Ein modellbasiertes Verfahren zur Entwicklung von VTOL-MAVs mittels einer 3D-Echtzeit-Simulation

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Dipl.-Ing. (FH) Thomas Linkugel M.Eng.

aus Winsen (Luhe)

Tübingen
2017

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen
Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation	:	01.12.2017
Dekan	:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter	:	Prof. Dr. Andreas Schilling
2. Berichterstatter	:	Prof. Dr. Hanspeter A. Mallot
3. Berichterstatter	:	Prof. Klaus H. Well, Ph.D.

Für meine Familie

ABSTRACT

The number of available *Micro Air Vehicles* (MAVs) has increased substantially within the last decade, resulting in larger and more diverse applications for unmanned aerial platforms. *Vertical Take-Off and Landing* (VTOL)-MAVs have become especially popular among scientific users. Key research topics include methods for autonomous navigation and exploration, or even the design of novel control strategies. The core of all such flight platforms is the autopilot, that stabilizes the MAV based on the acquired sensor data. Whenever new or enhanced applications need to be integrated into the aircraft, a modification of the autopilot is indispensable. However, significant obstacles and uncertainties have to be overcome in this process which can lead to the total loss of the aircraft.

In this work, a novel model based development approach is presented, by which different VTOL-MAVs are transformed into virtual reality. For this purpose a 3D real-time simulation is developed, based upon a precise modeling of the flight platform, the integrated sensors, actors and the platform-specific dynamic flight characteristics. The core of this simulation is the *Software in the Loop* (SiL) approach, whereby the entire processing software of the autopilot is executed in identical form, both on the simulated as well as on the real flight platform. For this purpose a commercially available quadrocopter is transferred completely to the simulation in the first step. In the second step an innovative framework is developed, which combines a novel autopilot, the simulation as well as a full demonstrator flight platform. This innovative combination allows the design of new algorithms or the optimization of filtering or control parameters within the simulation. The complete development and evaluation can be performed using the virtual MAV. Due to the design in virtual reality, the development time is reduced significantly and, based on the availability of ground truth data, the precision of algorithms can be maximized. After the development phase in the simulation, all algorithms can directly (*Software in the Loop*) be exported onto the real platform. The risk of inadvertently destroying a flight platform during the development phase is reduced significantly. This holistic approach allows the easy and unproblematic integration of this autopilot in different projects.

ZUSAMMENFASSUNG

Innerhalb des letzten Jahrzehnts ist ein deutlicher Anstieg an verfügbaren *Micro Air Vehicles* (MAVs) zu verzeichnen. Immer größer und vielfältiger werden die Einsatzgebiete dieser unbemannten Flugplattformen. Gegenstand aktuellster Forschung ist dabei vor allem die Entwicklung von neuen, innovativen und intelligenten Algorithmen zur automatischen Navigation und Regelung solcher Systeme. Kern aller Flugplattformen bildet dabei der Autopilot. Dieser stabilisiert das MAV auf Basis der erfassten Sensordaten. Sollen nun neue oder erweiterte Anwendungen in die Flugplattform integriert werden, ist ein partieller Eingriff in den Autopiloten unabdingbar. Dabei treten jedoch erhebliche Hindernisse und Unwägbarkeiten auf, welche bis zum Totalverlust der Flugplattform führen können.

Diese Arbeit zeigt einen modellbasierten Entwicklungsansatz, in dem verschiedene *Vertical Take-Off and Landing* (VTOL)-Systeme in die virtuelle Realität überführt werden. Hierzu wird eine 3D-Echtzeit-Simulation entwickelt, in welcher die Flugplattformen detailgenau abgebildet werden. Sämtliche Komponenten, beispielsweise Sensoren oder Antriebe, werden präzise modelliert. Den Kern der Simulation stellt dabei der *Software in the Loop* (SiL)-Ansatz dar, wodurch die gesamte Verarbeitungssoftware des Autopiloten in identischer Form, sowohl in der Simulation als auch auf der realen Flugplattform, ausgeführt wird. Hierzu wird im ersten Schritt ein kommerziell verfügbarer Quadrokopter vollständig in die Simulation überführt. Im zweiten Schritt wird ein innovatives *Framework* entwickelt, welches einen neuen Autopiloten, die Simulation sowie eine vollständige Demonstrator-Flugplattform verbindet. Durch diese neuartige Kombination kann das Design und die Entwicklung sämtlicher Algorithmen innerhalb der Simulation auf dem virtuellen MAV stattfinden. Durch die Verfügbarkeit von *Ground-Truth*-Daten und einer realistischen Flugumgebung können Fehler und Probleme frühzeitig in der Simulation erkannt und beseitigt werden. Erst nach der Evaluation und Verifikation in der Simulation wird die Software ohne weitere Änderungen oder Anpassungen auf die reale Flugplattform überführt. Dieser ganzheitliche Ansatz erlaubt die einfache und unproblematische Integration dieses Autopiloten in unterschiedliche Projekte.

DANKSAGUNG

An erster Stelle möchte ich meinen beiden Betreuern Herrn Prof. Dr. Andreas Schilling und Herrn Prof. Dr. Hanspeter A. Mallot danken, dass sie mir diese Promotion ermöglicht haben, ungeachtet meiner parallelen Vollzeittätigkeit in der Industrie und der damit verbundenen Entfernung zwischen Tübingen und Arnsberg. Meinem Doktorvater Herrn Prof. Dr. Andreas Schilling gilt mein besonderer Dank, nicht allein für die zahlreichen fachlichen Gespräche und den stetigen Ideenaustausch. Vor allem danke ich ihm für sein großes Vertrauen und Interesse, welches er mir und meiner Arbeit stets entgegengebracht hat. Herrn Prof. Dr. Hanspeter A. Mallot danke ich für seine stetige Bereitschaft zu fachlichen Diskussionen, sein Interesse an meiner Arbeit sowie für die Übernahme des Korreferats.

Bei der Firma AirRobot möchte ich mich für die Zurverfügungstellung der Flugplattform *AR100B* sowie den Zugang zur Hard- und Software des Autopiloten bedanken. Des Weiteren danke ich meinen Kollegen für die angenehme Arbeitsatmosphäre und im Besonderen Alexander Wiebe, Daniel Luscher und Rainer Eichhorn für ihre Unterstützung bei einigen meiner Flugexperimente und Testaufbauten sowie Marlies Lübke für das Korrekturlesen verschiedener Publikationen.

Allen Beteiligten des Promotionsverbundes „*Vision based Flying Robots*“ der Universität Tübingen danke ich für die Möglichkeit, an ihren Treffen teilnehmen zu dürfen. Ebenfalls möchte ich den Doktoranden von Prof. Dr. Andreas Schilling für die vielen interessanten Gespräche und die immer zuvorkommende Atmosphäre danken.

Für ihre unermüdliche Hilfe bei der Korrektur dieser Dissertation gilt mein spezieller Dank meiner Verlobten Elena, Dorothea Pleßow sowie Charlotte Ségalen und Martin Bender.

Meinen Eltern möchte ich für ihr stets entgegengebrachtes Interesse an meiner Arbeit und ihre immerwährende Unterstützung danken.

Nicht zuletzt danke ich Elena und unseren beiden großartigen Kindern Maximilian und Lea, die ihre Ansprüche während meiner Promotion am stärksten für meine Ziele zurückstellten. Sie haben mir beständig das Vertrauen und den Rückhalt gegeben und mich fortwährend unterstützt, ungeachtet der zahlreichen Abende und Wochenenden, die sie mich an den Schreibtisch abgetreten haben.

INHALTSVERZEICHNIS

VEREINBARUNG

In dieser Arbeit wird zur eindeutigen Darstellung ein Dezimalpunkt als Dezimaltrennzeichen genutzt. Als Tausendertrennzeichen wird ein nicht umbrechendes Leerzeichen festgelegt. Englische Begriffe werden im Text großgeschrieben und *kursiv* kenntlich gemacht.

Teil I

EINFÜHRUNG UND GRUNDLAGEN

1 EINLEITUNG

Die Automatisierung von unterschiedlichsten stationären Systemen und Fahrzeugen zeigt seit einigen Jahren ein schier unaufhörliches Wachstum. In Zeiten, in denen intelligente Autos, Staubsaugerroboter oder selbstfahrende Rasenmäher nicht mehr der *Science-Fiction* angehören, zeichnet sich ein immenser Markt für fortschrittliche Technologien ab, welcher vor allem in Richtung künstliche Intelligenz voranschreitet. Dabei erstreckt sich die Vielfalt und Einsatzmöglichkeit von intelligenten Systemen über alle Bereiche fahrender, schwimmender oder auch fliegender Agenten. Besonders Verfahren zur intelligenten bzw. autonomen Steuerung dieser Systeme sind Bestandteil aktuellster Forschungen.

1.1 EVOLUTION DER UNBEMANNTEN LUFTFAHRT

1.1.1 *Die Geschichte des Autopiloten*

Historisch begann die Entwicklung der ersten automatisierten Steuerungssysteme Anfang des 20. Jahrhunderts mit der Erfindung des Kreiselkompasses durch den deutschen Wissenschaftler *Dr. Hermann Anschütz-Kaempfe* (siehe [10]). Nur wenige Jahre später wurde auf Basis des Kreiselkompasses eines der ersten automatischen Steuerungssysteme zur Kurskorrektur für die Schifffahrt gebaut. Unklar ist, wer das erste System in ein Schiff integrierte, denn parallel zu den Entwicklungen in Deutschland arbeitete auch der Amerikaner *Elmer A. Sperry* an der Entwicklung von Kreiselkompassen für die Schifffahrt (siehe [143, 151, 163]). Bekannt ist jedoch, dass der Sohn *Sperrys*, *Lawrence B. Sperry*, gemeinsam mit seinem Vater den Kreiselkompass in die Luftfahrt überführte und 1916 den ersten „mechanischen bzw. automatischen Piloten“ für Flugzeuge patentieren ließ (siehe [149, 150]).

In den folgenden Jahren entwickelten sich zunehmend weitere automatische Steuerungssysteme und der Begriff *Automatic Flight Control System* (AFCS), kurz *Autopilot*, etablierte sich (siehe [112]). Mit dem Beginn des zweiten Weltkriegs und der Entwicklung der *A-4* (später umbenannt in *V-2*) Rakete wurde ein neuartiges Flugsteuerungskonzept unter der Leitung von *Wernher von Braun* entwickelt. Dieses basierte auf einem sogenannten Trägheitsnavigationssystem *(engl. Inertial Navigation System* (INS)), welches die Weiterentwicklung des Kreiselkompasses durch Kopplung mit Beschleunigungssensoren darstellte (siehe [23]). Revolutionär zu den bisherigen mechanischen Autopiloten war ein Konzept, welches unter dem Begriff *Fly-by-Wire* in

der heutigen Luftfahrt etabliert ist. Hierbei werden Steuerungsanlagen, wie Ruder oder Leitwerke, nicht mehr direkt vom Piloten über Schubstangen oder Stahlseile bedient, sondern elektronisch gesteuert. Hierzu erzeugt ein Steuerungsrechner elektrische Signale, welche dann von Aktoren an den Steuerungsanlagen umgesetzt werden. Bis in die frühen 1970er Jahre basierten die Steuerungsrechner noch größtenteils auf analogen Computern. 1972 wurde dann von der NASA eine *Vought F-8C Crusader* (Projektname: *F-8 Digital Fly-By-Wire*) mit dem ersten vollständig digitalen *Fly-by-Wire*-System ausgerüstet (siehe [162]).

Mit Einführung von digitalen Autopiloten und *Fly-by-Wire*-Systemen wuchsen auch die Einsatzmöglichkeiten beträchtlich. So konnten nun Flugzeuge entwickelt werden, welche aerodynamisch instabil und somit ausschließlich mit Hilfe eines Autopiloten überhaupt manövrierbar waren. Eines der bekanntesten Flugzeuge dieser Art ist der Nurflügler *B-2 Spirit* von Northrop Grumman (siehe [126]). Neben den bisherigen Flugzeugen, welche Start- und Landebahnen benötigen, rückte nun eine weitere Klasse von Fluggeräten in den Fokus von Autopiloten, sogenannte *Vertical Take-Off and Landing* (VTOL)-Systeme.

1.1.2 *Vertical Take-Off and Landing-Systeme*

Historisch betrachtet ist das Konzept senkrecht startender und landender Flugsysteme nicht neu. Schon Ende des 14. Jahrhunderts entwarf *Leonardo da Vinci* die sogenannte „*Luftschraube*“ (siehe Abbildung 1), welche die Grundlage für heutige Helikopter bilden sollte.

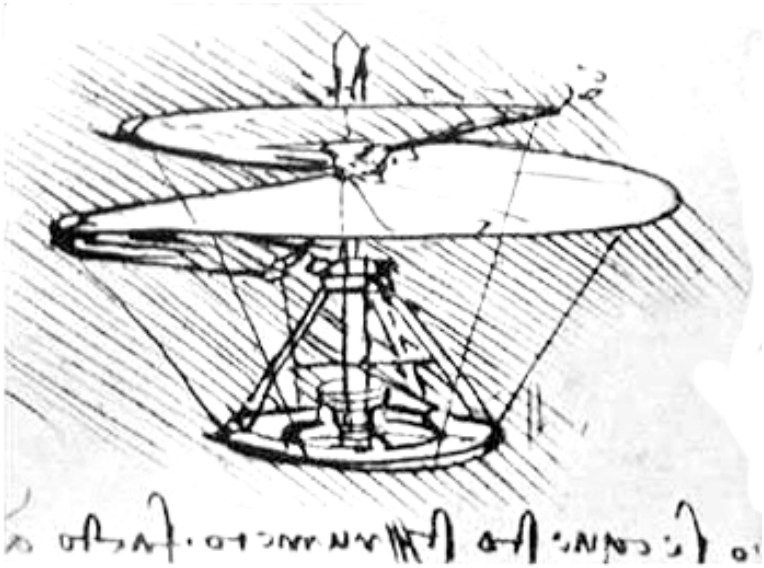


Abbildung 1: *Leonardo da Vincis „Luftschraube“* aus dem Manuskript B, Folio 83v, *(Quelle: Bibliotheque de l'Institut de France, Paris)*

Es sollte jedoch noch weitere fünf Jahrhunderte dauern, bis der französische Ingenieur *Paul Cornu* 1907 den ersten bemannten, senkrecht startenden Drehflügler entwickelte, welcher für einige Sekunden vom Erdboden abheben konnte. Danach vergingen noch weitere zwei Jahrzehnte und mehrere Entwicklungsschritte, zum Beispiel von *Igor I. Sikorsky*, *Jen C. Ellehammer* oder *Stephan Petróczy*, bis in den 1920er Jahren *Etienne Oemichen* eines der ersten stabil fliegenden VTOL-Systeme baute. Mit dem *Oemichen No. 2* entwickelte er einen Quadrotordrehflügler, mit dem er 1924 gleich mehrere Rekorde bei der *Fédération Aéronautique Internationale* (FAI) aufstellte (siehe: [59]). Besonders bedeutend war der Flug einer einen Kilometer langen Kreisbahn in einer Flugzeit von sieben Minuten und 40 Sekunden. Auf Basis dieser frühen Entwicklungen sollten im weiteren Verlauf der Geschichte noch verschiedenste VTOL-Systeme in Erscheinung treten, wie zum Beispiel der *Bell 47* Helikopter (siehe Tabelle 1c auf Seite 6), welcher 1946 als erster Helikopter eine zivile Luftfahrtzulassung in den USA erhielt. Weitere historische Entwicklungen der Drehflügler sind in [94] im Kapitel „*A History of Helicopter Flight*“ zu finden.

Neben den beiden klassischen Ansätzen eines Multirotorsystems, wie dem *Oemichen No.2*, und dem eines Helikopters, wie dem *Bell 47*, wurden im Laufe des 20. Jahrhunderts weitere Konzepte von senkrecht startenden und landenden Systemen verfolgt. Dabei spielten hybride Systeme aus der Kombination von Drehflüglern und konventionellen Flugzeugen eine bestimmende Rolle (siehe Tabelle 1a und 1b). Diese Hybriden sollten die Vorteile eines VTOL-Systems mit der Effizienz eines Tragflächenflugzeugs verbinden. In Tabelle 1 auf Seite 6 ist eine Übersicht der vier Klassen von VTOL-Systemen dargestellt. Obwohl das erste stabil fliegende VTOL-System ein Multirotorsystem war, tendierte die Entwicklung sehr stark zu dem heutigen Helikopter-Ansatz, bestehend aus Haupt- und Heckrotor. Der Ansatz des Quadrotor- bzw. Multirotorsystems wurde bis zum Ende des 20. Jahrhunderts nur bedingt, z. B. durch den *Aerial-Jeep* der Curtiss-Wright Corporation, weiterverfolgt. Erst 60 Jahre später, Anfang des 21. Jahrhunderts, stellte die deutsche Firma e-Volo mit dem *VC200* (siehe Tabelle 1d) im Jahr 2012 erneut ein bemanntes Multirotorsystem für den zivilen Markt vor.

Jedoch etablierte sich mit Hilfe der weitreichenden technologischen Errungenschaften in der bemannten Luftfahrt sowohl durch die fortschrittliche Entwicklung von VTOL-Systemen als auch durch die Erfindung des Autopiloten und des *Fly-by-Wire*-Systems eine neuartige, hochtechnologische Sparte in der Aeronautik. Diese sollte durch fortschrittliche Navigations- und Steuerungssysteme nun auch unbemannte Luftfahrzeuge (*engl. Unmanned Aerial Vehicles* (UAVs)) hervorbringen.

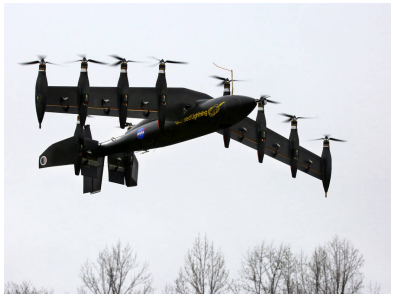
BEISPIEL	KLASSE
 (a) Greased Lightning *(Quelle: ©NASA D.C. Bowman)*	Kippflügler *engl. Tilt-Wing*
 (b) V-22 Osprey *(Quelle: ©Boeing)*	Kipprotor *engl. Tilt-Rotor*
 (c) BELL 47 G-3B-1 *(Quelle: ©The Flying Bulls)*	Helikopter
 (d) VC200 Volocopter *(Quelle: ©e-volo, Nikolay Kazakov)*	Multirotor

Tabelle 1: Arten von *Vertical Take-Off and Landing* (VTOL)-Systemen

1.1.3 *Unbemannte Luftfahrzeuge*

Bevor eine Übersicht zur Entwicklung von UAVs erfolgt, soll zunächst eine Darstellung und Erläuterung der notwendigen Begrifflichkeiten stattfinden. Hierzu wird die amerikanische Bundesluftfahrtbehörde (*Federal Aviation Administration* (FAA)) aus [53] zitiert, in der es heißt:

> „UNMANNED AIRCRAFT *(UA). — The term „unmanned aircraft" means an aircraft that is operated without the possibility of direct human intervention from within or on the aircraft.*"
>
> „UNMANNED AIRCRAFT SYSTEM *(UAS). — The term „unmanned aircraft system" means an unmanned aircraft and associated elements (including communication links and the components that control the unmanned aircraft) that are required for the pilot in command to operate safely and efficiently in the national airspace system.*"

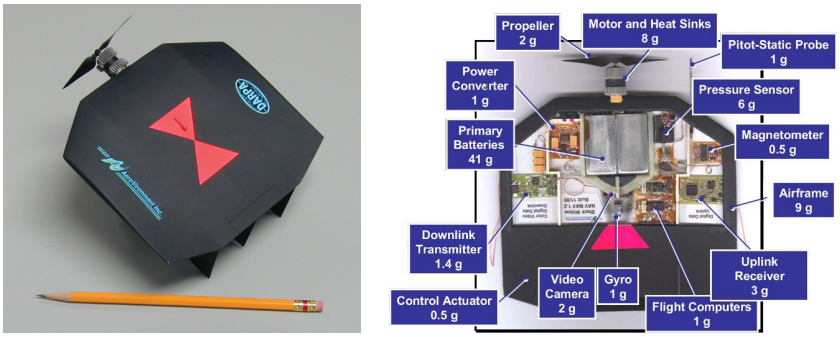
Dabei sind die Begrifflichkeiten UA sowie UAV inhaltlich gleichbedeutend, wobei im weiteren Verlauf dieser Arbeit der allgemein gebräuchlichere Begriff UAV verwendet wird. Im Zusammenhang von unbemannten Luftfahrzeugen wird ebenfalls der Begriff des *Remotely Piloted Aircraft System* (RPAS) benutzt, wobei dieser Begriff inhaltlich gleichbedeutend mit dem Begriff UAS angewendet wird. Auch hier wird im Weiteren mit dem allgemein häufiger genutzten Begriff UAS gearbeitet.

Aus historischer Sicht inspirierten unbemannte Flugsysteme von Anbeginn die Entwicklungen in der Luftfahrt. Dabei dienten diese Systeme zunächst ausschließlich zu Evaluierungs- bzw. Entwicklungszwecken. Wird jedoch als Grundlage für ein unbemanntes Luftfahrzeug die technologische Verfügbarkeit von Navigationssystemen und elektromechanischen Steuerungsanlagen vorausgesetzt, begann die Geschichte der automatisch navigierenden Flugsysteme mit der Entwicklung der *V-2* Rakete. 15 Jahre nach dem Ende des zweiten Weltkriegs entwickelte Lockheed 1960 eines der bis heute schnellsten UAVs in der Geschichte, die *D-21* mit einer Fluggeschwindigkeit von Mach 4. Bis Ende des 20. Jahrhunderts folgten weitere vereinzelte Entwicklungen von unbemannten Systemen. Ab Mitte der 1990er Jahre ist jedoch ein rasanter Anstieg in der Entwicklung von UAVs zu verzeichnen (siehe [169]). Ein Grund dafür könnten die gravierenden technologischen Fortschritte dieser Epoche sein. So konnte die Genauigkeit von Gyroskopen durch die Einführung von Ring Laser Gyroskopen (RLGs) sowie von Faserkreiseln (*engl. Fiber Optic Gyroscopes* (FOGs)) in den 1970er Jahren beträchtlich gesteigert werden. Trotzdem waren diese Kreiselinstrumente immer noch sehr komplex und groß im Aufbau. Einen Durchbruch in der Gyroskoptechnologie erreichte im Jahr 1985 die Firma Silicon Sensing mit der Vorstellung und Markteinführung eines der ersten *Solid State*-Gyroskope auf Basis eines Vibrationskreisels (*engl. Vibrating Structure Gyroscope* (VSG))

(siehe [148]). Zudem stellte in den 1990er Jahren die Firma Analog Devices mit dem *ADXL50* einen als mikroelektromechanisches System (*engl. Micro-Electro-Mechanical System* (MEMS)) gefertigten Beschleunigungssensor für die Großserienproduktion vor. Mit diesen technologischen Errungenschaften war es jetzt nicht nur möglich UAVs zu entwickeln, sondern auch ihre Baugröße beträchtlich zu verringern.

Diesen Meilenstein der Technologieentwicklung erkannte ebenfalls schon Anfang der 1990er Jahre die *Defense Advanced Research Projects Agency* (DARPA) (siehe [76]) und finanzierte von 1996 bis 2000 verschiedene *Micro Air Vehicle* (MAV)-Projekte im Rahmen des *Small Business Innovation Research* (SBIR)-Programms. Zielsetzung dieses Vorhabens war es, die Entwicklung eines maximal 15 Zentimeter (6 inch) großen MAVs, welches in der Lage ist, eine Echtzeit-Videoübertragung zu einer Bodenstation, eine effektive Einsatzreichweite von bis zu 10 Kilometern und Flugzeiten von 20 Minuten bis zu 2 Stunden zu erreichen (siehe [166]). Die Ergebnisse dieses Förderprogramms waren beachtlich. Eines der herausragenden MAVs hieraus war die *Schwarze Witwe* (*engl. Black Widow*) von AeroVironment (siehe Abbildung 2a). Dieses nur 15 Zentimeter große MAV hatte schon im Jahr 2000 bei einem Abfluggewicht von weniger als 100 Gramm einen Autopiloten (siehe Abbildung 2b), welcher eigenständig die Höhe, Fluggeschwindigkeit und den Kurs (*engl. Heading*) kontrollieren konnte. Zudem konnte es bis zu 2 Kilometer ferngesteuert werden und dabei gleichzeitig ein analoges Echtzeit-Video übertragen (siehe [63]).

Aber auch bei der Entwicklung von Multirotorsystemen war dieser technologische Fortschritt revolutionär. Durch die kostengünstige Serienfertigung von Gyroskopen reagierte auch der kommerzielle Markt und präsentierte die ersten automatisch stabilisierten Systeme. Schon zu Beginn der 1990er Jahre erschien auf dem japanischen Markt der Quadrokopter *Keyence Gyrosaucer II*, welcher zwei Gyroskope integriert hatte, mit denen er automatisch seine Lage stabilisieren konnte (siehe [3, 4]). Gleichzeitig wurde in den USA und Deutschland an derartigen VTOL-MAVs intensiv entwickelt. Ende der



(a) Black Widow (*Quelle: [176]*) (b) Komponenten (*Quelle: [176]*)

Abbildung 2: *Black Widow*-MAV von AeroVironment

(a) Roswell Flyer (*Quelle: M. Dammar*) (b) Intellicopter (*Quelle: ©WIWA-VI*)

Abbildung 3: Quadrokopter – der Anfang

1990er Jahre stellte *M. Dammar* den *Roswell Flyer* vor (siehe Abbildung 3a), der zukunftsweisend schon drei Gyroskope zur Erfassung der Drehbewegungen einsetzte. Ab 1999 übernahm Draganfly Innovations (siehe [43]) diese Technologie. Diese ersten Quadrotor-MAVs hatten jedoch lediglich eine Flugzeit von drei bis fünf Minuten. Im Jahr 2003 erschien einer der ersten Multikopter, der in Deutschland kommerziell vertrieben wurde. Der *Intellicopter* (siehe Abbildung 3b) von der WIWA-VI (siehe [174]), die bis heute unter dem Label AirRobot (siehe [6]) VTOL-MAVs entwickelt, nutzte ebenfalls drei *Solid State* MEMS-Gyroskope zur Lageregelung. Im Gegensatz zu den bisherigen Flugplattformen hatte dieser Quadrokopter durch Nutzung der neuartigen Lithium-Polymer (Li-Po)-Batterietechnologie sowie den Einsatz von Leichtbau-Verbundwerkstoffen, wie carbonfaserverstärktem Kunststoff (CFK), eine deutlich gesteigerte Flugzeit von etwa 15 bis 20 Minuten. Neben diesem kommerziellen System erzielten im selben Jahr *K.-M. Doth* und *D. Gurdan* (beide später Mitgeschäftsführer von Ascending Technologies (siehe [13])) mit ihrem Quadrokopter den 4. Platz (siehe [88]) beim *Jugend forscht* Wettbewerb. Dieses System wurde später von Silverlit als *X-UFO* mit einem mechanischen Kreiselsystem für den Hobby- und Spielzeugmarkt vertrieben. Darauf folgend ist ab Mitte der 2000er Jahre die Entwicklung von MAVs, im Besonderen von Multirotor-VTOL-Systemen, geradezu explodiert.

1.2 MOTIVATION UND ZIELSETZUNG

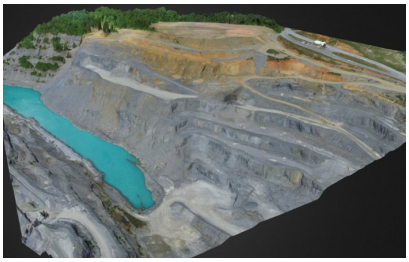
Wie zuvor dargestellt, ist innerhalb des letzten Jahrzehnts ein deutlicher Anstieg an verfügbaren UAVs zu verzeichnen. Immer größer und vielfältiger werden dabei die Einsatzgebiete von unbemannten Flugplattformen, wodurch, neben dem klassischen militärischen Einsatz, immer mehr zivile und wissenschaftliche Anwendungen in den Fokus rücken. So sind zum Beispiel nach der Atomkatastrophe von Fukushima im Jahr 2011 gleich zwei unterschiedliche UAVs zur unbemannten Aufklärung eingesetzt worden (siehe [16, 74]). Ebenso wird

an Strategien gearbeitet, wie die Nutzung von unbemannten Systemen zukünftig Unglücke vermeiden kann. So bekam Anfang 2015 BNSF Railway erstmals in den USA eine offizielle Freigabe zum Betrieb unbemannter Multirotorsysteme zur Inspektion von Bahngleisen (siehe [54, 160]). Gerade die Inspektion, Vermessung oder Dokumentation von technischen Infrastrukturen (siehe Abbildung 4a), Industrieanlagen (siehe Abbildung 4b oder 4c), Denkmälern oder historischen Bauwerken (siehe Abbildung 4d) aus der Luft zeigen einen erheblichen praktischen, aber auch wirtschaftlichen Mehrwert. Im Gegensatz zu den konventionellen Methoden oder einer klassischen bemannten Befliegung mittels Helikopter oder Flugzeug können mit diesen deutlich kleineren, unbemannten Systemen photogrammetrische Bestimmungen aus nahezu allen Perspektiven vorgenommen werden. Dabei dezimiert sich sowohl der planerische als auch der kostentechnische Aufwand um ein Vielfaches.

Neben diesen, zum Teil sicherheitskritischen und spezialisierten Anwendungsfeldern wird ebenfalls an alltäglicheren Anwendungen gearbeitet. So untersuchte der zur Deutschen Post AG gehörende Express-Lieferdienst DHL in einem Forschungsprojekt, in Kooperation mit der Rheinisch-Westfälischen Technischen Hochschule (RWTH) Aachen und dem UAV-Hersteller Microdrones, den Transport von Warensendungen mit einem UAV, dem sogenannten *Paketkopter* (siehe [38]). Parallel zu diesem Vorhaben wird auch international an

(a) Inspektion eines Flugzeugs (*Quelle: ©Airbus S.A.S, R. Gnecco*)

(b) Inspektion einer Biogasanlage mit Farb- und Thermalkamera (*Quelle: ©AirRobot*)



(c) 3D-Rekonstruktion eines Steinbruchs (*Quelle: ©DroneDeploy* [18])

(d) 3D-Rekonstruktion eines historischen Bauwerks (*Quelle: ©T. Kanand*)

Abbildung 4: Einsatzmöglichkeiten von MAVs

der Lieferung von unterschiedlichsten Objekten gearbeitet. Der US-amerikanische Versandhändler Amazon arbeitet an der *Amazon Prime Air*, die zukünftig Bestellungen von bis zu 2.27 Kilogramm (5 pounds) mittels UAV in weniger als 30 Minuten an den Kunden ausliefern soll (siehe [7]). Auch der US-amerikanische Technologiekonzern Google arbeitet innerhalb seiner Forschungsabteilung Google[x] an einem UAV, das, wie *Amazon Prime Air*, Pakete in wenigen Minuten zum Kunden bringen soll. Das Besondere an Googles *Project Wing* ist jedoch das Konzept. Im Gegensatz zu den Multirotorkonzepten von DHL und Amazon setzt Google auf eine Kombination aus einem vierrotorigen VTOL-System in Kombination mit einem Tragflächenflugzeug, bei dem die Ware mittels einer Seilwinde abgesetzt werden soll. Google plant dabei, innerhalb der nächsten zwei Jahre ein marktreifes Liefersystem zu präsentieren, bei dem es im Besonderen auf die autonomen Flugfähigkeiten ankommen soll (siehe [15]).

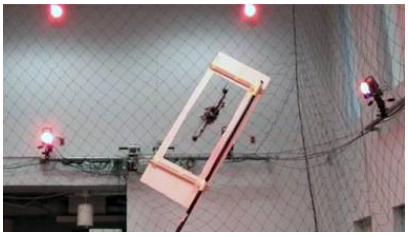
Durch diese aktuellen und zukunftsweisenden professionellen Anwendungsgebiete und den überaus motivierten Forschungsanstrengungen kapitalstarker Unternehmen hat das wirtschaftliche Wachstum des UAV-Marktes ein enormes Potential entwickelt. Allein für die USA wird in den zwei Jahren von 2015 bis 2017 ein ökonomischer Gesamtumsatz von 13.6 Milliarden Dollar erwartet. Bis zum Jahr 2025 soll dieser sogar auf geschätzte 82.1 Milliarden Dollar steigen und mehr als hunderttausend neue Arbeitsplätze geschaffen werden (siehe [84]).

Nicht zuletzt steigt die Dichte an betriebsbereiten UAVs durch den Hobbysektor. So sind von der *AR.Drone* des französischen Unternehmens Parrot seit der Markteinführung 2010 schon mehr als eine halbe Million Exemplare verkauft worden (siehe [132]). Ebenso zeigt das chinesische Technologieunternehmen DJI mit seinen verschiedenen Multirotorsystemen für den Endverbraucher ein außerordentliches Marktpotenzial und erwirtschaftete 2013 einen Umsatz von 130 Millionen Dollar. Zusätzlich wurde ein sogenannter *Venture Capital-Fonds* gegründet, wodurch sich nach aktuellen Schätzungen der Wert des Unternehmens auf etwa 10 Milliarden Dollar beläuft (siehe [40]).

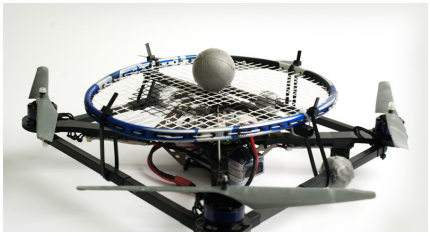
Neben dem rasant ansteigenden wirtschaftlichen Interesse an dieser Technologie sind auch wissenschaftliche Forschungen und Entwicklungen sowie die Nutzung von UAVs, speziell MAVs, in unterschiedlichen Disziplinen beträchtlich gestiegen. So zeigen [110] oder auch [175] den Einsatz eines UAVs zur Erfassung von meteorologischen Daten. Neben dieser Anwendung können auch Forschungsdisziplinen wie die Zoologie vom Einsatz von MAV-Systemen profitieren, wie in [45] für photogrammetrische Studien bei Walen in freier Wildbahn gezeigt wird. Sehr verbreitete Einsatzgebiete von unbemannten Systemen sind in der Geologie und Geodäsie zu finden. Naturgemäß sind hier die zu erfassenden Objekte in meist entlegenen oder schwer zugänglichen Gebieten verortet, wodurch sich die Nutzung von unbe-

mannten Systemen anbietet. In [125] und [156] wird beispielhaft vorgestellt, wie die Erfassung und Beobachtung von alpinen Rutschhängen durch Nutzung eines Multirotor-MAVs erheblich vereinfacht und die Effektivität gesteigert werden kann. In [47] wird grundlegend dargestellt, wie UAVs in der Photogrammetrie gewinnbringend eingesetzt werden können, und es werden verschiedene Anwendungsgebiete in dieser Disziplin aufgezeigt. Neben diesen Einsatzmöglichkeiten ist die Grundlagenforschung im Bereich der unbemannten Systeme ein ebenso elementarer Bestandteil der wissenschaftlichen Arbeit. Die bekanntesten Forschergruppen sind zum Beispiel an der Eidgenössischen Technischen Hochschule (ETH) in Zürich (siehe [48]), an der Universität von Pennsylvania (siehe [64]) oder auch an den Universitäten von Stanford und Berkeley (siehe [77]) zu finden. Ein zentrales Forschungsthema bildet dabei der Entwurf neuartiger Regelungsstrategien für dynamische und präzise Flugmanöver. In [107] wird beispielsweise eine adaptive Reglerstrategie vorgestellt, die es dem MAV erlaubt, optimale Flips iterativ anhand von diskreten Schlüsselsequenzen zu erlernen. Weiter wird in [116] eine Methode vorgestellt, welche es Quadrokoptern erlaubt, durch schmale, senkrechtstehende Spalten zu fliegen (siehe Abbildung 5a) oder sich kopfüber an Oberflächen anzudocken. Auch unerwartete Exkurse in den Sport (siehe [121]), wo Quadrokopter mit Bällen jonglieren (siehe Abbildung 5b), oder in die Musikwelt (siehe [39]), in der MAVs ein ganzes Orchester ersetzen, werden in dieser Disziplin gezeigt.

Ein ebenso signifikantes Forschungsgebiet liegt in der autonomen Navigation und Exploration. Hier werden die unbemannten Systeme mit weiteren Sensoren wie Laserscannern, Kameras, Radar etc. ausgestattet, um die Umgebung zu erfassen, zu kartieren und sich selbst in dieser zu lokalisieren. In [68] wird beispielsweise ein 2D-*Simultaneous Localization and Mapping* (SLAM)-Verfahren zur *Indoor*-Navigation für ein Quadrotor-MAV gezeigt. In [170] werden indes visuelle (kamerabasierte) Verfahren untersucht, um einem MAV auch in großen und unbekannten Umgebungen eine langfristige Navigation zu ermögli-



(a) Aggressive Flugmanöver (*Quelle: ©D. Mellinger*)



(b) Tennisspielender Quadrokopter (*Quelle: ©Institute for Dynamic Systems and Control, ETH Zurich, R. D'Andrea*)

Abbildung 5: Quadrokopter – in der Forschung

chen. Einen ebenso visuellen, jedoch konzeptionell anderen Ansatz, verfolgen die Autoren in [108]. Hier werden zur Lokalisierung in urbanen Straßen die visuellen Daten des MAVs mit Bilddaten aus einem texturierten 3D-Stadtmodell, wie es zum Beispiel ©*Google Street View* bereitstellt, verglichen.

Die Breite an Forschungsgebieten ist immens. Dabei erstrecken diese sich von den genannten Schwerpunkten, in denen das UAV als Individuum betrachtet wird, auch immer weiter in Richtung Schwarmintelligenz, in der mehrere unbemannte Systeme im Kollektiv ein vorgegebenes Ziel verfolgen.

Durch die gezeigten wirtschaftlichen wie auch wissenschaftlichen Perspektiven wird prädiziert, dass sowohl die Verfügbarkeit als auch die Nutzung von UAVs in den nächsten Jahren weiter rapide zunehmen wird. Damit wachsen im gleichen Maße jedoch auch die Anforderungen an die Flugsysteme selbst, vor allem aber auch an deren Autopiloten. Sichere, und effektive Entwicklungsmethoden sind daher wesentliche Anforderungen an zukünftige Systeme. Aus diesen Faktoren ist die Zielsetzung dieser Arbeit abzuleiten. Dabei sollen in diesem Rahmen vor allem drei markante Herausforderungen für die aktuelle und zukünftige Entwicklung von MAVs untersucht werden. Diese sind:

- Erhöhung der Sicherheit der Systeme
- Effektivität und Optimierung der Entwicklung
- Innovationsfähigkeit und wissenschaftliche Einbindung

Dementsprechend soll eine innovative Entwicklungsumgebung für Multirotor-MAVs erarbeitet werden, welche eine detaillierte, echtzeitfähige Simulationsumgebung mit einem neuartigen, plattformunabhängigen Autopiloten verknüpft. Ziel ist es, einen Systementwurf darzustellen, welcher auf unterschiedlichen VTOL-MAVs eingesetzt werden kann. Hierbei steht jedoch nicht allein der plattformunabhängige Autopilot im Vordergrund, sondern ein vollständiges *Framework* zur MAV-Entwicklung. Ein besonderer Fokus wird deswegen auf das echtzeitfähige Simulationssystem gelegt. Dieses ermöglicht das Design von neuen Algorithmen sowie die Optimierung von Filter- oder Reglerparametern. Dabei können alle Entwicklungsschritte vollständig am virtuellen MAV erarbeitet und evaluiert werden. Bedingt durch den Entwurf in der virtuellen Realität, kann die Entwicklungszeit maßgeblich verkürzt sowie, auf Basis der präzisen *Ground-Truth*-Daten, die Genauigkeit von Algorithmen maximiert werden. Nach der Entwicklungsphase in der Simulation können sämtliche Algorithmen etc. direkt (*Software in the Loop* (SiL)) auf die reale Plattform übertragen werden. Durch diese Kombination aus virtuellem Systementwurf, präzisen *Ground-Truth*-Daten und einem einheitlichen Autopiloten für vielfältige MAVs ist es möglich, eine Vielzahl von Rahmenbedingungen, externen Einflüssen und Fehleranalysen während der

Entwicklungsphase abzudecken. Dies dient sowohl der Sicherheit dieser Systeme, den Innovationsmöglichkeiten als auch einer effektiven Entwicklung und damit nicht zuletzt der Wirtschaftlichkeit der Umsetzung neuer Konzepte.

