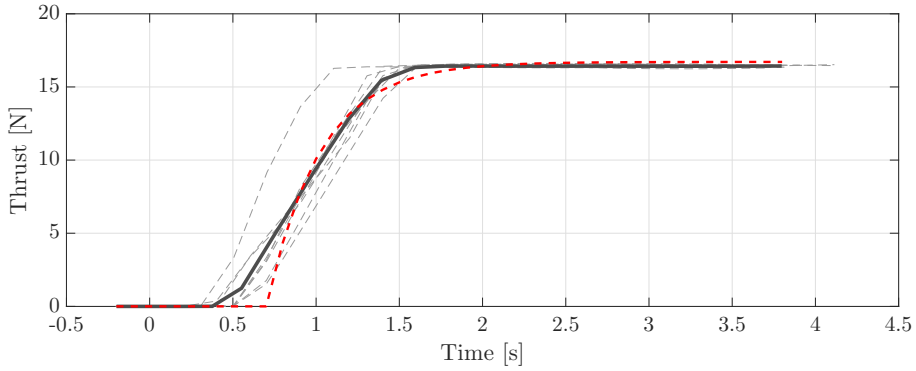


FIGURE 2.8: Raw (grey), average (black) and modeled (red) step responses for 75% PWM.

The propulsion system model is verified by comparing the system output with real system observations. A saw-tooth signal is used as input. The input signal as well as the real and the modeled system output are shown in Figure 2.9.

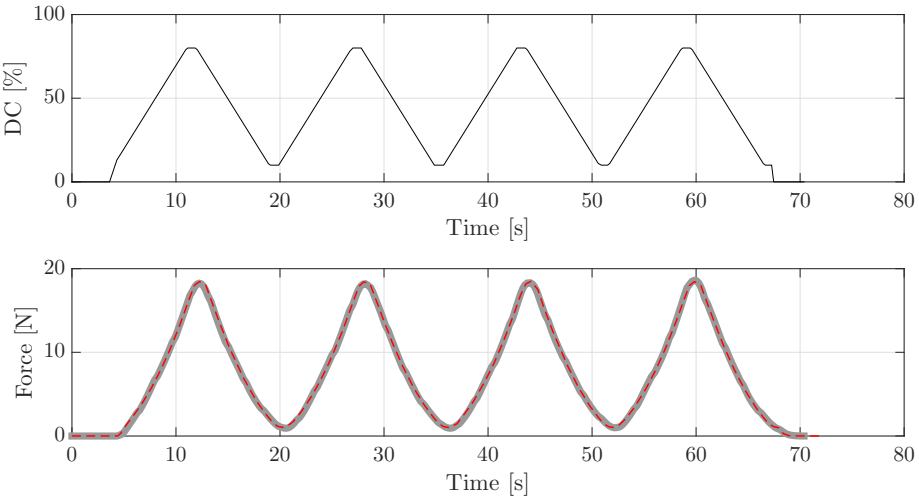


FIGURE 2.9: Thrust simulation (red) and real system (grey) output for a saw-tooth input signal (top).

Torque The torque second order polynomial is obtained in a similar way as the thrust model, while the system delay is assumed to be identical. The motor is mounted vertically again, but this time on a rotary platform. Using a flexible crossbeam connected to the rotary platform, the force generated through the counter rotation of the rotary platform can be measured. The respective torque is simply the product of the observed force and the crossbeam length. The steady-state torque is measured for different PWM duty cycles and a second order polynomial fit is obtained. The fitted model is shown in Figure 2.10, while the identified parameters are listed in Table 2.1. A system verification similar to the thrust model could not be carried out due to the low sample rate of the scale used for the experiments.

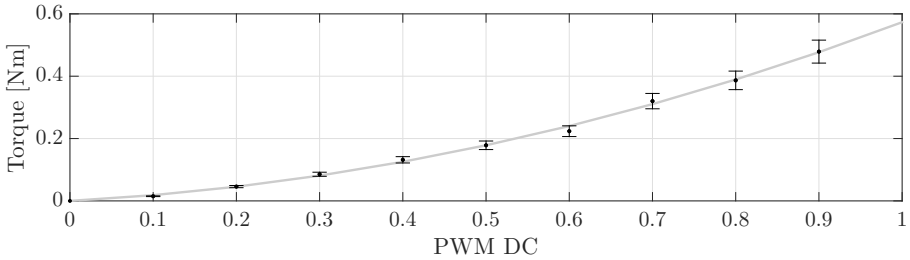


FIGURE 2.10: Steady state torque for different PWM duty cycles.

Parameter	T_τ	T_d	a_T	b_T	a_τ	b_τ
MN4014	0.321	-0.503	19.17	7.90	0.43	0.14

TABLE 2.1: Identified MN4014 parameters.

2.2 Avionics

The avionic subsystem can be divided into the flight controller, mandatory and optional sensors as well as interfaces. The flight controller is the main processing unit of the UAV and hence responsible for low-level sensor interfacing as well as the Guidance, Navigation and Control (GNC) of the UAV. The mandatory sensor suite includes inertial and environmental sensors together with radio receivers and transmitters that are able to obtain information about the UAV's ego-motion. Optional sensors can be interfaced with the developed flight controller and provide additional or complementary navigation information. Optional wireless communication devices can be used, too, to extend the drones' communication range if required.

2.2.1 Flight Controller

The flight controller is based on the Udoo Neo board from SECO [76] and extended using a custom made sensor carrier board. The two credit-card sized flight controller boards are shown in Figure 2.11.

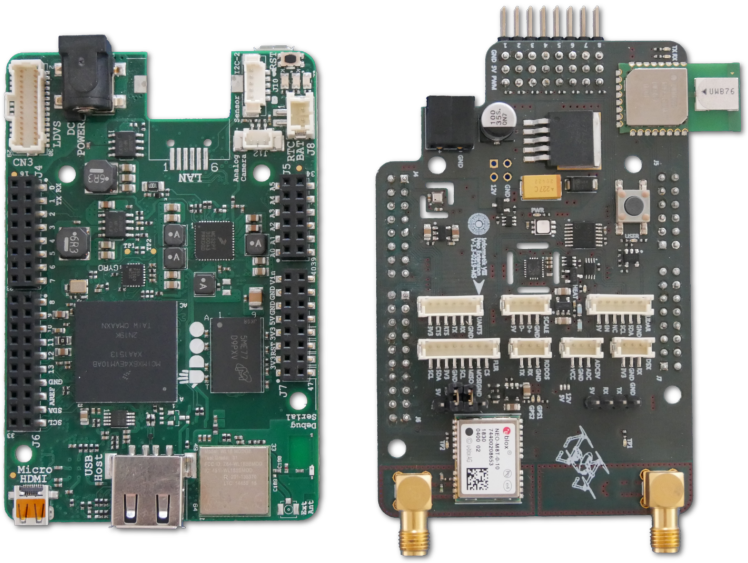


FIGURE 2.11: Udoo Neo board with custom sensor extension.

The Udoo Neo board hosts the i.MX6 SoloX consumer grade application processor MCIMX6X4EVM10AB with a heterogeneous dual-core from NXP [77]. The heterogeneous dual-core consists of an application core, a ARM Cortex-A9 at 1 GHz, and a real-time core, a ARM Cortex-M4 at 200 MHz with a single precision FPU. The application core can access 1 GB Double Data Rate 3 (DDR3) Synchronous Dynamic Random-Access Memory (SDRAM). The real-time core has 64 kB Tightly-Coupled Memory (TCM) and can additionally access 128 kB On-Chip Random-Access Memory (OCRAM) as well as a part of the application core's SDRAM. The shared part of the SDRAM is used for inter-core communication. Additionally, DC-DC converters from 12 V to 5 V and to 3.3 V, an on-board IMU and a magnetometer as well as a WiFi and Bluetooth chip are integrated onto the Udoo Neo board. Various General Purpose Input/Outputs (GPIOs), low-level interfaces such as Inter-Integrated Circuit (I2C) and Serial Peripheral Interface (SPI) as well as an analog camera interface can be easily accessed using the respective sockets or pin headers. High-level sensors can be accessed using the USB interface.

A custom designed extension board is mounted on top of the Udoo Neo board as a breakout shield using the pin headers. The extension board houses an additional DC-DC converter for up to 30 V input, the sensor suite required for the UAV's GNC system as well as interfaces for additional sensors and payloads. Eight PWM interfaces allow to control ESCs or other servo motors if required. The mandatory sensor suite includes another IMU, a magnetometer, a barometric pressure sensor, a UWB transceiver, two GNSS receivers and an input voltage monitor circuit. Each sensor has its own electrically switchable voltage regulator providing a very stable supply. A multi color status Light-Emitting Diode (LED), a user button and dedicated serial ports for both cores might be used for debugging purposes. Advanced debugging options, e.g. single instruction stepping and single register access, are possible but due to the heterogeneous architecture difficult to set up [78].

A flight controller connectivity diagram is shown in Figure 2.12. Components and interfaces depicted with solid frames are required or integrated into the extension board, while dashed components can be connected through the appropriate interfaces if required. The real-time core is connected to low-level devices that require a high sample rate, have strict timing constraints, are critical for a successful UAV flight or have simple interfaces. The real-time core is responsible for the actuator control. The application core handles the wireless communication as well as computational expensive tasks. It is interfaced with high-level sensors and payloads as well as sensors with a high data rate, such as raw data GNSS receivers. However, the respective interfaces are not strictly tied to each core, but can be configured according to the user requirements using the Resource Domain Controller (RDC). Both cores communicate using a messaging unit with hardware interrupts and access to the shared SDRAM. The inter-core communication is described in detail in Section 5.4. The remain of this section describes the mandatory and optional sensors for the UAV GNC in detail. Payload examples from real missions are described in the next section.

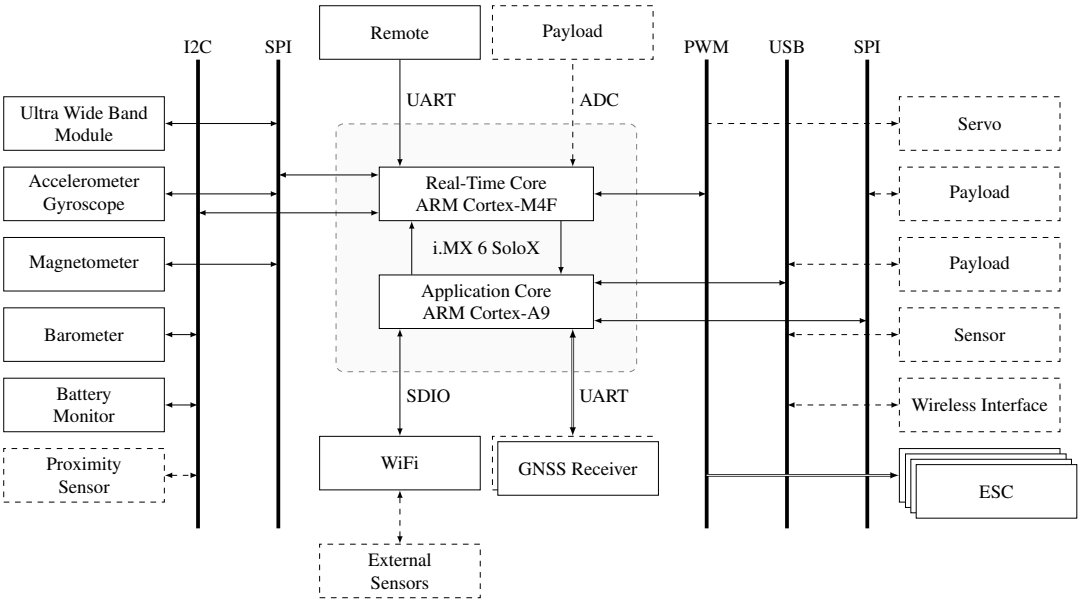
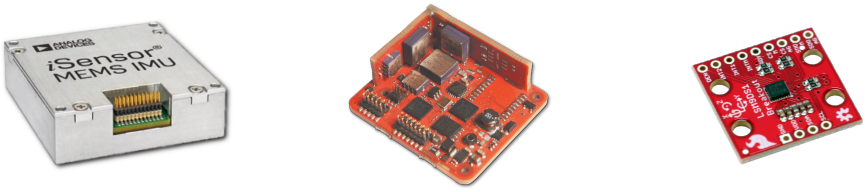


FIGURE 2.12: Flight controller connectivity: Mandatory (solid) and optional components (dashed).

2.2.2 Inertial Measurement Unit

The Inertial Measurement Unit (IMU) is the primary source for ego-motion estimation of any UAV. Traditionally, a IMU is a device that combines a 3D accelerometer with a 3D gyroscope. However, since the popularity gain of MEMS, it is quite common to include a 3D magnetometer into the same chip, too, in order to compensate for the gyroscope drift around its nadir axis. This section addresses only traditional IMUs although the developed flight controller relies on the ST LSM9DS1, a IMU with an integrated magnetometer [79]. Magnetometers are described separately in the section hereafter.

In order to understand the different error sources for IMUs, generic measurement models are derived for the accelerometer and the gyroscope in Section 2.2.2.1. Some of these measurement errors can be mitigated using different calibration techniques as described in Section 2.2.2.2. Other errors are caused by vibrations and can be reduced using appropriate software filters or mechanical countermeasures as described in Section 2.2.2.3. Based on the noise characteristics, the applicable temperature range and dependencies as well as the manufacturing process of MEMS IMUs, three different sensor grades can be distinguished: Tactical/Military-grade, industrial-grade or consumer-grade. Examples of each sensor grade are shown in Figure 2.13.



(A) ADIS16495 [80] (B) XSensMTi-100 [81] (C) LSM9DS1

FIGURE 2.13: Tactical-, industrial- and consumer-grade IMUs.

High-end MEMS sensors have excellent noise characteristics, wide temperature ranges and compensate for thermal dependencies. In contrast, consumer grade sensors are produced in large quantities and are therefore quite cheap as indicated in Table 2.2. Military- or industrial-grade IMUs are typically temperature calibrated or regulated, while consumer-grade IMUs need to compensate for temperature effects explicitly. The temperature effects on the LSM9DS1 are addressed in Section 2.2.2.4. While the noise parameters, namely the bias stability and the sensor random walk, are typically given in the respective data sheet for high-end IMUs, these parameters have to be determined experimentally for low-cost MEMS IMUs. The noise and sensor characteristics for the three different sensor grades as well as the process of estimating the noise parameters are described in Section 2.2.2.5.

Device	**ADIS16495**	**XSensMTi-100**	**LSM9DS1**
Grade	military	industrial	consumer
Type	Acc, Gyro	Acc, Gyro, Mag	Acc, Gyro, Mag
Price	2380 €	1776 €	5 €

TABLE 2.2: Different IMU grades.

2.2.2.1 IMU Measurement Modeling

IMUs, especially low-cost consumer grade IMUs, suffer from scaling and axis misalignment errors. A detailed error model for the accelerometer and the gyroscope measurements are derived in [82].

Accelerometer The accelerometer model is given by:

$$\boldsymbol{a}_{\mathcal{B},raw} = \boldsymbol{T}_{\boldsymbol{a}} \left(\boldsymbol{a}_{\mathcal{B},true} - \boldsymbol{R}_{\mathcal{N}\mathcal{B}}{}^{\top} \boldsymbol{g}_{\mathcal{N}} \right) + \boldsymbol{a}_b + \boldsymbol{a}_w \tag{2.6}$$

$$\dot{\boldsymbol{a}}_b = \boldsymbol{a}_{bw} \tag{2.7}$$

where $\boldsymbol{a}_{\mathcal{B},raw}$ is the accelerometer raw measurement in the IMU body frame, $\boldsymbol{T}_{\boldsymbol{a}} \in \mathbb{R}^{3\times3}$ is the gain and misalignment error matrix, $\boldsymbol{a}_{\mathcal{B},true}$ is the true acceleration acting on the IMU, $\boldsymbol{R}_{\mathcal{N}\mathcal{B}}{}^{\top}\boldsymbol{g}_{\mathcal{N}}$ is the Earth's gravity vector in the body frame, $\boldsymbol{a}_b$ is the accelerometer bias and $\boldsymbol{a}_w$, $\boldsymbol{a}_{bw}$ are Gaussian measurement noise and accelerometer bias instability, respectively. If vibrations act on the sensor, they are modeled by the truly observed acceleration, too. If the observation of these vibrations is not desired, appropriate countermeasures need to be considered.

Gyroscope The gyroscope is modeled by:

$$\boldsymbol{\omega}_{\mathcal{B},raw} = \boldsymbol{T}_{\boldsymbol{\omega}}\boldsymbol{\omega}_{\mathcal{B},true} + \boldsymbol{T}_{\times} \left(\boldsymbol{a}_{\mathcal{B},true} - \boldsymbol{R}_{\mathcal{N}\mathcal{B}}{}^{\top} \boldsymbol{g}_{\mathcal{N}} \right) + \boldsymbol{\omega}_b + \boldsymbol{\omega}_w \tag{2.8}$$

$$\dot{\boldsymbol{\omega}}_b = \boldsymbol{\omega}_{bw} \tag{2.9}$$

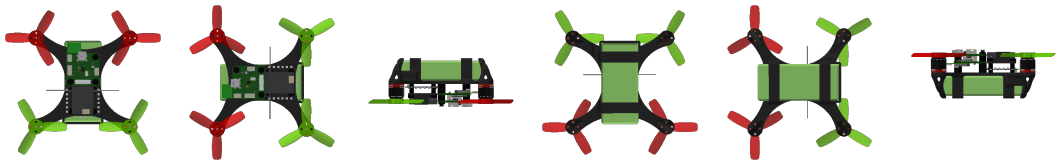
where $\boldsymbol{\omega}_{\mathcal{B},raw}$ is the gyroscope raw measurement in the IMU body frame, $\boldsymbol{T}_{\boldsymbol{\omega}} \in \mathbb{R}^{3\times3}$ is the gain and misalignment error matrix, $\boldsymbol{\omega}_{\mathcal{B},true}$ is the true angular rate acting on the IMU, $\boldsymbol{T}_{\times} \in \mathbb{R}^{3\times3}$ is transformation matrix describing the gyroscope's acceleration sensitivity, $\boldsymbol{\omega}_b$ is the gyroscope bias and $\boldsymbol{\omega}_w$, $\boldsymbol{\omega}_{bw}$ are Gaussian measurement noise and gyroscope bias instability, respectively.

2.2.2.2 IMU Calibration

The gain and misalignment error matrices $\boldsymbol{T}_{\boldsymbol{a}}$ and $\boldsymbol{T}_{\boldsymbol{\omega}}$ can be estimated using calibration methods as described in [83, 84]. However, relying on external tools, such as robotic arms or optical tracking systems, both calibration approaches are limited to specific IMU setups and do not consider axis misalignment errors due to IMU mounting on the UAV frame itself.

For both approaches, the cross sensor effect $\boldsymbol{T}_{\times}$ is considered to be small and mostly mitigated by factory calibration. Requiring a generic in field calibration of UAV mounted IMUs, a different calibration scheme is described within this work. Instead of calibrating the inertial sensors, the sensors used for complementary attitude correction are calibrated. For the roll and pitch attitude correction, the accelerometer which observes the Earth's gravitational field can be used. Depending on the application scenario, the yaw angle can be corrected using a magnetometer, a GNSS compass or optical reference systems that provide attitude information. The gyroscope calibration coefficients $\boldsymbol{T}_{\boldsymbol{\omega}}$ and $\boldsymbol{T}_{\times}$ are neglected, however, it is assumed that dynamic gyroscope bias estimates within the ego-motion estimation framework of this work are able to compensate for the scaling and misalignment errors of Equation (2.8).

Accelerometer The accelerometer calibration method is based on the work done in [85, 86]. The applied method allows a fast in field calibration of UAV mounted IMUs without the need of an expensive or external reference systems. However, a circular bubble level might be used to improve the calibration. In order to calibrate the accelerometer axis misalignment, the UAV is rotated into six different orientations as indicated in Figure 2.14 while raw accelerometer measurements are collected. The temperature is assumed to be constant throughout calibration and the bias instability is neglected for the calibration time.



(A) x-up (B) y-up (C) z-up (D) x-down (E) y-down (F) z-down

FIGURE 2.14: Drone orientations for accelerometer calibration.

For each orientation, a nominal axis of the UAV is pointing in parallel to the Earth's gravity vector, three times in the same and three times in the opposite direction. The raw data is collected over n samples for each orientation and averaged:

$$\bar{\boldsymbol{g}}_{\bullet} = \frac{1}{n}\sum_{i=0}^{n} \boldsymbol{g}_{\bullet}(i) \tag{2.10}$$

where $\bullet \in \{x^+, y^+, z^+, x^-, y^-, z^-\}$ indicates one of the six directions. The obtained measurements are stacked together as:

$$\bar{\boldsymbol{g}}_{raw} = \begin{bmatrix} \bar{\boldsymbol{g}}_{x^+} & \bar{\boldsymbol{g}}_{y^+} & \bar{\boldsymbol{g}}_{z^+} & \bar{\boldsymbol{g}}_{x^-} & \bar{\boldsymbol{g}}_{y^-} & \bar{\boldsymbol{g}}_{z^-} \end{bmatrix} \in \mathbb{R}^{3\times 6} \tag{2.11}$$

where the indices denote the axis that is parallel to the Earth's gravity vector and their superscripts indicate their respective direction. The collected data is compared to the combined expected gravity vectors:

$$\boldsymbol{g}_{true} = \begin{bmatrix} -\boldsymbol{I}_{3\times 3} & \boldsymbol{I}_{3\times 3} \end{bmatrix} \tag{2.12}$$

Since the UAV is kept stationary during the six calibration orientations, except the Earth's gravity vector no additional force is acting on the UAV, meaning that $\boldsymbol{a}_{\mathcal{B},true} = \boldsymbol{0}$. Based on Equation (2.6), six independent equations can be obtained. With this system of equations, the gain and error matrix $\boldsymbol{T}_a$ can be calculated, assuming that the measurement noise $\boldsymbol{a}_w$ is canceled out using a sufficiently big enough sample size n and that the bias remains constant throughout the calibration time, i.e. $\boldsymbol{a}_{wb} = 0$. The accelerometer bias $\boldsymbol{a}_b$ can be approximated as:

$$\boldsymbol{a}_b = \frac{1}{6}\bar{\boldsymbol{g}}_{raw} \cdot \boldsymbol{1}_{6\times 1} \tag{2.13}$$

The overdetermined equation system based on Equation (2.6) can be written as:

$$\bar{\boldsymbol{g}}_{raw} = \boldsymbol{T}_{\boldsymbol{a}} \left(-\boldsymbol{g}_{true}\right) + \boldsymbol{a}_b \cdot \boldsymbol{1}_{1\times 6} \tag{2.14}$$

The system of equations can be solved using singular value decomposition [87]:

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{U} \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{V}^\top \\ &= (\bar{\boldsymbol{g}}_{raw} - \boldsymbol{a}_b \cdot \mathbf{1}_{1\times 6}) \cdot (-\boldsymbol{g}_{true})^\top \end{aligned} \tag{2.15}$$

where $\boldsymbol{U} \in \mathbb{R}^{3\times 3}$ and $\boldsymbol{V} \in \mathbb{R}^{3\times 3}$ are orthonormal matrices and $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times 3}$ is a rectangular diagonal matrix. The gain and misalignment error matrix $\boldsymbol{T}_a$ is then given by:

$$\boldsymbol{T_a} = \boldsymbol{U} \cdot \boldsymbol{V} \tag{2.16}$$

Figure 2.15 shows the raw values $\bar{\boldsymbol{g}}_{raw}$, true reference direction as well as the calibrated result $\boldsymbol{T}_a^\top (\bar{\boldsymbol{g}}_{raw} - \boldsymbol{a_b} \cdot \mathbf{1}_{1\times 6})$ in the six directions. In the example below the corresponding Tait-Bryan correction angles around the x, y and z axis are $-0.4°$, $4.5°$ and $0.2°$, respectively.

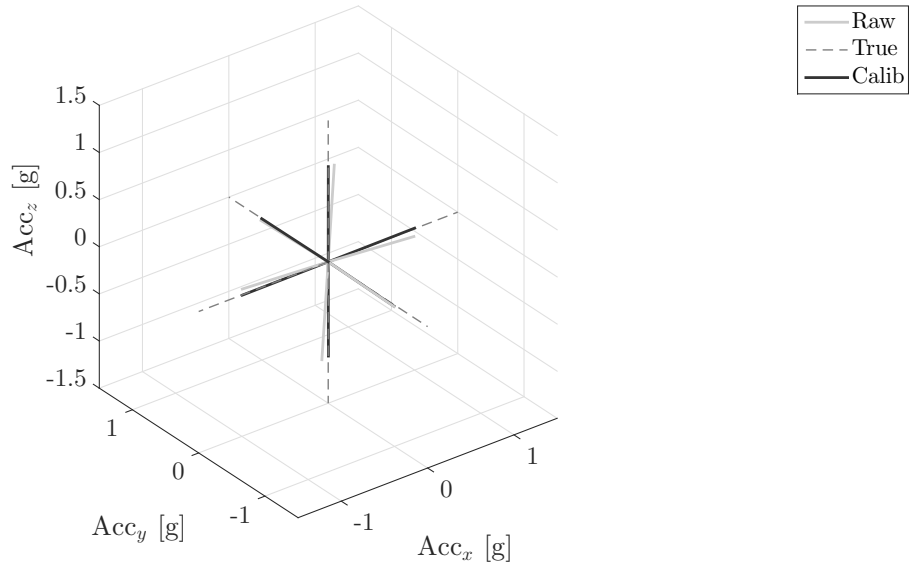


FIGURE 2.15: Raw, true and calibrated accelerometer readings.

Gyroscope As previously mentioned, no explicit gyroscope calibration is performed. Instead, it is assumed that the gyroscope errors can be compensated using an dynamically adjusted bias estimation. The dynamic bias estimation is realized using complementary sensors in the ego-motion estimation framework described in Section 3.2. However, the initial bias needs to be estimated just before take off. Therefore, the gyroscope readings for 1 second at the according data output rate are averaged.

$$\boldsymbol{\omega}_{b,0} = \frac{1}{n} \sum_{i=0}^{n} \boldsymbol{\omega}_{\mathcal{B},raw} \tag{2.17}$$

It is important that the UAV does not move during this time. If this requirement can not be met, e.g. take off on a moving platform, the initial bias is set to zero. A take off is prohibited until the bias is estimated correctly using the ego-motion framework.

2.2.2.3 Vibration Filtering

Imbalances of the rotating parts, such as the rotor and the propellers, cause vibrations that are observed by the accelerometer. Consequently, the accelerometer raw measurements are superimposed by vibrations and thus deteriorated. It is therefore necessary to mitigate the vibration effects. In [88], the frequency response of a UAV is analyzed dynamically and notch filters are applied to filter the detected noise frequencies. In [89], extensive frame modeling and noise analysis are carried out. In order to dampen the disturbances the authors suggest a combination of physical and software counter measures. In this work, the problem is addressed by combining mechanical and software based filter approaches, too. The disturbances are mitigated in a physical way using suitable vibration damping counter measures as well as signal preprocessing steps in form of digital signal filters, that eliminate the remaining vibrations.

Figure 2.16 illustrates the mechanical vibration damping, while Figure 2.17 shows the time and frequency signal of a single accelerometer axis during a UAV hover flight as raw data and with different filters applied.

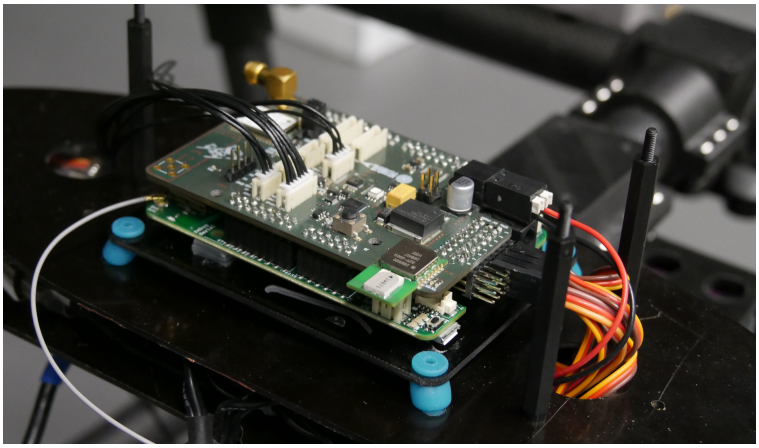


FIGURE 2.16: Vibration mitigation using rubber dampers.

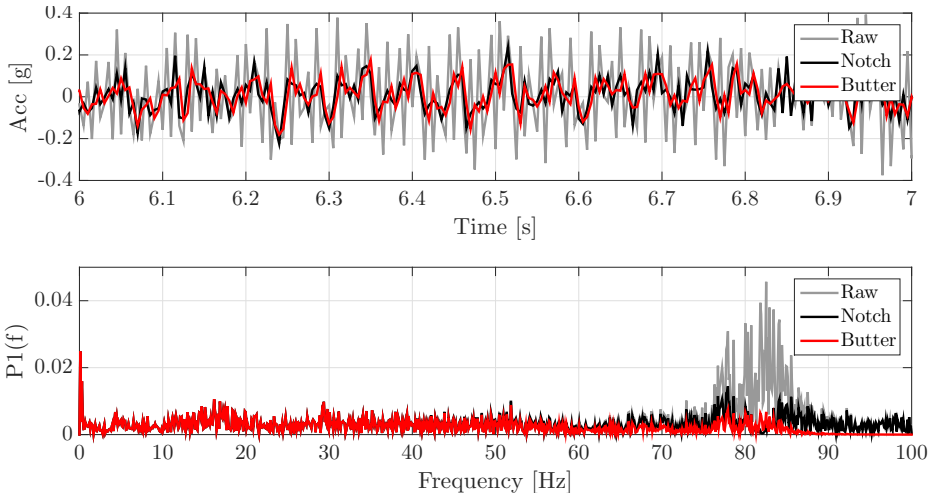


FIGURE 2.17: Raw and filtered accelerometer data: A second order notch filter and a second order Butterworth filter are compared in time (top) and frequency domain (bottom).

Despite of mounting the accelerometer using vibration dampers, noise frequency peaks in the raw signal at approximately 82 Hz can be still observed. The raw measurements are compared to the output of a second order notch filter and a second order Butterworth filter. As indicated in [88], notch filters allow to single out narrow frequency bands without introducing a phase delay. However, inspecting the frequency plot in Figure 2.17, a rather wide frequency band of approximately 10 Hz needs to be filtered out. The notch filter is therefore centered at 82 Hz and has a rather wide bandwidth of 20.5 Hz. The Butterworth filter uses a cut-off frequency of 60 Hz. Phase delay and damping characteristics for signals below 50 Hz are similar for both filters. However, since higher frequencies are eliminated using the low-pass filter, too, and since the phase delay is small for low-order Butterworth filters, it is the filter of choice. Additionally, there is no need for multiple narrow bandwidth filters, since the disturbances are concentrated around a single frequency peak.

2.2.2.4 IMU Temperature Dependencies

As MEMS sensors are silicon based, their measurement characteristics vary with temperature. Their zero measurement offset is affected mostly by changes in temperature and is different for various MEMS devices of the same type, especially for devices using low-cost consumer grade manufacturing. Scaling factors and hence non-orthogonality vary significantly with temperature, too [90]. Traditionally, temperature dependencies for low-cost MEMS are mitigated by modeling the sensor behavior in climate or thermal chambers as shown in Figure 2.18.

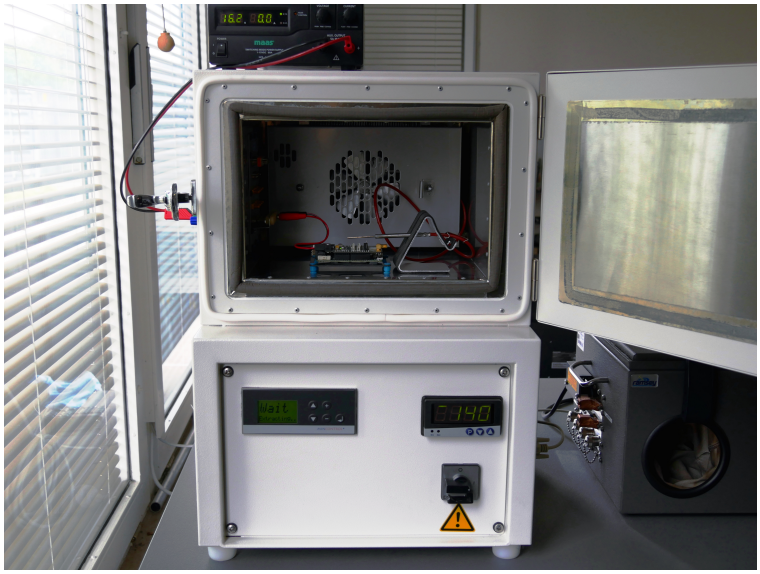


FIGURE 2.18: Flight controller in the Vötsch VT4002 thermal chamber.

While static experiments are simple and sufficient for bias estimation, scale factors and non-orthogonalities demand different and very time consuming approaches. The calibration procedure described for accelerometers applied at different temperatures can be used to estimate temperature dependencies of accelerometer scale factors and non-orthogonalities. In order to estimate gyroscope scale factors, a calibrated rotation table within the thermal chamber and a series of rotational experiments at different temperatures and orientations is required. The complex relation between temperature and bias, scale error and non-orthogonalities can be described

in different ways. A popular method relies on thermal models expressed as polynomial functions [91, 92]. The polynomial correction functions are easy to obtain in terms of curve fitting, however they do not necessarily cope with non-linearities well enough. A more robust, but also more complex approach applies neural networks for the compensation of temperature related errors [93–95].

In order to analyze the temperature dependencies of the LSM9DS1 IMU utilized in this work, the flight controller is placed inside a Vötsch VT4002 thermal chamber [96]. Power is provided using the jack plugs which are integrated into the thermal chamber, while a wireless communication link is used to log the sensor's raw data. The thermal chamber is controlled using the Vötsch serial interface and a simple Python script running on a RaspberryPi.

The orientation of the flight controller remains unchanged while the temperature is varied according to the plot shown in Figure 2.19. The temperature is measured using the IMU's temperature sensor directly. The thermal chamber is commanded to cool down from room temperature to -20°C and then to heat up to 35°C. The observed temperature is slightly higher, due to a temperature offset.

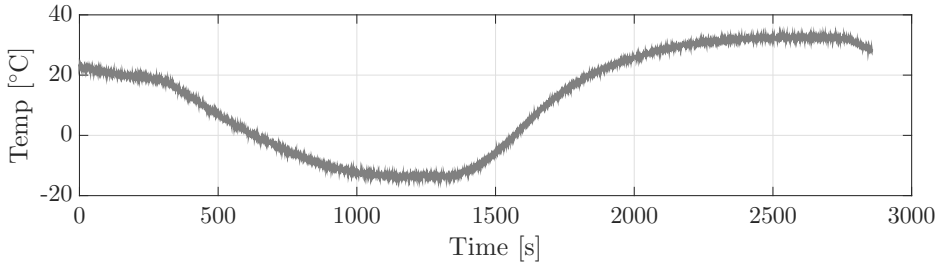


FIGURE 2.19: Thermal profile for the LSM9DS1 temperature analysis.

The raw data from the sensor is recorded and shown in Figure 2.20. The gyroscope temperature bias dependencies could be well approximated using low-order polynomials. However, the accelerometer measurements display different temperature dependencies, depending on whether the device is in the cooling or the heating phase of the temperature cycle in Figure 2.19. For the z-axis measurement, the influence of temperature dependent scaling errors can be clearly observed, too.

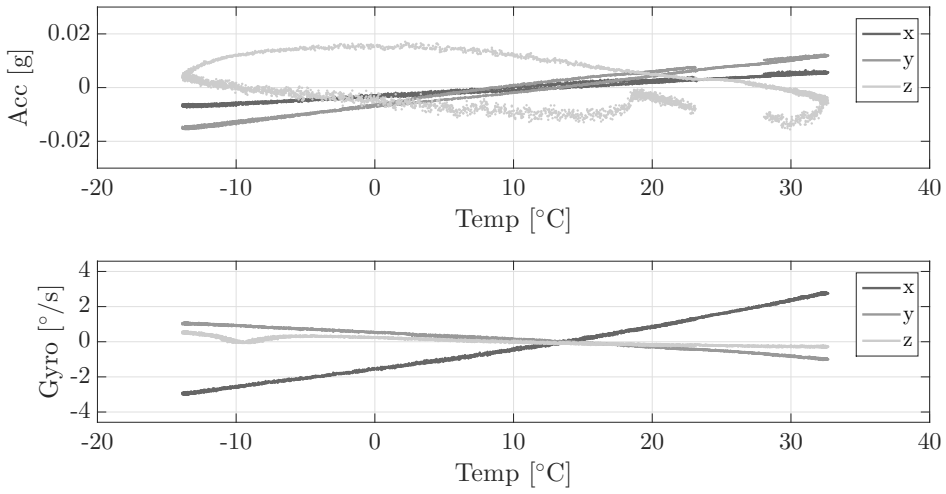
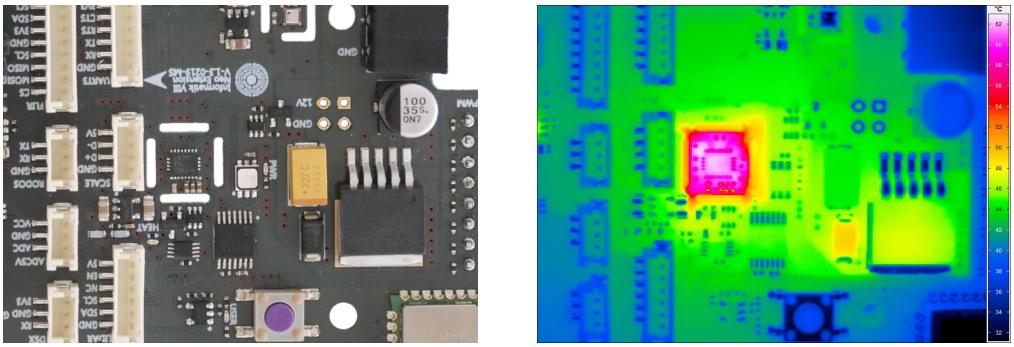


FIGURE 2.20: Temperature dependencies of the LSM9DS1.

Experiments conducted with several flight controllers yielded different thermal calibration parameters for each individual MEMS sensor. Similar to the conclusion made in [94], the implications of these experiments are that extensive thermal calibrations have to be performed for each flight controller separately. In order to perform a calibration procedure as described in [95], collecting the required data for a single sensor suite takes almost 4 days. Because of the exorbitant amount of time required for a single flight controller temperature calibration, an electric temperature stabilization instead of an algorithmic temperature compensation is implemented.

The temperature stabilization is realized using a network of resistors that is integrated into the flight controller. The heating network is located on the bottom side of the PCB, directly beneath the LSM9DS1 and has a heating power of a little more than 1 W. Using PCB through holes and PCB cut outs, a local and controlled heat transfer is possible as shown in Figure 2.21.



(A) RGB image (B) Thermal Image

FIGURE 2.21: Thermal isolation and heating of the flight controller mounted LSM9DS1.

The heating network is used to keep the temperature of the IMU at a fixed level above the surrounding temperature. The exact temperature should be selected according to the flight environment and the expected temperatures, since the heating power is limited. As a rule of thumb, 10 °C above the ambient temperature are easy to obtain and more than sufficient. The calibration procedure described in Section 2.2.2.2 should be carried out at the same temperature level.

The closed-loop thermal control is sketched in Figure 2.22. The duty cycle of a PWM signal is adjusted using a simple Proportional-Integral-Derivative Controller (PID), controlling the current flow through the resistor network. The resulting temperature is observed using the built-in thermometer of the LSM9DS1.

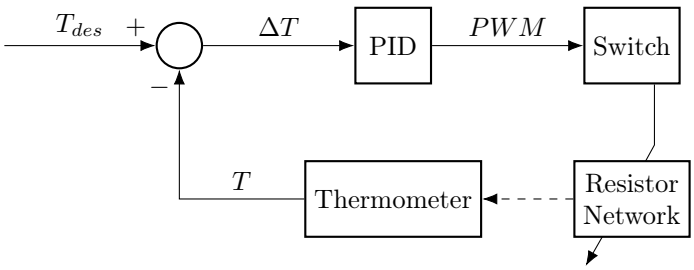


FIGURE 2.22: Temperature control loop.

Figure 2.23 shows the stability of the thermal control for $T_{des} = 40°$. Although the temperature control is very stable and the temperature fluctuations remain within the range of 0.2 °C, temperature depended scaling effects of the accelerometer are still present. However, the fluctuations caused by the temperature dependent scaling effect are limited to a few mg, allowing to consider them as measurement noise for simplification.

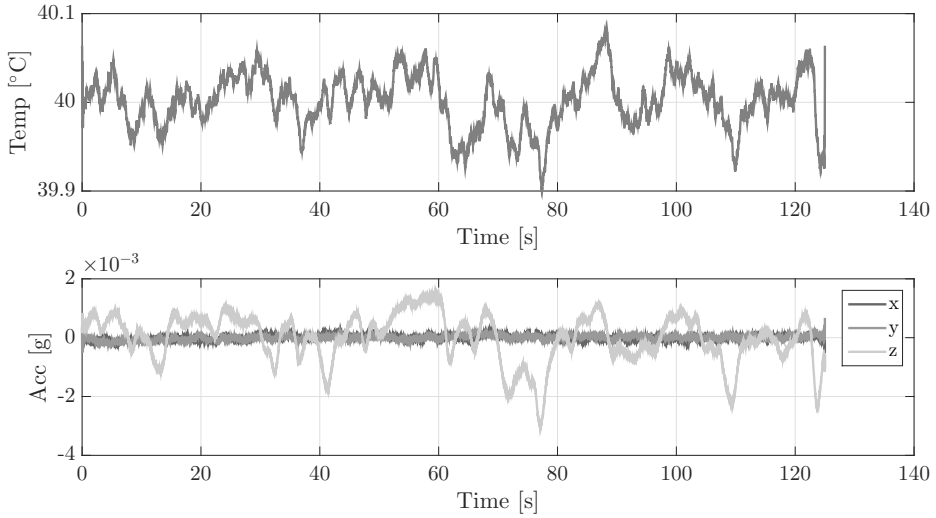


FIGURE 2.23: Temperature scaling effects for the accelerometer.

2.2.2.5 IMU Noise Modeling

There are many parameters describing the quality of different sensors allowing comparability. The maximum sample rate, the applicable measurement range as well as the measurement resolution are well known and typical quantities that allow to compare different sensor capabilities. However, the bias instability and the random walk are by far more important indicators for inertial sensors like accelerometers and gyroscopes since they rely on integration in order to obtain a valid navigation solution [97].

Bias Instability Inertial sensors have a zero output offset, a so called bias. The bias depends among others on temperature fluctuations and flicker noise in the electronics and is therefore not constant. The bias instability describes how much the bias changes over time. The lower the bias instability, the fewer the bias will change and therefore indicates a higher quality inertial sensor. The unit of the bias instability is expressed in the same domain as the bias itself, e.g. $[°/\mathrm{s}]$ for a gyroscope and $\left[m/s^2\right]$ for an accelerometer.

Random Walk The random walk describes the error that arises from the temporal integration of an inertial sensor. The Angular Random Walk (ARW) describes the angular error from the temporal integration of a gyroscope, while the Velocity Random Walk (VRW) describes the velocity error after integrating accelerometer readings. In other words, the random walk characterizes the noise of an inertial sensor in terms of its integral error. The random walk is typically expressed as noise density, hence the ARW is given in $\left[°/s/\sqrt{Hz}\right]$ and the VRW $\left[m/s^2/\sqrt{Hz}\right]$.

Since traditional low-cost MEMS sensor state estimation relies on precise estimates of system uncertainties, modeling those uncertainties is crucial. The bias instability as well as the random walk indicate how reliable the pose estimation based on pure temporal integration is. Consequently, both parameters need to be known if the integration error should be assessed by the ego-motion framework. Typically, high quality MEMS sensors provide information about bias instability and its noise density or random walk in their data sheet, however, for low-cost sensors these parameters have to be determined experimentally. The most practicable way to determine the bias instability and the random walk for a specific sensor is defined in the IEEE Standard 952-1997 [98].

The method proposed in the standard utilizes the so called Allan variance model which is originally a analysis technique designed for characterizing the frequency stability of clocks. The technique can be applied to any signal to determine the character of the underlying noise processes. The Allan variance is computed by dividing a long sequence of measurement raw data y into bins of a certain length τ. Next, the average $a_y(\tau)_i$ is computed for each bin i . The Allen variance is then calculated as:

$$AVAR(\tau) = \frac{1}{2(n-1)} \sum_{i=0}^{n-1} \left(a_y(\tau)_{i+1} - a_y(\tau)_i \right)^2 \tag{2.18}$$

where n is the total number of bins for the fixed averaging time τ. The noise characteristics can be determined by plotting different averaging times τ against the Allen deviance on a log-log scale. The Allen deviance is simple given by:

$$ADEV(\tau) = \sqrt{AVAR(\tau)} \tag{2.19}$$

Different noise characteristics can be read directly from the Allen deviance a graph as illustrated in Figure 2.24, such as the bias instability and the random walk. The relation between the Allan deviance and this noise parameters is explained in detail in [98]. White noise appears on the Allan deviance plot with a negative slope of 0.5. The random walk is obtained by fitting a straight line with the same slope to the graph and reading its value for $\tau = 1$. The bias instability is the flat part of the Allen deviance graph and can be approximated by the minimum value of the graph.

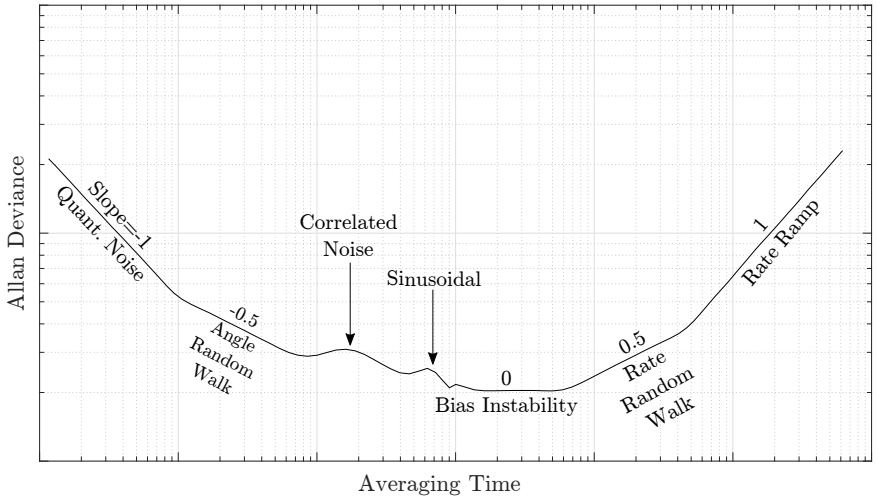


FIGURE 2.24: Allan deviance according to IEEE Standard 952-1997.

Gyroscope The noise characteristics for the gyroscope are calculated using a slightly modified approach. First, the sample data is recorded at a fixed room temperature with 952 Hz for approximately 3.5 hours resulting in over 12 million data points for each axis. Subsequently, the overlapping Allan deviance (OADEV) $\sigma_y(\tau)$ is calculated. The OADEV follows the definition of the standard ADEV but allows overlapping bins and hence gives a better, more stable result. Next, the obtained OADEV graph is approximated by fitting primitives $k\tau^\mu$ with the respective slope μ as depicted in Figure 2.24. The fitted slopes are then combined to the final deviance $\Sigma\tau^\mu$. The calculations are done in MATLAB using the *AVAR* tool [99].

The result for the LSM9DS1 gyroscope is shown in Figure 2.25 and the corresponding noise characteristics for each axis are listed in Table 2.3. Table 2.4 compares average bias instability and angular random walk from the three LSM9DS1 axes with the values given for the previously mentioned sensor grades.

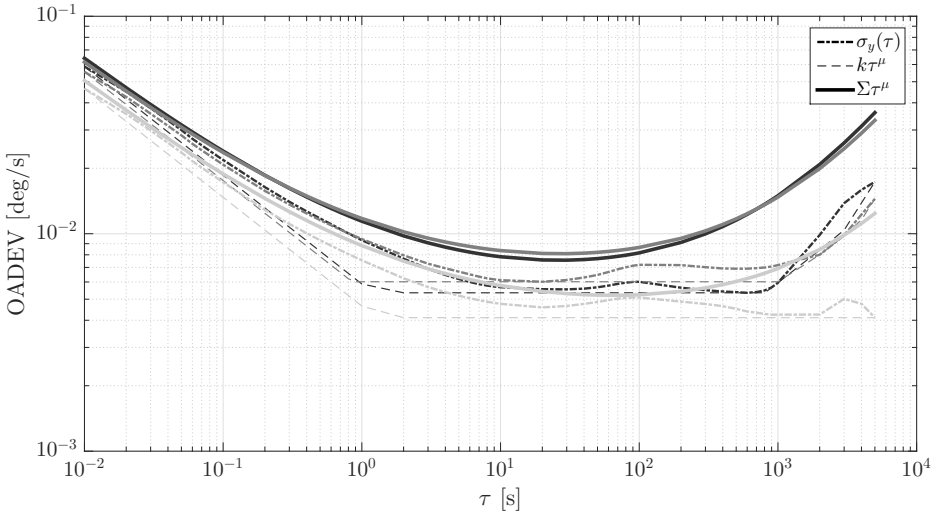


FIGURE 2.25: Overlapping Allan deviance for the LSM9DS1 gyroscope: *x*-axis (black), *y*-axis (grey), *z*-axis (light-grey).

Gyroscope Axis	x	y	z	**mean**
Bias Instability $[^\circ/\mathrm{h}]$	27	29	19	25
Angular Random Walk $\left[^\circ/\sqrt{\mathrm{h}}\right]$	0.68	0.70	0.53	0.64

TABLE 2.3: Gyroscope bias instability and angular random walk.

Device	**ADIS16495**	**XSens MTi-100**	**LSM9DS1**
Grade	military	industrial	consumer
Sample Rate [kHz]	4.25	10	0.952
Max. Range $[^\circ/\mathrm{s}]$	$\pm$2000	$\pm$1000	$\pm$2000
ADC Resolution [bit]	32	16	16
Bias Instability $[^\circ/\mathrm{h}]$	0.8	10	25
ARW $\left[^\circ/\sqrt{\mathrm{h}}\right]$	0.09	0.30	0.64

TABLE 2.4: Comparison between different gyroscope grades.

Accelerometer The accelerometer noise characteristics are determined analogously to the gyroscope. The accelerometer raw data is collected simultaneously to the gyroscope data at a common data output rate of 952 Hz.

The result for the LSM9DS1 accelerometer is shown in Figure 2.26 and the corresponding noise characteristics for each axis are listed in Table 2.5. The overlapping Allen deviation plot for the z-axis shows a correlated noise bump that results most likely from temperature dependent scaling effects despite temperature stabilization. Table 2.7 compares average bias instability and velocity random walk from the three accelerometer axes with the values given for the previously mentioned sensor grades.

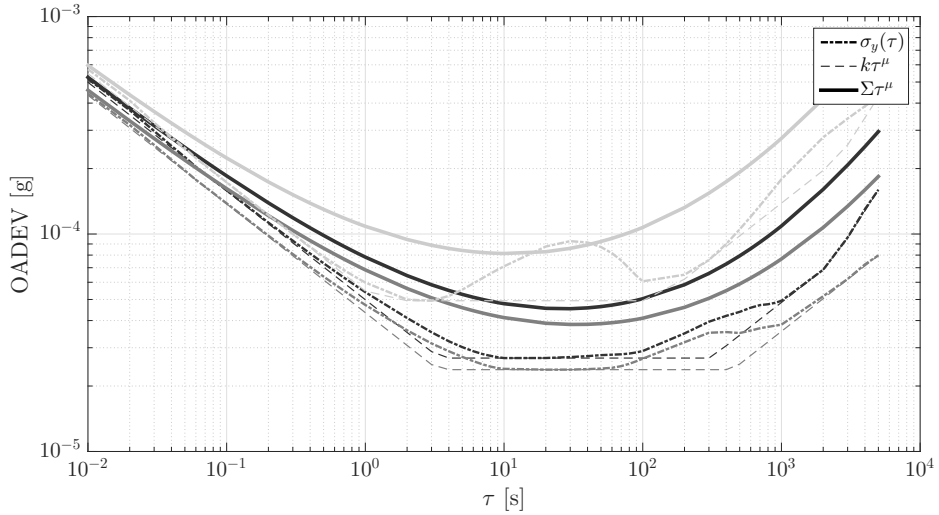


FIGURE 2.26: Overlapping Allan deviance for the LSM9DS1 accelerometer: x-axis (black), y-axis (grey), z-axis (light-grey).

Accelerometer Axis	$\boldsymbol{x}$	$\boldsymbol{y}$	$\boldsymbol{z}$	**mean**
Bias Instability $[\mu g]$	45	38	81	55
Velocity Random Walk $\left[m/s/\sqrt{h}\right]$	0.046	0.040	0.064	0.050

TABLE 2.5: Accelerometer bias instability and velocity random walk.

Device	**ADIS16495**	**XSens MTi-100**	**LSM9DS1**
Grade	military	industrial	consumer
Sample Rate [kHz]	4.25	10	0.952
Range $[g]$	± 8	± 20	± 16
Resolution [bit]	32	16	16
Bias Stability $[\mu g]$	3.2	15	55
VRW $\left[m/s/\sqrt{h}\right]$	0.008	0.032	0.050

TABLE 2.6: Comparison between different accelerometer grades.

2.2.3 Magnetometer

The magnetometer is typically used to compensate drift caused by the gyroscope integration around the UAV's nadir axis. Since magnetometers provide absolute measurements, temporal integration is not required and the sensor's random walk therefore not of interest. Its noise density, however, as well as its measurement sensitivity allow to compare and select a suitable sensor for targeted environments, e.g. an environment with a weak horizontal magnetic field component in Polar regions.

Traditional consumer grade MEMS magnetometers are based on the Hall effect, while the latest generation utilizes magnetic inductance, elevating their measurement characteristics to military grade at a simultaneously low price. Both sensor types are shown in Figure 2.27. While the traditional Hall sensors come in a single die with a very small form factor, magnetometers based on magnetic inductance rely on three separate coils for each axis and an additional logic chip. The measurement characteristics of two different sensor are compared in Table 2.7. The main benefit of the military grade RM3100 is the low sensor noise which, in theory, allows the sensor to be used at high latitudes where the horizontal component of the magnetic field is comparatively small. Unfortunately, the RM3100 is a relatively new sensor and was therefore not considered during the latest iteration of the flight controller. Currently, the magnetometer integrated into the LSM9DS1 is used as magnetic reference.

(A) RM3100 [100] (B) LSM9DS1

FIGURE 2.27: Military and consumer grade magnetometer with real size ratio.

Device	**RM3100**	**LSM9DS1**
Grade	military	consumer
Principle	Magnetic Inductance	Hall Effect
Sample Rate $[Hz]$	147	80
Sensitivity $[nT]$	13	14
Noise $[nT]$	15	750
Price	20 €	5 €

TABLE 2.7: Comparison between different magnetometer grades.

Similar to the IMU, the magnetometer is described in detail within this section. After introducing the measurement model in Section 2.2.3.1, a suitable in field calibration technique is described in Section 2.2.3.2. Subsequently, Section 2.2.3.3 describes the magnetic declination as an additional error source that needs to be compensated for if the magnetometer is used as a magnetic compass. Lastly, the temperature dependencies of the magnetometer are analyzed in Section 2.2.3.4.

2.2.3.1 Measurement Modeling

The magnetometer measurement model can be expressed in a similar way as the IMU models and reads:

$$\boldsymbol{m}_{\mathcal{B},raw} = \boldsymbol{T}_m \left(\boldsymbol{R}_{\mathcal{NB}}{}^{\top} \boldsymbol{m}_{\mathcal{N},true} + \boldsymbol{m}_{hard} \right) + \boldsymbol{m}_d + \boldsymbol{m}_w \tag{2.20}$$

where $\boldsymbol{m}_{\mathcal{B},raw}$ is the magnetometer raw measurement in the IMU body frame, $\boldsymbol{T}_m \in \mathbb{R}^{3\times3}$ is a symmetric matrix modeling the soft-iron effects that deforms the magnetic field readings, $\boldsymbol{R}_{\mathcal{NB}}{}^{\top} \boldsymbol{m}_{\mathcal{N},true}$ is the true magnetic field observed in the body frame, $\boldsymbol{m}_{hard}$ is the hard-iron offset, $\boldsymbol{m}_d$ are magnetic disturbances and $\boldsymbol{m}_w$ is Gaussian measurement noise. The hard-iron effect results from permanently magnetized ferromagnetic components in vicinity of the magnetometer and includes additional zero field offsets. In contrast, soft-iron effects are caused by a magnetic field induced by the geomagnetic field onto non-magnetized ferromagnetic components. Different gain factors along the magnetometer axes, as well as non-orthogonalities between the axes are also modeled by the soft-iron effect [101].

2.2.3.2 Magnetic Calibration

For the magnetic calibration, the hard- and soft-iron effects need to be eliminated. Similar to the IMU calibration, the calibration method for the magnetometer should be applicable in field and hence independent of external tools. The magnetometer calibration method used in this work is based on [101]. Figure 2.28 shows the raw data ellipsoid as well as the corrected data for a single magnetometer calibration.

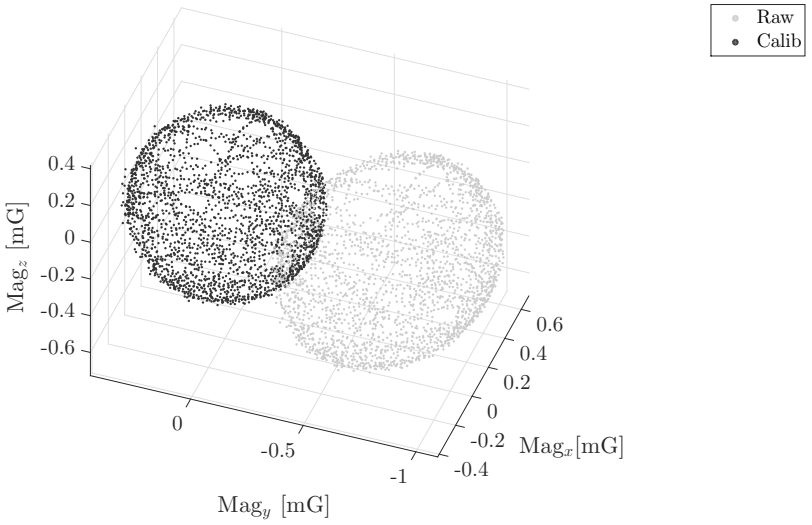


FIGURE 2.28: Raw and calibrated magnetometer readings.

The calibration data is collected by rotating the UAV platform with mounted magnetometer randomly around all principle axes. In order to guarantee a uniform distribution of the collected sample points, an iterative thinning algorithm is applied according to [42]. The thinning algorithm calculates the distance between adjacent points for each sample and eliminates it from the sample set if the calculated distance is smaller than the minimum distance between two points for a regular polyhedron with triangular faces consisting of the same number of samples. The remaining, uniformly distributed samples are approximated with an ellipsoid using the method

introduced in [102]. The ellipsoid is described in terms of its center $\boldsymbol{c}_e$, the three radii $\boldsymbol{r}_e$ and the respective radii direction vectors as a matrix $\boldsymbol{E}_e = \begin{bmatrix} \boldsymbol{e}_x & \boldsymbol{e}_y & \boldsymbol{e}_z \end{bmatrix} \in \mathbb{R}^{3\times3}$. The fitted ellipsoid can be used to simply calculate the calibration parameters as:

$$\boldsymbol{m}_{hard} = \boldsymbol{c}_e \tag{2.21}$$

$$\boldsymbol{T}_m = \boldsymbol{E}_e \left[\min\left(\boldsymbol{r}_e\right) \operatorname{diag}\left(\boldsymbol{r}_e\right)^{-1} \right] \boldsymbol{E}_e{}^{\top} \tag{2.22}$$

2.2.3.3 Magnetic Declination

If the magnetometer is used as an angular reference, e.g. as a magnetic compass, the magnetic declination is added as an additional error term and needs to be compensated for. The magnetic declination $\Delta\psi_{Decl}$ is the misdirection of every magnetic compass and describes the angular difference between the magnetic and geographic pole for a certain location. If the magnetometer location is known, the magnetic declination can be calculated using the World Magnetic Model [103]. In Germany, the declination is relatively small and approximately between $2\,^\circ$ and $4\,^\circ$. This changes drastically for high latitudes as can be seen in Figure 2.29. The north heading pseudo measurement ψ_N based on a magnetometer observation can be described as:

$$\psi_N = \operatorname{atan2}\left(\boldsymbol{m}_{\mathcal{B},raw_y}, \boldsymbol{m}_{\mathcal{B},raw_x}\right) - \Delta\psi_{Decl} \tag{2.23}$$

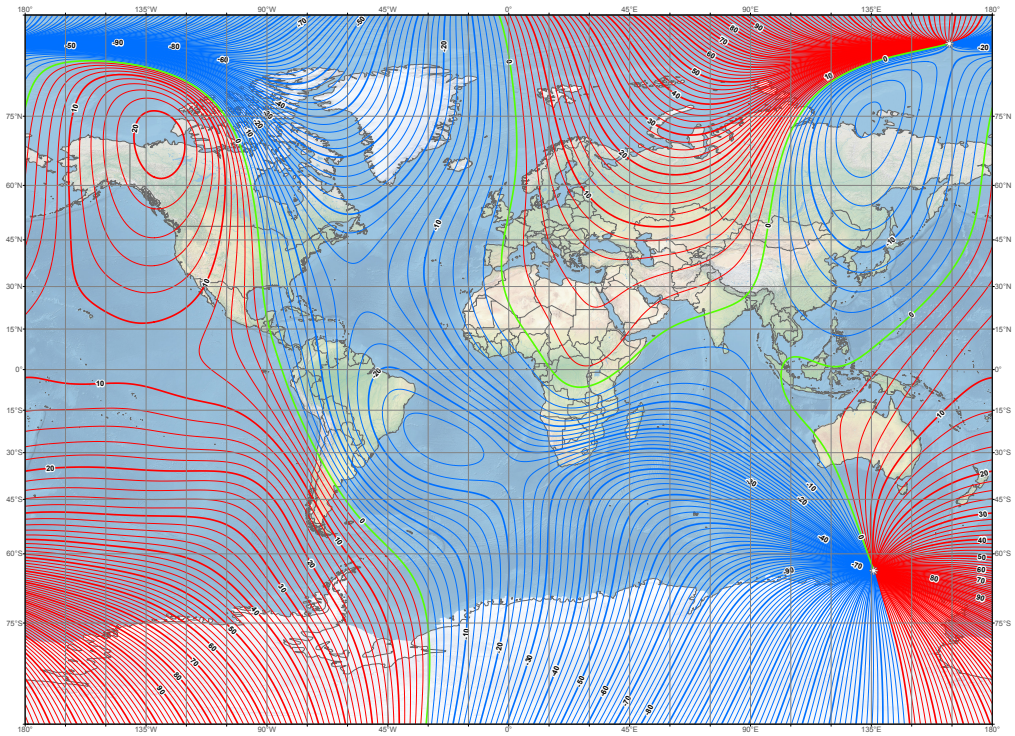


FIGURE 2.29: Magnetic declination map for 2020 [103]. The green line indicates a declination of zero degrees, while a red and a blue line indicate a positive and a negative declination, respectively.