1.3 GLIEDERUNG DER ARBEIT

Grundsätzlich gliedert sich diese Arbeit in drei Hauptabschnitte:

1. Einführung und Grundlagen
2. Simulationsbasierte Entwicklung von Multirotorsystemen
3. Zusammenfassung und Diskussion

Im Grundlagenabschnitt beginnt Kapitel 2 mit der Einführung in die Multirotorsysteme. Dies umfasst im Besonderen den Stand der Technik verfügbarer Flugplattformen sowie die Erörterung des Funktionsprinzips dieser Systeme. Ebenfalls wird an dieser Stelle die Ausgangsplattform dieser Arbeit vorgestellt.

Aufbauend auf dem grundsätzlichen Funktionsprinzip beschreibt Kapitel 3 die benötigten Koordinatensysteme und die zugehörigen Transformationen.

Mit diesem Wissen erfolgt die ausführliche Herleitung aller genutzter Sensoren und Messsysteme in Kapitel 4. Hier wird sowohl auf die Funktionsprinzipien der unterschiedlichen Sensoren als auch auf die Verarbeitungsmechanismen der Messsysteme eingegangen. Angefangen mit der Diskussion des *Strapdown*-Algorithmus über die Verarbeitung der barometrischen Luftdruckmessung erfolgt des Weiteren die Beschreibung der Erdmagnetfeldmessung sowie die Verarbeitung der Daten globaler Navigationssatellitensysteme (*engl. Global Navigation Satellite System* (GNSS)).

Der Grundlagenabschnitt schließt mit der Diskussion und Herleitung von etablierten Verfahren zur Datenfilterung und Sensorfusion in Kapitel 5. Neben der Sensordatenvorverarbeitung mittels digitaler Filter, wie Transversal- und Rekursivfilter, werden an dieser Stelle zudem das Konzept und die Umsetzungsvarianten des Kalman-Filters detailliert hergeleitet.

Nach der Themeneinführung und der Darstellung der Grundlagen erfolgt im zweiten Abschnitt die Vorstellung des innovativen *Frameworks* zur simulationsbasierten Entwicklung von MAVs.

Kapitel 6 startet mit der Einführung in die Simulation von Multirotorsystemen. Neben der Beschreibung des Stands der Technik erfolgt ebenfalls an dieser Stelle der Systemüberblick und das Konzept der hier entwickelten Simulation. Dabei wird besonders auf das *Software in the Loop* (SiL)-Integrationskonzept eingegangen.

Eine wesentliche Voraussetzung für die realitätsgetreue Abbildung in der Simulation ist die Bestimmung eines präzisen Modells aller

Komponenten der Flugplattform. Dementsprechend wird die Systemanalyse und Modellierung in Kapitel 7 detailliert diskutiert.

Mit den dort hergeleiteten mathematischen Modellen für Sensoren, Antriebe und den flugdynamischen Kenngrößen wird in Kapitel 8 anhand der Ausgangsplattform die Überführung in die virtuelle Realität erläutert. Anhand von einem *Indoor-* sowie einem *Outdoor-*Szenario werden daraufhin die Ergebnisse dieser Simulation vorgestellt und diskutiert.

Nach der Vorstellung der entwickelten Simulationsumgebung zeigt Kapitel 9 ein neues und innovatives *Framework* zur effizienten und risikoarmen Entwicklung von VTOL-MAVs. Der Umfang dieses MAV-*Frameworks* geht dabei über die zuvor erarbeitete Simulationsumgebung deutlich hinaus. Dementsprechend wird ebenfalls ein neuer Autopilot für den plattformunabhängigen Einsatz in diesem Kapitel beschrieben. Das Besondere an diesem Konzept ist die vollständige Entwicklung aller Softwarekomponenten innerhalb der Simulation. Dies wird anhand der entwickelten Algorithmen zur Datenvorfilterung und -fusion demonstriert. Ebenfalls geht dieses Kapitel auf die entwickelte Hardware sowie auf die entworfene Demonstrator-Flugplattform detailliert ein. Daraufhin werden die Ergebnisse der erfolgreichen Integration des Autopiloten auf zwei unterschiedliche Flugplattformen gezeigt. Ein besonderer Mehrwert dieses *Frameworks* wird abschließend anhand eines neuartigen Verfahrens zum automatischen Starten aus komplexen und hochdynamischen Situationen nachgewiesen.

Die Arbeit schließt mit der Zusammenfassung der Ergebnisse in Kapitel 10. An dieser Stelle findet ebenfalls ein Ausblick auf zukünftige Arbeiten und Anwendungsmöglichkeiten statt.

1.4 EIGENE PUBLIKATIONEN

Im Rahmen dieser Dissertation sind zudem folgende Publikationen entstanden:

1. MODELLBASIERTE 3D-ECHTZEIT-SIMULATION VON MICRO-UAS,
   Linkugel, T., Schilling, A. und Mallot, H. A.,
   Photogrammetrie, Laserscanning, Optische 3D-Messtechnik,
   Beiträge der Oldenburger 3D-Tage 2012,
   Seiten 245 – 254, 2012

2. MODELLBASIERTE 3D-ECHTZEIT-SIMULATION VON MICRO-UAS,
   Linkugel, T., Schilling, A. und Mallot, H. A.,
   Allgemeine Vermessungsnachrichten (AVN) - 03/2013,
   Seiten 90 – 97, 2013

3. MUSTERBASIERTE OPTISCHE POSITIONSSCHÄTZUNG ZUR REGELUNG EINES MICRO-UAV – EIN SIMULATIONSBASIERTER ANSATZ,
   Linkugel, T., Schilling, A. und Mallot, H. A.,

Photogrammetrie, Laserscanning, Optische 3D-Messtechnik,
Beiträge der Oldenburger 3D-Tage 2013,
Seiten 344 – 355, 2013

4. Another Step towards Measuring the World from the Air: Model-based 3D Real-time Simulation of Micro-UAV
Linkugel, T. und Schilling, A.
Photogrammetric Week '13,
Seiten 181 – 191, 2013

5. Musterbasierte optische Positionsschätzung zur Regelung eines micro-UAV: ein simulationsbasierter Ansatz,
Linkugel, T., Schilling, A. und Mallot, H. A.,
Allgemeine Vermessungsnachrichten (AVN) - 10/2013,
Seiten 331 – 341, 2013

6. Simulation Based Development of Micro Air Vehicles – One step further towards a more efficient development
Linkugel, T. und Schilling, A.
International Micro Air Vehicle Conference and Flight Competition (IMAV) 2015 Aachen,
2015

7. The FireFly MAV-Framework – Closing the Gap in Micro Air Vehicle Development
Linkugel, T., Schilling, A. und Mallot, H. A.,
Photogrammetrie, Laserscanning, Optische 3D-Messtechnik,
Beiträge der Oldenburger 3D-Tage 2016,
Seiten 178 – 192, 2016

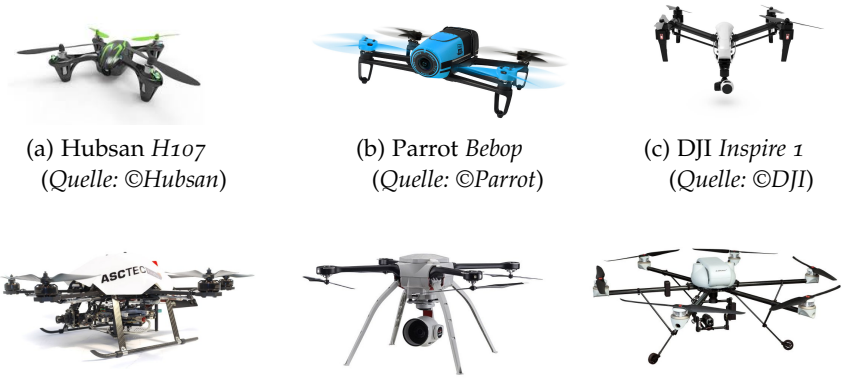
2 MULTIROTORSYSTEME

Wie im vorherigen Kapitel einleitend dargestellt, ist innerhalb des letzten Jahrzehnts ein deutlicher Anstieg an verfügbaren UAVs zu verzeichnen. Durch immer größer und vielfältiger werdende Aufgabengebiete von unbemannten Flugplattformen besteht eine ebenso vielfältige Verfügbarkeit von unterschiedlichsten Systemen. Im folgenden Kapitel soll daher der derzeitige Stand der Technik, das grundlegende Funktionsprinzip von Multirotorsystemen und die Ausgangssituation dieser Arbeit diskutiert werden.

2.1 STAND DER TECHNIK

Multikoptersysteme sind heutzutage in den verschiedensten Varianten und in einem Preisbereich von weniger als 100 € bis hin zu mehr als 100 000 € verfügbar. Ebenso wie der Preis variiert bei diesen Systemen gleichermaßen Größe, Ausstattung und Flugzeit. Die günstigsten Systeme sind im Hobby- bzw. Spielzeugsegment zu finden. Hier sind Quadrokopter, wie der in Abbildung 6a auf Seite 18 dargestellte Hubsan *H107*, für etwa 50 € bis 100 € zu bekommen. Dabei bieten selbst derart günstige Modelle schon viele Features, wie eine automatische Lagestabilisierung oder eine analoge Videoübertragungseinheit. Ein weiteres System für den Hobbyanwender ist 2014 von der Firma Parrot mit der *Bebop* vorgestellt worden (siehe Abbildung 6b). Dieser Quadrokopter hat bei einer Baugröße von nur 33 cm × 38 cm × 3.6 cm und einem Gewicht von nur 420 g trotzdem eine sehr innovative Ausstattung, zum Beispiel *Full High Definition* (HD)-Videoübertragung per *WiFi*, vollständig digitale 3-Achsen Bildstabilisierung oder auch eine automatische Positions- und Lagestabilisierung. Dabei liegt der Anschaffungspreis bei lediglich 400 € bis 800 €. Nachteilig bei diesen beiden sehr kleinen Systemen ist jedoch die Flugzeit von etwa 5 bis 11 Minuten sowie die zum Teil begrenzte Reichweite der Fernsteuerungen. Das, wie schon einleitend dargestellt, sehr erfolgreich agierende Unternehmen DJI versucht diesen Einschränkungen entgegenzuwirken und verfolgt dabei den Übergang vom Hobby- zum semiprofessionellen System. Das aktuelle DJI Modell, die *Inspire 1* (siehe Abbildung 6c), liegt mit einem Anschaffungspreis von etwas über 3000 € schon deutlich über den gängigen Hobbysystemen. Dafür zeigt die *Inspire 1* jedoch auch markante Unterschiede in der Flugsteuerungsreichweite mit angegebenen 2 km und einer gesteigerten Flugzeit von bis zu 18 Minuten. Die Ausstattung dieses Systems ist mit einem 3-Achsen stabilisierten Kameragim-

bal mit integrierter 4 K Kamera und einer digitalen HD-Videoübertragung zum Piloten gleichfalls auf dem neuesten Stand. Aktuellste Entwicklungen bei DJI zeigen mit der *Matrice 100* die Bemühungen um einen weiteren vielversprechenden Markt. So präsentierten sie Mitte 2015 ein weitgehend offenes System mit den technischen Besonderheiten der *Inspire 1* in Kombination mit verschiedenen *Software Development Kits* (SDKs) für den Forschungs- und Entwicklungsmarkt. Einen Markt, der bisher besonders von der *Research Line* der Firma Ascending Technologies, zum Beispiel mit ihrem Hexakopter *AscTec Firefly* (siehe Abbildung 6d), dominiert wurde. Diese Systeme bilden eine besondere Klasse, da ihr Fokus vor allem auf wissenschaftliche Nutzer ausgelegt ist. Dabei sind Schnittstellen zu Entwicklungswerkzeugen, wie zum Beispiel *MATLAB & Simulink* oder dem *Robot Operating System* (ROS), gleichermaßen von Interesse wie SDKs zur direkten Interaktion mit dem System. Speziell auf diese Gruppe der Multirotor-MAVs, besonders auf die *Research Line* von Ascending Technologies und auf die MikroKopter-Systeme von HiSystems, wird in Kapitel 9 nochmals eingegangen. Neben diesen Gruppen aus Hobby-, semiprofessionellen und wissenschaftlichen Flugsystemen soll nun die vierte und letzte Gruppe diskutiert werden. Diese Systeme werden vorwiegend in kommerziellen Anwendungen, wie zur Inspektion oder Vermessung, eingesetzt oder dienen behördlichen Aufgaben. Wobei diese Gruppe selbst in zwei Kategorien geteilt werden soll: zum einen in Systeme, die in einem Preissegment von etwa 15 000 € bis 50 000 € liegen und zum anderen in die Kategorien ab 50 000 € bis über 100 000 € Anschaffungskosten pro System. Die Systeme in der ersten Kategorie haben sehr ähnliche Ausstattungscharakteristika. Üblicherweise liegen hier die Flugzeiten zwischen 20 bis 35 Minuten bei einer effektiven Steuerungsreichweite von 500 m bis 1000 m. Dabei können unterschiedliche Farbkameras aus dem *Consumer*-Bereich oder auch



(a) Hubsan *H107* (*Quelle: ©Hubsan*)

(b) Parrot *Bebop* (*Quelle: ©Parrot*)

(c) DJI *Inspire 1* (*Quelle: ©DJI*)

(d) AscTec *Firefly* (*Quelle: ©Ascending Technologies*)

(e) Aeryon *Skyranger* (*Quelle: ©Aeryon Labs*)

(f) AirRobot *AR200* (*Quelle: ©AirRobot*)

Abbildung 6: Übersicht von VTOL-MAVs

Wärmebildkameras in diese Systeme integriert werden. Die Bildübertragung zum Piloten wird in der Regel analog, selten digital, mit einer *Standard Definition* (SD)-Videoqualität übermittelt. Die Nutzlast kann dabei 200 g bis 250 g, zum Beispiel bei der Microdrones *MD4-200*, oder 800 g bis 1000 g beim *Falcon 8* von Ascending Technologies bzw. der Microdrones *MD4-1000*, bis hin zu 2000 g beim Aibotix *Aibot X6* reichen. Die zweite Kategorie weicht von diesen Merkmalen ab. So liegt der *Skyranger* (siehe Abbildung 6e) der kanadischen Firma Aeryon Labs zwar in einem ähnlichen Größen- bzw. Gewichts- und Nutzlastkapazitätsbereich wie die zuvor genannten Systeme, hebt sich jedoch in der spezifischen Ausstattung deutlich ab. Damit ist dies eines der wenigen Systeme, das in dieser Größe eine Flugzeit von bis zu 50 Minuten bei einer effektiven Reichweite für Fernsteuerung und *Full* HD-Videoübertragung von bis zu 3 km bzw. 5 km bietet. Außerdem kann dieses System mit einer 3-Achsen stabilisierten Kamera geordert werden, die mit einem 30-fach optischen Zoomobjektiv und einer Fotoauflösung von 20 Megapixeln aufwartet. Gegenüber dem *Skyranger* steht beispielsweise der *AR200* (siehe Abbildung 6f) der Firma AirRobot. Dieser liegt zwar mit dem Anschaffungspreis in derselben Kategorie, verfolgt jedoch ein anderes Ziel. So lag ein Schwerpunkt bei der Entwicklung des *AR200*, dessen Autopiloten-Software mit der in dieser Arbeit vorgestellten Simulationsumgebung entwickelt worden ist, auf der Kombination eines Fluggerätes mit möglichst großer Nutzlastkapazität bei trotzdem ausdauernder Flugzeit. Dementsprechend kann dieses System Nutzlasten von bis zu 3000 g bei Flugzeiten bis zu 35 Minuten transportieren. Dabei unterscheidet sich dieses System deutlich in der Größe mit 2.2 m Durchmesser und einem Abfluggewicht von 12 kg von den bisher vorgestellten Systemen.

Diese exemplarisch gewählten Modelle von Flugplattformen sollen in erster Linie aufzeigen, welches Spektrum derzeit auf dem Markt an VTOL-MAVs zur Verfügung steht. Prinzipiell kann festgestellt werden, dass die Systeme in fast allen Kategorien ähnliche Charakteristika der automatischen Fähigkeiten besitzen. So verfügen die Systeme, ausgenommen die Hobbysysteme im untersten Preissegment, über eine automatische Lage- und Positionshaltung sowie über ein automatisiertes Abfliegen einer vorgegebenen Wegpunktstrecke (Wegpunktnavigation). Maßgebliche Unterscheidungen sind derzeit vor allem in der Flugzeit, in der Nutzlastkapazität oder in der technischen Ausstattung, wie Kameras, Videoübertragung oder Steuerungsreichweite, zu finden. Eine Ausnahme zu den vorgestellten Systemen bildet das zum Zeitpunkt der Fertigstellung dieser Arbeit von DJI vorgestellte System *Mavic Pro*. Dieser Quadrokopter zeigt ein herausragendes Spektrum an neuartigen Funktionalitäten bis hin zu einem optischen Kollisionsvermeidungssystem bei einem Anschaffungspreis von weniger als 1500 €. Auf dieses MAV wird nochmals am Ende der Arbeit in Kapitel 10 *Zusammenfassung und Ausblick* Bezug genommen.

2.2 FUNKTIONSPRINZIP

Wie schon in Kapitel 1.1.2 vorgestellt, hat die Entwicklung von Drehflüglersystemen schon früh, beispielsweise 1920 mit dem *Oemichen No.2*, begonnen. Hierbei unterscheiden sich die Multirotorsysteme maßgeblich in ihrem Aufbau und ihrer Funktion von ihrem Artverwandten, dem Helikopter. Um im Folgenden das Funktionsprinzip und vor allem die charakteristischen Eigenschaften der Multirotorsysteme zu beschreiben, soll zuvor ein kurzer Exkurs zum Helikopter gemacht werden.

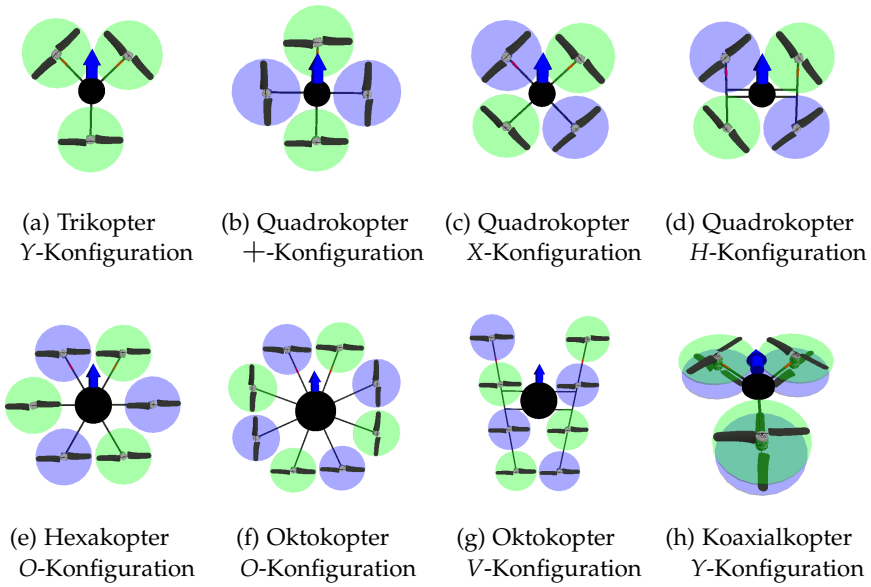
Eine Besonderheit dieses Drehflüglerprinzips ist die gleichbleibende Drehzahl des Haupt- sowie des Heckrotors. Eine vertikale Bewegung, also der Steig- bzw. Sinkflug, wird durch die *kollektive* Blattverstellung (*engl. Collective Pitch*) des Hauptrotors erzeugt. Durch die damit verbundene gleichmäßige Veränderung des Anstellwinkels der Rotorblätter wird die Auftriebskraft kontrolliert. Analog dazu ist eine Horizontalbewegung durch die *zyklische* Blattverstellung (*engl. Cyclic Pitch*) möglich. Hier wird der Anstellwinkel der Rotorblätter jeweils nur rotationsperiodisch verstellt, was zu einem Drehmoment und damit zum Nicken (*Pitch*) oder Rollen (*Roll*) des Helikopters führt. Gesteuert werden diese Blattverstellungen in der Regel durch eine Taumelscheibe. Um den Helikopter ebenfalls um die Hochachse steuern zu können (Gieren oder *Yaw*), wird der Anstellwinkel der Heckrotorblätter verändert. Dies sorgt ebenfalls für die Kompensation des Giermoments des Hauptrotors. Damit kann ein Helikopter mittels vier Steuergrößen geregelt werden (siehe auch [36]). Ein Nachteil dieser Systeme ist jedoch der komplizierte mechanische und damit wartungsintensive Aufbau.

Vor allem in dieser mechanischen Komplexität unterscheiden sich die hier im Vordergrund stehenden Multirotorsysteme. Wie der Begriff schon erwarten lässt, werden hier mehrere Antriebe, in der Regel vier oder mehr, verwendet. Die Rotoren haben bei diesen Flugsystemen einen festen Anstellwinkel (*engl. Fixed Pitch*) und sind horizontal zur Plattform ausgerichtet. Eine Vertikalbewegung wird durch eine gleichmäßige Veränderung der Drehzahlen aller Antriebe erzeugt. Durch die damit proportionale Veränderung der Auftriebskraft beginnt das System zu steigen bzw. zu sinken. Soll indes eine Horizontalbewegung ausgeführt werden, muss, analog zum Helikopter, das Flugsystem in eine Richtung verkippt werden. Dies kann um die *Roll*-Achse für Seitwärtsbewegungen oder um die *Pitch*-Achse für Vor- bzw. Rückwärtsbewegungen geschehen und wird durch Variation der Drehzahlen von achsensymmetrisch zueinander liegenden Motoren erreicht. Das Drehmoment jedes einzelnen Antriebs wird kompensiert, indem entgegengesetzt drehende Rotoren zueinander angeordnet sind. So kann die Plattform ebenfalls um die Hochachse manövriert werden, indem die Drehzahl der rechtsdrehenden Roto-

ren gegenüber den linksdrehenden Rotoren verändert wird. Dabei gilt, dass ebenso wie Helikopter auch Multirotorsysteme nicht aerodynamisch eigenstabil sind und daher kontinuierlich nachgeregelt werden müssen. Bevor jedoch in dieser Arbeit mit der abstrakten, mathematischen Systemanalyse begonnen wird, soll in den folgenden zwei Abschnitten ein grundsätzliches Verständnis für das Funktionsprinzip von Multirotorsystemen entwickelt werden. Dabei wird explizit auf den prinzipiellen Aufbau, die Funktionalität, die Steuerung und die Stabilisierung eingegangen.

2.2.1 *Mechanische Konfigurationen*

Multikopter sind in verschiedensten Konfigurationen zu finden. Dabei sind in der Regel vier oder mehr Antriebe integriert, wobei die Rotoren gegenläufig zueinander angeordnet sind (farblich in *grün* und *blau* gekennzeichnet). Eine Besonderheit bildet der Trikopter (siehe Abbildung 7a), bei dem die Drehrichtung aller drei Rotoren gleich ist. Hier wird über einen vierten Antrieb, in der Regel einen Servomotor, die Ausrichtung um die Längsachse des hinteren Rotors verdreht, um so das notwendige Gegendrehmoment zu erzeugen. Die bekannteste und am häufigsten genutzte Multikopterbauart ist der Quadrokopter, welcher als +-, *X*- oder *H*-Konfiguration betrieben werden kann (siehe Abbildung 7b – 7d). In einigen Applikationen wird die *X*-Konfiguration des Quadrokopters um zwei oder vier Antriebe erweitert, um zum Beispiel das maximal mögliche Abfluggewicht zu steigern (siehe Abbildung 7e – Hexakopter oder 7f – Oktokopter).



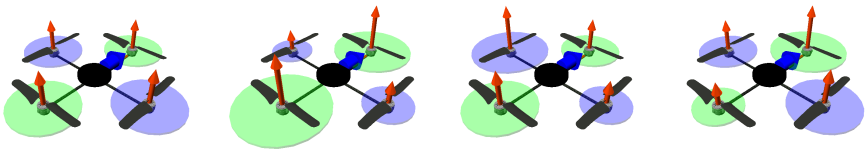
(a) Trikopter *Y*-Konfiguration
(b) Quadrokopter +-Konfiguration
(c) Quadrokopter *X*-Konfiguration
(d) Quadrokopter *H*-Konfiguration
(e) Hexakopter *O*-Konfiguration
(f) Oktokopter *O*-Konfiguration
(g) Oktokopter *V*-Konfiguration
(h) Koaxialkopter *Y*-Konfiguration

Abbildung 7: Übersicht von Multikopterkonfigurationen

Dabei hat ein Oktokopter konzeptionell ein redundant ausgelegtes Rotorsystem, wodurch bei Ausfall eines Antriebs die verbleibenden Antriebe derart geregelt werden können, dass das System weiter betriebsbereit wäre. Bei den zuvor genannten Systemkonfigurationen mit vier bzw. sechs Antriebseinheiten ist dies nur mittels umfangreicher Software- oder Hardwareanpassungen möglich. So wird beispielsweise in [120] ein Verfahren gezeigt, wie ein Quadrokopter bei Ausfall von einem bzw. zwei Rotoren weiterhin in der *Roll-* und *Pitch*-Achse stabilisierbar ist, jedoch mit der Einschränkung, dass das System signifikant um die *Yaw*-Achse rotiert. Weiter wird in [2] ein Verfahren vorgestellt, wie ein Multikopter, im Speziellen ein Hexakopter, bei Ausfall eines Antriebs stabilisiert werden kann, indem ein Rotor entgegen seiner Solldrehrichtung angesteuert wird. Aufbauend auf dem kreisförmigen Oktokopterprinzip wird in [1] eine *V*-Konfiguration (siehe Abbildung 7g) patentiert, welche neben der redundant ausgelegten Antriebskonfiguration ein offenes Sichtfeld für Kameraapplikationen an der Front des Systems bietet. Ebenfalls ein patentiertes Antriebskonzept (siehe [173]) ist in Abbildung 7h dargestellt. Dieser Konfigurationstyp basiert auf drei jeweils koaxial angeordneten Antrieben. Selbstverständlich sind noch eine Vielzahl von anderen Konfigurationsarten denkbar bzw. bekannt. So zeigt [44] eine Kombination aus Trikopter mit einem großen Hauptrotor mit festem Kollektivanstellwinkel. Des Weiteren wird in [25] eine Kombination aus einem Koaxialkopter mit Steuerflächen und einem ganzheitlichen passiven Anschlagschutz vorgestellt. Im weiteren Verlauf dieser Arbeit sollen jedoch vordringlich die Quadro- sowie die Hexakopterkonfigurationen näher betrachtet werden.

2.2.2 *Stabilisierung und Regelung*

Nachdem die unterschiedlichen Konfigurationsmöglichkeiten von Multikoptern vorgestellt worden sind, soll nun das Funktionsprinzip veranschaulicht werden. Hierzu soll exemplarisch ein Quadrokopter in +-Konfiguration genutzt werden und anhand dessen die grundlegenden Verfahren zur Stabilisierung und Regelung aufgezeigt werden. Wie schon in Abschnitt 2.2 angemerkt, zählen Multikopter wie Helikopter zu den nicht aerodynamisch eigenstabilen Flugsystemen und müssen daher kontinuierlich aktiv stabilisiert werden. Dies wird bei Multikoptern durch Veränderung der Drehzahl der einzelnen Antriebe realisiert. Dazu sind in Abbildung 8 vier prinzipielle Systemzustände dargestellt. Zur Vereinfachung wird hier angenommen, dass alle vier Antriebe ein äquivalentes Verhalten haben und bei gleichen Drehzahlen ebenfalls gleiche Kräfte und Momente erzeugen. Demnach kann der Schwebeflug (*Hover* siehe Abbildung 8a) des MAVs durch eine identische Drehzahl aller Antriebe erreicht werden, wobei die Summe aller Kräfte der Einzelantriebe $\vec{F}_j$ gleich dem Produkt



(a) *Hover* (b) *Yaw* (c) *Roll* (d) *Pitch*

Abbildung 8: Prinzipielle Rotordrehzahlen bei unterschiedlichen Flugzuständen

aus Masse der Flugplattform m und der Schwerebeschleunigung $\vec{g}$ ist (siehe Gleichung 2.1). Gleichermaßen kann das Steigen bzw. Sinken der Flugplattform durch kollektives Erhöhen beziehungsweise Senken der Rotordrehzahlen erzeugt werden. Durch die paarweise, gegenläufig angeordneten Antriebe (im Uhrzeigersinn (*engl. Clockwise* (CW)) – in *blau* dargestellt – und gegen den Uhrzeigersinn (*engl. Counterclockwise* (CCW)) – in *grün* dargestellt) gilt, dass sich die Drehmomente aller Einzelantriebe $\vec{M}_j$ im Gleichlauf gegenseitig ausgleichen (siehe Gleichung 2.2). Im idealen *Hover*-Zustand würde demnach gelten:

$$\sum_{j=1}^{4} \vec{F}_j = -m \cdot \vec{g} \tag{2.1}$$

und

$$\sum_{j=1}^{4} \vec{M}_j = 0\,. \tag{2.2}$$

Soll die Flugplattform nun eine gleichförmige Drehbewegung um die Hochachse (*Yaw*) erzeugen, werden die beiden gegenläufigen Rotorpaare entgegengesetzt zueinander in der Drehzahl inkrementiert bzw. dekrementiert, was ein ungleiches Drehmoment und somit die Rotation des Systems bewirkt (siehe Abbildung 8b). In ähnlicher Weise kann das System um seine *Roll*- bzw. *Pitch*-Achse gesteuert werden. Hierzu werden lediglich die Drehzahlen der nicht auf der Rotationsachse angeordneten Rotorpaare zueinander verändert (siehe Abbildung 8c und 8d). Da hierbei jeweils beide Antriebe dieselbe Drehrichtung haben sowie die Drehzahländerung um denselben Betrag auf beide Antriebe entgegengesetzt angewendet wird, gleichen sich die Drehmomente wiederum aus.

Da die zuvor getroffene Vereinfachung der Gleichwertigkeit aller Antriebskomponenten in der Realität nicht zutreffend ist, muss das System kontinuierlich die Kräfte und Momente der einzelnen Antriebe kompensieren, um so eine stabile Fluglage erreichen zu können. Dies wird im Allgemeinen durch einen Regelkreis gelöst, welcher aus

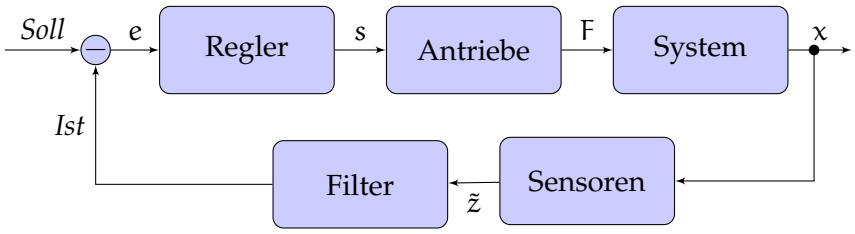


Abbildung 9: Vereinfachtes Blockschema des Regelkreises eines Multirotorsystems

der *Soll*-Wertvorgabe und der Rückkopplung durch integrierte Sensoren (*Ist*-Wert) die Steuergröße (s) für die Drehzahlsteller der einzelnen Motoren errechnet. Ein typischer Regelkreis ist in Abbildung 9 dargestellt. Daraus ableitend ergeben sich drei Grundelemente eines Multirotorsystems: Als erstes sei der Autopilot (AFCS) genannt, bestehend aus verschiedenen Sensoren, Algorithmen zur Datenfilterung und Sensorfusion sowie der eigentlichen Regelung (Regler); hinzu kommen die Antriebe, bestehend jeweils aus Motor, Rotor und Drehzahlsteller, und abschließend das System, welches das physikalische Verhalten widerspiegelt.

Die automatische Stabilisierung umfasst dabei die Regelung der Flugplattform um alle drei Raumachsen und in der Flughöhe. Dieser auf unterster Ebene (*Low-Level*) implementierte Regelkreis dient, neben der indirekten Steuerung der Flugplattform durch den Piloten, ebenfalls den übergeordneten (*High-Level*) Algorithmen zur automatischen Positionshaltung oder der Wegpunktnavigation.

2.3 DIE AUSGANGSPLATTFORM

Wie in Abschnitt 2.1 gezeigt, ist die Auswahl an verfügbaren Multirotorsystemen groß. Neben den zuvor dargestellten Charakteristika

Abbildung 10: Basisplattform AirRobot *AR100B* (*Quelle: ©AirRobot*)

der unterschiedlichen Plattformen ist für dieses Forschungsvorhaben jedoch das maßgebliche Entscheidungskriterium für die Auswahl des Systems die Kooperation mit dem Systemhersteller AirRobot. Durch den damit verbundenen Zugriff auf alle Hard- sowie Softwarekomponenten einschließlich des proprietären Quellcodes, sind die Entwicklungsmöglichkeiten mit diesem System deutlich größer, als es Systeme anderer Hersteller erlauben würden. Mittels dieser Voraussetzung ist es möglich, das in dieser Arbeit angestrebte Vorhaben einer detaillierten, echtzeitfähigen Simulationsumgebung, welche sich besonders durch ein *Software in the Loop* (SiL)-Modul auszeichnet, umzusetzen. Aus diesem Grund bildet der Quadrokopter *AR100B* der Firma AirRobot (siehe Abbildung 10) die Ausgangsplattform. Dieses Multirotor-VTOL-Flugsystem ist mit vier bürsten- und getriebelosen Elektromotoren ausgerüstet. Bei einem Gesamtdurchmesser von 1.0 m und einem Abfluggewicht von 1300 g beträgt die Flugzeit zwischen 22 und 24 Minuten. Dabei können individuelle Nutzlasten von bis zu 250 g vom System transportiert werden. Der integrierte Autopilot übernimmt die automatische Lage- und Positionsstabilisierung bzw. -regelung. Eng gekoppelt mit dem *Flight Management System* (FMS) kann außerdem eine automatische Navigation nach Wegpunkten erfolgen.

3

KOORDINATENSYSTEME UND TRANSFORMATIONEN

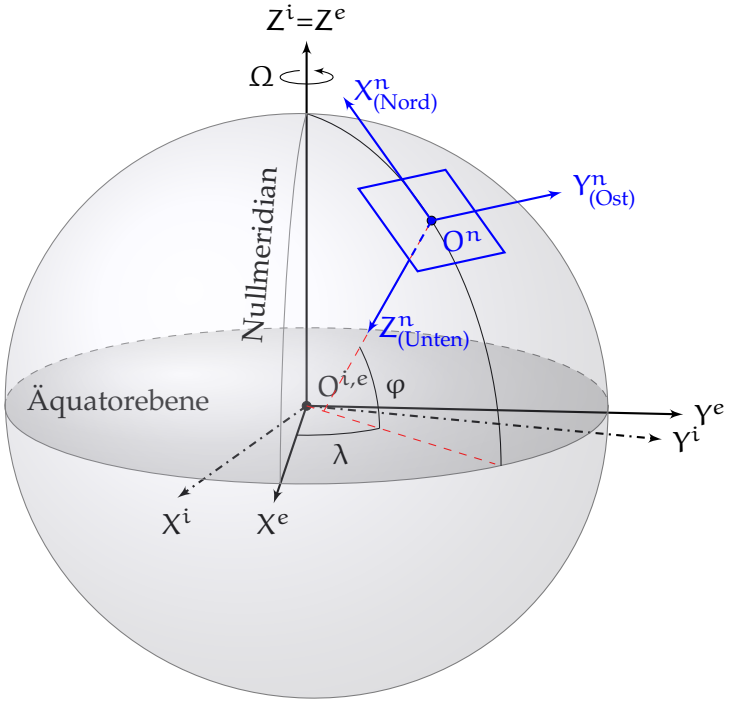
Nachdem in dem vorherigen Kapitel der aktuelle Stand, das grundlegende Funktionsprinzip sowie die Ausgangsplattform dieser Arbeit vorgestellt worden sind, sollen in diesem Kapitel die verwendeten Koordinatensysteme erläutert und die notwendigen Transformationen beschrieben werden.

3.1 KOORDINATENSYSTEME

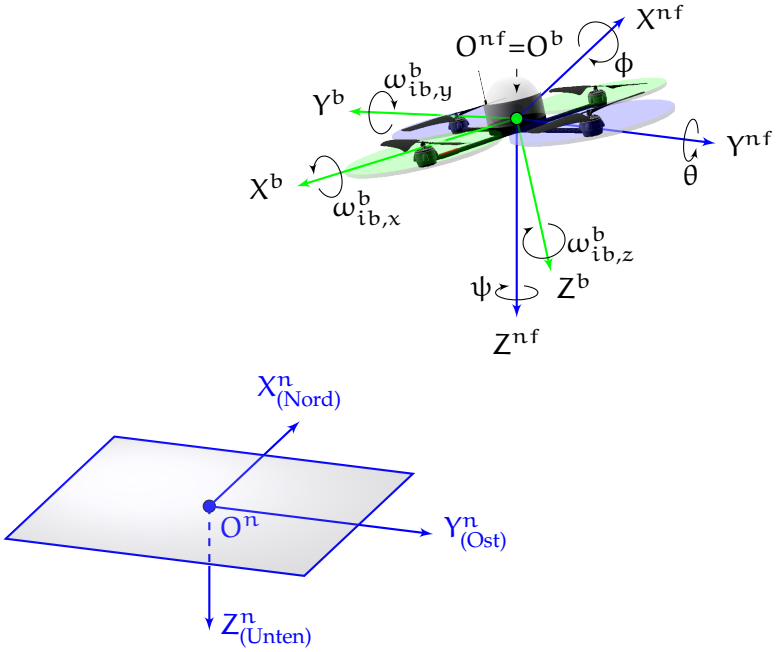
Zur Beschreibung von Position, Geschwindigkeit und Orientierung von Fahrzeugen ist die Definition von Bezugssystemen unerlässlich. Dabei werden in der Regel unterschiedliche Koordinatensysteme genutzt, welche durch festgelegte Bezugspunkte eindeutig beschreibbar sind (siehe [29, 161, 172]). In dieser Arbeit werden daher die nachstehenden vier Koordinatensysteme eingehender betrachtet (siehe Abbildung 11 auf Seite 28):

- das Inertialkoordinatensystem (*i-Frame*),
- das erdfeste Koordinatensystem (*e-Frame*),
- das Navigationskoordinatensystem (*n-Frame*)
- sowie das körperfeste Koordinatensystem (*b-Frame*).