2.2.3.4 Temperature Dependencies

Similar to the IMU, MEMS magnetometers using the Hall effect are silicon based, too. Although temperature dependencies are present, the topic of thermal magnetometer dependencies and their compensation is often neglected in literature. The temperature dependencies of the magnetometer are analyzed using the same thermal chamber and setup as described in Section 2.2.2.4. The results are shown in Figure 2.30. The temperature decencies can be eliminated by applying the previously described calibration procedure at different temperatures levels. To obtain a calibration model for temperatures in between the different calibration levels, a linear approximation can be used. However, since the magnetometer is integrated into the LSM9DS1, its temperature is stabilized, too and hence no additional temperature compensation is required.

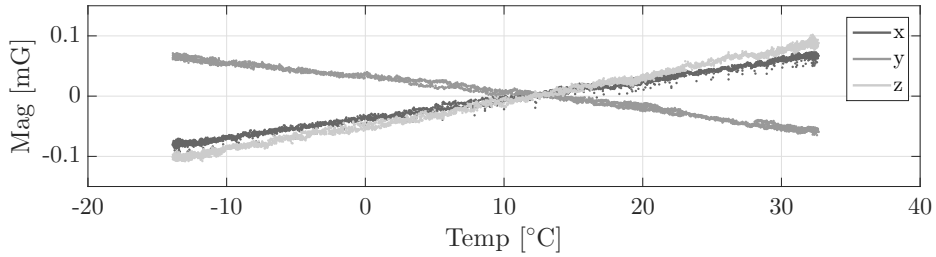


FIGURE 2.30: Temperature dependencies of the LSM9DS1 magnetometer.

2.2.4 Barometric Pressure Sensor

The Bosch BMP388 [104] digital barometric pressure sensor is used as an altitude reference and connected to the real-time Cortex-M4 over I2C. The design of the sensor was specifically targeted towards drone applications. Temperature dependencies of the barometric pressure measurements are compensated using the built-in thermometer. The relation between the pressure p_h and the altitude h itself is given by the barometric height formula:

$$p_h = p_r \left(1 - \frac{\Gamma_0 \cdot h}{T_r}\right)^{\frac{gM}{R\Gamma_0}} \tag{2.24}$$

where p_r and T_r are the pressure and the temperature at the reference altitude, respectively, $\Gamma_0 = 0.0065\,\mathrm{K/m}$ is the temperature lapse rate within the Troposphere, $g = 9.80665\,\mathrm{m/s^2}$ is the Earth's gravity constant, $M = 0.0289644\,\mathrm{kg/mol}$ is the molar mass of the Earth's air and $R = 8.3144598\,\mathrm{J/(mol \cdot K)}$ is the universal gas constant. Solving Equation (2.24) for the altitude h, the altitude can be calculated with respect to the pressure and the temperature at the reference height. The maximum sample rate is 200 Hz with a measurement noise of approximately 1.2 Pa which equals a noise level of 10 cm. The barometer measurements y_{baro} are simply modeled as:

$$y_{baro} = h_{baro,true} + h_w \tag{2.25}$$

where $h_{baro,true}$ is the true altitude with respect to the utilized reference and h_w is white Gaussian noise with a standard deviation of $\sigma_{baro} = 10\,\mathrm{cm}$.

2.2.5 Global Navigation Satellite System Receiver

Two precision timing single frequency uBlox NEO-M8T GNSS receivers are integrated into the flight controller. They are connected to the application processor using a dedicated UART interface each. While for the positioning of a UAV a single GNSS receiver is sufficient, the use of two receivers allows to implement a GNSS heading reference, a so-called GNSS compass. The GNSS measurements models can be simplified to:

$$\boldsymbol{p} = \boldsymbol{p}_{true} + \boldsymbol{p}_w \tag{2.26}$$
$$\boldsymbol{v} = \boldsymbol{v}_{true} + \boldsymbol{v}_w \tag{2.27}$$
$$\theta = \theta_{true} + \theta_w \tag{2.28}$$

where $\boldsymbol{p}$ and $\boldsymbol{v}$ are the observed position and velocity, respectively, and θ is the observed heading computed by the GNSS compass. The observations' real values are indicated with the subscript *true*, while additive noise terms are labeled with the subscript w. The noise quantity of each observation depends on the receiver grade, the utilized number of GNSS frequencies as well as the applied navigation technique. For reference, three different GNSS receiver grades from uBlox are shown in Figure 2.31 and their characteristics compared in Table 2.8.

(A) ZED-F9P [105] (B) NEO-M8T [106] (C) SAM-M8Q [107]

FIGURE 2.31: uBlox GNSS modules: High precision dual-frequency module ZED-F9P, timing module NEO-M8T and small scale GNSS module with integrated antenna SAM-M8Q.

Device	**ZED-F9P**	**NEO-M8T**	**SAM-M8Q**
Frequencies	L1 & L2	L1	L1
Grade	professional	professional	consumer
Price	176 €	80 €	22 €
Raw Data	yes	yes	no
Antenna	ANN-MB-00 [108]	TW2990 [109]	integrated
Footprint [mm²]	17.0 × 22.0	12.2 × 16.0	15.5 × 15.5

TABLE 2.8: GNSS module comparison.

The uBlox SAM-M8Q is a consumer grade GNSS receiver with integrated antenna. It provides a navigation solution with standard GNSS accuracy to the user at 10 Hz. In contrast to the consumer grade receiver, the two professional grade receivers provide a complete navigation solution with standard GNSS precision but additionally raw observations in the shape of code- and carrier-observations as well as Doppler shift measurements. The NEO-M8T outputs its data at 10 Hz, while the

ZED-F9P provides new measurements and navigation solutions at 20 Hz. Both receivers require external antennas. Their raw observations can be used to compute precise navigation solutions using differential GNSS techniques such as RTK (Section 4.5.3) and Precise Point Positioning (PPP) (Section 4.5.4). Being one of the key components within this work, the Global Navigation Satellite System and its observables as well as standard and advanced navigation techniques are described in detail in Chapter 4. Using the NEO-M8T, an external processor is required to compute advanced navigation techniques. In contrast, some RTK functionalities are already integrated into the ZED-F9P. Despite of being the inferior receiver, the NEO-M8T is used within this work for the following reasons:

1. The NEO-M8T is a trade-off between an affordable and a high-end professional GNSS receiver, combining a reasonable price and high accuracy raw data measurements
2. The NEO-M8T was used at an early stage of this work and therefore the acquired knowledge base was rather extensive by the time the ZED-F9P was publicly released in 2019
3. The external implementation of the advanced navigation algorithms allows to integrate complementary sensors for a more robust navigation solution

Although a single frequency receiver is used, the methods described within this work are without any exception valid for multi frequency GNSS receivers. Hence, the integration of the superior ZED-F9P is something that should be addressed in the next hardware iteration of the flight controller.

2.2.6 Ultra-wideband Transceiver

Traditionally designed for short-range low-energy high-bandwidth communication, Ultra-Wide Band (UWB) radio technology emerged as a very popular local positioning system solution in warehouse and industrial applications since well over a decade. With the availability of cost efficient and miniaturized UWB transceivers, the interest in research at public institutions and universities has also increased. Hereby, low-cost and reliable indoor navigation using small UAVs is one of the most researched topics [110–115]. In this section, the fundamentals of UWB technology are briefly recapped before describing Double-Sided Two-Way Ranging (DS-TWR), a Time of Arrival (TOA)-based UWB ranging technique, and introducing a UWB distance measurement model. The DecaWave DWM1000 module [116], as shown in Figure 2.32, is connected to the real-time Cortex-M4 using a dedicated SPI interface.

FIGURE 2.32: DecaWave UWB module DWM1000 [116].

2.2.6.1 UWB Radio Signal Characteristics

The idea of UWB radio communication is derived directly from the Shannon-Hartley theorem, describing the channel capacity C in [bits/s] of a radio signal with additive white Gaussian noise N as:

$$C = B \cdot \log\left(1 + \frac{S}{N}\right) \tag{2.29}$$

where B is the signal bandwidth in [Hz] and S is the signal transmission power. The ratio of signal transmission power to received signal noise power is commonly referred to as signal-to-noise ratio (SNR) in [dB]. Analyzing Equation (2.29), it is obvious that low-power radio communication with a large channel capacity can be achieved by increasing the signal bandwidth. In contrast to traditional radio frequency technology with a single frequency carrier wave, UWB radio technology relies on a wide frequency spectrum for transmission in order to implement a short-range low-energy high-bandwidth communication link. Specifically, a UWB radio is defined as an antenna transmission where the emitted signal bandwidth is least 500 MHz or the fractional bandwidth is greater than 0.2 of the arithmetic center frequency [117]. A traditional, frequency modulated radio signal is compared to a UWB signal with bi-phase modulation in Figure 2.33 below. Additionally, different frequency bands of standardized narrow-band and UWB signals as well as their respective transmission power are depicted.

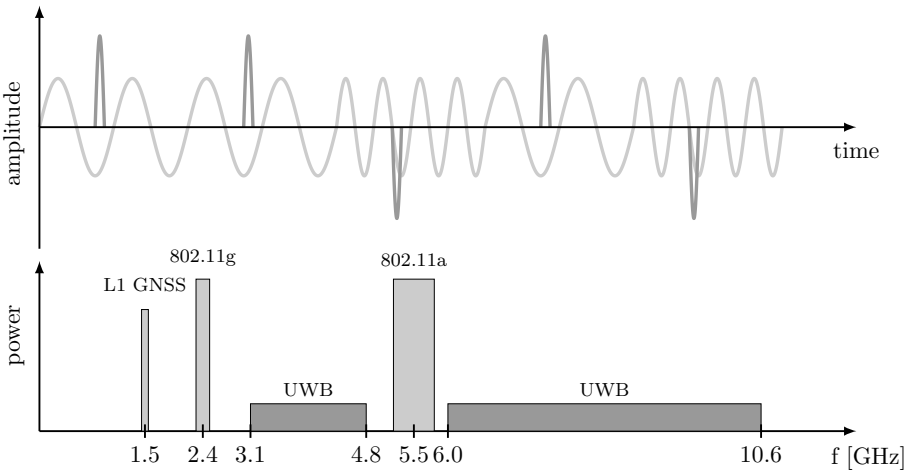


FIGURE 2.33: Comparison between UWB (dark) and narrow-band radio frequency signals (bright).

The UWB radio was originally referred to as pulse radio, since the wide frequency spectrum of a UWB radio has the temporal representation of pulse. The pulse shape of UWB signals can be used for time stamping message transmissions and hence the time of message reception with a high precision. If the time of transmission and reception are known with respect to a common time base, the distance between the transmitter and the receiver can be calculated, if the signal propagation time of the respective medium is known.

The DWM1000 module provides four different center frequencies (3.5 GHz, 4.0 GHz, 4.5 GHz and 6.5 GHz) with different spectral pulse widths ranging from 499.2 MHz to up to 1331.2 MHz [116]. Throughout this work, channel number seven with a center frequency of 6489.6 MHz and a nominal bandwidth of 1081.6 MHz is used.

2.2.6.2 UWB Radio Ranging

Depending on the specific application and the UWB anchor network constellation, different ranging techniques are used, such as Time Difference of Arrival (TDOA) and Time of Arrival (TOA). While TDOA approaches allow for a high scalability and high update rates, precise clock synchronization between UWB network anchors is required [118]. In contrast to this, TOA approaches such as Double-Sided Two-Way Ranging (DS-TWR) eliminate the need for precise clock synchronization at the cost of additional ranging messages, resulting in lower update rates. In this work, the distances between two UWB modules are obtained using Double-Sided Two-Way Ranging. Figure 2.34 illustrates a complete ranging sequence for two transceivers.

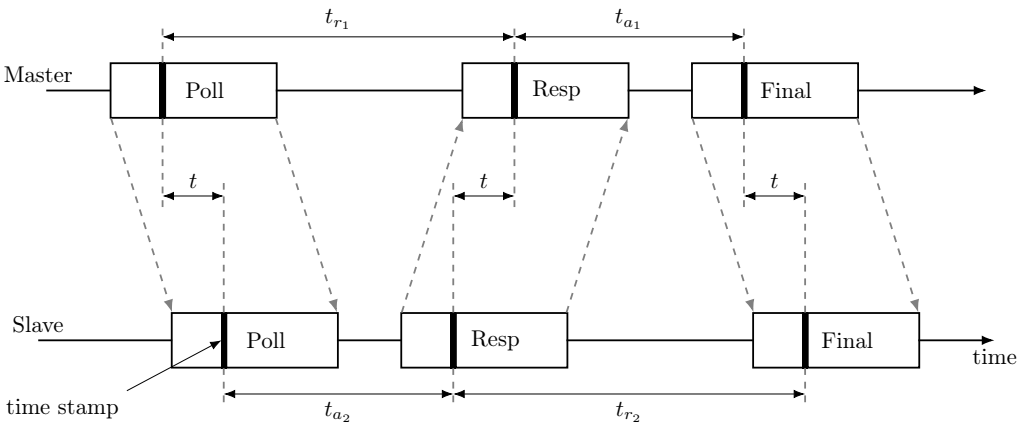


FIGURE 2.34: DS-TWR scheme using two UWB transceivers based on [119].

Each message contains a pulse signal that is used to mark the exact transmission time of the message. The DS-TWR is initiated by the master transmitting a poll message to the slave. The slave detects the time of the pulse signal using its own clock and responses with a second message after a fixed time delay t_{a_2}. Once the master detects the arrival time of the pulse of the second message, the first round trip time t_{r_1} can be calculated. After a fixed delay t_{a_1}, a final message is sent from the master. Besides the timing pulse, this final message contains the calculated round trip time t_{r_1} and the delay time t_{a_1}. Once the final message is received at the slave, the second round trip time t_{r_2} can be calculated. The propagation time t can be calculated according to [119] as:

$$t = \frac{t_{r_1} \cdot t_{r_2} - t_{a_1} \cdot t_{a_2}}{t_{r_1} + t_{r_2} + t_{a_1} + t_{a_2}} \tag{2.30}$$

Using the time of flight t and the speed of light in air c_{air}, the range d_{UWB} between the transceivers can be calculated as:

$$d_{UWB} = t \cdot c_{air} \tag{2.31}$$

Figure 6.18 shows a histogram for 5000 range errors using DS-TWR. Based on the histogram, the raw distance measurements $d_{UWB,raw}$ are assumed to have additive noise with zero-mean. The standard deviation is found to be $\sigma_{UWB} \approx 25.7$mm. In addition to the additive noise, a measurement offset can be observed. This bias is caused by small variations in the manufacturing process of the antenna and can

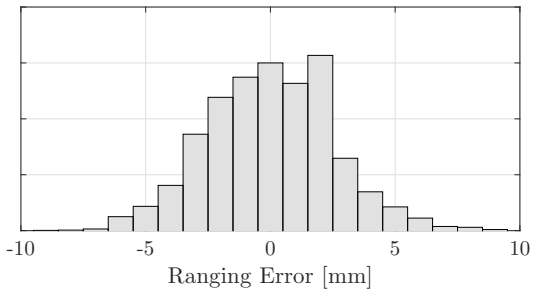


FIGURE 2.35: Histogram for 5000 DS-TWR distance measurements.

be determined using the calibration method described in [120]. Hence, the full measurement model for UWB modules using DS-TWR is given as:

$$d_{UWB,raw} = d_{true} + d_b + d_w \tag{2.32}$$

where d_{true} is the true distance, d_b is the bias caused by the antenna delay and d_w is Gaussian noise with $d_w \sim \mathcal{N}\left\{0, \sigma^2_{UWB}\right\}$.

Figure 2.36 illustrates the ranging scheme in case of multiple slaves. All slaves are idle until a polling message is received. After reception, each slaves prepares to send a response message after a respective delay t_{Xa_2} for each slave X. Here, the delay is a distinct multiple of t_{Aa_2}, e.g. $t_{Ba_2} = 2 \cdot t_{Aa_2}$. The master distributes the measured round trip and local response times in a single final message. The propagation times t_A and t_B as well as the respective distances can be calculated according to the Equations (2.30) and (2.31).

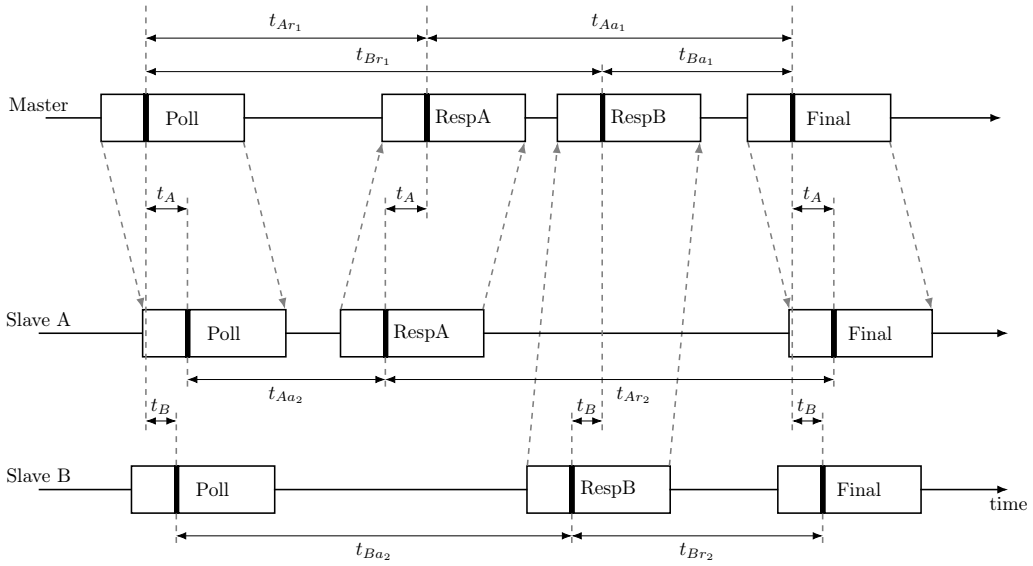


FIGURE 2.36: DS-TWR scheme using a single master and several slave modules based on [119].

2.2.7 Remote Control Interfaces

Communication Interface The Udoo Neo board is equipped with Texas Instruments WL1831 module [121] to provide WiFi and Bluetooth connectivity. An external 2.4 GHz antenna can be used instead of the on-board micro strip antenna to enhance performance and reliability. WiFi is used as primary interface for ground station communication.

Remote Control Interface The flight controller provides a serial interface for commercial remote control receivers supporting the Spektrum 1024/2048 protocol [122] and the SBUS standard [123]. The Spektrum protocol is exclusively used by Spektrum remote controllers. The SBUS standard is used by a wide range of remote control vendors such as Futaba, Frsky and DJI. The two systems that are currently used with the developed flight controller are shown in Figure 2.37. The Spektrum DX8 transmits control inputs using eight channels on a wideband direct-sequence modulated 2.4 GHz signal [124].The DJI Air unit uses the 5.7 GHz band for transmitting remote control signals, a real-time video feed and on-screen display telemetry [125].

(A) Spektrum DX8 [124] (B) DJI Air Unit [125]

FIGURE 2.37: Supported remote controls.

Additional Interfaces Other interfaces on the flight controller expansion board are exposed using Molex connectors, giving access to an additional 3.3 V SPI interface, a 3.3 V and a 5 V I2C interface, a full UART with hardware flow-control as well as to two ADC channels. Depending on the user requirements and the connected periphery, each interface can be used with either the real-time or the application core. Serial debugging ports for both cores are also exposed.

2.2.8 Optional Components

As previously mentioned, a number of additional interfaces is available for connecting optional sensors or communication devices that are not mandatory for the flight controller functionality. In GNSS denied environments, additional positioning sensors are required. Different positioning sensors and systems that can be interfaced with the flight controller are described in Section 2.2.8.1. For an autonomous landing in an unknown environment, additional proximity sensors that are able to detect the relative height of the UAV might be required. Two proximity sensor examples that were used with this flight controller are introduced in Section 2.2.8.2. For long range missions or missions in remote locations, additional communication interfaces are required. The optional communication interfaces are described in Section 2.2.8.3.

2.2.8.1 Positioning Sensors and Systems

Positioning sensors and systems can be grouped depending on their frame of reference into body frame fixed and external frame referenced components. They are typically based on optical measurements, however, radio-based positioning systems, e.g. using a network of UWB anchors and different radio ranging techniques, increase in popularity.

Body Frame Fixed Sensors Both body fixed sensors introduced here provide egomotion estimations using image processing techniques. They are both depicted in Figure 2.38. The sensors are suitable for unknown environments, e.g. autonomous flight trough an unknown building, since they are independent of external infrastructure. However, as both sensors rely on optical measurements, sufficiently good lighting is a basic requirement for the use of either of the sensors.

(A) ADNS3080 (B) Intel T265 [126]

FIGURE 2.38: Body fixed positioning sensors.

The ADNS3080 is an optical mouse sensor that measures translational 2D pixel displacement at 2 kHz in the body frame $\boldsymbol{\Delta p}_{\mathcal{B}}$ using optical flow [127]. Adjusting its optical configuration, the sensor can be used on-board a UAV. A detailed sensor description and system evaluation is given in [128]. The sensor is connected via SPI and is typically mounted on the UAV platform facing downwards with its principle image axes aligned to the body frame axes. For a known height and a fixed optical setup, the pixel displacement per frame can be converted into a velocity measurement using a height depended scaling factor $\gamma(z)$. The current UAV attitude represented by the rotation matrix $\boldsymbol{R}_{\mathcal{NB}}$ needs to be corrected for, too. The resulting 2D velocity measurements $\boldsymbol{v}_{\mathcal{B},xy}(z)$ can be modeled with additive white Gaussian noise as $\boldsymbol{\Delta p}_{w,ADNS}$:

$$\boldsymbol{v}_{\mathcal{B},xy}(z) = \boldsymbol{R}_{\mathcal{NB}}^{\top}\gamma(z)\left(\boldsymbol{\Delta p}_{\mathcal{B},true} + \boldsymbol{\Delta p}_{w,ADNS}\right) \tag{2.33}$$

The Intel RealSense T265 is a visual SLAM sensor using a stereo camera setup and can be interfaced using the USB port of the flight controller. In contrast to the optical flow sensor, the T265 provides a full pose estimation, namely its attitude and a 3D position, rather than just information about the sensor displacement [126]. A USB 3 port is required to transfer the pose estimation and the raw images, simultaneously. The maximum sample rate for pose estimation is 200 Hz. The pose estimation is relative to a starting point and can be modeled using additive white Gaussian noise $\boldsymbol{p}_{w,T265}$ and $\boldsymbol{q}_{w,T265}$, too:

$$\boldsymbol{p}_{\mathcal{B}} = \boldsymbol{p}_{\mathcal{B},true} + \boldsymbol{p}_{w,T265} \tag{2.34}$$

$$\boldsymbol{q}_{\mathcal{B}} = \boldsymbol{q}_{\mathcal{B},true} + \boldsymbol{q}_{w,T265} \tag{2.35}$$

External Frame Sensors Different external positioning systems can be used with the developed flight controller. This section describes two systems that were already used together with the flight controller. However, the flight controller is not limited to those system and other external positioning systems with similar measurement models can be used, too.

The first system is the OptiTrack Flex 3 commercial motion capture system [129]. It is installed in the flight hall of the Aerospace Computer Science department of the University of Wuerzburg and consists of 16 external infrared light emitting cameras. Two cameras and a camera hub are shown in Figure 2.39.

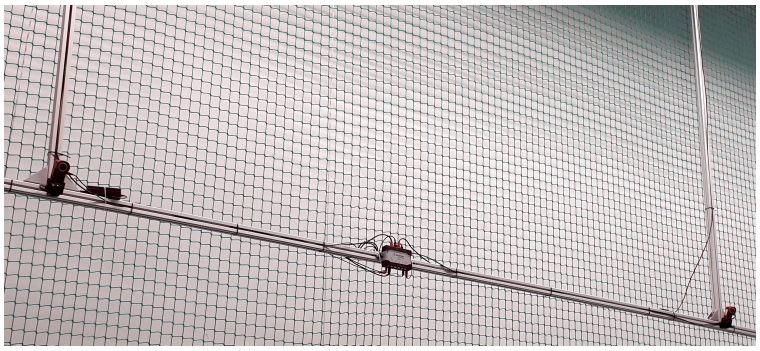


FIGURE 2.39: A part of the OptiTrack Flex 3 motion capture system.

Figure 2.40 shows the motion capture system pipeline. The OptiTrack motion capture software runs on a Windows PC and streams the pose of each detected rigid body i over a local network. The stream is captured by the UAV ground station and the position of the respective UAV is transmitted wirelessly to the flight controller.

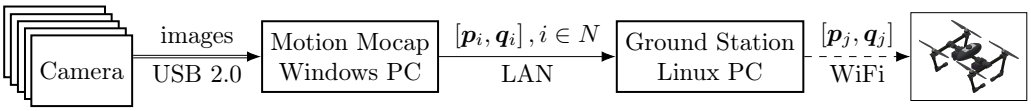


FIGURE 2.40: Motion capture system pipeline.

The setup covers an area of about 6 m×8 m up to a height of approximately 6 m. Infrared light reflecting markers are mounted on the UAV. The marker positions are computed in the 2D image of each camera and used to calculate the 3D position in overlapping images using triangulation. If several markers are rigidly mounted on a platform, its attitude can be estimated, too. The full pose estimation data in the navigation frame $\mathcal{N}$ is provided at up to 100 Hz. It provides a position accuracy of 1 cm and an angular error of less than 1°. Modeling the observation errors as additive white Gaussian noise, the measurement equations read:

$$\boldsymbol{p}_{\mathcal{N}} = \boldsymbol{p}_{\mathcal{N},true} + \boldsymbol{p}_{w,OptiTrack} \tag{2.36}$$

$$\boldsymbol{q}_{\mathcal{N}} = \boldsymbol{q}_{\mathcal{N},true} + \boldsymbol{q}_{w,OptiTrack} \tag{2.37}$$

The optical tracking system is mostly used for system verification and the development of new algorithms, where a precise and absolute reference is required. In particular the ego-motion estimation framework of this work (Section 3.2) was verified and evaluated using the motion capture system.

Another possibility is a Local Positioning System (LPS) based on the UWB ranging scheme introduced in Section 2.2.6. The LPS approach implemented within this

work is fully described in [130] and briefly summarized here. A network of UWB anchors can be randomly deployed under the constraint that at least one anchor is not on the same plane as the others. Based on the ranging information between the fixed anchors a local coordinate frame is constructed. Distance measurements between each anchor and a UWB transceiver on the UAV are collected sequentially. Using the known anchor positions as well as the obtained ranging information, the UAV position can be estimated using Multi-Dimensional Scaling (MDS). The position estimation is performed in real time on the UAV. Figure 2.41 shows a LPS with four fixed anchors and a mobile UWB tag mounted on the UAV.

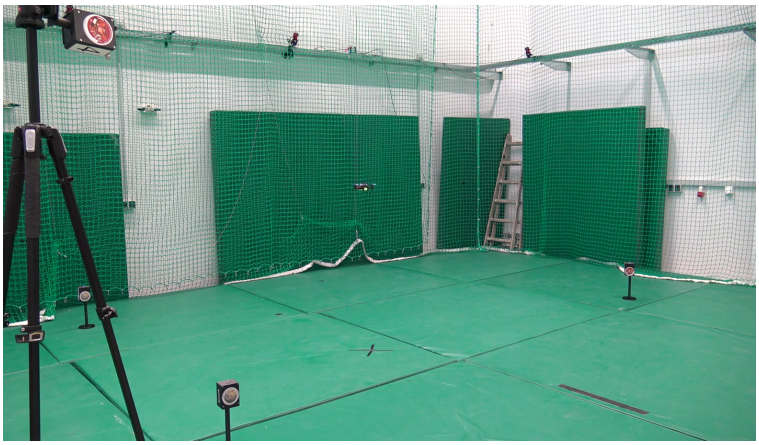


FIGURE 2.41: LPS based on UWB anchors and DS-TWR.

For reference, both positioning systems are compared in Figure 2.42. Clearly, the motion capture system has a better accuracy than the UWB based positioning. However, due to the mobile use and simple system setup as well as the comparatively low system price, the UWB-based positioning system provides a valuable alternative. The 3D position accuracy of the UWB localization system is approximately 10 cm. The position observations can be be modeled with Equation (2.36) and a correspondingly adapted noise term $\boldsymbol{p}_{w,UWB}$.

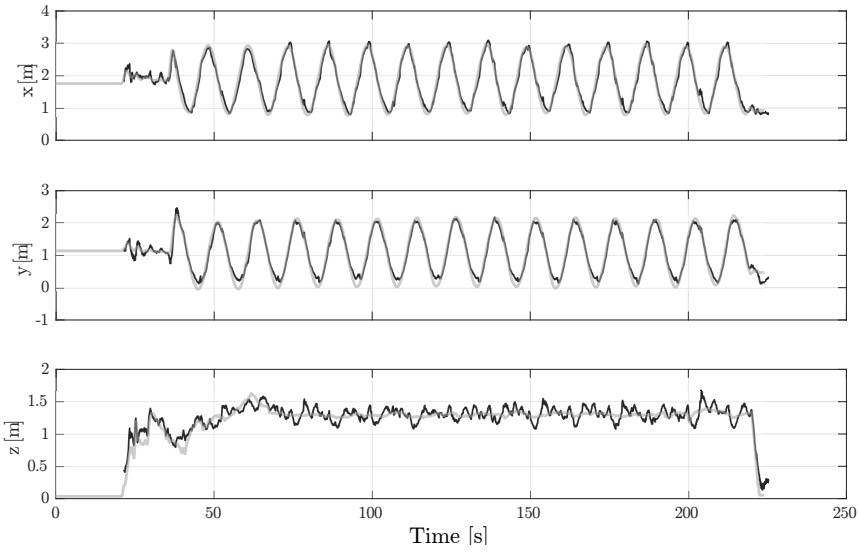


FIGURE 2.42: UWB (black) vs. OptiTrack (grey) positioning system.

2.2.8.2 Proximity Sensors

In general, proximity sensors are able to detect nearby objects or obstacles without physical contact by emitting a signal and analyzing the signal return. Depending on the type of the emitted signal, acoustic, electro-optic and radar sensors are distinguished. Each signal has different characteristics regarding its propagation time and direction as well as material depended surface reflection. Acoustic signals travel at the speed of sound in the respective medium and are reflected by rigid materials and water. Electro-optic signals travel at the speed of light in the respective medium and are not necessarily reflected by glass or materials with a shiny surface, since the amount of reflection depends on signal's incident angle. Radar signals have the same propagation speed as light and are strongly reflected by metal, however, reflections of other materials can be detected, too. The transmission cones of acoustic and radar signals are wide compared to electro-optical signals. A general proximity sensor model for the distance $d_{\mathcal{B}}$ with additive noise d_w is given by:

$$d_{\mathcal{B}} = \begin{cases} d_{\mathcal{B},true} + d_w & \text{for a reflective surface} \\ \text{nan} & \text{for a non-reflective surface} \end{cases} \tag{2.38}$$

As of today, radar sensors require extensive electronics and software for signal processing and interpretation and are therefore targeted towards very specific tasks. In contrast, acoustic and electro-optical sensors are comparatively easy to use and integrate. Figure 2.43 shows an electro-optic and an acoustic sensor that can be interfaced over I2C with FARN for autonomous landing. The electro-optic Lidar-Lite has a range 40 m and an accuracy of 1 cm at up to 500 Hz [131]. The acoustic SRF-02 uses ultra-sound and has a range 5 m and an accuracy of 2 cm [132].

(A) Lidar-Lite v3 [131] (B) SRF02 [132]

FIGURE 2.43: Different proximity sensors.

Figure 2.44 shows the readings of both proximity sensors during a flight over a mixture of ice and water during the Arctic expedition PS93.2. The Lidar-Lite is configured to detect weak reflections, too, allowing to detect the water surface.

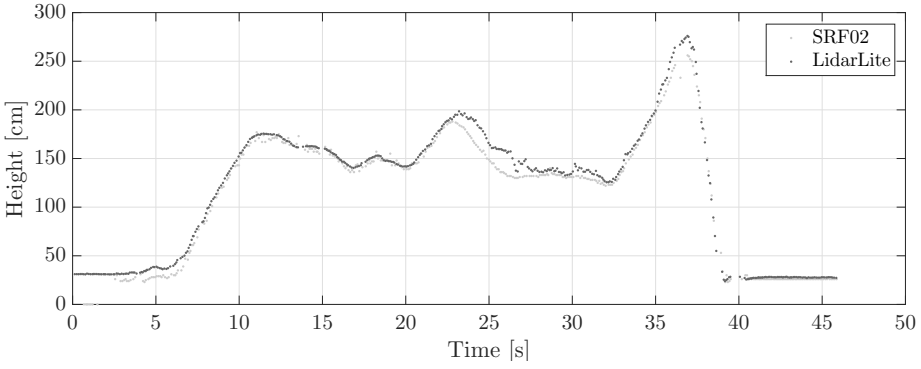


FIGURE 2.44: Proximity sensors during a flight over ice and water.

2.2.8.3 Wireless Interfaces

As previously stated, the main communication link is provided by the WL1831 WiFi module. However, depending on the specific mission, it might be required to extend the flight controller's communication capabilities. Due to its range limitations, WiFi is not suitable for long range communication. If appropriate infrastructure is available, Long Term Evolution/4G (LTE/4G) mobile technology can be a profitable alternative. In remote areas with little or no infrastructure at all, such as Arctic environments, independent long range communication techniques are required. While LTE/4G provides excellent data rates and an acceptable latency, traditional point-to-point long range communication suffers low data rates. Figure 2.45 shows two communication modules that were already used together with the flight controller.

(A) DWM-222 [133] (B) AMB8636.

FIGURE 2.45: Wireless communication module.

The DLink DWM-222 is a LTE/4G module that can be interfaced over USB. Up- and down-link data rates are comparable to WiFi with a slightly higher latency which in turn depends on the respective network constellation. Additional networking infrastructure or services like a VPN-tunnel, a server or dynamic DNS might be required, too.

The Amber Wireless AMB8636 is simply connected over UART. The radio signal is modulated using Gaussian Frequency Shift Keying (GFSK) at 868 MHz [134]. The half duplex communication link has a maximum data of 50 kbit/s bridging a distance of almost 14 km. Its latency is range dependent but typically limited to a few milliseconds. Due to the low data rate, the communication load needs to be reduced to essential telemetry and only necessary telecommands.

The communication properties of all three devices are summarized and compared in Table 2.9.

Device	**WL1831**	**DWM-222**	**AMB8636**
RF Technology	WiFi	LTE	GFSK 868 MHz
Duplex	full	full	half
Stand-alone	yes	no	yes
Latency [ms]	≤ 100	≤ 250	n.a.
Up-link [Mbit/s]	80	150	0.05
Down-link [Mbit/s]	80	50	
Range [km]	≤ 0.5	n.a.	≤ 14

TABLE 2.9: Comparison between different communication modules.

2.3 Payload

The UAV platform and the flight controller described in this Chapter were successfully used in several real-world applications. This section outlines the UAV configuration as well as the added payload for the two projects ROBEX and MIDRAS.

ROBEX Within the ROBEX project, the dual antenna UAV platform was deployed allowing to use a GNSS compass. The UAV was equipped with both, the acoustic and the electro-optic proximity sensor for autonomous landing on ice as well as the long range 868 MHz communication module. As pilot supporting payload, the UAV was equipped with two cameras: A FLIR Lepton thermal camera [135] for ice floe detection and a RGB tilt FPV-camera for remote operation. As a scientific payload, a SeaBird Photo-synthetically Active Radiation (PAR) sensor was interfaced with the flight controller to record reference data for the AUV. Figure 2.46 shows the complete UAV setup as well as a thermal image recorded at an ice edge.

(A) Sensor suite. (B) Ice edge detection

FIGURE 2.46: ROBEX payload.

MIDRAS In contrast, for the MIDRAS project the single antenna configuration was deployed. Only the electro-optic proximity sensor and the WiFi communication link were used. Within MIDRAS, two cooperative UAVs carried a net together. As additional payload, each of the two UAVs was equipped with a load cell to measure net impacts as well as a net detachment mechanism. The full configuration is shown in Figure 2.47.

FIGURE 2.47: MIDRAS payload.

Chapter 3

Ego-motion Estimation and Low-Level Control

This chapter describes the mathematical concepts and algorithms that are deployed within FARN for ego-motion estimation and low-level control of a UAV.

Throughout literature, various attitude representations are used for UAVs, however, the mathematical background of the respective operators is not always clearly defined and a common source for errors. Therefore, within this chapter the mathematical background as well as different conventions regarding attitude representation are introduced in Section 3.1. Based on those mathematical concepts, the core of the developed flight controller, namely the ego-motion estimation framework which builds upon an Error-State Kalman Filter (ESKF) and the implemented control architecture are derived in Section 3.2 and 3.3, respectively.

3.1 Attitude Representation

In the area of autonomous robotics, knowing the robot's state precisely, especially its pose and twist, at any given time is fundamentally important for autonomous operations. Particularly for flying robots and especially in case of underactuated systems like quadrotor UAVs, knowing the robot's attitude is essential since by changing the attitude of a quadrotor, the remaining three translational degrees of freedom are affected. Generally, the attitude of a rigid body in 3D is represented by a rotation from a reference frame to a frame that is fixed to the rigid body.

Since there are many different ways to describe a rotation, this section establishes their mathematical background by clarifying necessary rotation conventions such as the difference between active and passive rotations and by presenting the algebraic concepts of different rotation representations, namely Tait-Bryan angles and rotation matrices, angle axis as well as quaternions. For an in depth comprehension of rotations and attitude representation, the reader is referred to [136–138].

3.1.1 Rotation Conventions

A common source for errors dealing with rotations arises from the careless use of rotational conventions, such as the handedness of the coordinate frames and the difference between active and passive rotations. Given a right handed reference coordinate frame and a vector $\boldsymbol{v}$, a rotational transformation around an angle α should be applied.

The transformation can be applied in two ways illustrated in Figure 3.1. Depending on whether the vector $\boldsymbol{v}$ is rotated about the angle α, or whether the vector is described from within another target coordinate frame that is obtained by rotating

about α in the opposite direction, an active or a passive rotation should be considered, respectively. In both cases, the resulting vectors $\boldsymbol{v}'$ are numerically identical, however, they are described from within different frames. Active rotations in right handed coordinate frames that rotate a vector from a reference frame $\mathcal{R}$ to a target frame $\mathcal{T}$ will be indicated as $\boldsymbol{R}_{\mathcal{RT}}$, it holds:

$$\boldsymbol{R}_{\mathcal{RT}}(\alpha) = (\boldsymbol{R}_{\mathcal{RT}})^{-1}(-\alpha) = \boldsymbol{R}_{\mathcal{TR}}(-\alpha) \tag{3.1}$$

FIGURE 3.1: Active rotation (left) and passive rotation (right) around an angle α. The sign of α changes depending on the type of rotation.

3.1.2 Tait-Bryan Angles and Rotation Matrices

The most common way to describe the orientation of a local frame with respect to a global frame is by the means of three subsequent rotations. Depending on whether the axis of rotation is local or global frame fixed, intrinsic and extrinsic rotations are distinguished, respectively. Any rotation sequence expressed by intrinsic rotations can be equivalently transformed into an extrinsic sequence.

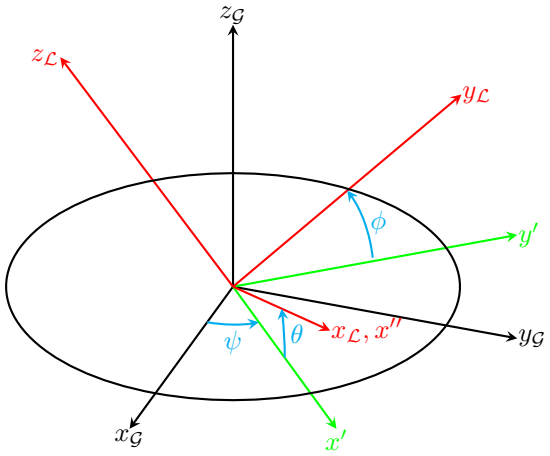


FIGURE 3.2: Rotation represented by intrinsic Tait-Bryan angles following a z-y'-x'' sequence. Note that ψ is negative in this case.

In avionic applications, Tait-Bryan angles are commonly used to describe the orientation of a plane or a drone. The intrinsic rotation angles around the z, y' and x'' axis are generally known as yaw ψ, pitch θ and roll ϕ. In this rotation sequence, intermediate axes are indicated by an additional super-scripted dash, meaning that y' is the new intermediate local frame y axis after the first rotation and x'' is the new x axis after the second rotation and therefore equivalent to the local frame axis $x_{\mathcal{L}}$. Figure 3.2 shows the intrinsic rotation sequence z-y'-x''.

The intrinsic rotation sequence z-y'-x'' is equal to the extrinsic sequence x-y-z. The individual extrinsic rotations can be written as:

$$\boldsymbol{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{3.2}$$

$$\boldsymbol{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.3}$$

$$\boldsymbol{R}_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

Concatenating these three rotations according to the x-y-z sequence yields an active rotation that allows to rotate any vector given in the global frame. Given the x axis directional vector $\boldsymbol{e}_{\mathcal{G}_x}$ in the global frame we can calculate the x axis directional vector in the local frame $\boldsymbol{e}_{\mathcal{L}_x}$ as follows:

$$\begin{aligned} \boldsymbol{e}_{\mathcal{L}_x} &= \boldsymbol{R}_{\mathcal{GL}}\boldsymbol{e}_{\mathcal{G}_x} \\ &= \boldsymbol{R}(\psi,\theta,\phi)\boldsymbol{e}_{\mathcal{G}_x} \\ &= \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi)\boldsymbol{e}_{\mathcal{G}_x} \\ &= \begin{bmatrix} \mathrm{c}_\theta\mathrm{c}_\psi & \mathrm{s}_\phi\mathrm{s}_\theta\mathrm{c}_\psi - \mathrm{c}_\phi\mathrm{s}_\psi & \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{c}_\psi + \mathrm{s}_\phi\mathrm{s}_\psi \\ \mathrm{c}_\theta\mathrm{s}_\psi & \mathrm{s}_\phi\mathrm{s}_\theta\mathrm{s}_\psi + \mathrm{c}_\phi\mathrm{c}_\psi & \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{s}_\psi - \mathrm{s}_\phi\mathrm{c}_\psi \\ -\mathrm{s}_\theta & \mathrm{s}_\phi\mathrm{c}_\theta & \mathrm{c}_\phi\mathrm{c}_\theta \end{bmatrix} \boldsymbol{e}_{\mathcal{G}_x} \end{aligned} \tag{3.5}$$

In order to describe a vector given in the local frame $\boldsymbol{v}_{\mathcal{L}}$ with respect to the global frame, a passive rotation is applied:

$$\begin{aligned} \boldsymbol{v}_{\mathcal{G}} &= (\boldsymbol{R}_{\mathcal{LG}})^{-1}\boldsymbol{v}_{\mathcal{L}} \\ &= \boldsymbol{R}_{\mathcal{GL}}\boldsymbol{v}_{\mathcal{L}} \end{aligned} \tag{3.6}$$

A drawback of Tait-Bryan angles as rotation representation is the so called gimbal lock. A gimbal lock occurs if the rotation angle around the pitch axis is equal to $\pm\frac{\pi}{2}$. In this case, a rotational degree of freedom is lost. Given the complete rotation matrix, it is impossible to recover the original yaw ψ and roll ϕ angles:

$$\begin{aligned} \boldsymbol{R}(\psi, \tfrac{\pi}{2}, \phi) &= \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\tfrac{\pi}{2})\boldsymbol{R}_x(\phi) \\ &= \begin{bmatrix} 0 & \mathrm{s}_\phi\mathrm{c}_\psi - \mathrm{c}_\phi\mathrm{s}_\psi & \mathrm{c}_\phi\mathrm{c}_\psi + \mathrm{s}_\phi\mathrm{s}_\psi \\ 0 & \mathrm{s}_\phi\mathrm{s}_\psi + \mathrm{c}_\phi\mathrm{c}_\psi & \mathrm{c}_\phi\mathrm{s}_\psi - \mathrm{s}_\phi\mathrm{c}_\psi \\ -1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & \sin(\phi-\psi) & \cos(\phi-\psi) \\ 0 & \cos(\phi-\psi) & -\sin(\phi-\psi) \\ -1 & 0 & 0 \end{bmatrix} \end{aligned} \tag{3.7}$$

3.1.3 Axis-Angle

An alternative rotation representation that overcomes the problem of the gimbal lock is called axis-angle and is illustrated in Figure 3.3. The rotation is described by the means of a principle rotation vector and the rotation angle.

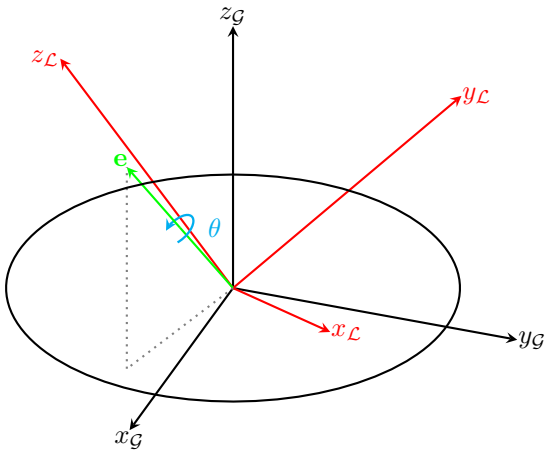


FIGURE 3.3: Rotation represented by rotation axis $\boldsymbol{e}$ and the rotation angel θ.

The axis-angle $\boldsymbol{\theta}$ consists of unit vector $\boldsymbol{e}$ indicating the principal axis of rotation and an angle θ:

$$\boldsymbol{\theta} \triangleq \begin{pmatrix} \theta & \boldsymbol{e} \end{pmatrix} = \begin{pmatrix} \theta & \begin{bmatrix} \boldsymbol{e}_x \\ \boldsymbol{e}_y \\ \boldsymbol{e}_z \end{bmatrix} \end{pmatrix} \tag{3.8}$$

Given a specific orientation represented by rotation matrix $\boldsymbol{R}$ based on the rotation matrix form of Tait-Bryan angles, the equivalent axis-angle can be calculated as:

$$\begin{aligned} \boldsymbol{e} &= \frac{1}{2\sin(\theta)} \begin{bmatrix} \boldsymbol{R}_{3,2} - \boldsymbol{R}_{2,3} \\ \boldsymbol{R}_{1,3} - \boldsymbol{R}_{3,1} \\ \boldsymbol{R}_{2,1} - \boldsymbol{R}_{1,2} \end{bmatrix} \\ \theta &= \arccos\left(\frac{\mathrm{Tr}(\boldsymbol{R}) - 1}{2}\right) \end{aligned} \tag{3.9}$$

where $\mathrm{Tr}(\boldsymbol{R})$ is the trace of $\boldsymbol{R}$ and $\boldsymbol{R}_{i,j}$ is the element in row i, column j. A vector in the global frame $\boldsymbol{v}$ can be rotated with the Rodrigues' rotation formula:

$$\boldsymbol{v}' = \cos\theta\,(\boldsymbol{v}) + \sin\theta\,(\boldsymbol{e} \times \boldsymbol{v}) + (1 - \cos\theta)\,(\boldsymbol{e} \circ \boldsymbol{v})\,\boldsymbol{e} \tag{3.10}$$

The main limitation of the axis angle representation is that subsequent rotations can not be computed without an additional conversion to another rotation representation such as Tait-Bryan angles or rotation matrices. Therefore, although the axis angle representation is very intuitive and comes with a reduction of free parameters compared to the Tait-Bryan angles and rotation matrices, it is not suitable for tracking the orientation of a UAV due to the increased computational load.

3.1.4 Quaternions

A third way to describe rotations mathematically is the use of quaternions. A quaternion is a hyper complex number of rank four and can be represented in the following way:

$$\boldsymbol{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{3.11}$$

where $\boldsymbol{q} \in \mathbb{H}$, $\{q_0, q_1, q_2, q_3\} \in \mathbb{R}$ and $\{i, j, k\} \in \mathbb{C}$ are complex numbers defined as:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.12}$$

Another, commonly used representation considers the first part of the quaternion as a scalar, while the complex part is expressed as a vector:

$$\boldsymbol{q} \triangleq \begin{bmatrix} q_0 \\ \boldsymbol{q}_v \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3.13}$$

3.1.4.1 Quaternion Properties

Next, some important properties of quaternions are outlined. The product of two quaternions $\boldsymbol{p}$ and $\boldsymbol{q}$ is given by the Hamilton product:

$$\boldsymbol{p} \otimes \boldsymbol{q} = \begin{bmatrix} p_0 q_0 - \boldsymbol{p}_v^\top \boldsymbol{q}_v \\ p_0 \boldsymbol{q}_v + q_0 \boldsymbol{p}_v + \boldsymbol{p}_v \times \boldsymbol{q}_v \end{bmatrix} = \begin{bmatrix} p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3 \\ p_0 q_1 + p_1 q_0 + p_2 q_3 - p_3 q_2 \\ p_0 q_2 - p_1 q_3 + p_2 q_0 + p_3 q_1 \\ p_0 q_3 + p_1 q_2 - p_2 q_1 + p_3 q_0 \end{bmatrix} \tag{3.14}$$

which can be also expressed as a matrix multiplication:

$$\boldsymbol{p} \otimes \boldsymbol{q} = \left[\boldsymbol{p}\right]_L \boldsymbol{q} = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3.15}$$

It is important to note that the quaternion product is in the general case non-commutative:

$$\boldsymbol{p} \otimes \boldsymbol{q} \neq \boldsymbol{q} \otimes \boldsymbol{p} \tag{3.16}$$

However, the quaternion product is associative:

$$(\boldsymbol{p} \otimes \boldsymbol{q}) \otimes \boldsymbol{r} = \boldsymbol{p} \otimes (\boldsymbol{q} \otimes \boldsymbol{r}) \tag{3.17}$$

The multiplicative identity is defined as:

$$\boldsymbol{q}_1 = \begin{bmatrix} 1 \\ \boldsymbol{0}_v \end{bmatrix} \tag{3.18}$$

which translates to an identity matrix for the quaternion product:

$$\left[\boldsymbol{q}_1\right]_L = \boldsymbol{I}_{4\times 4} \tag{3.19}$$

In order to describe valid rotations in 3D space, unit quaternions have to be used. Unit quaternions are all quaternions with unity norm. The norm of a quaternion follows the definition of complex numbers and is given by:

$$\|\boldsymbol{q}\| = \sqrt{{q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2} \tag{3.20}$$

The complex conjugate of a quaternion is defined by:

$$\boldsymbol{q}^* = \begin{bmatrix} q_0 \\ -\boldsymbol{q}_v \end{bmatrix} \tag{3.21}$$

which yields:

$$\boldsymbol{q} \otimes \boldsymbol{q}^* = \begin{bmatrix} \|\boldsymbol{q}\|^2 \\ \boldsymbol{0}_v \end{bmatrix} \tag{3.22}$$

The inverse of a quaternion $\boldsymbol{q}^{-1}$ is given by:

$$\boldsymbol{q}^{-1} = \frac{\boldsymbol{q}^*}{\|\boldsymbol{q}\|^2} \tag{3.23}$$

For unity quaternions, meaning that $\|\boldsymbol{q}\| = 1$, the inverse is equivalent to the complex conjugate quaternion:

$$\boldsymbol{q}^{-1} = \boldsymbol{q}^* \tag{3.24}$$

3.1.4.2 Quaternion Rotations

Unit quaternions can not only be used to describe rotations or the orientation of an object, moreover, they can also be used to rotate a vector $\boldsymbol{v}$ given in 3D space:

$$\begin{bmatrix} 0 \\ \boldsymbol{v}' \end{bmatrix} = \boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{v} \end{bmatrix} \otimes \boldsymbol{q}^* \tag{3.25}$$

Using Equations 3.14 and 3.25 the relation between quaternions and rotation matrices can be derived:

$$\boldsymbol{R}\{\boldsymbol{q}\} = \begin{bmatrix} {q_0}^2+{q_1}^2-{q_2}^2-{q_3}^2 & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_1q_2+q_0q_3) & {q_0}^2-{q_1}^2+{q_2}^2-{q_3}^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & {q_0}^2-{q_1}^2-{q_2}^2+{q_3}^2 \end{bmatrix} \tag{3.26}$$

In contrast to the axis angle representation, a sequence of intrinsic rotations using quaternions can be simply written as the product of the individual rotations:

$$\boldsymbol{q}_{\mathcal{AC}} = \boldsymbol{q}_{\mathcal{AB}} \otimes \boldsymbol{q}_{\mathcal{BC}} \tag{3.27}$$

which can be directly converted to the rotation matrix representation as:

$$\boldsymbol{R}_{\mathcal{AC}} = \boldsymbol{R}\{\boldsymbol{q}_{ac}\} = \boldsymbol{R}\{\boldsymbol{q}_{\mathcal{AB}} \otimes \boldsymbol{q}_{\mathcal{BC}}\} = \boldsymbol{R}\{\boldsymbol{q}_{\mathcal{AB}}\}\boldsymbol{R}\{\boldsymbol{q}_{\mathcal{BC}}\} = \boldsymbol{R}_{\mathcal{AB}}\boldsymbol{R}_{\mathcal{BC}} \tag{3.28}$$

3.1.4.3 Conversion to Tait-Bryan Angles

For a more convenient way to interpret quaternions, they can be converted to Tait-Bryan angles using the following equation:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}\left(2\left(q_0 q_1 + q_2 q_3\right), 1 - 2\left({q_1}^2 + {q_2}^2\right)\right) \\ \text{asin}\left(2\left(q_0 q_2 - q_1 q_3\right)\right) \\ \text{atan2}\left(2\left(q_0 q_3 + q_1 q_2\right), 1 - 2\left({q_2}^2 + {q_3}^2\right)\right) \end{bmatrix} \tag{3.29}$$

Given a set of Tait-Bryan angles, we can construct a valid unit quaternion with:

$$\boldsymbol{q} = \begin{bmatrix} \mathrm{c}_{\phi/2}\mathrm{c}_{\theta/2}\mathrm{c}_{\psi/2} + \mathrm{s}_{\psi/2}\mathrm{s}_{\theta/2}\mathrm{s}_{\psi/2} \\ \mathrm{s}_{\phi/2}\mathrm{c}_{\theta/2}\mathrm{c}_{\psi/2} - \mathrm{c}_{\psi/2}\mathrm{s}_{\theta/2}\mathrm{s}_{\psi/2} \\ \mathrm{c}_{\phi/2}\mathrm{s}_{\theta/2}\mathrm{c}_{\psi/2} + \mathrm{s}_{\psi/2}\mathrm{c}_{\theta/2}\mathrm{s}_{\psi/2} \\ \mathrm{c}_{\phi/2}\mathrm{c}_{\theta/2}\mathrm{s}_{\psi/2} - \mathrm{s}_{\psi/2}\mathrm{s}_{\theta/2}\mathrm{c}_{\psi/2} \end{bmatrix} \tag{3.30}$$

3.1.4.4 Conversion to Axis Angle

Similar, quaternions can be converted to the axis angle representation. For $q_0 \neq \pm 1$, we can calculate the axis $\boldsymbol{e}$ and the angle θ with:

$$\theta = 2\arccos\left(q_0\right) \tag{3.31}$$

$$\boldsymbol{e} = \frac{1}{\sqrt{1 - q_0^2}} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3.32}$$

For the case that $q_0 = \pm 1$, the rotation angle θ is zero and the rotation axis can be chosen arbitrarily. In order to convert an axis angle pair to a quaternion we can use the following relation:

$$\boldsymbol{q}\left\{\boldsymbol{\theta}\right\} = \boldsymbol{q}\left\{\theta, \boldsymbol{e}\right\} = \begin{bmatrix} \cos\left(\dfrac{\theta}{2}\right) \\ \sin\left(\dfrac{\theta}{2}\right)\boldsymbol{e} \end{bmatrix} \tag{3.33}$$

3.1.4.5 Quaternion Derivative

The derivative of a quaternion can be obtained from the differential coefficient:

$$\frac{d\boldsymbol{q}(t)}{dt} = \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(t + \Delta t) - \boldsymbol{q}(t)}{\Delta t} \tag{3.34}$$

Assuming that there is a local angular perturbation that can be described by an axis-angle $\Delta\boldsymbol{\theta}_{\mathcal{L}} = \Delta\theta \cdot \boldsymbol{u}$, we can express the quaternion disturbance $\Delta\boldsymbol{q}_{\mathcal{L}}$ using Equation 3.33 and the small angle approximation with $\sin(x) \approx x$ and $\cos(x) \approx 1$:

$$\Delta\boldsymbol{q}_{\mathcal{L}} \triangleq \lim_{\Delta\theta_{\mathcal{L}} \to 0} \boldsymbol{q}\left\{\Delta\boldsymbol{\theta}_{\mathcal{L}}\right\} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\Delta\boldsymbol{\theta}_{\mathcal{L}} \end{bmatrix} \tag{3.35}$$

The angular perturbation can be described in terms of a local angular rate $\Delta\boldsymbol{\omega}_{\mathcal{L}}$:

$$\Delta\boldsymbol{\omega}_{\mathcal{L}} \triangleq \frac{d\boldsymbol{\theta}_{\mathcal{L}}}{dt} = \lim_{\Delta t \to 0} \frac{\Delta\boldsymbol{\theta}_{\mathcal{L}}}{\Delta t} \tag{3.36}$$

Developing Equation 3.34 using the definition for small angle disturbances we get the quaternion derivative as:

$$\begin{aligned}
\dot{\boldsymbol{q}} &\triangleq \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(t+\Delta t) - \boldsymbol{q}(t)}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(t) \otimes \Delta\boldsymbol{q}_{\mathcal{L}} - \boldsymbol{q}(t)}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(t) \otimes (\Delta\boldsymbol{q}_{\mathcal{L}} - \boldsymbol{q}_1)}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(t) \otimes \left(\begin{bmatrix} 1 \\ \frac{1}{2}\Delta\boldsymbol{\theta}_{\mathcal{L}} \end{bmatrix} - \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \right)}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(t) \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\Delta t \cdot \boldsymbol{\omega}_{\mathcal{L}} \end{bmatrix}}{\Delta t} \\
&= \frac{1}{2}\boldsymbol{q}(t) \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{\mathcal{L}} \end{bmatrix}
\end{aligned} \tag{3.37}$$

3.1.4.6 Quaternion Conventions in this Work

Solà emphasizes the importance of choosing and clarifying quaternion conventions [139]. Throughout literature, different conventions are used but not always clarified. In total, there are twelve different ways to define quaternions which can be broken down to four main choices.

1. Quaternion component order
2. Handedness (*left*/*right*)
3. Rotation of frame (*passive*) or vector (*active*)
4. Direction of rotation operation (*Local-to-Global*/*Global-to-Local*)

The two most common choices are Hamilton and from the Jet Propulsion Laboratory (JPL) shown in Table 3.1. Within this work the *passive Hamilton* convention is used.

Convention	JPL	Hamilton
Component order	$(\boldsymbol{q}_v, q_0)$	$(q_0, \boldsymbol{q}_v)$
Handedness	Left-handed	Right-handed
Rotation	passive	passive
Right-to-left products	*Global-to-Local*	*Local-to-Global*
	$\boldsymbol{q} \triangleq \boldsymbol{q}_{\mathcal{L}\mathcal{G}}$	$\boldsymbol{q} \triangleq \boldsymbol{q}_{\mathcal{G}\mathcal{L}}$
	$\boldsymbol{v}_{\mathcal{L}} = \boldsymbol{q} \otimes \boldsymbol{v}_{\mathcal{G}} \otimes \boldsymbol{q}^*$	$\boldsymbol{v}_{\mathcal{G}} = \boldsymbol{q} \otimes \boldsymbol{v}_{\mathcal{L}} \otimes \boldsymbol{q}^*$

TABLE 3.1: JPL and Hamilton quaternion conventions.

3.2 Error-state Kalman Filter

Having established different ways of representing the UAV's orientation, the actual attitude determination within the context of the ego-motion estimation framework of the developed flight controller is described next. The goal of the ego-motion estimation is to determine the UAV's system state by the means of sensor fusion. The full system state includes the UAV's orientation and angular rates around each principle axis as well as its 3D position and velocity in space.