Im Folgenden werden diese Koordinatensysteme beschrieben und grundlegende Zusammenhänge, wie beispielsweise die Transformationen zueinander, vorgestellt. Zur eindeutigen Bezeichnung der Größen wie Beschleunigung, Drehrate, Geschwindigkeit sowie Position innerhalb der unterschiedlichen Koordinatensysteme wird jede Größe mit hochgestellten Indizes versehen. Messwerte, die zum Beispiel bezüglich des körperfesten Koordinatensystems aufgenommen werden, wie etwa die Beschleunigung $\vec{a}^{b}$, werden mit dem Index $\vec{\square}^{b}$ versehen. Eine Besonderheit stellen Systemgrößen dar, welche zusätzlich referenziert werden müssen. Beispielsweise sei hierzu die Drehrate $\vec{\omega}_{ie}^{n}$ genannt. Neben dem hochgestellten Index $\vec{\square}^{n}$, welcher kennzeichnet, dass es sich um eine Drehrate im Navigationskoordinatensystem handelt, zeigen die beiden tiefgestellten Indizes $\vec{\square}_{ie}$, dass es sich um die Drehrate des erdfesten Koordinatensystems bezüglich des Inertialkoordinatensystems handelt. Umgangssprachlich wird diese als die Erddrehrate bezeichnet. Diese Nomenklatur ist aus [161] übernommen.



(a) WGS-, ECEF- und lokales NED-Koordinatensystem



(b) Lokales NED- und köperfestes Koordinatensystem

Abbildung 11: Koordinatensysteme (in Anlehnung an [29])

3.1.1 *Inertialkoordinatensystem*

Beim Inertialkoordinatensystem (*i-Frame*) handelt es sich um ein ruhendes Koordinatensystem, welches starr bezüglich der Fixsterne ist. Es wird durch die drei Koordinatenachsen X^i, Y^i und Z^i definiert, wobei der Koordinatenursprung O^i im Erdmittelpunkt ist. X^i und Y^i liegen hierbei in der Äquatorebene und die Z^i-Achse ist deckungsgleich mit der Polarachse der Erde.

3.1.2 *Erdfestes Koordinatensystem*

Das erdfeste (*e-Frame*) oder auch *Earth-Centered Earth-Fixed* (ECEF)-Koordinatensystem ist, gleichermaßen wie das Inertialkoordinatensystem, ein dreidimensionales, kartesisches Koordinatensystem. Es hat ebenfalls seinen Ursprung O^e im Mittelpunkt der Erde (*Earth-Centered*). Die Koordinatenachsen sind fest mit der Erde gekoppelt und drehen sich mit der Erdrotation Ω (*Earth-Fixed*). Die X^e-Achse verläuft vom Koordinatenursprung durch den Schnittpunkt von Nullmeridian und Äquatorebene. Orthogonal zur X^e-Achse verläuft die Y^e-Achse, ebenfalls in der Äquatorebene. Die Z^e-Achse weist in Richtung des Nordpols und ist koinzident mit der Rotationsachse der Erde. Die Positionsangabe eines beliebigen Punktes kann dabei als Vektor $\vec{p}^e$ in kartesischen Koordinaten erfolgen mit:

$$\vec{p}^e = \begin{pmatrix} p_x^e \\ p_y^e \\ p_z^e \end{pmatrix}. \tag{3.1}$$

Neben der erdfesten Positionsangabe eines Punktes als ECEF-Koordinate ist die Angabe auf Basis eines globalen Referenzsystems gängige Praxis. Vor allem in der satellitengestützten Navigation wird allgemein das *World Geodetic System* (WGS) genutzt. Dieses soll nachfolgend erläutert werden.

World Geodetic System

Das *World Geodetic System* (WGS) ist eines der bekanntesten geodätischen Referenzsysteme zur einheitlichen Positionsangabe auf der Erde. Dieses Referenzsystem basiert auf einem Erdmodell, welches die Erde näherungsweise durch einen Rotationsellipsoiden beschreibt. Der Ellipsoid wird dabei durch folgende vier Parameter definiert:

- große Halbachse (Äquatorradius) R_a
- kleine Halbachse (Polradius) R_b
- Abflachung des Ellipsoiden f
- Exzentrizität e

Analog zum ECEF-Koordinatensystem ist auch der Referenzellipsoid erdfest und hat seinen Mittelpunkt im Massenschwerpunkt der Erde. Das bekannteste WGS-Modell ist das WGS-84, welches erstmals 1984 veröffentlicht wurde und seitdem kontinuierlich präzisiert wird. Darin sind die Parameter des Ellipsoiden wie folgt festgelegt:

$$R_a = 6\,378\,137.0\,\mathrm{m} \tag{3.2}$$

$$R_b = R_a(1-f) = 6\,356\,752.3142\,\mathrm{m} \tag{3.3}$$

$$f = \frac{R_a - R_b}{R_a} = \frac{1}{298.257223563} \tag{3.4}$$

$$e = \sqrt{f(2-f)} = 0.0818191908426\,. \tag{3.5}$$

Die Positionsangabe eines beliebigen Punktes $\vec{p}^{\,wgs}$ kann damit durch die geographische Länge λ, geographische Breite φ und Höhe h bezüglich des Referenzellipsoiden erfolgen (siehe Gleichung 3.6).

$$\vec{p}^{\,wgs} = \begin{pmatrix} \lambda \\ \varphi \\ h \end{pmatrix} \tag{3.6}$$

Dabei gibt der Längengrad λ den Winkel des Punktes bezüglich des Nullmeridians und φ den Winkel zur Äquatorebene an. Die Höhe beschreibt dabei die vertikale Entfernung zum Referenzellipsoiden.

3.1.3 *Navigationskoordinatensystem*

Das Navigationskoordinatensystem (*n-Frame*) ist, wie das ECEF-Koordinatensystem, ein kartesisches Koordinatensystem. Hier ist jedoch zwischen zwei unterschiedlichen Ursprungspunkten zu differenzieren. Beim *lokal* definierten Navigationskoordinatensystem ist dieser ein fester Punkt auf der Erdoberfläche (O^n), genauer auf dem WGS-84-Rotationsellipsoiden (siehe Abbildung 11a auf Seite 28) und bildet in der Regel den Startpunkt der Flugplattform ab. Dem entgegen befindet sich der Ursprungspunkt beim *flugplattformbasierten* Navigationskoordinatensystem im Schwerpunkt der Flugplattform selbst (O^{nf}) (siehe Abbildung 11b). Gleichermaßen gilt für beide Definitionen, dass die X^n- sowie Y^n-Achse parallel zur Tangentialebene des WGS-84-Rotationsellipsoiden verläuft und in Nord- bzw. in Ostrichtung weist. Orthogonal zu dieser Tangentialebene zeigt die Z^n-Achse nach

unten und ist parallel zur Schwerebeschleunigung. Die Positionsangabe eines Punktes im Navigationskoordinatensystem kann somit als Vektor $\vec{p}^{\,n}$ mit den skalaren Komponenten p_x^n, p_y^n und p_z^n erfolgen. Durch den Zusammenhang der Achsenorientierungen wird dieses Koordinatensystem auch als *North-East-Down* (NED)-Koordinatensystem mit den skalaren Vektorkomponenten p_n^n, p_e^n und p_d^n bezeichnet und somit gilt:

$$\vec{p}^{\,n} = \begin{pmatrix} p_x^n \\ p_y^n \\ p_z^n \end{pmatrix} = \begin{pmatrix} p_n^n \\ p_e^n \\ p_d^n \end{pmatrix}. \tag{3.7}$$

Beim flugplattformbasierten Navigationskoordinatensystem ist zu beachten, dass bei einer Bewegung der Flugplattform und damit einer Verschiebung des Navigationskoordinatensystems bezüglich des Erdellipsoiden gilt, dass die X^n- sowie Y^n-Achse kontinuierlich in Nord- und Ostrichtung ausgerichtet werden muss. Die Drehrate des Navigationskoordinatensystems wird in der Literatur als Transportrate bezeichnet (siehe [161]). Da im Falle dieser Flugplattform jedoch nur mit Bewegungen über kurze Distanzen (einige 100 Meter bis wenige Kilometer) und Flugzeiten von < 30 Minuten zu rechnen ist, kann dieser Effekt vernachlässigt werden. Somit gilt, dass die Transformation vom lokalen zum flugplattformbasierten Navigationskoordinatensystem lediglich aus einer Translation besteht, welche durch den Verschiebungsvektor der Koordinatenursprungspunkte beschrieben wird (siehe [29]).

3.1.4 *Körperfestes Koordinatensystem*

Das letzte an dieser Stelle zu beschreibende Koordinatensystem ist das körperfeste Koordinatensystem (*b-Frame*)[1] (siehe Abbildung 11b auf Seite 28). Dieses ist starr mit der Flugplattform gekoppelt und definiert seinen Koordinatenursprung O^b in dessen Schwerpunkt. Die drei Achsen weisen in Plattformlängsrichtung nach vorne (X^b), nach rechts (Y^b) und nach unten (Z^b). Messwerte, die bezüglich des körperfesten Koordinatensystems aufgenommen werden, wie beispielsweise der Beschleunigungsvektor $\vec{a}^{\,b}$, können durch die skalaren Vektorkomponenten a_x^b, a_y^b und a_z^b entsprechend dargestellt werden:

$$\vec{a}^{\,b} = \begin{pmatrix} a_x^b \\ a_y^b \\ a_z^b \end{pmatrix}. \tag{3.8}$$

1 aus dem engl. *(b)* für *Body-Frame*

3.2 KOORDINATENTRANSFORMATIONEN

Da die hier darzustellenden Sensor- und Messsysteme jedoch die Messdaten in unterschiedlichen Koordinatensystemen referenzieren, ist es unabdingbar bezüglich dieser transformieren zu können. Die notwendigen Gleichungen werden im Folgenden erläutert. Dazu werden die Transformationsberechnungen vom lokalen zum globalen Referenzsystem beschrieben.

3.2.1 *Körperfestes und Navigationskoordinatensystem*

Die Umrechnung vom körperfesten Koordinatensystem in das *flugplattformbasierte* Navigationskoordinatensystem beschreibt die Transformation bezüglich zweier kartesischer Koordinatensysteme mit selbem Ursprungspunkt (siehe Abbildung 11b auf Seite 28). Die Orientierung dieser beiden Koordinatensysteme zueinander kann dabei durch drei unabhängige Drehwinkel, die sogenannten Eulerwinkel, beschrieben werden. In der Luftfahrt werden diese Winkel als *Roll* (ϕ), *Pitch* (θ) und *Yaw* bzw. *Heading* (ψ) bezeichnet[2]. Die in der Luftfahrt gültige Drehfolge vom Navigationskoordinatensystem in das körperfeste Koordinatensystem wird definiert durch die drei aufeinanderfolgenden Rotationen (siehe [161]):

- Rotiere mit dem *Yaw*-Winkel (ψ) um die Z^{nf}-Achse
- Rotiere mit dem *Pitch*-Winkel (θ) um die entstandene Y-Achse
- Rotiere mit dem *Roll*-Winkel (ϕ) um die entstandene X-Achse

Durch die Rotationsmatrix oder auch Richtungskosinusmatrix $\mathbf{C}_n^b$ kann somit die Transformation vom Navigationskoordinatensystem in das körperfeste Koordinatensystem beschrieben werden. Beispielhaft soll dazu der Schwerebeschleunigungsvektor $\vec{g}_l^n$, gegeben im Navigationskoordinatensystem

$$\vec{g}_l^n = \begin{pmatrix} 0, & 0, & g_0 \end{pmatrix}^T , \tag{3.9}$$

in das körperfeste Koordinatensystem gedreht werden. Es gilt:

$$\vec{a}^b = -\mathbf{C}_n^b \vec{g}_l^n \ . \tag{3.10}$$

Den entgegengesetzten Fall, vom körperfesten Koordinatensystem in das Navigationskoordinatensystem, beschreibt dabei die transponierte Rotationsmatrix. Da es sich um eine orthogonale Matrix handelt, gilt, dass die transponierte gleich der inversen Matrix ist und somit folgt:

$$\mathbf{C}_b^n = \mathbf{C}_n^{b,T} = \mathbf{C}_n^{b,-1} \ . \tag{3.11}$$

2 Im deutschen Sprachgebrauch können die drei Eulerwinkel auch als Roll-, Nick- und Gier-Winkel *(engl. Roll-, Pitch- and Yaw-/Heading-Angle)* bezeichnet werden.

Die Rotationsmatrix zur Transformation des körperfesten Koordinatensystems in das Navigationskoordinatensystem als Funktion der Eulerwinkel ergibt sich zu[3]:

$$\mathbf{C}_b^n = \begin{pmatrix} c(\theta)c(\psi) & \begin{matrix}-c(\phi)s(\psi)\\ +s(\phi)s(\theta)c(\psi)\end{matrix} & \begin{matrix}s(\phi)s(\psi)\\ +c(\phi)s(\theta)c(\psi)\end{matrix} \\ c(\theta)s(\psi) & \begin{matrix}c(\phi)c(\psi)\\ +s(\phi)s(\theta)s(\psi)\end{matrix} & \begin{matrix}-s(\phi)c(\psi)\\ +c(\phi)s(\theta)s(\psi)\end{matrix} \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{pmatrix} . \tag{3.12}$$

Der Vorteil bei der Darstellung der Rotationsmatrix mittels Eulerwinkeln ist die intuitive Lesbarkeit. Jedoch hat diese Winkelrepräsentation den Nachteil des sogenannten *Gimbal Lock*. Hier tritt bei einem *Pitch*-Winkel von $\pm\frac{\pi}{2}$ eine Singularität auf. Dies führt zu einer Mehrdeutigkeit in der Lösung der Rotationsmatrix. Eine weitere Möglichkeit zur Orientierungsbeschreibung von zwei kartesischen Koordinatensystemen zueinander kann mittels Orientierungsvektor $\vec{\sigma}$ oder durch ein auf die Länge $|\vec{q}| = 1$ normiertes Quaternion $\vec{q}$ formuliert werden. Der Zusammenhang zwischen Orientierungsvektor, Quaternion, Eulerwinkel und Richtungskosinusmatrix soll an dieser Stelle nicht eingehender betrachtet werden. Hierzu sei auf die Abhandlungen in [115, 161] oder [172] verwiesen.

3.2.2 *Navigations- und erdfestes Koordinatensystem*

Die Transformation vom erdfesten Koordinatensystem in das *lokale* Navigationskoordinatensystem und umgekehrt ist besonders wichtig zur Beschreibung der Navigationslösung. Wie in Kapitel 3.1.3 diskutiert gilt dabei, dass zwischen lokalem und flugplattformbasiertem Navigationskoordinatensystem kein Unterschied bezüglich ihrer Ausrichtung, sondern lediglich eine Translation der Ursprungspunkte vorliegt. Im erdfesten Koordinatensystem kann der Ursprungspunkt des lokalen Navigationskoordinatensystems durch $\vec{p}_0^{\,e}$ und des flugplattformbasierten Navigationskoordinatensystems durch $\vec{p}^{\,e}$ dargestellt werden. Der Ursprungspunkt des lokalen Navigationskoordinatensystems entspricht dabei dem Startpunkt (*Take-Off-Point*) der Flugplattform. Somit kann die Position der Flugplattform im Navigationskoordinatensystem $\vec{p}^{\,n}$ relativ zum Startpunkt angegeben werden durch:

$$\vec{p}^{\,n} = \mathbf{C}_e^n(\vec{p}^{\,e} - \vec{p}_0^{\,e}) . \tag{3.13}$$

3 Aus Platzgründen werden die sin() und cos() Funktionen mit s() und c() abgekürzt.

Die Richtungskosinusmatrix $\mathbf{C}_e^n$ gibt darin die Transformation vom ECEF- zum NED-Koordinatensystem an und wird beschrieben durch den Zusammenhang:

$$\mathbf{C}_e^n = \begin{pmatrix} -\sin(\varphi_0)\cos(\lambda_0) & -\sin(\varphi_0)\sin(\lambda_0) & \cos(\varphi_0) \\ -\sin(\lambda_0) & \cos(\lambda_0) & 0 \\ -\cos(\varphi_0)\cos(\lambda_0) & -\cos(\varphi_0)\sin(\lambda_0) & -\sin(\varphi_0) \end{pmatrix} . \quad (3.14)$$

Dabei beschreibt λ_0 den Längengrad und φ_0 den Breitengrad des Ursprungspunktes $\vec{p}_0^e$.

3.2.3 *Erdfestes Koordinatensystem und WGS-84*

Die letzte, in diesem Zusammenhang zu beschreibende Transformation leitet sich aus dem Kontext, dass GNSS-Empfänger in der Regel eine Positionsangabe im WGS-84-Format (siehe Gleichung 3.6) ausgeben, ab. Wie im vorherigen Abschnitt gezeigt, werden jedoch für die Navigationslösung Positionsangaben im kartesischen ECEF-Koordinatensystem benötigt. Daraus folgt die Umrechnung von WGS-84-Positionsangaben in das ECEF-Koordinatensystem durch:

$$p_x^e = (N + h)\cos(\varphi)\cos(\lambda) \quad (3.15)$$

$$p_y^e = (N + h)\cos(\varphi)\sin(\lambda) \quad (3.16)$$

$$p_z^e = (N(1 - e^2) + h)\sin(\varphi) \, . \quad (3.17)$$

Hierbei beschreibt λ die geographische Länge, φ die geographische Breite und h die Höhe im WGS-84-Format. Der Parameter N ist der Krümmungsradius des ersten Vertikals, welcher sich berechnet zu:

$$N = \frac{R_a}{\sqrt{1 - e^2\sin(\varphi)^2}} \, . \quad (3.18)$$

Dieser kann als Funktion des Breitengrads φ, dem Äquatorradius R_a (siehe Gleichung 3.2) und der Exzentrizität e (siehe Gleichung 3.5) berechnet werden.

4

SENSOREN UND MESSSYSTEME

Nachdem im vorherigen Kapitel die notwendigen Koordinatensysteme und Transformationen erläutert worden sind, werden in diesem Kapitel die genutzten Sensoren vorgestellt. Einleitend ist hierzu in Abbildung 12 der Zusammenhang der unterschiedlichen Sensoren mit ihren spezifischen Koordinatensystemen dargestellt. Daraus ist ersichtlich, dass ein „*State of the Art*"-Sensorsystem von MAVs grundlegend aus fünf unterschiedlichen Sensorgruppen besteht. Im Kontext der zuvor erläuterten Koordinatensysteme wird ebenfalls der Aspekt der Koordinatentransformation mit den spezifischen Nomenklaturen in dieser Abbildung aufgegriffen.

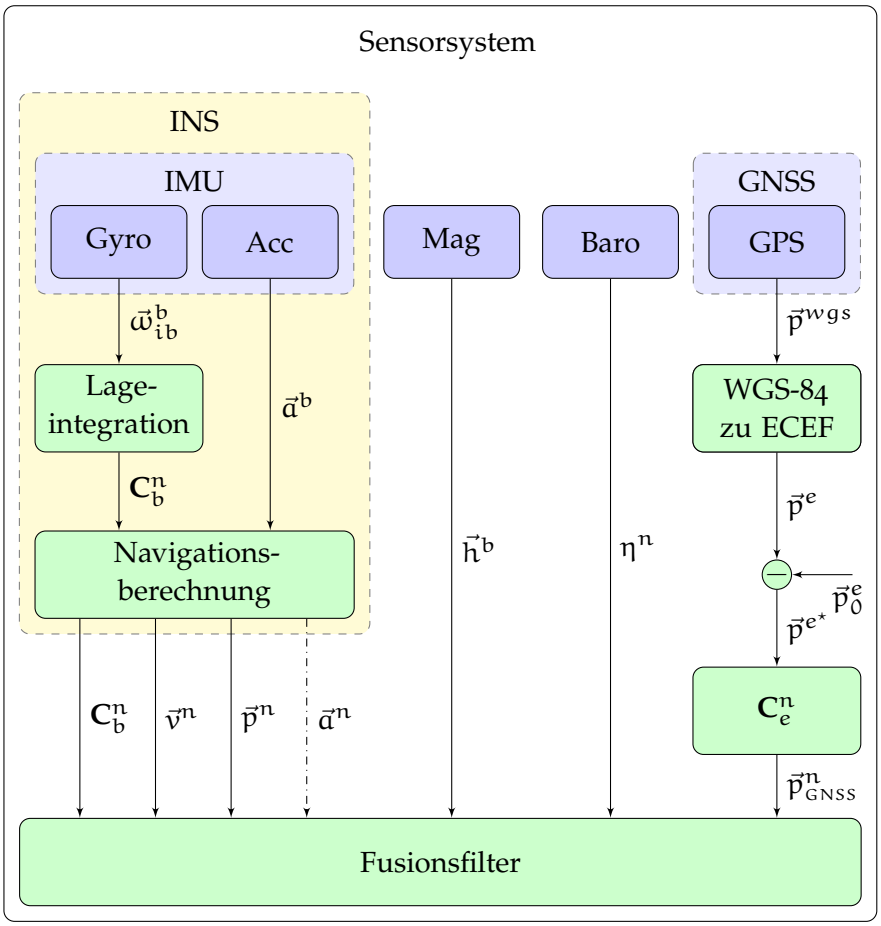


Abbildung 12: Prinzip eines Sensorsystems für MAVs

4.1 INERTIAL-MESSSYSTEM

4.1.1 *Funktionsweise*

Bei einem Trägheitsnavigationssystem (*engl. Inertial Navigation System* (INS)) handelt es sich um ein Sensorsystem, welches die Lage und die Position eines Körpers im Raum bestimmt. Zur vollständigen Beschreibung solcher Bewegungen bedarf es der Messung von sechs kinematischen Freiheitsgraden, die sich in jeweils drei translatorische und drei rotatorische gliedern (siehe auch [177]). Diese sechs Freiheitsgrade werden typischerweise durch ein Inertial-Messsystem (*engl. Inertial Measurement Unit* (IMU)) erfasst, welches aus jeweils drei orthogonal zueinander angeordneten Beschleunigungssensoren und Gyroskopen aufgebaut ist. Die IMU ist dabei auf der Trägerplattform, d. h. auf dem MAV im Drehpunkt, montiert und erfasst die jeweiligen Beschleunigungen ($\vec{a}^{b}$) und Drehraten ($\vec{\omega}_{ib}^{b}$) im körperfesten Koordinatensystem. Das übergeordnete INS bestimmt aus den aufgenommenen Messwerten fortlaufend die Position ($\vec{p}^{n}$), Geschwindigkeit ($\vec{v}^{n}$) und Lage ($\mathbf{C}_{b}^{n}$) der Flugplattform (siehe *gelber* Kasten in Abbildung 12 auf Seite 35).

4.1.2 *Strapdown-Algorithmus*

Der *Strapdown*-Algorithmus (siehe Abbildung 13) dient als Berechnungsvorschrift des Trägheitsnavigationssystems. Hierbei werden anhand der aktuellen Messwerte aus der Inertialsensorik die Navigationsgrößen relativ zu den, als bekannt vorausgesetzten Startparametern, wie Startpunkt ($\vec{p}_{0}^{n}$), Lage ($\mathbf{C}_{b,0}^{n}$) und Geschwindigkeit ($\vec{v}_{0}^{n}$), propagiert. Hierbei gliedert sich die *Strapdown*-Berechnung in drei gesonderte Integrationsschritte:

- Propagation der Lage durch Integration der Drehrate
- Propagation der Geschwindigkeit durch Integration der Beschleunigung
- Propagation der Position durch Integration der Geschwindigkeit

Der *Strapdown*-Algorithmus bezieht sich dabei auf ein ruhendes, inertiales Navigationssystem. Im Fall der Erde handelt es sich jedoch nicht um ein derartiges System, weshalb im Allgemeinen auch die folgenden Terme berücksichtigt werden müssen: Erddrehrate, Transportrate, Coriolisbeschleunigung sowie ein Modell der Erdgravitation (siehe Abbildung 13 *grauer* Kasten).

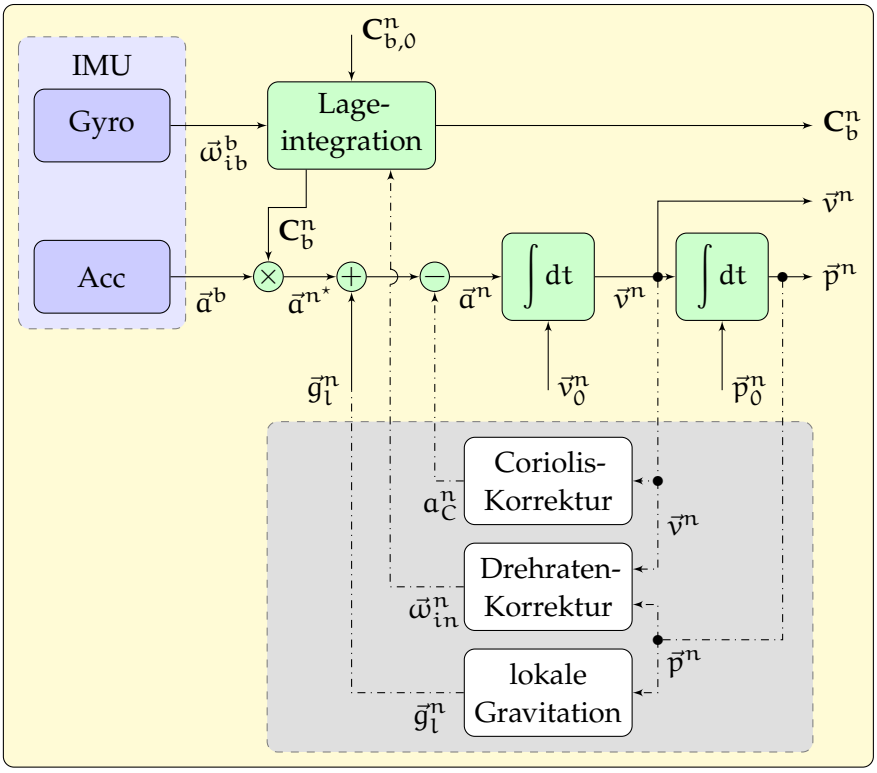


Abbildung 13: *Strapdown*-Algorithmus Blockdiagramm

Propagation der Lage

Im ersten Schritt der *Strapdown*-Berechnung wird die Lage bestimmt. Hierzu ist die 3×3-Rotationsmatrix $\mathbf{C}_b^n$ zu berechnen. Dies geschieht durch Integration der gemessenen Drehraten der Gyroskope. Dabei ist jedoch zu beachten, dass sich die gemessene Drehrate bezüglich des körperfesten Koordinatensystems ($\vec{\omega}_{ib}^b$) aus drei Einzeltermen zusammensetzt. Neben der eigentlichen Drehrate des Systems ($\vec{\omega}_{nb}^b$) wirken ebenfalls noch die Erddrehrate ($\vec{\omega}_{ie}^n$) und die Transportrate ($\vec{\omega}_{en}^n$), wie in Gleichung 4.1 dargestellt:

$$\vec{\omega}_{ib}^b = \vec{\omega}_{nb}^b + \mathbf{C}_n^b \underbrace{\left[\vec{\omega}_{ie}^n + \vec{\omega}_{en}^n\right]}_{\vec{\omega}_{in}^n}. \tag{4.1}$$

Fasst man im Folgenden die Terme Erddrehrate ($\vec{\omega}_{ie}^n$) und Transportrate ($\vec{\omega}_{en}^n$) zu $\vec{\omega}_{in}^n$ zusammen und formt es zur gesuchten Größe, der eigentlichen Drehrate des Systems ($\vec{\omega}_{nb}^b$) um, ergibt sich:

$$\vec{\omega}_{nb}^b = \vec{\omega}_{ib}^b - \mathbf{C}_n^b \vec{\omega}_{in}^n \,. \tag{4.2}$$

Bedingt durch die geringe Güte der in dieser Arbeit genutzten MEMS-Gyroskope können sowohl die Erddrehrate als auch die Transportrate vernachlässigt werden, womit sich die Gleichung 4.2 vereinfacht zu:

$$\vec{\omega}_{nb}^b \approx \vec{\omega}_{ib}^b \,. \tag{4.3}$$

An dieser Stelle wird ausgenutzt, dass die Lage der Flugplattform äquivalent zur Rotationsmatrix ($\mathbf{C}_b^n$) ebenfalls als Orientierungsvektor ($\vec{\sigma}$) ausgedrückt werden kann. Wie in [161] dargestellt, kann gezeigt werden, dass die Änderung des Orientierungsvektors ($\dot{\vec{\sigma}}$) in Abhängigkeit von der Drehrate ($\vec{\omega}_{nb}^b$) durch die Bortzsche Orientierungsvektordifferentialgleichung[1] berechnet werden kann zu:

$$\begin{aligned}\dot{\vec{\sigma}} = \vec{\omega}_{nb}^b + \frac{1}{2}\vec{\sigma} \times \vec{\omega}_{nb}^b \\ + \frac{1}{|\vec{\sigma}|^2}\left(1 - \frac{|\vec{\sigma}|\sin(|\vec{\sigma}|)}{2(1-\cos(|\vec{\sigma}|)}\right)\vec{\sigma} \times \left(\vec{\sigma} \times \vec{\omega}_{nb}^b\right).\end{aligned} \tag{4.4}$$

Wie [161] und [172] zeigen, können die trigonometrischen Funktionen aus Gleichung 4.4 näherungsweise mittels einer Reihenentwicklung numerisch gelöst werden. Vernachlässigt man dabei alle höheren Terme ab der 3. Ordnung, dann ergibt sich

$$\dot{\vec{\sigma}} = \vec{\omega}_{nb}^b + \frac{1}{2}\vec{\sigma} \times \vec{\omega}_{nb}^b + \frac{1}{12}\vec{\sigma} \times \left(\vec{\sigma} \times \vec{\omega}_{nb}^b\right). \tag{4.5}$$

Auf der Grundlage, dass der letzte Term von Gleichung 4.5 nur einen sehr kleinen Einfluss auf die Lösung hat, kann dieser, um den Berechnungsaufwand weiter zu reduzieren, ebenfalls vernachlässigt werden. Unter der Voraussetzung von hohen Updateraten der Inertialsensorik zeigen [115] sowie [172], dass die Lageänderung $\Delta\vec{\sigma}_k$ im Zeitintervall von t_{k-1} bis t_k durch den Zusammenhang

$$\Delta\vec{\sigma}_k = \int_{t_{k-1}}^{t_k} \vec{\omega}_{nb}^b \, dt \tag{4.6}$$

genähert werden kann. Beschreibt man diesen Zusammenhang mit der Abtastzeit Δt der IMU, ergibt sich für die Lageänderung

$$\Delta\vec{\sigma}_k = \vec{\omega}_{nb,k}^b \Delta t \, . \tag{4.7}$$

Mit der Näherung aus Gleichung 4.3 folgt:

$$\Delta\vec{\sigma}_k \approx \vec{\omega}_{ib,k}^b \Delta t \, . \tag{4.8}$$

Somit kann die Lage der Flugplattform $\mathbf{C}_{b,k}^n$ zum Zeitpunkt t_k durch die bekannte vorherige Orientierung $\mathbf{C}_{b,k-1}^n$ und mit Hilfe des berechneten Lageinkrementes $\Delta\vec{\sigma}_k$ bestimmt werden.

Propagation der Geschwindigkeit

Aufbauend auf der Bestimmung der Lage der Flugplattform kann im Folgenden die Geschwindigkeit bestimmt werden. Hierzu ist zuerst

1 Eine ausführliche Darstellung und die Herleitung der Gleichung ist in [19] zu finden.

die gemessene Beschleunigung ($\vec{a}^b$) mittels der Rotationsmatrix ($\mathbf{C}_b^n$) aus dem körperfesten in das Navigationskoordinatensystem zu transformieren:

$$\vec{a}^{n^\star} = \mathbf{C}_b^n \vec{a}^b. \tag{4.9}$$

Die transformierte Beschleunigung wird vorerst noch mit dem Index $\vec{\square}^{n^\star}$ versehen, da zur Berechnung der Geschwindigkeit zuvor zwei weitere Terme betrachtet werden müssen. Diese sind die Coriolisbeschleunigung (a_C^n) und die lokale Erdgravitation ($\vec{g}_l^n$). Die Coriolisbeschleunigung ergibt sich aus der Summe der Erddrehrate ($\vec{\omega}_{ie}^n$) und der Transportrate ($\vec{\omega}_{en}^n$) sowie dem Kreuzprodukt der Geschwindigkeit der Flugplattform ($\vec{v}^n$) zu:

$$a_C^n = \left(2\vec{\omega}_{ie}^n + \vec{\omega}_{en}^n\right) \times \vec{v}^n. \tag{4.10}$$

Anhand der aktuellen Position des Systems ($\vec{p}^n$) kann über ein Gravitationsmodell der Erde die lokale Erdgravitation ($\vec{g}_l^n$) bestimmt werden. Zusammengefasst ergibt sich entsprechend für die Beschleunigung im Navigationskoordinatensystem $\vec{a}^n$:

$$\vec{a}^n = \mathbf{C}_b^n \vec{a}^b - \left(2\vec{\omega}_{ie}^n + \vec{\omega}_{en}^n\right) \times \vec{v}^n + \vec{g}_l^n \ . \tag{4.11}$$

Wird daraufhin die daraus resultierende Beschleunigung im Zeitintervall t_{k-1} bis t_k integriert, erhält man die Geschwindigkeitsänderung gemäß:

$$\Delta\vec{v}_k^n = \int_{t_{k-1}}^{t_k} \vec{a}^{n^\star}\,dt - \int_{t_{k-1}}^{t_k} \underbrace{\left(2\vec{\omega}_{ie}^n + \vec{\omega}_{en}^n\right) \times \vec{v}^n\,dt}_{\approx 0} + \int_{t_{k-1}}^{t_k} \vec{g}_l^n\,dt \ . \tag{4.12}$$

Wie im vorherigen Abschnitt bereits erläutert, können, bedingt durch die geringe Güte der genutzten MEMS-Sensoren, verschiedene Vereinfachungen getroffen werden. In diesem Fall ist die Coriolisbeschleunigung (Kreuzproduktterm) zu vernachlässigen, da diese nicht von den Messfehlern des Beschleunigungssensors zu unterscheiden ist (siehe Gleichung 4.12). Des Weiteren sind die in dieser Arbeit behandelten Flugplattformen auf einen festgelegten Einsatzradius von maximal 1000 m ausgelegt, wodurch die Annahme getroffen werden kann, dass es sich in diesem Bereich um ein konstantes Erdgravitationsfeld handelt. Für Arnsberg, dem Ort, an dem diese Arbeit entstanden ist, folgt nach [135], dass $|\vec{g}_l^n| = 9.8114\,\mathrm{m/s^2}$ ist. Mit diesen gültigen Vereinfachungen kann die Geschwindigkeitsänderung durch den zeitdiskret abgetasteten Beschleunigungssensor mit der Abtastzeit Δt berechnet werden zu:

$$\Delta\vec{v}_k^n = \left(\mathbf{C}_{b,k}^n \vec{a}_k^b + \vec{g}_l^n\right) \Delta t \ . \tag{4.13}$$

Die gesuchte Geschwindigkeit ergibt sich nach Gleichung 4.13 zu:

$$\vec{v}_k^n = \vec{v}_{k-1}^n + \underbrace{\left(\vec{a}_k^{n^\star} + \begin{pmatrix} 0 \\ 0 \\ g_0 \end{pmatrix} \right)}_{\vec{a}_k^n} \Delta t \, . \tag{4.14}$$

Propagation der Position

Im letzten Schritt des *Strapdown*-Algorithmus wird die Positionsänderung bestimmt. Hierzu wird lediglich die Geschwindigkeit im Zeitintervall von t_{k-1} bis t_k integriert:

$$\Delta\vec{p}_k^n = \int_{t_{k-1}}^{t_k} \vec{v}^n \, dt \, . \tag{4.15}$$

Die Position kann demnach als Funktion der Abtastzeit Δt mittels der Trapezregel numerisch integriert werden:

$$\vec{p}_k^n = \vec{p}_{k-1}^n + \left(\frac{\vec{v}_{k-1}^n + \vec{v}_k^n}{2} \right) \Delta t \, . \tag{4.16}$$

Ausgedrückt mit der Beschleunigung $\vec{a}_k^n$, kann Gleichung 4.16 umgeformt werden zu:

$$\vec{p}_k^n = \vec{p}_{k-1}^n + \vec{v}_{k-1}^n \Delta t + \frac{1}{2} \vec{a}_k^n \Delta t^2 . \tag{4.17}$$

4.1.3 *Zusammenfassung und Diskussion*

Die Nutzung einer IMU in Verbindung mit dem *Strapdown*-Algorithmus hat spezifische Vorteile. Besonders die Eigenschaft, dass die vollständige Navigationslösung aus Positions-, Geschwindigkeits- und Lagebestimmung durch die integrierte und von äußeren Faktoren unabhängige Inertialsensorik möglich ist, zeichnet dieses Messsystem aus. Hierbei ist es jedoch erforderlich, dass die Initialbedingungen für die Position ($\vec{p}_0^n$), die Lage ($\mathbf{C}_{b,0}^n$) und die Geschwindigkeit ($\vec{v}_0^n$) bekannt sind. Aufgrund dessen, dass die aktuellen Navigationsparameter aus der letzten berechneten Lösung und den aktuellen Messwerten der Gyroskope und Beschleunigungssensoren bestimmt werden, bezeichnet man den *Strapdown*-Algorithmus ebenfalls als ein rekursives Berechnungsverfahren. Dieses Verfahren kann daher ohne weitere Stützinformationen auch nur über kurze Zeit eine genaue Navigationslösung bestimmen, da die inhärenten Sensorfehler im zeitlichen Verlauf zu steigenden Fehlern bezüglich Position, Geschwindigkeit und Lage führen. Aus diesem Grund werden im Folgenden weitere Sensorsysteme betrachtet, welche zur Stützung der Inertialsensorik genutzt werden können.

4.2 MAGNETOMETER

Eines der ältesten, wetterunabhängigen Navigationsinstrumente stellt der Magnetkompass dar, welcher die Himmelsrichtung bzw. den Navigationskurs durch Messung des Erdmagnetfeldes bestimmt. Der genaue Ursprung kann nicht mit Sicherheit rekonstruiert werden. Es ist jedoch bekannt, dass in China schon im 11. Jahrhundert eine frühe Form des heutigen Magnetkompasses zur Navigation entwickelt worden ist. Etwa ein Jahrhundert später wurde dieses Prinzip ebenfalls im Mittelmeerraum und im weiteren Europa zur Navigation in der Seefahrt eingesetzt und weiterentwickelt (siehe [62, 122]). Hier ist vor allem der Aufbau des Kompasses grundlegend verändert worden, wodurch seine Genauigkeit deutlich gesteigert werden konnte. Gegen Ende des 15. Jahrhunderts entwarf *Leonardo da Vinci* einen kardanisch gelagerten Kompass, um zusätzlich Neigungseinflüsse, bedingt durch die Schiffsbewegung, zu kompensieren. Erst etwa vier Jahrhunderte später entwickelte *Carl Friedrich Gauß* 1832 eine Methode zur Messung der absoluten Intensität des Magnetfeldes und präsentierte damit das erste Magnetometer (siehe [71]). Die bis dahin äußerst komplexen mechanischen Aufbauten können heute durch hochintegrierte elektronische Sensoren abgebildet werden, welche nachfolgend näher betrachtet werden.