This section describes the core component of the ego-motion estimation framework: An Error-State Kalman Filter (ESKF) based on [139] which is significantly extended within this work. The extended ESKF includes a compensation mechanism for magnetic disturbances, a method to include delayed measurements as well as detailed measurement models for the integration of complementary sensor data.

The idea of a ESKF is to estimate the error $\delta\hat{\boldsymbol{x}}$ of the current state estimate $\hat{\boldsymbol{x}}$ using a Kalman Filter rather than the state itself. The current state is estimated by integrating inertial measurements and accumulated errors are corrected for with external measurements at a much lower data rate. Sometimes the ESKF approach is referred to as the indirect form of a EKF [140], while a traditional EKF is referred to as the direct form [141]. Comparing traditional EKF formulations with the ESKF approach, several strong advantages can be observed [142, 143]:

1. The separation between the inertial sensor integration and the actual Kalman Filter allows to update the state estimate independently and even in the absence of new external measurements.

2. As a consequence, non-linearities due to high dynamics are handled by inertial system propagation, while slowly varying system errors can be estimated using linearized models.

3. Furthermore, since the error dynamics are slow compared to the system dynamics, external sensors can operate at a much lower frequency than the inertial ones. In the absence of new measurements, pure inertial navigation can be used as temporary fall back solution up to a couple of minutes [143], however the quality depends highly on the navigation grade of the utilized inertial sensors.

Estimating errors only, the ESKF system states will always be close to zero giving a number of additional advantages:

4. Traditional EKF implementations rely on quaternions for attitude representation, in order to avoid singularities due to gimbal lock. Since attitude error angles are always small, a gimbal lock is avoided and hence there is no need for quaternions, thus reducing the ESKF system state by one.

5. Another benefit is that small angle approximations and other assumptions made during linearization are always valid because the error state is close to zero. Similarly, the system Jacobians are easy and fast to compute since higher order terms are very small and can be therefore neglected.

3.2.1 State Estimation

The work flow for a single ESKF iteration is shown in Figure 3.4. In the ESKF, high frequency IMU data is used as system input $\boldsymbol{u}$ and integrated into the nominal state estimate $\hat{\boldsymbol{x}}$ using a system of non-linear propagation functions $\boldsymbol{f}_{\boldsymbol{x},k}$ and the best state estimate that is available at discrete time k. The nominal state estimate neglects noise terms and model imperfections which are described by the uncertainty term $\boldsymbol{v}$. Since those uncertainties are neglected, possible errors $\delta\boldsymbol{x}$ accumulate during the system state propagation and hence need to be corrected for. The accumulated errors are estimated as $\delta\hat{\boldsymbol{x}}$ using the ESKF. Based on the system propagation equations $\boldsymbol{f}_{\boldsymbol{x},k}$, the system of error state propagation equations $\boldsymbol{f}_{\delta\boldsymbol{x},k}$ can be derived. The ESKF predicts the error state noise covariance matrix $\boldsymbol{P}$ based on the current state estimate and the noise characteristics of the system input covariance matrix $\boldsymbol{Q_i}$. The error state noise covariance matrix $\boldsymbol{P}$ is corrected once complementary sensor data $\boldsymbol{y}$ is available. Typically, complementary sensor data (e.g. GNSS, vision, magnetometer) is acquired at a frequency that is lower than the data rate of the inertial sensors used for the state propagation. For attitude determination, however, the accelerometer can be used to correct for roll and pitch error estimates, too. Similar to a regular EKF, a measurement prediction $\hat{\boldsymbol{z}}$ is made based on the latest state estimate and compared to the real measurement. By predicting the innovation covariance $\boldsymbol{S}$ based on the measurement noise covariance $\boldsymbol{R}$ and the current error state covariance $\boldsymbol{P}$, the Kalman Filter gain $\boldsymbol{K}$ can be calculated and used to estimate the current error state. The error state is injected into the current state estimate and reset to zero during the next filter iteration. The error state's covariance matrix is updated in order to reflect this reset.

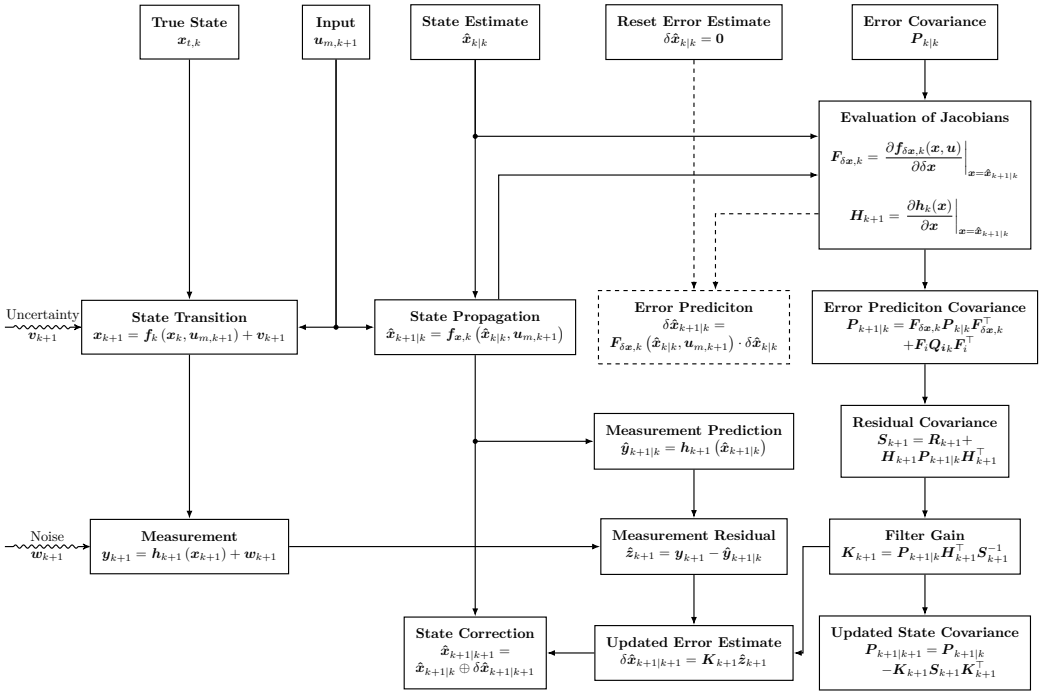


FIGURE 3.4: Flowchart of the ESKF for one cycle. The steps indicated with dashed boxes can be omitted.

3.2.1.1 System Kinematics in Continuous Time

Next, the system kinematics in continuous time for a quaternion based ESKF are derived. Based on this, the system kinematics in discrete time are developed afterwards. The kinematics are based on the work done by [139]. However, in this filter approach, the gravity vector is assumed to be constant and magnetic disturbances are added to the error state. The magnetic disturbances are used to estimate local magnetic fields that superpose the Earth's magnetic field and therefore cause errors during the heading estimation. Table 3.2 lists all variables of the ESKF and indicates whether they are obtained in the body fixed frame $\mathcal{B}$ or the navigation frame $\mathcal{N}$. Biases and local disturbances are always with respect to the $\mathcal{B}$ frame.

	True	*Nominal*	*Error*	*Measured*	*Noise*
Full State	$\boldsymbol{x}_t$	$\boldsymbol{x}$	$\delta\boldsymbol{x}$		
Position	$\boldsymbol{p}_{\mathcal{N},t}$	$\boldsymbol{p}_{\mathcal{N}}$	$\delta\boldsymbol{p}_{\mathcal{N}}$		
Velocity	$\boldsymbol{v}_{\mathcal{N},t}$	$\boldsymbol{v}_{\mathcal{N}}$	$\delta\boldsymbol{v}_{\mathcal{N}}$		
Quaternion	$\boldsymbol{q}_{\mathcal{NB},t}$	$\boldsymbol{q}_{\mathcal{NB}}$	$\delta\boldsymbol{q}_{\mathcal{NB}}$		
Rotation Matrix	$\boldsymbol{R}_{\mathcal{NB},t}$	$\boldsymbol{R}_{\mathcal{NB}}$	$\delta\boldsymbol{R}_{\mathcal{NB}}$		
Euler Angle			$\delta\boldsymbol{\theta}_{\mathcal{NB}}$		
Accelerometer Bias	$\boldsymbol{a}_{b,t}$	$\boldsymbol{a}_b$	$\delta\boldsymbol{a}_b$		$\boldsymbol{a}_{bw}$
Gyroscope Bias	$\boldsymbol{\omega}_{b,t}$	$\boldsymbol{\omega}_b$	$\delta\boldsymbol{\omega}_b$		$\boldsymbol{\omega}_{bw}$
Magnetic Disturbance	$\boldsymbol{m}_{d,t}$	$\boldsymbol{m}_d$	$\delta\boldsymbol{m}_d$		$\boldsymbol{m}_{dw}$
Acceleration	$\boldsymbol{a}_{\mathcal{B},t}$			$\boldsymbol{a}_{\mathcal{B},m}$	$\boldsymbol{a}_w$
Angular Rate	$\boldsymbol{\omega}_{\mathcal{B},t}$			$\boldsymbol{\omega}_{\mathcal{B},m}$	$\boldsymbol{\omega}_w$
Magnetic Field	$\boldsymbol{m}_{\mathcal{B},t}$			$\boldsymbol{m}_{\mathcal{B},m}$	$\boldsymbol{m}_w$
Gravity Vector	$\boldsymbol{g}_{\mathcal{N}}$				

TABLE 3.2: Variables for the quaternion based ESKF system model.

True State If the true acceleration and the true angular rate of a body are perfectly known, the current position, attitude and velocity of the body can be derived by simply integrating the inertial data. Therefore, the true kinematics in continuous time can therefore be written as:

$$\dot{\boldsymbol{p}}_{\mathcal{N},t} = \boldsymbol{v}_{\mathcal{N},t} \tag{3.38.1}$$

$$\dot{\boldsymbol{v}}_{\mathcal{N},t} = \boldsymbol{a}_{\mathcal{N},t} \tag{3.38.2}$$

$$\dot{\boldsymbol{q}}_{\mathcal{NB},t} = \frac{1}{2}\boldsymbol{q}_{\mathcal{NB},t} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{\mathcal{B},t} \end{bmatrix} \tag{3.38.3}$$

$$\dot{\boldsymbol{a}}_{b,t} = \boldsymbol{a}_{bw} \tag{3.38.4}$$

$$\dot{\boldsymbol{\omega}}_{b,t} = \boldsymbol{\omega}_{bw} \tag{3.38.5}$$

$$\dot{\boldsymbol{m}}_{d,t} = \boldsymbol{m}_{dw} \tag{3.38.6}$$

where the quaternion derivative $\dot{\boldsymbol{q}}_{\mathcal{NB},t}$ is given by Equation (3.37) for a quaternion that encodes the rotation from the navigation frame $\mathcal{N}$ to the body $\mathcal{B}$ and an angular velocity given in the body frame $\boldsymbol{\omega}_{\mathcal{B},t}$.

For real IMUs however, measurement errors that can not be mitigated by calibration need to be considered, too. Today, modern inertial navigation modules provide integrated magnetometer readings in addition to acceleration and angular rate measurements. Based on the detailed measurement models in Section 2.2, simplified measurement models after calibration for a IMU and a magnetometer can be derived as:

$$\boldsymbol{a}_{\mathcal{B},m} = \boldsymbol{R}_{\mathcal{NB},t}{}^{\top} \left(\boldsymbol{a}_{\mathcal{N},t} - \boldsymbol{g}_{\mathcal{N}}\right) + \boldsymbol{a}_{b,t} + \boldsymbol{a}_{w} \tag{3.39}$$

$$\boldsymbol{\omega}_{\mathcal{B},m} = \boldsymbol{\omega}_{\mathcal{B},t} + \boldsymbol{\omega}_{b,t} + \boldsymbol{\omega}_{w} \tag{3.40}$$

$$\boldsymbol{m}_{\mathcal{B},m} = \boldsymbol{R}_{\mathcal{NB},t}{}^{\top} \left(\boldsymbol{m}_{\mathcal{N},t}\right) + \boldsymbol{m}_{d,t} + \boldsymbol{m}_{w} \tag{3.41}$$

where the rotation matrix is derived from the quaternion according to Equation (3.26) $\boldsymbol{R}_{\mathcal{NB},t} \triangleq \boldsymbol{R}\{\boldsymbol{q}_{\mathcal{NB},t}\}$. Solving Equation (3.39) and (3.40) for the true values and substituting it back into the system of Equations (3.38) the kinematic system yields:

$$\dot{\boldsymbol{p}}_{\mathcal{N},t} = \boldsymbol{v}_{\mathcal{N},t} \tag{3.42.1}$$

$$\dot{\boldsymbol{v}}_{\mathcal{N},t} = \boldsymbol{R}_{\mathcal{NB},t} \left(\boldsymbol{a}_{\mathcal{B},m} - \boldsymbol{a}_{b,t} - \boldsymbol{a}_{w}\right) + \boldsymbol{g}_{\mathcal{N}} \tag{3.42.2}$$

$$\dot{\boldsymbol{q}}_{\mathcal{NB},t} = \frac{1}{2}\boldsymbol{q}_{\mathcal{NB},t} \otimes \begin{bmatrix} 0 \\ \left(\boldsymbol{\omega}_{\mathcal{B},m} - \boldsymbol{\omega}_{b,t} - \boldsymbol{\omega}_{w}\right) \end{bmatrix} \tag{3.42.3}$$

$$\dot{\boldsymbol{a}}_{b,t} = \boldsymbol{a}_{bw} \tag{3.42.4}$$

$$\dot{\boldsymbol{\omega}}_{b,t} = \boldsymbol{\omega}_{bw} \tag{3.42.5}$$

$$\dot{\boldsymbol{m}}_{d,t} = \boldsymbol{m}_{dw} \tag{3.42.6}$$

which can be written as the true system state $\boldsymbol{x}_t$, noisy system input $\boldsymbol{u}_m$ and white Gaussian noise $\boldsymbol{v}$:

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{p}_{\mathcal{N},t} \\ \boldsymbol{v}_{\mathcal{N},t} \\ \boldsymbol{q}_{\mathcal{NB},t} \\ \boldsymbol{a}_{b,t} \\ \boldsymbol{\omega}_{b,t} \\ \boldsymbol{m}_{d,t} \end{bmatrix} \qquad \boldsymbol{u}_m = \begin{bmatrix} \boldsymbol{a}_{\mathcal{B},m} \\ \boldsymbol{\omega}_{\mathcal{B},m} \end{bmatrix} \qquad \boldsymbol{v} = \begin{bmatrix} \boldsymbol{a}_{w} \\ \boldsymbol{\omega}_{w} \\ \boldsymbol{a}_{bw} \\ \boldsymbol{\omega}_{bw} \\ \boldsymbol{m}_{dw} \end{bmatrix} \tag{3.43}$$

We can rewrite the kinematic system in terms of a system of functions $\boldsymbol{f}$:

$$\dot{\boldsymbol{x}}_t = \boldsymbol{f}\left(\boldsymbol{x}_t, \boldsymbol{u}_m\right) + \boldsymbol{v} \tag{3.44}$$

In order to utilize the advantages of the ESKF approach, the system given with the Equations in (3.42) has to be reformulated in terms of nominal and error system state. The true system state $\boldsymbol{x}_t$ is therefore split into the nominal system state $\boldsymbol{x}$ and its errors state $\delta\boldsymbol{x}$ as $\boldsymbol{x}_t = \boldsymbol{x} \oplus \delta\boldsymbol{x}$, where $\oplus$ denotes a generic composition and is given by:

$$\boldsymbol{x}_t = \boldsymbol{x} \oplus \delta\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_{\mathcal{N}} + \delta\boldsymbol{p}_{\mathcal{N}} \\ \boldsymbol{v}_{\mathcal{N}} + \delta\boldsymbol{v}_{\mathcal{N}} \\ \boldsymbol{q}_{\mathcal{NB}} \otimes \delta\boldsymbol{q}_{\mathcal{NB}} \\ \boldsymbol{a}_{b} + \delta\boldsymbol{a}_{b} \\ \boldsymbol{\omega}_{b} + \delta\boldsymbol{\omega}_{b} \\ \boldsymbol{m}_{d} + \delta\boldsymbol{m}_{d} \end{bmatrix} \tag{3.45}$$

For local disturbance error angles $\delta\boldsymbol{\theta}_{\mathcal{NB}}$ close to zero, the local quaternion perturbation $\delta\boldsymbol{q}_{\mathcal{NB}}$ can be obtained by using the small angle approximation according to Equation (3.35) as:

$$\delta\boldsymbol{q}_{\mathcal{NB}} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta}_{\mathcal{NB}} \end{bmatrix} \tag{3.46}$$

The nominal state and the reduced error state with the minimal number of parameters required for attitude representation can be written as:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_{\mathcal{N}} \\ \boldsymbol{v}_{\mathcal{N}} \\ \boldsymbol{q}_{\mathcal{NB}} \\ \boldsymbol{a}_b \\ \boldsymbol{\omega}_b \\ \boldsymbol{m}_d \end{bmatrix} \in \mathbb{R}^{19\times 1} \qquad \delta\boldsymbol{x} = \begin{bmatrix} \delta\boldsymbol{p}_{\mathcal{N}} \\ \delta\boldsymbol{v}_{\mathcal{N}} \\ \delta\boldsymbol{\theta}_{\mathcal{NB}} \\ \delta\boldsymbol{a}_b \\ \delta\boldsymbol{\omega}_b \\ \delta\boldsymbol{m}_d \end{bmatrix} \in \mathbb{R}^{18\times 1} \tag{3.47}$$

Nominal State Using the representation in Equation (3.47), the true system kinematics in Equations (3.42) can be split into the nominal state kinematics and linearized error state kinematics. Since noise and other perturbations are handled by the error state kinematics, the nominal state kinematics are simply derived as:

$$\dot{\boldsymbol{p}}_{\mathcal{N}} = \boldsymbol{v}_{\mathcal{N}} \tag{3.48.1}$$
$$\dot{\boldsymbol{v}}_{\mathcal{N}} = \boldsymbol{R}_{\mathcal{NB}}\left(\boldsymbol{a}_{\mathcal{B},m} - \boldsymbol{a}_b\right) + \boldsymbol{g}_{\mathcal{N}} \tag{3.48.2}$$
$$\dot{\boldsymbol{q}}_{\mathcal{NB}} = \frac{1}{2}\boldsymbol{q}_{\mathcal{NB}} \otimes \begin{bmatrix} 0 \\ (\boldsymbol{\omega}_{\mathcal{B},m} - \boldsymbol{\omega}_b) \end{bmatrix} \tag{3.48.3}$$
$$\dot{\boldsymbol{a}}_b = 0 \tag{3.48.4}$$
$$\dot{\boldsymbol{\omega}}_b = 0 \tag{3.48.5}$$
$$\dot{\boldsymbol{m}}_d = 0 \tag{3.48.6}$$

Error State The linearized error dynamics are derived in [139], where second order terms for the velocity and angular error dynamics are neglected based on the small error state assumptions:

$$\dot{\delta\boldsymbol{p}}_{\mathcal{N}} = \delta\boldsymbol{v} \tag{3.49.1}$$
$$\dot{\delta\boldsymbol{v}}_{\mathcal{N}} = -\boldsymbol{R}_{\mathcal{NB}}\left[\boldsymbol{a}_{\mathcal{B},m} - \boldsymbol{a}_b\right]_\times \delta\boldsymbol{\theta} - \boldsymbol{R}_{\mathcal{NB}}\delta\boldsymbol{a}_b - \boldsymbol{R}_{\mathcal{NB}}\boldsymbol{a}_w \tag{3.49.2}$$
$$\dot{\delta\boldsymbol{\theta}}_{\mathcal{NB}} = -\left[\boldsymbol{w}_{\mathcal{B},m} - \boldsymbol{w}_b\right]_\times \delta\boldsymbol{\theta} - \delta\boldsymbol{\omega}_b - \boldsymbol{\omega}_w \tag{3.49.3}$$
$$\dot{\delta\boldsymbol{a}}_b = \boldsymbol{a}_{bw} \tag{3.49.4}$$
$$\dot{\delta\boldsymbol{\omega}}_b = \boldsymbol{\omega}_{bw} \tag{3.49.5}$$
$$\dot{\delta\boldsymbol{m}}_d = \boldsymbol{m}_{dw} \tag{3.49.6}$$

where the $[\bullet]_\times$ describes the skew operator that produces the cross-product matrix for a given vector $\mathbf{v}$:

$$[\mathbf{v}]_\times \triangleq \begin{bmatrix} 0 & -\mathrm{v}_z & \mathrm{v}_y \\ \mathrm{v}_z & 0 & -\mathrm{v}_x \\ -\mathrm{v}_y & \mathrm{v}_x & 0 \end{bmatrix} \tag{3.50}$$

3.2.1.2 System Kinematics in Discrete Time

The previously stated system kinematics in continuous time need to be discretized, in order to be applied to discrete time steps and measurements. The error state and the nominal system state can be considered separately. In order to improve the readability of the discrete kinematic equations, frame designations will be neglected hereafter, however, the introduced frames remain valid. The discrete true state $\boldsymbol{x}_{t,k}$ at time k can be written as:

$$\boldsymbol{x}_{t,k} = \boldsymbol{x}_k \oplus \delta\boldsymbol{x}_k \tag{3.51}$$

Nominal State As aforementioned above, one advantage of the ESKF is that the nominal system state kinematics given by the Equations in (3.48) are free of noise and disturbances. We can introduce a propagation function for the discrete nominal state with:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_{\boldsymbol{x},k}\left(\boldsymbol{x}_k, \boldsymbol{u}_{m,k+1}\right) \tag{3.52}$$

Since $\boldsymbol{f}_{\boldsymbol{x},k}$ is noise and disturbance free, standard numeric integration methods can be applied to propagate the nominal state estimate over time. Throughout literature, the Euler and the midpoint method are most commonly used, since they offer sufficient accuracy and simultaneously low computational complexity at high propagation rates [83, 144–146]. Higher order integration methods, like Runge-Kutta methods are also popular [147, 148]. Using the Euler method and assuming that the derivative remains constant between two consecutive measurements, the discrete nominal state kinematics are given by:

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \boldsymbol{v}_k\Delta t + \frac{1}{2}\left[\boldsymbol{R}\left(\boldsymbol{a}_{m,k+1} - \boldsymbol{a}_{b,k}\right) + \boldsymbol{g}\right]\Delta t^2 \tag{3.53.1}$$

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \left[\boldsymbol{R}\left(\boldsymbol{a}_{m,k+1} - \boldsymbol{a}_{b,k}\right) + \boldsymbol{g}\right]\Delta t \tag{3.53.2}$$

$$\boldsymbol{q}_{k+1} = \boldsymbol{q} \otimes \boldsymbol{q}\left\{\left(\boldsymbol{\omega}_{m,k+1} - \boldsymbol{\omega}_{b,k}\right)\Delta t\right\} \tag{3.53.3}$$

$$\boldsymbol{a}_{b,k+1} = \boldsymbol{a}_{b,k} \tag{3.53.4}$$

$$\boldsymbol{\omega}_{b,k+1} = \boldsymbol{\omega}_{b,k} \tag{3.53.5}$$

$$\boldsymbol{m}_{d,k+1} = \boldsymbol{m}_{d,k} \tag{3.53.6}$$

where Δt is the time between samples k and $k+1$ and $\boldsymbol{q}\left\{\boldsymbol{\omega}\Delta t\right\}$ is the quaternion obtained using Equation (3.33) and integrating the angular rate $\boldsymbol{\omega}$ using Euler's method:

$$\boldsymbol{q}\left\{\boldsymbol{\omega}\Delta t\right\} = \begin{bmatrix} \cos\dfrac{\|\boldsymbol{\omega}\|\Delta t}{2} \\ \dfrac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}\sin\dfrac{\|\boldsymbol{\omega}\|\Delta t}{2} \end{bmatrix} \tag{3.54}$$

Error State The error state models the system uncertainties such as noise and disturbances as stochastic signals. Since the integration of stochastic signals results in random pulses, the discrete error state model derived from the Equations in (3.49) reads:

$$\delta \boldsymbol{p}_{k+1} = \delta \boldsymbol{p}_k + \delta \boldsymbol{v}_k \Delta t \tag{3.55.1}$$
$$\delta \boldsymbol{v}_{k+1} = \delta \boldsymbol{v}_k + \left(-\boldsymbol{R}\left[\boldsymbol{a}_{m,k+1} - \boldsymbol{a}_{b,k}\right]_\times \delta \boldsymbol{\theta}_k - \boldsymbol{R}\delta \boldsymbol{a}_{b,k}\right)\Delta t + \boldsymbol{v_i} \tag{3.55.2}$$
$$\delta \boldsymbol{\theta}_{k+1} = \boldsymbol{R}^\top \left\{\left(\boldsymbol{\omega}_{m,k+1} - \boldsymbol{\omega}_{b,k}\right)\Delta t\right\} \delta \boldsymbol{\theta}_k - \delta \boldsymbol{\omega}_{b,k}\Delta t + \boldsymbol{\theta_i} \tag{3.55.3}$$
$$\delta \boldsymbol{a}_{b,k+1} = \delta \boldsymbol{a}_{b,k} + \boldsymbol{a_i} \tag{3.55.4}$$
$$\delta \boldsymbol{\omega}_{b,k+1} = \delta \boldsymbol{\omega}_{b,k} + \boldsymbol{\omega_i} \tag{3.55.5}$$
$$\delta \boldsymbol{m}_{d,k+1} = \delta \boldsymbol{m}_{d,k} + \boldsymbol{m_i} \tag{3.55.6}$$

where $\boldsymbol{v_i}$, $\boldsymbol{\theta_i}$, $\boldsymbol{a_i}$, $\boldsymbol{\omega_i}$ and $\boldsymbol{m_i}$ are the random impulses applied to the according error states and $\boldsymbol{R}\{\boldsymbol{\omega}\Delta t\}$ can be obtained using Equations (3.26) and (3.33) and the following relationship:

$$\boldsymbol{R}\{\boldsymbol{\omega}\Delta t\} = \boldsymbol{R}\{\boldsymbol{q}\{\boldsymbol{\omega}\Delta t\}\} \tag{3.56}$$

The impulses are modeled by white Gaussian noise with zero mean. For a Gaussian distributed random variable $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and a distribution that can be described with the covariance matrix $\boldsymbol{\Sigma}$ we write:

$$\boldsymbol{x} \sim \mathcal{N}\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} \tag{3.57}$$

Their respective covariance matrices are obtained by integrating the covariances of the respective noise signals over the sample time Δt:

$$\boldsymbol{v_i} \sim \mathcal{N}\{\boldsymbol{0}, \boldsymbol{V_i}\}, \qquad \text{with } \boldsymbol{V_i} = \sigma^2_{\boldsymbol{a}_w}\Delta t^2 \boldsymbol{I}_{3\times 3} \tag{3.58}$$
$$\boldsymbol{\theta_i} \sim \mathcal{N}\{\boldsymbol{0}, \boldsymbol{\Theta_i}\}, \qquad \text{with } \boldsymbol{\Theta_i} = \sigma^2_{\boldsymbol{\omega}_w}\Delta t^2 \boldsymbol{I}_{3\times 3} \tag{3.59}$$
$$\boldsymbol{a_i} \sim \mathcal{N}\{\boldsymbol{0}, \boldsymbol{A_i}\}, \qquad \text{with } \boldsymbol{A_i} = \sigma^2_{\boldsymbol{a}_{bw}}\Delta t \boldsymbol{I}_{3\times 3} \tag{3.60}$$
$$\boldsymbol{\omega_i} \sim \mathcal{N}\{\boldsymbol{0}, \boldsymbol{\Omega_i}\}, \qquad \text{with } \boldsymbol{\Omega_i} = \sigma^2_{\boldsymbol{\omega}_{bw}}\Delta t \boldsymbol{I}_{3\times 3} \tag{3.61}$$
$$\boldsymbol{m_i} \sim \mathcal{N}\{\boldsymbol{0}, \boldsymbol{M_i}\}, \qquad \text{with } \boldsymbol{M_i} = \sigma^2_{\boldsymbol{m}_{dw}}\Delta t \boldsymbol{I}_{3\times 3} \tag{3.62}$$

where $\sigma_{\boldsymbol{a}_w}$ is the velocity random walk and $\sigma_{\boldsymbol{a}_{bw}}$ is the bias instability of the accelerometer and $\sigma_{\boldsymbol{\omega}_w}$ is the angular random walk and $\sigma_{\boldsymbol{\omega}_{bw}}$ is the bias instability of the gyroscope. $\sigma_{\boldsymbol{m}_{dw}}$ describes the expected deviation of magnetic field measurements due to electro magnetic disturbances detected in the vicinity of the magnetometer, e.g. UAV motor current. Random walk and bias instability can be determined experimentally or are given in the IMU datasheet, while the influence of electro-magnetic disturbances are application dependent and have to be determined experimentally.

The sensor characteristics required for the implementation of the presented sensor fusion approach are derived in Section 2.2.2.1.

Error State Jacobian and Perturbation The error state Jacobian and the perturbation matrix can be obtained by inspecting the discrete error kinematics. Subdividing each equation into deterministic and stochastic parts, we can rearrange and pack the Equations in (3.55) as:

$$\begin{aligned}\delta \boldsymbol{x}_{k+1} &= \boldsymbol{f}_{\delta \boldsymbol{x},k}\left(\boldsymbol{x}_k, \delta \boldsymbol{x}_k, \boldsymbol{u}_{m,k+1}, \boldsymbol{i}\right) \\ &= \delta \boldsymbol{x}_k \cdot \underbrace{\left.\frac{\partial \boldsymbol{f}_{\delta \boldsymbol{x},k}}{\partial \delta \boldsymbol{x}}\right|_{\boldsymbol{x},\boldsymbol{u}_m}}_{\boldsymbol{F}_{\delta \boldsymbol{x},k}(\boldsymbol{x}_k,\boldsymbol{u}_{m,k+1})} + \boldsymbol{i} \cdot \underbrace{\left.\frac{\partial \boldsymbol{f}_{\delta \boldsymbol{x},k}}{\partial \boldsymbol{i}}\right|_{\boldsymbol{x},\boldsymbol{u}_m}}_{\boldsymbol{F}_{\boldsymbol{i}}}\end{aligned} \tag{3.63}$$

where $\boldsymbol{i}$ is a vector containing all perturbation impulses:

$$\boldsymbol{i} = \begin{bmatrix}\boldsymbol{v}_i & \boldsymbol{\theta}_i & \boldsymbol{a}_i & \boldsymbol{\omega}_i & \boldsymbol{m}_i\end{bmatrix}^\top \in \mathbb{R}^{15\times 1} \tag{3.64}$$

The system transition matrix $\boldsymbol{F}_{\delta \boldsymbol{x},k}$ is the Jacobian of $\boldsymbol{f}_{\delta \boldsymbol{x},k}$ with respect to the error state vector $\delta \boldsymbol{x}$ evaluated at the latest nominal state estimation $\hat{\boldsymbol{x}}_{k+1|k}$ and the current measurements $\boldsymbol{u}_{m,k+1}$. It is given by:

$$\begin{aligned}\boldsymbol{F}_{\delta \boldsymbol{x},k} &= \left.\frac{\partial \boldsymbol{f}_{\delta \boldsymbol{x},k}\left(\boldsymbol{x}_k, \delta \boldsymbol{x}_k, \boldsymbol{u}_{m,k+1}, \boldsymbol{i}\right)}{\partial \delta \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}, \boldsymbol{u}_m=\boldsymbol{u}_{m,k+1}} \\ &= \begin{bmatrix}\boldsymbol{I} & \boldsymbol{I}\Delta t & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & -\boldsymbol{R}\left[\boldsymbol{a}_{m,k+1} - \boldsymbol{a}_{b,k}\right]_\times \Delta t & -\boldsymbol{R}\Delta t & \boldsymbol{0} & \boldsymbol{I}\Delta t & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{R}^\top\left\{\left(\boldsymbol{\omega}_{m,k+1} - \boldsymbol{\omega}_{b,k}\right)\Delta t\right\} & \boldsymbol{0} & -\boldsymbol{I}\Delta t & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}}\end{aligned} \tag{3.65}$$

The perturbation matrix $\boldsymbol{F}_{\boldsymbol{i}}$ is the Jacobian of $\boldsymbol{f}_{\delta \boldsymbol{x},k}$ with respect to the perturbation vector $\boldsymbol{i}$ and given by:

$$\begin{aligned}\boldsymbol{F}_{\boldsymbol{i}} &= \frac{\partial \boldsymbol{f}_{\delta \boldsymbol{x},k}\left(\boldsymbol{x}_k, \delta \boldsymbol{x}_k, \boldsymbol{u}_{m,k+1}, \boldsymbol{i}\right)}{\partial \boldsymbol{i}} \\ &= \begin{bmatrix}\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\end{aligned} \tag{3.66}$$

The ESKF prediction equation are then given as:

$$\delta \hat{\boldsymbol{x}}_{k+1|k} = \boldsymbol{F}_{\delta \boldsymbol{x},k}\left(\hat{\boldsymbol{x}}_{k+1|k}, \boldsymbol{u}_{m,k+1}\right) \cdot \delta \hat{\boldsymbol{x}}_{k|k} \tag{3.67}$$

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{F}_{\delta \boldsymbol{x},k} \boldsymbol{P}_{k|k} \boldsymbol{F}_{\delta \boldsymbol{x},k}^\top + \boldsymbol{F}_{\boldsymbol{i}} \boldsymbol{Q}_{\boldsymbol{i}} \boldsymbol{F}_{\boldsymbol{i}}^\top \tag{3.68}$$

where $\boldsymbol{P}$ is the covariance matrix of the error state estimate $\delta \hat{\boldsymbol{x}}$ with $\delta \boldsymbol{x} \sim \mathcal{N}\left\{\delta \hat{\boldsymbol{x}}, \boldsymbol{P}\right\}$ and $\boldsymbol{Q}_{\boldsymbol{i}}$ is the covariance matrix of the perturbation impulses:

$$\boldsymbol{Q}_{\boldsymbol{i}} = \operatorname{diag}\left(\boldsymbol{V}_i, \boldsymbol{\Theta}_i, \boldsymbol{A}_i, \boldsymbol{\Omega}_i, \boldsymbol{M}_i\right) \tag{3.69}$$

3.2.2 Complementary Sensor Integration

The error states can be observed once a complementary sensor measurement arrives. Typically, the complementary sensor is sampled at a lower rate than the ESKF nominal state is propagated using inertial measurements. Common complementary observations include GNSS measurements, barometric height readings, proximity sensor distances, visual ego-motion estimations or odometry data. However, for attitude estimation, the Earth gravity vector observed by an accelerometer as well as magnetic field measurements can be used as complementary source, too. Similar to a standard EKF, the measurement is predicted based on some linear or non-linear function $\boldsymbol{h}$ that depends on the system state $\boldsymbol{x}$ and has additive Gaussian noise $\boldsymbol{w} \sim \mathcal{N}\{\boldsymbol{0}, \boldsymbol{R}\}$. The measurement prediction $\boldsymbol{y}$ is the given by:

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}_t) + \boldsymbol{w} \tag{3.70}$$

In order to process the ESKF covariance matrices correctly, the measurement Jacobian $\boldsymbol{H}$ with respect to the error state $\delta\boldsymbol{x}$ needs to be computed. It can be developed using the chain rule for partial derivatives:

$$\begin{aligned} \boldsymbol{H} &\triangleq \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x})}{\partial \delta \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \underbrace{\left.\frac{\partial \boldsymbol{h}(\boldsymbol{x})}{\partial \boldsymbol{x}_t}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}}}_{\boldsymbol{H}_{\boldsymbol{x}}} \cdot \underbrace{\left.\frac{\partial \boldsymbol{x}_t}{\partial \delta \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}}}_{\boldsymbol{X}_{\delta\boldsymbol{x}}} \end{aligned} \tag{3.71}$$

The Jacobian matrix $\boldsymbol{H}_{\boldsymbol{x}}$ depends on the measurement equation only and can be derived in the same way as for a regular EKF. $\boldsymbol{X}_{\delta\boldsymbol{x}}$ is derived in [139] as:

$$\begin{aligned} \boldsymbol{X}_{\delta\boldsymbol{x}} &= \left.\frac{\partial\left(\boldsymbol{x} \oplus \delta\boldsymbol{x}\right)}{\partial \delta \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \begin{bmatrix} \frac{\partial(\boldsymbol{p}+\delta\boldsymbol{p})}{\partial\delta\boldsymbol{p}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{\partial(\boldsymbol{v}+\delta\boldsymbol{v})}{\partial\delta\boldsymbol{v}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \frac{\partial(\boldsymbol{q}\otimes\delta\boldsymbol{q})}{\partial\delta\boldsymbol{q}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \frac{\partial(\boldsymbol{a}_b+\delta\boldsymbol{a}_b)}{\partial\delta\boldsymbol{a}_b} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \frac{\partial(\boldsymbol{\omega}_b+\delta\boldsymbol{\omega}_b)}{\partial\delta\boldsymbol{\omega}_b} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \frac{\partial(\boldsymbol{m}_b+\delta\boldsymbol{m}_b)}{\partial\delta\boldsymbol{m}_b} \end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \begin{bmatrix} \boldsymbol{I}_{6\times6} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_{\delta\boldsymbol{\theta}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{9\times9} \end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \end{aligned} \tag{3.72}$$

where $\boldsymbol{Q}_{\delta\boldsymbol{\theta}}$ is derived using the chain rule and the small error angle assumption from Equation (3.35) with $\delta\boldsymbol{q} \rightarrow \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix}$:

$$\begin{aligned} \boldsymbol{Q}_{\delta\boldsymbol{\theta}} &= \left.\frac{\partial\left(\boldsymbol{q} \otimes \delta\boldsymbol{q}\right)}{\partial\delta\boldsymbol{\theta}}\right|_{\boldsymbol{q}=\hat{\boldsymbol{q}}_{k+1|k}} \\ &= \left.\frac{\partial\left(\left[\boldsymbol{q}\right]_L \delta\boldsymbol{q}\right)}{\partial\delta\boldsymbol{q}}\right|_{\boldsymbol{q}=\hat{\boldsymbol{q}}_{k+1|k}} \cdot \left.\frac{\partial\delta\boldsymbol{q}}{\partial\delta\boldsymbol{\theta}}\right|_{\delta\boldsymbol{\theta}=0} \\ &= \left[\hat{\boldsymbol{q}}_{k+1|k}\right]_L \cdot \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.73}$$

Once the Jacobian is evaluated, the ESKF follows the standard EKF equations. Hence, the measurement residual $\hat{\boldsymbol{z}}$ and the innovation covariance $\boldsymbol{S}$ are given with:

$$\hat{\boldsymbol{z}}_{k+1} = \boldsymbol{y}_{k+1} - \boldsymbol{h}\left(\hat{\boldsymbol{x}}_{k+1|k}\right) \tag{3.74}$$

$$\boldsymbol{S}_{k+1} = \boldsymbol{R}_{k+1} + \boldsymbol{H}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^{\top} \tag{3.75}$$

where $\boldsymbol{R}_{k+1}$ is the covariance describing Gaussian properties of the measurement $\boldsymbol{y}_{k+1}$. The Kalman filter gain $\boldsymbol{K}$ is calculated as:

$$\boldsymbol{K}_{k+1} = \boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^{\top}\boldsymbol{S}_{k+1}^{-1} \tag{3.76}$$

Using the Kalman filter gain, the error state $\delta\hat{\boldsymbol{x}}_{k+1|k+1}$ and its covariance matrix $\boldsymbol{P}_{k+1|k+1}$ can be updated:

$$\delta\hat{\boldsymbol{x}}_{k+1|k+1} = \boldsymbol{K}_{k+1}\hat{\boldsymbol{z}}_{k+1} \tag{3.77}$$

$$\boldsymbol{P}_{k+1|k+1} = \left(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}\right)\boldsymbol{P}_{k+1|k} \tag{3.78}$$

The covariance update in Equation (3.78) is known to suffer poor numerical stability. Throughout literature, the Joseph form is the preferred update method, however, computationally quite expensive. Therefore, the covariance update is done using a symmetric update form and an additional measure proposed in [149] to ensure that the covariance matrix remains symmetric and positive despite the influence of numerical inaccuracies:

$$\boldsymbol{P}_{k+1|k+1} = \boldsymbol{P}_{k+1|k} - \boldsymbol{K}_{k+1}\boldsymbol{S}_{k+1}\boldsymbol{K}_{k+1}^{\top} \tag{3.79}$$

$$\boldsymbol{P}_{k+1|k+1} \leftarrow \frac{1}{2}\left(\boldsymbol{P}_{k+1|k+1} + \boldsymbol{P}_{k+1|k+1}^{\top}\right) \tag{3.80}$$

In the final step, the estimated error states need to be injected into the nominal filter state. This can be done using the appropriate compositions from Equation (3.51):

$$\hat{\boldsymbol{x}}_{k+1|k+1} = \hat{\boldsymbol{x}}_{k+1|k} \oplus \delta\hat{\boldsymbol{x}}_{k+1|k+1} \tag{3.81}$$

After the injection, the error state estimate is reset to zero. Strictly speaking, its covariance matrix needs to be updated accordingly in order to reflect the reset step. However, as noted by [139], the correction factor depends on the small angle error $\delta\boldsymbol{\theta}$ only and thus can be neglected in major cases.

3.2.2.1 Accelerometer

The accelerometer can be used as complementary sensor to compensate for the gyroscope drift around the roll and pitch axis. The raw accelerometer measurements are processed as depicted in Figure 3.5.

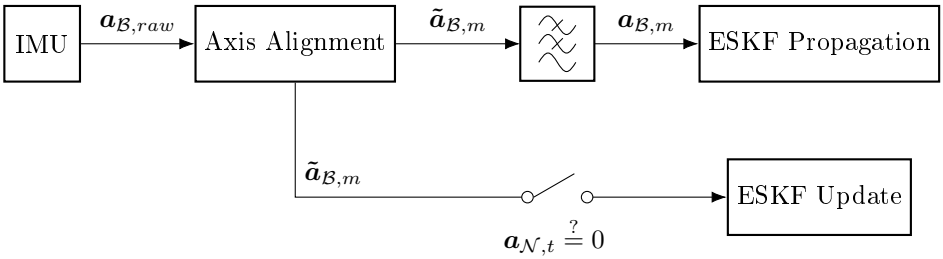


FIGURE 3.5: ESKF accelerometer integration.

First, the axis misalignment error of the IMU is corrected according to the calibration algorithm in Section 2.2.2.2. For the ESKF propagation in Equation (3.53), noise caused by vibrations are mitigated using mechanical damping and a low-pass filter as described in Section 2.2.2.3. Finally, for the ESKF attitude update, an algorithm decides whether the UAV is accelerated by another force than the Earth gravitation. If no other force is acting on the UAV, the accelerometer can be used update the attitude.

Conditioning The accelerometer raw measurements have to be analyzed to detect whether another force despite the Earth's gravity is acting upon the UAV or not. Since the accelerometer measurements are noisy, the simplest method is to observe the current UAV velocity estimate. If the velocity is close to zero or remains constant within certain thresholds, the accelerometer measurement can be used for the attitude update. However, in cases where no external position or velocity reference is available, this approach fails since the velocity estimate without complementary corrections is drifting and unreliable. A similar problem occurs in pedestrian tracking applications, where zero-velocity conditions have to be detected. Skog et al. evaluate different zero velocity detection algorithms in [150] and conclude that the stance hypothesis optimal detector (SHOE) outperforms other approaches. As mentioned in [151], the SHOE detector can be used to detect whether the UAV is accelerating, too, and is given by:

$$\frac{1}{W}\sum_{k=n}^{n+W-1}\frac{1}{\sigma_a^2}\left\|\boldsymbol{a}_{\mathcal{B},m}(k)-g\frac{\bar{\boldsymbol{a}}_{\mathcal{B},m,n}}{\|\bar{\boldsymbol{a}}_{\mathcal{B},m,n}\|}\right\|^2+\frac{1}{\sigma_\omega^2}\left\|\boldsymbol{\omega}_{\mathcal{B},m,k}\right\|^2<\gamma \tag{3.82}$$

where W is a parameter describing the window size, σ_a^2 and σ_ω^2 are used to tune a weighing ratio between the gyroscope and the accelerometer measurements, g is the gravity constant, γ is the detection threshold and $\bar{\boldsymbol{a}}_{\mathcal{B},m,n}$ is the accelerometer sample mean over W samples:

$$\bar{\boldsymbol{a}}_{\mathcal{B},m,n}=\frac{1}{W}\sum_{k=n}^{n+W-1}\boldsymbol{a}_{\mathcal{B},m}(k) \tag{3.83}$$

If the Equation (3.82) holds true, the UAV is considered to be in non-accelerating mode. The results of the SHOE detector are shown in Figure 3.6 for a UAV flight .

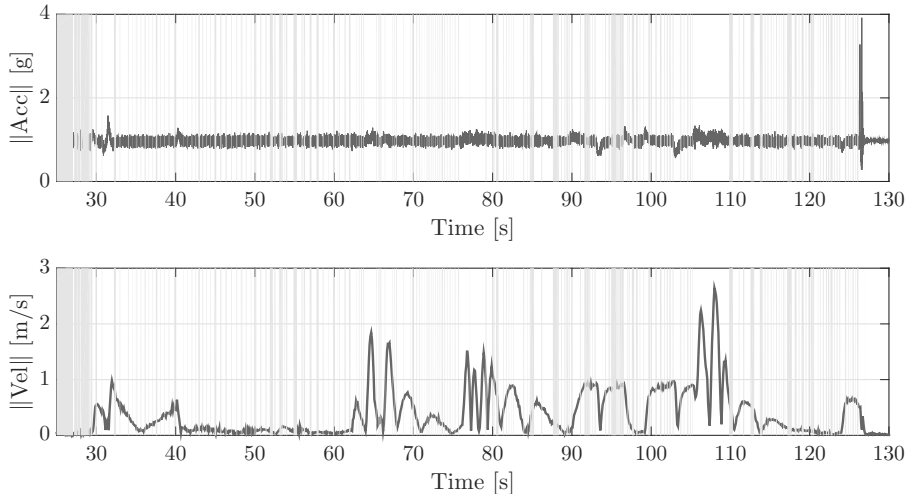


FIGURE 3.6: SHOE acceleration detection: W=10, σ_a^2=σ_ω^2, γ=25 and a 200 Hz sample rate. The upper plot shows the absolute acceleration and the lower plot the UAV speed. A gray background indicates a non-accelerating state detected by SHOE.

ESKF Integration Assuming that no other forces act on the UAV so that $\boldsymbol{a}_{\mathcal{N},t} = 0$, the measurement model for the accelerometer attitude update based on Equation (3.39) reads:

$$\begin{aligned} \boldsymbol{y_g} = \tilde{\boldsymbol{a}}_{\mathcal{B},m} &= \boldsymbol{h_g}(\boldsymbol{x}) + \boldsymbol{a}_w \\ &= -\boldsymbol{R}_{\mathcal{NB}}{}^{\top} \boldsymbol{g}_{\mathcal{N}} + \boldsymbol{a}_b + \boldsymbol{a}_w \end{aligned} \tag{3.84}$$

If only the gravitational force is acting upon the UAV with $\boldsymbol{g}_{\mathcal{N}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\top}$ and Equation (3.26), the measurement Jacobian can be evaluated by:

$$\begin{aligned} \boldsymbol{H_g} &= \left. \frac{\partial \boldsymbol{h_g}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \begin{bmatrix} \boldsymbol{0}_{3\times 6} & \dfrac{\partial \boldsymbol{h_g}(\boldsymbol{x})}{\partial \boldsymbol{q}_{\mathcal{NB}}} & \dfrac{\partial \boldsymbol{h_g}(\boldsymbol{x})}{\partial \boldsymbol{a}_b} & \boldsymbol{0}_{3\times 6} \end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \begin{bmatrix} \boldsymbol{0}_{3\times 6} & \dfrac{\partial \left(-\boldsymbol{R}\{\boldsymbol{q}_{\mathcal{NB}}\}^{\top} \boldsymbol{g}_{\mathcal{N}} \right)}{\partial \boldsymbol{q}_{\mathcal{NB}}} & \boldsymbol{I}_{3\times 3} & \boldsymbol{0}_{3\times 6} \end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \end{aligned} \tag{3.85}$$

where the remaining partial derivative can be written as:

$$\frac{\partial \left(-\boldsymbol{R}\{\boldsymbol{q}_{\mathcal{NB}}\}^{\top} \boldsymbol{g}_{\mathcal{N}} \right)}{\partial \boldsymbol{q}_{\mathcal{NB}}} = 2 \begin{bmatrix} q_2 & -q_3 & q_0 & -q_1 \\ -q_1 & -q_0 & -q_3 & -q_2 \\ -q_0 & q_1 & q_2 & -q_3 \end{bmatrix} \tag{3.86}$$

3.2.2.2 Magnetometer

The magnetometer can be used for attitude correction, too. The yaw bias and electromagnetic disturbance can be estimated under the assumption that the external magnetic field $\boldsymbol{m}_{\mathcal{N}}$ is known for the current location in the navigation frame, e.g. the Earth's magnetic field is measured before take-off and remains approximately constant throughout the flight. Errors and misdirection are estimated using the world magnetic model [103]. Constant magnetic disturbances close to the magnetometer are removed by calibration as described in Section 2.2.3.2.

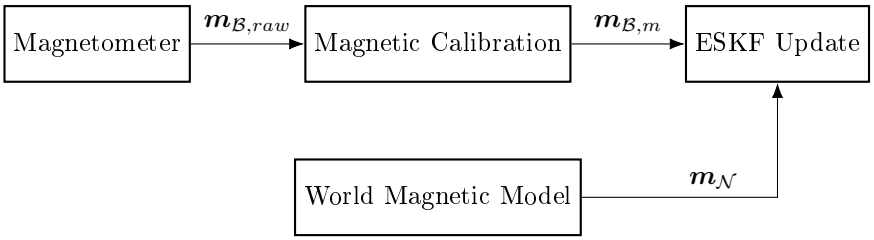


FIGURE 3.7: ESKF magnetometer integration.

The measurement model can be derived from Equation (3.41) as:

$$\begin{aligned} \boldsymbol{y}_{\boldsymbol{m}} &= \boldsymbol{h}_{\boldsymbol{m}}\left(\boldsymbol{x}\right) + \boldsymbol{m}_{w} \\ &= \boldsymbol{R}_{\mathcal{N}\mathcal{B}}{}^{\top}\boldsymbol{m}_{\mathcal{N}} + \boldsymbol{m}_{d} + \boldsymbol{m}_{w} \end{aligned} \tag{3.87}$$

The magnetometer measurement Jacobian is then given by:

$$\begin{aligned} \boldsymbol{H}_{\boldsymbol{m}} &= \left.\frac{\partial \boldsymbol{h}_{\boldsymbol{m}}\left(\boldsymbol{x}\right)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \left[\mathbf{0}_{3\times 6} \quad \frac{\partial \boldsymbol{h}_{\boldsymbol{m}}\left(\boldsymbol{x}\right)}{\partial \boldsymbol{q}_{\mathcal{N}\mathcal{B}}} \quad \mathbf{0}_{3\times 6} \quad \frac{\partial \boldsymbol{h}_{\boldsymbol{m}}\left(\boldsymbol{x}\right)}{\partial \boldsymbol{m}_{d}}\right]_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \left[\mathbf{0}_{3\times 6} \quad \frac{\partial\left(\boldsymbol{R}\left\{\boldsymbol{q}_{\mathcal{N}\mathcal{B}}\right\}^{\top}\boldsymbol{m}_{\mathcal{N}}\right)}{\partial \boldsymbol{q}_{\mathcal{N}\mathcal{B}}} \quad \mathbf{0}_{3\times 6} \quad \boldsymbol{I}_{3\times 3}\right]_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \end{aligned} \tag{3.88}$$

where the quaternion Jacobian can be evaluated with $\boldsymbol{m}_{\mathcal{N}} = \begin{bmatrix} m_n & m_e & m_d \end{bmatrix}^{\top}$ as:

$$\frac{\partial\left(\boldsymbol{R}\{\boldsymbol{q}_{\mathcal{N}\mathcal{B}}\}^{\top}\boldsymbol{m}_{\mathcal{N}}\right)}{\partial \boldsymbol{q}_{\mathcal{N}\mathcal{B}}} = 2\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}\left[\boldsymbol{q}_{\mathcal{N}\mathcal{B}}\right]_{L}^{\top}\begin{bmatrix} m_n & 0 & -m_d & m_e \\ 0 & m_n & m_e & m_d \\ -m_d & m_e & -m_n & 0 \\ m_e & m_d & 0 & -m_n \end{bmatrix} \tag{3.89}$$

3.2.2.3 Independent Tait-Bryan

If an additional Tait-Bryan reference is available, providing independent yaw, roll or pitch observations, it can be integrated into the ESKF, too, although the attitude is represented using quaternions. Direct Tait-Bryan observations can be provided for example by a GNSS attitude determination system, a sun sensor or an optical tracking system. The respective attitude Tait-Bryan angle $\mathcal{T} \in \{\phi, \theta, \psi\}$ can be predicted by extracting the required information from the estimated quaternion as:

$$\begin{aligned} y_{\mathcal{T}} &= \boldsymbol{h}_{\mathcal{T}}(\boldsymbol{x}) + \mathcal{T}_w \\ &= \mathcal{T}\{\boldsymbol{q}_{\mathcal{NB}}\} + \mathcal{T}_w \end{aligned} \tag{3.90}$$

where $\mathcal{T}_w$ is additive Gaussian noise with variance $\sigma_{\mathcal{T}}^2$ and zero mean and the respective attitude information can be obtained from the quaternion using the relations from Equation (3.29):

$$\phi\{\boldsymbol{q}_{\mathcal{NB}}\} = \operatorname{atan2}\big(\underbrace{2(q_0q_1 + q_2q_3)}_{\phi_y}, \underbrace{1 - 2\left({q_1}^2 + {q_2}^2\right)}_{\phi_x}\big) \tag{3.91}$$

$$\theta\{\boldsymbol{q}_{\mathcal{NB}}\} = \operatorname{asin}\big(\underbrace{2(q_0q_2 - q_1q_3)}_{\theta_x}\big) \tag{3.92}$$

$$\psi\{\boldsymbol{q}_{\mathcal{NB}}\} = \operatorname{atan2}\big(\underbrace{2(q_0q_3 + q_1q_2)}_{\psi_y}, \underbrace{1 - 2\left({q_2}^2 + {q_3}^2\right)}_{\psi_x}\big) \tag{3.93}$$

The respective measurement Jacobians can be calculated with:

$$\begin{aligned} \boldsymbol{H}_{\mathcal{T}} &= \left.\frac{\partial \boldsymbol{h}_{\mathcal{T}}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \begin{bmatrix} \mathbf{0}_{1\times 6} & \dfrac{\partial\left(\mathcal{T}\{\boldsymbol{q}_{\mathcal{NB}}\}\right)}{\partial \boldsymbol{q}_{\mathcal{NB}}} & \mathbf{0}_{1\times 9} \end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \end{aligned} \tag{3.94}$$

where the partial derivatives can be written as:

$$\frac{\partial\left(\phi\{\boldsymbol{q}_{\mathcal{NB}}\}\right)}{\partial \boldsymbol{q}_{\mathcal{NB}}} = \frac{2}{\phi_x^2 + \phi_y^2} \begin{bmatrix} q_1\phi_x \\ 2q_2\phi_y + q_0\phi_x \\ 2q_3\phi_y + q_4\phi_x \\ q_3\phi_x \end{bmatrix}^\top \tag{3.95}$$

$$\frac{\partial\left(\theta\{\boldsymbol{q}_{\mathcal{NB}}\}\right)}{\partial \boldsymbol{q}_{\mathcal{NB}}} = \frac{2}{\sqrt{1 - \theta_x^2}} \begin{bmatrix} q_2 \\ -q_3 \\ q_0 \\ -q_1 \end{bmatrix}^\top \tag{3.96}$$

$$\frac{\partial\left(\psi\{\boldsymbol{q}_{\mathcal{NB}}\}\right)}{\partial \boldsymbol{q}_{\mathcal{NB}}} = \frac{2}{\psi_x^2 + \psi_y^2} \begin{bmatrix} q_3\psi_x \\ q_2\psi_x \\ 2q_2\psi_y + q_1\psi_x \\ 2q_3\psi_y + q_0\psi_x \end{bmatrix}^\top \tag{3.97}$$

3.2.2.4 Direct State Observations

System states that can be directly observed, such as the position, relative height measurements, the velocity or the attitude represented by a quaternion can be integrated into the ESKF directly. It is important to note, that the measurements should be transformed to the respective frame of reference of the ESKF. Position and velocity observations are typically obtained by optical systems, such as an optical tracking or visual odometry systems, or by radio ranging methods based on GNSS or UWB. Direct quaternion observations can be provided using optical systems, too. The measurement equation reads simply:

$$\boldsymbol{y}_{\hat{\boldsymbol{x}}} = \boldsymbol{H}_{\hat{\boldsymbol{x}}}\left(\boldsymbol{x}\right) + \boldsymbol{\sigma}_{w} \tag{3.98}$$

where the entry of the respective state parameter in the observation matrix $\boldsymbol{H}_{\hat{\boldsymbol{x}}} \in \mathbb{R}^{n\times 19}$ is equal to one and n is the number of parameters that are directly observed. Common observation matrices are given in Table 3.3. The respective covariance matrices need to reflect the uncertainties of the applied reference system. It is assumed that all measurements have white Gaussian noise.

	Position	*Relative Height*	*Velocity*	*Quaternion*
$\boldsymbol{y}_{\hat{\boldsymbol{x}}}$	$\boldsymbol{y}_{\boldsymbol{p}} \in \mathbb{R}^3$	$y_{\Delta z} \in \mathbb{R}$	$\boldsymbol{y}_{\boldsymbol{v}} \in \mathbb{R}^3$	$\boldsymbol{y}_{\boldsymbol{q}} \in \mathbb{R}^4$
$\mathbf{H}$	$\begin{bmatrix}\boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times16}\end{bmatrix}$	$\begin{bmatrix}\mathbf{0}_{1\times2} & 1 & \mathbf{0}_{1\times16}\end{bmatrix}$	$\begin{bmatrix}\mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times13}\end{bmatrix}$	$\begin{bmatrix}\mathbf{0}_{4\times6} & \boldsymbol{I}_{4\times4} & \mathbf{0}_{4\times9}\end{bmatrix}$

TABLE 3.3: Observation matrices for directly observable states.

3.2.3 Delayed Measurements

The integration of real world complementary sensors is often linked to the time delayed availability of their measurements. Complex processing, such as visual pose estimation or the calculation of a carrier phase-based GNSS solutions, require time and cause a delay that needs to be compensated for in the sensor fusion algorithm. Additional delays resulting from the message transfer between an external processing system and the platform running the actual ESKF algorithm need to be considered, too. For direct EKF formulations, delayed measurements are extrapolated in time based on past and present state estimates [152, 153]. In this case, the error caused by extrapolation is compensated for by propagating and correcting state covariance estimates allowing to compute an optimal Kalman gain. For error based EKFs, the error state is augmented by means of stochastic cloning, in order to estimate propagation errors during the time delay. Hereby, the true time of the relative state measurement is indicated by a hardware signal in order to clone the corresponding error state [145].

The idea of extrapolating predictions from the direct filter approach is transferred to indirect filters. As hardware signals indicating new measurements are not available for all sensor types, a time horizon is introduced in form of a state history containing previously corrected system state estimates $\hat{\boldsymbol{x}}_{k|k}$ instead. Since in the indirect ESKF approach, the accuracies of the state estimates are unknown, instead of extrapolating the delayed measurements, they are compared to previous state estimates. Therefore, previous estimates are stored in the state history and are matched in time if new delayed measurements are received. Under the assumption that both, the error dynamics and the measurement delays are small, no further corrections are

required. In case of large measurement delays or very high error dynamics, additional delay uncertainty needs be considered, too. This should be done by increasing the observation uncertainty matrix $\boldsymbol{R}$ accordingly, or by estimating the uncertainty of the state propagation during the delay. Figure 3.8 illustrates the implemented approach.

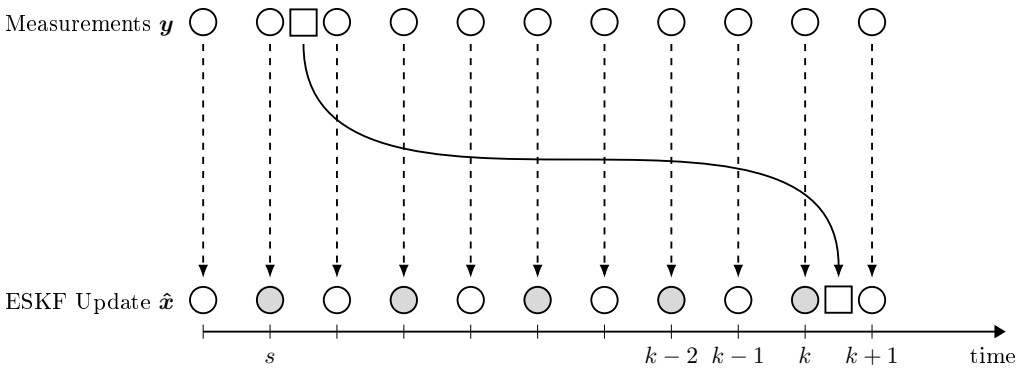


FIGURE 3.8: ESKF Integration of time delayed measurements. Inertial measurements (○), stored system states (◎) and complementary measurements (□).