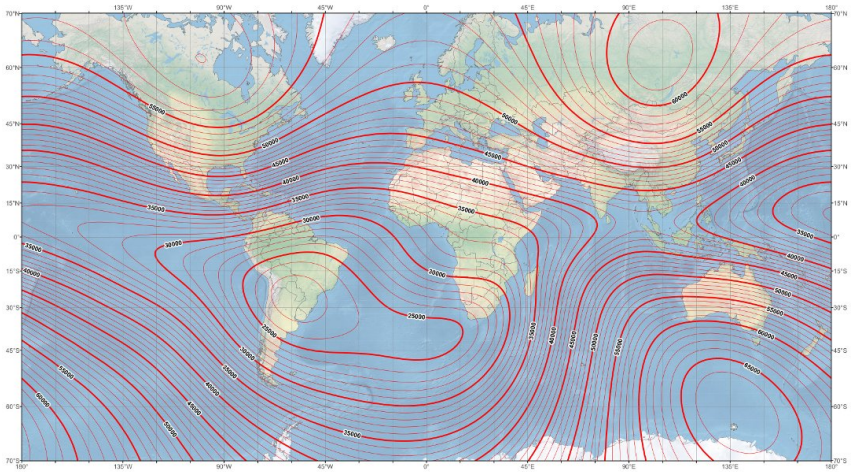
4.2.1 *Funktionsweise*

Der elektronische Magnetfeldkompass wird zur Stützung des INS, insbesondere des *Yaw*-Winkels (ψ), genutzt. Prinzipiell erfasst dieses Sensorsystem über die orthogonal zueinander angeordneten Sensorachsen $(h_x, h_y, h_z)^T$ die magnetische Flussdichte des Erdmagnetfeldes. An dieser Stelle wird jedoch darauf verzichtet, die physikalischen Eigenschaften und den Aufbau des Erdmagnetfeldes herzuleiten, und stattdessen auf [144] verwiesen, wo eine sehr detaillierte Beschreibung zu finden ist.

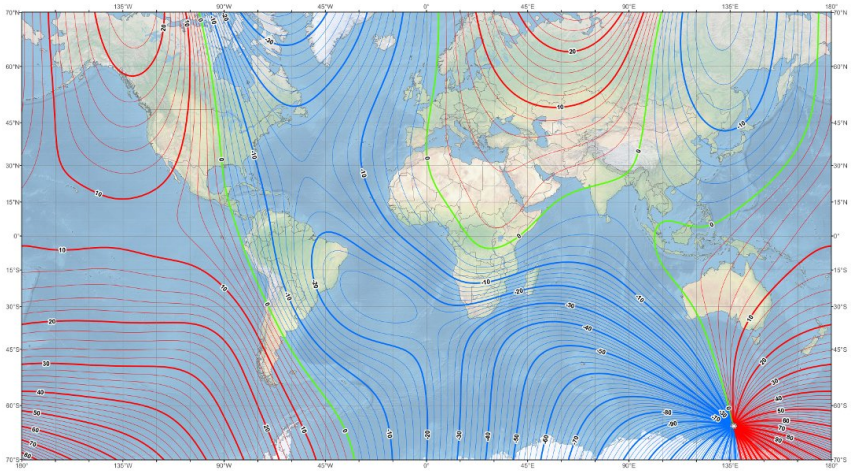
Im hier betrachteten Fall ist das Magnetometer fest mit der Flugplattform verbunden und erfasst somit die Stärke des Magnetfeldes bezüglich des körperfesten Koordinatensystems ($\vec{h}^b$) (siehe auf Seite 35 Abbildung 12). Über die Richtungskosinusmatrix ($\mathbf{C}_b^n$) kann der Vektor des Erdmagnetfeldes im Navigationskoordinatensystem ($\vec{h}^n$) bestimmt werden durch:

$$\begin{pmatrix} h_n^n \\ h_e^n \\ h_d^n \end{pmatrix} = \mathbf{C}_b^n \begin{pmatrix} h_x^b \\ h_y^b \\ h_z^b \end{pmatrix}. \tag{4.18}$$

Dieser Vektor kann ebenfalls durch ein Modell des Erdmagnetfeldes bestimmt werden, sofern die aktuelle Position (geographische Länge λ, geographische Breite φ und Höhe h) bekannt ist. Ein verbrei-



(a) Absolute Intensitäten des Erdmagnetfeldes *(Quelle: [123])*



(b) Deklinationsübersicht *(Quelle: [123])*

Abbildung 14: Erdmagnetfeldintensität und Deklination nach WMM 2015

tetes Modell zur Berechnung der Stärke des Erdmagnetfeldes ist das *World Magnetic Model* (WMM)[2], welches in Kooperation des US-amerikanischen *National Geophysical Data Center* (NGDC) und dem *British Geological Survey* (BGS) entwickelt worden ist. Die Parameter des Modells müssen jedoch in regelmäßigen Abständen aktualisiert werden, da sich das Magnetfeld im Laufe der Zeit verändert. Eine Übersicht der aktuellen geographischen Verteilung der magnetischen Intensität zeigt Abbildung 14a, welche auf Basis des WMM von 2015 abgeleitet ist. Es ist ersichtlich, dass die absolute Intensität des Erdmagnetfeldes $|\vec{h}^n|$ zwischen 22.0 µT am Äquator und 67.0 µT z. B. am Südpol differiert. Beispielsweise ergibt sich nach diesem Modell für Arnsberg ein

2 Neben dem WMM kann auch das *International Geomagnetic Reference Field* (IGRF) genutzt werden (siehe [71, 115, 123]).

Magnetfeld mit der absoluten Intensität von $|\vec{h}^n| = 49.0089\,\mu T$ mit dem Vektor

$$\begin{pmatrix} h_n^n, & h_e^n, & h_d^n \end{pmatrix}^T = \begin{pmatrix} 19.3121, & 0.6515, & 45.0388 \end{pmatrix}^T \mu T\,. \quad (4.19)$$

Es ist jedoch zu beachten, dass das Magnetometer lediglich die Richtung und Intensität der örtlichen Magnetfeldlinien bestimmt. Der Azimutwinkel (*Yaw* oder *Heading*) bezüglich des geographischen Nordpols kann im Regelfall bis zu $\pm 25°$ von der bestimmten magnetischen Nordrichtung abweichen. Eine Ausnahme bilden die Pole selbst, an denen der Fehler noch deutlich größer ist, wie in Abbildung 14b gezeigt wird. Diese Abweichung wird als Deklination bezeichnet und kann in Abhängigkeit von der geographischen Position ebenfalls anhand des WMM berechnet werden. Für eine eingehendere Betrachtung dieser Thematik sei auf [32] und [71] verwiesen.

4.2.2 *Zusammenfassung und Diskussion*

Das Magnetometer stellt durch die Möglichkeit der absoluten Bestimmung der Nordrichtung eine sinnvolle und gewinnbringende Erweiterung des INS dar. Zur Stützung mittels Magnetometer muss jedoch unterschiedlichen Einflüssen und Fehlern Rechnung getragen werden. Neben der beschriebenen geographischen Missweisung (*Deklination*) müssen auch systematische Fehler, wie ferromagnetische Einflüsse und elektrische Störfelder, beachtet werden. Hängt diese zusätzliche Abweichung (*Deviation*) von Einflüssen der Flugplattform ab, können diese in der Regel durch Kalibrierverfahren kompensiert werden. Die Herleitung eines solchen Kalibrierverfahrens ist im Anhang B.3 ab Seite 228 dargestellt.

4.3 BAROMETRISCHE HÖHENBESTIMMUNG

Grundlegend dient das Barometer der Erfassung des statischen Absolut-Luftdrucks und wurde Mitte des 17. Jahrhunderts von *Evangelista Torricelli*[3] in Form des Quecksilberbarometers (*auch Torricellisches Rohr*) erfunden (siehe [24]). Im Allgemeinen dient das Barometer meteorologischen Zwecken. Jedoch kann aufgrund des gemessenen Luftdrucks ebenfalls auf die Höhe bezüglich eines Referenzpunktes geschlossen werden.

4.3.1 *Funktionsweise*

Die barometrische Höhenformel beschreibt dabei den grundsätzlichen Zusammenhang zwischen Luftdruck und Höhe. Da jedoch der

3 Nach *Evangelista Torricelli* ist ebenfalls die Maßeinheit für Druck *Torr* ≈ *Millimeter-Quecksilbersäule* [Torr, mmHg] benannt.

Temperatur auf Meereshöhe	T_0	$= 288.15\,\mathrm{K} \mathrel{\widehat{=}} 15\,^\circ\mathrm{C}$
Luftdruck auf Meereshöhe	p_0	$= 1013.25\,\mathrm{hPa}$
Luftdichte auf Meereshöhe	ρ_0	$= 1.2250\,\mathrm{kg/m^3}$
Schwerebeschleunigung	g_0	$= 9.80665\,\mathrm{m/s^2}$
Universelle Gaskonstante	R	$= 8.314\,\mathrm{J/(mol\,K)}$
Mittlere molare Masse	M	$= 0.02896\,\mathrm{kg/mol}$

Tabelle 2: Parameter für die ISA

gemessene Luftdruck neben der gesuchten Höhe ebenfalls von Parametern wie Temperatur, Luftdichte oder Luftfeuchtigkeit abhängig ist, wird bei der internationalen Höhenformel der Bezugspunkt auf Meereshöhe mit einheitlichen Parametern für die dort geltende Atmosphäre definiert. Diese international gültige Standardatmosphäre (*engl. International Standard Atmosphere* (ISA)) ist von der *International Civil Aviation Organization* (ICAO) im Dokument 7488/2, Second Edition, 1964 bzw. in [80] festgelegt. In Tabelle 2 sind die dort definierten Größen aufgelistet. Neben der ISA sind weitere nationale sowie internationale Normungen verfügbar. Diese unterscheiden sich jedoch erst ab einer Höhe von 32 km über Meereshöhe und sind daher für den hier untersuchten Fall nicht weiter relevant (siehe [115]).

Entsprechend der ISA gilt, dass die relative Luftfeuchte 0 % beträgt, d. h. die Luft wird als absolut trockenes Gas angenommen. Zudem wird der Temperaturgradient a in dem hier relevanten Bereich der Troposphäre (bis 11 000 m über Meereshöhe) mit $a = -0.0065\,\mathrm{K/m}$ angegeben. Hiervon ausgehend kann der barometrische Luftdruck $p(h)$ durch die in Tabelle 2 eingeführten Terme in Abhängigkeit von der Höhe (h) über Meereshöhe formuliert werden zu:

$$p(h) = p_0 \cdot \left(1 - \frac{a \cdot h}{T_0}\right)^{\frac{M \cdot g_0}{R \cdot a}} . \tag{4.20}$$

Wird Gleichung 4.20 zum gesuchten Term $h(p)$ umformuliert, ergibt sich:

$$h(p) = \frac{T_0}{a} \cdot \left(1 - \frac{p}{p_0}^{\frac{R \cdot a}{M \cdot g_0}}\right) . \tag{4.21}$$

Unter Berücksichtigung des in dieser Applikation geplanten eingeschränkten und erdnahen Flughöhenprofils zeigt [115], dass das isentrope/polytrope Atmosphärenmodell durch das isotherme Modell bis zu einer maximalen Flughöhe[4] von $\approx 1000\,\mathrm{m}$ über Meereshöhe approximiert werden kann. Somit lässt sich der formale Zusammenhang aus Gleichung 4.21 vereinfachen zu:

$$h(p) = -\frac{p_0}{\rho_0 \cdot g_0} \cdot \ln\left(\frac{p}{p_0}\right) . \tag{4.22}$$

4 In der Luftfahrt wird diese auch als Dienstgipfelhöhe bezeichnet.

Da jedoch das Navigationskoordinatensystem als *North-East-Down* definiert ist, gilt der Zusammenhang zwischen der Höhe in Abhängigkeit des Luftdrucks $h(p)$ zu der in Abbildung 12 auf Seite 35 formal eingeführten barometrischen Höhe[5] im Navigationskoordinatensystem η^n entsprechend:

$$\eta^n = -h(p) \, . \tag{4.23}$$

4.3.2 *Zusammenfassung und Diskussion*

Die Auswertung des Luftdrucks mittels Barometer stellt eine unabhängige Methode zur absoluten Höhenbestimmung dar. Jedoch kann eine Stützung des INS nur unter festen Rahmenbedingungen zielführend sein. So kann es aufgrund von Wettereinflüssen (Hoch- bzw. Tiefdruck) zu deutlichen Druckabweichungen und damit zur fehlerhaften Höhenbestimmung kommen[6]. Zur Kompensation dieser Abweichungen empfiehlt sich eine relative Höhenbestimmung, beispielsweise zum Startpunkt. Zudem ist die Positionierung in der Flugplattform zu bedenken, um Fehlmessungen, bedingt durch die Luftströmung der Rotoren, zu vermeiden. Nicht zuletzt ergeben sich durch den exponentiellen Zusammenhang zwischen Luftdruck und Höhe nennenswerte Anforderungen an das Messinstrument. So kann leicht über Gleichung 4.22 gezeigt werden, dass bei einer Höhenauflösung des Barometers von 1 cm das Messinstrument eine Auflösung von mindestens 0.0012 hPa, was ≈ 20 bit entspricht, benötigt. Im weiteren Verlauf dieser Arbeit wird das Sensorsystem zur barometrischen Höhenbestimmung, wie in der gängigen Fachliteratur üblich, als *Barometric-Altimeter*, kurz *Baro-Altimeter*, bezeichnet.

4.4 GLOBALE NAVIGATIONSSATELLITENSYSTEME

Im Gegensatz zur indirekten Positionsbestimmung des INS bilden die globalen Navigationssatellitensysteme (*engl. Global Navigation Satellite System* (GNSS)) eine direkte Methode zur Positionsbestimmung. Eines der ersten Systeme dieser Art war das 1960 von den US-Amerikanern entwickelte *Transit* bzw. *Navigation Satellite System* (NAVSAT). Mit insgesamt zehn Satelliten erreichte dieses System eine Genauigkeit von 200 m. Aktuell sind vier Satellitennavigationssysteme in Betrieb bzw. im Aufbau (siehe [70]):

GPS: Das *Global Positioning System* (GPS) oder präziser das *Navigation System with Timing and Ranging* (NAVSTAR)-GPS ist das Folgesystem von *Transit*. Die Entwicklung dieses globalen Satellitenna-

5 Für die barometrische Höhe wird aus dem griechischen Alphabet das η gewählt, was im lateinischen dem kleinen h entspricht.

6 Eine Luftdruckänderung von lediglich 0.1 % gegenüber dem Luftdruck auf Meereshöhe entspricht einer Höhendifferenz von ≈ 8.4 m.

vigationssystems wurde in den 1970er Jahren von den US-Amerikanern vornehmlich zu militärischen Zwecken gestartet. 1995 bestand dieses System aus 24 Satelliten und erreichte damit die volle Funktionalität (*engl. Full Operational Capability* (FOC)). Seitdem im Jahr 2000 die zuvor integrierte künstliche Ungenauigkeit abgeschaltet worden ist, ist dieses System auch für zivile Bereiche mit deutlich höherer Genauigkeit nutzbar.

GLONASS: Das von der russischen Föderation betriebene *Globalnaja Nawigazionnaja Sputnikowaja Sistema* (GLONASS) ist vom Aufbau her dem US-amerikanischen GPS sehr ähnlich. Etwa ein Jahr nach dem FOC von GPS war auch das russische Satellitennavigationssystem vollständig ausgebaut und bestand ebenfalls aus 24 Satelliten. Aufgrund fehlender finanzieller Investitionen verringerte sich die Anzahl aktiver Satelliten bis 2001 auf sieben. Erst durch die Investition des russischen Staates von 67 Milliarden Rubel (etwa 1.8 Milliarden Euro) konnte das System bis 2011 wieder auf den ursprünglichen Funktionsumfang ausgebaut werden.

BEIDOU / COMPASS: Neben den USA und Russland entwickelt gleichermaßen China ein eigenes Satellitennavigationssystem. Im Gegensatz zu GPS oder GLONASS handelt es sich bei *BeiDou* in seiner ersten Ausbaustufe jedoch um ein regionales Navigationssystem, welches auf geostationären Satelliten basiert. In den nächsten Generationen soll *BeiDou* ebenfalls zu einem globalen Satellitennavigationssystem ausgebaut und bis zum Jahr 2020 auf insgesamt 35 Satelliten (davon 5 geostationär) erweitert werden (siehe [105]).

GALILEO: Das in Kooperation der Europäischen Union (EU) und der europäischen Weltraumorganisation (*engl. European Space Agency* (ESA)) im Aufbau befindliche Satellitennavigationssystem *Galileo* wurde in seiner Entwurfsphase vordringlich für zivile Anwendungen konzipiert. Nach anfänglichen Verzögerungen starteten im Oktober 2011 die ersten zwei von vier geplanten Satelliten im Rahmen der *In-Orbit Validation* (IOV)-Phase. Bis zur FOC-Phase sollen 14 weitere Satelliten betriebsbereit sein, so dass zum Probebetrieb von *Galileo* insgesamt 18 Satelliten zur Verfügung stehen. Bis zum Jahr 2020 soll dann die vollständige Verfügbarkeit und Einsatzbereitschaft mit 30 geplanten Satelliten abgeschlossen sein (siehe [49, 50]).

4.4.1 *Funktionsweise*

Grundsätzlich weisen die zuvor genannten globalen Navigationssatellitensysteme eine vergleichbare Funktionsweise sowie Systemarchitektur auf. Dabei basiert die Positionsbestimmung des Empfängers

grundsätzlich auf der Auswertung der übertragenen Satelliteninformationen, wie beispielsweise Bahninformationen, Korrekturinformationen und einem Zeitstempel. Technische Unterschiede sind vor allem in der Art der Informationsübertragung, wie zum Beispiel das genutzte Modulationsverfahren sowie dessen Trägerfrequenzen, zu finden.

Systemarchitektur

Die Systemarchitektur der verschiedenen *Global Navigation Satellite Systems* kann allgemein in drei Segmente unterteilt werden:

WELTRAUMSEGMENT: Dieses besteht aus den Navigationssatelliten, die kontinuierlich ihre spezifischen Informationen aussenden. Diese setzen sich aus den *Ephemeriden*, welche die Informationen zur Satellitenumlaufbahn mathematisch beschreiben, dem *Zeitstempel* und dem *Almanach*, der die Bahndaten aller Satelliten in reduzierter Genauigkeit sowie zusätzliche Integritätsinformationen enthält, zusammen.

BODENSEGMENT: Die verschiedenen Kontrollstationen auf der Erde koordinieren und überwachen den gesamten Satellitenverband. Aufgabe des Bodensegments ist dabei die Kontrolle und Korrektur der Bahndaten und der satelliteneigenen Atomuhren. Dabei wird zur exakten Positionsbestimmung des Satelliten der inverse Ansatz der eigentlichen GNSS-Positionsbestimmung genutzt. Hierzu wird mittels vier bekannter Positionen auf der Erde (*Kontrollstationen*) die genaue Position des Satelliten und dessen Uhrengenauigkeit berechnet.

NUTZERSEGMENT: Das Nutzersegment stellt die Anwender und Applikationen des globalen Navigationssatellitensystems dar. Hier werten hochintegrierte Signalempfänger die Satelliteninformationen aus und können darüber die eigene Position sowie Geschwindigkeit bestimmen.

Positionsbestimmung

Aufgrund der alleinigen Auswertung von GPS-Signalen in den im Folgenden genutzten Flugplattformen wird im Rahmen dieser Arbeit ausschließlich die nähere Funktionsweise vom GPS betrachtet und ausgeführt. Jedoch sei für eine ausführliche Diskussion der Eigenschaften und Funktionsweisen aller Satellitennavigationsmethoden auf die Arbeiten von [70, 109] und [180] verwiesen, welche sehr detailliert auf die komplexen Zusammenhänge eingehen und die Grundlage der hier geführten Diskussion bilden.

Zur Übermittlung der Satelliteninformationen, wie Ephemeriden oder des Almanachs, nutzt das GPS zwei unabhängige Trägerfrequen-

zen im Bereich von 1 GHz bis 2.6 GHz (L-Band). Hierzu senden zwar alle Satelliten auf den selben Trägerfrequenzen (1575.42 MHz $\widehat{=}$ L1-Frequenz und 1227.60 MHz $\widehat{=}$ L2-Frequenz), jedoch mit einem spezifischen Code, der auf die Trägerwelle aufmoduliert wird. Es reicht jedoch aus, die L1-Frequenz zu empfangen[7], um die Position und Geschwindigkeit vollständig zu bestimmen. Um die Informationen bzw. die Satelliten zu identifizieren, wird das *Code Division Multiple Access* (CDMA)-Multiplexverfahren bei der Übertragung eingesetzt. Aufgrund dessen, dass jedem Satelliten eine spezielle Pseudozufallsfolge (*engl. Pseudo Random Noise* (PRN))[8] zugeordnet ist, kann der Empfänger durch Korrelationsverfahren die Satelliteninformationen zurückgewinnen und die Navigationsnachricht des jeweiligen Satelliten decodieren. Die Navigationsnachricht enthält unter anderem die Ephemeriden, den Almanach sowie die Systemzeit. Der Empfänger kann durch die Laufzeit des Signals die Entfernung zum Satelliten bestimmen. Da jedoch die empfängerinterne Uhr nicht mit der Satellitenuhr (GPS-Systemzeit) synchronisiert ist, kommt es zu einem Zeitfehler und somit zu einer fehlerbehafteten Laufzeitmessung. Aus diesem Grund wird die bestimmte Entfernung als *Pseudorange* bezeichnet. Zur eindeutigen Bestimmung der Position muss nun ein Gleichungssystem aus vier Unbekannten (Position des Empfängers und des Uhrenfehlers zwischen der Empfängeruhr und Satellitenuhr) gelöst werden. Empfängt der GPS-Empfänger dementsprechend mindestens vier Satelliten gleichzeitig und störungsfrei, zeigt [70] eine Methode zur Lösung der vier Unbekannten mittels Minimierung der Fehlerquadratsumme. Somit kann eine Positionslösung bestimmt werden. Allgemein erhält man von aktuellen GPS-Empfängern die Positionslösung im WGS-84-Koordinatensystem in Form eines Vektors aus der geographischen Länge λ, der geographischen Breite φ und der Höhe h:

$$\vec{p}^{\,wgs} = \begin{pmatrix} \lambda \\ \varphi \\ h \end{pmatrix}. \tag{4.24}$$

Zur Bestimmung der Navigationslösung, wie in Abbildung 12 auf Seite 35 gezeigt, wird diese globale Positionsangabe in ein lokales Koordinatensystem entsprechend Abschnitt 3.2.2 und Abschnitt 3.2.3 transformiert. Somit kann die Position der Flugplattform $\vec{p}^{\,n}_{GNSS}$ im Navigationskoordinatensystem bestimmt werden.

7 In der Regel handelt es sich bei den kostengünstigen Empfängern um Ein-Frequenz-Empfänger, welche die L1-Frequenz auswerten.

8 Beim GPS werden diese Pseudozufallsfolgen als *Coarse/Acquisition* (C/A)- und *Precise/Encrypted* (P)-*Code* bezeichnet.

4.4.2 *Zusammenfassung und Diskussion*

Die Nutzung eines globalen Navigationssatellitensystems hat spezifische Vorteile. Von den bisher beschriebenen Sensorsystemen ist das GNSS das einzige, welches ohne bekannte Startparameter die Position und Geschwindigkeit absolut bestimmen kann. Dabei zeigt sich, dass das globale System zum inertial arbeitenden INS komplementäre Eigenschaften aufweist. So ist die Positionsbestimmung beim INS rekursiv und damit einer Fehlerdrift unterworfen. Beim GNSS erfolgt die Positionsbestimmung absolut, auf Basis des Triangulationsprinzips. Dabei kann die verhältnismäßig geringe Aktualisierungsrate des globalen Systems[9] (1 Hz bis 10 Hz) durch das INS (100 Hz bis 1000 Hz) kompensiert werden. Aus diesen Gründen ist eine Kopplung beider Systeme zielführend und in vielen Anwendungen, beispielsweise in der Luftfahrt, gängige Praxis. Dabei können die Fehler des kurzzeitgenauen Trägheitsnavigationssystems durch das langzeitgenaue globale Navigationssystem bestimmt und somit kompensiert werden. Im Gegenzug kann ein Ausfall des GNSS durch das INS temporär überbrückt werden, beispielsweise wenn durch Abschattung oder Mehrwegeausbreitung der Empfang oder die Integrität des Satellitensignals nicht gewährleistet ist. Dabei sind in der Literatur verschiedene Kopplungsmöglichkeiten zu finden. Diese werden bezeichnet als *Uncoupled*, *Loosely Coupled*, *Tightly Coupled* und *Ultra-Tightly/Deeply Coupled*. Die Grundprinzipien dieser Methoden sind im Anhang A.1 ab Seite 213 zu finden. Üblicherweise werden derartige Kopplungsmethoden auf Basis von stochastischen Verfahren zur Sensorfusion umgesetzt. Den Ausgangspunkt bildet üblicherweise das Kalman-Filter, auf welches unter anderem im folgenden Kapitel 5 *Datenfilterung und Sensorfusion* eingegangen wird.

9 Diese Aktualisierungsraten beziehen sich auf gängige, kostengünstige Produkte. Für Spezialanwendungen sind einige wenige *High-End*-GNSS-Empfänger verfügbar, die Updateraten bis zu 100 Hz aufweisen.

5

DATENFILTERUNG UND SENSORFUSION

Aufbauend auf der Beschreibung der unterschiedlichen Sensoren und Messsysteme im vorhergehenden Kapitel soll im Folgenden eine Diskussion etablierter Filteralgorithmen stattfinden. Dabei liegt der Fokus prinzipiell auf zwei Arten von Filtern, einerseits zur Sensordatenvorfilterung und andererseits zur Sensordatenfusion. Beide Techniken dienen zur Rauschminderung des Signals sowie zur Genauigkeitssteigerung der gesuchten Messgröße. Beispielhaft für die Anwendung derartiger Filtertechniken sei die zuvor betrachtete Stützung eines INS mit weiteren Sensoren genannt. Neben der Klasse der digitalen Filter wird ebenfalls das Zustandsraumfilter (Kalman-Filter) behandelt.

5.1 TRANSVERSALFILTER

In vielen Anwendungsbereichen stellt die Vorverarbeitung verschiedenster Sensordaten eine wichtige Anforderung dar. Hier gilt es vordringlich, rauschbehaftete Rohdaten zu glätten oder bestimmte Frequenzen zu filtern. Besonders im Bereich der relevanten digitalen Signalverarbeitung haben sich die Filterarten des *Finite Impulse Response* (FIR) bzw. *Infinite Impulse Response* (IIR) besonders hervorgetan. So zeichnet sich das Transversalfilter, auch FIR-Filter[1] genannt, besonders durch seine Filterstabilität aus, da die Impulsantwort endlich ist. Definitionsgemäß ist ein Filter M-ter Ordnung aus M Verzögerungselementen (z^{-1}) und $M+1$ Filterkoeffizienten ($b_0 \dots b_M$) aufgebaut. Die allgemeine Filtergleichung ergibt sich somit zu:

$$y_k = \sum_{j=0}^{M} b_j s_{k-j} \, . \tag{5.1}$$

Dementsprechend errechnet sich das Ausgangssignal y_k aus einer Linearkombination von $M+1$ Eingangssignalen. Aufgrund der Kausalitätsbedingung kann das Ausgangssignal jedoch nur vom Eingangssignal s_k und den M zurückliegenden Eingangswerten abhängen. Aufgrund dieser Filterarchitektur kann von einem linearen, zeitinvarianten (*engl. Linear Time-Invariant* (LTI)) System gesprochen werden. Der schematische Aufbau ist in Abbildung 15 auf Seite 52 dargestellt.

1 Eines der wohl bekanntesten FIR-Filter ist das gleitende Mittelwertfilter (*engl. Moving-Average-Filter*), bei dem alle $M+1$ Filterkoeffizienten den Wert $\frac{1}{M+1}$ besitzen.

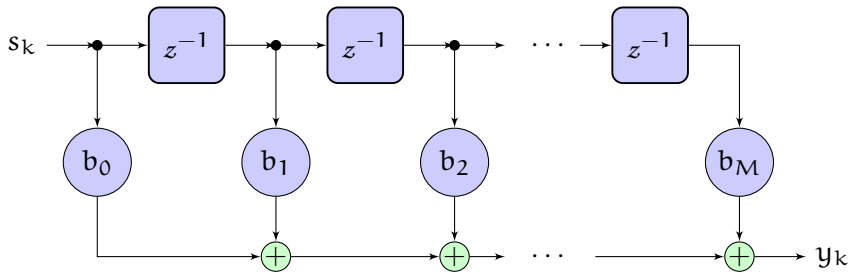


Abbildung 15: Prinzip eines FIR-Filters

5.2 REKURSIVFILTER

Im Gegensatz zum FIR-Filter beinhaltet das Rekursivfilter (IIR-Filter) ebenfalls einen Rückkopplungszweig und kann dadurch eine unendlich lange Impulsantwort liefern (*engl. Infinite Impulse Response*). Demzufolge ist das Ausgangssignal nicht mehr allein vom Eingangssignal abhängig, sondern ebenfalls von den vorherigen Ausgangswerten. Daraus folgt:

$$y_k = \sum_{j=0}^{M} b_j s_{k-j} + \sum_{j=1}^{N} a_j y_{k-j} \,. \tag{5.2}$$

Grundsätzlich ist dieses Filter nicht phasenlinear. Im Gegensatz zum Transversalfilter kann allerdings bei gleicher Filterordnung eine deutlich stärkere Filterwirkung erzielt werden. Der prinzipielle Aufbau eines IIR-Filters ist in Abbildung 16 dargestellt. Der rekursive Anteil

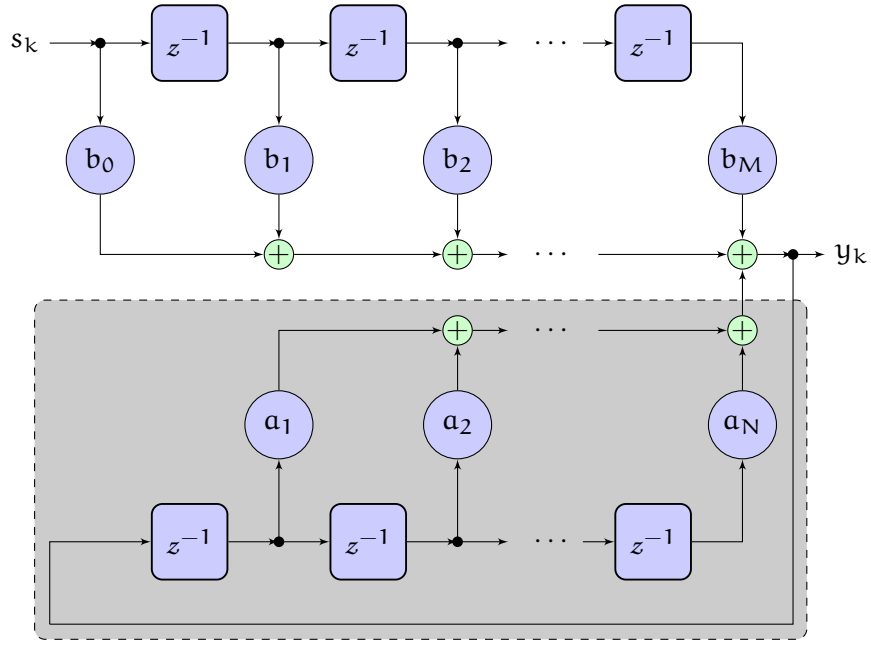


Abbildung 16: Prinzip eines IIR-Filters

ist in *grau* hinterlegt. Dieser spezielle digitale Filter findet in Kapitel 9 seine Anwendung, da dieser dort zur effizienten Sensorvorfilterung der IMU eingesetzt wird. Für eine detailliertere Auseinandersetzung mit der Theorie dieser digitalen Filter sei an dieser Stelle auf [78] verwiesen.

5.3 KALMAN-FILTER

Das Kalman-Filter gehört zur Gruppe der stochastischen Filter und zählt zu den populärsten Filtern in der Signalverarbeitung. Unter dem Titel „*A new approach to linear filtering and prediction problems*“ (siehe [89]) veröffentlichte im Jahr 1960 *R.E. Kalman* seine Berechnungsmethode zur optimalen Filterung von linearen, zeitdiskreten Systemen, welche schnell besondere Aufmerksamkeit im *Apollo*-Programm der NASA erhielt (siehe [66, 146]). Im Gegensatz zu klassischen Ansätzen, wie dem FIR- oder dem IIR-Filter, liegt der Grundgedanke bei diesem Filteransatz in der Modellierung des Systems im Zustandsraum, wobei über den rekursiven Berechnungsansatz der optimale Systemzustand geschätzt wird[2]. Die Umsetzung erfolgt dabei in zwei Schritten, bei denen zuerst der Systemzustand prädiziert (*engl. Prediction*) und im Weiteren durch den Messwert korrigiert (*engl. Correction*) wird. Dabei wird im Vorhersageschritt der zeitlich zurückliegende (*rekursive*) Schätzwert genutzt, um den aktuellen Systemzustand zu prädizieren. Im Korrekturschritt wird daraufhin der Messwert hinzugefügt, womit die Kovarianz des Schätzfehlers neu bestimmt werden kann.

Im Folgenden soll zum Grundverständnis des Kalman-Filters auf dessen mathematische Zusammenhänge eingegangen werden. Darüber hinaus sei für eine umfassende Diskussion auf folgende Arbeiten verwiesen: Für eine vollständige mathematische Beschreibung des Filteransatzes sei die Originalarbeit von *R.E. Kalman* (siehe [89]) genannt. Des Weiteren geben *Bar-Shalom et al.* in [14] eine ausführliche Beschreibung und Diskussion zur Zustandsschätzung von nichtlinearen Systemen, beispielsweise bei der Objektverfolgung (*engl. Tracking*). *Welch* und *Bishop* geben in [171] eine grundsätzliche Einführung in dieses Themengebiet. Eine sehr umfassende Darstellung des Kalman-Filters ist von *Grewal* und *Andrews* in [65] zu finden.

5.3.1 *Das lineare zeitdiskrete Kalman-Filter*

Bei der Grundform des Kalman-Filters handelt es sich um die Zustandsraumdarstellung eines linearen, zeitdiskreten Systems. Der Sys-

2 Im Zustandsraummodell des Kalman-Filters kann neben dem vorhersagbaren deterministischen Anteil auch ein zufälliger, nicht-deterministischer Einfluss abgebildet werden (vergleiche [115]).

temzusammenhang kann dabei durch die Differenzengleichung im Zustandsraum modelliert werden als[3]

$$\vec{x}_k = \mathbf{\Phi}_{k-1}\vec{x}_{k-1} + \mathbf{B}_{k-1}\vec{u}_{k-1} + \mathbf{G}_{k-1}\vec{w}_{k-1}. \tag{5.3}$$

Der Systemzustand wird darin durch den Vektor $\vec{x}_k \in \mathbb{R}^n$ repräsentiert. Die $n \times n$-Matrix $\mathbf{\Phi}_{k-1}$ wird als Transitionsmatrix bezeichnet und beschreibt den Übergang des Zustandsvektors vom Zeitpunkt t_{k-1} in den Zeitpunkt t_k. Neben dem vergangenen Systemzustand können ebenfalls weitere Eingangsgrößen durch den Vektor $\vec{u}_{k-1} \in \mathbb{R}^m$ mittels der $n \times m$-Steuermatrix $\mathbf{B}_{k-1}$ in den Zustandsraum transformiert werden. Zufällige, nicht-deterministische Einflüsse, wie das System- bzw. Prozessrauschen, werden durch den Term $\vec{w}_{k-1} \in \mathbb{R}^l$ abgebildet und können durch die $n \times l$-Einflussmatrix $\mathbf{G}_{k-1}$ in das Zustandsraummodell einfließen. Allgemein gilt für die Betrachtung des Kalman-Filters, dass die Rauschprozesse als normalverteilt, mittelwertfrei und weiß definiert sind. Dementsprechend kann das System- bzw. Prozessrauschen als Zufallsvariable mit dem Erwartungswertvektor 0 und der Kovarianzmatrix $\mathbf{Q}_{k-1}$ durch Gleichung 5.4 dargestellt werden:

$$\vec{w}_{k-1} \sim N(0, \mathbf{Q}_{k-1}). \tag{5.4}$$

Nach der Prädiktion des neuen Systemzustands wird im zweiten Schritt die Messgröße $\tilde{\vec{z}}_k \in \mathbb{R}^p$ verarbeitet. Hierbei dient die $p \times n$-Messmatrix $\mathbf{H}_k$ zur Transformation vom Zustandsvektor zur Messgröße. Das Messrauschen wird durch den Vektor $\vec{v}_k \in \mathbb{R}^p$ abgebildet. Somit ergibt sich das Messmodell zu:

$$\tilde{\vec{z}}_k = \mathbf{H}_k\vec{x}_k + \vec{v}_k\,. \tag{5.5}$$

Ebenfalls gilt für das Messrauschen, dass es sich um einen normalverteilten, mittelwertfreien und weißen Rauschprozess mit dem Erwartungswertvektor 0 und der Kovarianzmatrix $\mathbf{R}_k$ handelt. Es folgt:

$$\vec{v}_k \sim N(0, \mathbf{R}_k). \tag{5.6}$$

Es gilt, dass das Messrauschen $\vec{v}_k$ und das Systemrauschen $\vec{w}_{k-1}$ unkorreliert sind. Durch die hier eingeführten Bedingungen eines linearen Systemmodels und der normalverteilten, mittelwertfreien, weißen und unkorrelierten Rauschprozesse gilt der Systemzustandsvektor $\vec{x}_k$ ebenfalls als normalverteilt. Diese mehrdimensionalen Normalverteilungen $N(\hat{\vec{x}}_k, \mathbf{P}_k)$ werden durch den Erwartungswertvektor $\hat{\vec{x}}_k$ und die Kovarianzmatrix $\mathbf{P}_k$ bestimmt[4]. Es gilt:

$$\hat{\vec{x}}_k = E[\vec{x}_k], \tag{5.7}$$

3 Die Schreibweise des Systemmodells in Bezug auf den Rauschterm ist sowohl mit als auch ohne die Einflussmatrix $\mathbf{G}$ gebräuchlich (siehe [171, 172]). Im Weiteren wird der Rauschterm als additive Größe mit der Einflussmatrix betrachtet.

4 Für eine eingehendere mathematische Diskussion dieser Zusammenhänge sei auf [65] verwiesen.

$$\mathbf{P}_k = \mathrm{E}\left[\left(\vec{x}_k - \hat{\vec{x}}_k\right)\left(\vec{x}_k - \hat{\vec{x}}_k\right)^T\right]. \tag{5.8}$$

Zur Bestimmung dieser Parameter liefert das Kalman-Filter ein optimales, rekursives, mathematisches Verfahren.

Bevor im Folgenden die Kalman-Filtergleichungen erörtert werden können, müssen zuvor weitere Definitionen getroffen werden. Wie die Gleichung 5.7 zeigt, handelt es sich bei $\hat{\vec{x}}_k$ um den Schätzwert des tatsächlichen Systemzustands $\vec{x}_k$. Der Schätzwert wird dementsprechend mit dem Index $\hat{\square}$ gekennzeichnet. Des Weiteren wird, wie Eingangs schon dargestellt, die Kalman-Filterberechnung in zwei Schritten ausgeführt, indem zuerst der Systemzustand prädiziert und dann durch den Messwert korrigiert wird. Steht die Verarbeitung des Messwertes noch aus, handelt es sich um einen sogenannten *a priori*[5] Wert, welcher durch den Index $\square^-$ dargestellt wird. Nach erfolgter Messwertverarbeitung wird von einem *a posteriori*[6] Schätzwert gesprochen, welcher durch ein $\square^+$ gekennzeichnet ist.

Prädiktion

Im Prädiktionsschritt bestimmt das Kalman-Filter aus dem *a posteriori* Systemzustand $\hat{\vec{x}}^+_{k-1}$ und dem Eingangsgrößenvektor $\vec{u}_{k-1}$ die neue Zustandsvektorschätzung. Aufgrund des Zustandsraummodells aus Gleichung 5.3 ergibt sich der *a priori* Zustandsvektor $\hat{\vec{x}}^-_k$ zu:

$$\hat{\vec{x}}^-_k = \mathbf{\Phi}_{k-1}\hat{\vec{x}}^+_{k-1} + \mathbf{B}_{k-1}\vec{u}_{k-1}. \tag{5.9}$$

Ebenfalls wird in diesem Schritt die *a priori* Schätzfehlerkovarianzmatrix prädiziert. Diese ergibt sich als Funktion der Kovarianzmatrix des *a posteriori* Schätzfehlers $\mathbf{P}^+_{k-1}$ und der Kovarianzmatrix des System- bzw. Prozessrauschens $\mathbf{Q}_{k-1}$:

$$\mathbf{P}^-_k = \mathbf{\Phi}_{k-1}\mathbf{P}^+_{k-1}\mathbf{\Phi}_{k-1}{}^T + \mathbf{G}_{k-1}\mathbf{Q}_{k-1}\mathbf{G}^T_{k-1}. \tag{5.10}$$

Korrektur

Im zweiten Schritt des Kalman-Filters soll der *a priori* Schätzwert durch den Messwert korrigiert werden. Der *a posteriori* Zustandsvektor wird als Funktion des *a priori* Zustandsvektors und der gewichteten Differenz zwischen erwartetem Messwert und tatsächlichem Messwert durch die Zustandsgleichung 5.11 bestimmt:

$$\hat{\vec{x}}^+_k = \hat{\vec{x}}^-_k + \mathbf{K}_k\left(\tilde{\vec{z}}_k - \mathbf{H}_k\hat{\vec{x}}^-_k\right). \tag{5.11}$$

5 Abgeleitet aus dem lateinischen *a priori*, vom Früheren her.
6 Abgeleitet aus dem lateinischen *a posteriori*, vom Späteren her.

Der Term $(\tilde{\vec{z}}_k - \mathbf{H}_k \hat{\vec{x}}_k^-)$ wird allgemein auch als *Innovation* bezeichnet. Die Gewichtungsmatrix $\mathbf{K}_k$, auch als *Kalman-Gain* oder Verstärkungsfaktor bezeichnet, muss zuvor durch Minimierung der *a posteriori* Fehlerkovarianzmatrix durch Gleichung 5.12 bestimmt werden:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} . \tag{5.12}$$

Zuletzt wird die Kovarianzmatrix des Schätzfehlers aktualisiert:

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^- . \tag{5.13}$$

Zu Beginn der Kalman-Filterschleife sind zudem die Anfangsbedingungen $\hat{\vec{x}}_0$ und $\mathbf{P}_0$ zu initialisieren:

$$\hat{\vec{x}}_{k-1}^+ = \hat{\vec{x}}_0 \tag{5.14}$$
$$\mathbf{P}_{k-1}^+ = \mathbf{P}_0 . \tag{5.15}$$

5.3.2 *Kalman-Filter für nichtlineare Systeme*

In vielen Applikationen, beispielsweise bei der Sensorfusion von *Inertial Navigation Systems* (INS), kann jedoch nicht von einem linearen, zeitdiskreten Systemmodell ausgegangen werden. Um trotzdem die zuvor aufgestellten Filtergleichungen nutzen zu können, ist eine lineare Approximation des Systemverhaltens notwendig. Sofern die Nichtlinearitäten des Systems gering sind, kann dies mittels Linearisierung um einen Arbeitspunkt erreicht werden.

Linearisierung einer nichtlinearen Funktion

Ausganspunkt ist das nichtlineare, zeitkontinuierliche System mit[7]:

$$\dot{\vec{x}}(t) = \vec{f}\left(\vec{x}(t), \vec{u}(t), t\right) + \mathbf{G}(t)\vec{w}(t) \qquad \text{mit} \qquad \vec{x}(t_0) = \vec{x}_0 \tag{5.16}$$
$$\tilde{\vec{z}}(t) = \vec{h}\left(\vec{x}(t), t\right) + \vec{v}(t) . \tag{5.17}$$

Mit den Linearisierungspunkten $\bar{\vec{\square}}(t)$ und der Forderung nach einer hinreichend geringen Abweichung $\Delta\vec{\square}(t)$ von diesen gilt:

$$\vec{x}(t) = \bar{\vec{x}}(t) + \Delta\vec{x}(t) \tag{5.18}$$
$$\vec{u}(t) = \bar{\vec{u}}(t) + \Delta\vec{u}(t) . \tag{5.19}$$

Durch Einsetzen in Gleichung 5.16 und 5.17 folgt dementsprechend:

$$\dot{\vec{x}}(t) = \vec{f}\left(\bar{\vec{x}}(t) + \Delta\vec{x}(t), \bar{\vec{u}}(t) + \Delta\vec{u}(t), t\right) + \mathbf{G}(t)\vec{w}(t) \tag{5.20}$$
$$\tilde{\vec{z}}(t) = \vec{h}\left(\bar{\vec{x}}(t) + \Delta\vec{x}(t), t\right) + \vec{v}(t) . \tag{5.21}$$

7 Es gilt $\dot{\vec{x}}(t) = \frac{\partial \vec{x}(t)}{\partial t}$.

Gemäß der Taylor-Reihenentwicklung und dem Abbruch nach dem linearen Glied gilt näherungsweise:

$$\begin{aligned}\dot{\vec{x}}(t) \approx \vec{f}\left(\bar{\vec{x}}(t), \bar{\vec{u}}(t), t\right) &+ \left.\frac{\partial \vec{f}\left(\vec{x}(t), \vec{u}(t), t\right)}{\partial \vec{x}}\right|_{\vec{x}(t)=\bar{\vec{x}}(t)} \Delta\vec{x}(t) \\ &+ \left.\frac{\partial \vec{f}\left(\vec{x}(t), \vec{u}(t), t\right)}{\partial \vec{u}}\right|_{\vec{u}(t)=\bar{\vec{u}}(t)} \Delta\vec{u}(t) \\ &+ \mathbf{G}(t)\vec{w}(t) \ .\end{aligned} \tag{5.22}$$

Analog kann Gleichung 5.21 entwickelt werden zu:

$$\tilde{\vec{z}}(t) \approx \vec{h}\left(\bar{\vec{x}}(t), t\right) + \left.\frac{\partial \vec{h}\left(\vec{x}(t), t\right)}{\partial \vec{x}}\right|_{\vec{x}(t)=\bar{\vec{x}}(t)} \Delta\vec{x}(t) + \vec{v}(t) \ . \tag{5.23}$$

Des Weiteren gilt:

$$\dot{\vec{x}}(t) = \dot{\bar{\vec{x}}}(t) + \Delta\dot{\vec{x}}(t) \tag{5.24}$$

$$\dot{\bar{\vec{x}}}(t) = \vec{f}\left(\bar{\vec{x}}(t), \bar{\vec{u}}(t), t\right) \ . \tag{5.25}$$

Wird dies in Gleichung 5.22 eingesetzt, ergibt sich:

$$\begin{aligned}\Delta\dot{\vec{x}}(t) = &\left.\frac{\partial \vec{f}\left(\vec{x}(t), \vec{u}(t), t\right)}{\partial \vec{x}}\right|_{\vec{x}(t)=\bar{\vec{x}}(t)} \Delta\vec{x}(t) \\ &+ \left.\frac{\partial \vec{f}\left(\vec{x}(t), \vec{u}(t), t\right)}{\partial \vec{u}}\right|_{\vec{u}(t)=\bar{\vec{u}}(t)} \Delta\vec{u}(t) + \mathbf{G}(t)\vec{w}(t) \ .\end{aligned} \tag{5.26}$$

Mit den Beziehungen

$$\tilde{\vec{z}}(t) = \bar{\vec{z}}(t) + \Delta\vec{z}(t) \tag{5.27}$$

$$\bar{\vec{z}}(t) = \vec{h}\left(\bar{\vec{x}}(t), t\right) \ , \tag{5.28}$$

eingesetzt in Gleichung 5.23, folgt schließlich:

$$\Delta\vec{z}(t) = \left.\frac{\partial \vec{h}\left(\vec{x}(t), t\right)}{\partial \vec{x}}\right|_{\vec{x}(t)=\bar{\vec{x}}(t)} \Delta\vec{x}(t) + \vec{v}(t) \ . \tag{5.29}$$

Mit dem jeweils linearisierten Systemmodell aus Gleichung 5.26 und analog dazu dem Messmodell aus Gleichung 5.29 können die linearisierten Transformationsmatrizen[8] entsprechend:

$$\mathbf{F}(t) = \left.\frac{\partial \vec{f}\left(\vec{x}(t), \vec{u}(t), t\right)}{\partial \vec{x}}\right|_{\vec{x}(t)=\bar{\vec{x}}(t)} \tag{5.30}$$

8 Ebenfalls kann das nichtlineare, zeitkontinuierliche System als Funktion von $\dot{\vec{x}}(t) = \vec{f}\left(\vec{x}(t), \vec{u}(t), \vec{w}(t), t\right)$ sowie $\tilde{\vec{z}}(t) = \vec{h}\left(\vec{x}(t), \vec{v}(t), t\right)$ (nicht additiver Rauschterm) beschrieben werden, womit sich zwei weitere Transitionsmatrizen ergeben zu $\mathbf{G}(t) = \frac{\partial f(\vec{t})}{\partial \vec{w}(t)}$ und $\mathbf{V}(t) = \frac{\partial h(\vec{t})}{\partial \vec{v}(t)}$ (siehe [171]).

$$\mathbf{B}(t) = \left.\frac{\partial \vec{f}\,(\vec{x}(t), \vec{u}(t), t)}{\partial \vec{u}}\right|_{\vec{u}(t)=\bar{\vec{u}}(t)} \tag{5.31}$$

$$\mathbf{H}(t) = \left.\frac{\partial \vec{h}\,(\vec{x}(t), t)}{\partial \vec{x}}\right|_{\vec{x}(t)=\bar{\vec{x}}(t)} \tag{5.32}$$

ausgedrückt werden. Die Transformationsmatrizen entsprechen der Jacobi-Matrix einer vektorwertigen Funktion $\vec{f}$ im Linearisierungspunkt $\bar{\vec{x}}$:

$$\left.\frac{\partial \vec{f}}{\partial \vec{x}}\right|_{\vec{x}=\bar{\vec{x}}} = \left.\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}\right|_{\vec{x}=\bar{\vec{x}}} . \tag{5.33}$$

Das Error State Space Kalman-Filter

Das linearisierte oder *Error State Space* Kalman-Filter schätzt den Systemzustand nicht direkt, sondern lediglich die Fehler des vermuteten Systemzustandes. Daher wird auch häufig von einem indirekten Kalman-Filter gesprochen (siehe [172]). Ausgangspunkt sei ebenfalls ein nichtlineares, zeitkontinuierliches System in der Form von:

$$\dot{\vec{x}}(t) = \vec{f}\,(\vec{x}(t), t) + \mathbf{B}(t)\vec{u}(t) + \mathbf{G}(t)\vec{w}(t) \tag{5.34}$$

$$\tilde{\vec{z}}(t) = \vec{h}\,(\vec{x}(t), t) + \vec{v}(t)\ . \tag{5.35}$$

Des Weiteren gilt:

$$\vec{w}(t) \sim \mathrm{N}\,(0, \mathbf{Q}(t)) \tag{5.36}$$

$$\vec{v}(t) \sim \mathrm{N}\,(0, \mathbf{R}(t))\,. \tag{5.37}$$

Entsprechend Gleichung 5.34 folgt der geschätzte Systemzustand zu:

$$\dot{\hat{\vec{x}}}(t) = \vec{f}\left(\hat{\vec{x}}(t), t\right) + \mathbf{B}(t)\vec{u}(t)\ . \tag{5.38}$$

Wird daraufhin das nichtlineare Systemzustandsmodell mittels einer Taylor-Entwicklung, wie durch Gleichung 5.22 gezeigt, linearisiert, kann Gleichung 5.34 approximiert werden zu:

$$\begin{aligned} \dot{\vec{x}}(t) \approx \vec{f}\left(\bar{\vec{x}}(t), t\right) + \left.\frac{\partial \vec{f}\,(\vec{x}(t), t)}{\partial \vec{x}}\right|_{\vec{x}(t)=\bar{\vec{x}}(t)} \left(\vec{x}(t) - \bar{\vec{x}}(t)\right) \\ + \mathbf{B}(t)\vec{u}(t) + \mathbf{G}(t)\vec{w}(t) \end{aligned} \tag{5.39}$$

und entsprechend Gleichung 5.38 zu:

$$\begin{aligned} \dot{\hat{\vec{x}}}(t) \approx \vec{f}\left(\bar{\vec{x}}(t), t\right) + \left.\frac{\partial \vec{f}\,(\vec{x}(t), t)}{\partial \vec{x}}\right|_{\vec{x}(t)=\bar{\vec{x}}(t)} \left(\hat{\vec{x}}(t) - \bar{\vec{x}}(t)\right) \\ + \mathbf{B}(t)\vec{u}(t)\ . \end{aligned} \tag{5.40}$$

Wie eingangs angeführt, prädiziert das *Error State Space* Kalman-Filter nicht direkt den Systemzustand, sondern den Fehler zwischen dem tatsächlichen und dem vermuteten Systemzustand, womit gilt[9]:

$$\Delta\vec{x}(t) = \vec{x}(t) - \hat{\vec{x}}(t) \; . \tag{5.41}$$

Der Schätzfehler kann in guter Näherung durch Subtraktion der Gleichung 5.40 von Gleichung 5.39 bestimmt werden. Als Linearisierungspunkt wird der geschätzte Systemzustand gewählt. Es folgt:

$$\begin{aligned} \dot{\vec{x}}(t) - \dot{\hat{\vec{x}}}(t) = & \left.\frac{\partial\vec{f}\,(\vec{x}(t),t)}{\partial\vec{x}}\right|_{\vec{x}(t)=\hat{\vec{x}}(t)} \left(\vec{x}(t) - \hat{\vec{x}}(t)\right) \\ & + \mathbf{G}(t)\vec{w}(t) \; . \end{aligned} \tag{5.42}$$

Wie in Gleichung 5.30 eingeführt, kann die linearisierte Transformationsmatrix als Jacobi-Matrix der vektorwertigen Funktion $\vec{f}\,(\vec{x}(t),t)$ bestimmt werden. Somit ergibt sich für die gesuchte Systemmatrix:

$$\mathbf{F}(t) = \left.\frac{\partial\vec{f}\,(\vec{x}(t),t)}{\partial\vec{x}}\right|_{\vec{x}(t)=\hat{\vec{x}}(t)} \; . \tag{5.43}$$

Die linearisierte Systemzustandsgleichung kann entsprechend Gleichung 5.42 durch Einsetzen von Gleichungen 5.41 und 5.43 beschrieben werden durch

$$\Delta\dot{\vec{x}}(t) = \mathbf{F}(t)\Delta\vec{x}(t) + \mathbf{G}(t)\vec{w}(t) \; . \tag{5.44}$$

Gemäß der indirekten Systemzustandsschätzung werden ebenfalls die Messwerte als Differenz zwischen dem vorliegenden und dem geschätzten Messwertvektor verarbeitet. Ausgangspunkt ist das nichtlineare Messmodell aus Gleichung 5.35, das, analog Gleichung 5.23, entsprechend einer Taylor-Reihe entwickelt wird. Somit kann der geschätzte Messwertvektor näherungsweise bestimmt werden mit

$$\hat{\vec{z}}(t) \approx \vec{h}\left(\bar{\vec{x}}(t),t\right) + \left.\frac{\partial\vec{h}\,(\vec{x}(t),t)}{\partial\vec{x}}\right|_{\vec{x}(t)=\bar{\vec{x}}(t)} \left(\hat{\vec{x}}(t) - \bar{\vec{x}}(t)\right). \tag{5.45}$$

In Analogie zu der Systemzustandsformulierung aus Gleichung 5.42 wird auch in diesem Fall die Differenz aus der approximierten Funktion des vorliegenden (entspricht Gleichung 5.23) und des geschätzten Messwertvektors (siehe Gleichung 5.45) gebildet. Mit dem geschätzten Systemzustand als Linearisierungspunkt folgt:

$$\tilde{\vec{z}}(t) - \hat{\vec{z}}(t) = \left.\frac{\partial\vec{h}\,(\vec{x}(t),t)}{\partial\vec{x}}\right|_{\vec{x}(t)=\hat{\vec{x}}(t)} \left(\vec{x}(t) - \hat{\vec{x}}(t)\right) + \vec{v}(t) \; . \tag{5.46}$$

9 In [172] ist die Herleitung des *Error State Space* Kalman-Filters als Funktion $\Delta\vec{x}(t) = \hat{\vec{x}}(t) - \vec{x}(t)$ dargestellt, womit sich unterschiedliche Vorzeichen bezüglich der hier diskutierten Herleitung ergeben. Daher wird an dieser Stelle auf die Darstellung in [65] verwiesen.

Somit entspricht auch hier die Messmatrix der Jacobi-Matrix der vektorwertigen Funktion $\vec{h}\,(\vec{x}(t), t)$. Es folgt:

$$\mathbf{H}(t) = \left.\frac{\partial \vec{h}\,(\vec{x}(t), t)}{\partial \vec{x}}\right|_{\vec{x}(t)=\hat{\vec{x}}(t)} . \tag{5.47}$$

Wird analog zur Gleichung 5.41 der Messfehler beschrieben durch

$$\Delta\vec{z}(t) = \tilde{\vec{z}}(t) - \hat{\vec{z}}(t) \ , \tag{5.48}$$

kann das linearisierte Messmodel nach Gleichung 5.46 formuliert werden zu:

$$\Delta\vec{z}(t) = \mathbf{H}(t)\Delta\vec{x}(t) + \vec{v}(t) \ . \tag{5.49}$$

Hierbei gilt jedoch, dass der zu verarbeitende Messfehler $\Delta\vec{z}(t)$ aus der Differenz des Messwertes $\tilde{\vec{z}}(t)$ und des nichtlinearen Messmodells $\vec{h}\,(\vec{x}(t), t)$ bestimmt wird:

$$\Delta\vec{z}(t) = \tilde{\vec{z}}(t) - \vec{h}\left(\hat{\vec{x}}(t), t\right) . \tag{5.50}$$

Um ebenfalls den Anforderungen des Kalman-Filters eines zeitdiskreten Modells zu entsprechen, muss das Zustandsmodell diskretisiert werden. Es gilt:

$$\mathbf{\Phi}_{k-1} = \mathbf{I} + \mathbf{F}(t_{k-1})\Delta t \tag{5.51}$$

$$\mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^T = \mathbf{G}(t_k)\mathbf{Q}(t_k)\mathbf{G}^T(t_k)\Delta t \ . \tag{5.52}$$

Dabei ist Δt die Zeitspanne des Intervalls $[t_k - t_{k-1}]$ und $\mathbf{I}$ die Einheitsmatrix. Zur Herleitung dieser Zusammenhänge sei auf [65] verwiesen. Dementsprechend gilt nach Gleichung 5.44 für das linearisierte sowie zeitdiskrete Systemzustandsmodell:

$$\Delta\hat{\vec{x}}_k^- = \mathbf{\Phi}_{k-1}\Delta\hat{\vec{x}}_{k-1}^+ + \mathbf{G}_{k-1}\vec{w}_{k-1} . \tag{5.53}$$

Das zeitdiskrete, nichtlineare Messmodell ergibt sich zu:

$$\tilde{\vec{z}}_k = \vec{h}_k\vec{x}_k + \vec{v}_k \ . \tag{5.54}$$

Somit folgt für die diskreten Kalman-Filtergleichungen:

1. *Prädiktion*:

$$\Delta\hat{\vec{x}}_k^- = \mathbf{\Phi}_{k-1}\Delta\hat{\vec{x}}_{k-1}^+ \tag{5.55}$$

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}^+{\mathbf{\Phi}_{k-1}}^T + \mathbf{G}_{k-1}\mathbf{Q}_{k-1}\mathbf{G}_{k-1}^T . \tag{5.56}$$

2. *Korrektur*:

$$\Delta\hat{\vec{x}}_k^+ = \Delta\hat{\vec{x}}_k^- + \mathbf{K}_k\Big(\underbrace{\tilde{\vec{z}}_k - \vec{h}_k(\hat{\vec{x}}_k^-)}_{\Delta\vec{z}_k} - \mathbf{H}_k\Delta\hat{\vec{x}}_k^-\Big) \tag{5.57}$$

$$\mathbf{H}_k = \left.\frac{\partial \vec{h}_k\,(\vec{x}_k)}{\partial \vec{x}_k}\right|_{\vec{x}_k=\hat{\vec{x}}_k} \tag{5.58}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{5.59}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- . \tag{5.60}$$

Das erweiterte Kalman-Filter

Ebenso wie das zuvor hergeleitete linearisierte (indirekte) Kalman-Filter, dient auch das erweiterte Kalman-Filter (*engl. Extended Kalman-Filter* (EKF)) zur Systemzustandsschätzung von nichtlinearen Systemmodellen. Maßgeblicher Unterschied dieser beiden nichtlinearen Verfahren ist beim EKF die direkte Zustandsschätzung des Systems selbst. Daher wird dieses Filter in der Literatur ebenfalls als direktes bzw. *Total State Space* Kalman-Filter bezeichnet. Wie zu Beginn dieses Abschnittes eingeführt, wird das nichtlineare System beschrieben als:

$$\dot{\vec{x}}(t) = \vec{f}\left(\vec{x}(t), \vec{u}(t), t\right) + \mathbf{G}(t)\vec{w}(t)\,; \quad \vec{w}(t) \sim N\left(0, \mathbf{Q}(t)\right) \tag{5.61}$$

$$\tilde{\vec{z}}(t) = \vec{h}\left(\vec{x}(t), t\right) + \vec{v}(t)\,; \quad \vec{v}(t) \sim N\left(0, \mathbf{R}(t)\right). \tag{5.62}$$

Entsprechend der zuvor diskutierten Linearisierung von nichtlinearen Funktionen ergeben sich die Transformationsmatrizen wie folgt zu:

$$\mathbf{F}(t) = \left.\frac{\partial \vec{f}\left(\vec{x}(t), \vec{u}(t), t\right)}{\partial \vec{x}}\right|_{\vec{x}(t)=\hat{\vec{x}}(t)} \tag{5.63}$$

$$\mathbf{H}(t) = \left.\frac{\partial \vec{h}\left(\vec{x}(t), t\right)}{\partial \vec{x}}\right|_{\vec{x}(t)=\hat{\vec{x}}(t)}. \tag{5.64}$$

Analog des *Error State Space* Kalman-Filters soll auch hier das EKF im zeitdiskreten Zustandsraum bestimmt werden. Ausgehend vom nichtlinearen Zustandsmodell

$$\vec{x}_k = \vec{f}_{k-1}\left(\vec{x}_{k-1}, \vec{u}_{k-1}\right) + \mathbf{G}_{k-1}\vec{w}_{k-1} \tag{5.65}$$

und dem zeitdiskreten, nichtlinearen Messmodell

$$\tilde{\vec{z}}_k = \vec{h}_k\left(\vec{x}_k\right) + \vec{v}_k\,, \tag{5.66}$$

können die diskreten Kalman-Filtergleichungen mit den Gleichungen 5.51 und 5.52 formuliert werden zu:

1. *Prädiktion*:

$$\hat{\vec{x}}_k^- = \vec{f}_{k-1}\left(\hat{\vec{x}}_{k-1}^+, \vec{u}_{k-1}\right) \tag{5.67}$$

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}^+{\mathbf{\Phi}_{k-1}}^T + \mathbf{G}_{k-1}\mathbf{Q}_{k-1}\mathbf{G}_{k-1}^T. \tag{5.68}$$

2. *Korrektur*:

$$\hat{\vec{x}}_k^+ = \hat{\vec{x}}_k^- + \mathbf{K}_k\left(\tilde{\vec{z}}_k - \vec{h}_k(\hat{\vec{x}}_k^-)\right) \tag{5.69}$$

$$\mathbf{H}_k = \left.\frac{\partial \vec{h}_k\left(\vec{x}_k\right)}{\partial \vec{x}_k}\right|_{\vec{x}_k=\hat{\vec{x}}_k^-} \tag{5.70}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{5.71}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-. \tag{5.72}$$

5.4 ZUSAMMENFASSUNG UND DISKUSSION

Im letzten Abschnitt sind zwei grundlegend verschiedene Filterarten vorgestellt worden. Dabei sind sowohl das IIR- als auch das Kalman-Filter für das in Kapitel 9 gezeigte *FireFly* MAV-*Framework* essenziell. In diesem wird beispielsweise ein IIR-Filter zur effizienten digitalen Vorfilterung der Sensordaten eingesetzt. Des Weiteren wird im Zusammenhang mit dem neuen MAV-*Framework* auf zwei praktische Systementwürfe des Kalman-Filters, zum einen im *Error State Space* und zum anderen im *Total State Space*, eingegangen.

Teil II

SIMULATIONSBASIERTE ENTWICKLUNG VON MULTIROTORSYSTEMEN

6 SIMULATION VON MULTIROTORSYSTEMEN

Bedingt durch immer neue Aufgabengebiete von unbemannten Flugsystemen, ist innerhalb des letzten Jahrzehnts ein deutlicher Anstieg in der Entwicklung und Verfügbarkeit neuer VTOL-MAV-Systeme zu verzeichnen. Immer größer und vielfältiger werden dabei die Einsatzgebiete von unbemannten Flugplattformen. So ist zum Beispiel die Nutzung eines solchen Systems bei Großveranstaltungen als fliegende Kameraplattform eine attraktive Einsatzmöglichkeit für die Medien- und Filmindustrie. Dabei ereignen sich jedoch immer wieder Unfälle mit diesen Systemen. Beispielsweise stürzte bei einem internationalen Ski-Alpin Wettbewerb ein Multirotorsystem ab und verfehlte einen der Sportler nur knapp (siehe [152]). Durch diesen und ähnliche Vorfälle ist ein Nachweis des sicheren Betriebes dieser Systeme eine elementare Forderung, der auch der Gesetzgeber inzwischen intensiv nachgeht. Hierbei ist die Belastbarkeit von mechanischen Komponenten, verwendeten Materialien oder auch Antrieben durch verschiedene Prüfverfahren bereits im Entwicklungsstadium nachzuweisen. Die Zuverlässigkeit der Software ist hingegen nur mit einem fertiggestellten Prototyp und nur im begrenzten Umfang durch zeit- und kostenintensive Feldtests zu erbringen.

Ein weiterer, aber nicht minder wichtiger Bereich ist die Neu- oder Weiterentwicklung dieser MAV-Systeme. Unabhängig davon, ob wirtschaftliche oder wissenschaftliche Interessen hinter der Entwicklung stehen, ist ein grundlegendes Element zur Verifikation der Entwicklungsergebnisse die Verfügbarkeit von Validierungsdaten. So ist im Besonderen bei der Beurteilung der Algorithmen zur Flugregelung, Lagebestimmung, Positionsschätzung und Umgebungskartierung die Verfügbarkeit von *Ground-Truth*-Daten ein wesentlicher und gleichermaßen komplexer Bestandteil des Validierungsprozesses. Üblicherweise wird hierzu auf *Motion-Capture*-Systeme zurückgegriffen, welche jedoch bauartbedingt Limitierungen im Umgebungs- sowie Dynamikbereich aufweisen.

Aus genannten Gründen wird im Folgenden eine Simulationsumgebung vorgestellt, in die ein bestehendes MAV realitätsnah überführt und die Software des Autopiloten vollständig integriert wird. Die hier erarbeitete Lösung basiert auf der präzisen Modellierung der einzelnen Komponenten, wie Sensoren, Antriebe und dem Bewegungsmodell der Flugplattform. Die dabei erreichte realitätsnahe Abbildung der Sensordaten und der Flugeigenschaften in Echtzeit erlaubt, in Verbindung mit den jederzeit verfügbaren und synchronisierten *Ground-Truth*-Daten, eine effektive und zeiteffiziente Methode zur Entwick-

lung und Validierung von MAV-Algorithmen. Ein signifikantes Ergebnis ist die direkte Portierbarkeit der Software zwischen simuliertem und realem MAV.

6.1 STAND DER TECHNIK

In vielen wissenschaftlichen Bereichen werden Simulationen zur Analyse von heterogenen Zusammenhängen bzw. dynamischen Prozessen genutzt. Mit dem Aufkommen immer leistungsfähigerer Computer können immer komplexere Modelle abgebildet und in computergestützten Simulationen analysiert werden. So zeigen beispielsweise Strömungssimulationen (*engl. Computational Fluid Dynamics* (CFD)) zur Analyse von Flugzeugtragflächen oder Windkraftanlagen nur ein Anwendungsgebiet hierfür auf. Fahr- bzw. Flugsimulatoren werden zur Ausbildung an komplexen Systemen und unter risikoreichen Bedingungen eingesetzt. Mit der rasanten Leistungssteigerung im Bereich der *Consumer-PCs* waren nun auch aufwendige Berechnungen zur Simulation von verschiedensten Systemen möglich, ohne hierzu eine umfangreiche Infrastruktur schaffen zu müssen. Somit bildeten sich eine Vielzahl neuer Anwendungsfelder.

Besonders in der Robotik etablierten sich numerische Simulationen auf Basis von Systemmodellen. Zu den sehr umfangreichen und häufig hierzu genutzten wissenschaftlichen Werkzeugen zählt *MATLAB & Simulink* (siehe [159]). Jedoch eignet sich *MATLAB & Simulink* durch die bedingte Echtzeitfähigkeit für den in dieser Arbeit angestrebten Simulationsumfang nur eingeschränkt, weshalb es an dieser Stelle nicht weiterverfolgt wird. Durch immer leistungsfähigere und detailliertere Computerspiel-Engines entwickelten sich ebenfalls die Möglichkeiten von Simulationsumgebungen weiter. Sie kombinieren heute eine 3D-Visualisierungs-Engine sowie eine Physik-Engine miteinander. Infolgedessen ist es möglich, neben den numerischen Ergebnissen auch das physikalische Verhalten in einer virtuellen Umgebung zu analysieren. Aktuell zählen *V-REP* (siehe [57]), *Webots* (siehe [118]) sowie der an das *Robot Operating System* (ROS) (siehe [141, 145]) angegliederte *Gazebo* (siehe [92]) zu den bekanntesten Robotiksimulatoren. Mit Hilfe dieser Werkzeuge ist es möglich, dynamische Systeme in die virtuelle Realität zu überführen und dort zu analysieren. Darauf aufbauend ist der Gegenstand verschiedenster Arbeiten die Simulation von MAVs. So zeigen [181] beispielsweise die Simulation eines Prallluftschiffes und [30] die eines kleinen Coax-Helikopters auf Basis der Simulationsumgebung *Webots*. Durch die verbreitete Nutzung von Quadrokoptern rückt deren Simulation in den letzten Jahren ebenfalls immer mehr in den Fokus wissenschaftlicher Arbeiten. In [93] wird mit *SwarmSimX* die Schwarmsimulation von mehreren interagierenden Quadrokoptern auf Basis einer eigenen Simulationsumgebung gezeigt. Auch in dem Robotiksimulator *USARSim* (siehe [31]) ist

ein 3D-Modell des auch in dieser Arbeit genutzten *AR100B* Quadrokopters implementiert, wobei dieser lediglich mit einem rudimentären physikalischen Modell und ohne die Software des realen Autopiloten umgesetzt worden ist. *Open Source* Autopilotprojekte, beispielsweise der *ArduPilot* (siehe [11]) oder auch das *PX4*-Projekt (siehe [114, 138]), haben die Möglichkeiten der Simulation ebenfalls erkannt. Sie bieten Schnittstellen zu verschiedenen Simulations-Engines, wie beispielsweise *Flightgear* (siehe [55]), *jMAVSim* (siehe [86]), *JSBSim* (siehe [87]) oder *X-Plane* (siehe [178]), durch eine Hardware-Integration des Autopiloten (*engl. Hardware in the Loop* (HiL)) an. Bei diesen Simulationsumgebungen werden gleichermaßen nur rudimentäre Modelle der Sensoren, Antriebe und aerodynamischen Einflussgrößen genutzt.

Die in dieser Arbeit vorgestellte Simulation wurde dabei vor allem durch folgende Fragestellung inspiriert:

Wie ist eine effiziente sowie risikoarme Weiter- und Neuentwicklung dieser fliegenden Systeme unter reproduzierbaren Bedingungen möglich?

Dabei unterscheidet sich diese Arbeit besonders in einem wichtigen Aspekt von den vorherigen: Mit dieser Simulation wird es möglich sein, nicht nur vorhandene, schon fliegende Systeme in Echtzeit zu simulieren, sondern vollständig neue Plattformen und Algorithmen innerhalb der virtuellen Umgebung zu entwickeln und anschließend direkt auf die reale Plattform zu überführen. Ausschlaggebend ist hierbei die vollständige Integration der Software des genutzten Autopiloten in den Simulationskreis (*engl. Software in the Loop* (SiL)). Somit können nach abgeschlossener Optimierung und Validierung sämtliche Softwarekomponenten direkt aus der Simulation auf das reale MAV übertragen werden. Ebenfalls können, basierend auf den vorhandenen *Ground-Truth*-Daten, Fehler oder Divergenzen frühzeitig erkannt und eliminiert werden. Potenzielle Risiken, eine reale Flugplattform zu zerstören, werden somit maßgeblich minimiert. Einen ähnlichen Ansatz verfolgen die Arbeiten von [58] und [117]. Beide Publikationen stellen die Simulation eines Quadrokopters auf Basis der Simulationsumgebung *Gazebo* (siehe [92]) in Verbindung mit dem *Robot Operating System* vor. Im Vergleich zu den zuvor vorgestellten Arbeiten, behandeln diese Beiträge schon eine deutlich umfangreichere Betrachtung der physikalischen Modellierung des MAVs und nutzen ein Sensormodell, bei dem die realen Sensorfehler durch einen Bias und ein weißes, normalverteiltes und mittelwertfreies Rauschen modelliert werden. Die in dieser Arbeit vorgestellte Simulation[1] geht allerdings über den Umfang der bisherigen Arbeiten deutlich hinaus. Dabei steht im Besonderen die wesentlich detailliertere Modellierung, nicht nur die der flugplattformspezifischen Kenngrößen und

1 Erstveröffentlicht im Beitrag: *Modellbasierte 3D-Echtzeit-Simulation von micro-UAS* (2012) (siehe [100]).

Antriebe, sondern auch die der realen Sensoren und Sensorsysteme, im Vordergrund. So werden über die Modellierung von Bias und normalverteiltem Rauschen hinaus auch die Ausrichtungs- und Skalenfaktorfehler der Sensoren, welche einen signifikanten Einfluss auf die Modellierungsgenauigkeit haben, durch die *Misalignment*-Matrix berücksichtigt. Die Parameter dieser Sensormodelle werden dabei mittels präzise entwickelter Kalibriermethoden aus den realen Sensordaten ermittelt. Ebenfalls werden die realen Antriebe genauestens vermessen sowie ein detailliertes Bewegungsmodell erarbeitet. Dadurch werden neben einem nahezu identischen, aerodynamischen Verhalten der Flugplattform selbst, vor allem auch die Algorithmen zur Sensorfusion und Systemregelung mit realistischen Daten versorgt, wodurch eine vollwertige und reproduzierbare Entwicklung am virtuellen MAV erst möglich wird. Die Prozessierung der gesamten Simulation in Echtzeit ermöglicht ein realistisches, visuelles Feedback des Fluggerätes in seiner virtuellen Umgebung.

6.2 SYSTEMÜBERBLICK UND KONZEPT

Der grundlegende Gedanke dieser Simulation liegt in der realistischen virtuellen Abbildung des realen MAVs in Echtzeit. Um dieses Ziel zu erreichen, sind zum einen die Sensordaten realitätsnah nachzubilden und zum anderen die physikalischen Eigenschaften der Antriebe und der Plattform präzise zu modellieren.

Ausgangspunkt ist die kommerzielle Entwicklungsumgebung *Webots*[2] (siehe [118]). Aufgrund der Integration der *Open Source* Bibliotheken OGRE3D[3] zur grafischen Visualisierung und einer modifizierten Version von ODE[4] als Physik-Engine bietet diese Entwicklungsumgebung viele Möglichkeiten bei der Simulation von mobilen Robotern. Beispielsweise sind virtuelle, ideale Sensoren für die Beschleunigung, die Drehraten, die Ausrichtung und die Position in einer Basisversion verfügbar. Darüber hinaus erlaubt das SDK den direkten Zugriff auf die Programmierschnittstellen von ODE, um eigene physikalische Modelle zu implementieren. Die Integration von grafischen 3D-Modellen findet unter Verwendung der Beschreibungssprache VRML97[5] statt. Nicht zuletzt ermöglicht die vollständige Programmierung der verwendeten Bibliotheken sowie der in dieser Arbeit entwickelten Simulation in den Programmiersprachen C und C++ eine effiziente Implementierung. Kombiniert mit der Nutzung verschiedener Hardwarebeschleunigungen, wie zum Beispiel *OpenGL*, ergibt sich die angestrebte echtzeitfähige Simulation des Gesamtsystems. *Webots* besteht dabei aus den in Abbildung 17 auf Seite 70 (*blauer* Kasten *unten*) dar-

2 In dieser Arbeit wird Webots in der Version 6.4.2 verwendet.
3 *Object-Oriented Graphics Rendering Engine 3D* (OGRE3D).
4 *Open Dynamics Engine* (ODE).
5 *Virtual Reality Modeling Language 97* (VRML97).

gestellten drei Grundelementen: *virtuelle Welt*, *Simulations-Controller* und *Physik-Plug-In*.

VIRTUELLE WELT: Die *virtuelle Welt* umfasst die Definition und Darstellung aller Objekteigenschaften und eine vollständige geometrische Beschreibung des simulierten Roboters und dessen virtueller Umgebung. Dementsprechend wird das *Computer-Aided Design* (CAD)-Modell des MAVs in dieser implementiert. Das MAV-Modell wird zudem um die Definitionen der virtuellen Sensorprototypen und deren geometrische Position erweitert. Gespeichert werden diese Daten von *Webots* im *Worldfile*.