Inertial measurements (○) are used to propagate the ESKF state $\hat{\boldsymbol{x}}$ at every time step k. In the depicted example, only every second state estimate is stored in the state history, indicated by (◎). Depending on the available system memory, the filter propagation rate and the expected time delay, the history size n and resolution r should be chosen suitably. A complementary sensor provides a measurement (□) that is received between the propagation times k and $k+1$. The true observed state, however, lies in the past. Based on the time stamp of the complementary measurement $t_{\boldsymbol{y}_d}$, the state estimate $\hat{\boldsymbol{x}}_{s|s}$ that is stored in the history and has a time stamp $t_{\hat{\boldsymbol{x}}_{s|s}}$ that closest is determined:

$$\hat{\boldsymbol{x}}_{s|s} = \underset{s \in \mathcal{H}}{\arg\min} \left[\left\| t_{\hat{\boldsymbol{x}}_{s|s}} - t_{\boldsymbol{y}_d} \right\|^2 \right] \tag{3.99}$$

where $\mathcal{H} = \{k, k-r, k-2r, \ldots, k-nr\}$. The delayed measurement is processed at time k, after the error state prediction. The time delayed measurement residual $\hat{\boldsymbol{z}}_{k+1}$ based on the delayed measurement $\boldsymbol{y}_d$ is then given by:

$$\hat{\boldsymbol{z}}_{k+1} = \boldsymbol{y}_d - \boldsymbol{h}\left(\hat{\boldsymbol{x}}_{s|s}\right) \tag{3.100}$$

3.3 Low-Level Control

Having established the ego-motion estimation algorithm, the low-level control of the proposed flight controller is described next. The goal of the low-level controller is to guarantee a stable and smooth UAV movement despite disturbances and system uncertainties, allowing the UAV to precisely follow high-level commands. The high-level commands are a subset of different system inputs, depending on the respective UAV task. Here, the high level inputs range from 3D positions $\boldsymbol{p} = \begin{bmatrix} x & y & z \end{bmatrix}$ and 3D velocities $\boldsymbol{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}$ to absolute attitude angles $\begin{bmatrix} \psi & \theta & \phi \end{bmatrix}$ or angular turning rates $\boldsymbol{\omega}_{\mathcal{B}}$. The concept is presented in Figure 3.9.

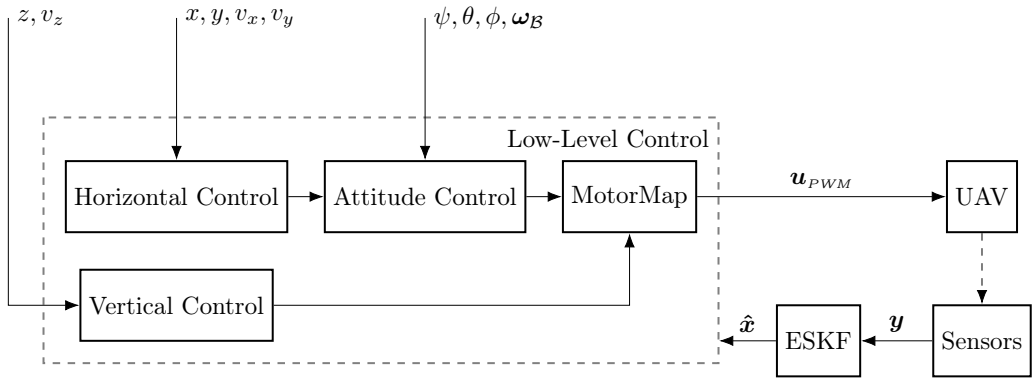


FIGURE 3.9: Low-level control concept.

In order to allow the low-level control scheme to be easily adapted for different multi rotor designs, a modular and cascaded control architecture is implemented. For each input domain and its respective degree of freedom, different controllers are utilized and cascaded if appropriate. The outputs of the final control stage for each rotational degree of freedom are merged using an explicit UAV propulsion model described by a so called motor map. The motor map is derived from the Newton-Euler kinematic equations of a rigid body applied to the geometry of a specific multi rotor. It should be noted, that the control architecture depicted above is limited to traditional, under-actuated multi rotors where the horizontal movement is controlled by the UAV's attitude. To be more specific, the complete control scheme applies only to UAV platforms with fixed rotor axes that are parallel and where gyroscopic torques cancel each other out. This is the case for any platform with a symmetric motor configuration where an equal amount of clockwise and counter-clockwise rotors is used and the rotor axes point upwards. Nevertheless, the attitude control scheme can be applied to the UAV platforms with arbitrary rotor directions and tilting rotors, too. For the translational control, however, the control scheme has to be adapted.

The remainder of this Chapter describes each controller of the cascaded approach illustrated in Figure 3.9, namely the attitude, horizontal and vertical control. Section 3.3.1 introduces the general multi rotor model and Section 3.3.2 derives the motor map for a quadcopter UAV in H-configuration. Sections 3.3.3 and 3.3.4 provide detailed insights and simulation results for the attitude and the position control, respectively.

3.3.1 Multicopter Modeling

A general multi rotor model can be derived using the Newton-Euler kinematics for rigid bodies. The total forces $\boldsymbol{f}_{\mathcal{C}} \in \mathbb{R}^3$ and torques $\boldsymbol{\tau}_{\mathcal{C}} \in \mathbb{R}^3$ acting on the multi rotor platform within a control reference frame $\mathcal{C}$ can be described as:

$$\begin{bmatrix} \boldsymbol{f}_{\mathcal{C}} \\ \boldsymbol{\tau}_{\mathcal{C}} \end{bmatrix} = \begin{bmatrix} m\boldsymbol{I}_{3\times 3} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{cm} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_{\mathcal{C}} \\ \dot{\boldsymbol{\omega}}_{\mathcal{C}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\omega}_{\mathcal{C}} \times \boldsymbol{I}_{cm}\boldsymbol{\omega}_{\mathcal{C}} \end{bmatrix} \tag{3.101}$$

where $\boldsymbol{a}_{\mathcal{C}} \in \mathbb{R}^3$ are the linear accelerations and $\boldsymbol{\omega}_{\mathcal{C}} \in \mathbb{R}^3$ are the angular rates in the control frame, $m \in \mathbb{R}_+$ is the mass of the UAV and $\boldsymbol{I}_{cm} \in \mathbb{R}^{3\times 3}$ moment of inertia. For any arbitrary multi rotor configuration with N fixed rotors that are displaced from the center of gravity by a displacement vector $\boldsymbol{l}_{\mathcal{C},i}$, the sum of the generated forces and torques can be expressed as:

$$\begin{bmatrix} \boldsymbol{f}_{\mathcal{C}} \\ \boldsymbol{\tau}_{\mathcal{C}} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} \boldsymbol{f}_i(u_i) \\ \sum_{i=1}^{N} \boldsymbol{l}_{\mathcal{C},i} \times \boldsymbol{f}_i(u_i) + \boldsymbol{\tau}_i(u_i) \end{bmatrix} \tag{3.102}$$

where $\boldsymbol{f}_i(u_i)$ models the thrust and $\boldsymbol{\tau}_i(u_i)$ models the torque for a certain control input u_i. The thrust and torque vector generated by each motor for a specific control command $u_{PWM,i}$ can be modeled as described in Section 2.1.2. Gyroscopic torques are considered to be small and are therefore neglected due to the low combined inertia of the motor/rotor pair and the fact that they are almost canceled out for multi rotors with fixed, parallel thrust vectors and an equal amount of left and right spinning rotors. Especially for tilt-rotor applications, however, gyroscopic torques should be considered.

Setting the general Newton-Euler approach equals to Equation (3.102) and solving for the linear and angular accelerations yields the complete multi rotor model as:

$$\begin{bmatrix} \boldsymbol{a}_{\mathcal{C}} \\ \dot{\boldsymbol{\omega}}_{\mathcal{C}} \end{bmatrix} = \begin{bmatrix} \frac{1}{m} \sum_{i=1}^{N} f_T(u_i) \cdot \boldsymbol{e}_i \\ \boldsymbol{I}_{cm}^{-1} \left(\sum_{i=1}^{N} \boldsymbol{l}_{\mathcal{C},i} \times f_T(u_i) \cdot \boldsymbol{e}_i - \operatorname{sgn}(\Omega_i) f_\tau(u_i) \cdot \boldsymbol{e}_i \right) \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I}_{cm}^{-1} \left(\boldsymbol{\omega}_{\mathcal{C}} \times \boldsymbol{I}_{cm}\boldsymbol{\omega}_{\mathcal{C}} \right) \end{bmatrix} \tag{3.103}$$

Equation (3.103), the discrete kinematics derived in Equation (3.53), the propulsion model derived in Section 2.1.2 as well as the mass properties of the utilized UAV platform in Section 2.1 can be used to model the complete UAV dynamics in MATLAB Simulink. The model is used to evaluate the proposed control architecture. The plots presented in the following sections result from this model.

3.3.2 Motor Map

The motor map or mixing matrix is used to map the forces and torques required by the respective controllers to the UAV geometry. The motor map can be obtained from Equation (3.102), however a simplified quadratic relation between motor control signal u_i and the thrust $f_T(u_i)$ and torque $f_\tau(u_i)$ generated by each motor is assumed:

$$f_T(u_i) \approx a_T u_i^2 \tag{3.104}$$

$$f_\tau(u_i) \approx a_\tau u_i^2 \tag{3.105}$$

Using the two simplifications above and neglecting the gyroscopic torques of Equation (3.102), the UAV kinematics can be expressed as the product of an allocation matrix $\boldsymbol{A}$ and the squared control signals $\boldsymbol{u}^2$ [154]:

$$\begin{aligned}\begin{bmatrix}\boldsymbol{f}_\mathcal{C}\\ \boldsymbol{\tau}_\mathcal{C}\end{bmatrix} &\approx \begin{bmatrix}\sum_{i=1}^{N} a_T u_i^2 \boldsymbol{e}_i \\ \sum_{i=1}^{N} \left(\boldsymbol{l}_{\mathcal{C},i} \times a_T u_i^2 \boldsymbol{e}_i + a_\tau u_i^2 \boldsymbol{e}_i\right)\end{bmatrix} \\ &= \underbrace{\begin{bmatrix}\dots & a_T \boldsymbol{e}_i & \dots \\ \dots & \boldsymbol{l}_{\mathcal{C},i} \times a_T \boldsymbol{e}_i + a_\tau \boldsymbol{e}_i & \dots\end{bmatrix}}_{\text{Allocation matrix } \boldsymbol{A}} \cdot \underbrace{\begin{bmatrix}\vdots \\ u_i^2 \\ \vdots\end{bmatrix}}_{\boldsymbol{u}^2}\end{aligned} \tag{3.106}$$

The allocation matrix $\boldsymbol{A} \in \mathbb{R}^{6\times N}$ describes the influence of each of the N actuators on the respective degree of freedom. If $\boldsymbol{A}$ has not full row rank, the UAV platform is under-actuated. The motor map $\boldsymbol{M}$ is defined as the inverse operation of the allocation matrix. However, since $\boldsymbol{A}$ is generally not a square matrix, the corresponding motor map $\boldsymbol{M}$ can be calculated using the pseudo-inverse of $\boldsymbol{A}$ assuming that the columns of $\boldsymbol{A}$ are linearly independent:

$$\boldsymbol{M} = \left(\boldsymbol{A}^\top \boldsymbol{A}\right)^{-1} \boldsymbol{A}^\top \tag{3.107}$$

In the following, the derivation of a valid motor map is exemplified for a quad rotor in H-configuration with a control frame of reference $\mathcal{C}$ as depicted in Figure 3.10. The center of gravity distance vectors for each actuator $\boldsymbol{l}_{\mathcal{C},i}$ and the motor rotational axes $\boldsymbol{e}_i$ are given as:

$$\boldsymbol{l}_{\mathcal{C},1} = \begin{bmatrix}l_x \\ -l_y \\ 0\end{bmatrix}, \quad \boldsymbol{l}_{\mathcal{C},2} = \begin{bmatrix}-l_x \\ -l_y \\ 0\end{bmatrix}, \quad \boldsymbol{l}_{\mathcal{C},3} = \begin{bmatrix}-l_x \\ l_y \\ 0\end{bmatrix}, \quad \boldsymbol{l}_{\mathcal{C},4} = \begin{bmatrix}l_x \\ l_y \\ 0\end{bmatrix} \tag{3.108}$$

$$\boldsymbol{e}_1 = \boldsymbol{e}_2 = \boldsymbol{e}_3 = \boldsymbol{e}_4 = \begin{bmatrix}0 \\ 0 \\ 1\end{bmatrix} \tag{3.109}$$

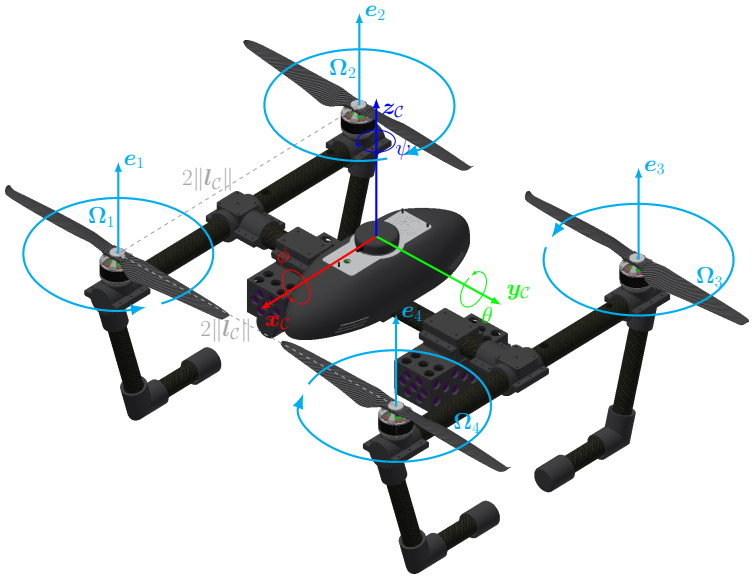


FIGURE 3.10: Actuators, displacement vector and control frame for a quad rotor UAV in H-configuration.

Using Equation (3.106), the allocation matrix $\boldsymbol{A}_{4H}$ can be derived as:

$$\boldsymbol{A}_{4H} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ a_T & a_T & a_T & a_T \\ -a_T l_y & -a_T l_y & a_T l_y & a_T l_y \\ -a_T l_x & a_T l_x & a_T l_x & -a_T l_x \\ -a_\tau & a_\tau & -a_\tau & a_\tau \end{bmatrix} \tag{3.110}$$

The pseudo-inverse of $\boldsymbol{A}_{4H}$ gives the motor map $\boldsymbol{M}_{4H}$ as:

$$\boldsymbol{M}_{4H} = \begin{bmatrix} 0 & 0 & \frac{1}{4a_T} & -\frac{1}{4a_T l_y} & -\frac{1}{4a_T l_x} & -\frac{1}{4a_\tau} \\ 0 & 0 & \frac{1}{4a_T} & -\frac{1}{4a_T l_y} & \frac{1}{4a_T l_x} & \frac{1}{4a_\tau} \\ 0 & 0 & \frac{1}{4a_T} & \frac{1}{4a_T l_y} & \frac{1}{4a_T l_x} & -\frac{1}{4a_\tau} \\ 0 & 0 & \frac{1}{4a_T} & \frac{1}{4a_T l_y} & -\frac{1}{4a_T l_x} & \frac{1}{4a_\tau} \end{bmatrix} \tag{3.111}$$

The control outputs can be modified in such a way that the constants of Equation (3.111) are already included into the required thrust and torques, e.g. $f_z \stackrel{!}{=} 4a_T u_{th}$. Thus, we obtain the very familiar control law as:

$$\boldsymbol{u}_{PWM} = \boldsymbol{M}_{4H} \begin{bmatrix} \boldsymbol{f}_C \\ \boldsymbol{\tau}_C \end{bmatrix} = \begin{bmatrix} u_{PWM,1} \\ u_{PWM,2} \\ u_{PWM,3} \\ u_{PWM,4} \end{bmatrix} = \begin{bmatrix} u_{th} - u_{\tau_x} - u_{\tau_y} - u_{\tau_z} \\ u_{th} - u_{\tau_x} + u_{\tau_y} + u_{\tau_z} \\ u_{th} + u_{\tau_x} + u_{\tau_y} - u_{\tau_z} \\ u_{th} + u_{\tau_x} - u_{\tau_y} + u_{\tau_z} \end{bmatrix} \tag{3.112}$$

3.3.3 Attitude Control

For each rotational degree of freedom, two cascaded controllers are used. The cascade for a single rotational degree of freedom is shown in Figure 3.11.

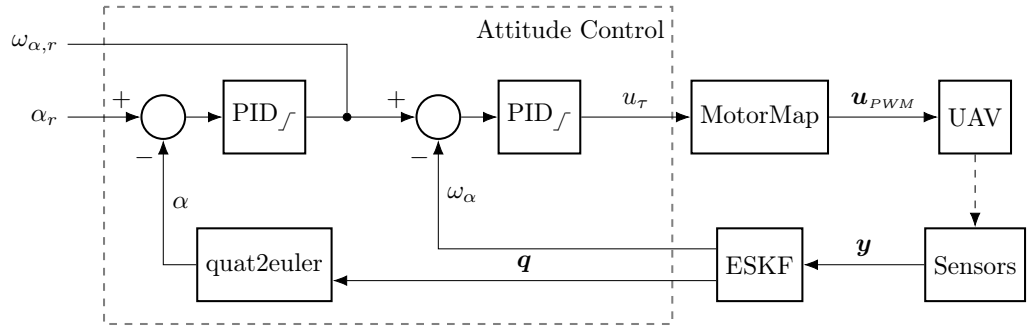


FIGURE 3.11: Cascaded attitude controller.

Each controller used within the cascade builds upon a generic Proportional-Integral-Derivative Controller (PID) with output limitations and anti-windup. For reference, the generic PID controller is shown in Figure 3.12. References and observations are given in the control frame of reference.

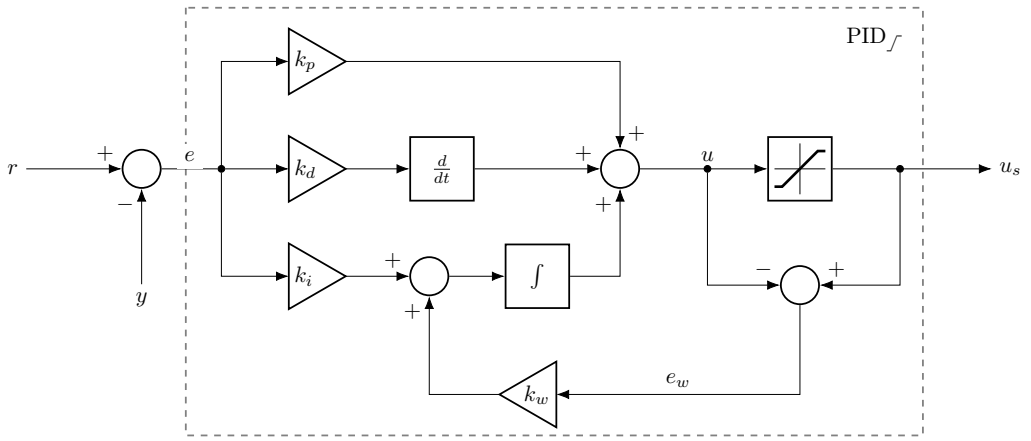


FIGURE 3.12: PID controller with output limitation and anti-windup.

The discrete integral is calculated using the trapezoidal rule, while the discrete time derivative is based on a low-pass filtered differential coefficient, if the respective derivative is not estimated by the ESKF, e.g. the angular acceleration:

$$\int_{t-\Delta t}^{t} e(t)dt \approx \Delta t \cdot \frac{e(t-\Delta t)+e(t)}{2} \tag{3.113}$$

$$\frac{\mathrm{d}e(t)}{\mathrm{d}t} \approx \gamma \frac{\mathrm{d}e(t-\Delta t)}{\mathrm{d}t} + (1-\gamma)\frac{e(t)-e(t-\Delta t)}{\Delta t} \tag{3.114}$$

where $\gamma \in [0;1]$ is a smoothing parameter and Δt is the sampling time. Since the remain follows the standard approach for discrete PID controllers, the interested reader is referred to standard literature for details such as [155, 156].

The inner controller regulates the angular velocity ω_α for the respective degree of freedom while the outer controller adjusts the angular velocity of reference $\omega_{\alpha,r}$ according to the current angular error $\alpha_e = \alpha_r - \alpha$. The inner controller calculates the required control torque around each axis u_τ that is forwarded to the motor map which in turn controls the actuators in order to provide the necessary torque. The integral gain of the inner control loop should be non-zero in order to compensate for center of gravity displacements and propulsion related differences, e.g. manufacturing differences, effects of aging as well as blade flapping. As an example for the control behavior of the cascaded attitude control, the roll step response for a step of 25° is shown in the figure below.

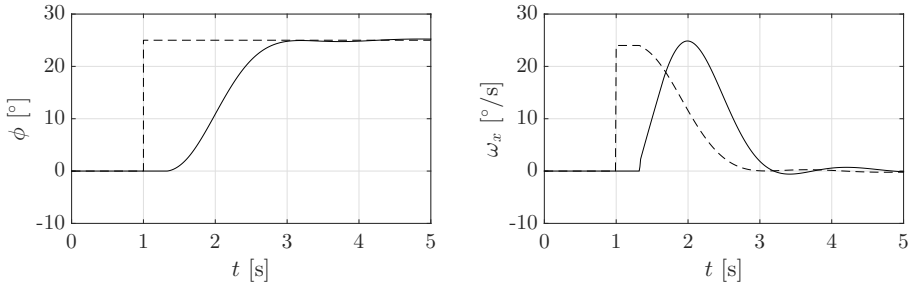


FIGURE 3.13: Simulated step response roll.

One advantage of the cascaded approach is that the outer controller can be bypassed by commanding a desired angular velocity directly, allowing an easy implementation of an ACRO/rate mode. Another advantage is the simplified tuning process. Once the inner control loop is able to keep track of a commanded angular velocity, the trade-off between a faster and a more robust response can be simply adjusted by adapting the outer proportional gain.

3.3.4 Position Control

Due to the under actuated nature of traditional multi rotor platforms, the position control is split into a vertical and horizontal control components.

3.3.4.1 Vertical Control

Similar to the attitude control, the vertical control is a cascade of two output limited PID controllers as shown in Figure 3.14.

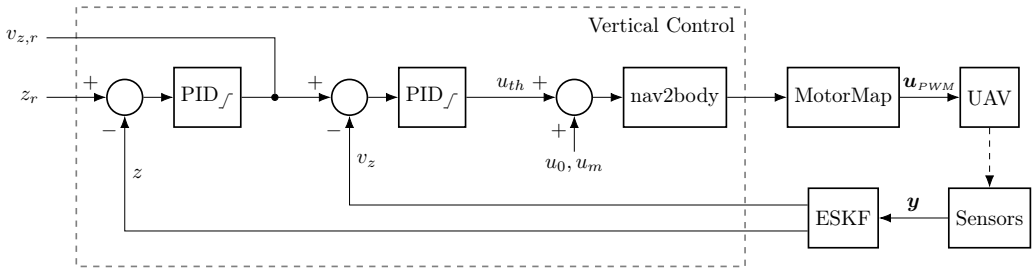


FIGURE 3.14: Cascaded vertical control.

For the vertical control, references and observations are given in the navigation frame. The outer loop establishes the reference speed v_r for the inner control loop and the maximum desired vertical speed can be controlled by the outer loop limits. The inner controller calculates the required thrust command u_{th} to achieve the desired speed. Two constants u_0 and u_m are added to the controller output in order to keep the rotors spinning at all time and to compensate for the UAV weight, respectively. The overall thrust needs to be converted to the control frame of reference in order to compensate for pitch and roll rotations. The advantages of the cascaded approach compared to a single height control loop are similar to the cascaded attitude control. The vertical speed has its own control loop, meaning that the outer controller can be bypassed if commanding the UAV in a velocity mode is desired. Again, tuning the aggression of the cascade is simply done by the proportional gain of the outer loop. Additional speed limitations allow to implement a controlled and very smooth flight behavior as shown for a simulated takeoff in Figure 3.15.

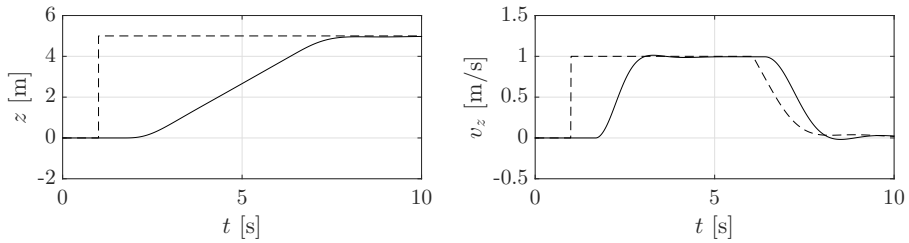


FIGURE 3.15: Simulated step response vertical control.

The UAV takes off slowly increasing its vertical speed until the limit set by the outer loop of 1 m/s is reached. Getting closer to the desired height of 5 m, the reference velocity is decreased and the UAV decelerates.

3.3.4.2 Horizontal Control

Since the horizontal movement of the under actuated system under consideration is linked to a change in the orientation, the horizontal control is a cascade of four generic PID controllers as depicted in Figure 3.16. The first stage of the cascade control determines the reference velocity, the second stage the reference roll and pitch angle which are subsequently used by the attitude control. Since references and observations are given in the navigation frame, the output of the velocity controller needs to be converted to the control frame. Hence, external disturbances, such as wind, can be compensated for by the integral part of the velocity control, while body fixed disturbances are compensated for by the attitude control.

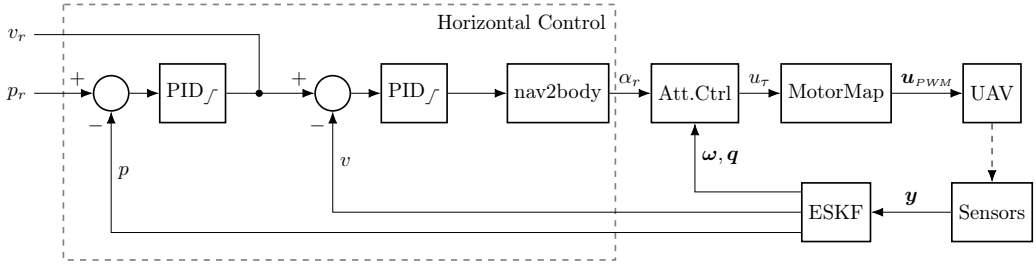


FIGURE 3.16: Cascaded horizontal controller.

Reference signals as well as the simulated behavior of each cascade for a single direction step input are shown in Figure 3.17. The same benefits arise for the cascaded horizontal control as for the other cascaded approaches.

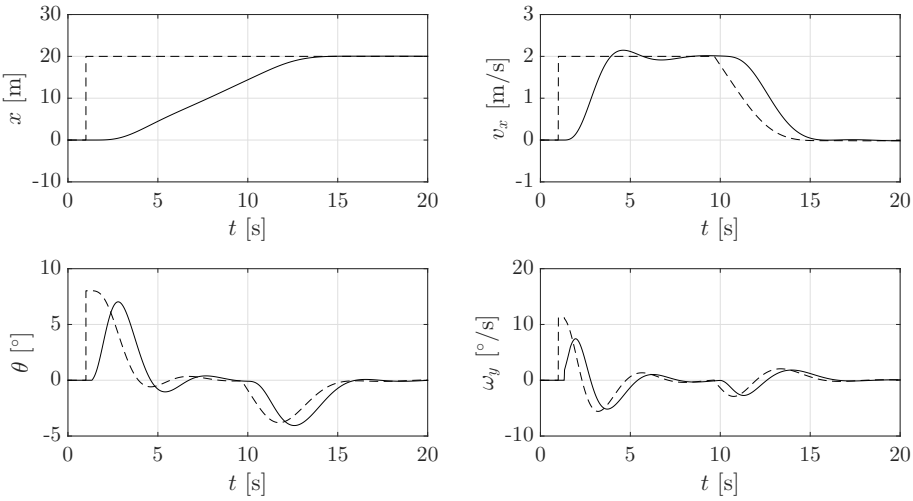


FIGURE 3.17: Simulated step response position.

It should be noted that the limitation of the reference velocity of the first stage of the cascade is dynamically adjusted according to the direction and the distance d of the reference position:

$$v_{x,lim} = v_{lim} \cdot \frac{\Delta x}{d} \quad \text{and} \quad v_{y,lim} = v_{lim} \cdot \frac{\Delta y}{d} \tag{3.115}$$

The different flight paths for a velocity limit with and without directional adjustment are shown in Figure 3.18.

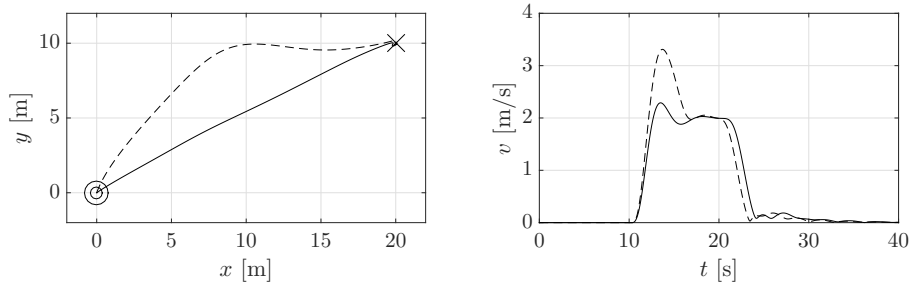


FIGURE 3.18: Simulated step response in with a simultaneous change in x and y. The solid line indicates the flight path with and the dashed line without adjusted velocity limitations.

Chapter 4

Satellite Navigation

This chapter introduces satellite navigation principles that are utilized within this thesis. The goal of satellite based navigation systems is to provide 3D positioning and timing services using satellite transmitted radio signals and passive receivers. Depending on their signal coverage and functionality, satellite navigation systems can be categorized into three different groups: Global Navigation Satellite System (GNSS), Regional Navigation Satellite Systems and Space-Based Augmentation System (SBAS). Current GNSSs are described in Section 4.1 and SBASs are described in Section 4.2. Existing Regional Navigation Satellite Systems such as the Japanese Quasi-Zenith Satellite System (QZSS) and the Indian Regional Navigation Satellite System (IRNSS) are not considered within this work. Section 4.3 presents GNSS observables and introduces their respective mathematical models. Different error sources of the respective observables are discussed in Section 4.4. In Section 4.5, different approaches for GNSS-based positioning as well as their benefits and limitations are described. The information presented within this Chapter is mainly based on [157–162]. For an in depth introduction, the interested reader is referred to the respective sources.

4.1 Global Navigation Satellite Systems

Global Navigation Satellite System (GNSS) is the collective term for satellite based navigation systems that provide global navigation services. All satellite systems share a common architecture consisting of three main components shown in Figure 4.1: the space segment, the control segment and the user segment.

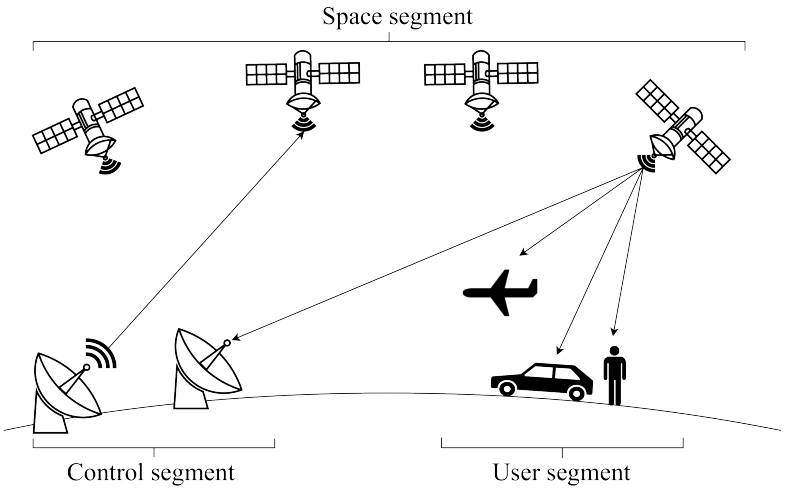


FIGURE 4.1: The GNSS architecture.

The space segment refers to the satellite constellation of each system that broadcasts radio signals on multiple carrier frequencies. The radio signals transmit ranging information and navigation data for each satellite. The transmitting satellite can be identified by using different channel access methods. A second communication channel allows the satellites to receive data uploaded from the control segment.

The control segment is a network of monitoring, control and uplink stations located on Earth. It provides orbital information to the space segment, functionality for satellite maintaining and station keeping as well as correction and calibration parameters to counteract different error sources.

The user segment can be loosely defined as different GNSS signal receivers. Depending on their application and precision requirements, receivers can be divided into different categories. High-cost multi-frequency receivers with sub-centimeter accuracy are used in military and avionic applications, while consumer grade receivers rely on single-frequency signals with an inaccuracy of a few meters.

By the time of writing, three GNSSs are fully operational: The American Global Positioning System (GPS), the Russian Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS) and the European Galileo system. The Chinese BeiDou (Compass) system is expected to be operational by 2020 but not considered in this work. Each of those systems provides free of charge open services to all users as well as restricted services that are only available to authorized users.

Constellation	*Nominal Number of Satellites*	*Operational Number of Satellites*	*Number of* Planes	*Period*	*Orbits per Sidereal Day*	*Inclination*	*Radius*
GPS	24	31	6	11 hr 58 min	2	55°	26,580 km
GLONASS	24	24	3	11 hr 15 min	2.125	64.8°	25,500 km
Galileo	27	22	3	14 hr 5 min	1.7	56°	29,620 km

TABLE 4.1: Orbit parameters for different GNSS constellations.

4.1.1 GPS

The Navigation Satellite Timing and Ranging GPS (NAVSTAR GPS), or simply GPS, was originally developed as a military navigation system by the U.S. government. The development started in 1978 and full operational capability was obtained in 1994. The GPS orbit parameters are listed in Table 4.1 and the 24 hour tracks for all currently operating GPS satellites are visualized in Figure 4.2.

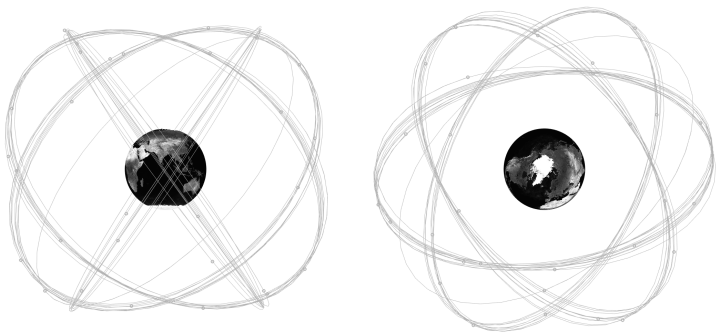


FIGURE 4.2: GPS constellation and orbital tracks for 24 hours.

Ten different navigation signals are broadcast across three frequency bands (L1: 1,575.42 MHz, L2: 1,227.60 MHz, L5: 1,176.45 MHz). Code Division Multiple Access (CDMA) is used for signal separation and illustrated in Figure 4.3. For CDMA, the navigation data is multiplied with a pseudo-random spreading code (PRN-Code) that is unique to each transmitter and known to the receivers. The mixed signal is than modulated onto the according carrier wave signal. The codes used for CDMA can be grouped into coarse acquisition code (C/A-code), encrypted precision code (P(Y)-code), civil codes (CM- & CL-code) and military code (M-code). In addition to the ranging navigation messages, ephemeris information expressed as 16 quasi-Keplerian orbit parameters, satellite clock calibration data, the almanac data for up to 32 satellites as well as the Klobuchar ionosphere propagation delay correction model for single-frequency users are transmitted.

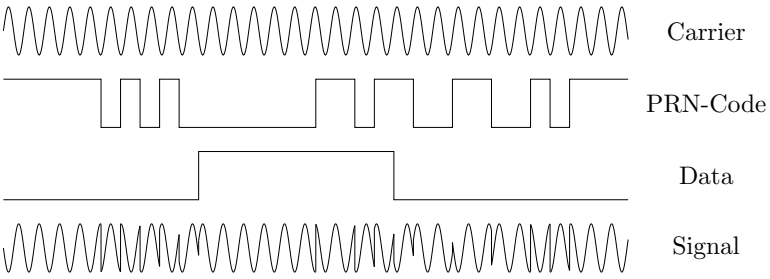


FIGURE 4.3: GPS signal modulation using CDMA.

4.1.2 GLONASS

Simultaneously to the development of the American GPS in the mid-1970s, Russia started to develop its own navigation system, the Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS). GLONASS achieved full operational capability in 1995 and provides civil as well as military signals. The GLONASS orbit parameters are listed in Table 4.1 and the 24 hour tracks for all currently operating GLONASS satellites are visualized in Figure 4.4. In contrast to GPS, GLONASS relied for signal separation initially on Frequency Division Multiple Access (FDMA) only, however, within the course of the modernization program starting in 2011, CDMA signals were added. The original FDMA signals are broadcast on 14 channels ranging from $1,598.0625 + 0.5625i$ MHz in the L1 band and $1,246.9375 + 0.4375i$ MHz in the L2 band, with $i \in [0, 13]$.

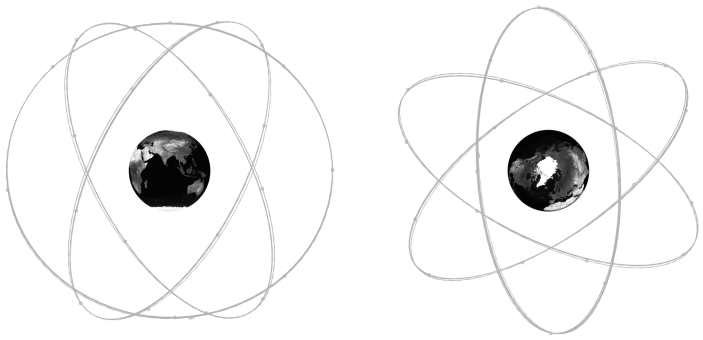


FIGURE 4.4: GLONASS constellation and orbital tracks for 24 hours.

4.1.3 Galileo

The development of the European GNSS Galileo was initiated in 1999 by the European Union and the European Space Agency. Full operational capability with 26 satellites was achieved in 2016. Galileo provides an open service (OS) available to all users and a public regulated service (PRS) reserved for emergency and security services as well as the military. The Galileo orbit parameters are listed in Table 4.1 and the 24 hour tracks for all currently operating Galileo satellites are visualized in Figure 4.5. In contrast to GLONASS, Galileo was specifically designed to be used alongside GPS rather than instead of it. The ten navigation signals broadcasted by Galileo use three different frequency bands E1, E5 and E6 as well as CDMA. The E5 signal uses a modified Binary Offset Carrier modulation with a 15 MHz offset, resulting in an E5a and an E5b signal (E1: 1,575.42 MHz, E5a: 1,176.43 MHz, E5b: 1,207.14 MHz, E6: 1,278.75 MHz). The Galileo E1 and the GPS L1 band as well as the Galileo E5a and the GPS L5 band use identical frequencies.

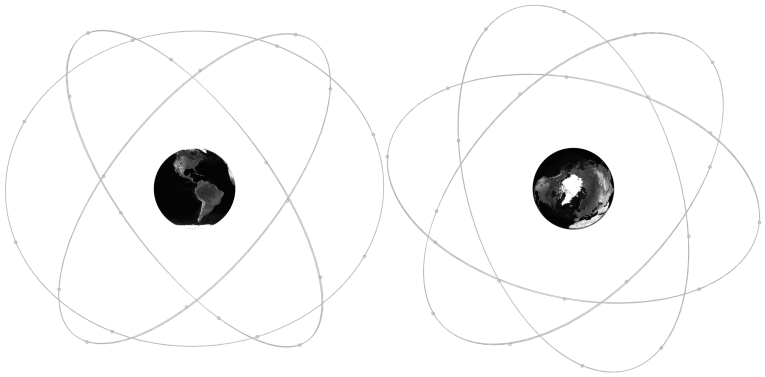


FIGURE 4.5: Galileo constellation and orbital tracks for 24 hours.

4.2 Space-Based Augmentation System

The Space-Based Augmentation System (SBAS) is a GNSS augmentation system that provides regional differential corrections and integrity alerts using geostationary satellites. Each SBAS uses a network of monitor stations across its coverage area to estimate clock and ephemeris corrections for each satellite as well as coefficients for the ionospheric model. The correction data is spread using a common signal with a unique PRN-code in the L1 frequency band. In total there are six SBASs currently under active development or already fully functional. The North-American Wide Area Augmentation System (WAAS) was the first SBAS and gained full operational capability in 2000, while the European equivalent European Geostationary Navigation Overlay System (EGNOS) broadcasts correction data since 2006. Both systems broadcast only GPS correction data, however, additional corrections for Galileo using EGNOS are under consideration. Other SBASs exist for Japan, India, Russia and China but are not relevant within this work.

4.3 Observables

The typical GNSS user equipment consists of four components: The antenna, the receiver, the ranging processor and the navigation processor.

The antennas have a peak sensitivity close to the carrier frequency of the desired signals. Without phase center calibration, the navigation solution is not necessarily valid for the physical center of the antenna but rather its electrical phase center. In contrast to hand-held devices with highly integrated antennas and receiver front ends, cables are used on vehicles between the antenna and the receiver front end to allow an optimal antenna placement. A common-mode lag is introduced on the incoming signal by the cable which is in the same magnitude as the receiver clock offset and therefore not considered separately.

The receiver digitizes the incoming signal and forwards its measurements to the ranging processor. Depending on the quality of the reference oscillator controlling the receiver, different receiver categories are distinguished. In low-cost applications, temperature-compensated crystal oscillators (TCXO) with a clock drift of a few hundred milliseconds per day are used. Advanced high precision receivers, on the other hand, rely on oven-controlled crystal oscillators (OCXO) or atomic clocks.

The ranging processor provides three primary raw measurements, namely the pseudo-range, the pseudo-range rate or Doppler measurement and the carrier phase measurement. The pseudo-range is obtained by correlating the received GNSS signal with a receiver generated code, the Doppler shift can be directly measured and allows to calculate the pseudo-range rate while the carrier phase is measured by shifting the receiver-generated phase to track the received phase.

Finally, the raw measurements are processed using a navigation processor. State of the art uBlox GNSS modules provide the functionalities of combined receiver, ranging and navigation processor. Moreover, modules from the uBlox timing series allow to access the ranging processors raw measurements and process them using advanced navigation algorithms, such as carrier phase-based or precise point positioning. The GNSS observables are modeled and discussed in the following sections with reference to [161, 162].

4.3.1 Code Pseudo-ranges

The code pseudo-range is an approximation for the distance between a navigation satellite and a GNSS receiver. It is calculated using the Time of Arrival (TOA) at the receiver t_r of a particular radio signal feature that has a transmission time t^s estimated by the transmitter. The error-prone pseudo-range $P^s_{r,i}(t)$ is then given by:

$$P^s_{r,i}(t) = (t_r - t^s)\, c \tag{4.1}$$

where the superscript s indicates the transmitting satellite, the subscript r the utilized receiver, the subscript i the carrier frequency band and c the speed of light. For a known satellite position $\boldsymbol{r}^s\,(t^s) = (x^s, y^s, z^s)$ and a receiver position $\boldsymbol{r}_r\,(t_r) = (x_r, y_r, z_r)$ both given in the ECEF frame, the geometric range $\rho^s_{P,r,i}$ between a satellite s and a receiver r in the ECI frame can be expressed as:

$$\rho^s_{r,i}\,(t_r, t^s) = \|\boldsymbol{r}^s\,(t^s) - \boldsymbol{r}_r\,(t_r)\| + \frac{\omega_e}{c}\,(x^s y_r - y^s x_r) \tag{4.2}$$

where ω_e is the rotational speed of the Earth and the additive term is called Sagnac effect [163]. The accuracy of the pseudo-range measurement is affected by a number

of errors caused by the transmitter, the receiver and by the transmitting media. It can therefore be expressed as the sum of the geometric satellite-receiver range and different error terms as:

$$\begin{aligned} P^s_{r,i}(t_r) = \rho^s_{r,i}(t_r, t^s) + I^s_{r,i} + T^s_r + dm^s_{r,i} + c\left[d_{r,i}(t_r) + d^s_i(t^s)\right] \\ + c\left[dt_r(t_r) - dt^s(t^s)\right] + \epsilon^s_{P,r,i} \end{aligned} \tag{4.3}$$

where the different terms are:

$P^s_{r,i}$	code observation for satellite s, receiver r and frequency band i [m]
$\rho^s_{r,i}$	geometrical distance between receiver r and satellite s [m]
t_r	TOA time at the receiver [s]
t^s	transmission time at the satellite [s]
$I^s_{r,i}$	frequency depended ionospheric delay [m]
T^s_r	tropospheric delay [m]
$dm^s_{r,i}$	code multipath error [m]
$d_{r,i}, d^s_i,$	receiver and satellite instrumental delays [s]
c	speed of light [m/s]
dt_r, dt^s	receiver and satellite clock errors [s]
$\epsilon^s_{P,r,i}$	unmodeled code errors [m]

The atmospheric effects on the signal are described by the correction terms I and T for the ionosphere and the troposphere, respectively. The term dm models multipath errors that are introduced by signal replicas due to reflections. The clock errors dt and the instrumental delays d are satellite and receiver hardware depended.

4.3.2 Carrier Phase Measurement

The carrier phase measurement is a *"measurement on the beat frequency between the received carrier of the satellite signal and a receiver-generated reference frequency [164]"* . The carrier phase measurement $\varphi^s_{r,i}$ can be modeled as:

$$\begin{aligned} \varphi^s_{r,i} &= \varphi_{r,i}(t_r) - \varphi^s_i(t^s) + N^s_{r,i} + \epsilon_{\varphi,r,i} \\ &= \left[f \cdot (t_r + dt_r(t_r) - t_0) + \varphi_{r,i}(t_0)\right] \\ &\quad - \left[f \cdot (t^s + dt^s(t^s) - t_0) + \varphi^s_i(t_0)\right] + N^s_{r,i} + \epsilon^s_{\varphi,r,i} \\ &= \frac{c}{\lambda_i}(t_r - t^s) + \frac{c}{\lambda_i}\left[dt_r(t_r) - dt^s(t^s)\right] \\ &\quad + \left[\varphi_{r,i}(t_0) - \varphi^s_i(t_0)\right] + N^s_{r,i} + \epsilon^s_{\varphi,r,i} \end{aligned} \tag{4.4}$$

with the following terms:

$\varphi_{r,i}$	phase of the receiver's local oscillator at frequency band i [cycles]
φ^s_i	phase of transmitted signal at frequency band i [cycles]
t_r	TOA time at the receiver [s]
t^s	transmission time at the satellite [s]
$N^s_{r,i}$	carrier phase integer ambiguity [cycles]
f	carrier frequency [Hz]
dt_r, dt^s	receiver and satellite clock errors [s]
t_0	reference time for phase synchronization [s]
$\epsilon^s_{\varphi,r,i}$	unmodeled phase errors [cycles]
λ_i	carrier wavelength at frequency band i [m]

If the carrier phase measurement is given in meters, it is referred to as phase-range measurement. The carrier phase measurement can be simply converted by multiplying with the carrier wavelength λ_i:

$$\begin{aligned}\Phi^s_{r,i}(t_r) &= \lambda_i \varphi^s_{r,i} \\ &= c\,(t_r - t^s) + c\,[dt_r(t_r) - dt^s\,(t^s)] \\ &\quad + \lambda_i\,[\varphi_{r,i}\,(t_0) - \varphi^s_i\,(t_0)] + \lambda_i N^s_{r,i} + \lambda_i \epsilon^s_{\varphi,r,i}\end{aligned} \tag{4.5}$$

Similar to the pseudo-range in Equation (4.1), the term $c\,(t_r - t^s)$ is error prone in case of phase-range measurements, too. A more detailed model for the phase-range $\Phi^s_{r,i}(t_r)$ is therefore given by:

$$\begin{aligned}\Phi^s_{r,i}(t_r) = \rho^s_{r,i}\,(t_r, t^s) - I^s_{r,i} + T^s_r + \delta m^s_{r,i} + c\,[\delta_{r,i}(t_r) + \delta^s_i(t^s)] \\ + c\,[dt_r(t_r) - dt^s\,(t^s)] + \lambda_i\,[\varphi_{r,i}\,(t_0) - \varphi^s_i\,(t_0)] + \lambda_i N^s_{r,i} + \epsilon^s_{\Phi,r,i}\end{aligned} \tag{4.6}$$

where the different terms are given as:

$\Phi^s_{r,i}$	phase range observation for satellite s, receiver r and frequency band i [m]
$\rho^s_{r,i}$	geometrical distance between receiver r and satellite s [m]
$I^s_{r,i}$	frequency depended ionospheric delay [m]
T^s_r	tropospheric delay [m]
$\delta m^s_{r,i}$	carrier multipath error [m]
$\delta_{r,i}, \delta^s_i$	carrier receiver and satellite instrumental delays [s]
dt_r, dt^s	receiver and satellite clock errors [s]
λ_i	carrier wavelength at frequency band i [m]
φ	phase of generated carrier signal [cycles]
t_0	reference time for phase synchronization [s]
$N^s_{r,i}$	carrier phase integer ambiguity [cycles]
$\epsilon^s_{\Phi,r,i}$	unmodeled phase errors [m]

The error terms are similar to the pseudo-range model in Equation (4.3). Since the ionosphere is a dispersive media, the ionospheric delay has different signs for phase-range and pseudo-range measurements. A radio wave propagating through the ionosphere experiences a propagation delay and a phase advance [165]. An additional parameter in the phase-range model is the carrier phase integer ambiguity $N^s_{r,i}$ which describes the unknown number of complete carrier phase cycles between signal transmission and phase measurement. The integer ambiguity $N^s_{r,i}$ and the initial phases of the satellite $\varphi^s_f\,(t_0)$ and the receiver $\varphi_{r,i}\,(t_0)$ can be condensed as the carrier phase bias:

$$B^s_{r,i} = \varphi_{r,i}\,(t_0) - \varphi^s_i\,(t_0) + N^s_{r,i} \tag{4.7}$$

4.3.3 Doppler Measurement

The Doppler-shift measurement is byproduct obtained from the receiver's carrier tracking loop. The phase change caused by the relative motion between satellite and receiver is predicted during the tracking process and used to adjust the carrier tracking loop. The frequency observed at the receiver f_r is given by:

$$f_r = \left(\frac{c + v_r}{c + v^s}\right) f^s \tag{4.8}$$

where v_r and v^s are the speeds of the receiver and the satellite, respectively, c is the speed of light and f^s is the carrier wave frequency generated by the satellite s. Since the receiver and satellite speed are small compared to the speed of the carrier wave, the Doppler-shift $f^s_{D,r}$ can be approximated by:

$$\Delta f^s_{D,r} \approx -f^s \frac{\partial \left(t_r - t^s\right)}{\partial t} \tag{4.9}$$

The Doppler-shift can be used to estimate the pseudo-range rate:

$$\dot{P}^s_{r,i} \approx -\lambda_i \Delta f^s_{D,r} \tag{4.10}$$

4.4 Error Sources

The navigation signal is distorted by different errors which can be categorized by their origin. Possible error sources are atmospheric effects, the receiver hardware and signal reflections close to the receiver hardware as well as the satellite clock and its orbit. Depending on the applied navigation technique, the utilized receiver hardware and its environment, different error sources can be eliminated using differential techniques or are small enough to be neglected. Hereinafter, the different error sources are explained and it is shown how the errors can be mitigated within the special context of precise low-cost navigation for UAVs. The GNSS errors are modeled and discussed in the following sections with reference to [157].

4.4.1 Atmospheric Propagation Errors

The satellite transmitted radio signal is disturbed by particles in the atmosphere of the Earth. Signal refraction occurs especially in the troposphere and the ionosphere. The length of the signal path through the atmosphere varies, depending on the satellite elevation. The relation between the satellite elevation θ^s_r and the introduced delay by the troposphere T^s_r is approximated by:

$$T^s_r \propto \sqrt{1 - \left(\frac{\cos \theta^s_r}{1.001}\right)^2} \tag{4.11}$$

For the ionosphere, the influence on the signal delay $I^s_{r,i}$ based on the satellite elevation is roughly described by the following relationship:

$$I^s_{r,i} \propto \sqrt{1 - \left(\frac{R \cos \theta^s_r}{R + h_i}\right)^2} \tag{4.12}$$

where R is the average Earth radius and h_i is the mean ionospheric height, about 350 km. From Equations (4.11) and (4.12) it can be seen, that a lower satellite elevation results in a larger error term. This is due to the longer signal path through the respective atmospheric layer and therefore larger signal refraction. To counteract large delays caused by low satellite elevations, satellites with an elevation below a certain threshold are typically excluded from the navigation solution using an elevation mask. Depending on the application, a suitable elevation mask is applied taking the satellite Signal-to-Noise-Ratios (SNRs) and their elevation angles into account. In order to remove the remaining delay terms added by troposphere and ionosphere, different models are established to estimate the respective error magnitudes.

4.4.1.1 Troposphere

The troposphere is the lowest layer of the Earth's atmosphere and extends to a height of about 15 km. For signals at GNSS frequencies, the troposphere is non-dispersive and causes a modulation and phase delay, compared to the free space propagation of a signal. 90% of the actual tropospheric delay is caused by dry gases and is relatively stable. The remaining delay is due to water vapor and varies considerably. A rather accurate model to describe the influence of the troposphere was introduced by Saastamoinen [166, 167]:

$$T_r^s = \frac{0.002277}{\sin\theta_r^s}\left[p + \left(\frac{1255}{T}\right)e - \tan^2\left(\frac{\pi}{2} - \theta_r^s\right)\right] \tag{4.13}$$

where p is the total pressure [hPa], T is the absolute temperature of the air [K] and e is the partial pressure of the water vapor [hPa]. The three variables can be calculated using the standard atmospheric model with:

$$p = 1013.25 \cdot \left(1 - 2.2557 \cdot 10^{-5} h\right)^{5.2568} \tag{4.14}$$

$$T = 15.0 - 6.5 \cdot 10^{-3} h + 273.15 \tag{4.15}$$

$$e = 6.108 \cdot \exp\left(\frac{17.15T - 4684.0}{T - 38.45}\right) \cdot \frac{h_{rel}}{100} \tag{4.16}$$

where h is the geodetic height above the mean sea level and h_{rel} is the relative humidity [%]. If SBAS information is available, more precise tropospheric models can be used [168]. The typical tropospheric range error for satellite at zenith is about 2.5 m.

4.4.1.2 Ionosphere

The ionosphere is a layer of the Earth's atmosphere that extends from 60 km to up to 1000 km and consists mostly of ionized gases and free electrons. Since the ionization is caused by solar radiation, the amount of ionized gases depends on the time of day and the solar cycle. In contrast to the troposphere, the ionosphere is a dispersive medium at GNSS frequencies that causes a group delay and a phase advance. This code-carrier divergence is indicated with different signs for the error term $I_{r,i}^s$ in Equation (4.3) and (4.6).

In case of single frequency GNSS receiver, the ionospheric delay needs to be estimated. Similar to the tropospheric error, different models can be applied in order to estimate the ionospheric delay. The Klobuchar model assumes a single thin layer at 350 km height and estimates the ionospheric delay depending on the carrier frequency and the location-dependent total electron content along the signal path through the ionosphere [169]. The delay can be approximated by:

$$I_{r,i}^s \approx \frac{40.3}{f_i^2}\text{sTEC} \tag{4.17}$$

where f_i is the carrier signal frequency and sTEC is the slant total electron content [$10^{16}\text{e}^-/\text{m}^2$] which describes the *"number of free electrons in a column through the ionosphere with a cross-sectional area of one square meter [170]"*. The parameters required to calculate the sTEC according to the Klobuchar model are broadcast by GPS. While there are no corrections for ionospheric errors broadcast by GLONASS, the Galileo system transmits parameters required for the NeQuick-G model. NeQuick-G is an

adaption by European Space Agency (ESA) of the original NeQuick model introduced by Di Gionvanni and Radicella [171]. According to [172], both, the Klobuchar and the NeQuich-G model get comparable performance in the position domain and reduce the ionospheric RMSE of about 50%.

Dual- or multi-frequency receivers, on the other hand, can eliminate the effect of the ionosphere through a linear combination (LC) of pseudo-range and and range-phase measurements:

$$P^s_{r,LC} = \frac{f_1^2 \cdot P^s_{r,1} - f_2^2 \cdot P^s_{r,2}}{f_1^2 - f_2^2} \tag{4.18}$$

$$\Phi^s_{r,LC} = \frac{f_1^2 \cdot \Phi^s_{r,1} - f_2^2 \cdot \Phi^s_{r,2}}{f_1^2 - f_2^2} \tag{4.19}$$

Using the ionosphere-free combination, first order error terms are canceled out, however, higher order terms with an error range of up to a few centimeters are not considered [173, 174]. If scintillations are present, the ionosphere-free linear combination of L1 and L2 phase residuals can reach up to 2-3 meters [175].

For low-cost single-frequency receivers in a differential GNSS setup with a short baseline length ($<$ 10 km) between rover and base-station, it is assumed that the GNSS signals travel approximately along the same path through the Earth's atmosphere to both receivers. The tropospheric and ionospheric effects are therefore almost eliminated using double-differences. The double-difference approach is introduced in Section 4.5.3.

4.4.2 Receiver and Multipath Errors

Another error source is the user hardware itself and its operational environment. Low-cost user equipment for embedded applications with single frequency receivers suffers from poor signal quality and high thermal noise resulting in inaccuracies of up to several meters, while high-end geodetic receivers allow for millimeter positioning. Despite the utilized hardware, signal reflections caused by the receiver surrounding environment, such as nearby buildings, trees or mountains, considerably impair the quality of the navigation solution.

4.4.2.1 Receiver Errors

All GNSS receivers introduce error terms due to clock offsets dt_r, frequency dependent instrumental delays $d_{r,i}$ and signal transmission lags caused by antenna-receiver cables. In practice, the signal transmission lag from the cable is so small, that it can not be distinguished from the clock offset itself. For a fixed carrier-wave frequency, clock offsets and instrumental delays can be eliminated by comparing the TOA of signals from two satellites with known transmission times. However, GLONASS uses FDMA to make the signals from individual satellites distinguishable. Therefore, additional frequency dependent instrumental delays between different satellite channels, so-called inter-frequency biases need to be considered[176].

Low-cost GNSS receivers, such as receivers with temperature-compensated crystal oscillators (TCXO), suffer from frequency drifts and thermal noise. Typically, long term errors and drift are compensated for by the navigation processor [157, 160]. Thermal noise needs to be addressed by stochastic means. Based on the utilized antenna grade, the physical center and the phase center do not necessarily coincide

and might therefore, depending on the user requirements, demand a calibration of the phase center offset.

4.4.2.2 Multipath Errors

Multipath errors are caused by the simultaneous reception of a signal directly transmitted from the satellite to the receiver and signals which have been reflected by the presence of surrounding objects as shown in Figure 4.6.

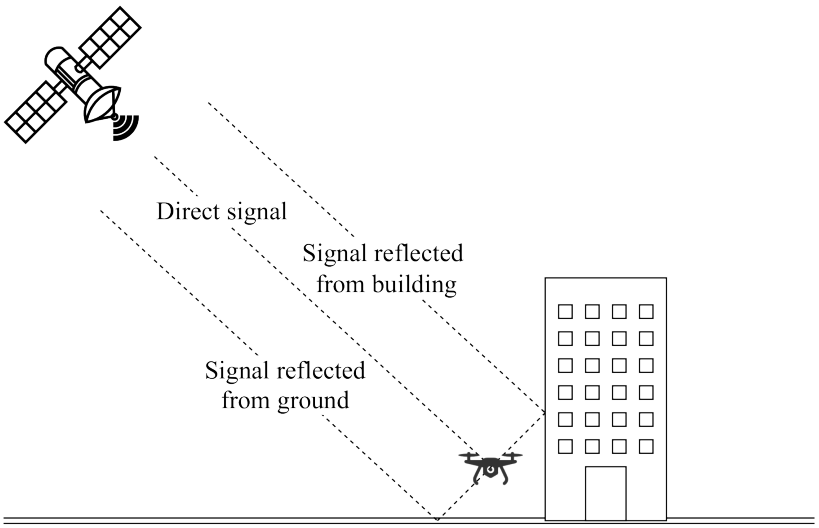


FIGURE 4.6: Simultaneous reception of direct and reflected signals.

The simultaneous reception of both, direct and reflected signals, causes an error in the tracking loop of the receiver hardware. Both, code and carrier phase observation are affected by multipath. The error caused by code multipath ranges typically between 1 and 5 meters, while the carrier phase multipath error is usually just a few centimeters. Within this work, it is assumed that signal reflections from buildings and trees are negligible for UAVs in typical open sky scenarios. One way to limit the multipath error is to physically shield the signal reflections due to ground proximity. According to [177], for low-cost antennas, this can be easily achieved by using ground plates as shown in Figure 4.7. However, multipath mitigation is not limited to shielding reflected signals and optimizing antenna characteristics. Another approach exploits the temporal correlation of the code multipath in order to estimate the error [178].

(A) UAV with GNSS compass. (B) UAV with single GNSS receiver.

FIGURE 4.7: Ground multipath mitigation through ground plates.

4.4.3 Ephemeris and Satellite Clock Errors

In addition to the errors caused by the signal propagation through various layers of the atmosphere, the receiving unit and signal reflections by surrounding objects, the precision used to describe characteristics of the transmitting satellites is another error source. Both, the ephemeris data describing the satellite orbit as well as the estimated satellite clock inaccuracies introduce additional errors. For applications with standard precision requirements, orbit and clock error corrections are broadcast by the space segment of each GNSS as part of the navigation message. The combined error due to satellite broadcast orbit and clock errors is called Signal-in-space Ranging Error (SISRE) [179]. Table 4.2 lists the global average SISRE for different GNSSs.

Constellation	**GPS**	**GLONASS**	**Galileo**
SISRE [m]	0.7 ± 0.02	1.9 ± 0.1	1.6 ± 0.3

TABLE 4.2: Global average Signal-in-space Ranging Error for different Global Navigation Satellite Systems [180].

4.4.3.1 Ephemeris Errors

The precision of the satellite orbit determination is limited by insufficient knowledge of the forces acting on the spacecraft, such as the solar radiation pressure, errors of the gravitational field model and uncertainties about the initial state [181]. Standard precision ephemeris data is broadcast by each GNSS satellite, while precise orbit data is provided by International GNSS Service (IGS) centers. GPS and Galileo broadcast the orbital parameters as a set of 16 quasi-Keplerian elements, while the GLONASS navigation message contains the initial spacecraft position and velocity as well as gravitational effects caused by the Moon and the Sun. For each GNSS, the satellite orbits are tracked from Earth and the respective ephemeris data is periodically updated so that the broadcast navigation messages are usually valid for up to one hour. Precise orbit data is distributed using the Networked Transport of RTCM via Internet Protocol (NTRIP) [182], a protocol for streaming GNSS correction data specified by the Radio Technical Commission for Maritime Services (RTCM) [183].

4.4.3.2 Satellite Clock Errors

Although the atomic clocks of GNSS satellites are very accurate, relativistic effects, clock drift and offset need to be compensated for. Therefore, the GNSS satellite clock error dt^s can be modeled as:

$$dt^s(t) = \tilde{dt}^s(t) + \Delta_{rel}(t) \tag{4.20}$$

where $\tilde{dt}^s(t)$ is a polynomial function describing the clock offset based on broadcast parameters or precise clock error estimates provided by IGS centers and $\Delta_{rel}(t)$ is a small relativistic correction [184]. In case of GPS or Galileo, a second order polynomial function is used to model the clock offset, while for GLONASS only a first order polynomial is considered. Broadcast clock errors provide an accuracy in the order of nanoseconds, while IGS services provide an accuracy at the order of 100 picoseconds.

4.5 Navigation Techniques

Depending on the required accuracy and precision of the final GNSS navigation solution, the utilized GNSS receivers and the available budget, different navigation techniques can be applied. Almost all state of the art consumer-grade GNSS receivers offer single epoch positioning algorithms (Section 4.5.1) and filtered navigation solutions (Section 4.5.2). More advanced, still near-consumer-grade hardware provides raw measurement data to allow the use of carrier phase observables in order to apply RTK GNSS algorithms with centimeter level accuracy (Section 4.5.3). Most recent GNSS receivers implement RTK solution directly. In case of multi-band GNSS receivers, PPP can be applied in order to combine several GNSS refinement techniques yielding near-survey-grade results (Section 4.5.4).

4.5.1 Single Epoch Navigation

The single epoch navigation solution performs a weighted non-linear Least Square Estimation (LSE) on an epoch-by-epoch basis. The non-linear LSE is solved using the Levenberg-Marquardt method [87]. Assuming an initial state $\hat{\boldsymbol{x}}_0$ a solution for the non-linear LSE can be found by iteratively converging the following expression:

$$\hat{\boldsymbol{x}}_{j+1} = \hat{\boldsymbol{x}}_j + \left(\boldsymbol{H}^\top \boldsymbol{W} \boldsymbol{H}\right)^{-1} \boldsymbol{H}^\top \boldsymbol{W} \left(\boldsymbol{y} - \boldsymbol{h}(\hat{\boldsymbol{x}}_j)\right) \tag{4.21}$$

where $\hat{\boldsymbol{x}}_j$ is the state estimate at iteration j, $\boldsymbol{y}$ is the observation vector, $h(\hat{\boldsymbol{x}}_j)$ is the measurement equation based on the current state estimate, $\boldsymbol{W}$ is a measurement weighing matrix and $\boldsymbol{H}$ is the Jacobian of the measurement equation. Formally, the solution $\hat{\boldsymbol{x}}$ of the non-linear LSE is defined as:

$$\hat{\boldsymbol{x}} = \lim_{j \to \infty} \hat{\boldsymbol{x}}_j \tag{4.22}$$

If code pseudo-ranges and Doppler-shift observables are available, both, the receiver position and its velocity as well as the clock offset and drift rate can be estimated.

4.5.1.1 Position and Clock Offset Estimation

The receiver position as well as the receiver clock offset can be estimated using the code pseudo-range observation. From Equation (4.2) and (4.3), the measurement model for single epoch code-based positioning can be derived. For the sake of simplicity, only single frequency band observations are considered since the underlying principles can easily be extended and applied to a multi-band scenario. The frequency band indicating subscript i is therefore dropped. If the instrumental delays d^s and d_r are assumed to be constant for a certain carrier wave frequency band i, they can be corrected or estimated together with the clock errors dt^s and dt_r, respectively. Furthermore, if the code multipath error dm_r^s and the other unmodeled code errors $\epsilon_{P,r}^s$ are assumed to be small, and if the satellite positions $\boldsymbol{r}^s$ as well as their clock errors dt^s are known, the unknown, remaining parameters are the receiver position $\boldsymbol{r}_r$ and the receiver clock error cdt_r. The unknown system state $\boldsymbol{x}_{\boldsymbol{r}}$ at epoch k can be described as:

$$\boldsymbol{x}_{\boldsymbol{r},k} = \begin{bmatrix} \boldsymbol{r}_r^\top & cdt_r \end{bmatrix}^\top \tag{4.23}$$

The initial state estimate $\hat{\boldsymbol{x}}_{\boldsymbol{r},1,0}$ for the first epoch can be set to zero, while for subsequent state estimates the previous solution can be used as initial state:

$$\hat{\boldsymbol{x}}_{\boldsymbol{r},k+1,j=0} = \lim_{j\to\infty} \hat{\boldsymbol{x}}_{\boldsymbol{r},k,j} \tag{4.24}$$

Since there are four unknown system parameters from Equation (4.23), at least four observations and therefore four valid satellites are necessary. The measurement vector $\boldsymbol{y_r}$ for the code pseudo-range observations at the receiver r from m satellites is given as:

$$\boldsymbol{y_r} = \begin{bmatrix} P_r^{s1} & P_r^{s2} & P_r^{s3} & \dots & P_r^{sm} \end{bmatrix}^\top \tag{4.25}$$

The simplified measurement equation can be derived from the code pseudo-range observation model in Equation (4.3) and taking into account the assumptions made beforehand:

$$\boldsymbol{h_r}(\boldsymbol{x_r}) = \begin{bmatrix} \rho_r^{s1} + I_r^{s1} + T_r^{s1} + c\left(dt_r - dt^{s1}\right) \\ \rho_r^{s2} + I_r^{s2} + T_r^{s2} + c\left(dt_r - dt^{s2}\right) \\ \rho_r^{s3} + I_r^{s3} + T_r^{s3} + c\left(dt_r - dt^{s3}\right) \\ \vdots \\ \rho_r^{sm} + I_r^{sm} + T_r^{sm} + c\left(dt_r - dt^{sm}\right) \end{bmatrix} \tag{4.26}$$

The error terms caused by the ionosphere I_r^s and the troposphere T_r^s need to be estimated and corrected for using an appropriate approach as described in Section 4.4. The measurement Jacobian of $\boldsymbol{h_r}(\boldsymbol{x})$ is obtained using the geometric range model in Equation (4.2) and neglecting the Sagnac effect correction terms:

$$\boldsymbol{H_r} = \left.\frac{\partial \boldsymbol{h_r}(\boldsymbol{x_r})}{\partial \boldsymbol{x_r}}\right|_{\boldsymbol{x_r}=\hat{\boldsymbol{x}}_{\boldsymbol{r},j}} = \begin{bmatrix} -\boldsymbol{e}_r^{s1\top} & 1 \\ -\boldsymbol{e}_r^{s2\top} & 1 \\ -\boldsymbol{e}_r^{s3\top} & 1 \\ \vdots & \vdots \\ -\boldsymbol{e}_r^{sm\top} & 1 \end{bmatrix}_{\boldsymbol{x_r}=\hat{\boldsymbol{x}}_{\boldsymbol{r},j}} \tag{4.27}$$

where $\boldsymbol{e}_r^s$ is the satellite Line-of-Sight (LOS) vector given by:

$$\boldsymbol{e}_r^s = \frac{\boldsymbol{r}^s\left(t^s\right) - \boldsymbol{r}_r\left(t_r\right)}{\left\|\boldsymbol{r}^s\left(t^s\right) - \boldsymbol{r}_r\left(t_r\right)\right\|} \tag{4.28}$$

The measurement weighing matrix $\boldsymbol{W_r}$ is given as:

$$\boldsymbol{W_r} = \operatorname{diag}\left(\sigma^2_{P_r^{s1}}, \sigma^2_{P_r^{s2}}, \sigma^2_{P_r^{s3}}, \dots, \sigma^2_{P_r^{sm}}\right) \tag{4.29}$$

where $\sigma_{P_r^s}$ is the standard deviation [m] for each observation of a specific satellite s, which depends on the satellite system F_{GNSS}, the satellite elevation θ_r^s, the received Signal-to-Noise-Ratio (SNR) C/N_0, the accuracy of the models used for ionospheric σ^2_{iono} and tropospheric σ^2_{tropo} corrections as well as the standard deviation of the ephemeris and satellite clock error σ^2_{eph}:

$$\sigma^2_{P_r^s} \approx f(F_{GNSS}, \theta_r^s, C/N_0) + \sigma^2_{iono} + \sigma^2_{tropo} + \sigma^2_{eph} \tag{4.30}$$

4.5.1.2 Velocity and Clock Drift Estimation

The receiver velocity and clock drift can be estimated in a similar manner as the position and the receiver clock offset, if Doppler-Shift measurements are available. The unknown state $\boldsymbol{x_v}$ at epoch k is given as:

$$\boldsymbol{x}_{\boldsymbol{v},k} = \begin{bmatrix} \boldsymbol{v}_r^\top & c\dot{dt}_r \end{bmatrix}^\top \tag{4.31}$$

where $\boldsymbol{v}_r$ is the receiver velocity in ECEF and $c\dot{dt}_r$ is the receiver clock drift. Just like before, the initial state estimate for the first epoch can be set to zero, while for subsequent state estimates the previous solution can be used as initial state. The pseudo-range rate measurement vector $\boldsymbol{y_v}$ can be obtained from the Doppler-Shift measurements and using Equation (4.10). For m observations, it is given as:

$$\begin{aligned} \boldsymbol{y_v} &= \begin{bmatrix} \dot{P}_r^{s_1} & \dot{P}_r^{s_2} & \dot{P}_r^{s_3} & \dots & \dot{P}_r^{s_m} \end{bmatrix}^\top \\ &= -\lambda_1 \begin{bmatrix} \Delta f_{D,r}^{s_1} & \Delta f_{D,r}^{s_2} & \Delta f_{D,r}^{s_3} & \dots & \Delta f_{D,r}^{s_m} \end{bmatrix}^\top \end{aligned} \tag{4.32}$$

The measurement equation $\boldsymbol{h_v}(\boldsymbol{x_v})$ can be obtained by deriving the pseudo-range measurement model using the assumption that there is no change in the ionospheric and tropospheric error between subsequent epochs. The measurement equation is then given as:

$$\boldsymbol{h_v}(\boldsymbol{x_v}) = \begin{bmatrix} \dot{\rho}_r^{s_1} + c\left(\dot{dt}_r - \dot{dt}^{s_1}\right) \\ \dot{\rho}_r^{s_2} + c\left(\dot{dt}_r - \dot{dt}^{s_2}\right) \\ \dot{\rho}_r^{s_3} + c\left(\dot{dt}_r - \dot{dt}^{s_3}\right) \\ \vdots \\ \dot{\rho}_r^{s_m} + c\left(\dot{dt}_r - \dot{dt}^{s_m}\right) \end{bmatrix} \tag{4.33}$$

where $\dot{\rho}_r^s$ is the geometric range-rate between receiver and satellite, given by:

$$\dot{\rho}_r^s = \boldsymbol{e}_r^{s\top}\left(\boldsymbol{v}^s - \boldsymbol{v_r}\right) + \frac{\omega_e}{c}\left(v_y^s x_r + y^s v_{x,r} - v_x^s y_r - x^s v_{y,r}\right) \tag{4.34}$$

Similar to the single epoch positioning, the measurement Jacobian $\boldsymbol{H_v}$ can be calculated as:

$$\boldsymbol{H_v} = \left.\frac{\partial \boldsymbol{h_v}(\boldsymbol{x_v})}{\partial \boldsymbol{x_v}}\right|_{\boldsymbol{x_v}=\hat{\boldsymbol{x}}_{\boldsymbol{v},j}} = \begin{bmatrix} -\boldsymbol{e}_r^{s_1\top} & 1 \\ -\boldsymbol{e}_r^{s_2\top} & 1 \\ -\boldsymbol{e}_r^{s_3\top} & 1 \\ \vdots & \vdots \\ -\boldsymbol{e}_r^{s_m\top} & 1 \end{bmatrix}_{\boldsymbol{x_v}=\hat{\boldsymbol{x}}_{\boldsymbol{v},j}} \tag{4.35}$$

The weighing matrix $\boldsymbol{W_v}$ can be modeled as :

$$\boldsymbol{W_v} = \text{diag}\left(\sigma^2_{\dot{P}_r^{s_1}}, \sigma^2_{\dot{P}_r^{s_2}}, \sigma^2_{\dot{P}_r^{s_3}}, \dots, \sigma^2_{\dot{P}_r^{s_m}}\right) \tag{4.36}$$

where $\sigma_{\dot{P}_r^s}$ is the standard deviation [m] for each range-rate observation of a specific satellite s and depends on the satellite system F_{GNSS}, the satellite elevation θ_r^s and the received Signal-to-Noise-Ratio (SNR) C/N_0.

4.5.2 Filtered Navigation

The position and velocity estimates from previous epochs can be used to predict the navigation solution at the next epoch. Prior information about the clock offset and clock drift rate are also included into the prediction. The current pseudo-range and pseudo-range rate observation are then used to correct the predicted navigation solution. In most GNSS user equipment, a Kalman Filter based state estimator is applied to obtain a filtered navigation solution. The main advantage of a filtered navigation solution is the smoothing of measurement noise and the possibility to maintain a navigation solution even under limited satellite signal availability for a short period of time. The fundamentals of a Kalman Filter are not discussed within this context, however, the reader is referred to [185] for a detailed introduction. Figure 4.8 summarizes the Kalman Filter framework for the state estimation of non-linear systems, hence an Extended Kalman Filter.

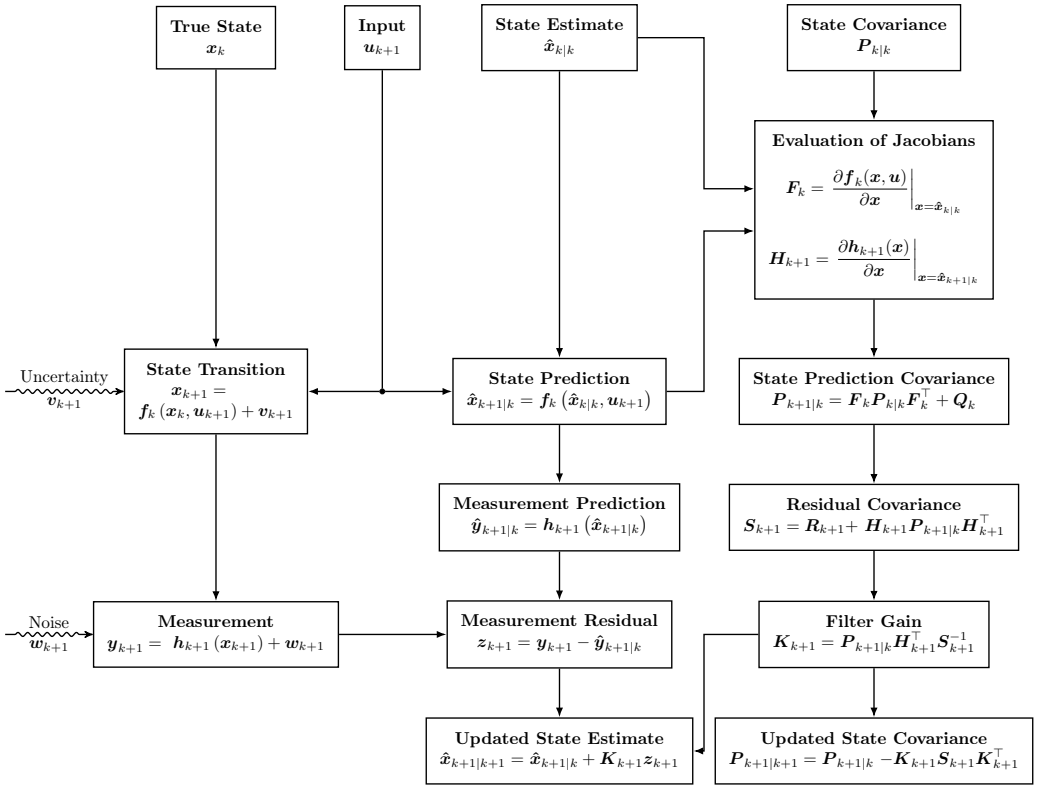


FIGURE 4.8: Flowchart of the EKF for one cycle [185].

The system state for a filtered navigation solution without the integration of inertial measurements can be simply obtained by stacking the single epoch estimations from Equation (4.23) and (4.31) together. This yields the true system state $\boldsymbol{x}$ at epoch k as:

$$\begin{aligned} \boldsymbol{x}_k &= \begin{bmatrix} \boldsymbol{x}_{\boldsymbol{r},k}^\top & \boldsymbol{x}_{\boldsymbol{v},k}^\top \end{bmatrix}^\top \\ &= \begin{bmatrix} \boldsymbol{r}_r^\top & cdt_r & \boldsymbol{v}_r^\top & c\dot{dt}_r \end{bmatrix}^\top \end{aligned} \tag{4.37}$$

Assuming a constant velocity model and a system input of zero, the system propagation function $\boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)$ in the ECEF frame is derived as:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ &= \boldsymbol{f}_k(\boldsymbol{x}_k, 0) \\ &= \begin{bmatrix} \boldsymbol{r}_r + \Delta t \cdot \boldsymbol{v}_r \\ cdt_r + \Delta t \cdot c\dot{dt}_r \\ \boldsymbol{v}_r \\ c\dot{dt}_r \end{bmatrix} \end{aligned} \tag{4.38}$$

where Δt is the sampling time between two consecutive GNSS measurements. The Jacobian of the propagation function is then simply given by:

$$\begin{aligned} \boldsymbol{F}_k &= \left.\frac{\partial \boldsymbol{f}_k(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k|k}} \\ &= \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I}\Delta t \\ 0 & \boldsymbol{I} \end{bmatrix} \end{aligned} \tag{4.39}$$

The measurement vector is the combination of pseudo-range and pseudo-range rate observations:

$$\begin{aligned} \boldsymbol{y}_k &= \begin{bmatrix} \boldsymbol{y}_{\boldsymbol{r},k}^\top & \boldsymbol{y}_{\boldsymbol{v},k}^\top \end{bmatrix}^\top \\ &= \begin{bmatrix} P_r^{s_1} & P_r^{s_2} & \dots & P_r^{s_m} & \dot{P}_r^{s_1} & \dot{P}_r^{s_2} & \dots & \dot{P}_r^{s_m} \end{bmatrix}^\top \end{aligned} \tag{4.40}$$

Therefore, the measurement equation is given by Equations (4.26) and (4.33) and reads:

$$\boldsymbol{h}_{k+1}(\boldsymbol{x}) = \begin{bmatrix} \rho_r^{s_1} + I_r^{s_1} + T_r^{s_1} + c\left(dt_r - dt^{s_1}\right) \\ \rho_r^{s_2} + I_r^{s_2} + T_r^{s_2} + c\left(dt_r - dt^{s_2}\right) \\ \rho_r^{s_3} + I_r^{s_3} + T_r^{s_3} + c\left(dt_r - dt^{s_3}\right) \\ \vdots \\ \rho_r^{s_m} + I_r^{s_m} + T_r^{s_m} + c\left(dt_r - dt^{s_m}\right) \\ \dot{\rho}_r^{s_1} + c\left(\dot{dt}_r - \dot{dt}^{s_1}\right) \\ \dot{\rho}_r^{s_2} + c\left(\dot{dt}_r - \dot{dt}^{s_2}\right) \\ \dot{\rho}_r^{s_3} + c\left(\dot{dt}_r - \dot{dt}^{s_3}\right) \\ \vdots \\ \dot{\rho}_r^{s_m} + c\left(\dot{dt}_r - \dot{dt}^{s_m}\right) \end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \tag{4.41}$$

where again $\dot{\rho}_{P,r}^s$ is the geometric range-rate between receiver and satellite and given by Equation (4.34). According to [160], a position error of 1 meter relates to a velocity error of approximately $5 \cdot 10^{-5}\mathrm{ms}^{-2}$. The dependence of pseudo-range rates on position is therefore quite weak and the $\partial\dot{\rho}_{P,r}^s / \partial \boldsymbol{r}_r$ terms can be consequently neglected.

Hence, the Jacobian of the measurement equation can be simplified to:

$$\begin{aligned}\boldsymbol{H}_{k+1} &= \left.\frac{\partial \boldsymbol{h}_{k+1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \begin{bmatrix} \dfrac{\partial \rho_r^s}{\partial \boldsymbol{r}_r} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \dfrac{\partial \dot{\rho}_{P,r}^s}{\partial \boldsymbol{r}_r} & \mathbf{0} & \dfrac{\partial \dot{\rho}_{P,r}^s}{\partial \boldsymbol{v}_r} & \mathbf{1} \end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \begin{bmatrix} \dfrac{\partial \rho_r^s}{\partial \boldsymbol{r}_r} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dfrac{\partial \dot{\rho}_{P,r}^s}{\partial \boldsymbol{v}_r} & \mathbf{1} \end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \begin{bmatrix} -\boldsymbol{E}_r^s & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\boldsymbol{E}_r^s & \mathbf{1} \end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}}\end{aligned} \tag{4.42}$$

where $\boldsymbol{E}_r^s$ is the collection of all LOS vectors between the receiver and the observed satellites:

$$\boldsymbol{E}_r^s = \begin{bmatrix} \boldsymbol{e}_r^{s_1\top} & \boldsymbol{e}_r^{s_2\top} & \boldsymbol{e}_r^{s_3\top} & \dots & \boldsymbol{e}_r^{s_m\top} \end{bmatrix}^\top \tag{4.43}$$

The main sources of increased uncertainty during the state propagation are changes in velocity due to user motion and the random walk of the receiver clock. There is also some additional phase noise on the clock offset [160]. In general, however, the system noise $\boldsymbol{v}(k)$ is inherently context-dependent. For small propagation intervals, the system noise covariance matrix $\boldsymbol{Q}$ can be simplified to:

$$\boldsymbol{Q} = \begin{bmatrix} \mathbf{0}_{3\times 3} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_{dt_r}^2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{Q}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_{\dot{dt}_r}^2 \end{bmatrix} \tag{4.44}$$

where $\sigma_{dt_r}^2$ and $\sigma_{\dot{dt}_r}^2$ are noise parameters that quantify the crystal accuracy and stability, while $\boldsymbol{Q}_r$ is a context-dependent noise matrix, reflecting dynamics of the receiver mounted platform:

$$\boldsymbol{Q}_r = \boldsymbol{C}_{E\mathcal{L}} \mathrm{diag}\left(\sigma_{r_h}^2, \sigma_{r_h}^2, \sigma_{r_v}^2\right) \boldsymbol{C}_{E\mathcal{L}}^\top \tag{4.45}$$

where σ_{r_h} and σ_{r_v} are the horizontal and vertical standard deviations of the receiver platform velocity in a local frame, respectively, and $\boldsymbol{C}_{E\mathcal{L}}$ is the transformation from the local frame to ECEF. While a pedestrian and a car experience different horizontal accelerations, they are both limited in their vertical movements. In contrast to that, a UAV can be subjected to high vertical accelerations. The measurement noise $\boldsymbol{w}(k)$ can be quantified similar to the weighing matrices of the single epoch solution in Equation (4.29) and (4.36). The measurement noise covariance matrix $\boldsymbol{R}$ can be written as:

$$\boldsymbol{R} = \mathrm{diag}\left(\sigma_{P_r^{s_1}}^2, \sigma_{P_r^{s_2}}^2, \sigma_{P_r^{s_3}}^2, \dots, \sigma_{P_r^{sm}}^2, \sigma_{\dot{P}_r^{s_1}}^2, \sigma_{\dot{P}_r^{s_2}}^2, \dots, \sigma_{\dot{P}_r^{sm}}^2\right) \tag{4.46}$$

4.5.3 Carrier-based Positioning for Short Baselines

If pseudo-range measurements are compared with the measurements made at a pre-surveyed location, slowly varying errors, such as the ephemeris prediction errors, satellite clock errors as well as ionospheric and tropospheric errors can be calibrated out. For applications where the reference station and the user are relatively close to each other, the accuracy of the code-based differential GNSS is limited by code tracking and multipath errors. Since carrier phase tracking is less noisy and exhibits smaller multipath disturbance, centimeter accuracy is attainable if relative carrier-based positioning techniques are applied [160]. Within the context of carrier phase-based positioning, Real-time Kinematic (RTK) describes differential GNSS techniques that are applicable in real-time for traditional setups and moving baselines. Within moving baseline setups, neither the reference station nor the user receiver need to be stationary. Consequently, RTK algorithms can be used in order to implement GNSS-based attitude determination systems. The most popular RTK approach is based on the double-differences, a technique that compares pseudo-range and carrier phase observations between different receivers and satellites in order to eliminate slowly varying error sources. The flowchart in Figure 4.9 shows the typical steps for RTK carrier-based positioning.

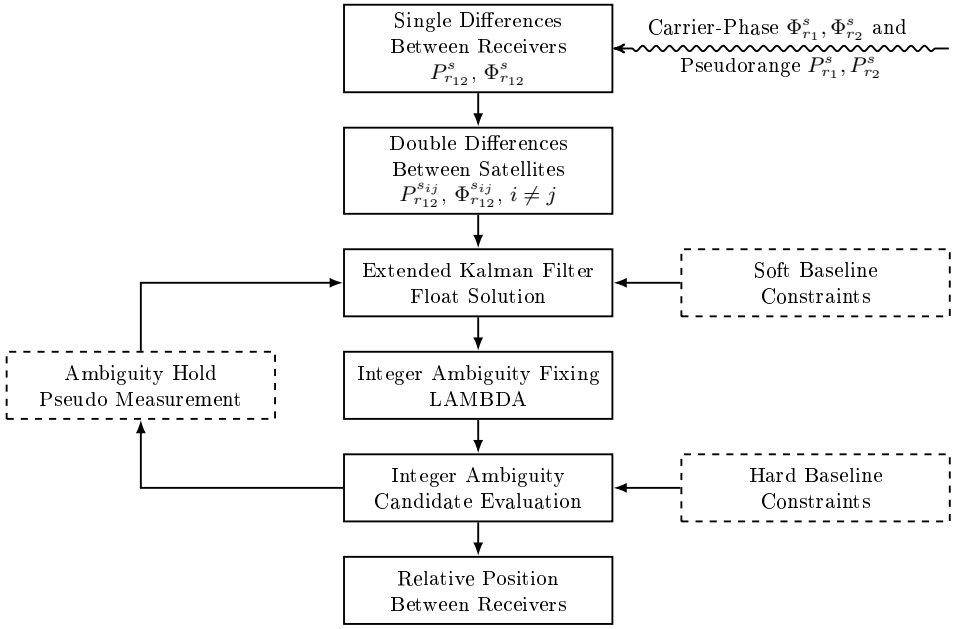


FIGURE 4.9: Flowchart for Carrier-based positioning with optional baseline constraints and integer ambiguity hold.

After the pseudo-range and carrier phase raw observations for a group of satellites s from different receivers r_1 and r_2 are obtained, the single-differences between receivers (see Section 4.5.3.1) and subsequently, the double-differences between receivers and satellites are computed (Section 4.5.3.2). Since carrier phase observations are ambiguous (see Equation (4.4)), the double-differenced carrier phase observations remain ambiguous, too. The ambiguities need to be resolved, in order to obtain the desired centimeter position accuracy. The integer ambiguity resolution is done in three steps. First, the integer ambiguities are estimated by determining a float solution using a EKF integrating the double-differenced pseudo-range and carrier

phase observations. The EKF float estimation is described in Section 4.5.3.3. Next, a search for a suitable combination of integer values based on the float ambiguities is executed (Section 4.5.3.4). In the final step, the candidate sets of integer ambiguities returned by the search are evaluated. If a suitable set of integers is found, the fixed ambiguities are used to correct the relative position between the two receivers. Conversely, if the integer ambiguities could not have been fixed correctly, the relative position is estimated based on the float solution of the integer ambiguities. Optional pseudo measurements (Section 4.5.3.5) and different baseline constraints (Section 4.5.3.6 and 4.5.3.7) can be utilized in order to include additional a priori knowledge about the baseline geometry allowing a more robust and faster determination of the double-differenced integer ambiguities.

4.5.3.1 Single Differences

Single difference models can be obtained by differencing observations from two different sources. These sources include different receivers, satellites, frequencies and epochs. An exhaustive analysis of these different models is presented in [161]. In the presented approach, the single difference is formed by differencing the observations between L1 single frequency receivers r_1 and r_2 for a common reference satellite s:

$$\begin{aligned} P^s_{r_{12}}(t_r) &= P^s_{r_2}(t_{r_2}) - P^s_{r_1}(t_{r_1}) \\ &= \rho^s_{r_2}(t_{r_2}, t^s) - \rho^s_{r_1}(t_{r_1}, t^s) \\ &\quad + I^s_{r_2} - I^s_{r_1} + T^s_{r_2} - T^s_{r_1} + dm^s_{r_2} - dm^s_{r_1} \\ &\quad + c\left[d_{r_2}(t_{r_2}) + d^s(t^s)\right] - c\left[d_{r_1}(t_{r_1}) + d^s(t^s)\right] \\ &\quad + c\left[dt_{r_2}(t_{r_2}) - dt^s(t^s)\right] - c\left[dt_{r_1}(t_{r_1}) - dt^s(t^s)\right] \\ &\quad + \epsilon^s_{P,r_2} - \epsilon^s_{P,r_1} \end{aligned} \tag{4.47}$$

$$\begin{aligned} \Phi^s_{r_{12}}(t_r) &= \Phi^s_{r_2}(t_{r_2}) - \Phi^s_{r_1}(t_{r_1}) \\ &= \rho^s_{r_2}(t_{r_2}, t^s) - \rho^s_{r_1}(t_{r_1}, t^s) \\ &\quad - I^s_{r_2} + I^s_{r_1} + T^s_{r_2} - T^s_{r_1} + \delta m^s_{r_2} - \delta m^s_{r_1} \\ &\quad + c\left[\delta_{r_2}(t_{r_2}) + \delta^s(t^s)\right] - c\left[\delta_{r_1}(t_{r_1}) + \delta^s(t^s)\right] \\ &\quad + c\left[dt_{r_2}(t_{r_2}) - dt^s(t^s)\right] - c\left[dt_{r_1}(t_{r_1}) - dt^s(t^s)\right] \\ &\quad + \lambda_1\left[\varphi_{r_2}(t_0) - \varphi^s(t_0)\right] - \lambda_1\left[\varphi_{r_1}(t_0) - \varphi^s(t_0)\right] \\ &\quad + \lambda_1 N^s_{r_2} - \lambda_1 N^s_{r_1} + \epsilon^s_{\Phi,r_2} - \epsilon^s_{\Phi,r_1} \end{aligned} \tag{4.48}$$

Using single differences, the satellite instrumental delays $d^s(t^s)$ and $\delta^s(t^s)$, the satellite clock error $dt^s(t^s)$ as well as the satellite phase offset $\varphi^s(t_0)$ is eliminated. For very short baselines ($< 10\,\text{km}$), the ionospheric and tropospheric signal path are almost identical and causing their errors to be almost canceled out [162, 186]. Assuming $I^s_{r_2} = I^s_{r_1}$ and $T^s_{r_2} = T^s_{r_1}$ and lumping the remaining terms together, the single difference model reads:

$$P^s_{r_{12}} = \rho^s_{r_{12}} + dm^s_{r_{12}} + cd_{r_{12}} + cdt_{r_{12}} + \epsilon^s_{P,r_{12}} \tag{4.49}$$

$$\begin{aligned} \Phi^s_{r_{12}} &= \rho^s_{r_{12}} + \delta m^s_{r_{12}} + c\delta_{r_{12}} + cdt_{r_{12}} \\ &\quad + \lambda_1\varphi_{r_{12}}(t_0) + \lambda_1 N^s_{r_{12}} + \epsilon^s_{\Phi,r_{12}} \end{aligned} \tag{4.50}$$

where the subscript r_{12} indicates the difference between the two receivers:

$$\bullet_{r_{12}} = \bullet_{r_2} - \bullet_{r_1} \tag{4.51}$$

Besides the relative clock errors $cdt_{r_{12}}$ and relative instrumental delays $d_{r_{12}}$ and $c\delta_{r_{12}}$, the initial relative receiver phase offset $\varphi_{r_{12}}(t_0)$ and the single difference integer ambiguity $N_{r_{12}}$ remain unknown. The single difference integer ambiguity is depicted in Figure 4.10. Due to the short baseline assumptions, the LOS vectors are assumed to be identical:

$$\boldsymbol{e}^s_{r_1} \approx \boldsymbol{e}^s_{r_2} \tag{4.52}$$

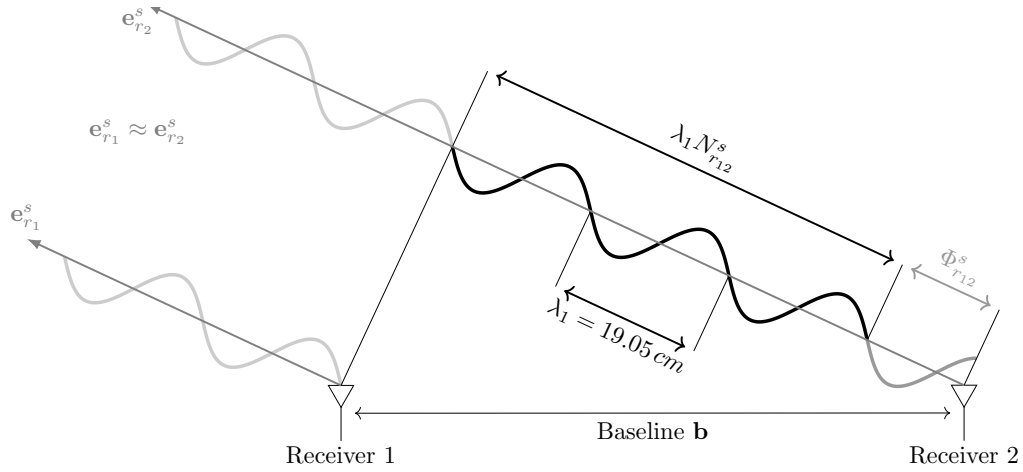


FIGURE 4.10: Integer ambiguity $N^s_{r_{12}}$ for the single difference between two L1 receivers, where $\boldsymbol{e}^s_r$ are the Line-of-Sight (LOS) vectors between the receivers and the satellite, $\Phi^s_{r_{12}}$ the phase-range difference between both receivers and λ_1 the L1 wavelength.

4.5.3.2 Double Differences

The goal of forming double differences is to eliminate the remaining receiver depended errors. The receiver clock errors and instrumental delays from Equation (4.49) and (4.50) are identical for satellite signals at the same frequency. In contrast, since GLONASS uses FDMA, so called inter channel biases need to be considered and removed beforehand [187]. For modern GNSS receivers that are designed for differential carrier phase-based positioning, it can be assumed that the local signal replicas are in phase, and hence that the relative receiver phase offset $\varphi_{r_{12}}(t_0)$ is canceled out [162]. The double difference model for a satellite pair i, j reads:

$$\begin{aligned} P^{s_{ij}}_{r_{12}} &= \left[P^{s_j}_{r_2} - P^{s_j}_{r_1}\right] - \left[P^{s_i}_{r_2} - P^{s_i}_{r_1}\right] \\ &= \rho^{s_j}_{r_{12}} - \rho^{s_i}_{r_{12}} + dm^{s_j}_{r_{12}} - dm^{s_i}_{r_{12}} + \epsilon^{s_j}_{P,r_{12}} - \epsilon^{s_i}_{P,r_{12}} \end{aligned} \tag{4.53}$$

$$\begin{aligned} \Phi^{s_{ij}}_{r_{12}} &= \left[\Phi^{s_j}_{r_2} - \Phi^{s_j}_{r_1}\right] - \left[\Phi^{s_i}_{r_2} - \Phi^{s_i}_{r_1}\right] \\ &= \rho^{s_j}_{r_{12}} - \rho^{s_i}_{r_{12}} + \delta m^{s_j}_{r_{12}} - \delta m^{s_i}_{r_{12}} \\ &\quad + \lambda_1 \left[N^{s_j}_{r_{12}} - N^{s_i}_{r_{12}}\right] + \epsilon^{s_j}_{\Phi,r_{12}} - \epsilon^{s_i}_{\Phi,r_{12}} \end{aligned} \tag{4.54}$$

Since the multipath error depends on the relative geometry between receiver, satellite and environment, it cannot be mitigated using double differences [162, 178]. Lumping the remaining terms together and assuming that the multipath error is small enough to be neglected for UAVs, the final double difference model is:

$$P_{r_{12}}^{s_{ij}} = \rho_{r_{12}}^{s_{ij}} + \epsilon_{P,r_{12}}^{s_{ij}} \tag{4.55}$$

$$\Phi_{r_{12}}^{s_{ij}} = \rho_{r_{12}}^{s_{ij}} + \lambda_1 N_{r_{12}}^{s_{ij}} + \epsilon_{\Phi,r_{12}}^{s_{ij}} \tag{4.56}$$

4.5.3.3 EKF Formulation

Theoretically, it is possible to compute the navigation solutions for all possible integer ambiguity combinations within the search area and then selecting the most consistent. However, in practice, this is not feasible due to the high computational load caused by the evaluation of over a million possible solutions [160]. In practical applications, the integer ambiguities $\boldsymbol{N}_{r_{12}}^{s_i}$ are first approximated as floating numbers $\hat{\boldsymbol{N}}_{r_{12}}^{s_i}$ using an Extended Kalman Filter (EKF). In order to avoid hand-over handling problems if the reference satellite is changed, single instead of double differenced ambiguities are estimated [157]. The unknown system state vector for single frequency receivers at epoch k can be then written as:

$$\begin{aligned} \boldsymbol{x}_k &= \begin{bmatrix} \boldsymbol{r}_{r_1,k}^\top & \boldsymbol{v}_{r_1,k}^\top & \hat{\boldsymbol{N}}_{r_{12}}^{s_i\ \top} \end{bmatrix}^\top \\ &= \begin{bmatrix} \boldsymbol{r}_{r_1,k}^\top & \boldsymbol{v}_{r_1,k}^\top & \hat{N}_{r_{12}}^{s_1} & \hat{N}_{r_{12}}^{s_2} & \hat{N}_{r_{12}}^{s_3} & \dots & \hat{N}_{r_{12}}^{s_m} \end{bmatrix}^\top \end{aligned} \tag{4.57}$$

where $\boldsymbol{r}_{r_1,k}$ and $\boldsymbol{v}_{r_1,k}$ are the receiver position and velocity of r_1 in the ECEF frame, respectively, and $\hat{\boldsymbol{N}}_{r_{12}}^{s_i}$ are the single differenced float ambiguities. In this case, the receiver r_2 is considered as reference station that can be both, stationary or moving. The system state can be propagated using a constant velocity model with zero input. The system transition is then given by:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1|k} &= \boldsymbol{F}\hat{\boldsymbol{x}}_{k|k} \\ &= \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{I}_{3\times3}\Delta t & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{3\times3} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{m\times m} \end{bmatrix} \hat{\boldsymbol{x}}_{k|k} \end{aligned} \tag{4.58}$$

where Δt is the time between two consecutive GNSS observations. The measurement vector $\boldsymbol{y}(k)$ is composed of the double differenced pseudo-range and carrier phase-range observations with respect to a fixed reference satellite j:

$$\boldsymbol{y}_k = \begin{bmatrix} \boldsymbol{P}_{r_{12}}^{s_j} \\ \boldsymbol{\Phi}_{r_{12}}^{s_j} \end{bmatrix} \tag{4.59}$$

For a measurement vector with satellite $j = 1$ as reference, the observation can be written as:

$$\boldsymbol{P}_{r_{12}}^{s_1} = \begin{bmatrix} P_{r_{12}}^{s_{12}} & P_{r_{12}}^{s_{13}} & P_{r_{12}}^{s_{14}} & \dots & P_{r_{12}}^{s_{1m}} \end{bmatrix}^\top \tag{4.60}$$

$$\boldsymbol{\Phi}_{r_{12}}^{s_1} = \begin{bmatrix} \Phi_{r_{12}}^{s_{12}} & \Phi_{r_{12}}^{s_{13}} & \Phi_{r_{12}}^{s_{14}} & \dots & \Phi_{r_{12}}^{s_{1m}} \end{bmatrix}^\top \tag{4.61}$$

Using the double difference model in Equation (4.55) and (4.56), the measurement model $\boldsymbol{h}_{k+1}$ can be derived as:

$$\boldsymbol{h}_{k+1}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{h}_{\boldsymbol{P},k+1}(\boldsymbol{x})^\top & \boldsymbol{h}_{\boldsymbol{\Phi},k+1}(\boldsymbol{x})^\top \end{bmatrix}^\top \tag{4.62}$$

where $\boldsymbol{h}_{\boldsymbol{P},k+1}$ and $\boldsymbol{h}_{\boldsymbol{\Phi},k+1}$ for the reference satellite $j = 1$ can be written as:

$$\boldsymbol{h}_{\boldsymbol{P},k+1}(\boldsymbol{x}) = \begin{bmatrix} \rho_{r_{12}}^{s_{12}} \\ \rho_{r_{12}}^{s_{13}} \\ \rho_{r_{12}}^{s_{14}} \\ \vdots \\ \rho_{r_{12}}^{s_{1m}} \end{bmatrix}, \quad \boldsymbol{h}_{\boldsymbol{\Phi},k+1}(\boldsymbol{x}) = \begin{bmatrix} \rho_{r_{12}}^{s_{12}} + \lambda_1 \left(N_{r_{12}}^{s_2} - N_{r_{12}}^{s_1} \right) \\ \rho_{r_{12}}^{s_{13}} + \lambda_1 \left(N_{r_{12}}^{s_3} - N_{r_{12}}^{s_1} \right) \\ \rho_{r_{12}}^{s_{14}} + \lambda_1 \left(N_{r_{12}}^{s_4} - N_{r_{12}}^{s_1} \right) \\ \vdots \\ \rho_{r_{12}}^{s_{1m}} + \lambda_1 \left(N_{r_{12}}^{s_m} - N_{r_{12}}^{s_1} \right) \end{bmatrix} \tag{4.63}$$

The measurement Jacobian $\boldsymbol{H}_{k+1}$ can be derived as:

$$\begin{aligned} \boldsymbol{H}_{k+1} &= \left. \frac{\partial \boldsymbol{h}_{k+1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \\ &= \begin{bmatrix} -\boldsymbol{D}\boldsymbol{E} & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{D}\boldsymbol{E} & \boldsymbol{0} & \lambda_1 \boldsymbol{D} \end{bmatrix}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} \end{aligned} \tag{4.64}$$

where $\boldsymbol{D} \in \mathbb{R}^{(m-1)\times m}$ is the single differencing matrix given as:

$$\boldsymbol{D} = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & \dots & 1 \end{bmatrix} \tag{4.65}$$

and $\boldsymbol{E}$ is the stacked vector of all LOS vectors:

$$\boldsymbol{E} = \begin{bmatrix} \boldsymbol{e}_r^{s_1\top} & \boldsymbol{e}_r^{s_2\top} & \dots & \boldsymbol{e}_r^{s_m\top} \end{bmatrix}^\top \tag{4.66}$$

Similar to the filtered navigation solution, the main source of increased uncertainty during the state propagation are changes in velocity due to user motion. The system noise covariance matrix $\boldsymbol{Q}$ for m satellites is hence given by:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{0}_{3\times 3} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q_r} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}_{(m-1)\times(m-1)} \end{bmatrix} \tag{4.67}$$

where $\boldsymbol{Q_r}$ follows the definition in Equation (4.45). The measurement covariance matrix $\boldsymbol{R}$ depends on the double differenced noise of the raw observations:

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{D}\boldsymbol{R}_P\boldsymbol{D}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}\boldsymbol{R}_\Phi\boldsymbol{D}^\top \end{bmatrix}^\top \tag{4.68}$$

where $\boldsymbol{R}_P$ and $\boldsymbol{R}_\Phi$ are given by the raw observation errors $\sigma_{P_r^s}^2$ and $\sigma_{\Phi_r^s}^2$ as:

$$\boldsymbol{R}_P = 2 \cdot \operatorname{diag}\left(\sigma_{P_r^{s_1}}^2, \sigma_{P_r^{s_2}}^2, \sigma_{P_r^{s_3}}^2, \dots, \sigma_{P_r^{s_m}}^2 \right) \tag{4.69}$$

$$\boldsymbol{R}_\Phi = 2 \cdot \operatorname{diag}\left(\sigma_{\Phi_r^{s_1}}^2, \sigma_{\Phi_r^{s_2}}^2, \sigma_{\Phi_r^{s_3}}^2, \dots, \sigma_{\Phi_r^{s_m}}^2 \right) \tag{4.70}$$

4.5.3.4 Ambiguity Fixing

In order to estimate the integer ambiguities of the double difference model in Equation (4.56), the EKF state and state covariance estimates need to be transformed from single differences to double differences:

$$\begin{aligned} \hat{\boldsymbol{x}}'_{k+1|k+1} &= \boldsymbol{G}\hat{\boldsymbol{x}}_{k+1|k+1} && (4.71)\\ &= \begin{bmatrix} \hat{\boldsymbol{r}}_{r_1}^\top & \hat{\boldsymbol{v}}_{r_1}^\top & \hat{\boldsymbol{N}}^\top \end{bmatrix}^\top \\ \boldsymbol{P}'_{k+1|k+1} &= \boldsymbol{G}\boldsymbol{P}_{k+1|k+1}\boldsymbol{G}^\top && (4.72)\\ &= \begin{bmatrix} \boldsymbol{Q}_{\hat{\boldsymbol{P}}} & \boldsymbol{Q}_{\hat{\boldsymbol{N}}\hat{\boldsymbol{P}}} \\ \boldsymbol{Q}_{\hat{\boldsymbol{P}}\hat{\boldsymbol{N}}} & \boldsymbol{Q}_{\hat{\boldsymbol{N}}} \end{bmatrix}^\top \end{aligned}$$

where $\boldsymbol{G}$ is the single to double difference transformation matrix and given as:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{I}_{6\times 6} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D} \end{bmatrix} \tag{4.73}$$

Given the double differenced float ambiguities $\hat{\boldsymbol{N}} \in \mathbb{R}^{(m-1)}$ and their respective covariance matrix $\boldsymbol{Q}_{\hat{\boldsymbol{N}}} \in \mathbb{R}^{(m-1)\times(m-1)}$, the integer ambiguities $\check{\boldsymbol{N}} \in \mathbb{Z}^{(m-1)}$ can be obtained by solving a Integer Least Square (ILS) problem expressed as:

$$\begin{aligned} \check{\boldsymbol{N}} &= \underset{\boldsymbol{N}\in\mathbb{Z}^{(m-1)}}{\arg\min} \left[\left(\hat{\boldsymbol{N}} - \boldsymbol{N}\right)^\top \boldsymbol{Q}_{\hat{\boldsymbol{N}}}{}^{-1} \left(\hat{\boldsymbol{N}} - \boldsymbol{N}\right) \right] && (4.74)\\ &= \underset{\boldsymbol{N}\in\mathbb{Z}^{(m-1)}}{\arg\min} \left[\left\| \hat{\boldsymbol{N}} - \boldsymbol{N} \right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}} \right] \end{aligned}$$

where the following notation is used:

$$(\bullet)^\top \boldsymbol{Q}^{-1} (\bullet) = \|\bullet\|^2_{\boldsymbol{Q}} \tag{4.75}$$

Next, an integer search is conducted to find a solution for the above problem. The search space can be defined by introducing a properly chosen constant χ^2:

$$\left\{ \boldsymbol{N} \in \mathbb{Z}^{(m-1)} \mid \left\| \hat{\boldsymbol{N}} - \boldsymbol{N} \right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}} \leq \chi^2 \right\} \tag{4.76}$$

ILS problems are known to be NP-hard. While integer rounding schemes are rather simple but time consuming, fixing integer ambiguities efficiently turns out to be quite complex and a non-trivial problem. With the Least-Square Ambiguity Decorrelation Adjustment (LAMBDA) approach, a well-known integer fixing strategy was introduced in [52, 188].

LAMBDA addresses the ILS problem in two steps: The problem reduction and the actual search. The search space reduction is conducted in order to allow a more efficient search. Its goal is to transform the highly elongated and correlated covariance matrix $\boldsymbol{Q}_{\hat{\boldsymbol{N}}}$ and the respective float ambiguities using the so-called $\boldsymbol{Z}$-transformation in such a way that the least-square integer estimation becomes trivial and hence more efficient. The utilized $\boldsymbol{Z}$-transformations are required to be integer and volume preserving in order to successfully decorrelate the integer ambiguities.

The $\boldsymbol{Z}$-transformation is defined as:

$$\boldsymbol{z} = \boldsymbol{Z}^\top \boldsymbol{N}, \quad \hat{\boldsymbol{z}} = \boldsymbol{Z}^\top \hat{\boldsymbol{N}}, \quad \boldsymbol{Q}_{\hat{\boldsymbol{z}}} = \boldsymbol{Z}^\top \boldsymbol{Q}_{\hat{\boldsymbol{N}}} \boldsymbol{Z} \tag{4.77}$$

where $\boldsymbol{Z} \in \mathbb{Z}^{(m-1)\times(m-1)}$ and $|\det(\boldsymbol{Z})| = 1$. Using the above transformation, the original ILS problem is transferred to a new one:

$$\check{\boldsymbol{z}} = \underset{\boldsymbol{z}\in\mathbb{Z}^{(m-1)}}{\arg\min} \left[\|\hat{\boldsymbol{z}} - \boldsymbol{z}\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{z}}}} \right] \tag{4.78}$$

In order to find a suitable $\boldsymbol{Z}$-transformation with the above characteristics, $\boldsymbol{L}^\top \boldsymbol{D}\boldsymbol{L}$-factorizations of $\boldsymbol{Q}_{\hat{\boldsymbol{N}}}$ and $\boldsymbol{Q}_{\hat{\boldsymbol{z}}}$ need to be considered:

$$\boldsymbol{Q}_{\hat{\boldsymbol{N}}} = \boldsymbol{L}^\top \boldsymbol{D}\boldsymbol{L}, \quad \boldsymbol{Q}_{\hat{\boldsymbol{z}}} = \boldsymbol{Z}^\top \boldsymbol{L}^\top \boldsymbol{D}\boldsymbol{L}\boldsymbol{Z} = \bar{\boldsymbol{L}}^\top \bar{\boldsymbol{D}}\bar{\boldsymbol{L}} \tag{4.79}$$

In the LAMBDA method, the $\boldsymbol{Z}$-transformation is constructed by a sequence of integer Gauss transformations and permutations in such a way that $\boldsymbol{Q}_{\hat{\boldsymbol{z}}}$ is as diagonal as possible and hence, the new covariance therefore decorrelated. Additionally, the diagonal entries of $\bar{\boldsymbol{D}}$ are sorted in a decreasing order. For a more detailed description of the approach, the reader is referred to [189]. A modified version of LAMBDA (MLAMBDA) identified the integer Gauss transformations and permutations as the most time consuming step and proposed a faster way to compute the $\boldsymbol{L}^\top \boldsymbol{D}\boldsymbol{L}$-factorization of $\boldsymbol{Q}_{\hat{\boldsymbol{z}}}$ by the use of symmetric pivoting and decorrelating the parameters by a combination of greedy selections and lazy transformations [190]. Figure 4.11 illustrates the highly correlated covariance ellipsoid for $\boldsymbol{Q}_{\hat{\boldsymbol{N}}}$, the decorrelated covariance ellipsoid for $\boldsymbol{Q}_{\hat{\boldsymbol{z}}}$ as well as the initial float ambiguities $\hat{\boldsymbol{N}}$ and their integer fixes $\check{\boldsymbol{N}}$ for a simple two dimensional problem.

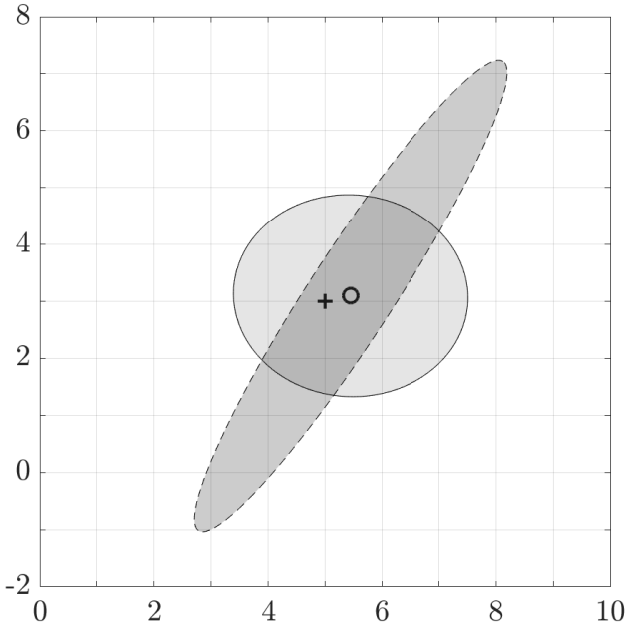


FIGURE 4.11: LAMBDA Integer Ambiguity Fixing for $\boldsymbol{Q}_{\hat{\boldsymbol{N}}} \in \mathbb{R}^{2\times2}$: Highly correlated covariance ellipsoid for $\boldsymbol{Q}_{\hat{\boldsymbol{N}}}$ (dashed), decorrelated covariance ellipsoid for $\boldsymbol{Q}_{\hat{\boldsymbol{z}}}$ (solid), float ambiguities $\hat{\boldsymbol{N}}$ (**o**) and fixed ambiguities $\check{\boldsymbol{N}}$ (**+**).

Once a suitable $\boldsymbol{Z}$-transformation is obtained, a discrete search is performed by applying a sequential conditional least-square estimation. The new search space is defined by:

$$\left\{ \boldsymbol{z} \in \mathbb{Z}^{(m-1)} \mid \|\hat{\boldsymbol{z}} - \boldsymbol{z}\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{z}}}} \leq \chi^2 \right\} \tag{4.80}$$

Another optimization introduced by MLAMBDA allows a dynamic shrinking of the search space during the search process [190]. The integer sets estimated in the new search space can be transformed back to the original ILS problem using the inverse $\boldsymbol{Z}$-transformation. Typically, LAMBDA computes several integer sets $\check{\boldsymbol{N}}_i$, which allows to evaluate the computed sets based on additional criteria. The simplest candidate set evaluation method compares the ratio between be best $\check{\boldsymbol{N}}_1$ and the second best $\check{\boldsymbol{N}}_2$ integer estimate:

$$R = \frac{\left\|\hat{\boldsymbol{N}} - \check{\boldsymbol{N}}_2\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}}}{\left\|\hat{\boldsymbol{N}} - \check{\boldsymbol{N}}_1\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}}} > R_{thres} \tag{4.81}$$

If the ratio exceeds a certain threshold R_{thres}, the integer ambiguities are assumed to be fixed correctly. Another candidate evaluation criteria considers additional baseline constraints and is introduced in Section 4.5.3.7. The fixed baseline solution can be computed by correcting the float estimate:

$$\begin{bmatrix} \check{\boldsymbol{r}}_{r_1} \\ \check{\boldsymbol{v}}_{r_1} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{r}}_{r_1} \\ \hat{\boldsymbol{v}}_{r_1} \end{bmatrix} + \boldsymbol{Q}_{\hat{\boldsymbol{P}}\hat{\boldsymbol{N}}} \boldsymbol{Q}_{\hat{\boldsymbol{N}}} \left(\hat{\boldsymbol{N}} - \check{\boldsymbol{N}} \right) \tag{4.82}$$

4.5.3.5 Pseudo Measurements

Pseudo measurements can be introduced to reduce the EKF convergence time and to correct the EKF state estimates once the integer ambiguities are resolved correctly. The float state estimates are forced towards the fixed integer solution. A pseudo measurement vector consists therefore of the fixed double differenced integer ambiguities:

$$\boldsymbol{y}_{\check{N}} = \check{\boldsymbol{N}}^{s_1}_{r_{12}} = \begin{bmatrix} \check{N}^{s_{12}}_{r_{12}} & \check{N}^{s_{13}}_{r_{12}} & \check{N}^{s_{14}}_{r_{12}} & \dots & \check{N}^{s_{1m}}_{r_{12}} \end{bmatrix}^\top \tag{4.83}$$

The measurement prediction is based on the current float estimates and constructed using the double difference matrix $\boldsymbol{D}$:

$$\begin{aligned} \boldsymbol{h}_{\check{N},k+1} &= \boldsymbol{H}_{\check{N},k+1} \hat{\boldsymbol{x}}_{k+1|k} \\ &= \begin{bmatrix} \boldsymbol{0} & \boldsymbol{D} \end{bmatrix} \hat{\boldsymbol{x}}_{k+1|k} \end{aligned} \tag{4.84}$$

Using the measurement noise covariance matrix, the weight of the pseudo measurements can be adjusted:

$$\boldsymbol{R}_{\check{N}} = \operatorname{diag}\left(\sigma_c^2, \sigma_c^2, \sigma_c^2, \dots\right) \tag{4.85}$$

4.5.3.6 Soft Baseline Constraints

Soft baseline constraints, are baseline constraints that use a priori information about the baseline configuration as an additional pseudo measurement during the float estimation step. Their purpose is to increase the accuracy and stability of the float estimates. Depending on the available information about the baseline configuration, either the baseline length or a vector describing the complete baseline can be used in this step. The baseline length and the baseline vector pseudo measurement can be used simultaneously or independently.

Baseline Length The baseline length constraint is described in [157]. The additional pseudo measurement is simply given by a length observation of the baseline:

$$\boldsymbol{y_L} = l_{baseline} \tag{4.86}$$

The baseline length is measured with a known observation accuracy:

$$\boldsymbol{R_L} = \sigma_{\boldsymbol{L}}^2 \tag{4.87}$$

The measurement prediction is obtained from the current rover position float estimate $\hat{\boldsymbol{r}}_{r_1}$ and the known base position $\boldsymbol{r}_{r_2}$, respectively:

$$\boldsymbol{h}_{\boldsymbol{L},k+1} = |\hat{\boldsymbol{r}}_{r_1} - \boldsymbol{r}_{r_2}| \tag{4.88}$$

The measurement Jacobian is then given by:

$$\boldsymbol{H}_{\boldsymbol{L},k+1} = \frac{\hat{\boldsymbol{r}}_{r_1} - \boldsymbol{r}_{r_2}}{|\hat{\boldsymbol{r}}_{r_1} - \boldsymbol{r}_{r_2}|} \tag{4.89}$$

For very short baselines, e.g. a UAV mounted GNSS compass, non-linearities might arise due to the rotation of the complete platform. According to [157], iterative pseudo measurement updates can be applied in order to cope with the non-linearities. The information about the baseline length can be simply measured in the case of a moving baseline configuration, where the rover and the base are mounted on a common platform and move together. In case of a fixed base station, the additional information can be obtained by other radio ranging techniques.

Baseline Vector The baseline vector constraint is an extension of the baseline length constraint for the case that the full baseline configuration is known. The approach is described in [45, 53]. The pseudo measurement is given by an externally observed or known baseline vector $\boldsymbol{b}_{\mathcal{L}}$ given in a local frame $\mathcal{L}$:

$$\boldsymbol{y_b} = \boldsymbol{C}_{E\mathcal{L}} \boldsymbol{b}_{\mathcal{L}} \tag{4.90}$$

where $\boldsymbol{C}_{E\mathcal{L}}$ is a rotation of the baseline vector from the local frame to the frame used by the underlying EKF, typically the ECEF frame. Similar, the accuracy of the a priori known baseline vector needs to be transformed into the according navigation frame of the EKF:

$$\boldsymbol{R_{bv}} = \boldsymbol{C}_{E\mathcal{L}} \begin{bmatrix} \sigma^2_{\boldsymbol{bv}_x} & 0 & 0 \\ 0 & \sigma^2_{\boldsymbol{bv}_y} & 0 \\ 0 & 0 & \sigma^2_{\boldsymbol{bv}_z} \end{bmatrix} \boldsymbol{C}_{E\mathcal{L}}^{\top} \tag{4.91}$$

where $\sigma_{\boldsymbol{bv}_i}$ is the measurement accuracy in the according axis i in the local frame. The measurement prediction is then simply given by:

$$\boldsymbol{h}_{\boldsymbol{bv},k+1} = \hat{\boldsymbol{r}}_{r_1} - \boldsymbol{r}_{r_2} \tag{4.92}$$

And its Jacobian reads:

$$\boldsymbol{H}_{\boldsymbol{bv},k+1} = \boldsymbol{I}_{3\times 3} \tag{4.93}$$

In case of a UAV mounted GNSS compass, the movement of the a priori measured baseline vector can be additionally tracked by inertial sensors.