SIMULATIONS-CONTROLLER: Der *Simulations-Controller* bildet die Schnittstelle zwischen dem *Worldfile*, dem *Application Programming Interface* (API) von *Webots* sowie der eigentlichen Applikation. Die zuvor definierten Sensoren werden an dieser Stelle implementiert. Des Weiteren werden von diesem Programm alle Unterprogramme, so auch das spätere SiL-Interface, ausgeführt. Dazu wird die Hauptschleife des *Simulations-Controllers* in diskreten Zeitschritten aufgerufen und die spezifischen Funktionen, beispielsweise die Sensordatenakquisition, ausgeführt.

PHYSIK-PLUG-IN: Die letzte von *Webots* bereitgestellte Schnittstelle umfasst das *Physik-Plug-In*, welches die Schnittstelle zur ODE bildet. In diesem können die physikalischen Eigenschaften des Roboters definiert werden. So können hier zum Beispiel spezifische Kräfte oder Momente mit den geometrischen Objekten des *Worldfiles* verknüpft werden.

Eine vollständige Beschreibung der Entwicklungsumgebung *Webots* ist in [35] zu finden.

Auf Basis der im *Simulations-Controller* implementierten Sensoren erfolgt die eigentliche Simulation des Autopiloten. Damit der Autopilot bzw. dessen Filter- und Sensorfusionsalgorithmen jedoch realistische Sensordaten nutzen, müssen die idealen Werte aus den virtuellen Sensoren mit Hilfe entsprechender Sensormodelle transformiert werden (dargestellt in Abbildung 17 auf Seite 70 – *gelbe* Kästen). Parameter, wie beispielsweise Rauschcharakteristik, Nullpunkt- oder Skalierungsfehler, werden auf Basis realer Sensordaten abgeleitet, wie im weiteren Verlauf in Kapitel 8.1 gezeigt wird. Zur Interaktion des Autopiloten mit der Simulation wird ein eigenes hierzu entwickeltes Schnittstellenmodul implementiert. Dieses SiL-*Interface* ordnet allen autopilotinternen Ein- und Ausgabefunktionen deren simulationsspezifische Äquivalente zu (siehe Abbildung 17 – *grüner* Kasten). Dementsprechend werden über dieses Modul die modellierten Sensordaten an den Softwarekern des Autopiloten bzw. an das *Flight Management System* (FMS) übergeben.

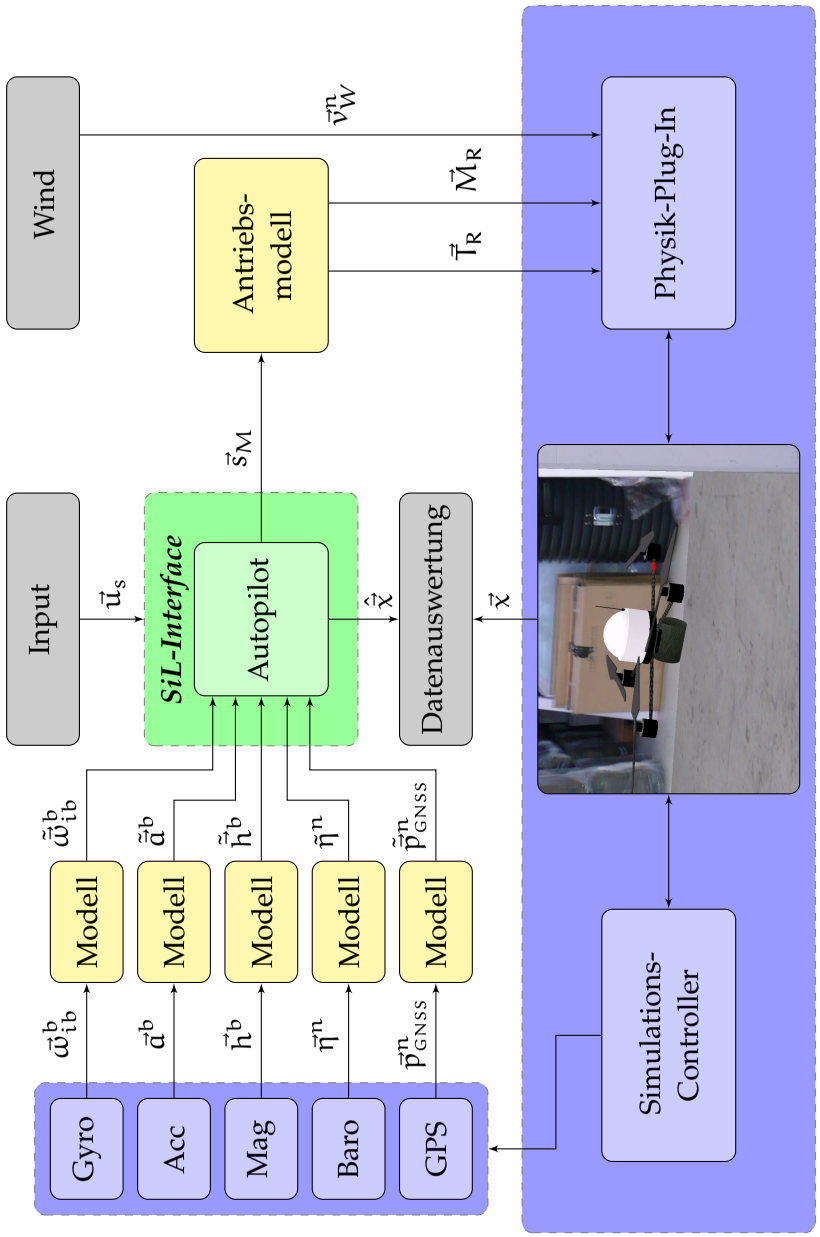


Abbildung 17: Konzept der modellbasierten 3D-Echtzeit-Simulation

Ebenfalls ist im SiL-*Interface* eine Schnittstelle zur pilotierten Steuerung des MAVs implementiert (siehe Abbildung 17 – *grauer* Kasten – *Input*). Über diese werden die Funktionen der Bodenkontrollstation durch ein handelsübliches Gamepad abgebildet. Mit den Steuerungseingaben durch den Piloten und den zuvor modellierten Sensordaten berechnet der Autopilot, analog zum realen Fluggerät, seine Systemzustandsschätzung $\hat{\vec{x}}$ und, entsprechend der internen Regelungsalgorithmen, die proportionalen Motorsteuergrößen $\vec{s}_M$ (siehe Abbildung 17). Diese Steuergrößen müssen mittels des in Kapitel 7.2.1 hergeleiteten und in Kapitel 8.1.2 bestimmten Antriebsmodells in die beiden entsprechenden physikalischen Größen Kraft und Drehmoment überführt werden. Die Länge des Kraftvektors $\vec{T}_R$ bzw. Momentenvektors $\vec{M}_R$ wird dabei durch die Anzahl der Antriebe bestimmt. Verarbeitet werden die physikalischen Größen der Antriebe vom *Physik-Plug-In*. Innerhalb dessen wird ein eigenständiger Controller implementiert, welcher das Bewegungsmodell und die spezifischen Kräfte und Momente über die entsprechenden ODE-Schnittstellen mit dem 3D-Objekt des MAVs verknüpft. Der externe Einfluss durch Wind kann zur Laufzeit der Simulation vom Benutzer in seiner Stärke variiert werden und fließt als Einflussgröße, definiert durch den Windgeschwindigkeitsvektor $\vec{v}_W^n$, in das Bewegungsmodell ein. Daraufhin berechnet *Webots* aus diesen Daten und dem hinterlegten physikalischen Modell der Flugplattform die dynamische Bewegung des MAVs und dementsprechend die neuen (idealen) Sensordaten für den nächsten Schleifenzyklus. In der *virtuellen Welt* erfolgt die grafische Visualisierung der Flugplattform innerhalb der erstellten virtuellen Umgebung. Besonders zielführend ist bei dieser entwickelten 3D-Echtzeit-Simulation, dass der tatsächliche Systemzustandsvektor $\vec{x}$ mit dem vom Autopiloten geschätzten $\hat{\vec{x}}$ direkt verglichen werden kann. Dazu wurde ein weiteres Modul zur Datenauswertung implementiert, welches den realen und den geschätzten Systemzustand synchronisiert speichert. Die wesentliche Innovation in diesem Konzept liegt allerdings in der SiL-Implementierung des Autopiloten. Dieser kann direkt und ohne Änderungen von der virtuellen in die reale Flugplattform überführt werden. Durch die realistische Modellierung aller Sensoren und physikalischen Eigenschaften können somit realitätsnahe Flugexperimente unter reproduzierbaren Bedingungen erfolgen. Dies erlaubt eine zeiteffiziente und risikoarme Arbeit mit diesen MAVs. Außerdem können durch die synchronisierten *Ground-Truth*-Daten unterschiedliche Algorithmen validiert, optimiert und sogar neu entwickelt werden. Grundlegende Voraussetzung ist hierzu die detaillierte und realitätsnahe Systembeschreibung und Modellierung aller Komponenten der Flugplattform, wie der Sensoren, Antriebe und der flugdynamischen Kenngrößen.

7 SYSTEMANALYSE UND MODELLIERUNG

Grundlegend zeigt diese Arbeit einen innovativen, simulationsbasierten Ansatz zur Entwicklung und Evaluierung unterschiedlicher VTOL-MAVs. Dabei ist zur erfolgreichen Umsetzung die Bestimmung eines präzisen Modells aller Sensoren, Antriebe und weiterer Einflussgrößen eine wesentliche Voraussetzung.

Nachdem in den vorherigen Kapiteln das elementare Funktionsprinzip der Flugplattform sowie der wesentliche Aufbau der Simulation und die Verarbeitung der Sensorsysteme diskutiert worden sind, werden im Folgenden die mathematischen Modelle entwickelt, welche zum einen die spezifischen Charakteristika der Sensoren und Messsysteme abbilden und zum anderen eine Beschreibung der mechanischen Struktur und der Antriebselemente wiedergeben.

7.1 SENSOREN UND MESSSYSTEME

Nach der in Kapitel 4 geführten Diskussion über die genutzten Sensoren und Messsysteme, soll im Folgenden eine mathematische Beschreibung der einzelnen Sensoren erfolgen. Hierzu wird ein Modell für jede Sensorgruppe aufgestellt, welches die spezifischen Fehler und Eigenschaften der Sensoren abbildet.

7.1.1 *Inertialsensorik*

Eine IMU basiert auf zwei unabhängigen Sensoreinheiten, eine für die Drehrate und eine für die Beschleunigung. Gyroskope erfassen dabei die Drehrate eines Körpers bezüglich einer spezifischen Drehachse. Allgemein kann der Zusammenhang zwischen der gemessenen Drehrate $\tilde{\vec{\omega}}_{ib}^{b}$ und der tatsächlichen Drehrate $\vec{\omega}_{ib}^{b}$ als Funktion gemäß

$$\tilde{\vec{\omega}}_{ib}^{b} = \mathbf{M}\vec{\omega}_{ib}^{b} + \vec{b}_{\omega} + \vec{n}_{\omega} \tag{7.1}$$

beschrieben werden. Ergänzend zu den Drehratensensoren bilden die Beschleunigungssensoren die zweite Komponente einer IMU. Grundsätzlich kann das Sensormodell von Beschleunigungssensoren analog zu dem des Gyroskopmodells, entsprechend der gemessenen Beschleunigung $\tilde{\vec{a}}^{b}$ und der tatsächlichen Beschleunigung $\vec{a}^{b}$, formuliert werden zu:

$$\tilde{\vec{a}}^{b} = \mathbf{M}\vec{a}^{b} + \vec{b}_{a} + \vec{n}_{a} \, . \tag{7.2}$$

Gemäß diesen Modellen setzt sich die Messgröße aus dem tatsächlichen Messwert und den nachstehenden Fehlertermen zusammen:

Ausrichtungs- und Skalenfaktorfehler der Sensorachsen (*engl. Misalignment* $\mathbf{M}$), Nullpunktfehler (*engl. Bias* $\vec{b}_\omega$) und Sensorrauschen (*engl. Noise* $\vec{n}_\omega$).

Ausrichtungs- und Skalenfaktorfehler

Die Sensor-*Misalignment*-Matrix $\mathbf{M}$ beschreibt die Winkelfehler der sensitiven Sensorachsen bezüglich des gewählten Referenzkoordinatensystems[1] sowie die Skalenfaktorfehler der Messgröße. Fertigungsbedingt sind im Allgemeinen die sensitiven Achsen des Drehratensensors bzw. des Beschleunigungssensors nicht absolut orthogonal zueinander. Diese Winkelfehler werden als systeminternes *Misalignment* bzw. systeminterner Winkelfehler ($\delta\vec{\Psi}_{sys}$) bezeichnet. Zudem ergibt sich ein weiterer integrationsbedingter Winkelfehler ($\delta\vec{\Psi}_{ref}$) bezüglich des Referenzkoordinatensystems. In der Regel ist der Gesamtfehler ($\delta\vec{\Psi}$) aus beiden Einzelfehlern von Interesse, welcher nach [104] abschätzbar ist durch:

$$\delta\vec{\Psi} = \sqrt{\delta\vec{\Psi}_{sys}^2 + \delta\vec{\Psi}_{ref}^2}\,. \tag{7.3}$$

Dieser Winkelfehlervektor wird durch die Nebendiagonalelemente ($M_{\square\square}$) der *Misalignment*-Matrix abgebildet. Die Hauptdiagonalelemente der Matrix spiegeln den Skalenfaktorfehler ($S_\square$) der einzelnen Sensorachsen wieder. Grundsätzlich werden diesem Term ebenfalls die Nichtlinearitäten des Sensors zugeschlagen, indem diese durch eine lineare Regression (*engl. Best Fit Straight Line*) approximiert werden. Mit diesen Elementen kann die 3×3-*Misalignment*-Matix $\mathbf{M}$ formuliert werden zu[2]:

$$\mathbf{M} = \begin{pmatrix} S_x & M_{xy} & M_{xz} \\ M_{yx} & S_y & M_{yz} \\ M_{zx} & M_{zy} & S_z \end{pmatrix}. \tag{7.4}$$

Nullpunktfehler

Die Nullpunktfehler (Bias) der Inertialsensorik werden durch den Vektor $\vec{b}_\square$ beschrieben. Allgemein setzt sich dieser aus einem *konstanten Bias*, einer *Bias-Drift* und einem Bias durch *thermische Effekte* zusammen. Im Fall der Gyroskope wirkt zusätzlich ein additiver Bias durch *beschleunigungsabhängige Effekte*.

1. *Konstanter Bias*: Nach jedem Einschalten des Sensors zeigt dieser einen Messfehler vom Nullpunkt. Eine zeitliche Integration

1 Im Rahmen dieser Arbeit wird das Referenzkoordinatensystem durch das körperfeste Koordinatensystem der Flugplattform gebildet.

2 Für die Herleitung der *Misalignment*-Matix $\mathbf{M}$ wird auf den Anhang B ab Seite 219 verwiesen.

der Drehrate mit einem solchen konstanten Fehler resultiert in einem linear ansteigenden Winkelfehler. In Analogie zum *Strapdown*-Algorithmus, in dem die Position durch zweifache Integration der Beschleunigung gebildet wird, wirkt sich ein Nullpunktfehler bei Beschleunigungssensoren zu einem quadratisch mit der Zeit anwachsenden Positionsfehler aus. Zur Laufzeit wird dieser Bias als konstant angenommen. Er kann sich jedoch bei jedem Systemneustart verändern. Daher wird dieser Fehler auch als *Turn-On-to-Turn-On Bias Stability* oder *Bias Repeatability* bezeichnet. Dementsprechend erfolgt die Angabe dieses Fehlerterms durch einen Wertebereich: bei Gyroskopen in $\pm\square°/s$ bzw. $\pm\square°/h$ und bei Beschleunigungssensoren[3] in $\pm\square\,m/s^2$.

2. *Bias-Drift*: Die *Bias-Drift* oder *Bias-Stabilität* beschreibt einen zeitlich variierenden Nullpunktfehler wegen des Funkelrauschens (*engl. Flicker-Noise*) der elektronischen Komponenten. Üblicherweise wird diese Bias-Drift auf Basis des $1/f$-Rauschens als *Random Walk* modelliert. Die Bias-Stabilität selbst wird in der Regel als 1σ Wert in $°/h$ bzw. m/s^2 angegeben und charakterisiert wie stark der Bias über eine vorgegebene Zeitspanne bei konstanten äußeren Umgebungsbedingungen variiert (siehe [177]). Diese Fehlergröße wird ebenfalls als *In-Run Bias Stability* bezeichnet.

3. *Thermische Effekte*: Der Bias der Inertialsensorik wird gleichfalls durch thermische Effekte beeinflusst. Diese Effekte können systemintern durch Selbsterwärmung der elektronischen Komponenten oder aufgrund variierender äußerer Umgebungseinflüsse induziert werden. Wegen des in der Regel nichtlinearen Zusammenhangs zwischen der Temperatur und dem Bias wird häufig ein weiterer Koeffizientenvektor ($\vec{c}_b$) in das Fehlermodell implementiert. Dieser wird nach [167] als Polynom dritten Grades bestimmt zu:

$$\vec{c}_b = \vec{c}_{b,0} + \vec{c}_{b,1}\Delta T + \vec{c}_{b,2}\Delta T^2 + \vec{c}_{b,3}\Delta T^3 \tag{7.5}$$

mit

$$\Delta T = [\text{Temperatur} - T_0]\,°C\,. \tag{7.6}$$

4. *Beschleunigungsabhängige Effekte*: Speziell bei Gyroskopen ist der beschleunigungsabhängige Effekt als eine weitere Einflussgröße auf den Nullpunktfehler anzuführen. Dieser, ebenfalls als *Linear Acceleration Effect* bezeichnete Fehler wird als Wert in $°/s/g$ bzw. $°/h/g$ angegeben. Teilweise ist zusätzlich zum linearen Term auch die quadratische Einflussfunktion mit $°/s/g^2$ bzw. $°/h/g^2$ angegeben (siehe [172]).

3 In der Literatur wird die Angabe von Fehlertermen bei Beschleunigungssensoren auch auf den Erdgravitationsvektor bezogen und somit in $mg = 10^{-3}\,g$ angegeben.

Sensorrauschen

Das sensorinhärente Rauschen fließt in die Modellierung durch den Term $\vec{n}_{\square}$ ein und wird als unkorreliert, weiß, normalverteilt und mittelwertfrei angenommen. Allgemein ist bei Drehratensensoren diese Rauschgröße durch den Terminus *Angular Random Walk* (ARW) beschrieben und in $^{\circ}/\sqrt{h}$ angegeben. Entsprechend des Beschleunigungssensors wird dieser Einflussterm als *Velocity Random Walk* (VRW) bezeichnet und in $^{m/s}/\sqrt{h}$ angegeben. Berechnet wird der ARW bzw. der VRW aus der Wurzel der spektralen Leistungsdichte des Rauschens. Gemäß Gleichung 7.7 kann, beispielhaft mit dem ARW, eine quantitative Aussage über die rauschbedingte Standardabweichung σ des Winkelfehlers Ψ_e getroffen werden. Somit wird die Drehrate eines ruhenden Gyroskops über den Zeitraum T integriert.

$$\sigma_{\Psi_e} = \text{ARW} \cdot \sqrt{T} \,. \tag{7.7}$$

Analog gilt selbiges mit dem VRW für die Standardabweichung des Geschwindigkeits- bzw. des Positionsfehlers. Zur Herleitung dieser Zusammenhänge sei auf [155], [172] und [177] verwiesen.

7.1.2 *Magnetometer*

Magnetometer werden üblicherweise zur Stützung des inertialen Navigationssystems und zur Bestimmung der absoluten Ausrichtung bezüglich Norden genutzt. Hierbei sind die Messwerte jedoch unterschiedlichen Fehlern unterworfen. Der Zusammenhang zwischen dem gemessenen Magnetfeld $\tilde{\vec{h}}^b$ und dem tatsächlichen Magnetfeld $\vec{h}^b$ kann als Funktion entsprechend

$$\tilde{\vec{h}}^b = \mathbf{M}\vec{h}^b + \vec{b}_h + \vec{n}_h \tag{7.8}$$

angegeben werden. Dabei erfasst das Magnetometer das lokale Erdmagnetfeld durch drei orthogonal zueinander angeordnete Sensorachsen. Gleichermaßen wie bei der Inertialsensorik sind diese – fertigungsbedingt – nicht ideal rechtwinklig zueinander bzw. zum Referenzkoordinatensystem angeordnet, was als *Misalignment* in den Nebendiagonalelementen der Matrix $\mathbf{M}$ berücksichtigt wird. Skalenfaktorfehler gehen über die Hauptdiagonalelemente in das Sensormodell ein.

Neben diesen internen Fehlern ist der Messwert jedoch besonders sensibel bezüglich äußerer Einflüsse, wie beispielsweise stromführende Leitungen oder ferromagnetische Materialien. Diese Einflussfaktoren werden als Hart- bzw. Weicheisen (*engl. Hard- and Soft-Iron*)-Effekte bezeichnet. Den Ausgangspunkt bildet ein ideales, nicht verzerrtes Magnetfeld, welches als Kugel mit dem Radius der absoluten Intensität $|\vec{h}^n| = 49.0089\,\mu T$ im Koordinatenursprung $(0,\, 0,\, 0)$ ent-

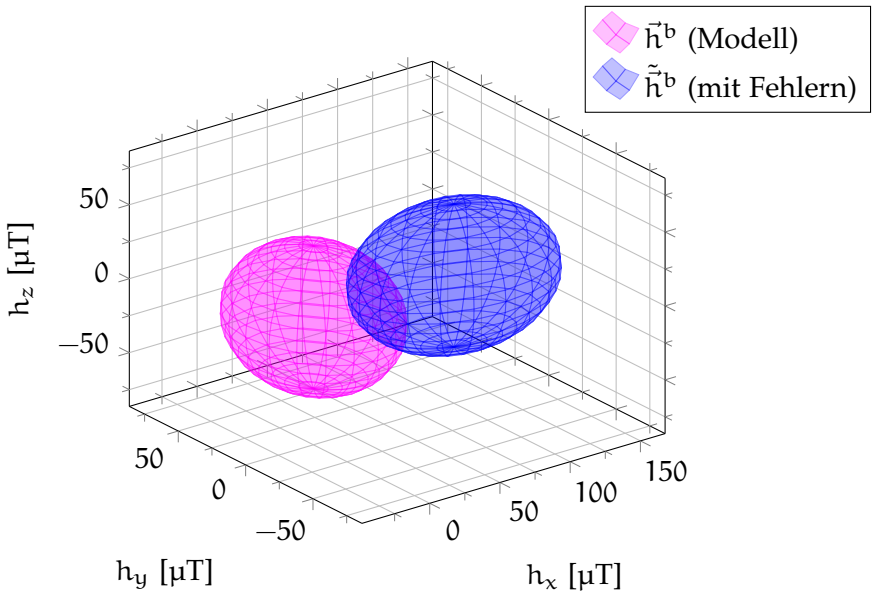


Abbildung 18: *Hard-* und *Soft-Iron*-Effekte bei Magnetometern

sprechend Abbildung 18 (*magenta* Kugel) dargestellt werden kann[4].

1. *Hard-Iron-Effekte*: Die *Hard-Iron*-Effekte erzeugen einen konstanten, additiven Wert zum lokalen Erdmagnetfeld und werden als Bias im Parameter $\vec{b}_h$ des Sensormodells erfasst. Die Folge ist eine Translation des Mittelpunktes der Kugel. Der modellierte Bias in Abbildung 18 (*blauer* Ellipsoid) beträgt:

$$\vec{b}_h = \begin{pmatrix} 100.0, & 0.0, & 0.0 \end{pmatrix}^T \mu T \, .$$

2. *Soft-Iron-Effekte*: Die *Soft-Iron*-Effekte verzerren die Messung des Erdmagnetfeldes selbst. Dadurch haben diese Effekte einen direkten Einfluss auf die sensitiven Achsen und können als Skalenfaktorfehler des Magnetometers modelliert werden. Somit werden diese der *Misalignment*-Matrix $\mathbf{M}$ zugeschlagen. Als Folge daraus ergibt sich eine Verzerrung der idealen Kugel zu einem Ellipsoid. Der modellierte Skalenfaktorfehler in Abbildung 18 (*blauer* Ellipsoid) ist:

$$\mathbf{M} = \begin{pmatrix} 1.4 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 1.0 \end{pmatrix} . \tag{7.9}$$

Ebenfalls zählen thermische Effekte zu den äußeren Einflüssen. Analog zur Darstellung bei der Inertialsensorik können diese über einen zusätzlichen Koeffizientenvektor in das Sensormodell einfließen.

4 $|\vec{h}^n| = 49.0089\,\mu T$ entspricht der absoluten Intensität des Erdmagnetfeldes in Arnsberg nach [123].

Sensorrauschen

Als letzter Term wird das sensorinhärente Rauschen in $\vec{n}_h$ modelliert. Es wird ebenfalls als unkorreliert, weiß, normalverteilt und mittelwertfrei angenommen.

7.1.3 *Baro-Altimeter*

Auf Grundlage des in Kapitel 4.3 hergeleiteten Zusammenhangs bezüglich Luftdruck und Höhe soll in diesem Abschnitt ein Sensormodell für das Baro-Altimeter eingeführt werden. Hierbei wird jedoch nicht der Luftdruck selbst, sondern die Höhe η^n modelliert. Dies ist vor allem anwendungsorientiert begründet, da der Navigationscomputer ebenfalls direkt die Höhe als Messwert verarbeitet. Dementsprechend wird das Sensormodell formuliert zu:

$$\tilde{\eta}^n = \eta^n + S(1 - e^{\frac{-t}{T}}) + b_\eta + n_\eta \; . \tag{7.10}$$

Die gemessene Höhe $\tilde{\eta}^n$ ergibt sich demnach als Funktion der tatsächlichen Höhe η^n, einem Nullpunktfehler b_η und dem Sensorrauschen n_η. Des Weiteren ist nach [34] mit dem Term $S(1 - e^{\frac{-t}{T}})$ der Höhenfehler durch den sensorinternen Erwärmungsprozess abgebildet. Mit der Zeitkonstante T des Erwärmungsprozesses konvergiert dieser gegen den einstellbaren Verstärkungsfaktor S. Wie in Kapitel 4.3 erläutert, ist die barometrische Höhenbestimmung stark äußeren Einflüssen unterworfen und somit nur über einen begrenzten Zeitraum bezüglich eines relativen Nullpunkts verlässlich. Die Höhendifferenz des relativen Nullpunkts zum absoluten Nullpunkt wird vorab vom gemessenen Wert abgezogen, womit sich η^n immer auf den relativen Nullpunkt bezieht. Das sensorinhärente Rauschen n_η wird ebenfalls an dieser Stelle als unkorreliert, weiß, normalverteilt und mittelwertfrei modelliert.

7.1.4 *GNSS*

Die Positionsbestimmung auf Basis eines globalen Navigationssatellitensystems (GNSS) ist, wie in Kapitel 4.4 veranschaulicht, ein sehr umfangreicher und komplexer Prozess. Fehler in der Navigationslösung hängen sowohl von verschiedenen systeminternen als auch von externen Einflussfaktoren ab. Eine quantitative Beschreibung der unterschiedlichen Fehlergrößen ist im Anhang A.2 auf Seite 217 zu finden. Eine derartige Modellierung wird aufgrund der Komplexität in dieser Arbeit nicht verfolgt. Stattdessen soll ein vereinfachtes Modell für die Positionsbestimmung in einem lokalen Koordinatensystem erstellt werden. Dies ist zielführend, da für die Verarbeitung in der multisensoriellen Navigationslösung die globale Position zuvor in ein lokales Koordinatensystem transformiert wird (siehe Abbildung 12 auf

Seite 35). Somit folgt das Modell für die *virtuell* gemessene Position im Navigationskoordinatensystem $\tilde{\vec{p}}^{n}_{GNSS}$ zu:

$$\tilde{\vec{p}}^{n}_{GNSS} = \vec{p}^{n}_{GNSS} + \vec{n}_{p} \ . \tag{7.11}$$

Die tatsächliche Position wird demzufolge mit einem unkorrelierten, weißen, normalverteilten und mittelwertfreien Rauschprozess überlagert.

7.2 VTOL-MAV-MODELL

Die Modellierung der aerodynamischen Kräfte und Momente von VTOL-MAVs ist ein sehr komplexer Bestandteil der Systemsimulation. Auf Basis der in Abschnitt 2.2 erörterten, grundlegenden Funktionsweise von Multirotorsystemen soll nun ein Bewegungsmodell geschaffen werden, welches das aerodynamische Verhalten der Flugplattform abbildet.

7.2.1 *Antriebsmodell*

Zur präzisen Modellierung müssen zuerst die einzelnen Antriebe genauer betrachtet werden. Grundsätzlich setzt sich hierbei jeder Antrieb aus einem Rotor mit einem festen Anstellwinkel (*engl. Fixed Pitch*) und einem zugehörigen bürstenlosen Gleichstrommotor (*engl. Brushless DC* (BLDC)*-Motor*) zusammen. Durch Änderung der Drehgeschwindigkeit ω_M kann entsprechend die Kraft und das Drehmoment variiert werden. Somit gilt es, einen funktionalen Zusammenhang zwischen Drehgeschwindigkeit und Schubkraft sowie Drehgeschwindigkeit und Drehmoment zu finden. Zur Modellierung wird das Antriebsverhalten der Flugplattform im Schwebezustand (*engl. Hover*) betrachtet.

Rotormodell

Gemäß der klassischen Impulstheorie (*engl. Momentum Theory*) wird der Rotor als gleichmäßig durchströmte aktive Scheibe (*engl. Actuator Disc*) angenommen (siehe [94]). Um nun eine Schubkraft (*engl. Thrust*) zu erzeugen, muss gemäß der Aerodynamik der Luftstrom durch die aktive Scheibe beschleunigt werden. Die Menge der durch den Rotor transportierten Luft pro Zeit, folgend als Massenstrom ($\dot{m}$) bezeichnet, berechnet sich als Funktion der Einströmungsgeschwindigkeit der Luft (v_i), der aktiven Fläche (A) und der Luftdichte (ρ) zu:

$$\dot{m} = \rho A v_i \ . \tag{7.12}$$

Wird von einem gleichförmigen Luftstrom ausgegangen, kann gezeigt werden, dass sich der Luftstrom hinter dem Rotor zusammenzieht und bis auf die maximale Geschwindigkeit w beschleunigt wird.

Entsprechend des Impulserhaltungssatzes ergibt sich die Schubkraft T_R aus dem Produkt von Massenstrom $\dot{m}$ und Geschwindigkeit zu:

$$T_R = \dot{m} w \, . \tag{7.13}$$

Aufgrund der Energieerhaltung muss zudem gelten, dass die abgegebene Leistung des Rotors gleich der erzeugten kinetischen Energie ist, woraus folgt:

$$T_R v_i = \frac{1}{2} \dot{m} w^2 . \tag{7.14}$$

Durch Einsetzen von Gleichung 7.13 in Gleichung 7.14 ergibt sich:

$$w = 2 v_i \, . \tag{7.15}$$

Somit kann durch Einsetzen von Gleichung 7.12 und 7.15 in Gleichung 7.13 die Schubkraft beschrieben werden durch:

$$T_R = \dot{m}(2v_i) = (\rho A v_i)(2v_i) = 2\rho A v_i^2 \, . \tag{7.16}$$

Zur Modellbildung ist es jedoch zielführend, die Schubkraft als Funktion der Winkelgeschwindigkeit des Rotors zu betrachten. Nach [94] kann die Einströmungsgeschwindigkeit (v_i) durch einen einheitenlosen Proportionalitätswert (*engl. Induced Inflow Ratio*) (λ_h), durch die Winkelgeschwindigkeit (ω_M) sowie den Radius des Rotors (R) definiert werden. Es gilt:

$$v_i = \lambda_h \omega_M R \, . \tag{7.17}$$

Durch Einsetzen in Gleichung 7.16 folgt:

$$\begin{aligned} T_R &= 2\rho A v_i^2 && (7.18)\\ &= 2\rho A (\lambda_h \omega_M R)^2 && (7.19)\\ &= 2\rho A (\lambda_h)^2 (\omega_M R)^2 . && (7.20) \end{aligned}$$

Ersetzt man nun das Produkt aus $2(\lambda_h)^2$ durch den Koeffizienten C_T der Rotorschubkraft (*engl. Rotor Thrust Coefficient*), dann folgt die aus der Literatur (siehe [94]) bekannte Formel für die Schubkraft:

$$T_R = C_T \rho A (\omega_M R)^2 . \tag{7.21}$$

Zu beachten ist, dass, entgegen der US-amerikanischen Schreibweise, im europäischen Raum zusätzlich der Faktor $1/2$ auf der rechten Seite hinzugefügt wird. Es folgt somit:

$$T_R = C_T \frac{1}{2} \rho A (\omega_M R)^2 . \tag{7.22}$$

Es zeigt sich, dass die Schubkraft T_R proportional zum Quadrat der Winkelgeschwindigkeit ω_M ist. Zur Modellierung wird der Parameter k_T eingeführt. Damit lautet das Modell für die Schubkraft:

$$T_R = k_T (\omega_M)^2 . \tag{7.23}$$

Zusätzlich zur Schubkraft erzeugt der Rotor durch den aerodynamischen Luftwiderstand der Rotorblätter ein Drehmoment (siehe [94]). Analog zur Schubkraft kann dieses ebenfalls mit einem Koeffizienten C_M für das Drehmoment (*engl. Rotor Shaft Torque Coefficient*) beschrieben werden:

$$M_R = C_M \frac{1}{2} \rho A (\omega_M R)^2 \cdot R \tag{7.24}$$

$$= C_M \frac{1}{2} \rho A R^3 \omega_M^2 \, . \tag{7.25}$$

Auch hier zeigt sich eine proportionale Abhängigkeit des Drehmoments zum Quadrat der Winkelgeschwindigkeit des Rotors. Somit folgt analog zur Schubkraft die Modellierung durch den Parameter k_M zu:

$$M_R = k_M (\omega_M)^2 . \tag{7.26}$$

Motormodell

Das physikalische Modell eines Gleichstrommotors kann allgemein durch die Ankerspannung U_A und die Drehratenänderung $\dot{\omega}_M$ bestimmt werden (siehe [91]). Es gilt:

$$U_A = R_A I_A + L_A \frac{dI_A}{dt} + k_1 \Phi_E \omega_M \tag{7.27}$$

$$\dot{\omega}_M = \frac{1}{J_A} \left(\underbrace{k_2 \Phi_E I_A}_{M_E} - M_L \right) . \tag{7.28}$$

Hierbei ist der Ankerwiderstand mit R_A, der zugeführte Ankerstrom mit I_A, die Induktivität der Ankerwicklung mit L_A, die Maschinenkonstanten mit k_1 und k_2, der elektromagnetische Fluss mit Φ_E und die Winkelgeschwindigkeit mit ω_M bezeichnet. Des Weiteren entspricht J_A dem Massenträgheitsmoment des Ankers. Gemäß der Gleichung 7.28 hängt die Winkelgeschwindigkeitsänderung direkt vom elektromagnetischen Drehmoment des Motors M_E und dem äußeren Lastmoment M_L ab.

Gilt die Voraussetzung, dass sich das System im quasistationären Zustand befindet (entspricht dem hier betrachteten *Hover*), ist die zeitliche Ableitung des Stromes $dI_A/dt = 0$. Zur besseren Lesbarkeit werden die Parameter $k_1 \Phi_E$ sowie $k_2 \Phi_E$ zu den spezifischen Motorkonstanten K_U und K_E zusammengefasst (siehe [73]). Demnach gilt:

$$U_A = R_A I_A + \underbrace{k_1 \Phi_E}_{K_U} \omega_M \tag{7.29}$$

$$M_E = \underbrace{k_2 \Phi_E}_{K_E} I_A \, . \tag{7.30}$$

Infolgedessen hängt das elektromagnetische Drehmoment direkt vom zugeführten Ankerstrom I_A ab. Wird die Gleichung 7.29 zu I_A umgestellt und in Gleichung 7.30 eingesetzt, folgt für das elektromagnetische Drehmoment:

$$M_E = \frac{K_E}{R_A}\left(U_A - K_U \omega_m\right). \tag{7.31}$$

Das Lastmoment M_L ergibt sich aus dem aerodynamischen Luftwiderstand des Rotors[5] (*engl. Drag Force*) und dem Rotorradius R. Dieses Moment wurde schon zuvor in Gleichung 7.24 bestimmt. Somit kann geschrieben werden:

$$M_L = M_R = C_M \frac{1}{2} \rho A R^3 \omega_M^2 \tag{7.32}$$

$$M_L = k_M (\omega_M)^2. \tag{7.33}$$

Eingesetzt in Gleichung 7.28 führt dies zu:

$$\dot{\omega}_M = \frac{1}{J_A}\left(\frac{K_E}{R_A}\left(U_A - K_U \omega_m\right) - k_M (\omega_M)^2\right). \tag{7.34}$$

Wie eingangs definiert, wird zur Modellierung der quasistationäre Zustand betrachtet. Dementsprechend ist $\dot{\omega}_M = 0$ und es folgt, dass

$$M_E = M_L \tag{7.35}$$

und somit

$$M_E = k_M (\omega_M)^2 \tag{7.36}$$

ist. Das elektromagnetische Drehmoment des Motors entspricht damit dem aerodynamischen Luftwiderstand der Rotorblätter. Das ist nur konsequent, da der Motor genau dieses Drehmoment aufbringen muss, um den Rotor zu drehen.

Zusammenfassend kann aufgrund von Gleichung 7.23 und Gleichung 7.36 das Antriebsmodell für die Schubkraft zu

$$T_R = k_T (\omega_M)^2 \tag{7.37}$$

und für das aerodynamische Drehmoment zu

$$M_R = k_M (\omega_M)^2 \tag{7.38}$$

bestimmt werden. Diese sind proportional zum Quadrat der Winkelgeschwindigkeit des Motors. Somit gilt es, zur Motormodellierung die Parameter k_T und k_M zu bestimmen.

5 Die Reibkräfte der Motorlager können vernachlässigt werden, da diese um ein Vielfaches kleiner sind als die Widerstandskraft des Rotors.

7.2.2 *Bewegungsmodell*

Im folgenden Abschnitt wird auf Grundlage des Antriebsmodells das Bewegungsmodell des VTOL-MAVs entwickelt. Hierzu gelten folgende Nebenbedingungen:

- Die Flugplattform ist starr (*engl. Rigid Body*).
- Die Flugplattform ist symmetrisch.

Zudem wurde zuvor für die Antriebe gezeigt:

- Schubkraft und Drehmoment sind proportional zum Quadrat der Winkelgeschwindigkeit des Motors bzw. Rotors.
- Die Rotoren sind starr – es gibt keine Schlaggelenke (*engl. Flapping Hinge*).