4.5.3.7 Hard Baseline Constraints

Hard baseline constraints, are baseline constraints that are applied during the integer ambiguity fixing step. Throughout literature, different approaches for hard baseline constrains are considered [191–194]. The various approaches depend on the available amount of a priori information and suggest different optimizations to increase the efficiency of hard baseline constraints. Similar to the soft baseline constraints, the constraints can be limited to the baseline length or a vector describing the baseline. Additional information derived from the baseline vector, such as the baseline heading and inclination angle, can be used as constraints, too [191]. The baseline constraint LAMBDA approach described here is limited to a priori knowledge about the baseline length and the baseline vector and builds upon the work done in [194].

In case of moving base scenarios, e.g. GNSS compass applications, the velocity is not part of the navigation solution, since only the relative position of the rover with respect to the base station is required. The reduced system state is then given by:

$$\begin{aligned} \hat{\boldsymbol{x}}'_{k+1|k+1} &= \boldsymbol{G}\hat{\boldsymbol{x}}_{k+1|k+1} \\ &= \begin{bmatrix} \hat{\boldsymbol{r}}_{r_1}^\top & \hat{\boldsymbol{N}}^\top \end{bmatrix}^\top \end{aligned} \tag{4.94}$$

$$\begin{aligned} \boldsymbol{P}'_{k+1|k+1} &= \boldsymbol{G}\boldsymbol{P}_{k+1|k+1}\boldsymbol{G}^\top \\ &= \begin{bmatrix} \boldsymbol{Q}_{\hat{\boldsymbol{R}}} & \boldsymbol{Q}_{\hat{\boldsymbol{N}}\hat{\boldsymbol{R}}} \\ \boldsymbol{Q}_{\hat{\boldsymbol{R}}\hat{\boldsymbol{N}}} & \boldsymbol{Q}_{\hat{\boldsymbol{N}}} \end{bmatrix}^\top \end{aligned} \tag{4.95}$$

where $\boldsymbol{G}$ is the single to double difference transformation matrix and given as:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{I}_{3\times 3} & \boldsymbol{0}_{3\times m} \\ \boldsymbol{0}_{(m-1)\times 3} & \boldsymbol{D} \end{bmatrix} \tag{4.96}$$

In general, the baseline vector can be be estimated using the float estimates of the rover position $\hat{\boldsymbol{r}}_{r_1}$ and the known position of the base $\boldsymbol{r}_{r_2}$ by:

$$\hat{\boldsymbol{b}}(\hat{\boldsymbol{N}}) = \hat{\boldsymbol{r}}_{r_1} - \boldsymbol{r}_{r_2} \tag{4.97}$$

The accuracy of the float baseline estimation can be calculated using the covariance matrices from Equation (4.95):

$$\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\hat{\boldsymbol{N}})} = \boldsymbol{Q}_{\hat{\boldsymbol{R}}} - \boldsymbol{Q}_{\hat{\boldsymbol{R}}\hat{\boldsymbol{N}}}\boldsymbol{Q}_{\hat{\boldsymbol{N}}}{}^{-1}\boldsymbol{Q}_{\hat{\boldsymbol{N}}\hat{\boldsymbol{R}}} \tag{4.98}$$

For a candidate set of integer ambiguities, the accordingly corrected baseline can be obtained similar to Equation (4.82):

$$\hat{\boldsymbol{b}}(\boldsymbol{N}) = \hat{\boldsymbol{b}}(\hat{\boldsymbol{N}}) + \boldsymbol{Q}_{\hat{\boldsymbol{R}}\hat{\boldsymbol{N}}}\boldsymbol{Q}_{\hat{\boldsymbol{N}}}^{-1}\left(\hat{\boldsymbol{N}} - \boldsymbol{N}\right) \tag{4.99}$$

A straight forward and rather simple approach, is applying an additional validation step during the integer ambiguity fixing by constraining the baseline length or vector for each candidate set:

$$\left|\hat{\boldsymbol{b}}(\boldsymbol{N}) - l_{baseline}\right| \leq \Delta l_{baseline} \tag{4.100}$$

$$\left\|\hat{\boldsymbol{b}}(\boldsymbol{N}) - \boldsymbol{C}_{E\mathcal{L}}\boldsymbol{b}_{\mathcal{L}}\right\|_I^2 \leq \left\|\boldsymbol{C}_{E\mathcal{L}}\Delta\boldsymbol{b}_{\mathcal{L}}\right\|_I^2 \tag{4.101}$$

with suitable values for $\Delta l_{baseline}$ and $\Delta\boldsymbol{b}_{\mathcal{L}}$. For all ambiguity candidates the baseline $\hat{\boldsymbol{b}}(\boldsymbol{N})$ is calculated and compared to the according threshold. If the computed baseline does not fall within a tolerable region, the integer candidate $\boldsymbol{N}$ is rejected. However, this simple method is computational expensive, since the baseline has to be computed and evaluated for each candidate.

BC-LAMBDA The baseline constrained LAMBDA method extends the ILS problem in Equation (4.74) to meet additional constraints given by a priori information about the baseline configuration:

$$\check{\boldsymbol{N}} = \underset{\boldsymbol{N}\in\mathbb{Z}^{(m-1)},\boldsymbol{b}\in\mathbb{R}^3}{\arg\min}\left[\left\|\hat{\boldsymbol{N}} - \boldsymbol{N}\right\|_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}}^2 + \left\|\hat{\boldsymbol{b}}(\boldsymbol{N}) - \boldsymbol{b}\right\|_{\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\boldsymbol{N})}}^2\right] \tag{4.102}$$

subject to:

$$\|\boldsymbol{b}\|_I^2 = l_{baseline}^2$$

This problem is commonly referred to as Integer Least Squares with Quadratic Equality (ILSQE). In [194], it is shown that the ILSQE can be rewritten as:

$$\check{\boldsymbol{N}} = \underset{\boldsymbol{N}\in\mathbb{Z}^{(m-1)}}{\arg\min}\left[\left\|\hat{\boldsymbol{N}} - \boldsymbol{N}\right\|_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}}^2 + \left\|\hat{\boldsymbol{b}}(\boldsymbol{N}) - \check{\boldsymbol{b}}(\boldsymbol{N})\right\|_{\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\boldsymbol{N})}}^2\right] \tag{4.103}$$

where $\check{\boldsymbol{b}}(\boldsymbol{N})$ is the constrained baseline vector and defined as:

$$\check{\boldsymbol{b}}(\boldsymbol{N}) = \underset{\boldsymbol{b}\in\mathbb{R}^3,\|\boldsymbol{b}\|_I^2 = l_{baseline}^2}{\arg\min}\left[\left\|\hat{\boldsymbol{b}}(\boldsymbol{N}) - \boldsymbol{b}\right\|_{\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\boldsymbol{N})}}^2\right] \tag{4.104}$$

Since there is no analytic solution for Equation (4.103), it has to be solved using an efficient search method. The ambiguity search space can be defined as:

$$\left\{\boldsymbol{N}\in\mathbb{Z}^{(m-1)} \mid \left\|\hat{\boldsymbol{N}} - \boldsymbol{N}\right\|_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}}^2 + \left\|\hat{\boldsymbol{b}}(\boldsymbol{N}) - \check{\boldsymbol{b}}(\boldsymbol{N})\right\|_{\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\boldsymbol{N})}}^2 \leq \chi^2\right\} \tag{4.105}$$

where χ^2 is a properly chosen constant. Choosing χ^2 is not trivial and discussed in detail in [194]. In order to keep the search space as small as possible, but still reasonably big, χ^2 is initially chosen to be equal to χ_{ILS}^2, which equals:

$$\chi_{ILS}^2 = \left\|\hat{\boldsymbol{N}} - \boldsymbol{N}_{ILS}\right\|_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}}^2 \tag{4.106}$$

where $\boldsymbol{N}_{ILS}$ is the solution from the LAMBDA method for the original ILS problem. If the search space with $\chi^2 = \chi^2_{ILS}$ does not contain a solution for Equation (4.103), the search space is incremented by χ^2_{ILS}. The actual search is then performed in two steps. First, all integer vectors that satisfy the unconstrained ILS problem for the current search space are collected. In the next step, all integer vectors that satisfy the inequality from Equation (4.105) are collected:

$$\left\|\hat{\boldsymbol{b}}(\boldsymbol{N}) - \check{\boldsymbol{b}}(\boldsymbol{N})\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\boldsymbol{N})}} \leq \chi^2 - \left\|\hat{\boldsymbol{N}} - \boldsymbol{N}\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}} \tag{4.107}$$

As a result for the ILSQE problem, the integer vector that minimizes Equation (4.103) is selected. The cost of each integer candidate set can be computed by:

$$\omega(\boldsymbol{N}) = \left\|\hat{\boldsymbol{N}} - \boldsymbol{N}\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}} + \left\|\hat{\boldsymbol{b}}(\boldsymbol{N}) - \check{\boldsymbol{b}}(\boldsymbol{N})\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\boldsymbol{N})}} \tag{4.108}$$

In order to find a ILSQE solution the constrained baseline vector $\check{\boldsymbol{b}}(\boldsymbol{N})$ has to be computed, which is computationally expensive. The computation of $\check{\boldsymbol{b}}(\boldsymbol{N})$ and therefore the Least Square with Quadratic Equality (LSQE) problem given by Equation (4.104) can be done using a LSQE solver as described in [195] or by using orthogonal projection as done in [194]. In order to speed up the computation of the ILSQE solution, a lower boundary for the left hand side of the inequality in Equation (4.107) is introduced, that is way more efficient to be evaluated. It can be shown that the following is a valid lower boundary:

$$\frac{\left(\left\|\hat{\boldsymbol{b}}(\boldsymbol{N})\right\| - l_{baseline}\right)^2}{\lambda_{max}} \leq \left\|\hat{\boldsymbol{b}}(\boldsymbol{N}) - \check{\boldsymbol{b}}(\boldsymbol{N})\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\boldsymbol{N})}} \leq \chi^2 - \left\|\hat{\boldsymbol{N}} - \boldsymbol{N}\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}} \tag{4.109}$$

where λ_{max} is the largest eigenvalue of $\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\boldsymbol{N})}$. Only if this lower boundary hold true, the computation of $\check{\boldsymbol{b}}(\boldsymbol{N})$ is required. The complete baseline constrained LAMDBA search can be summarized as follows:

Algorithm 4.1 BC-LAMBDA

Require: Float estimates $\hat{\boldsymbol{N}}, \hat{\boldsymbol{b}}(\hat{\boldsymbol{N}})$ and their covariances $\boldsymbol{Q}_{\hat{\boldsymbol{R}}}, \boldsymbol{Q}_{\hat{\boldsymbol{N}}}, \boldsymbol{Q}_{\hat{\boldsymbol{R}}\hat{\boldsymbol{N}}}$
  Compute $\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\boldsymbol{N})}$
  Compute λ_{max} as the largest eigenvalue of $\boldsymbol{Q}_{\hat{\boldsymbol{b}}(\boldsymbol{N})}$
  Compute the unconstrained ILS solution $\boldsymbol{N}_{ILS}$
  Set initial search space as $\chi^2_1 = \left\|\hat{\boldsymbol{N}} - \boldsymbol{N}_{ILS}\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}}$
  Create an empty set of integer candidates $\Omega = \{\}$
  while Ω is empty **do**
    for all $\boldsymbol{N}$ such that $\left\|\hat{\boldsymbol{N}} - \boldsymbol{N}\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}} \leq \chi^2_k$ **do**
      Compute $\hat{\boldsymbol{b}}(\boldsymbol{N})$
      if $\frac{\left(\left\|\hat{\boldsymbol{b}}(\boldsymbol{N})\right\| - l_{baseline}\right)^2}{\lambda_{max}} \leq \chi^2 - \left\|\hat{\boldsymbol{N}} - \boldsymbol{N}\right\|^2_{\boldsymbol{Q}_{\hat{\boldsymbol{N}}}}$ **then**
        Compute $\check{\boldsymbol{b}}(\boldsymbol{N})$
        Compute $\omega(\boldsymbol{N})$ and add $\boldsymbol{N}$ to Ω
      end if
    end for
    Increase search space $\chi^2_{k+1} = \chi^2_k + \chi^2_1$
  end while
  select candidate set $\boldsymbol{N}$ with smallest $\omega(\boldsymbol{N})$

4.5.4 Precise Point Positioning

Precise Point Positioning (PPP) is a class of positioning techniques that allows decimeter precision in real-time absolute positioning applications if a internet connection is available. Pure PPP approaches suffer from a rather long convergence time of about 20 minutes [196]. PPP provides high precision and accuracy for absolute positions, whereas RTK methods deliver focus on the relative position between base and rover. PPP combines dual-frequency ionosphere delay calibration and carrier smoothing of pseudo-ranges with precise orbit data and clock error corrections. In general, the PPP observation model for the ionosphere-free linear combination of GNSS observations(see Equation (4.18) and (4.19)) at two different satellite frequencies can be expressed as:

$$P^s_{r,LC} = \rho^s_r + c\left(dt_r\left(t_r\right) - dt^s\left(t^s\right)\right) + T^s_r + \epsilon^s_{P,r} \tag{4.110}$$

$$\begin{aligned}\Phi^s_{r,LC} = \rho^s_r + c\left(dt_r\left(t_r\right) - dt^s\left(t^s\right)\right) + T^s_r \\ + B^s_{r,LC} + d\Phi^s_{r,LC} + \epsilon^s_{\Phi,r}\end{aligned} \tag{4.111}$$

where $B^s_{r,LC}$ is the linear combination of two carrier phase biases (see Equation (4.7)) and $d\Phi^s_{r,LC}$ is the linear combination of two carrier phase correction terms. Although, the carrier phase correction terms can be neglected for short baseline applications in RTK setups and differential approaches, they need to be considered for PPP. The correction terms described in [157] model the phase center offset variation from satellite and receiver antennas as well as Earth tidal effects.

While PPP was originally developed for post-processing and is most commonly used in within those applications [160], modern PPP approaches acquire the correction data in quasi real-time over the internet. The precise correction data is provided by the International GNSS Service (IGS) or other GNSS analysis centers [197]. Table 4.3 lists the main characteristics and differences between RTK- and PPP-based approaches. Both approaches are able to provide a high precision real-time navigation solution. While RTK benefits from short warm-up periods (convergence time), centimeter accuracy and can operate with low-cost single frequency receivers only, PPP can operate as standalone without the need of an additional reference station. Another drawback of PPP is the required internet connectivity which is not always available. In [198], combined RTK and PPP approaches are described to merge the benefits of both methods.

Technique	*Number of Frequencies*	*Warm-Up Time*	Real-Time	*Receiver Infrastructure*	*Positioning*	*Precision*
RTK	1	$\leq$ 2 min	yes	Link to Base	relative	centimeter
PPP	2+	$\approx$ 20 min	yes	Internet	absolute	sub-decimeter

TABLE 4.3: Comparison between PPP and RTK positioning.

Since low-cost single frequency GNSS receivers are used within this work, PPP approaches are not discussed in detail, but rather just mentioned for the sake of completeness.

Chapter 5

Implementation

This chapter focuses on important implementation aspects of the previously introduced methods. For this purpose, details of the developed GNSS navigation applications for ROBEX and MIDRAS are presented in Section 5.1. The subsequent sections explain selected features of the software framework on which the developed flight controller FARN is based. The software framework can be split into three parts: The software running on the real-time core Cortex-M4 in Section 5.2, the software running on the application core Cortex-A9 in Section 5.3 and the software required for inter-core communication in Section 5.4. Figure 5.1 illustrates all software components.

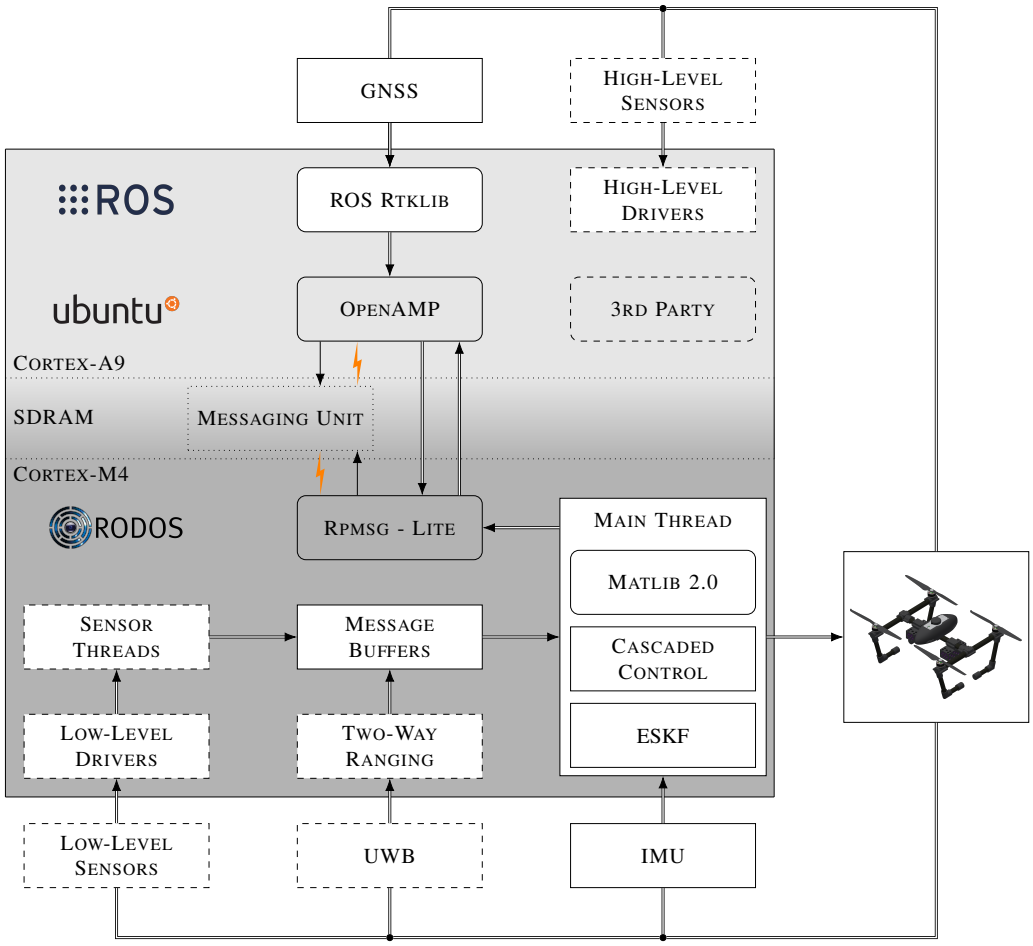


FIGURE 5.1: FARN software framework. Solid boxes indicate standalone modules, while round corners indicate libraries. 3rd party libraries or libraries that are only slightly modified have no filling. Optional components have dashed frames.

5.1 Navigation Applications

In the following, implementation details of the two advanced navigation applications that have been implemented for FARN are described. The first application is a GNSS compass suitable for autonomous UAVs and was developed within the ROBEX project in order to determine a robust and reliable heading solution in an Arctic environment. The second application describes a UWB augmented RTK positioning approach that was researched with the aim of positioning two UAVs relative to each other as precisely as possible within MIDRAS.

5.1.1 GNSS Compass

The GNSS compass combines ESKF attitude estimates, with GNSS code and carrier phase observations in order to determine a reliable heading without requiring a magnetic compass. If precise orbit data is available over NTRIP, it can be integrated into the GNSS compass application, too.

Using two GNSS antennas $\mathcal{A}_1$ and $\mathcal{A}_2$ rigidly mounted on the UAV frame at a distance d, the heading and pitch information of the GNSS baseline can be observed. The baseline pitch corresponds to the UAV roll, however, since gravity vector observations by the accelerometer are more accurate, they are the preferred reference for the corresponding UAV roll updates within the ESKF. The setup with the respective frames is shown in Figure 5.2.

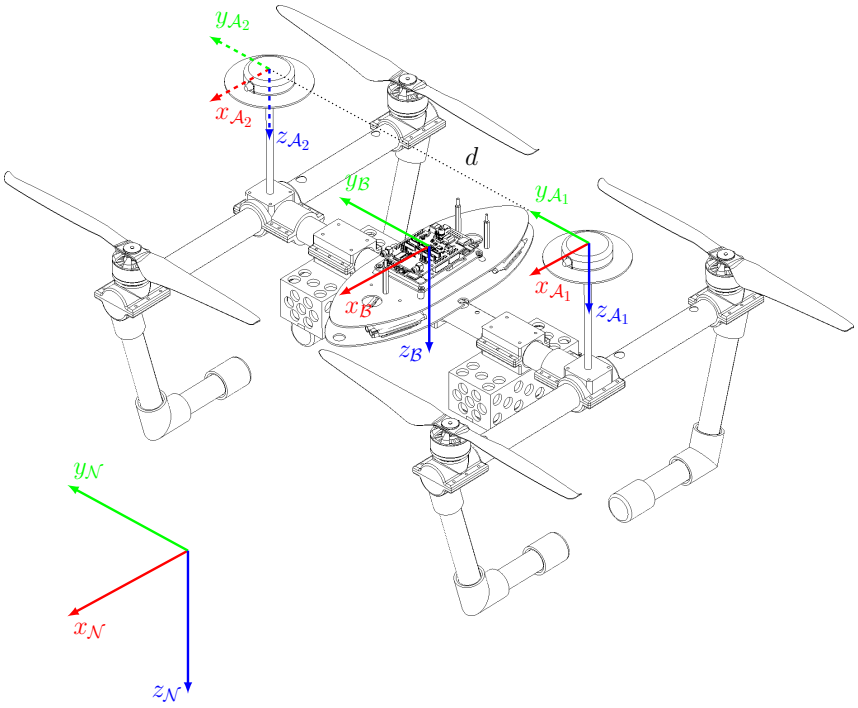


FIGURE 5.2: GNSS compass coordinate frames.

Due to the high computational load for carrier phase-based positioning and the strict hard real-time constraints for a UAV flight controller, a loosely-coupled approach is implemented according to Figure 5.3. The computational expensive tasks run on the Cortex-A9 at 10 Hz. The actual real-time ESKF attitude estimation runs on the Cortex-M4 with up to 952 Hz which is the maximum IMU sample rate.

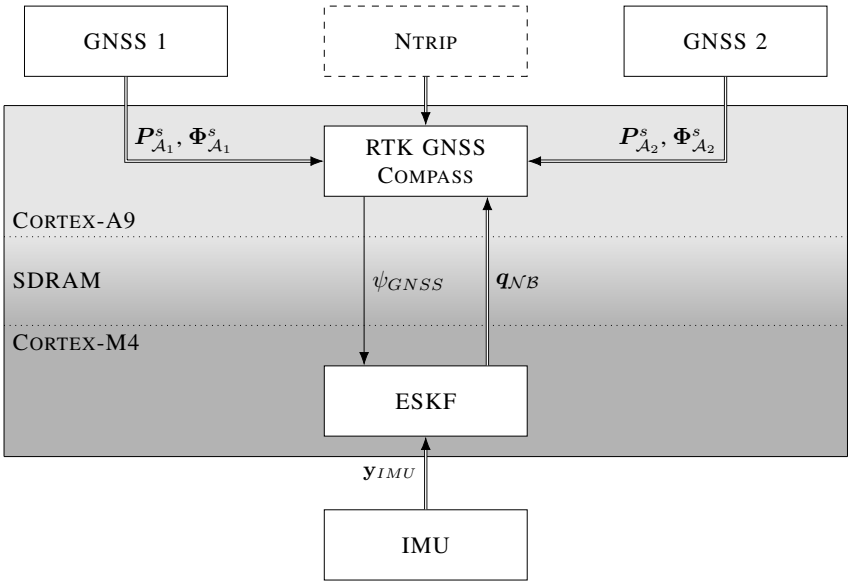


FIGURE 5.3: GNSS compass task distribution on the i.MX6sX.

The baseline vector in the local $\mathcal{A}_1$-frame is then simply given by:

$$\boldsymbol{b}_{\mathcal{A}_1} = \begin{bmatrix} 0 \\ d \\ 0 \end{bmatrix} \tag{5.1}$$

Using the current attitude quaternion estimate $\boldsymbol{q}_{\mathcal{NB}}$ describing the rotation from the navigation frame $\mathcal{N}$ to the body fixed frame $\mathcal{B}$, the baseline can be expressed with respect to $\mathcal{N}$-frame. In this case the $\mathcal{N}$-frame follows the North-East-Down (NED) convention. Since the antenna frames and the $\mathcal{B}$-frame are congruent, it holds:

$$\boldsymbol{b}_{\mathcal{N}} = \boldsymbol{R}\left\{\boldsymbol{q}_{\mathcal{NB}}\right\} \boldsymbol{b}_{\mathcal{A}_1} \tag{5.2}$$

Given raw observation from both antennas $\boldsymbol{P}^s_{(\mathcal{A}_1,\mathcal{A}_2)}$ and $\boldsymbol{\Phi}^s_{(\mathcal{A}_1,\mathcal{A}_2)}$ for code and carrier observations,respectively, the float baseline solution can be calculated as described in Section 4.5.3, where $\mathcal{A}_1$ is considered as moving base and $\mathcal{A}_2$ as rover. The baseline estimate in the NED-frame can be used as both, soft baseline vector constraint as well as hard baseline constraint during the integer fixing step.

Additionally to the hard baseline constraint, an extra validation step is added after integer fixing to prevent wrong fixes to be used in the ESKF. The validation step is introduced as:

$$\|\check{\boldsymbol{b}}_{\boldsymbol{\mathcal{N}}}(\boldsymbol{N}) - \boldsymbol{b}_{\mathcal{N}}\| \leq \tau \tag{5.3}$$

where τ is a threshold parameter describing the accuracy of the baseline vector a-priori information. Using the fixed baseline $\check{\boldsymbol{b}}_{\boldsymbol{\mathcal{N}}}(\boldsymbol{N})$ estimate in the NED-frame, the GNSS UAV heading ψ_{GNSS} can be calculated and integrated into the ESKF state estimate directly using the independent Tait-Bryan yaw update described in Section 3.2.2.3. ψ_{GNSS} is given by:

$$\psi_{GNSS} = \operatorname{atan2}\left(-\check{\boldsymbol{b}}_{\boldsymbol{\mathcal{N}},x}(\boldsymbol{N}), \check{\boldsymbol{b}}_{\boldsymbol{\mathcal{N}},y}(\boldsymbol{N})\right) \tag{5.4}$$

Hereby, it is important to note that ψ_{GNSS} describes the GNSS baseline heading rotated by 90 degrees to match the UAV orientation. All steps necessary for the GNSS compass are summarized in the RTK GNSS Compass algorithm below:

Algorithm 5.1 RTK GNSS Compass

Require: Code $\boldsymbol{P}^{s}_{(\mathcal{A}_1,\mathcal{A}_2)}$ and carrier $\boldsymbol{\Phi}^{s}_{(\mathcal{A}_1,\mathcal{A}_2)}$ raw observations, UAV attitude $\boldsymbol{q}_{\mathcal{NB}}$
  Compute baseline vector in NED-frame $\boldsymbol{b}_{\mathcal{N}}$
  Compute float estimates $\hat{\boldsymbol{N}}, \hat{\boldsymbol{b}}(\hat{\boldsymbol{N}})$ and their covariances
  Apply soft baseline constraints using $\boldsymbol{b}_{\mathcal{N}}$
  Compute BC-LAMBDA solution $\check{\boldsymbol{b}}_{\mathcal{N}}(\boldsymbol{N})$ in NED-frame using $\boldsymbol{b}_{\mathcal{N}}$
  if $\|\check{\boldsymbol{b}}_{\mathcal{N}}(\boldsymbol{N}) - \boldsymbol{b}_{\mathcal{N}}\| \leq \tau$ **then**
    Compute heading ψ_{GNSS} from $\check{\boldsymbol{b}}_{\mathcal{N}}(\boldsymbol{N})$
    ESKF yaw update
  end if

5.1.2 UWB Augmented RTK Positioning

The UWB augmented RTK positioning combines carrier phase-based GNSS observations and radio range information obtained with DS-TWR described in Section 2.2.6.2 in order to estimate the relative position between two agents. Depending on the scenario, either a fixed base station $\mathcal{B}$ or a moving base, e.g. the UAV master $\mathcal{M}$, can be used as GNSS reference. Both scenarios and the respective frames are shown in Figure 5.4.

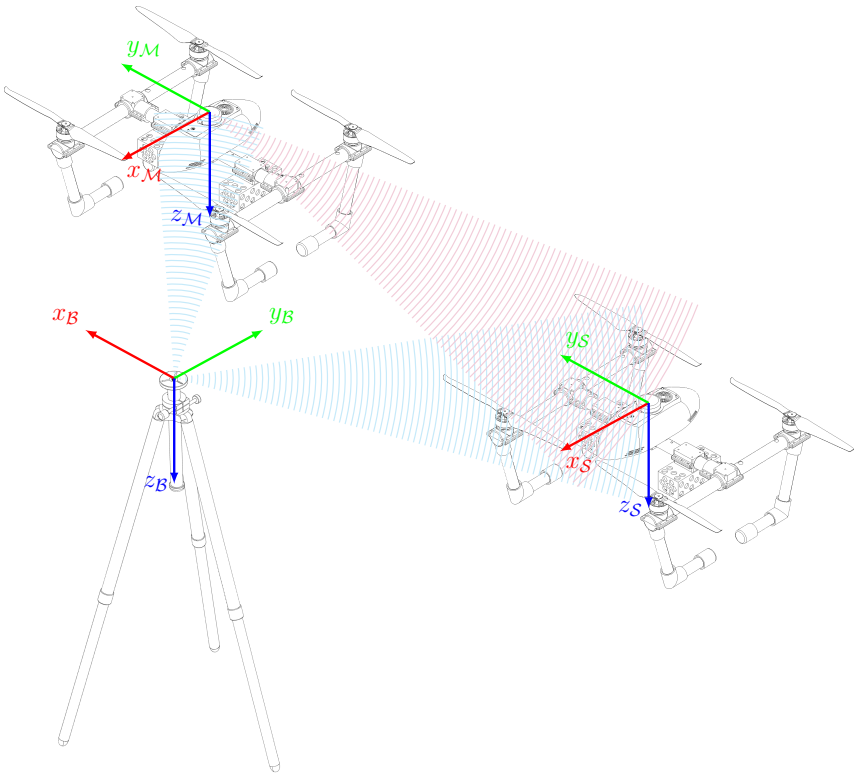


FIGURE 5.4: Two different position estimation concepts: Fixed and moving base. The waves indicate the augmented UWB radio ranging, for a fixed base and a moving base in blue and red, respectively.

Similar to the GNSS compass, the tasks for are distributed among the flight controller processing cores according to their real-time requirements and computational load. Figure 5.5 illustrates the task distribution and how the different components interact. Again, the computationally expensive RTK algorithms are run on the Cortex-A9. Conversely, the attitude estimation and UWB ranging using DS-TWR are considered as hard real-time and therefore handled by the Cortex-M4. While the actual computational load for DS-TWR is low compared to operations performed by the 19-state ESKF, timing constraints are more strict. As a direct consequence of this, a counter intuitive allocation of the respective thread priorities is required. In order to meet the hard real-time requirements for DS-TWR, the thread running the UWB communication needs a higher priority than the most critical task of the flight-controller, namely the attitude estimation and control.

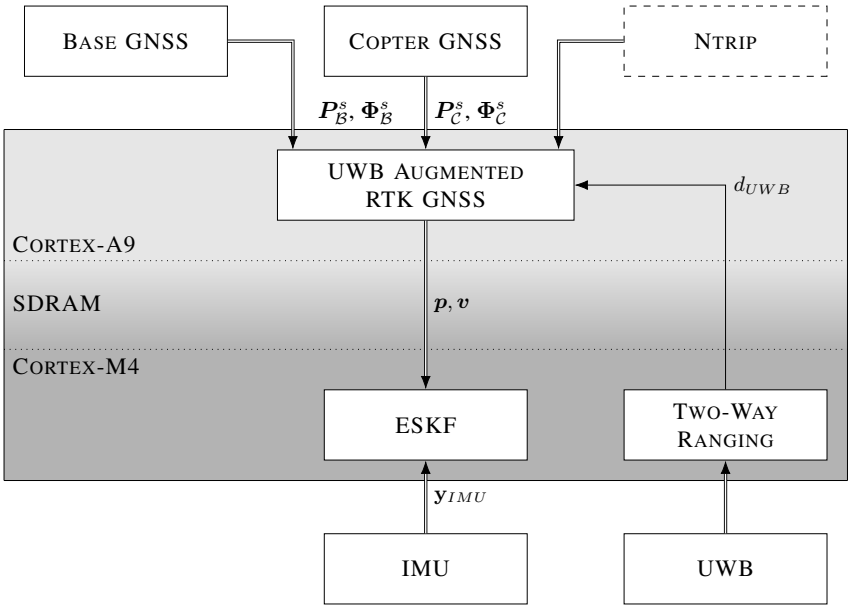


FIGURE 5.5: Task distribution for combined UWB/GNSS positioning on the i.MX6sX.

The ranging information obtained from DS-TWR can be used as baseline length constraint directly according to the algorithms in Section 4.5.3. The verification threshold τ should include UWB ranging inaccuracies and the quality of the UWB antenna delay calibration. The combined UWB/GNSS localization steps are summarized in Algorithm 5.2 below.

Algorithm 5.2 UWB RTK

Require: GNSS code $\boldsymbol{P}^s_{(r,b)}$ and carrier $\boldsymbol{\Phi}^s_{(r,b)}$ raw observations for rover and base
  Compute relative distance $\boldsymbol{d}$ using DS-TWR
  Compute float estimates $\hat{\boldsymbol{N}}, \hat{\boldsymbol{b}}(\hat{\boldsymbol{N}})$ and their covariances
  Apply soft baseline constraints using d_{UWB}
  Compute BC-LAMBDA solution $\boldsymbol{N}, \check{\boldsymbol{b}}(\boldsymbol{N})$
  if $\left| \|\check{\boldsymbol{b}}(\boldsymbol{N})\| - d_{UWB} \right| \leq \tau$ **then**
    ESKF position update
  end if

5.2 Real-time Core

The real-time core implements the actual flight controller application. The main task of this application is the ego-motion estimation and control of the UAV making it the most critical software component of the whole system that needs to be very reliable. In addition to that, the real-time core handles sensors with low-level interfaces, high update rates or hard real-time requirements.

Traditionally, flight controller applications are implemented using a classic software design with a constantly running main loop [41]. Within this main loop a more or less fixed scheduling scheme is used to sequentially poll different sensors and communication interfaces. While this software approach is very robust and predictive, maintaining as well as adding new functionality requires explicit knowledge about hardware operations and fault probabilities as well as sensor and operation specific timing requirements. In contrast, the flight controller application in this work implements an event-based software design based on the RTOS RODOS which is described in Section 5.2.1. Using a RTOS, the flight controller application can be designed in a modular way, allowing an easy integration of various sensor types as well as a high degree of mission specific adoptions and specializations.

The proposed software architecture tries to combine the benefits of an event-based software design using a RTOS with the robustness of the classic flight controller approach. This is achieved by dividing the flight controller application into several threads where a single thread, namely the MAIN-Thread, is able to entirely control the UAV. The MAIN-Thread is responsible for the ego-motion estimation and the control of the UAV at a preferably high frequency. It is designed following the classical software architecture for flight controllers with a single `while`-loop performing typical flight controller tasks. The different tasks are roughly outlined in Figure 5.6 which include: the acquisition of IMU data, the collecting and processing of data and telecommands from other threads, the current state estimation and state machine propagation as well as the actuator control. The current ego-motion estimation is the most CPU-intensive computation step and therefore highly optimized. Implementation details of the ego-motion estimation are described in Section 5.2.2. A single loop iteration without additional sensor data from other threads takes less than 1 ms resulting in a theoretical loop rate of 1 kHz.

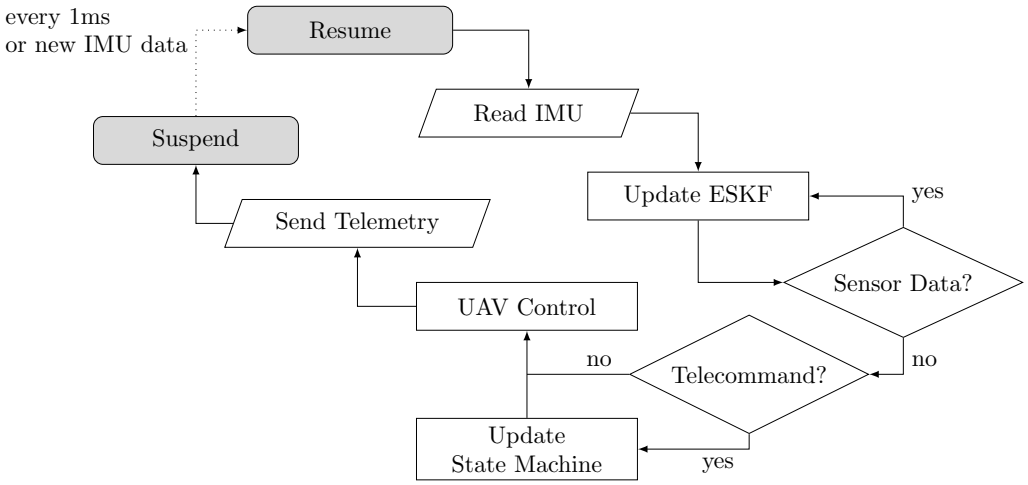


FIGURE 5.6: Simplified flowchart of the MAIN-Thread loop.

The other threads handle the communication with the different sensors and interfaces described in Chapter 2 as well as the communication with the application core. Depending on their timing requirements and characteristics, the remaining threads are prioritized. Scheduling policies as well as the main inter-thread communication mechanism are sketched in Figure 5.7.

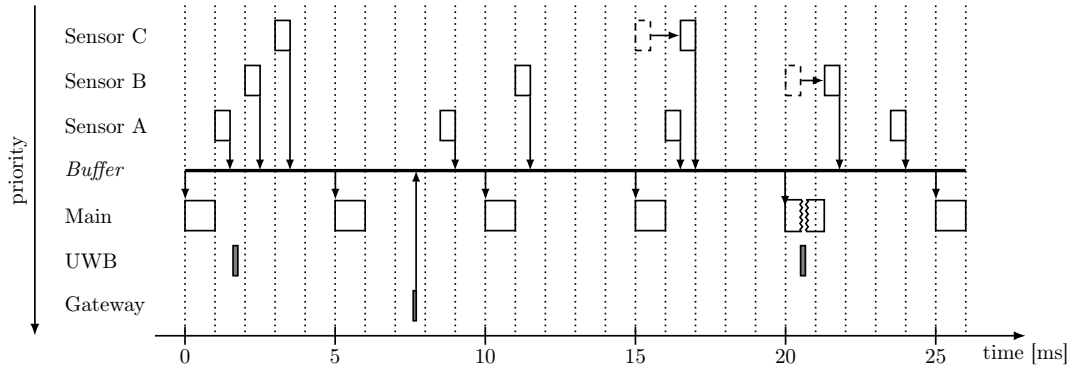


FIGURE 5.7: Scheduling and inter thread communication policies.

The MAIN-Thread has the longest execution time and can only be interrupted by Gateway or Ultra-Wide Band (UWB) events. The inter-core communication gateway is called periodically every second or if the reception of a new message is indicated by a Messaging Unit (MU) interrupt. For active ranging, the UWB-Thread is called every 200 ms initiating a new ranging sequence. For passive ranging, the UWB-Thread is resumed if the reception of a new message is indicated by a UWB module interrupt. Other sensor and communication threads can be either called periodically or resumed if new data is received or a new measurement available. Typical thread periods and wake-up events are listed in Table 5.1. The information gathered by each thread is shared to the MAIN-Thread using message buffers. The MAIN-Thread collects all available information from the respective buffers and updates the ego-motion estimate and its internal state machine accordingly. Since the MAIN-Thread frequency is high, the jitter and delay caused by the message buffer are small and can therefore be neglected. Since all update steps in the ego-motion state estimation algorithm can occur following a fixed update scheme, complex resource protection and management for the ego-motion estimation is not necessary. In a pure event-based approach, the access to the ego-motion estimation framework should be treated as any other resource access and hence protected by semaphores.

Thread	Period [ms]	Wake-up Source
MAIN	1-5	Gyro GPIO
Barometer	33	GPIO
LidarLite	10	n.a.
Battery Monitor	100	n.a.
Remote Control	25	UART RX
Gateway	1000	MU RX
DecaWave	200	RX GPIO

TABLE 5.1: Typical periods and wake-up sources.

5.2.1 Real-Time On-Board Dependable Operating System

The Cortex-M4 real-time core uses the Real-time Onboard Dependable Operating System (RODOS), a RTOS that was designed for embedded systems and application domains demanding high dependability [199, 200]. RODOS was originally developed at the German Aerospace Center (DLR) and is currently maintained and actively developed by the Aerospace Computer Science department of the University of Wuerzburg. It offers real-time priority controlled preemptive multi-threading as can be seen by the scheduling scheme in Figure 5.7. RODOS can run on specific hardware directly or as guest on top of other host operating systems.

For FARN on the i.MX 6 SoloX processor, RODOS was ported to run on top of NXP's MQX RTOS [201]. Similar to RODOS, MQX provides a multi-tasking kernel with a preemptive priority based scheduling, inter-process communication and synchronization mechanisms as well as IO-drivers and hardware abstraction. MQX builds upon processor and board specific libraries from the chip manufacturer NXP. The different abstraction layers underneath the flight controller application are shown for reference in Figure 5.8.

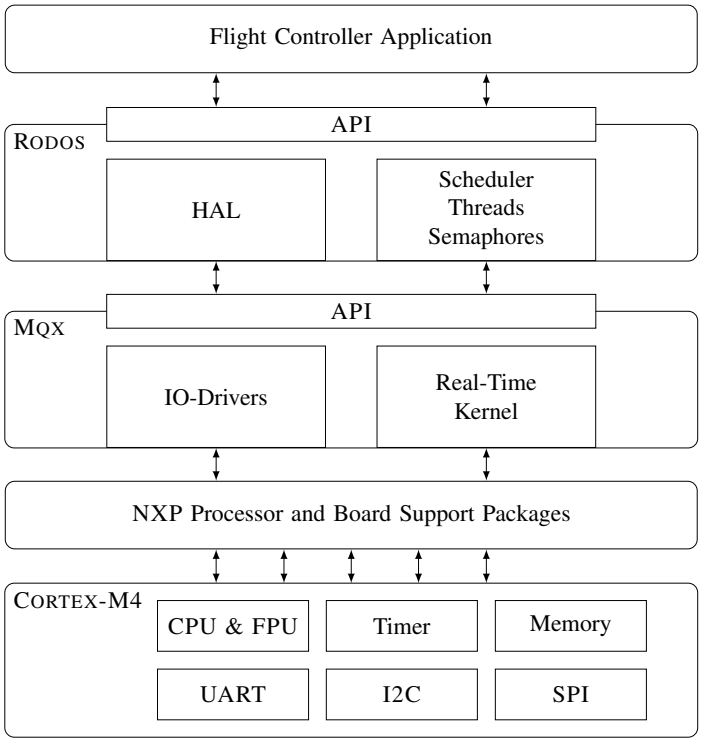


FIGURE 5.8: Abstraction layers.

One benefit of RODOS is its great portability to other platforms allowing to easily migrate the flight controller software to different UAV platforms. The flight controller application can run on any RODOS platform with according peripheral interfaces and drivers. In the context of this work, many UAV prototypes were developed and successfully flown using different flight controller boards based on the Cortex-M4 family from STMicroelectronics (STM32F4) [202]. Having a single core only, micro-controllers from the STM32F4 family offer a reduced system complexity compared to the i.MX 6 SoloX. However, the heterogeneous i.MX 6 SoloX processor is preferred for flight controller applications that implement real-time RTK processing and require a high bandwidth data exchange between both cores.

5.2.2 ESKF Implementation

The execution time of a single iteration of the flight controllers MAIN-Thread depends mostly on the efficiency of the Error-State Kalman Filter (ESKF) implementation. In order to maximize the computation speed of a ESKF iteration, different optimization steps are taken. If the accuracy is secondary, arithmetic operations on the Cortex-M4 should be performed using the built-in hardware FPU and single precision floats only. In fact, the computation of double precision operations is more efficient using CPU software emulation than on a single precision FPU [203]. Despite the use of the Cortex-M4 hardware FPU and compiler optimization flags for specific ESKF functions, additional optimization steps can be considered. This section describes the ESKF implementation in detail. The computation speed for a single filter propagation step of a naive implementation is compared to two implementations with a different degree of optimization in Figure 5.9.

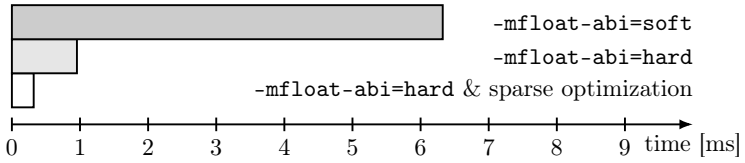


FIGURE 5.9: ESKF computation speed with different optimizations.

5.2.2.1 MATLIB 2.0

The RODOS support libraries include MATLIB, a lightweight and efficient library for commonly used mathematical operations in robotic, avionic and satellite applications. The RODOS MATLIB is extended within this work to include generic matrix and vector sizes for arbitrary data types and floating point precisions. At the same time, the API of the updated MATLIB 2.0 library remains compatible with its previous version. The class diagram in Figure 5.10 outlines class dependencies between the old API (gray) and the newly added generic classes (white).

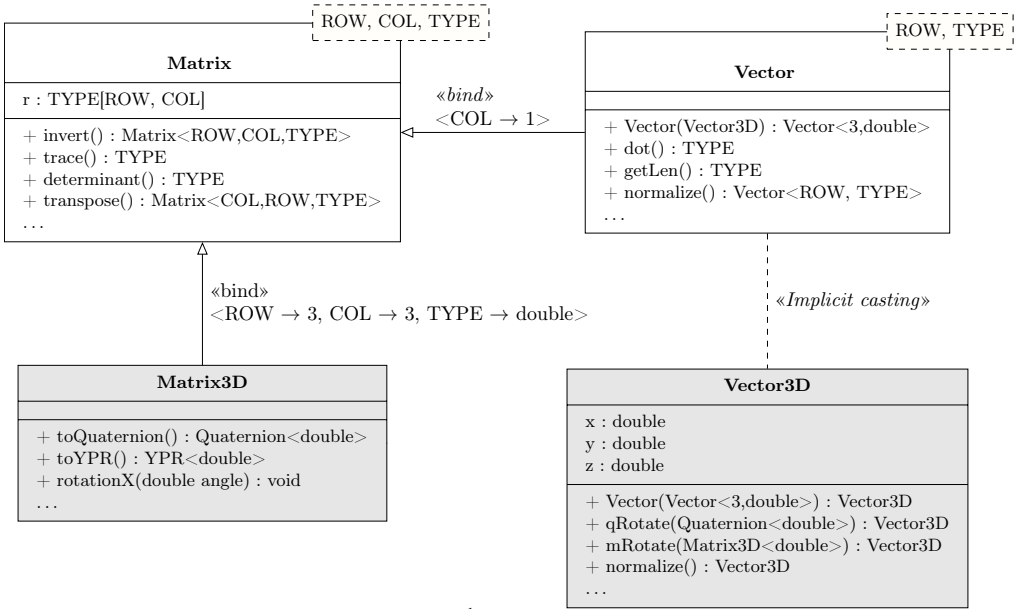


FIGURE 5.10: Simplified MATLIB 2.0 class diagram.

The MATLIB 2.0 implements different representations of rotations following the definitions in Section 3.1. Additional functions, such as the computation of the matrix inverse as well as the matrix trace and its determinant are added, too, according to the suggestions and implementation hints found in [204]. The ESKF implementation builds upon the MATLIB 2.0 library using generic matrix sizes with single precision floating point arithmetic.

5.2.2.2 Matrix Optimization

Examining the ESKF equations introduced in Chapter 3, it is noticeable that the different measurement Jacobian matrices $\boldsymbol{H}$ as well as the system transition matrix $\boldsymbol{F}_{\delta\boldsymbol{x},k}$ are sparse matrices with several entries that are equal to one and expose therefore a huge potential for optimization. Additionally, the covariance matrix $\boldsymbol{P}$ is symmetric. As an example, only non-zero elements of $\boldsymbol{F}_{\delta\boldsymbol{x},k}$ from Equation (3.65) are indicated with ■ below:

$$\boldsymbol{F}_{\delta\boldsymbol{x},k} = \begin{bmatrix} ■ & ■ & & & & & & & & & & & & & & & & \\ & ■ & ■ & & & & & & & & & & & & & & & \\ & & ■ & ■ & & & & & & & & & & & & & & \\ & & & ■ & ■ & & & & & & & & & & & & & \\ & & & & ■ & ■ & & & & & & & & & & & & \\ & & & & & ■ & ■ & & & & & & & & & & & \\ & & & & & & ■ & ■ & & & & & & & & & & \\ & & & & & & & ■ & ■ & & & & & & & & & \\ & & & & & & & & ■ & ■ & & & & & & & & \\ & & & & & & & & & ■ & ■ & & & & & & & \\ & & & & & & & & & & ■ & ■ & & & & & & \\ & & & & & & & & & & & ■ & ■ & & & & & \\ & & & & & & & & & & & & ■ & ■ & & & & \\ & & & & & & & & & & & & & ■ & ■ & & & \\ & & & & & & & & & & & & & & ■ & ■ & & \\ & & & & & & & & & & & & & & & ■ & ■ & \\ & & & & & & & & & & & & & & & & ■ & \end{bmatrix}$$

The code for sparse matrix optimized operations is generated using MATLAB's `symbolic` toolbox. Symbolic representations of each sparse matrix as well as the system covariance matrix $\boldsymbol{P}$ are obtained and the respective ESKF equations are evaluated in MATLAB. A simple bash script is used to convert the symbolic MATLAB output to MATLIB 2.0 compatible C++ code. As an example, let's consider the sparse transition matrix $\boldsymbol{F}$ and the system state covariances $\boldsymbol{P}$. The covariance propagation without system noise can be written as:

$$\boldsymbol{P}_{k+1} = \boldsymbol{F} \cdot \boldsymbol{P}_k \cdot \boldsymbol{F}^\top$$

The MATLIB implementation and part of the optimized code are shown below.

```
1  Matrix<18,18,float> P, F;          // Sparse matrices
2  // Matlib 2.0 implementation
3  P = F * P * F.transpose();      // Time complexity O(n^3)
4  // Optimized code
5  // Time complexity O(m*n), m is number of entries!=0, m>n
6  FP.r[0][0] = P.r[0][0] + F.r[0][3] * P.r[3][0];
7  FP.r[0][1] = P.r[0][1] + F.r[0][3] * P.r[3][1];
8  FP.r[0][2] = P.r[0][2] + F.r[0][3] * P.r[3][2];
9  ...
10 P.r[0][0] = FP.r[0][0] + FP.r[0][3]*(F.r[0][3]);
11 P.r[0][1] = FP.r[0][1] + FP.r[0][4]*(F.r[1][4]);
12 P.r[0][2] = FP.r[0][2] + FP.r[0][5]*(F.r[2][5]);
13 ...
```

LISTING 5.1: Covariance Propagation.

5.3 Application Core

The application core runs mission specific and computationally expensive tasks with soft real-time requirements. It uses the open-source operating system Ubuntu based on the Linux-kernel 4.1 [205] with custom extensions for enhanced inter-core communication and a custom device tree. The different tasks on the application core are implemented using the Robot Operating System (ROS) and are described in Section 5.3.1. Precise carrier phase-based GNSS positioning is implemented based upon the RTKLIB which is outlined in Section 5.3.2. Other important third-party software that is used is discussed briefly in Section 5.3.3.

5.3.1 Robot Operating System

The application core software is based on the open-source Robot Operating System (ROS), to be more precise, the Kinetic version [206]. In contradiction to its name, ROS is not an operating system in the traditional sense, but rather a meta-operating system that implements a communication layer allowing collaboration within a heterogeneous computer cluster. Despite, ROS also provides typical operating system services such as hardware abstraction, low-level device control and inter-process messaging. ROS includes additionally a great number of software libraries implementing commonly used robotic applications such as path planning, obstacle avoidance, machine perception, localization and mapping as well as data visualization.

The ROS communication layer implements a simple IP-based publish-subscribe mechanism allowing to easily setup the communication between a UAV swarm and a ground station. Each application task that provides or requires data from other tasks is run as a separate ROS node that publishes or subscribes to a specific topic. Topic origins are identified using automatic namespaces based on the respective machine's host name. The complete communication setup is sketched in Figure 5.11. The `roscore` node runs on a drone in order to allow operations without base. The drone running `roscore` is referred to as *master* and provides a wireless access point. The ROS nodes are indicated with circles, the gray circle implements the real-time core communication gateway described in Section 5.4.

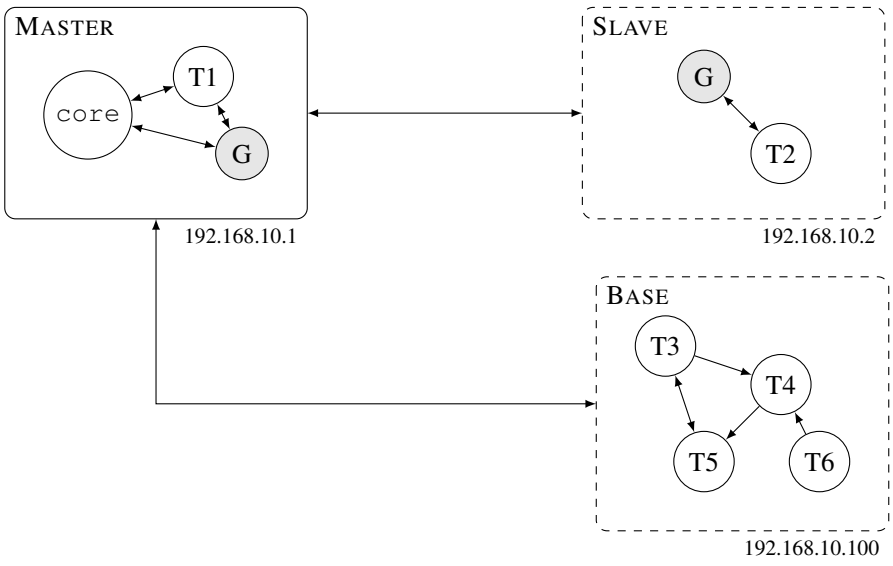


FIGURE 5.11: ROS communication setup.

5.3.2 RTKLIB

The RTKLIB [207] is the core of the precise carrier phase-based GNSS positioning. The library implements a great part of the navigation algorithms described in Section 4, in particular the standard GNSS EKF and the simple carrier phased-based positioning without any constraints. Within this work, the RTKLIB is extended to include the different baseline constraint techniques described in Sections 4.5.3.6 and 4.5.3.7. The ROS nodes `compass` and `relpos` wrap the RTKLIB server and implement the advanced navigation applications described in Section 5.1.1 and 5.1.2, respectively.

The respective ROS nodes as well as their communication dependencies are outlined for the GNSS compass application and the relative UAV swarm positioning in the two figures below. The topics and nodes are summarized in Table 5.2. The namespace of each topic indicated by the first tag describes the source host of the respective topic.

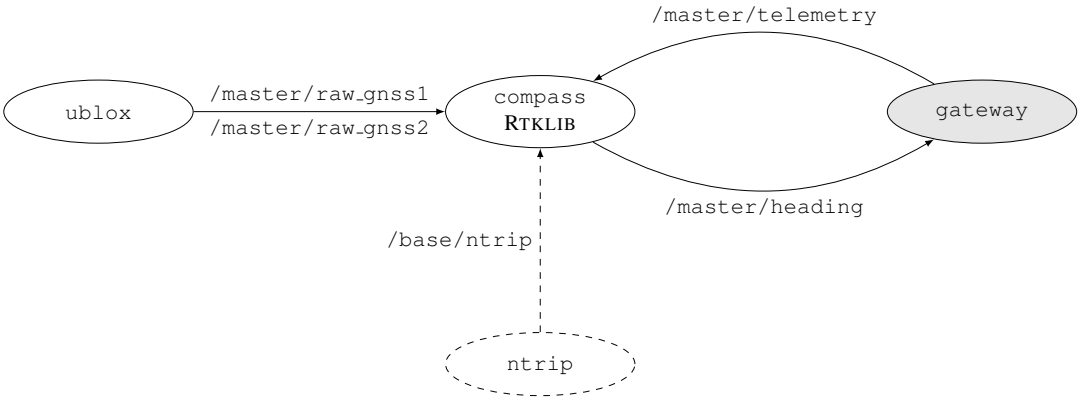


FIGURE 5.12: GNSS compass application.

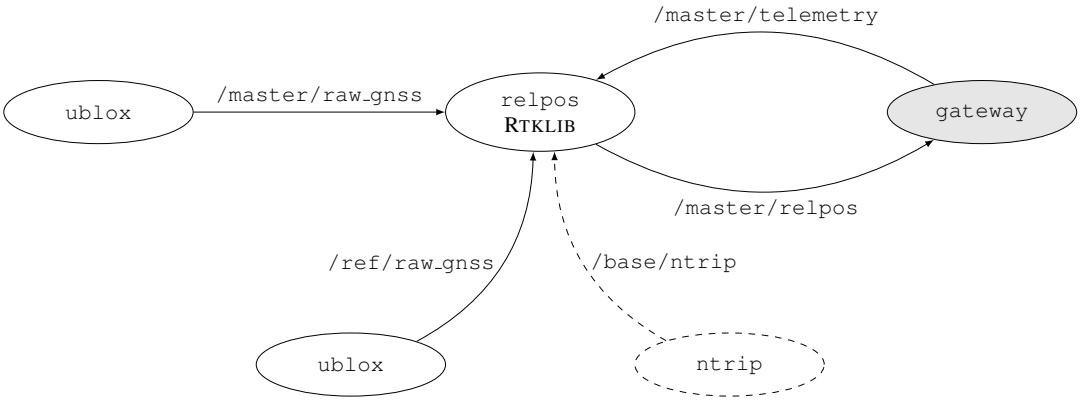


FIGURE 5.13: Relative positioning application.

Node	Function	*publish*	Description
`compass`	GNSS compass	`/heading`	GNSS heading solution
`relpos`	relative positioning	`/relpos`	relative position solution
`gateway`	inter-core communication	`/telemetry`	UAV state & sensors
`ublox`	read GNSS sensor	`/raw_gnss*`	GNSS raw data
`ntrip`	provide correction data	`/ntrip`	correction data

TABLE 5.2: ROS nodes required for RTK GNSS applications.

5.3.3 Third Party

In addition to the RTKLIB, other third part software is used on the application core. The software is mostly but not strictly limited to drivers supporting sensors that are interfaced with the application core directly, e.g. a ROS node for the FLIR Lepton thermal camera [208] or the Intel RealSense library and ROS wrapper for the Visual-SLAM system T265 [209]. It should be noted that both drivers need to be targeted and compiled for the i.MX 6SoloX platform specifically. Having only a USB 2 port, the T265 is limited to the navigation data output only, while streaming the raw images is not supported. In addition to driver libraries, the World Magnetic Model is used to compensate for magnetic declination and CHRONY for multi platform clock synchronization.

World Magnetic Model The World Magnetic Model [210] is integrated into the ROS RTKLIB wrapper. It computes and outputs the magnetic declination for a given location. For the case that the GNSS compass is used together with a magnetic compass, the magnetic declination is corrected for directly within the GNSS compass algorithm by applying a declination corrected baseline constraint.

CHRONY CHRONY [211] implements the Network Time Protocol (NTP) and can synchronize the respective system clock to different time sources, e.g. NTP servers and GNSS receivers using National Marine Electronics Association (NMEA) sentences or providing a Pulse Per Second (PPS) interface. To provide an accurate and synchronized time across all platforms, the clock referencing hierarchy according to Figure 5.14 is implemented.

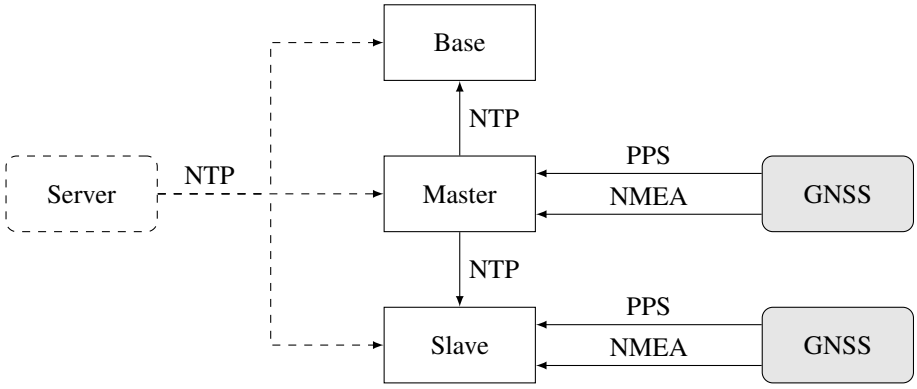


FIGURE 5.14: Chrony time synchronization.

All drones are synchronized to their respective GNSS receiver. The *master* is configured as a local NTP server providing time information for all clients connected to its access point. If the drones are used in a GNSS denied environment and a internet connection is available, additionally public NTP servers can be included. The estimated clock error between the different platforms if GNSS is available is below 2 μs, while the accuracy drops to a few milliseconds otherwise.

5.4 Inter-core Communication

Inter-core communication of heterogeneous multi core processors is commonly realized using shared memory. Both operating systems, ROS on the application core and the embedded RODOS on the real-time core, use a publish/subscribe mechanism for inter process communication. Therefore, the goal is to implement a inter-core communication using shared memory that allows to easily share information between different processes running on the different cores.

The proposed communication implementation can be divided into four different layers roughly following the OSI model as shown in Figure 5.15: The physical layer, the media access control layer, the transport layer and the data layer. The function of each layer as well as its implementation is described separately in the following sections. The three low-level layers are implemented according to the Remote Processor Messaging (RPMSG) standard [212, 213]. The top communication layer was developed within this work.