Ausgangspunkt für das Bewegungsmodell ist der Quadrokopter aus Abschnitt 2.3. Bei diesem System ist zu beachten, dass es zwar in X-Konfiguration gesteuert wird (siehe Abbildung 19 – *blauer* Pfeil), jedoch das köperfeste Koordinatensystem sowie die Antriebsanordnung einer +-Konfiguration entsprechen (siehe Abbildung 19). Der Hersteller des Systems mischt die Steuersignale des Piloten durch eine Transformation, bevor diese von der Flugplattform verarbeitet werden. Für das anschließend herzuleitende dynamische Modell bildet daher die eigentliche +-Konfiguration die Grundlage.

Translationsmodell

Das zweite Newtonsche Gesetz besagt, dass zur Änderung der Bewegung eines Körpers eine proportionale Kraft auf diesen wirken muss.

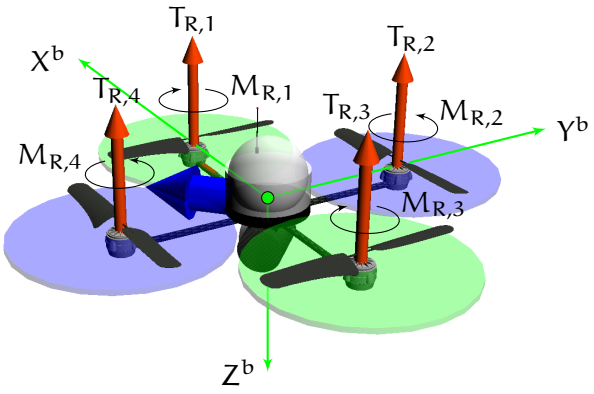


Abbildung 19: Quadrokoptermodell

Nach Euler gilt, dass diese Kraft (F) gleich dem Produkt aus Körpermasse (m) und Beschleunigung (a) ist. Somit ergibt sich:

$$F = ma \,. \tag{7.39}$$

Bezieht man dieses Aktionsprinzip auf die Flugplattform, folgt für den stationären Fall, dass die zugeführte Kraft ($\vec{F}^b_{R,\Sigma}$) durch die $N = 4$ Antriebe, transformiert in das Navigationskoordinatensystem, gleich der entgegenwirkenden Kraft ($\vec{F}^n_g$) aus Plattformmasse und Erdgravitation sein muss. Es gilt:

$$\vec{F}^n_g = -C^n_b \vec{F}^b_{R,\Sigma} \tag{7.40}$$

$$\vec{F}^n_g = -C^n_b \sum_{j=1}^{N} \vec{F}^b_{R,j} \tag{7.41}$$

$$m\vec{g}^n_l = -C^n_b \sum_{j=1}^{N} (0,\, 0,\, -T_{R,j})^T. \tag{7.42}$$

Stellt man entsprechend [172] die Bewegungsgleichung für das System auf, folgt:

$$\dot{\vec{v}}^n = \frac{1}{m} C^n_b \sum_{j=1}^{N} (0,\, 0,\, -T_{R,j})^T + \vec{g}^n_l. \tag{7.43}$$

Im dynamischen Fall wirkt zudem eine Kraft durch den Strömungswiderstand (*engl. Drag Force* F_D) entgegen der Bewegungsrichtung der Flugplattform. Diese ist proportional zum Quadrat der Geschwindigkeit v und kann allgemein beschrieben werden durch:

$$F_D = \frac{1}{2} \rho C_D A v^2. \tag{7.44}$$

Analog zur Antriebsmodellierung entspricht ρ der Luftdichte, A ist die wirkende Fläche und C_D ist der Strömungswiderstandskoeffizient. Betrachtet wird die wirkende Kraft durch den Strömungswiderstand im körperfesten Koordinatensystem der Flugplattform. Hierbei ist jedoch zu bedenken, dass sich richtungsabhängig wirkende Flächen ergeben. Durch die Symmetrie der Flugplattform wird angenommen, dass die horizontal wirkenden Flächen in x^b sowie in y^b Richtung gleich sind und durch A_{xy} definiert werden. Die vertikal wirkende Fläche in z^b-Richtung wird mit A_z bezeichnet. Zudem ergeben sich nach [115] ebenso richtungsabhängige Strömungswiderstandskoeffizienten, die analog der Flächen durch $C_{D,xy}$ sowie $C_{D,z}$ beschrieben werden. Es folgt:

$$\vec{F}^b_D = \frac{1}{2} \rho \vec{C}_D \vec{A} (\vec{v}^b)^2 \tag{7.45}$$

$$\vec{F}^b_D = \frac{1}{2} \rho \begin{pmatrix} C_{D,xy} A_{xy} \\ C_{D,xy} A_{xy} \\ C_{D,z} A_z \end{pmatrix} (\vec{v}^b)^2. \tag{7.46}$$

Da die Antriebskräfte im körperfesten Koordinatensystem wirken, soll eine Gesamtbetrachtung in eben diesem stattfinden. Die Summe aller auf die Flugplattform wirkenden Kräfte ergibt sich somit aus der gravitationsabhängigen Kraft $\vec{F}_g^b$, der Schubkraft der Antriebe $\vec{F}_{R,\Sigma}^b$ sowie aus der Strömungswiderstandskraft $\vec{F}_D^b$. Da jedoch der Betrag der Strömungswiderstandskraft durch $(\vec{v}^b)^2$ immer positiv ist, muss zusätzlich der Parametervektor $\vec{s}$ eingefügt werden, der die Wirkrichtung entsprechend dem Vorzeichen der Geschwindigkeit abbildet. Daraus ergibt sich:

$$\sum \vec{F}^b = \vec{F}_g^b + \vec{F}_{R,\Sigma}^b + \vec{s}\,\vec{F}_D^b \ . \tag{7.47}$$

Setzt man die hergeleiteten Einzelterme in Gleichung 7.47 ein, dann folgt:

$$\begin{aligned}\sum \vec{F}^b = C_n^b m \begin{pmatrix} 0 \\ 0 \\ g_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \sum_{j=1}^{N} -T_{R,j} \end{pmatrix} \\ + \vec{s}\,\frac{1}{2}\rho \begin{pmatrix} C_{D,xy} A_{xy} \\ C_{D,xy} A_{xy} \\ C_{D,z} A_z \end{pmatrix} (\vec{v}^b)^2 .\end{aligned} \tag{7.48}$$

Gemäß der Antriebsmodellierung kann mit Gleichung 7.37 auf Seite 82 die Schubkraft in Abhängigkeit von der Winkelgeschwindigkeit der Motoren angegeben werden. So kann das erweiterte Bewegungsmodell nach Gleichung 7.43 entwickelt werden zu:

$$\dot{\vec{v}}^n = \frac{1}{m} C_b^n \left(\begin{pmatrix} 0 \\ 0 \\ \sum_{j=1}^{N} -k_{T,j}\,\omega_{M,j}^2 \end{pmatrix} + \vec{s}\,\vec{F}_D^b \right) + \vec{g}_l^n . \tag{7.49}$$

Rotationsmodell

Die Herleitung des Rotationsmodells stützt sich auf die Ausführungen von [172]. Demnach kann das Drehmoment im Inertialkoordinatensystem $\vec{M}^i$ durch die zeitliche Ableitung des Drehimpulses $\vec{L}^i$ formuliert werden. Es gilt:

$$\vec{M}^i = \frac{\delta \vec{L}^i}{\delta t} \tag{7.50}$$

mit

$$\vec{L}^i = J^i \vec{\omega}_{ib}^i \ . \tag{7.51}$$

Der Drehimpuls kann selbst als Produkt aus Winkelgeschwindigkeit $\vec{\omega}_{ib}^{i}$ und Massenträgheitsmoment $\mathbf{J}^{i}$ bestimmt werden. Analog zum Translationsmodell ist es jedoch zielführend, die Momentenbetrachtung im körperfesten Koordinatensystem der Flugplattform durchzuführen, da ebenfalls die Trägheitsmomente der Flugplattform und die der Rotoren bezüglich diesem Koordinatensystem gegeben sind. Mit der Rotationsmatrix $\mathbf{C}_{b}^{i}$ gilt:

$$\vec{L}^{i} = \mathbf{C}_{b}^{i}\vec{L}^{b} \,. \tag{7.52}$$

Bezogen auf Gleichung 7.50 ist der Drehimpuls zeitlich abzuleiten. Es folgt:

$$\vec{M}^{i} = \dot{\mathbf{C}}_{b}^{i}\vec{L}^{b} + \mathbf{C}_{b}^{i}\dot{\vec{L}}^{b}. \tag{7.53}$$

Setzt man nach [172]

$$\dot{\mathbf{C}}_{b}^{i} = \mathbf{C}_{b}^{i}\left[\vec{\omega}_{ib}^{b}\times\right], \tag{7.54}$$

dann ergibt sich

$$\vec{M}^{i} = \mathbf{C}_{b}^{i}\left[\vec{\omega}_{ib}^{b}\times\right]\vec{L}^{b} + \mathbf{C}_{b}^{i}\dot{\vec{L}}^{b}. \tag{7.55}$$

Wird schließlich das Drehmoment in das köperfeste Koordinatensystem transformiert, erhält man:

$$\vec{M}^{b} = \vec{\omega}_{ib}^{b} \times \vec{L}^{b} + \dot{\vec{L}}^{b}. \tag{7.56}$$

Der Drehimpuls $\vec{L}^{b}$ setzt sich im Fall der Flugplattform aus zwei Komponenten zusammen. Zum einen besteht dieser aus dem Produkt aus Massenträgheitsmoment der Flugplattform $\mathbf{J}_{F}^{b}$ und deren Winkelgeschwindigkeit $\vec{\omega}_{ib}^{b}$. Dabei erfasst die Inertialmatrix $\mathbf{J}_{F}^{b}$ das gesamte Massenträgheitsmoment inklusive der Motoren und Rotoren. Zum anderen erzeugen die Antriebe einen weiteren Drehimpuls durch das Trägheitsmoment der einzelnen Rotoren $\mathbf{J}_{R}^{b}$ und deren Winkelgeschwindigkeiten $\vec{\omega}_{M,j}$. Somit ergibt sich:

$$\vec{L}^{b} = \mathbf{J}_{F}^{b}\vec{\omega}_{ib}^{b} + \sum_{j=1}^{N}\mathbf{J}_{R}^{b}\vec{\omega}_{M,j} \,. \tag{7.57}$$

Es wird dabei angenommen, dass es sich um gewuchtete, gleichwertige Rotoren mit identischem Massenträgheitsmoment $\mathbf{J}_{R}^{b}$ handelt. Setzt man dies in Gleichung 7.56 ein, erhält man:

$$\begin{aligned}\vec{M}^{b} = \quad & \vec{\omega}_{ib}^{b} \times \mathbf{J}_{F}^{b}\vec{\omega}_{ib}^{b} + \mathbf{J}_{F}^{b}\dot{\vec{\omega}}_{ib}^{b} \\ & + \vec{\omega}_{ib}^{b} \times \sum_{j=1}^{N}\mathbf{J}_{R}^{b}\vec{\omega}_{M,j} + \sum_{j=1}^{N}\mathbf{J}_{R}^{b}\dot{\vec{\omega}}_{M,j} \,.\end{aligned} \tag{7.58}$$

Wird Gleichung 7.58 nach $\dot{\vec{\omega}}^b_{ib}$ umgestellt, so folgt die Differentialgleichung für die Drehrate der Flugplattform $\vec{\omega}^b_{ib}$:

$$\dot{\vec{\omega}}^b_{ib} = \frac{1}{J^b_F}\left(\vec{M}^b - \vec{\omega}^b_{ib} \times J^b_F \vec{\omega}^b_{ib} - \vec{\omega}^b_{ib} \times \sum_{j=1}^{N} J^b_R \vec{\omega}_{M,j} - \sum_{j=1}^{N} J^b_R \dot{\vec{\omega}}_{M,j}\right). \tag{7.59}$$

Zuletzt müssen noch die externen Drehmomente $\vec{M}^b$ beschrieben werden. Diese ergeben sich durch die Antriebe und den Strömungswiderstand der Flugplattform zu:

$$\vec{M}^b = \vec{M}^b_{Antrieb} + \vec{M}^b_{Strömung} \,. \tag{7.60}$$

Aerodynamische Effekte, bedingt durch unterschiedliche Anströmungen der Rotoren, wirken sich gering gegenüber den Antriebsmomenten aus und können vernachlässigt werden. Der Strömungswiderstand der Flugplattform $\vec{F}^b_D$ (siehe Gleichung 7.45 auf Seite 84) greift jedoch in der Regel nicht exakt im Schwerpunkt der Flugplattform an und erzeugt somit ein Drehmoment. Der Abstand zum Schwerpunkt wird durch den Vektor $\vec{l}^b_D$ beschrieben. Das resultierende Drehmoment bestimmt sich nach [34] zu:

$$\vec{M}^b_{Strömung} = \vec{l}^b_D \times \vec{F}^b_D \,. \tag{7.61}$$

Aufgrund der einzelnen Rotorschubkräfte folgt der erste Teil der Antriebsdrehmomente $\vec{M}^{b\star}_{Antrieb}$ und wird durch die Summe der Kreuzprodukte aus der Länge des Motorarms $\vec{l}^b_{M,j}$ und der Schubkraft $\vec{F}^b_{R,j}$ bestimmt:

$$\vec{M}^{b\star}_{Antrieb} = \sum_{j=1}^{N} \vec{l}^b_{M,j} \times \vec{F}^b_{R,j} \,. \tag{7.62}$$

Hierbei gilt durch die Symmetrie der Flugplattform, dass die Länge aller Arme identisch ist

$$|\vec{l}^b_{M,j}| = l \tag{7.63}$$

und die Antriebswelle jeweils parallel zur z^b-Achse der Flugplattform verläuft:

$$\vec{F}^b_{R,j} = (0,\, 0,\, -T_{R,j})^T \,. \tag{7.64}$$

Somit erzeugen die Schubkräfte ausschließlich *Roll-* und *Pitch*-Momente.

$$\vec{M}^{b\star}_{Antrieb} = \begin{pmatrix} (T_{R,4} - T_{R,2})\, l \\ (T_{R,1} - T_{R,3})\, l \\ 0 \end{pmatrix}. \tag{7.65}$$

Bedingt durch den aerodynamischen Luftwiderstand der Rotoren, entsprechend der Herleitung in Gleichung 7.25 auf Seite 81, folgt ein weiteres Drehmoment bezüglich der z^b-Achse. Da die Drehrichtungen der Rotoren entgegengesetzt zueinander angeordnet sind, kompensieren sich zum Teil die Drehmomente der einzelnen Antriebe. Somit folgt für das gesamte Antriebsdrehmoment:

$$\vec{M}^b_{Antrieb} = \begin{pmatrix} (T_{R,4} - T_{R,2})\, l \\ (T_{R,1} - T_{R,3})\, l \\ \sum_{j=1}^{N} M_{R,j} \end{pmatrix}. \tag{7.66}$$

Gemäß der Antriebsmodellierung kann durch Gleichung 7.37 und Gleichung 7.38 auf der Seite 82 das Antriebsdrehmoment in Abhängigkeit der Winkelgeschwindigkeit der Motoren angegeben werden. Es folgt:

$$\vec{M}^b_{Antrieb} = \begin{pmatrix} \left(k_{T,4}\omega^2_{M,4} - k_{T,2}\omega^2_{M,2}\right) l \\ \left(k_{T,1}\omega^2_{M,1} - k_{T,3}\omega^2_{M,3}\right) l \\ \sum_{j=1}^{N} (-1)^{j-1} k_{M,j}\omega^2_{M,j} \end{pmatrix}. \tag{7.67}$$

7.3 ÄUSSERE EINFLUSSGRÖSSEN

Neben den bisherigen Systemeinflüssen sollen zusätzlich zwei äußere Einflüsse betrachtet werden. Erstens werden die hier betrachteten Flugplattformen sehr häufig im Außenbereich betrieben. Damit sind diese ebenfalls Wind ausgesetzt. Dieser soll gerade in Bezug auf die automatische Positions- und Wegpunktregelung in die Simulation mit einbezogen werden. Zweitens soll der Bodeneffekt betrachtet werden.

Windmodell

In dieser Arbeit wird ein statisches Windmodell implementiert. Das bedeutet, dass aufgrund des Strömungswiderstandes der Flugplattform sowie der Windgeschwindigkeit angreifende Kräfte simuliert werden. Dabei gilt das Modell

$$\vec{F}^b_W = \frac{1}{2}\rho \vec{C}_W \vec{A}(\mathbf{C}^b_n(\vec{v}^n_W)^2) \tag{7.68}$$

mit der auf die Flugplattform wirkenden Kraft $\vec{F}^b_W$, der richtungsabhängigen Fläche der Flugplattform $\vec{A}$ sowie der Windgeschwindigkeit $\vec{v}^n_W$. Hierbei wird, wie allgemein üblich, die Windgeschwindigkeit im Navigationskoordinatensystem vorgegeben (über Grund)

und entsprechend der Rotationsmatrix ins körperfeste Koordinatensystem der Flugplattform transformiert. So kann das Windmodell direkt in den Strömungswiderstand aus Gleichung 7.45 auf Seite 84 integriert werden. Mit der Fluggeschwindigkeit $\vec{v}^{n}$ relativ zum Boden (*engl. Groundspeed*) sowie der Windgeschwindigkeit $\vec{v}^{n}_{W}$ erweitert sich $\vec{F}^{b}_{D}$ zu:

$$\vec{F}^{b}_{D} = \frac{1}{2}\rho \vec{C}_{D} \vec{A} \left(\mathbf{C}^{b}_{n} (\vec{v}^{n} - \vec{v}^{n}_{W})^{2} \right). \tag{7.69}$$

Bodeneffekt

Der Bodeneffekt (*engl. In Ground Effect* (IGE)) beschreibt einen Effekt, der in Bodennähe auf die Flugplattform wirkt. Hierbei trifft die durch die Rotoren beschleunigte Luft (*engl. Downwash*) auf den Boden. Dieser verhindert ein schnelles Abströmen und sorgt für zusätzlichen Auftrieb. Dementsprechend benötigen die Antriebe weniger Leistung, um den geforderten Gesamtauftrieb zu erzeugen. In [22] ist der Bodeneffekt eines ähnlichen Quadrokopters untersucht worden. Hier stellte man experimentell fest, dass dieser lediglich bis zu einer Höhe vom Rotordurchmesser (2R) einen nachweislichen Einfluss hat. Da die Rotordurchmesser der in dieser Arbeit simulierten Flugplattformen ($2R < 40\,\text{cm}$) sehr gering sind, tritt dieser Effekt ausschließlich nur sehr kurzzeitig beim Starten und Landen auf und kann daher vernachlässigt werden.

7.4 ZUSAMMENFASSUNG UND DISKUSSION

In diesem Kapitel sind die Kenngrößen sowie Modellierungsparameter für die Sensoren, die Antriebe und die mechanische Struktur der Flugplattform hergeleitet worden. Auf Basis dieser Ergebnisse sollen nun anhand realer Sensoren, Antriebe und CAD-Modelle die spezifischen Parametervektoren bestimmt werden. Auf Grundlage des translatorischen und rotatorischen Bewegungsmodells des MAVs kann dieses in die virtuelle Realität überführt werden.

8 SIMULATION DES AR100B

Nachdem im vorherigen Kapitel die mathematischen Modelle der Sensoren, der Antriebe sowie die flugdynamischen Kenngrößen hergeleitet worden sind, wird im Folgenden die Ausgangsplattform in die virtuelle Realität überführt. Hierzu werden zuerst alle Modellierungsparameter anhand der realen Komponenten bestimmt. Daraufhin wird der gesamte Autopilot in das SiL-Interface integriert und ein Vergleich zwischen realem und simuliertem System vorgenommen. Mit der erfolgreichen Integration der Flugplattform in die virtuelle Welt sind dann eine Vielzahl von Optimierungen bzw. Erweiterungen am Autopiloten und dessen Regler- und Filteralgorithmen innerhalb der virtuellen Welt möglich. Zum Ende des Kapitels wird gezeigt, wie der vollständige Autopilot des *AR100B* auf Grundlage dieses simulationsbasierten Entwicklungsframeworks effizient, präzise und sicher auf neue VTOL-MAVs erweitert wird.

8.1 MODELLIERUNG DER FLUGPLATTFORM

Bevor jedoch mit der virtuellen (Weiter-) Entwicklung der Flugplattform begonnen werden kann, müssen zuvor präzise Modelle aller Komponenten bestimmt werden. Dabei ist die Genauigkeit aller Modellparameter von großer Bedeutung. Analog zur Herleitung in Kapitel 7 wird mit den Sensoren der Flugplattform begonnen.

8.1.1 *Sensoren*

Drehratensensoren

Das 3-achsige Gyroskop erfasst die Drehraten bezogen auf die drei Raumachsen im körperfesten Koordinatensystem. Zur Wiederholung aus Kapitel 7.1.1 kann das Modell des Sensors durch

$$\tilde{\vec{\omega}}_{ib}^{b} = \mathbf{M}\vec{\omega}_{ib}^{b} + \vec{b}_{\omega} + \vec{n}_{\omega} \tag{8.1}$$

beschrieben werden. Ausgangspunkt der Modellierung bildet das bereits im Werk kalibrierte Flugboard des *AR100B*, auf dem die IMU und eine Recheneinheit zur Flugregelung integriert sind.

Um die Parameter für die Ausrichtungs- und Skalenfaktorfehler der Sensorachsen (*Misalignment* $\mathbf{M}$), die Nullpunktfehler (*Bias* $\vec{b}_{\omega}$) und das Sensorrauschen (*Noise* $\vec{n}_{\omega}$) zu finden, muss zunächst ein Teststand entwickelt werden, der eine deutlich höhere Präzision aufweist als die schon werkseitig durchgeführten Sensorkalibrierungen.

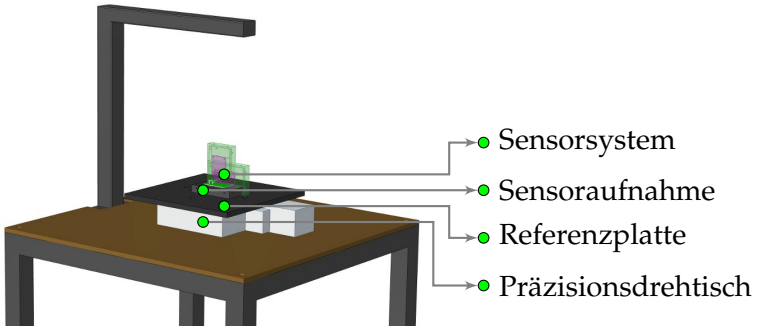


Abbildung 20: Aufbau für die Kalibrierung der Drehratensensoren

Hierzu wird auf der vorhergegangenen Masterarbeit (siehe [97]) aufgebaut, in der ein Präzisionsdrehtisch zur Vermessung eines 1-achsigen Gyroskops genutzt wurde. Der dort gezeigte Aufbau wird zu diesem Zweck weiterentwickelt (siehe Abbildung 20), um nun ein 3-achsiges Sensorsystem zu vermessen. Den Ausgangspunkt bildet auch in dieser Arbeit der Präzisionsdrehtisch *x-Act RT170ST* der Firma Linos (siehe [139]), welcher über eine mitgelieferte Interfacekarte (Schrittmotorkarte *M50 c.PCI*) mit einem PC verbunden wird. Um den hohen Anforderungen der Modellbildung gerecht zu werden, ist im Rahmen dieser Arbeit eine Software in C++ entwickelt worden, die den Messablauf synchronisiert steuert. Über diese Software können dem Drehtisch spezifische Drehraten und Drehratenbeschleunigungen vorgegeben werden. Außerdem zeichnet die Modellierungssoftware synchron die Daten des Drehratensensors und die Referenztrajektorie auf. Die Genauigkeit des Präzisionsdrehtisches wird vom Hersteller mit einer Auflösung von 0.001°, einer absoluten Positionierbarkeit von ±0.05° und einer reproduzierbaren Positionierbarkeit von ±0.01° angegeben. Zudem ist die Referenzplatte mit einer Ausrichtungsgenauigkeit von ±0.05° eingemessen. Wie durch die späteren Messergebnisse deutlich wird, ist die mit diesem Aufbau erreichte Genauigkeit um eine Zehnerpotenz besser als die werkseitige, manuell durchgeführte Kalibrierung. Damit die Parameter für Ausrichtungs- und Skalenfaktorfehler der Sensorachsen bestimmt werden können,

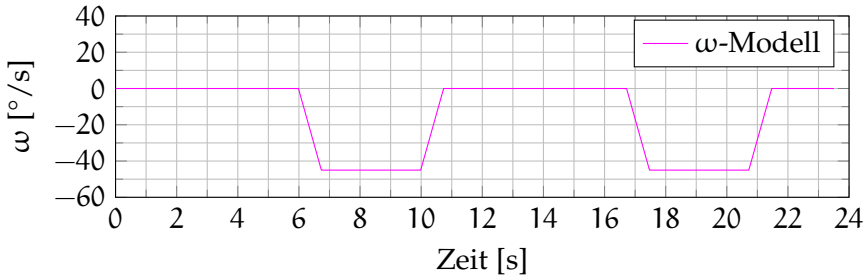


Abbildung 21: Referenzdatenmodell für die Kalibrierung der Drehratensensoren

wird jede Sensorachse gemäß des hierzu entwickelten Referenzdatenmodells (siehe Abbildung 21) vermessen. Hierzu rotiert der Drehtisch zweimal um 180° mit jeweils einer Drehgeschwindigkeit von −45°/s. Die Drehratenbeschleunigung wird zu $60°/s^2$ gewählt. Diese Werte entsprechen etwa der mittleren Dynamik der Flugplattform. Exemplarisch werden im Folgenden die Sensordaten bei einer Drehung um die körperfeste z-Achse untersucht. Zur vollständigen Sensormodellierung werden folgerichtig die Sensorfehler ebenfalls durch Drehung um die körperfeste x- sowie y-Achse bestimmt. Anschließend werden die aufgenommenen Sensordaten den Referenzdaten aus der Modelltrajektorie gegenübergestellt (siehe Abbildung 22). Hier zeigt sich, dass der Drehratensensor spezifische Fehler aufweist. Zur Verdeutlichung sind in Abbildung 23 auf Seite 94 die Fehler der jeweiligen Achse aufgetragen. Dementsprechend ist jede Sensorachse mit einem *Offset* behaftet. Dieser Bias kann durch die in Anhang B.1 ab

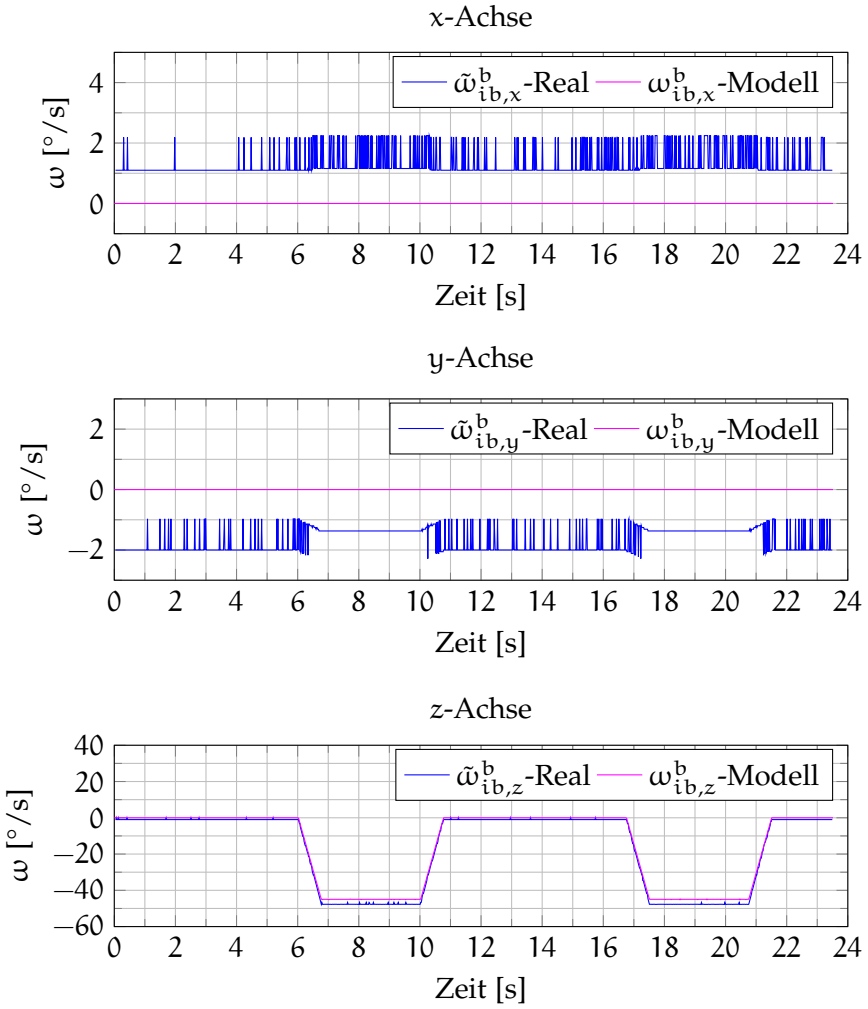


Abbildung 22: Gegenüberstellung der gemessenen Drehraten (Real) zu den Referenzdrehraten (Modell)

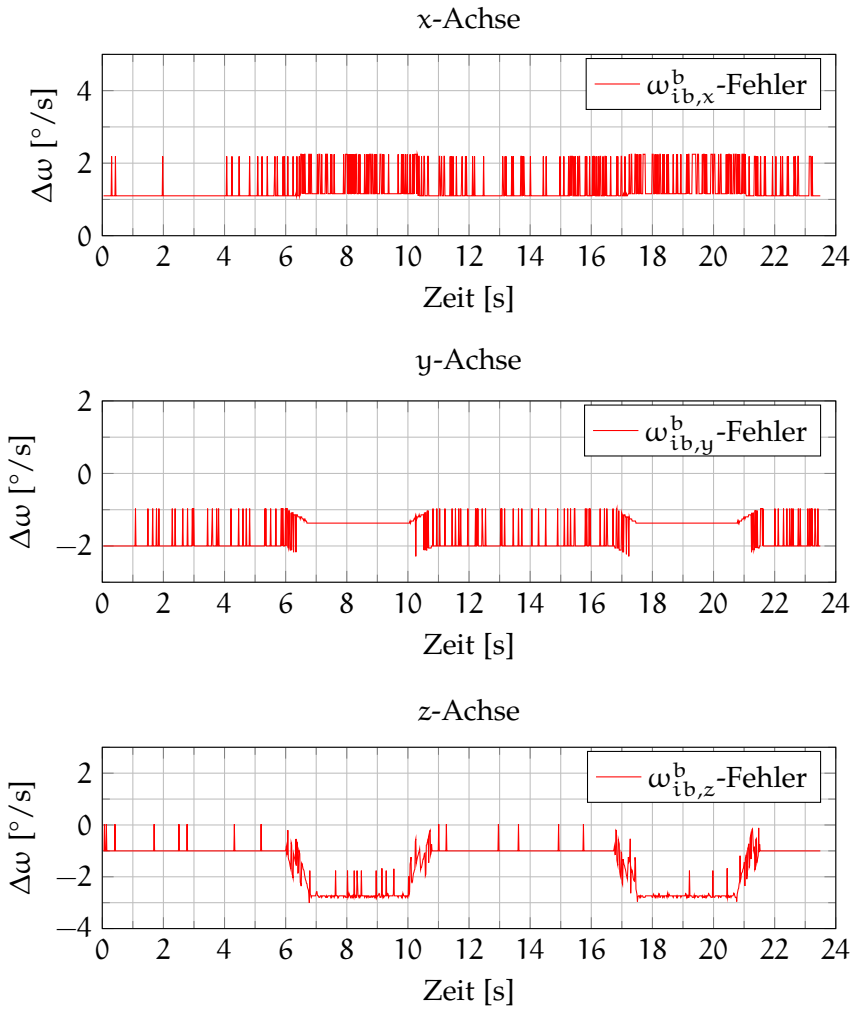


Abbildung 23: Drehratensensorfehler

Seite 219 beschriebene Sensorkalibrierung berechnet werden. Hierzu wird innerhalb der ersten 5.5 Sekunden der Mittelwert bestimmt, da sich hier das System in Ruhelage befindet. Es ergibt sich:

$$\vec{b}_\omega = \begin{pmatrix} 1.13 \\ -1.93 \\ -0.98 \end{pmatrix} °/s. \tag{8.2}$$

Des Weiteren unterliegt die z-Achse einem Skalierungsfehler, da die Sensorachse während der Drehbewegung eine konstant höhere Drehrate misst. Bedingt durch einen systeminternen Ausrichtungsfehler (*Misalignment*) der Sensorachsen ist ein Einfluss der Drehbewegung um die z-Achse ebenso als Einflussgröße in der x- sowie y-Achse ersichtlich. Wird gemäß der Herleitung in Anhang B.1 die Referenztrajektorie auf alle drei körperfesten Achsen angewandt, folgen die drei Skalenfaktorfehler der Sensorachsen entsprechend der Hauptdiagonalelemente der *Misalignment*-Matrix. Die Winkelfehler der Sensor-

achsen zueinander werden durch die Nebendiagonalelemente abgebildet. Zusammengefasst ergibt sich:

$$\mathbf{M} = \begin{pmatrix} 1.038110 & 0.000306 & -0.001273 \\ 0.014431 & 1.043070 & 0.008223 \\ -0.013033 & 0.009030 & 1.038440 \end{pmatrix} . \tag{8.3}$$

Um zuletzt das Rauschen des Sensors zu bestimmen, werden die mittels der neuen und deutlich genaueren Sensorkalibrierung ermittelten Parameter invers auf die Sensordaten angewendet und nochmals die Differenz zu den *Ground-Truth*-Daten der Modelltrajektorie gebildet. Aus Abbildung 24 wird ersichtlich, dass die ermittelten Parameter sehr genau den Fehler des Sensors bestimmen. Der mittlere Fehler (*engl. Root Mean Squared Error* (RMSE)) der Sensorachsen liegt bei:

$$\Delta\vec{\omega}^{b}_{ib,\mathrm{RMSE}} = \begin{pmatrix} 0.430, & 0.293, & 0.251 \end{pmatrix}^{T} {}^{\circ}/\mathrm{s} . \tag{8.4}$$

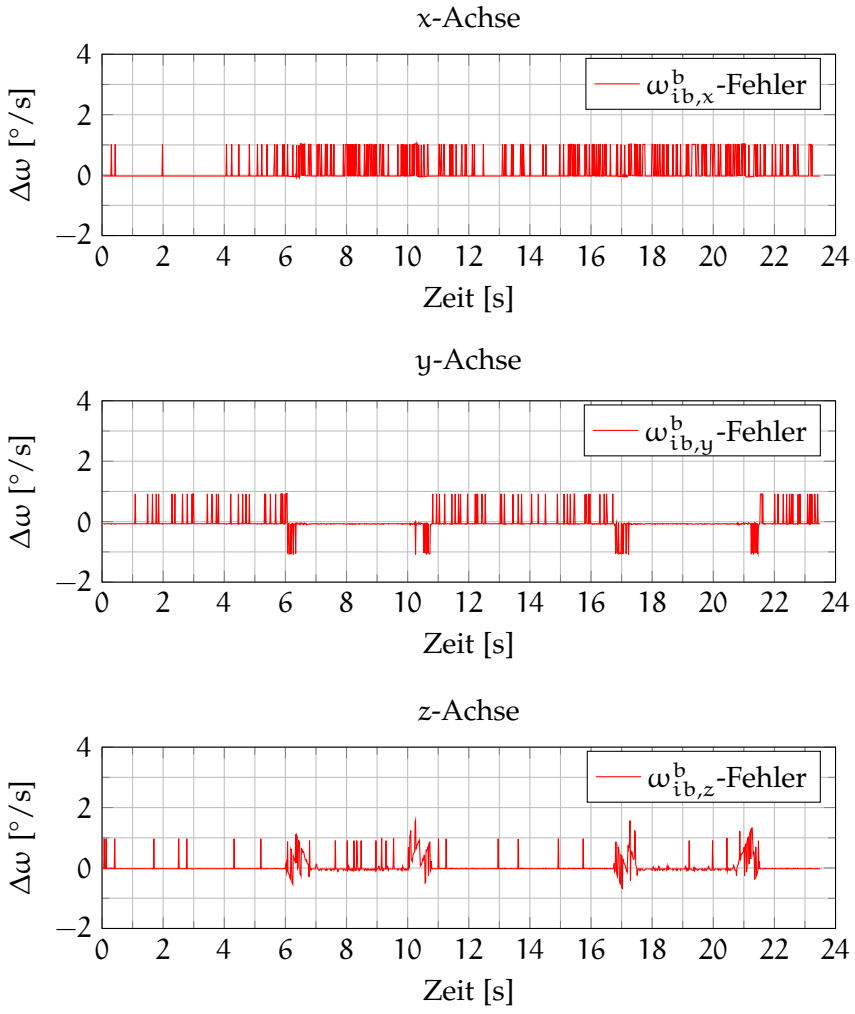


Abbildung 24: Drehratensensorfehler nach der Kalibrierung

Die Differenz zum Referenzwert wird somit vorwiegend durch das Sensorrauschen bestimmt. Bei einer detaillierteren Betrachtung von Abbildung 24 ist jedoch auffallend, dass das Rauschen sehr diskret ausgeprägt ist. Erklärbar ist dies durch die Quantisierung des Sensormesswertes. Bei dem verwendeten Drehratensensor wird die gemessene Drehrate als proportionale Spannung mit $2\,^{\mathrm{mV}/\circ/\mathrm{s}}$ (siehe [81, 82]) angegeben. Der integrierte *Analog-to-Digital-Converter* (ADC) des genutzten Mikrocontrollers (siehe [129]) hat eine Auflösung von 10 bit bei einer Referenzspannung von 3.3 V. Zusätzlich ist eine analoge Verstärkerstufe mit 6 dB zwischen Sensor und Mikrocontroller geschaltet. Somit folgt für die Quantisierung durch den ADC:

$$1\,\mathrm{LSB} = \frac{3.3\,\mathrm{V}}{2^{10}} = 3.223\,\mathrm{mV}. \tag{8.5}$$

Mit dem Verhältnis

$$1^\circ/\mathrm{s} = 2\,\mathrm{mV} \tag{8.6}$$

und dem Verstärkungsfaktor ($6\,\mathrm{dB} \approx 2$) folgt somit:

$$1\,\mathrm{LSB} = \frac{3.223\,\mathrm{mV}}{2 \cdot 2\,\frac{\mathrm{mV}}{^\circ/\mathrm{s}}} \approx 0.805^\circ/\mathrm{s}\,. \tag{8.7}$$

Dementsprechend können Drehraten maximal mit einer theoretisch möglichen Auflösung von $0.805^\circ/\mathrm{s}$ erfasst werden. Praktisch ist die Auflösung, begründet durch Bauteiltoleranzen etc., noch etwas geringer und beträgt in der x-Achse $1.10^\circ/\mathrm{s}$ und in der y- sowie z-Achse $1.00^\circ/\mathrm{s}$. Wird diese Erkenntnis auf den Nullpunktfehler übertragen, beträgt dieser auf der x-Achse genau $+1$ *Least Significant Bit* (LSB), auf der y-Achse -2 LSB und auf der z-Achse genau -1 LSB. Somit folgt der quantisierte Bias zu:

$$\vec{b}_\omega = \begin{pmatrix} 1.10 \\ -2.00 \\ -1.00 \end{pmatrix} ^\circ/\mathrm{s}\,. \tag{8.8}$$

Daraus ableitend werden für die Rauschbetrachtung die Sensordaten korrigiert und sind in Abbildung 25 *links* dargestellt. Zur besseren Lesbarkeit und um bei der Rauschanalyse Interpolationsfehler bei der Drehratenbeschleunigung zu vermeiden, wird ausschließlich die Ruhelage des Systems betrachtet. Hierbei ist auffällig, dass das Sensorrauschen positiv priorisiert ist. Um dies zu bestätigen, wird eine Histogrammanalyse durchgeführt. Hierbei werden die Fehler der Abtastungen (*engl. Samples*) gezählt und nach Werten sortiert. Dies ist in Abbildung 25 *rechts* dargestellt. Auf der y-Achse ist die auf 1 normierte Anzahl der *Samples* entsprechend ihrer Wahrscheinlichkeit P dargestellt. Auf der x-Achse ist der Drehratenfehler abgebildet. Hierbei wird deutlich, dass das Rauschen genau 1 LSB der ADC-Auflösung

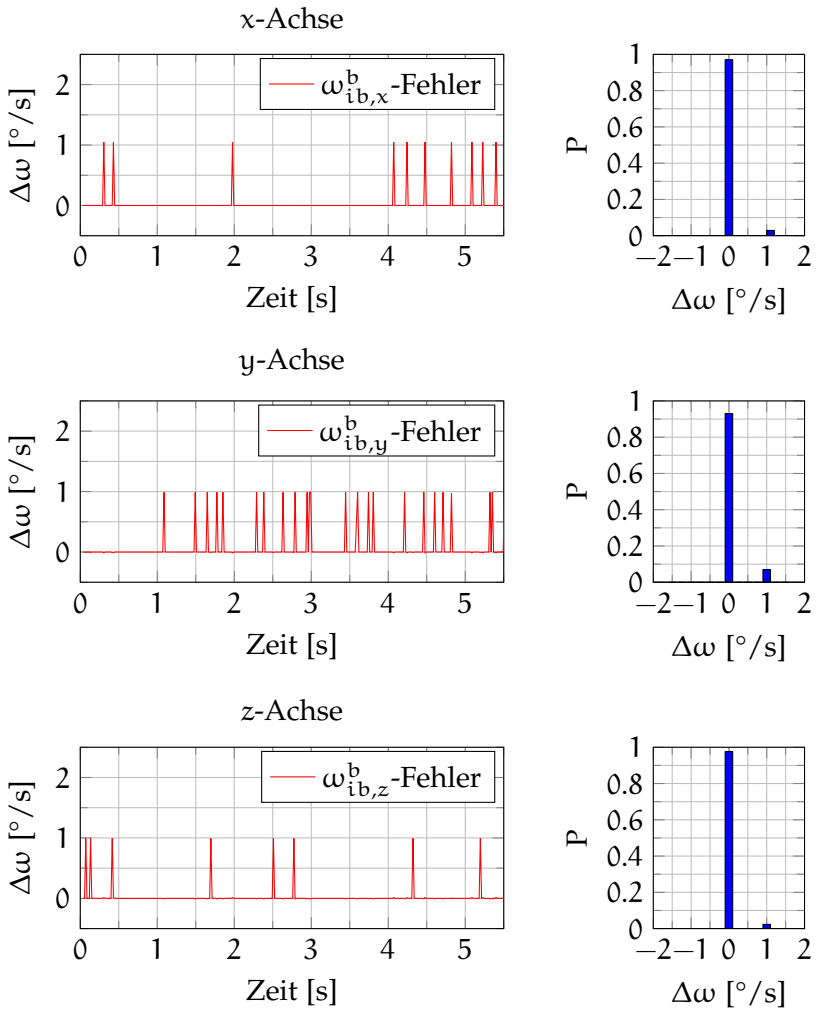


Abbildung 25: Sensorfehler (*links*) und Histogramm der Rauschverteilung (*rechts*) nach der Kalibrierung des Drehratensensors

entspricht. Dies ist nur konsequent, da der mittlere Fehler des eigentlichen Sensorrauschens mit 0.5 mV (RMSE) vom Hersteller des Drehratensensors (siehe [82]) angegeben wird. Dieser Wert liegt deutlich unter dem auflösbaren Bereich des verwendeten ADCs. Dieser Aspekt ist für die Sensormodellierung von entscheidender Bedeutung, da nicht direkt, wie in Abschnitt 7.1.1 dargestellt, von einem normalverteilten Rauschmodell ausgegangen werden kann.