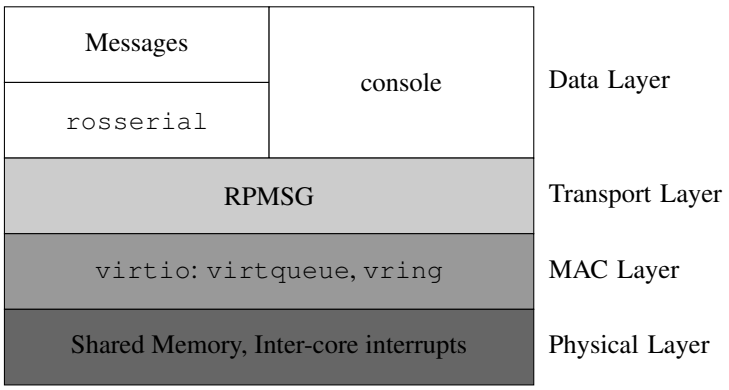


FIGURE 5.15: Abstractions layers for inter-core communication.

5.4.1 Low-level Layers

A great part of the low-level layers is distributed as a part of the Linux kernel or available within NXP's RPMSG-Lite implementation for small MCUs [214]. The physical layer was adapted to match the flight controllers memory layout on both cores. The MAC and the transport layer are part of the Linux kernel, but needed to be ported for the real-time core to RODOS on MQX.

Physical Layer The physical layer represents the shared memory and the inter-core Messaging Unit. Using the Linux device tree, part of the available SDRAM is reserved for inter-core communication. The reserved memory section is indicated to the real-time core in the according linker script. The Messaging Unit is responsible for signalizing the end of a shared SDRAM write access from one core by triggering an interrupt at the other core.

MAC Layer The media access control layer is implemented using `virtio`, an abstraction layer for devices on paravirtualized platforms [215]. The core of `virtio` provides `virtqueue`, a transport abstraction, and `vring`, a transport implementation. The `vring` implementation is based on a single-writer single-reader circular buffer allowing to realize a simple single core-to-core communication without the need for additional synchronization mechanisms.

Transport Layer The transport layer is implemented using RPMSG.

The Linux implementation is based on OpenAMP [213]. The OpenAMP RPMSG implementation allows to create a single communication channel `Y` between two cores with multiple endpoints `X`. The endpoints are exposed to the user space as `/dev/rpmsg_eptX.Y` using the Linux `sysfs` interface for RPMSGs [216].

The RODOS on MQX implementation is based upon NXP's RPMSG-Lite [214]. Here, the RPMSG-Lite communication interface is integrated into the RODOS hardware abstraction layer. A channel endpoint can be created and used like any other interface in RODOS as shown in the example below:

```
1  // message buffers
2  char buffer[MAX_LEN];
3  const char message[] = {"Hello World\n"};
4
5  // RPMSG Interface
6  RODOS::HAL_RPMSG ept(RPMSG_IDX1);   // RPMSG endpoint 1
7  ept.init();
8  // writing
9  ept.write(message, sizeof(message));
10 // reading
11 if(ept.isDataReady()){
12    size_t rec = ept.read(buffer, MAX_LEN);
13 }
```

LISTING 5.2: RODOS HAL RPMSG example.

5.4.2 Data Layer

The data layer implementation uses two different RPMSG endpoints. The first endpoint is simply used unidirectionally in order to display debug and notification messages using the RODOS built-in `PRINTF` function. The second endpoint is used for bidirectional messaging between RODOS threads and ROS nodes. Therefore, the ROS package `rosserial` [217] transmitting ROS serialized messages over a character device, e.g. a serial port or a network socket, is adapted to work with RPMSG endpoints. On the RODOS side, `rosserial` can be used directly or with the RODOS gateway mechanisms. The complete communication setup is summarized in Figure 5.16. It should be noted, that the developed RODOS `rosserial` port can be used with other interfaces, too, e.g. UART or WiFi UDP.

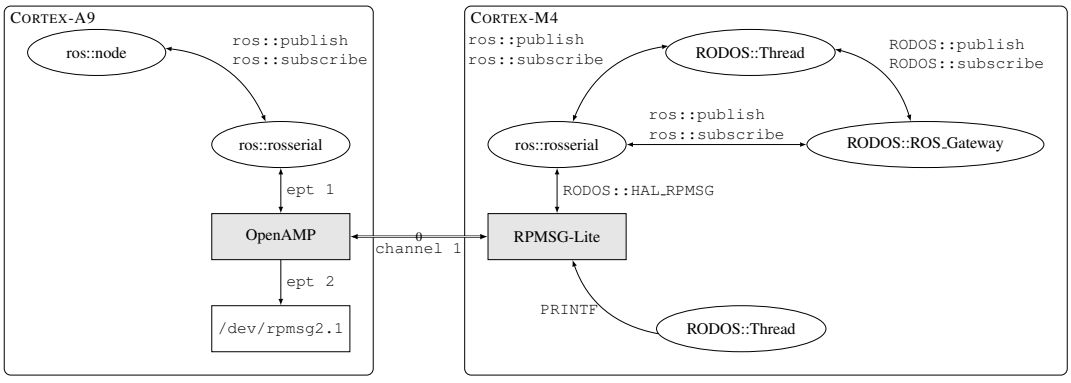


FIGURE 5.16: Complete communication setup.

For reference, an example usage of the ROS-RODOS gateway is shown below. For the corresponding ROS implementation on Linux, no changes from the standard approach are required.

```
#include "rodos.h"
#include "ros_gateway.h"
#include "ros.h"
#include "std_msgs/Float64.h"          // serialized ROS Message Type

ros::NodeHandle nh;                    // from ros::rosserial
RODOS::HAL_RPMSG ept(RPMSG_IDX1);      // RPMSG endpoint 1
ROS_Gateway ros_gw(&ept, &nh);         // Gateway between ROS <-> RODOS

// ROS_Topics inherit from RODOS::Topic
// The ROS_Topics use the ROS datatype std_msgs::Float64
// The first parameter is used as RODOS message ID
// The second parameter is used for ROS message advertising/subscribing
ROS_Topic<std_msgs::Float64> in_test_T (100, "ros_rodos_topic");
ROS_Topic<std_msgs::Float64> out_test_T (101, "rodos_ros_topic");

// ROS Topic Subscriber
static CommBuffer<std_msgs::Float64> in_test_buffer;
static Subscriber sub01(in_test_T, in_test_buffer, "GatewayTest");

class Gateway_Test: public Thread {
public:
  Gateway_Test() : Thread("GatewayTest") {}

  void init() {
    ept.init();
    ros_gw.init();
    ros_gw.addPublisher(&out_test_T);
    ros_gw.addSubscriber(&in_test_T);
  }

  void run() {
    std_msgs::Float64 msg;
    TIME_LOOP(0, 10 * MILLISECONDS){
      // receive a ROS Message with RODOS mechanism
      if(in_test_buffer.getOnlyIfNewData(msg)){
        // Output on RPMSG endpoint 2
        PRINTF("Got data: %lf\n", msg.data);
        msg.data -= 1.0;
      }
      // publish a ROS Message with RODOS mechanism
      out_test_T.publish();
    }
  }
} gateway_test;
```

LISTING 5.3: Gateway RODOS example.

Chapter 6

Evaluation

In this chapter, the performance of key components of the developed flight controller, namely its ego-motion estimation framework as well as the implemented control algorithms are evaluated. Since the flight controller development was targeted towards specific mission scenarios within the ROBEX and the MIDRAS projects, mission critical flight controller components are analyzed separately. Hereby, the focus is particularly on the performance of the GNSS compass in an Arctic environment along with the precise UAV navigation using different RTK setups.

6.1 Ego-motion Estimation and Control

The ego-motion estimation and control frameworks are evaluated within two different experiments. First, the long term stability of the attitude estimation is inspected. Next, the performance of the full state estimation is compared indoor to an optical tracking system. Simultaneously, the control behavior of the cascaded control architecture is assessed.

6.1.1 Long Term Stability

In order to proof the long term stability of the implemented ESKF for ego-motion estimation, temperature stabilized IMU raw data is collected for more than 12 hours at 200 Hz. For each sensor and each axis, more than 10 million samples are recorded within this long term experiment. The estimated angles are shown together with the IMU temperature in Figure 6.1.

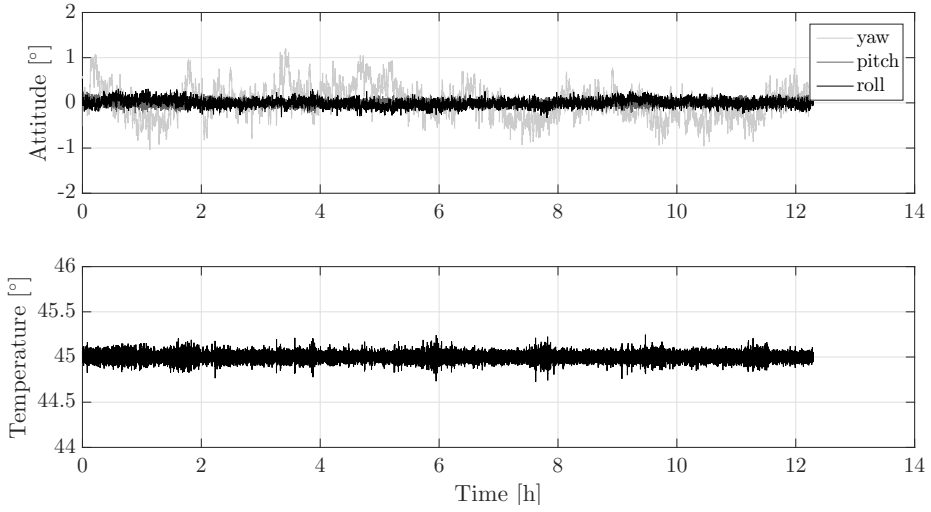


FIGURE 6.1: Long term attitude estimation and temperature control.

The attitude estimate is calculated offline using the embedded ESKF implementation. The required covariances, random walk parameters and bias instabilities are obtained as described in Section 2.2.2.5.

While the yaw estimate error fluctuates between $\pm 1°$, both, the absolute pitch and roll errors remain well below $\pm 0.25°$. The implemented temperature control keeps the IMU temperature stable at $45°C$. Inspecting the discrete attitude error probability distribution given in Figure 6.2 below, the attitude estimates can be described by a Gaussian distribution. Their respective RMSEs are listed in Table 6.1. The higher RMSE of the yaw axis is caused by noisy magnetometer readings. Despite the temperature control, a gyroscope bias change can be observed by comparing the ESKF bias estimate to the initial bias guess in Figure 6.3.

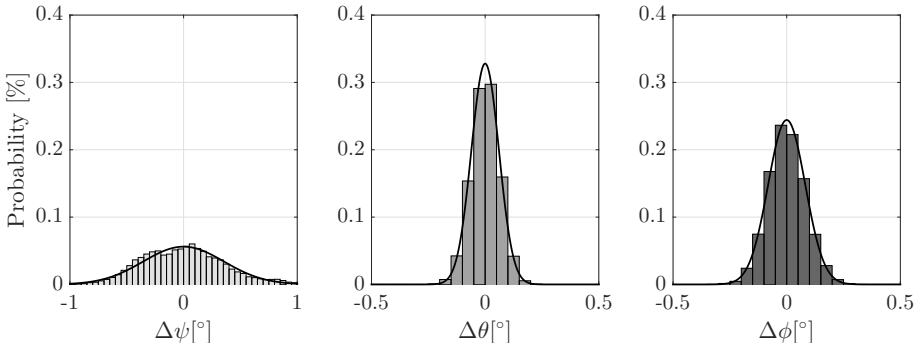


FIGURE 6.2: Discrete distribution and Gaussian approximation.

	Yaw ψ	Pitch θ	Roll ϕ
RMSE	0.37°	0.06°	0.08°

TABLE 6.1: Root mean square errors for attitude estimates.

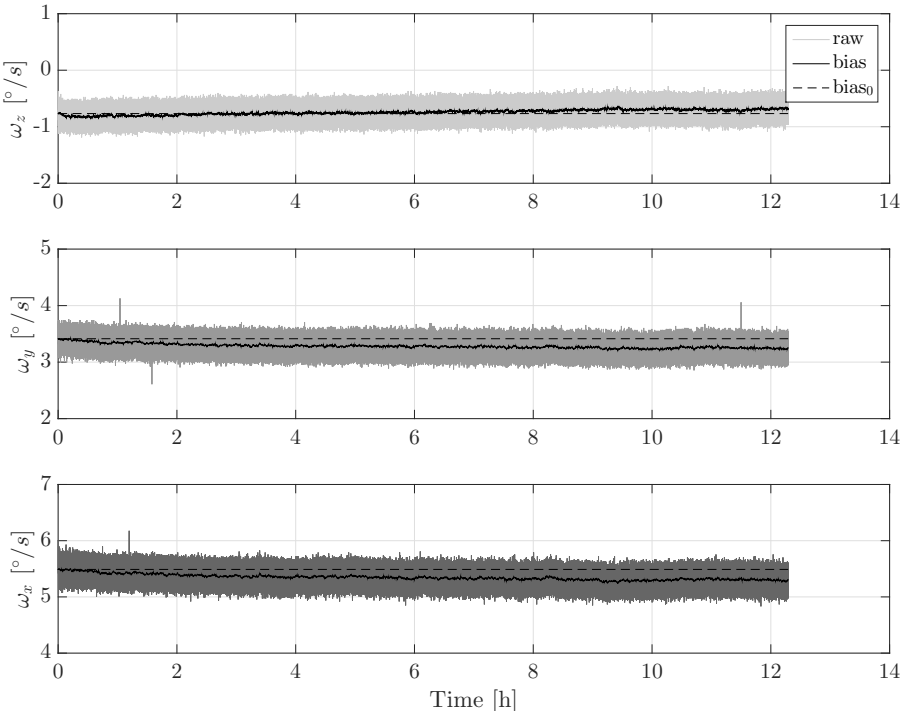


FIGURE 6.3: Gyroscope raw data, computed bias and initial estimate.

Since there are neither position nor velocity measurements available, their ESKF estimates suffer from significant drift caused by the integration of the noisy IMU measurements. Nevertheless, the experiment proofs the ESKF long term stability and its reliability. The estimated attitude RMSEs should be considered as a lower boundary, since in flight vibrations cause additional noise terms, especially for the roll and pitch estimates.

6.1.2 Indoor Flight

Next, the complete ego-motion estimation framework and the control architecture are evaluated. For state estimation, the IMU and the magnetometer, the barometer as well as the position estimates of the Intel RealSense T265 are used. Figure 6.4 summarizes the flight controller setup for this experiment. The estimated pose is compared to data captured using the optical tracking system described in Section 2.2.8.1. The UAV is manually started and commanded to follow a house-like pattern and land afterwards. The flight pattern and the drone are shown in Figure 6.5.

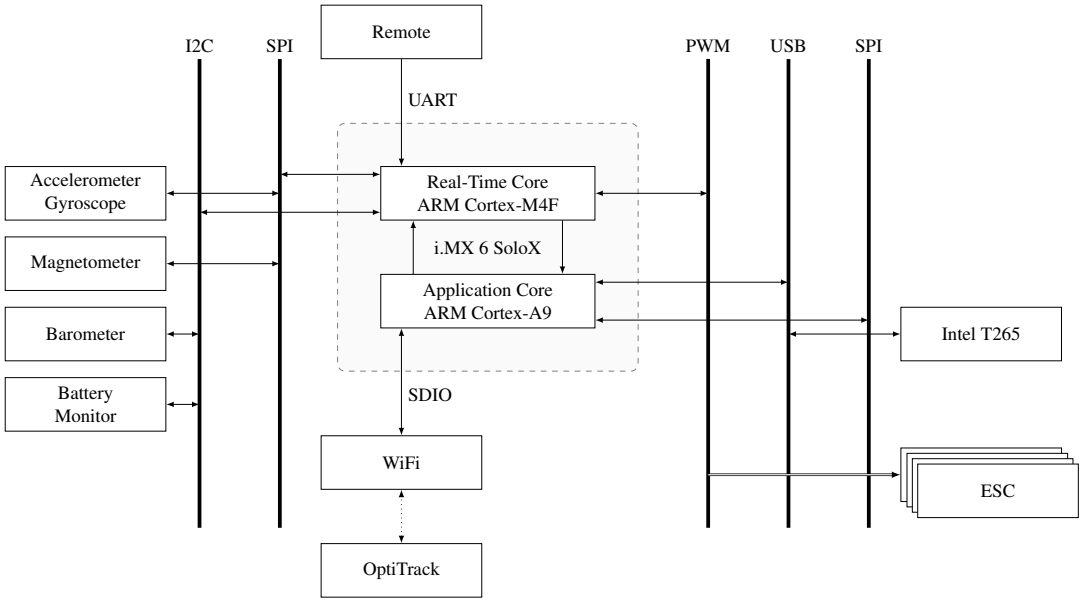


FIGURE 6.4: Flight controller setup for the indoor experiment.

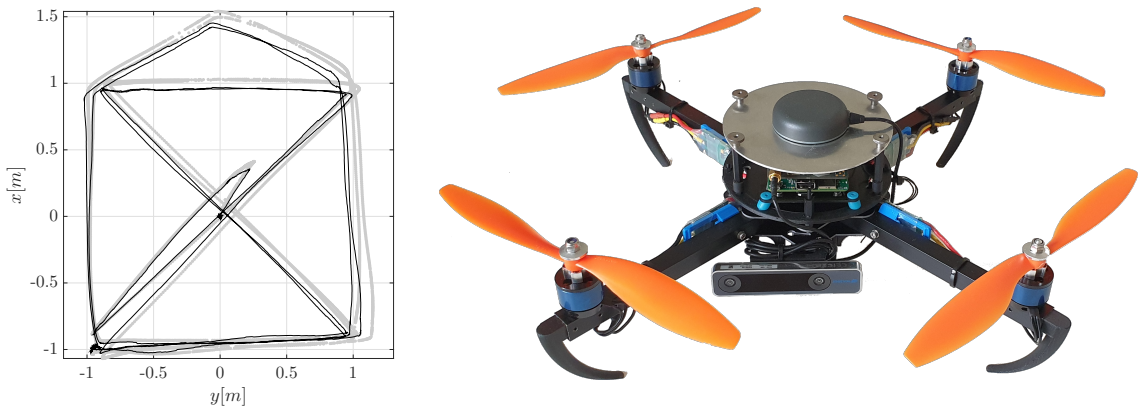


FIGURE 6.5: Flight pattern and UAV for the indoor experiment.

Since the tracking system does not provide velocity information, the velocity is derived from the optical position observations. Hence the reference velocity provides a rather low accuracy. Additionally, the attitude estimation relies heavily on the marker geometry. Due to limited mounting possibilities, the markers are mounted very close to each other which causes inaccuracies in the attitude reference tracking, too. Nevertheless, the optical tracking system is a reliable reference evaluating the ego-motion estimation.

Figures 6.6, 6.7 and 6.8 show the estimated motion in x, y and z direction, respectively. The estimated motion is compared to the optical tracking reference. The respective inputs for the cascaded control are also shown.

The estimated positions agree well with the position observations from the optical tracking system. The average error is below 6 cm with a maximum error of 22.5 cm. It should be noted that neither possible rotational offsets between the two positioning system nor the scaling errors are compensated for. The velocity observations of both systems, the ego-motion estimation framework and the optical tracking system are also in close agreement. Major derivations are caused by the rather simple velocity estimation from the optical tracking system. The attitude observations of optical tracking system are clearly more noisy than the ESKF estimated attitude. Nevertheless, the estimated attitudes coincide with each other, especially for larger tilting maneuvers.

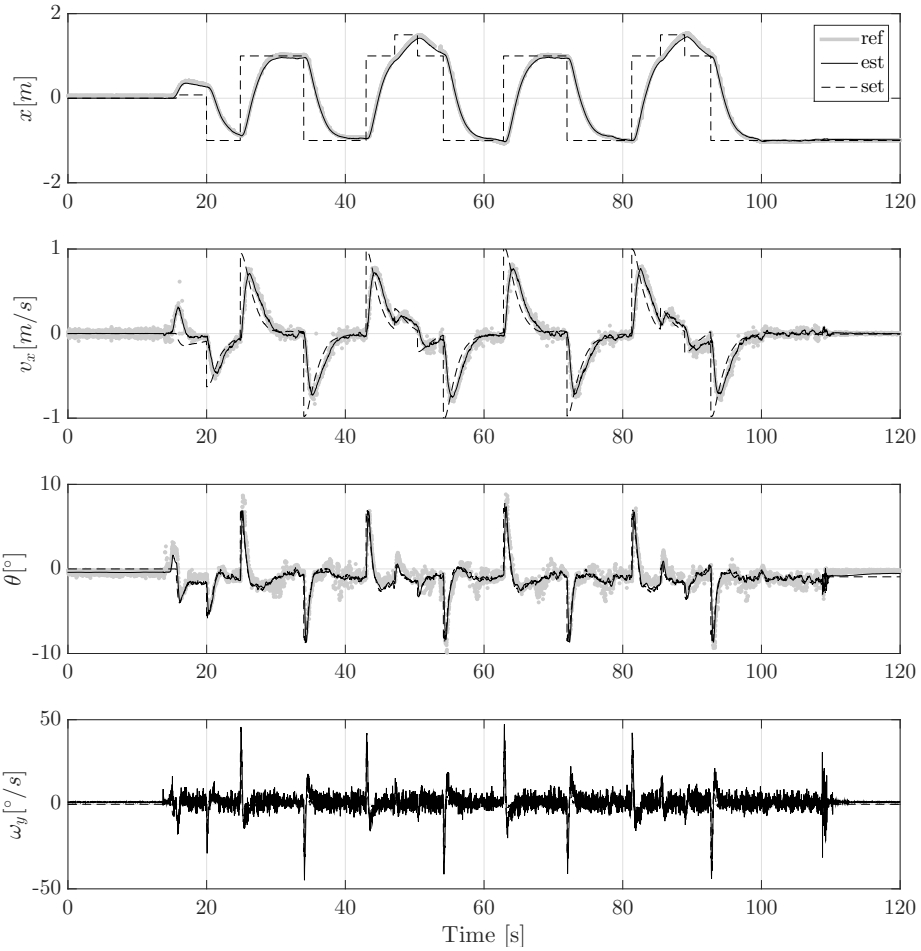


FIGURE 6.6: State estimation and control in x direction.

The real world control behavior corresponds very well to the system behavior observed in the simulation in Section 3.3. For the horizontal control, it can be observed that the controller cascades have a more aggressive response towards the most inner cascade which controls the rotational speed. This allows for a very precise horizontal control without significant position overshoot. The experiment proves that both subsystems, the ego-motion estimation framework and the implemented control architecture, perform very well.

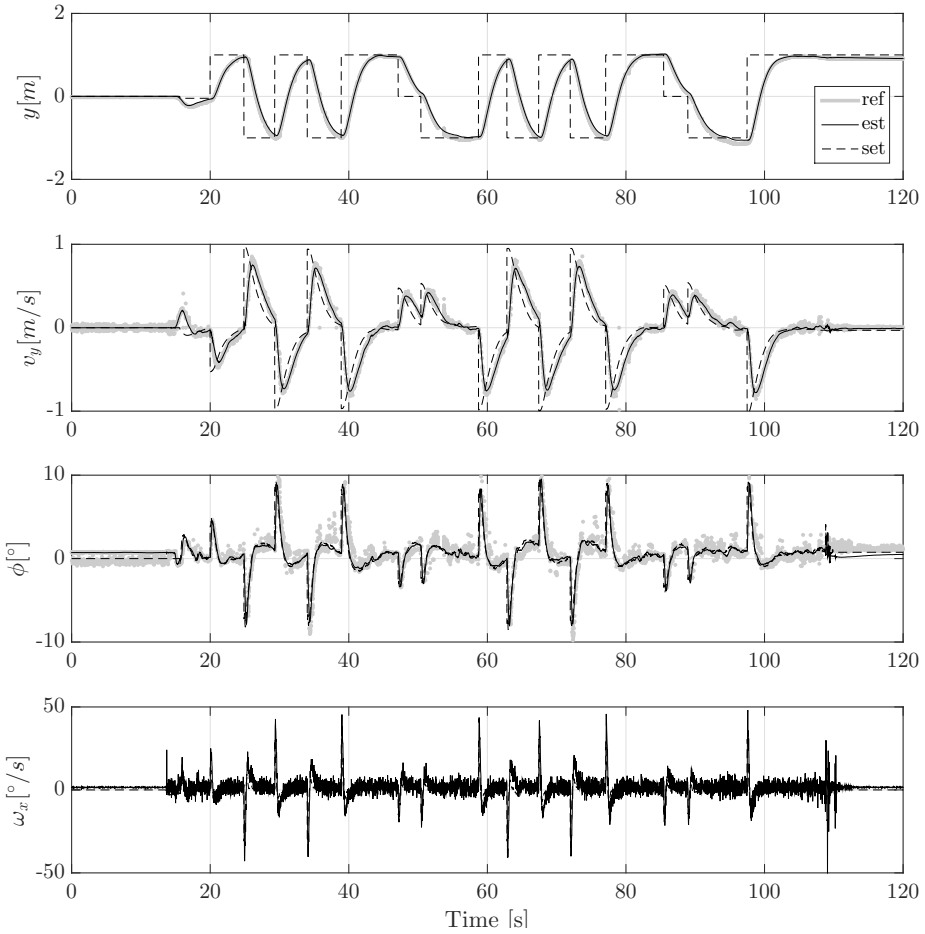


FIGURE 6.7: State estimation and control in y direction.

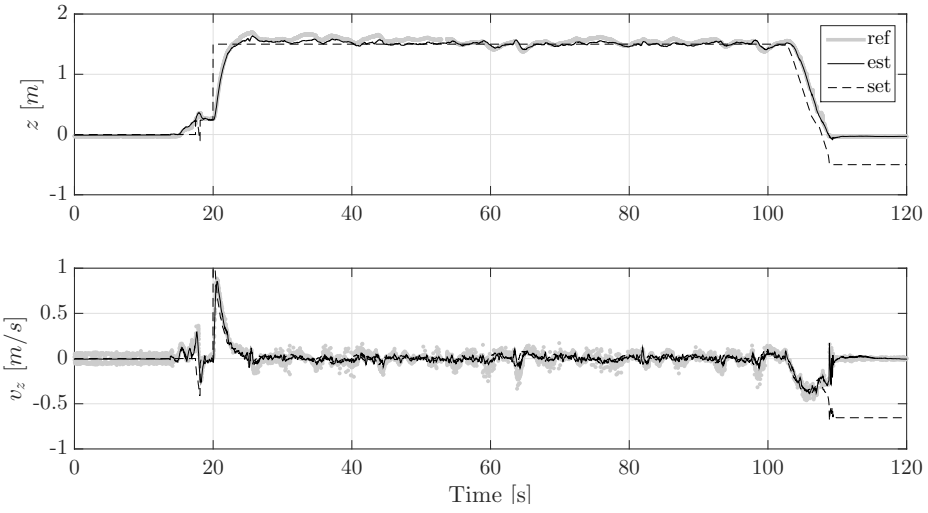


FIGURE 6.8: State estimation and control in z direction.

6.2 Arctic Environment

In summer 2017, the functionality of the first version of the flight controller was tested in Arctic environment as part of the Polarstern cruise PS 108 within the final project demonstration of ROBEX. One limitation of this earlier version of the flight controller was the exclusive use of GPS signals only. The full UAV flight controller configuration is shown in Figure 6.9.

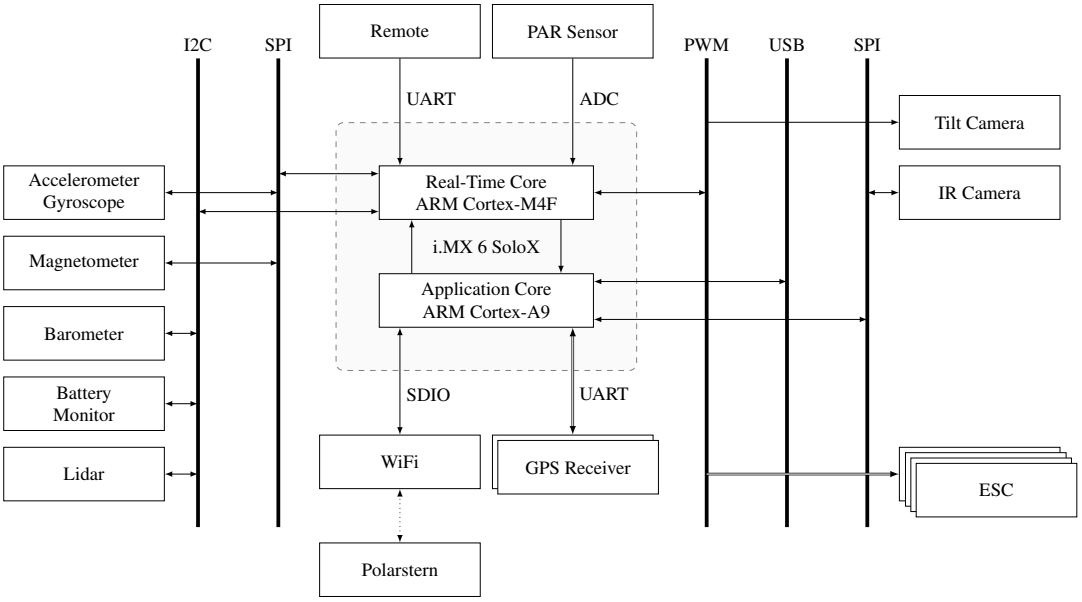


FIGURE 6.9: Flight controller setup for arctic environment.

The low satellite elevation at high latitudes, signal reflections and severe multi-path in ship vicinity degraded the available GPS raw observations significantly. For this reason, and because the GPS compass algorithm at the time applied soft baseline constraints only, the selected navigation algorithm did not compute a reliable solution near Polarstern.

In order to evaluate the different navigation systems regardless of the difficulties caused in the vicinity of the research vessel, flight experiments should have been conducted on suitable ice floes at approximately 80° North. Unfortunately, only one set of experiments could be performed on ice due to the limited availability of suitable ice floes. In this set of experiments the UAV was manually controlled and all data recorded. Within a later off-line analysis, both, the GPS compass and the traditional magnetic compass were evaluated with respect to their reliability and hence their suitability for autonomous flight in Arctic environments. The results are shown in Section 6.2.1. In order to compensate for the lack of flight time on ice, another experiment was conducted on board of Polarstern with the goal to evaluate and further improve the GPS compass algorithm. The experiment and its results are described in Section 6.2.2.

All remaining auxiliary and payload subsystems, such as the battery monitor, the barometric pressure sensor, the distance Lidar and the camera systems as well as the PAR sensor were evaluated in manual flights on ice and in the proximity of Polarstern's helipad. Since these subsystems worked very well and exactly as expected from the experiments performed before the actual mission, they will not be discussed in detail here.

6.2.1 Flight in Arctic Environment

In order to demonstrate the navigation capabilities of the developed flight controller and therefore the GPS compass in Arctic environment, a flight was conducted on an ice floe. Since the UAV electronics got accidentally damaged by swirling ice and snow from a nearby landing helicopter, only a single flight could be performed on ice. Figure 6.10 shows the UAV in air at 80° North during the flight experiment.

FIGURE 6.10: Flight experiment at 80° north.

During flight, the magnetic compass was used for heading estimation and the UAV was commanded to hold its position. All sensor raw observations were recorded for later analysis. Figure 6.11 compares the filtered magnetic heading and the raw estimate, with the GPS compass solution and integrated gyro rates. It can be seen, that the UAV drifted around yaw. The presented GPS solution applies the algorithm described in Section 5.1.1. The analysis shows, that the magnetic heading estimates at 80° North are too noisy and the horizontal component is way too weak to estimate a correct heading and therefore to fly autonomously. In contrast, the GPS heading is in close agreement with the orientation change approximated by the gyroscope.

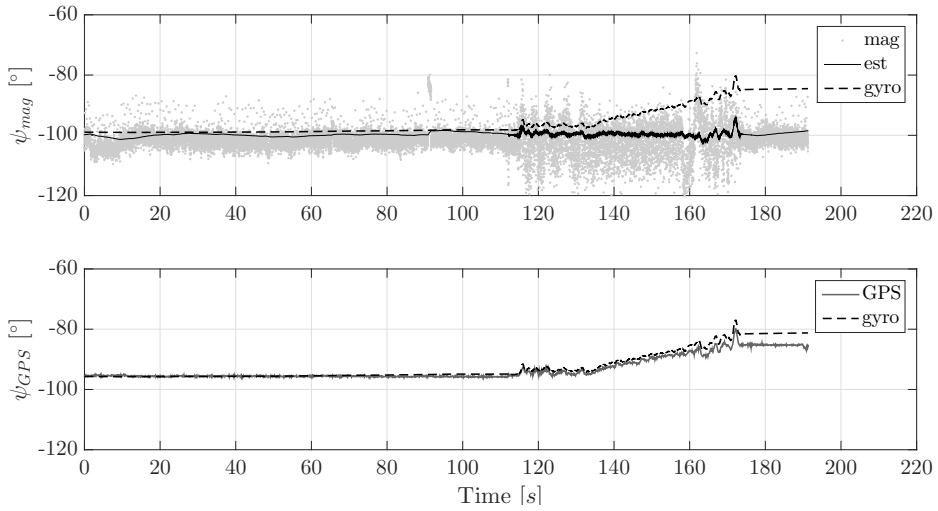


FIGURE 6.11: Comparison of the different heading estimates: Magnetic compass (top) and GPS compass (bottom).

6.2.2 RTK Heading Estimation

Within the second experiment during PS 108, the UAV was placed on board of Polarstern in such a way, that the vessels heading could be used as reference for the heading estimated by the GPS compass. Polarstern followed a predefined course in the shape of an eight in order to expose the GPS heading system to a wide range of satellite-ship constellations to generate different satellite visibility and multi-path scenarios. Figure 6.12 shows the eight track and the UAV setup on Polarstern. The satellite visibility during the experiment is shown in Figure 6.13. Although 16 different satellites were visible during the experiment, due to the low elevation and poor SNR, on average only 9 satellites could be used for the GPS heading estimation. The GPS heading is estimated according to approach described in Section 5.1.1.

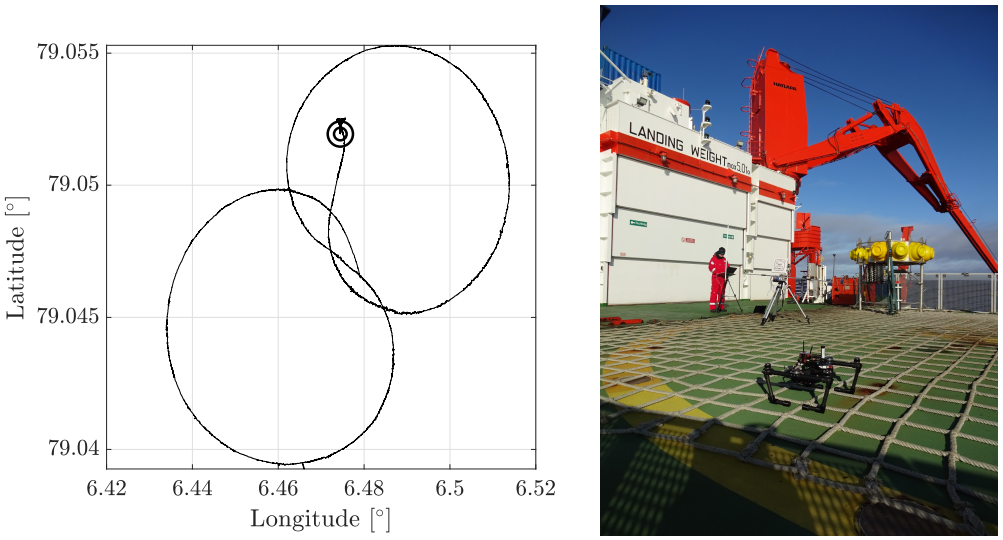


FIGURE 6.12: RTK heading: Polarstern track and experiment setup.

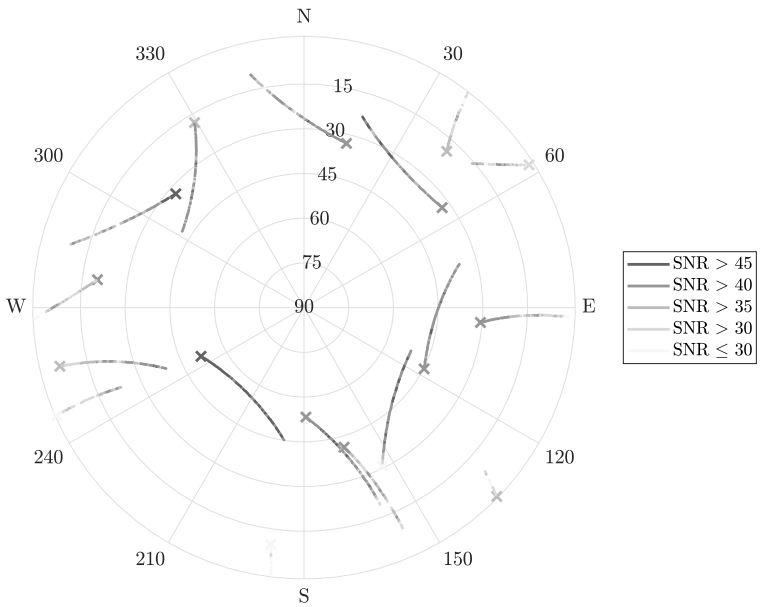


FIGURE 6.13: GPS availability during RTK heading experiment.

The upper part of Figure 6.14 compares the coupled ESKF GPS heading estimate with the Polarstern heading reference. Additionally, a heading solution based on integrated gyroscope measurements and the initial gyroscope bias estimate is calculated and plotted, too. It should be noted that the utilized flight controller version did not provide a IMU temperature stabilization and was therefore subjected to rather big and rapid bias changes. The lower part of Figure 6.14 plots the difference between the observed Polarstern heading and the coupled ESKF GPS compass solution and additionally the difference between the Polarstern heading and a standalone GPS compass.

The final difference between the pure gyroscope integration estimate and the Polarstern reference heading is more than 100°. In contrast to that, the coupled ESKF GPS compass solution using both, soft and hard baseline constraints allows to keep a valid heading estimate for more than 80 minutes with an RMSE of 0.8°. The RMSE can be compared to commercial single frequency multi GNSS solutions that allow an accuracy of 0.5°/m for a given baseline length in meters. The absolute maximum error is found to be 3.6°. The coupled ESKF GPS compass solution can be provided at up to 500 Hz in real-time on the developed flight controller platform. The GPS raw measurements are processed at a maximum rate of 10 Hz.

The heading accuracy obtained within this experiment is already good enough for an autonomous UAV flight in an Arctic environment. The experiment confirms that the proposed attitude estimation algorithm works well enough at high latitudes under poor receiving conditions using a single constellation only and therefore provides a reliable solution for low-cost real-time navigation in environments with degraded, disturbed or weak magnetic fields. Undoubtedly, making use of multiple GNSS constellations, the accuracy of the GNSS compass can be further improved.

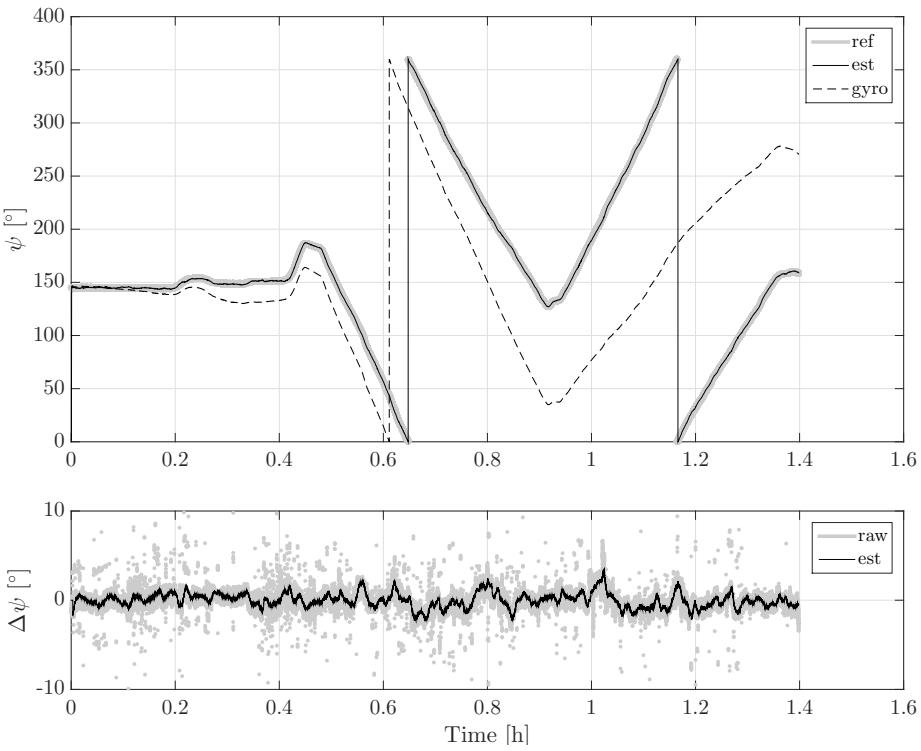


FIGURE 6.14: GPS compass heading estimate on Polarstern.

6.3 RTK Localization

The developed UAV flight controller allows precise navigation using different RTK setups. As described in Section 5.1.2, in principle, a distinction is made between the RTK setups with a fixed base and those with a moving reference station. Using a fixed reference station at a well known position, very accurate and absolute UAV velocity estimates can be computed. Conversely, in the case of a moving base setup, only relative position and velocity estimates are computed. Therefore, additional absolute velocity estimates are required in order to utilize the developed control algorithm. The relative position estimation is useful if a fixed base station can not be deployed, e.g. on a ship or if the UAV is supposed to land on another moving platform. Both RTK setups can be augmented with auxiliary location systems, e.g. UWB ranging.

Within this Section, two different RTK experiments are described. First, the moving base approach is compared to a static base setup whereby the moving base setup is augmented by additional UWB measurements in Section 6.3.1. In the second experiment described in Section 6.3.2, a UAV swarm is located using a fixed RTK reference station. Using the precise localization and navigation a cooperative swarm task is completed.

For both experiments a similar UAV core setup is used, with some differences in the respective payload configuration. Figure 6.15 shows the different UAV configurations. As primary sensors in both setups, a single frequency GNSS receiver and a MEMS IMU are used. The different payloads are indicated with dashed boxes: The Ultra-Wide Band module for the augmented radio ranging (gray) and the additional payload components within micro-drone defense system MIDRAS (white). The flight controller is connected wirelessly to a base station. The base station is equipped with the same hardware as the UAV and provides raw GNSS carrier observations in real-time.

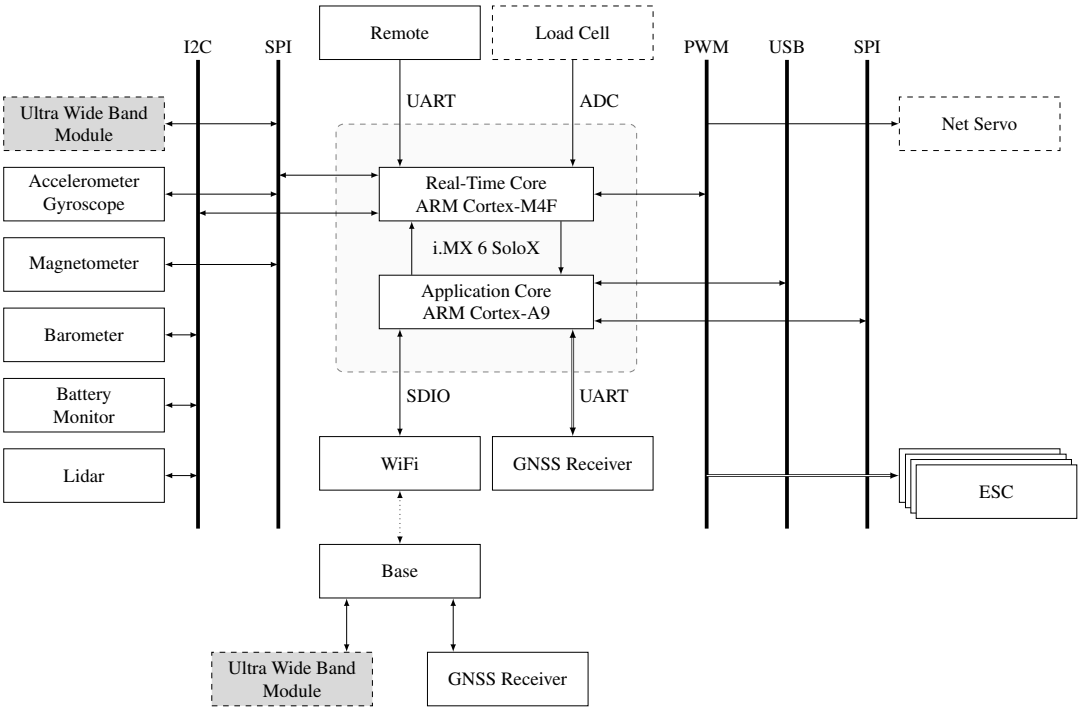


FIGURE 6.15: Flight controller setup for RTK positioning.

6.3.1 UWB augmented Moving Base

In order to demonstrate the moving base ego-motion estimation and to evaluate UWB-based GNSS augmentation, a UAV equipped with the developed flight controller and a single GNSS receiver is manually started and subsequently commanded to follow a house like pattern as shown in Figure 6.16. The reference station utilizes the exact same hardware as the UAV and its location is indicated in the Figure below, too. Both systems are equipped with a UWB transceiver for radio ranging.

The actual flight is performed using a static base setup. The raw observation are recorded and used afterwards to compute a moving base solution with auxiliary UWB range measurements off-line.

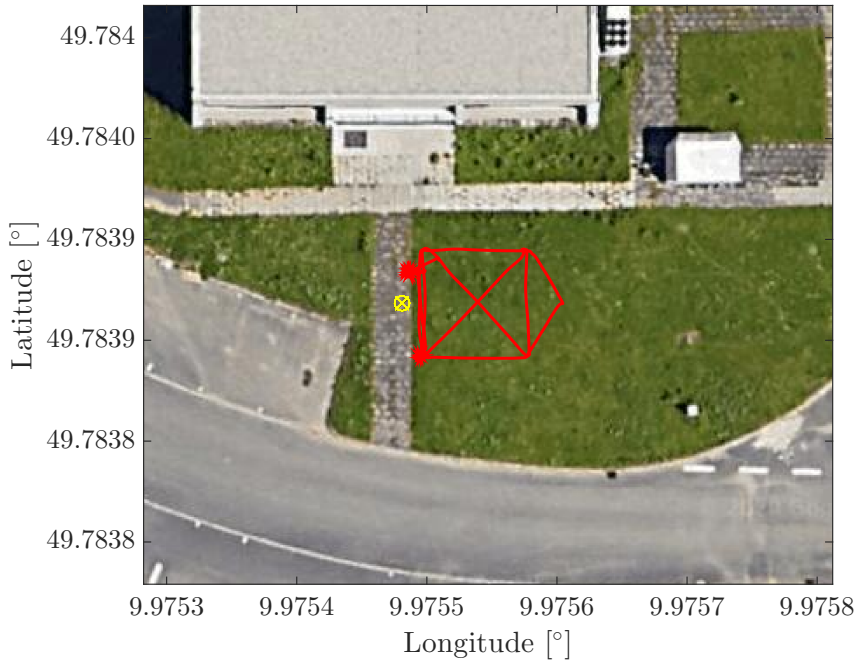


FIGURE 6.16: Flight pattern for moving base experiment.

In the off-line analysis, a ground truth estimate is compared with two different moving base solutions. One moving base solution is computed using only GNSS observations, while the other solution includes UWB measured radio ranges between the base and the UAV according to the algorithm described in Section 5.1.2.

As ground truth, a off-line calculated fixed base solution making use of all available navigation and correction data is utilized. Both moving base solutions are computed using a cold start, meaning that the navigation data for a given satellite needs to be acquired by the respective receiver first. Since this experiment should demonstrate the impact of UWB augmentation on a moving base setup, the roles of base and UAV are reversed. Therefore, the moving base solution uses the fixed reference station as a rover, while the UAV acts as the moving base.

Figure 6.17 compares the position estimates of the different approaches. The fixed base solution is indicated as FB, while the two moving base solutions are abbreviated by MB and additionally labeled in the case of the UWB augmented solution. The lower plot in Figure 6.17 indicates the solution quality of the combined UWB GNSS approach.

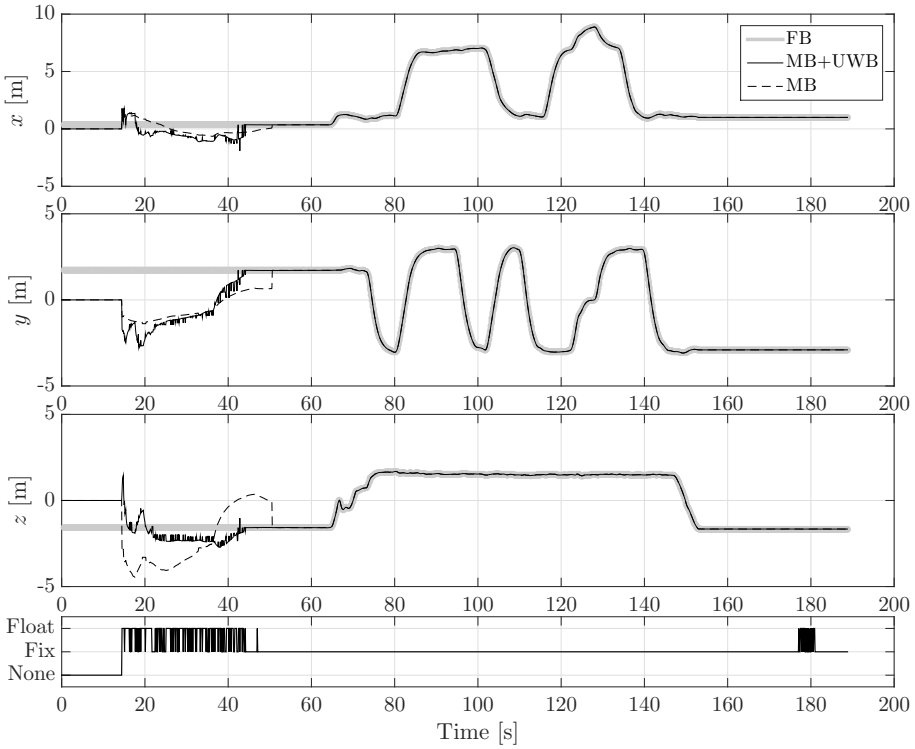


FIGURE 6.17: Position estimates of the moving base experiment.

Comparing the three position estimates, there are no significant differences regarding the position accuracy once an initial fix is obtained. However, considering the two moving base solution the initial fix is obtained 6 seconds faster using UWB augmentation. Inspecting the solution quality plot at the bottom, it can be seen that the UWB augmented system has difficulties obtaining a valid RTK fix after landing. The invalid fix is caused by the uneven UWB antenna radiation pattern which is not considered during the transceiver calibration and hence not compensated for. This becomes more evident, looking at the estimated UAV ranging error in Figure 6.18. While the ranging error is close to zero before take-off, an error of approximately 20 cm can be observed after landing. Inspecting the y position before and after the flight, the directional UWB ranging error is obvious.

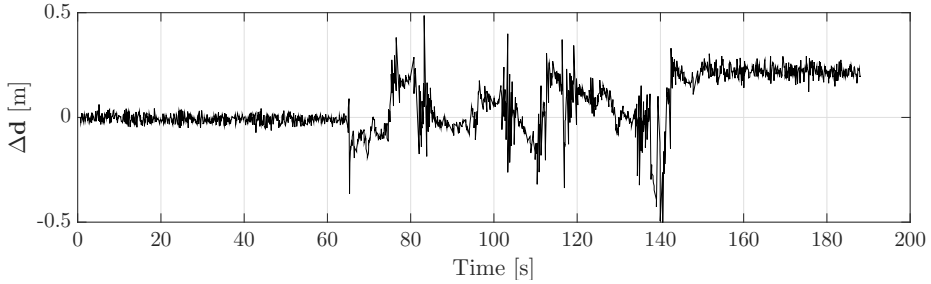


FIGURE 6.18: UWB distance error for the moving base experiment.

In summary, this experiment leads to the conclusion that a combined moving base UWB GNSS approach offers advantages during initialization or after a fix loss, if the used hardware is precisely calibrated. In practice, however, the effort required to correctly calibrate the UWB range errors outweighs the performance gain by far.

6.3.2 Formation Flight

The formation flight is performed as part of the final demonstration of the MIDRAS project. Three UAVs, each equipped with the developed flight controller, are located in real-time using the implemented RTK architecture and ego-motion estimation framework. Hereby, two of the drones perform a cooperative task carrying a net in order to autonomously catch the third UAV in mid-air as shown in Figure 6.19. The third drone is manually started and commanded to hold its current position. Once the target drone is in place, the other two drones take off, turn towards the target drone and catch it autonomously. After a successful catch, the target drone is transported to a predefined location and dropped together with the net.

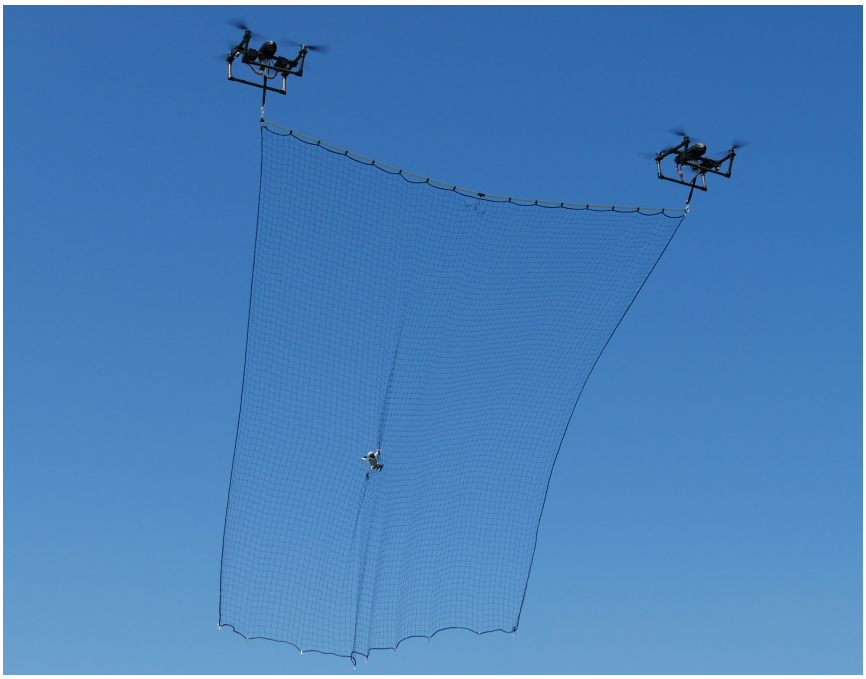


FIGURE 6.19: Two UAVs catching another drone with a net.

To guarantee a stable flight of the cooperative drones, the control architecture of the two drones needs to be slightly adapted to the mission specific requirements. However, the core of the cascaded control architecture remains the same. Therefore, two changes are applied to the standard control approach. Although the control adaptions are not part of this work, they are briefly introduced to demonstrate the flexibility of the implemented control architecture.

First, an additional distance controller is integrated using super position. The distance controller adjusts the desired velocity of the two cooperative drones depending on their currently observed distance. The desired correction is simply added as an offset to the velocity controller. Details are described in [69].

Additionally, an adaptive vertical speed controller is implemented in order to compensate for the extra weight and forces during and after the drone impact into the net. The adaptive controller can be simply integrated by replacing the standard PID speed controller at the most inner cascade of the vertical control. The adaptive controller used in this setup is explained in detail in [218].

Figure 6.20 shows the position estimate of each drone during the experiment as well as the relative distance between the two cooperative drones. Two characteristic moments are marked in the plot: The X marks the time of the drone catch, while the O marks the dropping of the target drone together with the net. Since there was no more accurate way to determine the position of the three drones simultaneously based on a different measurement technique available to our department at the time, no qualitative evaluation can take place at this point. Nevertheless, the relative distance between the two drones could be used as an indicator for the overall performance of the system. It should be noted that since the actual distance between the two systems depends heavily on the implementation of the distance controller, the significance of this indicator is limited. The lower plot indicates that the drones are able to keep the commanded distance of 4 m throughout the entire flight with a maximum deviation of roughly 35 cm.

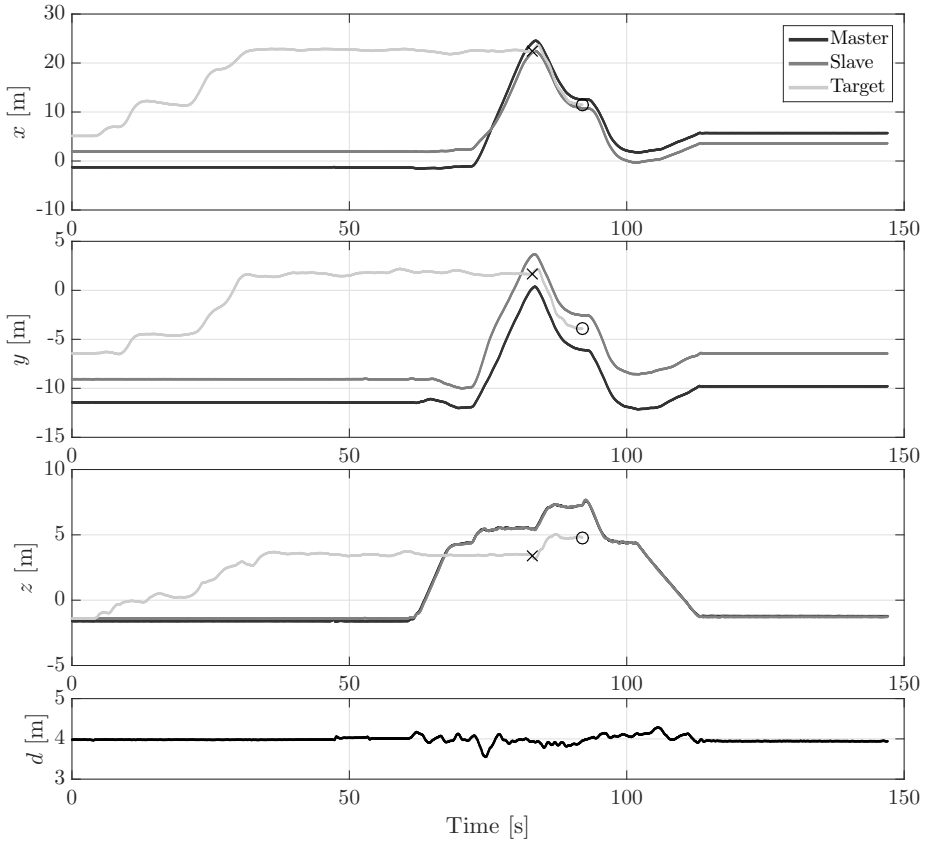


FIGURE 6.20: Position estimate of three drones.

Although a qualitative evaluation of the experiment is not possible, it demonstrates nevertheless the high accuracy of the implemented navigation system and its capability to be used in complex tasks for a swarm of cooperative drones.

Chapter 7

Conclusions

This work describes the development of FARN, a novel flight controller for multi-rotor Unmanned Aerial Vehicles (UAVs) that integrates precise Real-time Kinematic (RTK) navigation techniques. As a computational platform, a heterogeneous dual core processor is used, consisting of a real-time capable Cortex-M4 micro processing unit and a Cortex-A9 application processor. The developed software framework is based on the Real-time Onboard Dependable Operating System (RODOS) and the Robot Operating System (ROS) for the two processing cores, respectively. High-bandwidth real-time communication between the two processors is implemented using shared memory and a common message interface based on the publish/-subscribe messaging pattern. A 19-state quaternion-based Error-State Kalman Filter (ESKF) is used to estimate the UAV's ego-motion, namely the UAV's position, velocity, attitudes and angular rates. Additionally, the filter estimates inertial sensor biases as well as magnetic disturbances, too. The proposed cascaded control allows a dedicated control of the UAV's position, its velocity, attitude and angular rates. Hereby, the modular software architecture makes it possible to easily modify or completely replace different control levels. Carrier phase-based GNSS methods are deployed in order to compute very precise navigation solutions. Within the ROBEX project, the flight controller is equipped with a GPS compass and successfully tested in Arctic environment where a precise heading determination is possible despite the weak horizontal component of the Earth's magnetic field. Furthermore, an Ultra-Wide Band (UWB) augmented Real-time Kinematic positioning scheme is developed in the context of the MIDRAS project. Within this project, two cooperative UAVs, each equipped with the developed flight controller, are able to carry a commonly suspended net in order to catch a potentially dangerous drone in mid-air.

Despite the successful deployment of the flight controller in both targeted mission scenarios, different system limitations were identified. The respective limitations are considered with regard to both, past and future mission scenarios and are critically discussed in Section 7.1.

Subsequently, possible solutions with regard to future developments in the context of upcoming projects in order to overcome the identified system limitations are pointed out in Section 7.2. In addition, possible future applications are outlined.

The developed flight controller was not only used within the intended mission scenarios, but also within teaching and final theses at the Chair of Aerospace Computer Science. Therefore, this work concludes with Section 7.3 describing a small multi-rotor UAV setup used for teaching and by providing a short reference to selected final theses that utilize the software framework of the developed flight controller on different UAV platforms.

7.1 System Limitations

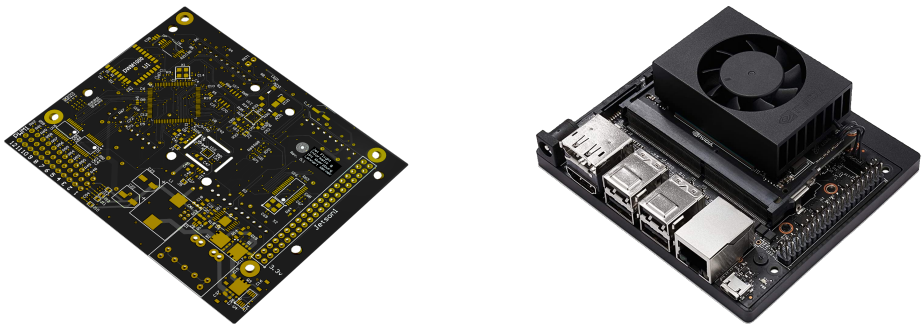
The developed flight controller and its carrier phase-based GNSS positioning methods work very reliably and provide a solid basis for future mission scenarios and research. However, two major system limitations have been identified and should be addressed in future software and hardware iterations.

First of all, the proposed UWB augmented RTK positioning approach provides only limited benefits at open sky conditions compared to standard RTK positioning. Although in theory there is no doubt that the combination of UWB and GNSS measurements is quite advantageous, the practical results shed a different light on this topic. As already discussed in Section 6.3.1, the ranging accuracy of the utilized UWB transceiver depends heavily on the performed UWB transceiver calibration. Especially the calibration of directional antenna characteristics is very extensive and requires a high effort. However, high UWB ranging accuracy and precision are mandatory to reliably resolve integer ambiguities using hard baseline constraints. The combination of high ranging noise and hard baseline constraints can cause wrong ambiguity fixes and thus degrade the overall navigation solution. In case of cycle slips or a GNSS signal lock loss however, the integration of additional ranging information might accelerate the ambiguity resolution. Within the context of the MIDRAS project and the associated experiments it has been shown that positioning based exclusively on standard RTK methods is sufficient. In summary, it can be said that the rather small benefits of UWB augmented RTK positioning in this particular application do not justify the high calibration effort. However, within environments that suffer from poor GNSS signal reception, the proposed UWB augmentation might improve the integer fixing significantly and hence be worth it.

The lack of computational power of the application core that has been identified as the second limitation. While the application processor is fast enough for the tasks that were considered during the design phase, namely the GNSS compass application and the real-time relative positioning using carrier phase-based GNSS observations, there is only limited computational power available for other tasks. Nevertheless, the GNSS compass is also an application that could benefit from more computational power. Regarding the GNSS compass at high latitudes, especially in ship vicinity signal reflections and hence multi-path errors are a major concern. Since the current GNSS compass does not estimate multi-path errors, even hard baseline constrained solutions expose a rather high RMSE. Although the current heading accuracy is good enough to fly autonomously at Arctic environments, results with a higher precision could be achieved by using multi constellation and multi frequency GNSS, tight sensor coupling and by estimating additional multi-path errors. However, these improvements come at the cost of a higher computational effort and can be therefore not achieved on the current platform. Within MIDRAS, the processing of multi constellation GNSS measurements results in long processing times that introduce an artificial measurement delay for the ESKF measurement updates. As a direct consequence, the mechanism to integrate delayed measurements into the ESKF had to be developed. With regard to the requirements in future projects that demand a reliable navigation in GNSS denied environments or the deployment of artificial intelligence in the form of neural networks for various on-board tasks, the available computation power on the current hardware platform will most likely be not sufficient.

7.2 Future Work

Because scientific progress is built on failure, future work should always aim to solve current problems and limitations. Having identified the limited computational power as the strongest limitation of the current flight controller version, this problem should be addressed first within future work. In fact, the next hardware platform is already under active development. With the availability of light weight and cost efficient embedded Artificial Intelligence (AI) computation platforms, the focus shifted towards a stand alone flight controller platform that can be easily interfaced with NVIDIA's embedded AI systems. The stand alone flight controller platform is currently designed and includes next generation inertial, magnetic and barometric pressure sensors. Furthermore, two optional dual-frequency GNSS receivers can be added to the flight controller using available board-to-board connectors. Figure 7.1 shows a preliminary PCB design and its respective NVIDIA companion.



(A) Preliminary PCB design. (B) Jetson Xavier NX [219].

FIGURE 7.1: Preliminary design flight controller and its companion.

Two NVIDIA Jetson developer kits with a small footprint are compared to the current Udoo Neo board in Table 7.1. All platforms offer additionally low-level interfaces such as I2C, SPI and UART. In case of the NVIDIA platforms, a dedicated real-time micro processor for attitude estimation and control is required. Consequently, the advantages of a heterogeneous processor such as the shared memory interface, the versatile interface configuration and the clock synchronization between the cores are lost and hence must be explicitly compensated for. However, because of NVIDIA's larger developer community, development times can be significantly reduced compared to small and exotic platforms like the Udoo Neo.

Device	**Jetson Nano** [220]	**Jetson Xavier NX** [219]	**Udoo Neo**
CPU	4× 1.4 GHz	2× 1.9 GHz 4× 1.4 GHz	1× 1 GHz
RAM	4 GB	8 GB	1 GB
GPU	128 CUDA cores	384 CUDA cores 48 Tensor cores	0
Footprint	100 mm × 80 mm	103 mm × 90.5 mm	89 mm × 59 mm
Weight	140 g	175 g	60 g
Interfaces	USB 3.0	USB 3.0, WiFi	USB 2.0, WiFi
max. Power	10 W	15 W	< 2 W
Price	109 €	429 €	65 €

TABLE 7.1: Flight controller platform comparison.

With regard to future projects, the new flight controller allows to apply advanced image processing techniques and Convolutional Neural Network (CNN) inference on-board the UAV in real-time. In combination with suitable sensors, this platform can be used to create 3D environment maps, localize itself using VSLAM algorithms, perform active path planning or implement a reactive collision avoidance system.

A possible application has been already developed and evaluated within a Bachelor's thesis [221] supervised by the author. In order to support the drone catching system developed within MIDRAS with real-time position and velocity information of the target drone, a real-time embedded visual detection and localization system for UAVs is utilized. The system uses the Intel RealSense D435i, an active RGB-D sensor. A CNN is used to detect a UAV in a RGB image in real-time on the Jetson Nano platform. Once the UAV is detected, it can be localized using the depth information if it is close enough to the sensor. Figure 7.2 shows a detected UAV at a distance of 1.5 m and the respective depth map.

(A) RGB (B) Depth

FIGURE 7.2: UAV detection and localization.

Within GNSS denied environments, for example indoor applications, a combination of a visual odometry sensor, e.g. the Intel RealSense T265, and an active depth sensor, e.g. the Intel RealSense D435i, can be used to create a 3D map for autonomous navigation and path planning. This topic is currently researched by the author and preliminary results are shown in Figure 7.3.

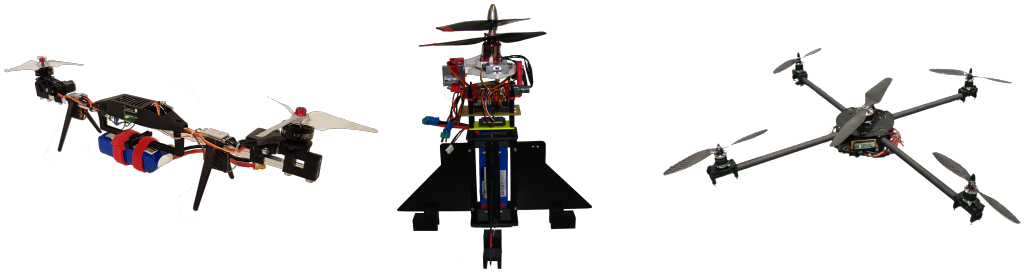
FIGURE 7.3: 3D map of the drone laboratory.

7.3 Impact on Teaching

Despite the fact that the developed flight controller is targeted towards research applications, it is also used by the author as a teaching platform within lectures and exercises and as a software base within supervised final theses. Using RODOS, the software framework can be deployed on other hardware platforms developed at the Chair of Aerospace Computer Science, e.g. the SKITH modules. The SKITH modules are small modular electronic boards that were developed for a wireless satellite technology demonstrator [222]. The main board is equipped with a UWB transceiver and a Cortex-M4 micro processor. Additionally, it provides board-to-board connectors that allow to easily extended its functionality using specialized add-on boards. The developed add-on boards include basic UAV sensors, like a GPS receiver, a IMU and a barometer, as well as PWM interfaces and a WiFi communication module. Using the stacked SKITH boards and the flight controller software framework, a small UAV platform was designed for teaching purposes. The UAV is shown in Figure 7.4. The platform is mounted in a cardanic suspension and used to teach different aspects of UAV programming to students, such as low-level sensor interfacing, basic attitude estimation schemes and control strategies.

FIGURE 7.4: UAV teaching platform.

Furthermore, the developed flight controller was used in supervised final thesis projects. Within those projects, the flight controller was deployed as a software base for different multi-rotor UAV designs and additionally as an evaluation platform for advanced sensor fusion concepts [223] and adaptive control strategies [224]. Three different UAV designs are shown in Figure 7.5.



(A) Bi-Copter [75]. (B) Coaxial-Copter [225]. (C) Penta-Copter [226].

FIGURE 7.5: Different multi-rotor UAV platforms.

Bibliography

[1] J.C. Hodgson, S.M. Baylis, R. Mott, A. Herrod, and R.H. Clarke. *"Precision Wildlife Monitoring using Unmanned Aerial Vehicles"*. In: *Scientific reports* 6.1 (2016), pp. 1–7. DOI: 10.1038/srep22574.

[2] J.A. Paredes, J. González, C. Saito, and A. Flores. *"Multispectral Imaging System with UAV Integration Capabilities for Crop Analysis"*. In: *2017 First IEEE International Symposium of Geoscience and Remote Sensing (GRSS-CHILE)*. IEEE. 2017, pp. 1–4. DOI: 10.1109/GRSS-CHILE.2017.7996009.

[3] M.P. Christiansen, M. Stigaard Laursen, R. Nyholm Jørgensen, S. Skovsen, and R. Gislum. *"Designing and Testing a UAV Mapping System for Agricultural Field Surveying"*. In: *Sensors* 17.12 (2017), p. 2703. DOI: 10.3390/s17122703.

[4] D. Mader, R. Blaskow, P. Westfeld, and C. Weller. *"Potential of UAV-based Laser Scanner and Multispecteal Camera Data in Building Inspection"*. In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 41 (2016). DOI: 10.5194/isprs-archives-XLI-B1-1135-2016.

[5] S. Sharif Mansouri, C. Kanellakis, E. Fresk, D. Kominiak, and G. Nikolakopoulos. *"Cooperative UAVs a Tool for Aerial Inspection of the Aging Infrastructure"*. In: *Field and Service Robotics*. Springer. 2018, pp. 177–189. DOI: 10.1007/978-3-319-67361-5_12.

[6] DHL Trend Research. *Unmanned Aerial Vehicles in Logistics- A DHL perspective on implications and use cases for the logistics industry*. Accessed: 2020-09-29. 2014. URL: https://www.dhl.de/content/dam/dhlde/images/ueber_uns/content/dhl_trend_report_uav.pdf.

[7] H. Surmann, R. Worst, T. Buschmann, A. Leinweber, A. Schmitz, G. Senkowski, and N. Goddemeier. *"Integration of UAVs in Urban Search and Rescue Missions"*. In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2019, pp. 203–209.

[8] H.A. Lauterbach, C.B. Koch, R. Hess, D. Eck, K. Schilling, and A. Nüchter. *"The Eins3D project—Instantaneous UAV-Based 3D Mapping for Search and Rescue Applications"*. In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2019, pp. 1–6. DOI: 10.1109/SSRR.2019.8848972.

[9] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Lynne Grixa, F. Ruess, M. Suppa, and D. Burschka. *"Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue"*. In: *IEEE robotics & automation magazine* 19.3 (2012), pp. 46–56. DOI: 10.1109/MRA.2012.2206473.

[10] J. Tiemann, F. Schweikowski, and C. Wietfeld. *"Design of an UWB Indoor-positioning System for UAV Navigation in GNSS-denied Environments"*. In: *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. 2015, pp. 1–7. DOI: 10.1109/IPIN.2015.7346960.

[11] N. Gageik, P. Benz, and S. Montenegro. *"Obstacle Detection and Collision Avoidance for a UAV with Complementary Low-cost Sensors"*. In: *IEEE Access* 3 (2015), pp. 599–609. DOI: 10.1109/ACCESS.2015.2432455.

[12] D. Kominiak, S. Sharif Mansouri, C. Kanellakis, and G. Nikolakopoulos. *"MAV Development Towards Navigation in Unknown and Dark Mining Tunnels"*. In: *arXiv preprint* (2020). arXiv: 2005.14433.

[13] F. Chataigner, P. Cavestany, M. Soler, C. Rizzo, J.-P. Gonzalez, C. Bosch, J. Gibert, A. Torrente, R. Gomez, and D. Serrano. *"ARSI: An Aerial Robot for Sewer Inspection"*. In: *Advances in Robotics Research: From Lab to Market*. Springer, 2020, pp. 249–274. DOI: 10.1007/978-3-030-22327-4_12.

[14] D. Thakur, G. Loianno, L. Jarin-Lipschitz, A. Zhou, and V. Kumar. *"Autonomous Inspection of a Containment Vessel using a Micro Aerial Vehicle"*. In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2019, pp. 1–7. DOI: 10.1109/SSRR.2019.8848936.

[15] C. Kanellakis, E. Fresk, S. Sharif Mansouri, D. Kominiak, and G. Nikolakopoulos. *"Autonomous Visual Inspection of Large-scale Infrastructures using Aerial Robots"*. In: *arXiv preprint* (2019). arXiv: 1901.05510.

[16] L. Jospin, A. Stoven-Dubois, and D.A. Cucci. *"Photometric Long-Range Positioning of LED Targets for Cooperative Navigation in UAVs"*. In: *Drones* 3.3 (2019), p. 69. DOI: 10.3390/drones3030069.

[17] B. Balaram, T. Canham, C. Duncan, H. F Grip, W. Johnson, J. Maki, A. Quon, R. Stern, and D. Zhu. *"Mars Helicopter Technology Demonstrator"*. In: *2018 AIAA Atmospheric Flight Mechanics Conference*. 2018, p. 0023. DOI: 10.2514/6.2018-0023.

[18] G. Heredia, A.E. Jimenez-Cano, I. Sanchez, D. Llorente, V. Vega, J. Braga, J.A. Acosta, and A. Ollero. *"Control of a Multirotor Outdoor Aerial Manipulator"*. In: *2014 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2014, pp. 3417–3422. DOI: 10.1109/IROS.2014.6943038.

[19] M. Fumagalli, R. Naldi, A. Macchelli, F. Forte, A.Q.L. Keemink, S. Stramigioli, R. Carloni, and L. Marconi. *"Developing an Aerial Manipulator Prototype: Physical Interaction with the Environment"*. In: *IEEE robotics & automation magazine* 21.3 (2014), pp. 41–50. DOI: 10.1109/MRA.2013.2287454.

[20] D. Wuthier, D. Kominiak, E. Fresk, and G. Nikolakopoulos. *"A geometric pulling force controller for aerial robotic workers"*. In: *IFAC-PapersOnLine* 50.1 (2017), pp. 10287–10292. DOI: 10.1016/j.ifacol.2017.08.1487.

[21] C. Korpela, M. Orsag, and P. Oh. *"Towards Valve Turning using a Dual-arm Aerial Manipulator"*. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 3411–3416. DOI: 10.1109/IROS.2014.6943037.

[22] B. Dai, Y. He, F. Gu, L. Yang, J. Han, and W. Xu. *"A Vision-based Autonomous Aerial Spray System for Precision Agriculture"*. In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2017, pp. 507–513. DOI: 10.1109/ROBIO.2017.8324467.

[23] A.S. Vempati, M. Kamel, N. Stilinovic, Q. Zhang, D. Reusser, I. Sa, J. Nieto, R. Siegwart, and P. Beardsley. *"Paintcopter: An Autonomous UAV for Spray Painting on Three-dimensional Surfaces"*. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 2862–2869. DOI: 10.1109/LRA.2018.2846278.

[24] P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza. *"Fast Trajectory Optimization for Agile Quadrotor Maneuvers with a Cable-suspended Payload"*. In: (2017). DOI: 10.5167/uzh-138923.

[25] M.R. Aagaah, E.M. Ficanha, and N. Mahmoudian. *"Drone having drone-catching feature"*. Pat. US Patent 10,005,556. 2018.

[26] F. Ruggiero, V. Lippiello, and A. Ollero. *"Aerial Manipulation: A Literature Review"*. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1957–1964. DOI: 10.1109/LRA.2018.2808541.

[27] A. Mohiuddin, T. Taha, Y.H. Zweiri, and D. Gan. *"A Survey of Single and Multi-UAV Aerial Manipulation"*. In: *Unmanned Systems* 8.2 (2020), pp. 119–147. DOI: 10.1142/S2301385020500089.

[28] B. Li, Z. Fei, and Y. Zhang. *"UAV Communications for 5G and Beyond: Recent Advances and Future Trends"*. In: *IEEE Internet of Things Journal* 6.2 (2018), pp. 2241–2263. DOI: 10.1109/JIOT.2018.2887086.

[29] K. Sreenath and V. Kumar. *"Dynamics, Control and Planning for Cooperative Manipulation of Payloads Suspended by Cables from Multiple Quadrotor Robots"*. In: *Robotics: Science and Systems* (2013). DOI: 10.15607/RSS.2013.IX.011.

[30] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza. *"Vision-based Autonomous Quadrotor Landing on a Moving Platform"*. In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE. 2017, pp. 200–207. DOI: 10.1109/SSRR.2017.8088164.

[31] T. Wulff. *"Physics and Ecology in the Marginal Ice Zone of the Fram Strait–a Robotic Approach"*.
HDL:10013/epic.48899. PhD thesis. University of Bremen, 2016.

[32] K. Gade. *"The Seven Ways to Find Heading"*. In: *The Journal of Navigation* 69.5 (2016), pp. 955–970. DOI: 10.1017/S0373463316000096.

[33] A. Consoli, J. Ayadi, G. Bianchi, S. Pluchino, F. Piazza, R. Baddour, et al. *"A Multi-Antenna Approach for UAV's Attitude Determination"*. In: *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*. 2015, pp. 401–405. DOI: 10.1109/MetroAeroSpace.2015.7180690.

[34] J. Reis, J. Sanguino, and A. Rodrigues. *"Baseline Influence on Single-frequency GPS Precise Heading Estimation"*. In: *Wireless Personal Communications* 64.1 (2012), pp. 185–196. DOI: 10.1007/s11277-012-0525-6.

[35] B.A. King and E.B. Cooper. *"Comparison of Ship's Heading Determined from an Array of GPS Antennas with Heading from Conventional Gyrocompass Measurements"*. In: *Deep Sea Research Part I: Oceanographic Research Papers* 40.11-12 (1993), pp. 2207–2216. DOI: 10.1016/0967-0637(93)90099-O.

[36] G. Lachapelle, M. E. Cannon, G. Lu, and B. Loncarevic. *"Shipborne GPS Attitude Determination During MMST-93"*. In: *IEEE Journal of Oceanic Engineering* 21.1 (1996), pp. 100–104. ISSN: 0364-9059. DOI: 10.1109/48.485206.

[37] P.J.G. Teunissen, G. Giorgi, and P.J. Buist. *"Testing of a new Single-Frequency GNSS Carrier Phase Attitude Determination Method: Land, Ship and Aircraft Experiments"*. In: *GPS Solutions* 15.1 (2011), pp. 15–28. ISSN: 1521-1886. DOI: 10.1007/s10291-010-0164-x.

[38] P. Henkel and C. Günther. *"Attitude Determination with Low-cost GPS/INS"*. In: *Proceedings of the 26-th ION GNSS+, Nashville* (2013), pp. 2015–2023. ION-ID: 11368.

[39] R.C. Hayward, D. Gebre-Egziabher, M. Schwall, J.D. Powell, and J. Wilson. *"Inertially Aided GPS Based Attitude Heading Reference System (AHRS) for General Aviation Aircraft"*. In: (1997), pp. 1415–1424. ION-ID: 2848.

[40] W. Stempfhuber and M. Buchholz. *"A Precise, Low-cost RTK GNSS System for UAV Applications"*. In: *Proc. of Unmanned Aerial Vehicle in Geomatics, ISPRS* (2011). DOI: 10.5194/isprsarchives-XXXVIII-1-C22-289-2011.

[41] N. Gageik. *"Autonome Quadrokopter zur Innenraumerkundung: AQopterI8, Forschung und Entwicklung"*. PhD thesis. 2015. URN: urn:nbn:de:bvb:20-opus-130240.

[42] Ardupilot. *Ardupilot*. Accessed: 2020-03-13. URL: http://www.ardupilot.org/.

[43] S.O.H. Madgwick, A.J.L. Harrison, and R. Vaidyanathan. *"Estimation of IMU and MARG Orientation using a Gradient Descent Algorithm"*. In: *2011 IEEE international conference on rehabilitation robotics*. IEEE. 2011, pp. 1–7. DOI: 10.1109/ICORR.2011.5975346.

[44] A.M. Sabatini. *"Quaternion-Based Extended Kalman Filter for Determining Orientation by Inertial and Magnetic Sensing"*. In: *IEEE transactions on Biomedical Engineering* 53.7 (2006), pp. 1346–1356. DOI: 10.1109/TBME.2006.875664.

[45] M. Strohmeier and S. Montenegro. *"Coupled GPS/MEMS IMU Attitude Determination of Small UAVs with COTS"*. In: *Electronics* 6.1 (2017). DOI: 10.3390/electronics6010015.

[46] F. Graas and M. Braasch. *"GPS Interferometric Attitude and Heading Determination: Initial Flight Test Results"*. In: *Navigation* 38.4 (1991), pp. 297–316. DOI: 10.1002/j.2161-4296.1991.tb01864.x.

[47] C.E. Cohen, B.W. Parkinson, and B.D. McNally. *"Flight Tests of Attitude Determination using GPS Compared Against an Inertial Navigation Unit"*. In: *Navigation* 41.1 (1994), pp. 83–97. DOI: 10.1002/j.2161-4296.1994.tb02323.x.

[48] R. Hirokawa and T. Ebinuma. *"A Low-cost Tightly Coupled GPS/INS for Small UAVs Augmented with Multiple GPS Aantennas"*. In: *Navigation* 56.1 (2009), pp. 35–44. DOI: 10.1002/j.2161-4296.2009.tb00442.x.

[49] P. Henkel and M. Iafrancesco. *"Tightly Coupled Position and Attitude Determination with Two Low-cost GNSS Receivers"*. In: *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*. IEEE. 2014, pp. 895–900. DOI: 10.1109/ISWCS.2014.6933480.

[50] G. Falco, M. Campo-Cossío Gutiérrez, E.P. Serna, F. Zacchello, and S. Bories. *"Low-cost Real-time Tightly-Coupled GNSS/INS Navigation System Based on Carrier-phase Double-differences for UAV Applications"*. In: 812 (2014), p. 841857.

[51] M.L. Sollie, T.H. Bryne, and T.A. Johansen. *"Pose Estimation of UAVs Based on INS Aided by Two Independent Low-cost GNSS Receivers"*. In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2019, pp. 1425–1435. DOI: 10.1109/ICUAS.2019.8797746.

[52] P.J.G. Teunissen. *"Least-squares Estimation of the Integer GPS Ambiguities"*. In: *Invited lecture, section IV theory and methodology, IAG general meeting, Beijing, China*. 1993.

[53] C. Eling, L. Klingbeil, and H. Kuhlmann. *"Real-time Single-frequency GPS/MEMS-IMU Attitude Determination of Lightweight UAVs"*. In: *Sensors* 15.10 (2015). DOI: 10.3390/s151026212.

[54] L. Bing, M. Qing-Hao, X. Fei, W. Jia-Ying, and S. Biao. *"Development of an on-board single-frequency GNSS RTK system for MAVs"*. In: *2015 34th Chinese Control Conference (CCC)*. IEEE. 2015, pp. 6005–6010. DOI: 10.1109/ChiCC.2015.7260579.

[55] K. Klausen, J.B. Moe, J.C. van den Hoorn, A. Gomola, T.I. Fossen, and T.A. Johansen. *"Coordinated Control Concept for Recovery of a Fixed-wing UAV on a Ship using a Net Carried by Multirotor UAVs"*. In: *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2016, pp. 964–973. DOI: 10.1109/ICUAS.2016.7502640.

[56] K. Klausen, T.I. Fossen, and T.A. Johansen. *"Autonomous Recovery of a Fixed-wing UAV using a Net Suspended by Two Multirotor UAVs"*. In: *Journal of Field Robotics* 35.5 (2018), pp. 717–731. DOI: 10.1002/rob.21772.

[57] S. Tansuriyong, M. Kyan, K. Numata, S. Taira, and T. Anezaki. *"Verification experiment for drone charging station using RTK-GPS"*. In: *2017 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*. 2017, pp. 229–232. DOI: 10.1109/ICIIBMS.2017.8279762.

[58] S. Låte. *"A Vertical Stack Approach to Cooperative Drone Lifting"*. MA thesis. NTNU, 2018.

[59] S. Zhao, Y. Chen, and J.A. Farrell. *"High-precision Vehicle Navigation in Urban Environments using an MEMS IMU and Single-frequency GPS Receiver"*. In: *IEEE transactions on intelligent transportation systems* 17.10 (2016), pp. 2854–2867. DOI: 10.1109/TITS.2016.2529000.

[60] J. Huff, A. Schultz, and M.U. de Haag. *"Assured Relative and Absolute Navigation of a Swarm of Small UAS"*. In: *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. IEEE. 2017, pp. 1–10. DOI: 10.1109/DASC.2017.8102070.

[61] J.N. Gross, Y. Gu, and B. Dewberry. *"Tightly-coupled GPS/UWB-ranging for Relative Navigation During Formation Flight"*. In: *Proc. 27th Int. Techn. Meeting ION Satellite Division*. 2014, pp. 1698–1708. ION-ID: 12556.

[62] J.N. Gross, Y. Gu, and M.B. Rhudy. *"Robust UAV Relative Navigation with DGPS, INS, and Peer-to-peer Radio Ranging"*. In: *IEEE Transactions on Automation Science and Engineering* 12.3 (2015), pp. 935–944. DOI: 10.1109/TASE.2014.2383357.

[63] D.S. Chiu and K.P. O'Keefe. *"Seamless Outdoor-to-indoor Pedestrian Navigation using GPS and UWB"*. In: *Proceedings of the 21st International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2008), The Institute of Navigation*. Vol. 1. 2008, pp. 322–333. ION-ID: 8165.

[64] G. MacGougan, K. O'Keefe, and D. Chiu. *"Multiple UWB Range Assisted GPS RTK in Hostile Environments"*. In: *ION GNSS*. 2008, pp. 3020–3035. ION-ID: 8208.

[65] J. Wang, Y. Gao, Z. Li, X. Meng, and C.M. Hancock. *"A Tightly-Coupled GPS/INS/UWB Cooperative Positioning Sensors System Supported by V2I Communication"*. In: *Sensors* 16.7 (2016), p. 944. DOI: 10.3390/s16070944.

[66] F. Wenzhöfer, T. Wulff, S. Floegel, S. Sommer, and C. Waldmann. *"ROBEX-Innovative Robotic Technologies for Ocean Observations, a Deep-sea Demonstration Mission"*. In: *OCEANS 2016 MTS/IEEE Monterey*. IEEE. 2016, pp. 1–8. DOI: 10.1109/OCEANS.2016.7761215.

[67] T. Wulff, S. Lehmenhecker, J. Hagemann, M. Busack, S. Tippenhauer, M. Strohmeier, and J. Rothe. *Revealing Physical and Ecological Dynamics at an Ice Edge - a Robotic Approach*.
HDL:10013/epic.fad984e3-bb78-4d36-bc91-501c9c6d64d3. Polar 2018, Davos, Switzerland, 2018.

[68] S. Tippenhauer, T. Wulff, S. Lehmenhecker, M. Strohmeier, T. Mikschl, and S. Montenegro. *Observing the Arctic with Autonomous Robots*.
HDL:10013/epic.50909. EGU General Assembly, 2017.

[69] J. Rothe, M. Strohmeier, and S. Montenegro. *"A Concept for Catching Drones with a Net Carried by Cooperative UAVs"*. In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2019, pp. 126–132. DOI: 10.1109/SSRR.2019.8848973.

[70] T. Pavlenko, M. Schütz, M. Vossiek, T. Walter, and S. Montenegro. *"Wireless Local Positioning System for Controlled UAV Landing in GNSS-Denied Environment"*. In: *2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. IEEE. 2019, pp. 171–175. DOI: 10.1109/MetroAeroSpace.2019.8869587.

[71] Markforged. *Markforged - 3D Printers*. Accessed: 2020-05-29. URL: https://markforged.com/.

[72] Hacker. *Hacker - Brushless Motors*. Accessed: 2020-05-29. URL: https://www.hacker-motor-shop.com/Akkus-und-Akkuzubehoer/Akkusuche/TF-ECO-X-5000-6S-MTAG.htm?SessionId=&a=article&ProdNr=95000631&p=11452.

[73] T-Motor. *T-Motor - The Safer Propulsion System*. Accessed: 2020-05-29. URL: http://uav-en.tmotor.com/.

[74] S. Kirby. *Open Source Firmware for ATmega-based Brushless ESCs*. Accessed: 2020-05-29. URL: https://github.com/sim-/tgy.

[75] L. Werner. *"Design and Modeling of a Bi-Copter"*. B.Sc. Thesis. University of Wuerzburg, 2018.

[76] SECO SPA. *Udoo Neo*. Accessed: 2020-06-04. URL: https://www.udoo.org/udoo-neo/.

[77] NXP. *MCIMX6X4EVM10AB*. Accessed: 2020-06-04. URL: https://www.nxp.com/part/MCIMX6X4EVM10AB#/.

[78] NXP Community. *i.MX6 SoloX Debugging*. Accessed: 2020-06-04. URL: https://community.nxp.com/thread/350148.

[79] ST Microelectronics. *LSM9DS1 - iNEMO inertial module*. Accessed: 2020-04-22. URL: https://www.st.com/en/mems-and-sensors/lsm9ds1.html.

[80] Analog Devices. *Tactical Grade Inertial Measurement Unit with Industry's Lowest SWaP+C*. Accessed: 2020-04-21. 2017. URL: https://www.analog.com/media/en/news-marketing-collateral/product-highlight/Tactical-Grade-IMU.pdf.

[81] A. Vydhyanathan, G. Bellusci, H. Luinge, and P. Slycke. *The Next Generation Xsens Motion Trackers for Industrial Applications*. Tech. rep. Xsens: Enschede, The Netherlands, 2015.

[82] M. Li, X. Yu H.and Zheng, and A.I. Mourikis. *"High-fidelity Sensor Modeling and Self-calibration in Vision-aided Inertial Navigation"*. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 409–416. DOI: 10.1109/ICRA.2014.6906889.

[83] E. Fresk, G. Nikolakopoulos, and T. Gustafsson. *"A Generalized Reduced-Complexity Inertial Navigation System for Unmanned Aerial Vehicles"*. In: *IEEE Transactions on Control Systems Technology* 25.1 (2017), pp. 192–207. ISSN: 2374-0159. DOI: 10.1109/TCST.2016.2542022.

[84] I. Frosio, F. Pedersini, and N.A. Borghese. *"Autocalibration of MEMS Accelerometers"*. In: *IEEE Transactions on Instrumentation and Measurement* 58.6 (2008), pp. 2034–2041. DOI: 10.1109/TIM.2008.2006137.

[85] M. Pedley. *"High Precision Calibration of a Three-axis Accelerometer"*. In: *Freescale Semiconductor Application Note* 1 (2013). URL: https://www.nxp.com/docs/en/application-note/AN4399.pdf.

[86] J.F. Zhang, J.P. Bai, J.B. Wu, J. Zeng, and X.S. Lai. *"Fast Field Calibration of MEMS-based IMU for Quadrotor's Applications"*. In: *Sensors & Transducers* 151.4 (2013), p. 1.

[87] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2006. DOI: 10.1007/978-0-387-40065-5.

[88] E. Fresk and G. Nikolakopoulos. *"Frame Induced Vibration Estimation and Attenuation Scheme on a Multirotor Helicopter"*. In: *53rd IEEE conference on decision and control*. IEEE. 2014, pp. 5698–5703. DOI: 10.1109/CDC.2014.7040281.

[89] Z. Li, M. Lao, S.K. Phang, M.R.A. Hamid, K.Z. Tang, and F. Lin. *"Development and Design Methodology of an Anti-Vibration System on Micro-UAVs"*. In: *International Micro Air Vehicle Conference and Flight Competition (IMAV)*. 2017, pp. 223–228.

[90] X. Niu, Y. Li, H. Zhang, Q. Wang, and Y. Ban. *"Fast Thermal Calibration of Low-grade Inertial Sensors and Inertial Measurement Units"*. In: *Sensors* 13.9 (2013), pp. 12192–12217. DOI: 10.3390/s130912192.

[91] P. Aggarwal, Z. Syed, X. Niu, and N. El-Sheimy. *"Cost-effective Testing and Calibration of Low Cost MEMS Sensors for Integrated Positioning, Navigation and Mapping Systems"*. In: *Proceedings of XIII FIG Conference*. 2006, pp. 8–13.

[92] G. Liu, F. Yang, X. Bao, and T. Jiang. *"Robust Optimization of a MEMS Accelerometer Considering Temperature Variations"*. In: *Sensors* 15.3 (2015), pp. 6342–6359. DOI: 10.3390/s150306342.

[93] D. Xia, S. Chen, S. Wang, and H. Li. *"Microgyroscope Temperature Effects and Compensation-control Methods"*. In: *Sensors* 9.10 (2009), pp. 8349–8376. DOI: 10.3390/s91008349.

[94] J.-K. Shiau, C.-X. Huang, M.-Y. Chang, et al. *"Noise Characteristics of MEMS Gyro's Null Dift and Temperature Compensation"*. In: *Journal of Applied Science and engineering* 15.3 (2012), pp. 239–246. DOI: 10.6180/jase.2012.15.3.04.

[95] G. Araghi et al. *"Temperature Compensation Model of MEMS Inertial Sensors Based on Neural network"*. In: *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE. 2018, pp. 301–309. DOI: 10.1109/PLANS.2018.8373395.

[96] Vötsch Industrietechnik. *Temperature Test Chamber VT 4002*. Accessed: 2020-04-22. URL: http://lampx.tugraz.at/~hadley/num/ch9/instruments/VT4002/VT4002_climate_chamber.pdf.

[97] O.J. Woodman. *An Introduction to Inertial Navigation*. Tech. rep. University of Cambridge, Computer Laboratory, 2007. DOI: 10.1119/1.3081061.

[98] *"IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros"*. In: *IEEE Std 952-1997* (1998), pp. 1–84.

[99] E. Ogier. *AVAR*. MATLAB Central File Exchange. 2020. URL: https://www.mathworks.com/matlabcentral/fileexchange/55765-avar.

[100] PNICorp. *RM3100 Geomagnetic Sensor*. Accessed: 2020-06-10. URL: https://www.pnicorp.com/rm3100/.

[101] T. Ozyagcilar. *"Calibrating an Ecompass in the Presence of Hard and Soft-iron Interference"*. In: *Freescale Semiconductor Application Note* (2012), pp. 1–17. URL: https://www.nxp.com/docs/en/application-note/AN4246.pdf.

[102] P. Yury. *Ellipsoid Fit*. https://www.mathworks.com/matlabcentral/fileexchange/24693-ellipsoid-fit. Accessed: 2020-03-13.

[103] A. Chulliat, S. Macmillan, P. Alken, C. Beggan, M. Nair, B. Hamilton, A. Woods, V. Ridley, S. Maus, and . Thomson. *"The US/UK World Magnetic Model for 2015-2020"*. In: (2015). URL: https://geomag.bgs.ac.uk/documents/WMM2015_Report.pdf.

[104] Bosch Sensortec. *BMP388 - Digital Pressure Sensor*. Accessed: 2020-04-22. URL: https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/pressure-sensors-bmp388.html.

[105] uBlox. *ZED-F9P high precision GNSS module*. Accessed: 2020-04-22. URL: https://www.u-blox.com/en/product/zed-f9p-module.

[106] uBlox. *NEO/LEA-M8T concurrent GNSS timing modules*. Accessed: 2020-04-22. URL: https://www.u-blox.com/en/product/neolea-m8t-series.

[107] uBlox. *MAX-M8Q Small GNSS module*. Accessed: 2020-04-22. URL: https://www.u-blox.com/en/product/max-m8-series?lang=de.

[108] uBlox. *ANN-MB series - Multi-band, high precision GNSS antennas*. Accessed: 2020-04-22. URL: https://www.u-blox.com/en/product/ann-mb-series.

[109] Tallysman. *TW2920 Single Band Antenna with L-Band*. Accessed: 2020-04-22. URL: https://www.tallysman.com/product/tw2920-single-band-antenna-with-l-band/.

[110] M.W. Mueller, M. Hamer, and R. D'Andrea. *"Fusing Ultra-wideband Range Measurements with Accelerometers and Rate Gyroscopes for Quadrocopter State Estimation"*. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 1730–1736. DOI: 10.1109/ICRA.2015.7139421.

[111] A. Ledergerber, M. Hamer, and R. D'Andrea. *"A Robot Self-localization System using One-way Ultra-wideband Communication"*. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 3131–3137. DOI: 10.1109/IROS.2015.7353810.

[112] K. Hausman, S. Weiss, R. Brockers, L. Matthies, and G.S. Sukhatme. *"Self-calibrating multi-sensor fusion with probabilistic measurement validation for seamless sensor switching on a UAV"*. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 4289–4296. DOI: 10.1109/ICRA.2016.7487626.

[113] J. Tiemann and C. Wietfeld. *"Scalable and Precise Multi-UAV Indoor Navigation using TDOA-based UWB Localization"*. In: *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. 2017, pp. 1–7. DOI: 10.1109/IPIN.2017.8115937.

[114] W. Shule, C. Martínez Almansa, J. Pena Queralta, Z. Zou, and T. Westerlund. *"UWB-Based Localization for Multi-UAV Systems and Collaborative Heterogeneous Multi-Robot Systems: a Survey"*. In: *arXiv preprint* (2020). arXiv: 2004.08174.

[115] J.P. Queralta, C. Martínez Almansa, F. Schiano, D. Floreano, and T. Westerlund. *"UWB-based system for UAV Localization in GNSS-Denied Environments: Characterization and Dataset"*. In: *arXiv preprint* (2020). arXiv: 2003.04380.

[116] DecaWave. *DWM1000 Module*. Accessed: 2020-04-22. URL: https://www.decawave.com/product/dwm1000-module/.

[117] Radiocommunication International Telecommunication Union. *Characteristics of ultra-wideband technology*. online. 2006. URL: http://www.itu.int/dms_pubrec/itu-r/rec/sm/R-REC-SM.1755-0-200605-I!!PDF-E.pdf.

[118] J. Tiemann, F. Eckermann, and C. Wietfeld. *"Multi-user Interference and Wireless Clock Synchronization in TDOA-based UWB Localization"*. In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. 2016, pp. 1–6. DOI: 10.1109/IPIN.2016.7743696.

[119] Decawave. *DW1000 User Manual-How to Use, Configure and Program the DWM 1000 UWB Transceiver*. Tech. rep. Decawave Limited, 2017. URL: https://www.decawave.com/sites/default/files/resources/dw1000_user_manual_2.11.pdf.

[120] Decawave. *Antenna Delay Calibration of DW1000-based Products and Systems*. Tech. rep. Decawave Limited, 2018. URL: http://www.decawave.com.

[121] Texas Instruments. *WiLink 8 industrial Wi-Fi, Bluetooth & Bluetooth Smart (Low energy) module*. 2016. URL: https://www.ti.com/product/WL1831MOD.

[122] Horizon Hobby. *"Specification for Spektrum Remote Receiver Interfacing"*. In: (2016). URL: https://www.spektrumrc.com/ProdInfo/Files/Remote%20Receiver%20Interfacing%20Rev%20A.pdf.

[123] M. Faessler. *SBUS Protocol*. Github. 2018. URL: https://github.com/uzh-rpg/rpg_quadrotor_control/wiki/SBUS-Protocol.

[124] Spektrum. *DX8 G2 System with AR8010T Receiver Mode 2*. Accessed: 2020-04-21. URL: https://www.spektrumrc.com/Products/Default.aspx?ProdId=SPM8015.

[125] DJI. *DJI FPV-System*. URL: https://www.dji.com/de/fpv/info.

[126] Intel RealSene Technology. *Intel RealSense Tracking Camera*. Accessed: 2020-04-22. URL: https://www.intelrealsense.com/tracking-camera-t265/.

[127] Avago Technologies. *ADNS-3080 High-Performance Optical Mouse Sensor*. Accessed: 2020-04-22. URL: http://www.farnell.com/datasheets/1931324.pdf.

[128] N. Gageik, M. Strohmeier, and S. Montenegro. *"An Autonomous UAV with an Optical Flow Sensor for Positioning and Navigation"*. In: *International Journal of Advanced Robotic Systems* 10.10 (2013), p. 341. DOI: 10.5772/56813.

[129] OptiTrack. *Flex 3*. Accessed: 2020-04-22. URL: https://optitrack.com/products/flex-3/.

[130] M. Strohmeier, T. Walter, J. Rothe, and S. Montenegro. *"Ultra-wideband based pose estimation for small unmanned aerial vehicles"*. In: *IEEE Access* 6 (2018). DOI: 10.1109/ACCESS.2018.2873571.

[131] Garmin. *Lidar Lite v3 Operation Manual and Technical Specifications*. Accessed: 2020-04-22. URL: https://buy.garmin.com/de-DE/DE/p/557294.

[132] Robot Electronics. *SRF02 Ultrasonic range finder - Technical Specification*. Accessed: 2020-04-22. URL: https://www.robot-electronics.co.uk/htm/srf02tech.htm.

[133] D-Link. *DWM-222 4G LTE USB Adapter*. Accessed: 2020-04-22. URL: https://eu.dlink.com/de/de/products/dwm-222-4g-lte-usb-adapter.

[134] AMBER wireless GmbH. *Manual AMB8636*. Accessed: 2020-04-22. URL: http://www.farnell.com/datasheets/1898782.pdf.

[135] FLIR. *FLIR LEPTON Engineering Datasheet*. Accessed: 2020-04-22. URL: https://www.flir.de/products/lepton/.

[136] J.B. Kuipers et al. *Quaternions and Rotation Sequences*. Vol. 66. Princeton university press Princeton, 1999. DOI: 10.2307/3621452.

[137] M. D. Shuster. *"A Survey of Attitude Representations"*. In: *Navigation* 8.9 (1993), pp. 439–517. DOI: 10.2514/6.2012-4422.

[138] J. Diebel. *"Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors"*. In: *Matrix* 58.15-16 (2006), pp. 1–35.

[139] J. Sola. *"Quaternion Kinematics for the Error-state Kalman Filter"*. In: *arXiv preprint* (2017). arXiv: 1711.02508.

[140] N. Trawny and S.I. Roumeliotis. *Indirect Kalman Filter for 3D Attitude Estimation*. Tech. rep. University of Minnesota, Dept. of Comp. Sci. & Eng., 2005.

[141] H. Qi and J.B. Moore. *"Direct Kalman Filtering Approach for GPS/INS Integration"*. In: *IEEE Transactions on Aerospace and Electronic Systems* 38.2 (2002), pp. 687–693. ISSN: 2371-9877. DOI: 10.1109/TAES.2002.1008998.

[142] V. Madyastha, V. Ravindra, S. Mallikarjunan, and A. Goyal. *"Extended Kalman Filter vs. Error State Kalman Filter for Aircraft Attitude Estimation"*. In: *AIAA Guidance, Navigation, and Control Conference*. 2011, p. 6615. DOI: 10.2514/6.2011-6615.

[143] S.I. Roumeliotis, G.S. Sukhatme, and G.A. Bekey. *"Circumventing Dynamic Modeling: Evaluation of the Error-state Kalman Filter Applied to Mobile Robot Localization"*. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*. Vol. 2. IEEE. 1999, pp. 1656–1663. DOI: 10.1109/ROBOT.1999.772597.

[144] A. R. Jiménez, F. Seco, J. C. Prieto, and J. Guevara. *"Indoor Pedestrian Navigation using an INS/EKF Framework for Yaw Drift reduction and a Foot-Mounted IMU"*. In: *2010 7th Workshop on Positioning, Navigation and Communication*. 2010, pp. 135–143. DOI: 10.1109/WPNC.2010.5649300.

[145] K. Schmid, F. Ruess, M. Suppa, and D. Burschka. *"State Estimation for Highly Dynamic Flying Systems using Key Frame Odometry with Varying Time Delays"*. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 2997–3004. DOI: 10.1109/IROS.2012.6385969.

[146] A. Widy and K. T. Woo. *"Robust Attitude Estimation Method for Underwater Vehicles with External and Internal Magnetic Noise Rejection using Adaptive Indirect Kalman Filter"*. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 2595–2600. DOI: 10.1109/IROS.2017.8206082.

[147] Y. S. Suh. *"Orientation Estimation Using a Quaternion-Based Indirect Kalman Filter With Adaptive Estimation of External Acceleration"*. In: *IEEE Transactions on Instrumentation and Measurement* 59.12 (2010), pp. 3296–3305. ISSN: 1557-9662. DOI: 10.1109/TIM.2010.2047157.

[148] A.I. Mourikis and S.I. Roumeliotis. *"A Multi-State Constraint Kalman Filter for Vision-Aided Inertial Navigation"*. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024.

[149] S. Lynen, M.W. Achtelik, S. Weiss, M.a Chli, and . Siegwart. *"A Robust and Modular Multi-sensor Fusion Approach Applied to MAV Navigation"*. In: *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2013, pp. 3923–3929. DOI: 10.1109/IROS.2013.6696917.

[150] I. Skog, J.-O. Nilsson, and P. Händel. *"Evaluation of Zero-velocity Detectors for Foot-mounted Inertial Navigation Systems"*. In: *2010 International Conference on Indoor Positioning and Indoor Navigation*. IEEE. 2010, pp. 1–6. DOI: 10.1109/IPIN.2010.5646936.

[151] R. Munguía and A. Grau. *"A Practical Method for Implementing an Attitude and Heading Reference System"*. In: *International Journal of Advanced Robotic Systems* 11.4 (2014), p. 62. DOI: 10.5772/58463.

[152] H.L. Alexander. *"State Estimation for Distributed Systems with Sensing Delay"*. In: *Data Structures and Target Classification*. Vol. 1470. International Society for Optics and Photonics. 1991, pp. 103–111. DOI: 10.1117/12.44843.

[153] T. Dall Larsen, N.A. Andersen, O. Ravn, and N. Kjølstad Poulsen. *"Incorporation of Time Delayed Measurements in a Discrete-time Kalman Filter"*. In: *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)*. Vol. 4. IEEE. 1998, pp. 3972–3977. DOI: 10.1109/CDC.1998.761918.

[154] D. Kotarski and J. Kasać. *"Generalized Control Allocation Scheme for Multirotor Type of UAVs"*. In: *Drones-Applications*. IntechOpen, 2018. DOI: 10.5772/intechopen.73006.

[155] M.A. Johnson and M.H. Moradi. *PID control*. Springer, 2005. ISBN: 978-1-84628-148-8.

[156] A. Visioli. *Practical PID control*. Springer Science & Business Media, 2006. ISBN: 978-1-84628-586-8.

[157] T. Takasu and A. Yasuda. *"Development of the Low-cost RTK-GPS Receiver with an Open Source Program Package RTKLIB"*. In: *International Symposium on GPS/GNSS*. International Convention Center Jeju Korea. 2009, pp. 4–6.

[158] G. Xu and Y. Xu. *GPS: Theory, Algorithms and Applications*. Springer, 2016. DOI: 10.1007/978-3-540-72715-6.

[159] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle. *GNSS–Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and More*. Springer, 2007. DOI: 10.1007/978-3-211-73017-1.

[160] P.D. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech house, 2013. ISBN: 978-1-60807-005-3.

[161] P.J.G. Teunissen and A. Kleusberg. *GPS for Geodesy*. Springer, 2012. DOI: 10.1007/978-3-642-72011-6.

[162] G. Giorgi and P.J.G. Teunissen. *"GNSS Carrier Phase-based Attitude Determination"*. In: *Recent Advances in Aircraft Technology*. InTech, 2012, pp. 193–220. DOI: 10.5772/38381.

[163] E.J. Post. *"Sagnac Effect"*. In: *Reviews of Modern Physics* 39.2 (1967), p. 475. DOI: 10.1103/RevModPhys.39.475.

[164] G. Strang and K. Borre. *Linear Algebra, Geodesy and GPS*. Siam, 1997. ISBN: 978-0961408862.

[165] M.M. Hoque and N. Jakowski. *"Ionospheric Propagation Effects on GNSS Signals and New Correction Approaches"*. In: *Global Navigation Satellite Systems: Signal, Theory and Applications* (2012), pp. 381–405. DOI: 10.5772/30090.

[166] J. Saastamoinen. *"Atmospheric Correction for the Troposphere and Stratosphere in Radio Ranging Satellites"*. In: *The Use of Artificial Satellites for Geodesy* 15 (1972), pp. 247–251. DOI: 10.1029/GM015p0247.

[167] M. Bevis, S. Businger, S. Chiswell, T.A. Herring, R.A. Anthes, C. Rocken, and R. H. Ware. *"GPS Meteorology: Mapping Zenith Wet Delays onto Precipitable Water"*. In: *Journal of applied meteorology* 33.3 (1994), pp. 379–386. DOI: 10.1175/1520-0450(1994)033<0379:GMMZWD>2.0.CO;2.

[168] J. Paul Collins. *"Assessment and Development of a Tropospheric Delay Model for Aircraft Users of the Global Positioning System"*. MA thesis. 1999.

[169] J.A. Klobuchar. *"Ionospheric time-delay algorithm for single-frequency GPS users"*. In: *IEEE Transactions on Aerospace and Electronic Systems* 3 (1987), pp. 325–331. DOI: 10.1109/TAES.1987.310829.

[170] J. Van Sickle. *GPS for Land Surveyors*. CRC Press, 2015. ISBN: 978-1-46658-310-8.

[171] G. Di Giovanni and S. M. Radicella. *"An analytical model of the electron density profile in the ionosphere"*. In: *Advances in Space Research* 10.11 (1990), pp. 27–30. DOI: 10.1016/0273-1177(90)90301-F.

[172] A. Angrisano, S. Gaglione, C. Gioia, M. Massaro, and S. Troisi. *"Benefit of the NeQuick Galileo Version in GNSS Single-Point Positioning"*. In: *International Journal of Navigation and Observation* (2013). DOI: 10.1155/2013/302947.

[173] S. Kedar, G.A. Hajj, B.D. Wilson, and M.B. Heflin. *"The Effect of the Second Order GPS Ionospheric Correction on Receiver Positions"*. In: *Geophysical Research Letters* 30.16 (2003). DOI: 10.1029/2003GL017639.

[174] M. Fritsche, R. Dietrich, C. Knöfel, A. Rülke, S. Vey, M. Rothacher, and P. Steigenberger. *"Impact of Higher-order Ionospheric Terms on GPS Estimates"*. In: *Geophysical research letters* 32.23 (2005). DOI: 10.1029/2005GL024342.

[175] C.S. Carrano, K.M. Groves, W.J. McNeil, and P.H. Doherty. *"Direct Measurement of the Residual in the Ionosphere-free Linear Combination During Scintillation"*. In: *Proceedings of the 2013 Institute of Navigation ION NTM meeting, San Diego, CA*. 2013. ION-ID: 10893.

[176] N. Reussner and L. Wanninger. *"GLONASS Inter-frequency Biases and Their Effects on RTK and PPP Carrier-phase Ambiguity Resolution"*. In: *Proceedings of ION GNSS*. 2011, pp. 712–716. ION-ID: 9632.

[177] uBlox. *"Achieving Centimeter Level Performance with Low Cost Antennas"*. In: *uBlox White Paper* (2016). URL: https://www.u-blox.com/en/downloads#tab-white-papers.

[178] P. Henkel and A. Sperl. *"Real-time Kinematic Positioning for Unmanned Air Vehicles"*. In: *2016 IEEE Aerospace Conference*. IEEE. 2016, pp. 1–7. DOI: 10.1109/AERO.2016.7500933.

[179] O. Montenbruck, P. Steigenberger, and A. Hauschild. *"Multi-GNSS Signal-in-space Range Error Assessment–Methodology and Results"*. In: *Advances in Space Research* 61.12 (2018), pp. 3020–3038. DOI: 10.1016/j.asr.2018.03.041.

[180] O. Montenbruck, P. Steigenberger, and A. Hauschild. *"Broadcast Versus Precise Ephemerides: A Multi-GNSS Perspective"*. In: *GPS solutions* 19.2 (2015), pp. 321–333. DOI: 10.1007/s10291-014-0390-8.

[181] O. L. Colombo. *"Ephemeris Errors of GPS Satellites"*. In: *Bulletin géodésique* 60.1 (1986), pp. 64–84. DOI: 10.1007/BF02519355.

[182] G. Weber, D. Dettmering, and H. Gebhard. *"Networked Transport of RTCM via Internet Protocol (NTRIP)"*. In: *A Window on the Future of Geodesy*. Springer, 2005, pp. 60–64. DOI: 10.1007/3-540-27432-4_11.

[183] *Radio Technical Commission for Maritime Services*. Accessed: 2020-10-01. URL: https://www.rtcm.org/.

[184] N. Ashby. *"Relativity in the Global Positioning System"*. In: *Living Reviews in relativity* 6.1 (2003), p. 1. DOI: 10.12942/lrr-2003-1.

[185] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. John Wiley & Sons, 2004. DOI: 10.1002/0471221279.

[186] T. Takasu and A. Yasuda. *"Kalman-filter-based Integer Ambiguity Resolution Strategy for Long-baseline RTK with Ionosphere and Troposphere Estimation"*. In: *Proceedings of the ION GNSS*. 2010, pp. 161–171. ION-ID: 9143.

[187] L. Wanninger. *"Carrier-phase Inter-frequency Biases of GLONASS Receivers"*. In: *Journal of Geodesy* 86.2 (2012), pp. 139–148. DOI: 10.1007/s00190-011-0502-y.

[188] P.J.G. Teunissen. *"The Least-squares Ambiguity Decorrelation Adjustment: A Method for Fast GPS Integer Ambiguity Estimation"*. In: *Journal of Geodesy* 70 (1995), pp. 65–82. DOI: 10.1007/BF00863419.

[189] P. De Jonge, C. Tiberius, et al. *The LAMBDA Methods for Integer Ambiguity Estimation: Implementation Aspects*. Vol. 12. Publications of the Delft Geodetic Computing Centre, 1996.

[190] X.-W. Chang, X. Yang, and T. Zhou. *"MLAMBDA: A Modified LAMBDA Method for Integer Least-squares Estimation"*. In: *Journal of Geodesy* 79.9 (2005), pp. 552–565. DOI: 10.1007/s00190-005-0004-x.

[191] A. Gong, X. Zhao, C. Pang, R. Duan, and Y. Wang. *"GNSS Single Frequency, Single Epoch Reliable Attitude Determination Method with Baseline Vector Constraint"*. In: *Sensors* 15.12 (2015), pp. 30093–30103. DOI: 10.3390/s151229774.

[192] C. Park and P.J.G. Teunissen. *"A New Carrier Phase Ambiguity Estimation for GNSS Attitude Determination Systems"*. In: *Proceedings of international GPS/GNSS symposium, Tokyo*. Vol. 8. 2003.

[193] P. Buist. *"The Baseline Constrained LAMBDA Method for Single Epoch, Single Frequency Attitude Determination Applications"*. In: *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2007)*. 2001, pp. 2962–2973. ION-ID: 7647.

[194] C. Park and P.J.G. Teunissen. *"Integer Least Squares with Quadratic Equality Constraints and its Application to GNSS Attitude Determination Systems"*. In: *International Journal of Control, Automation and Systems* 7.4 (2009), pp. 566–576. DOI: 10.1007/s12555-009-0408-0.

[195] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Vol. 3. JHU press, 2012. ISBN: 978-0-94653-600-9.

[196] S. Bisnath and Y. Gao. *"Precise Point Positioning"*. In: *GPS World* 20.4 (2009), pp. 43–50.

[197] J. Kouba. *A Guide to Using International GNSS Service (IGS) Products*. 2009.

[198] P.J.G. Teunissen and A. Khodabandeh. *"Review and Principles of PPP-RTK Methods"*. In: *Journal of Geodesy* 89.3 (2015), pp. 217–240. DOI: 10.1007/s00190-014-0771-3.

[199] University of Wuerzburg - Informatics 8. *RODOS Embedded Operating System*. gitlab. URL: https://gitlab.com/rodos.

[200] Wikipedia. *Rodos (operating system)*. URL: https://en.wikipedia.org/wiki/Rodos_%28operating_system%29.

[201] NXP. *MQX Software Solutions*. URL: https://www.nxp.com/design/software/embedded-software/mqx-software-solutions:MQX_HOME.

[202] STMicroelectronics. *STM32F4 Series*. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f4-series.html.

[203] ARM Technical Support Knowledge Articles. *How do I get the best performance when compiling floating point code for Cortex-M4F?* 2011. URL: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka15451.html.

[204] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes 3rd edition: The Art of Scientific Computing*. 2007.

[205] UdooBoard. *Linux Kernel Udoo Neo*. 2018. URL: https://github.com/fmntf/linux_kernel/tree/4.1.15_2.x-udoo.

[206] ROS. *ROS- Kinetic*. 2016. URL: http://wiki.ros.org/kinetic.

[207] T. Takasu. *RTKLIB: An Open Source Program Package for GNSS Positioning*. Tech. rep. 2013.

[208] K. Panayiotou and A. Triantafyllidiss. *ROS Package to interfere with Flir-Lepton LWIR sensor*. 2015. URL: https://github.com/angetria/flir_lepton.

[209] Intel RealSense Technology. 2018. URL: https://github.com/IntelRealSense/librealsense.

[210] National Centers for Environmental Information. *The World Magnetic Model and Associated Software*. 2019. URL: https://www.ngdc.noaa.gov/geomag/WMM/soft.shtml.

[211] Chrony. *Chrony Introduction*. 2019. URL: https://chrony.tuxfamily.org/.

[212] NXP Semiconductors. *RPMsg-Lite User's Guide*. Github. 2016. URL: https://nxpmicro.github.io/rpmsg-lite/index.html.

[213] OpenAMP. *OpeOpen project*. URL: https://www.openampproject.org/.

[214] NXP Semiconductors. *RPMsg implementation for small MCUs*. Github. 2017. URL: https://github.com/NXPmicro/rpmsg-lite.

[215] R. Russell. *"VIRTIO: Towards a De-facto Standard for Virtual I/O Devices"*. In: *ACM SIGOPS Operating Systems Review* 42.5 (2008), pp. 95–103. DOI: 10.1145/1400097.1400108.

[216] NXP Semiconductors. *Linux sysfs interface for RPMsg*. Github. 2016. URL: https://github.com/NXPmicro/rpmsg-sysfs.

[217] M. Ferguson. *A ROS client library for small, embedded devices, such as Arduino.* Github. URL: https://github.com/ros-drivers/rosserial.

[218] J. Rothe, J. Zevering, M. Strohmeier, and S. Montenegro. *"A Modified Model Reference Adaptive Controller (M-MRAC) Using an Updated MIT-Rule for the Altitude of a UAV"*. In: *Electronics* 9.7 (2020), p. 1104. DOI: 10.3390/electronics9071104.

[219] NVIDIA. *Jetson Xavier NX*. Accessed: 2020-08-22. URL: https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-xavier-nx/.

[220] NVIDIA. *Jetson Nano*. Accessed: 2020-08-22. URL: https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-nano/.

[221] C. Liman. *"Real-Time Embedded Visual Detection and Tracking of UAVs"*. B.Sc. Thesis. University of Wuerzburg, 2020.

[222] T. Mikschl, T. Walter, and S. Montenegro. *"SKITH - The Wireless Satellite"*. In: *4S Symposium*. Sorrento, Italy, 2018. URL: https://atpi.eventsair.com/QuickEventWebsitePortal/4s2018/4s.

[223] S. Beck. *"Komplementäre Positionsbestimmung durch GNSS und UWB für kooperative UAVs"*. M.Sc. Thesis. University of Wuerzburg, 2019.

[224] J.T. Zevering. *"Adaptive Altitude Controller for a Quadrocopter"*. B.Sc. Thesis. University of Wuerzburg, 2018.

[225] L. Grütter. *"Design und Implementierung eines Tailsitters"*. B.Sc. Thesis. University of Wuerzburg, 2016.

[226] P. Lenski. *"Design, Construction and Operation of a Pentacopter"*. M.Sc. Thesis. University of Wuerzburg, 2017.

Research in Aerospace Information Technology

This monograph series is published by the Chair of Aerospace Information Technology (Informatik VIII) of the University of Würzburg and presents innovative research regarding avionic systems for aerospace and terrestrial applications as well as the technology transfer between both fields.
The main research focus is on the development of reliable soft- and hardware for embedded applications that allow the autonomous operation of unmanned systems in challenging environments. This includes the development of new technologies such as wireless communication methods, distributed sensing and control strategies, sensor fusion algorithms, novel navigation methods and concepts for dependable software targeting the irreducible complexity.
Another research focus is on cooperative tasks of multi-agent systems, including homogeneous swarms and arbitrary heterogeneous constellations.
The developed technologies are deployed in numerous real-world applications such as small satellite systems, distributed sensor networks, unmanned aerial vehicles for extreme environments and other experimental platforms.

Herausgeber:

Prof. Dr. Sergio Montenegro

ISSN: 2747-4828

Zitiervorschlag:
Strohmeier, Michael (2021): FARN – A Novel UAV Flight Controller for Highly Accurate and Reliable Navigation. Research in Aerospace Technology, 1.
DOI: 10.25972/OPUS-22313