Abschließend muss noch ein weiterer Aspekt betrachtet werden. In der Verarbeitungskette des analogen Sensorsignals ist zusätzlich ein Tiefpassfilter integriert. Dieser dient vorwiegend zur Filterung des hochfrequenten Rauschanteils, welcher durch Vibrationen der Aktoren in das System induziert wird. Diese Antriebsvibrationen werden, wie in der Antriebsmodellierung später erläutert, in der Simulation des *AR100B* nicht berücksichtigt. Ebenfalls ist die Grenzfrequenz des Filters, ausgehend von der Flugplattformdynamik des *AR100B*, vernachlässigbar und wird in diesem Fall bei der Sensormodellierung

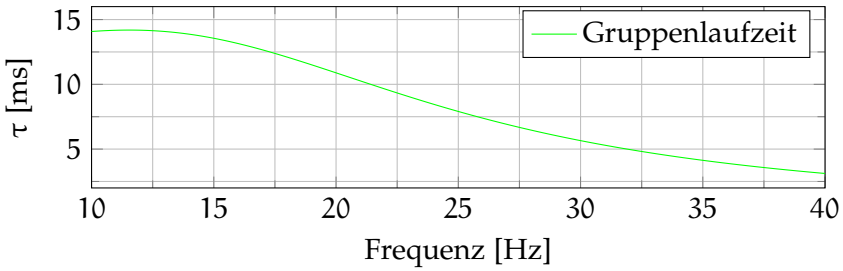


Abbildung 26: Gruppenlaufzeit des Tiefpassfilters des Drehratensensors

nicht eingehender betrachtet. Die Gruppenlaufzeit (τ) des Filters hingegen ist zwingend mit zu modellieren, da sich durch diese ein zeitlicher Versatz zwischen den unterschiedlichen Sensoren ergibt. Hierzu wurde der analoge Filter mit dem Simulationsprogramm für elektronische Schaltungen *PSpice*[1] (siehe [28]) modelliert und entsprechend die Gruppenlaufzeit bestimmt (siehe Abbildung 26). Somit sind nun sämtliche Sensorparameter bekannt und müssen als nächstes modelliert werden.

Die zuvor dargestellte Gruppenlaufzeit wird als PT_1-Glied implementiert:

$$\vec{\omega}^{b}_{ib,k(\tau)} = \frac{1}{T_v + 1}\left(\vec{\omega}^{b}_{ib,k} - \vec{\omega}^{b}_{ib,k-1}\right) + \vec{\omega}^{b}_{ib,k-1} \tag{8.9}$$

mit

$$T_v = \frac{\tau}{\Delta t}\,. \tag{8.10}$$

Die Abtastrate des Sensorsystems Δt entspricht genau der Updaterate des Autopiloten mit 125 Hz ($\Delta t = 8\,\text{ms}$). Die Gruppenlaufzeit τ wird mit dem Maximalwert von 14 ms umgesetzt. Die somit verzögerten, aber sonst idealen Sensordaten werden, wie in Gleichung 8.7 auf Seite 96 hergeleitet, jedoch mit der realen Quantisierungsschrittweite quantisiert. Im Anschluss werden diese, wie in Gleichung 8.1 auf Seite 91 gezeigt, mit der *Misalignment*-Matrix aus Gleichung 8.3 auf Seite 95 multipliziert. Daraufhin wird der in Gleichung 8.8 auf Seite 96 bestimmte Bias addiert. Abschließend wird noch der Rauschterm $\vec{n}_\omega$ modelliert. Wie zuvor dargestellt, ist durch das geringe Auflösungsverhältnis zwischen ADC des Mikrocontrollers und des Analogwertes des Drehratensensors das Rauschen nicht normalverteilt. Jedoch kann angenommen werden, dass das Sensorrauschen selbst normalverteilt, mit der Standardabweichung σ_ω, ist. Laut der Herstellerangaben (siehe [82]) beträgt das Rauschen 0.25°/s (RMSE). Somit folgt:

$$\vec{\sigma}_\omega = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \end{pmatrix} °/s\,. \tag{8.11}$$

1 *Simulation Program with Integrated Circuit Emphasis* (Spice).

Des Weiteren wird angenommen, dass durch die unterschiedlichen Einflüsse der Bauelemente des analogen Signalpfads nicht von einem mittelwertfreien Rauschen ausgegangen werden kann. Entsprechend liegt der Mittelwert des Rauschens näher an der oberen oder an der unteren Quantisierungsgrenze des ADCs und beeinflusst somit die quantisierte Messgröße unterschiedlich stark. Dieser Aspekt ist bei der Modellierung ebenfalls berücksichtigt worden. Dargestellt ist die Wirkung in Abbildung 27. Dort ist ersichtlich, dass trotz gleicher Standardabweichung die Intensität des Rauschens über die Zeit unterschiedlich stark ausgebildet ist. Abschließend wird das normalverteilte, mittelwertbehaftete Rauschen ebenfalls mit der zuvor hergeleiteten, tatsächlichen Auflösung quantisiert und gemäß der Darstellung 25 *rechts* auf Seite 97 mit den achsenpriorisierten Vorzeichen versehen. Zur Bewertung der Modellierungsgenauigkeit wird dieselbe Referenztrajektorie aus Abbildung 21 auf Seite 92 in der Simulati-

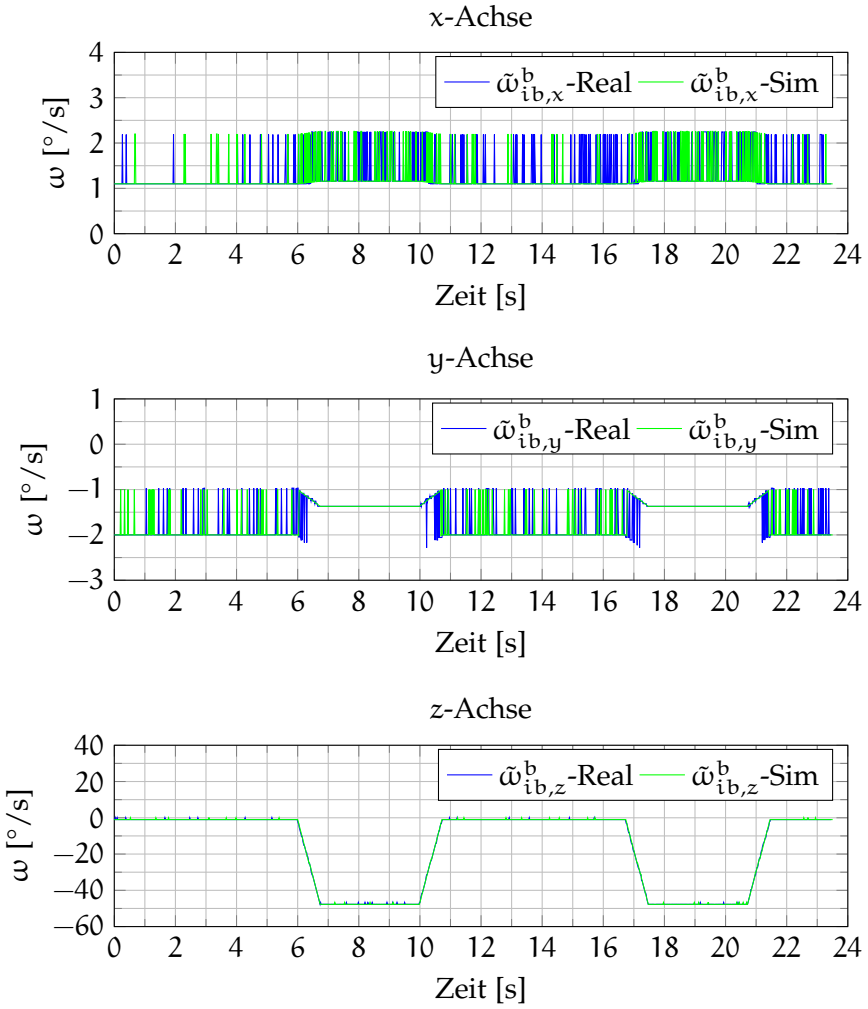


Abbildung 27: Gegenüberstellung der gemessenen Drehraten des realen Sensors zu den simulierten Sensordaten

on implementiert und vom modellierten, virtuellen Sensor erfasst. In Abbildung 27 sind dementsprechend die simulierten Messdaten den realen Sensordaten gegenübergestellt. Es zeigt sich, dass die Simulation das reale Sensorverhalten nahezu ideal reproduziert. Skalierungs- sowie Ausrichtungsfehler konnten, ebenso wie der Nullpunktfehler, abgebildet werden. Zudem ist die Quantisierung der Sensorsignale dem realen System nachempfunden. Wird anschließend der Fehler zwischen realem und simuliertem Sensor in Abbildung 28 betrachtet, ergibt sich lediglich eine Abweichung bedingt durch das Sensorrauschen. Dies ist nur konsequent, da der Ursprung beider Rauschquellen zwar als normalverteilt angenommen wird, diese jedoch zeitlich nicht korrelieren. Abschließend wird die Rauschverteilung mittels einer Histogrammanalyse durchgeführt. Diese ist in Abbildung 29 dargestellt. In der oberen Reihe ist die Rauschverteilung des realen und in der unteren die des simulierten Sensorsystems gezeigt. An-

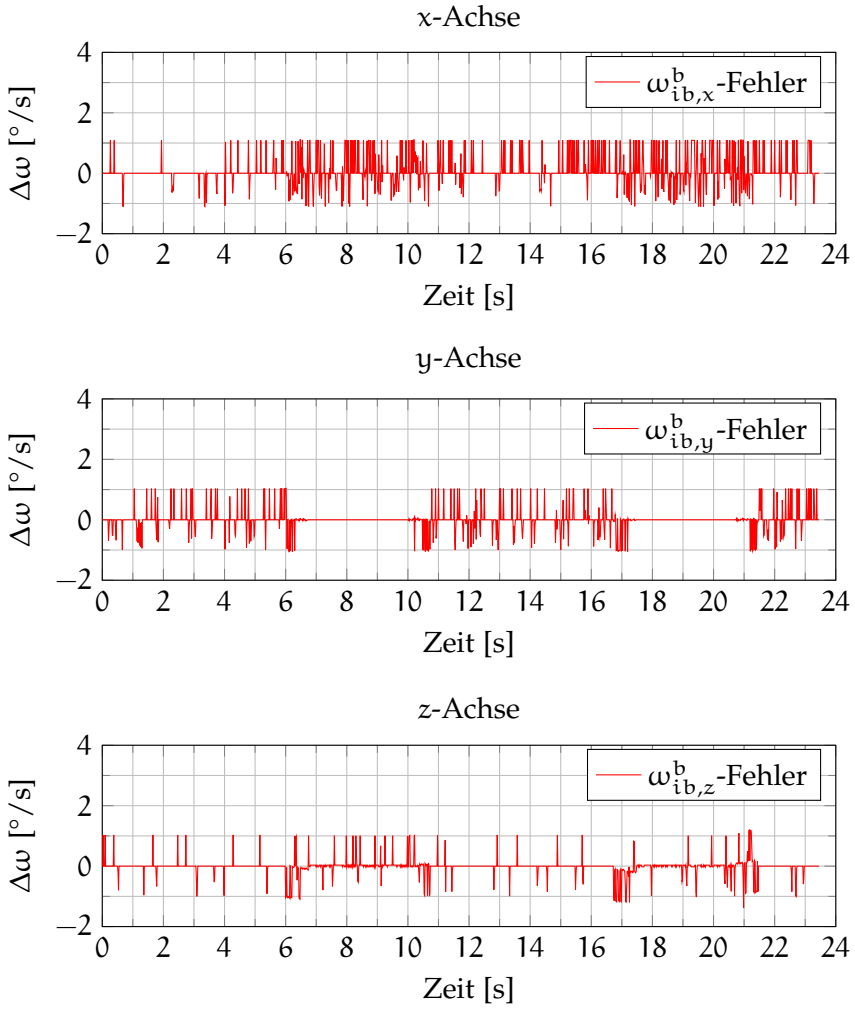


Abbildung 28: Fehler zwischen den realen und den simulierten Drehratensensordaten

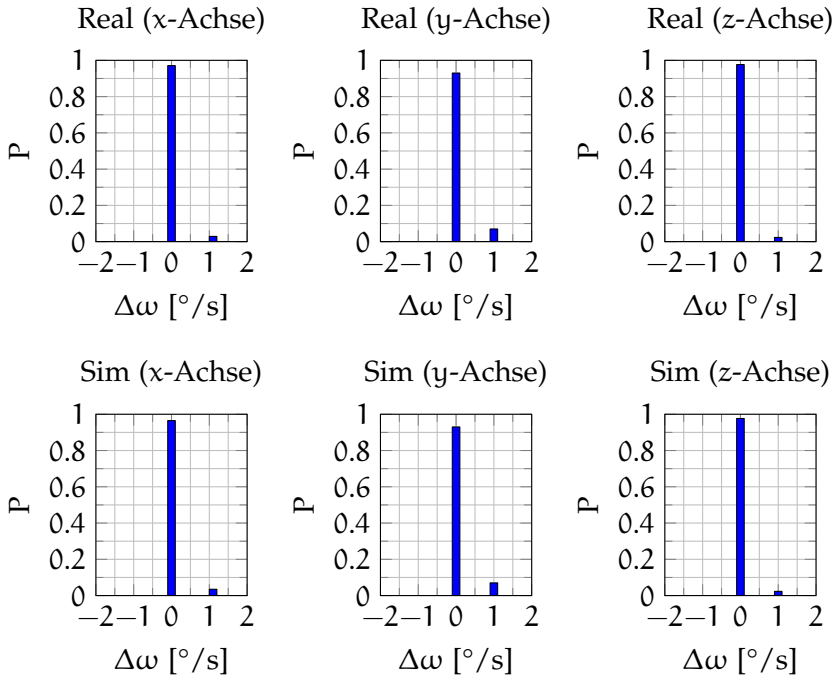


Abbildung 29: Histogramm der Rauschverteilung von realem und simuliertem Drehratensensor

hand dieser Gegenüberstellung ist nachvollziehbar, dass auch das Rauschmodell das reale Rauschen hinreichend genau abbildet. Der Fehler zwischen der realen und der simulierten Rauschverteilung liegt bei jeder Achse unter 0.5 %. Der mittlere Fehler zwischen den realen und den simulierten Sensordaten liegt bei jeder Sensorachse bei weniger als 0.475°/s (RMSE). Zusammenfassend ist festzustellen, dass ein präzises Modell für die Drehratensensoren erarbeitet worden ist.

Allgemein sei jedoch darauf hingewiesen, dass die Histogrammverteilung kein Indiz dafür ist, dass die Sensoren sehr genau messen. In diesem Fall gilt sogar genau das Gegenteil. Die Messabweichungen sind derart gering, weil das eigentliche Sensorrauschen sowie einige der typischen Sensorfehler weit unter der Auflösung des Mikrocontroller-ADCs liegen. Folglich kann auch lediglich eine Drehratenauflösung von $\approx 1°/s$ erreicht werden. Dieser Sachverhalt wird im späteren Abschnitt 8.4 noch eine wichtige Rolle spielen.

Beschleunigungssensoren

Zur Modellierung des Beschleunigungssensors werden im Folgenden die Eigenschaften des 3-achsigen Sensors ermittelt. Den Ausgangspunkt bildet, wie zuvor, das werkseitig kalibrierte Sensorsystem auf dem Flugboard des *AR100B*. Ebenso wie bei der gezeigten Methode für die Drehratensensoren, muss zur präzisen Bestimmung der Modellierungsparameter auch hier ein deutlich präziserer Messaufbau geschaffen werden. Die werkseitige Kalibrierung reicht nicht aus[2]. Hierzu wird das Sensorsystem ebenfalls auf der Referenzplatte des Präzisionsdrehtisches ausgerichtet. Wie bereits erläutert, ist diese mit einer Ausrichtungsgenauigkeit von $\pm 0.05°$ eingemessen. Zur Modellierung wird jede körperfeste Achse des Sensorsystems je-

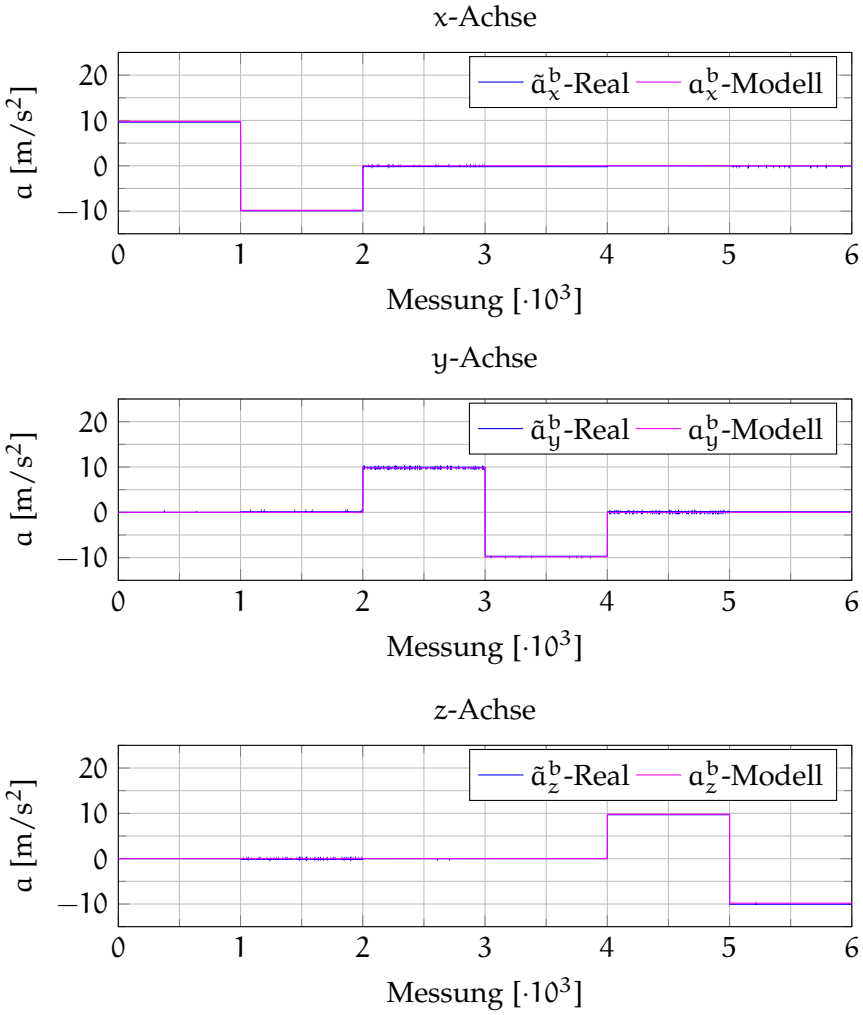


Abbildung 30: Gegenüberstellung der gemessenen Beschleunigungssensordaten (Real) zu den Referenzbeschleunigungen (Modell)

2 Im Anhang B.2 ab Seite 224 wird ein Kalibrierverfahren vorgestellt, welches etwa um eine Zehnerpotenz genauer ist als die Auflösung des Sensorsystems.

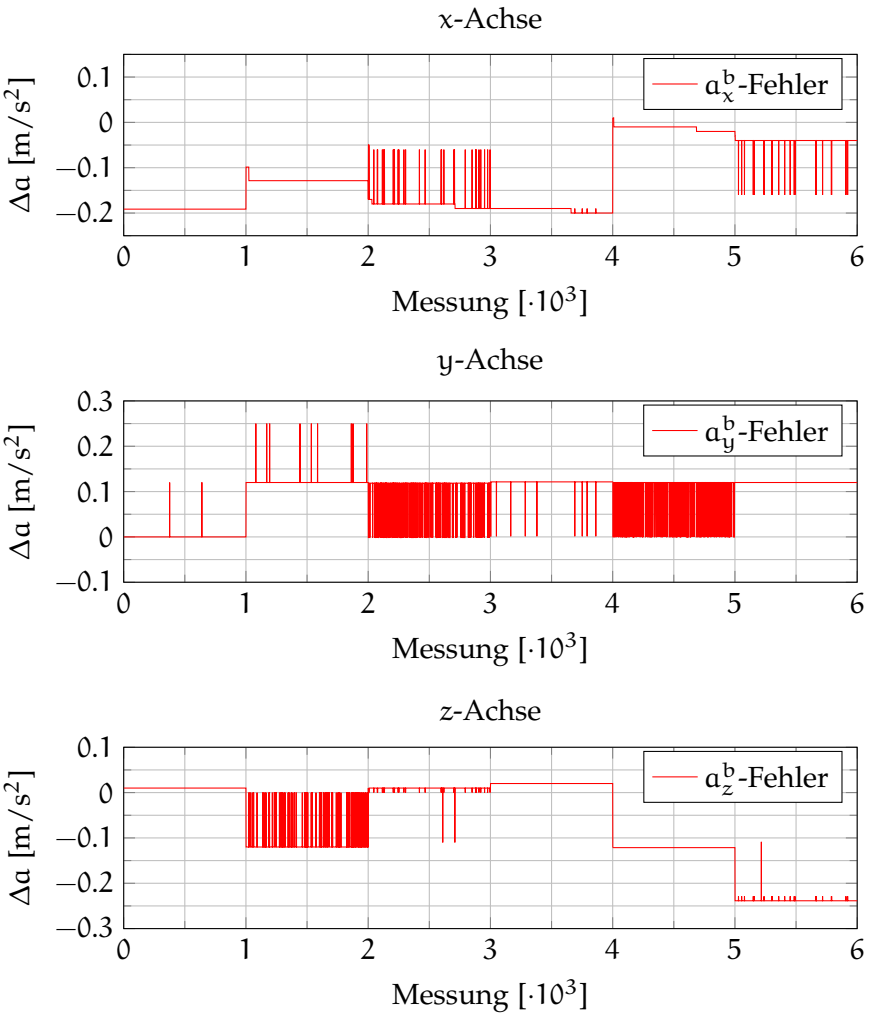


Abbildung 31: Beschleunigungssensorfehler

weils über 1000 Messungen achsenparallel zur Erdgravitation (g_z^n) ausgerichtet. Dies geschieht zuerst in positiver und darauf folgend in negativer Richtung. Die Messungen sind in Abbildung 30 dargestellt. Wird dementsprechend der Fehler als Differenz bezüglich Messung und tatsächlicher Erdgravitation bestimmt (siehe Abbildung 31), wird deutlich, dass das werkseitig kalibrierte Sensorsystem dennoch einen Nullpunktfehler sowie Ausrichtungs- und Skalenfaktorfehler der drei Sensorachsen aufweist, welche modelliert werden müssen. Aufbauend auf diesen Daten werden mit der in Anhang B.2 ab Seite 224 beschriebenen Kalibriermethode die Parameter des Sensorsystems bestimmt. Daraus ergibt sich für den Biasvektor:

$$\vec{b}_a = \begin{pmatrix} -0.16 \\ 0.11 \\ -0.18 \end{pmatrix} \mathrm{m/s^2}. \tag{8.12}$$

Für die *Misalignment*-Matrix ergibt sich:

$$\mathbf{M} = \begin{pmatrix} 1.000810 & 0.007478 & -0.008154 \\ -0.000119 & 0.999908 & 0.000000 \\ -0.012231 & -0.026500 & 0.999823 \end{pmatrix}. \tag{8.13}$$

Werden diese Parameter invers auf die Beschleunigungssensordaten angewendet, folgt der Messfehler nach der Korrektur entsprechend Abbildung 32. Qualitativ konnten der Bias sowie die Ausrichtungs- und Skalierungsfehler optimal bestimmt werden. Quantitativ beträgt der mittlere Fehler (RMSE) der einzelnen Sensorachsen

$$\Delta\vec{a}^{\,b}_{\mathrm{RMSE}} = \begin{pmatrix} 0.012, & 0.034, & 0.018 \end{pmatrix}^{T} \mathrm{m/s^2} \tag{8.14}$$

und ist etwa um eine Zehnerpotenz kleiner als nach der werkseitigen Kalibrierung. Bei weiterer Betrachtung von Abbildung 32 ist zudem

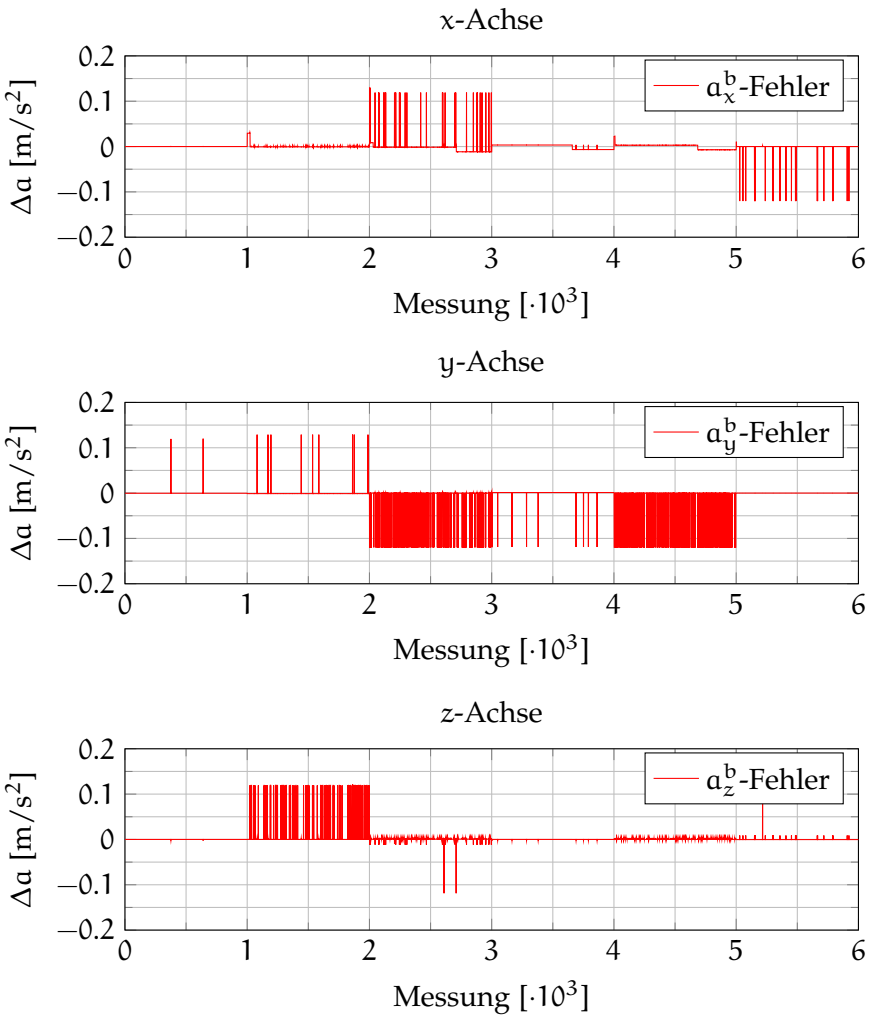


Abbildung 32: Beschleunigungssensorfehler nach der Sensorkalibrierung

ersichtlich, dass auch an dieser Stelle das Rauschen sehr diskret ausgeprägt ist. Dies konnte schon zuvor bei den Drehratensensoren beobachtet werden. Auch in diesem Fall ist dies mit der Quantisierung des Sensormesswertes erklärbar. Der verwendete Beschleunigungssensor hat einen Messbereich von $\pm 18\,\mathrm{g}$ und liefert den Beschleunigungswert als proportionale Spannung mit $57\,\mathrm{mV/g}$ (siehe [8]). Der analoge Messwert wird zusätzlich um 12 dB verstärkt und nachfolgend vom zuvor diskutierten Mikrocontroller-ADC digitalisiert. Daraus ergibt sich für die Quantisierung durch den ADC:

$$1\,\mathrm{LSB} = \frac{3.3\,\mathrm{V}}{2^{10}} = 3.223\,\mathrm{mV}. \tag{8.15}$$

Mit dem Verhältnis

$$1\,\mathrm{g} = 57\,\mathrm{mV} \tag{8.16}$$

und dem Verstärkungsfaktor ($12\,\mathrm{dB} \approx 4$) gilt:

$$1\,\mathrm{LSB} = \frac{3.223\,\mathrm{mV}}{4 \cdot 57\,\frac{\mathrm{mV}}{\mathrm{g}}} \approx 0.0141\,\mathrm{g} \approx 0.139\,\mathrm{m/s^2}\ . \tag{8.17}$$

Das hier zu modellierende Sensorsystem hat somit eine maximale theoretische Auflösung von $0.139\,\mathrm{m/s^2}$. Den realen Messwerten entsprechend beträgt die tatsächliche Auflösung in den drei Sensorachsen $0.12\,\mathrm{m/s^2}$. Dieser Unterschied ist vermutlich auf Bauteiltoleranzen in der Verstärkerstufe zurückzuführen, womit sich eine abweichende, in diesem Fall höhere Verstärkung ergibt. Mit dieser Erkenntnis folgt für die Modellierung die Quantisierungsschrittweite und somit der quantisierte Bias zu:

$$\vec{b}_a = \begin{pmatrix} -0.12 \\ 0.12 \\ -0.24 \end{pmatrix} \mathrm{m/s^2}. \tag{8.18}$$

Im weiteren Verlauf wird wie bei der Drehratensensormodellierung das Rauschen des Beschleunigungssensors unter Berücksichtigung des quantisierten Bias untersucht. Zur besseren Darstellung wird lediglich der letzte Messzyklus (Ausrichtung der z^b-Achse parallel zur Erdgravitation) betrachtet (siehe Abbildung 33 *links* auf Seite 106). Wird zur Beschreibung des Rauschens ebenfalls eine Histogrammanalyse angewendet (siehe Abbildung 33 *rechts*), ergibt sich ein ähnliches Bild wie beim Drehratensensor. Auch in diesem Fall ist das Rauschen diskret bei einem LSB ($0.12\,\mathrm{m/s^2}$) des Mikrocontroller-ADCs ausgebildet. Zudem tritt bezüglich jeder Sensorachse eine priorisierte Polarität auf. Die Erklärung hierfür ist analog zum Drehratensensor zu finden. Der Beschleunigungssensor definiert sein Rauschen nach

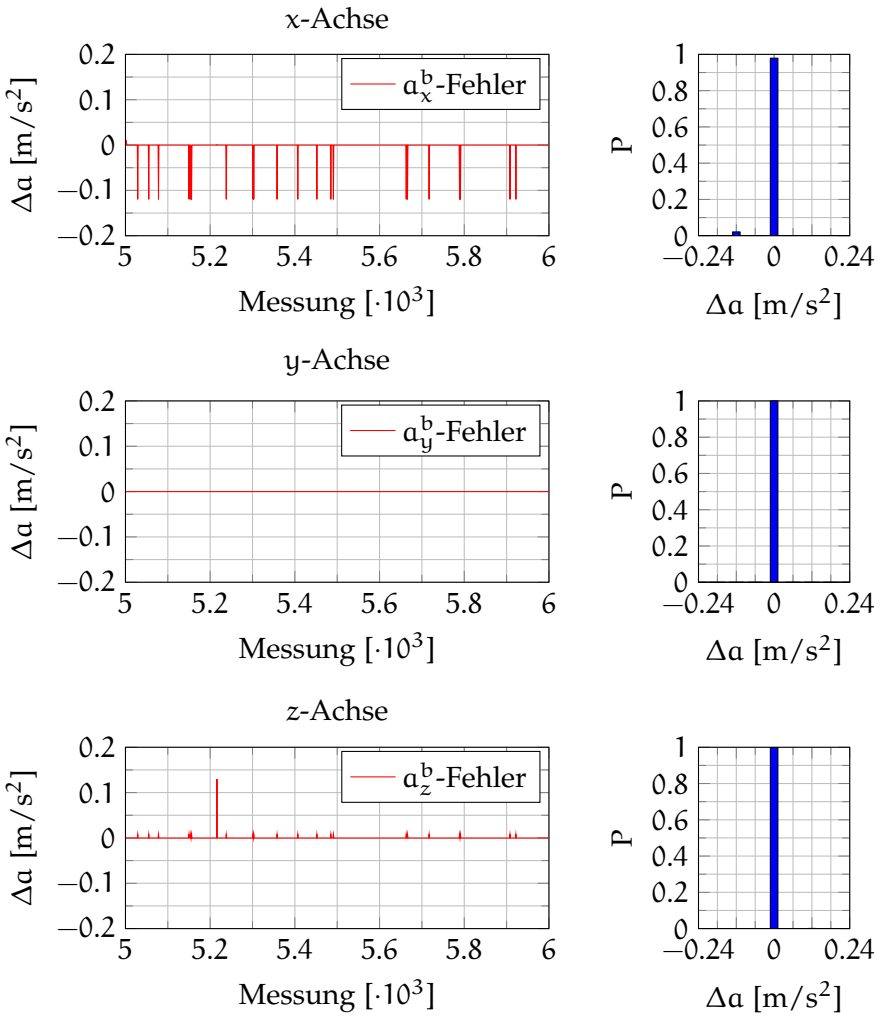


Abbildung 33: Sensorfehler (*links*) und Histogramm der Rauschverteilung (*rechts*) nach der Kalibrierung des Beschleunigungssensors

[8] mit der Rauschdichte (*engl. Noise Density* ND $= 320\,\mu g/\sqrt{Hz}$) und der Sensorbandbreite (*engl. Bandwith* BW $= 22\,Hz$) zu:

$$n_{RMSE} = ND(\sqrt{BW \cdot 1.6})\,, \tag{8.19}$$

$$n_{RMSE} = 320\,\frac{\mu g}{\sqrt{Hz}}(\sqrt{22\,Hz \cdot 1.6})\,, \tag{8.20}$$

$$n_{RMSE} = 1.90\,mg = 0.0186\,m/s^2\,. \tag{8.21}$$

Somit ist das Rauschen des Sensors mit $0.0186\,m/s^2$ um den Faktor 6 kleiner als das Auflösungsverhältnis des ADCs des Mikrocontrollers und kann damit nicht direkt erfasst werden. Dementsprechend ist auch in diesem Fall ein nicht normalverteiltes Rauschen zu beobachten und bei der Modellierung zu berücksichtigen. Ähnlich dem Drehratensensor ist im Analogpfad des Beschleunigungssensors ein Tiefpassfilter zur Filterung von hochfrequenten Störungen integriert. Auch an dieser Stelle wird dieses mittels *PSpice* modelliert und dessen maximale Gruppenlaufzeit mit 21 ms bestimmt. Die Abtastrate

des Beschleunigungssensorsystems beträgt analog der Drehratensensorik 125 Hz. Mit diesen Ergebnissen wird der virtuelle Sensor simuliert. Dazu wird die Signalverzögerung durch den Tiefpassfilter auch an dieser Stelle durch ein PT_1-Glied erzeugt. Daraufhin wird die jetzt verzögerte, aber sonst ideale Beschleunigung quantisiert und entsprechend des Sensormodells aus Gleichung 7.2 auf Seite 73 mit der errechneten *Misalignment*-Matrix aus Gleichung 8.13 auf Seite 104 multipliziert. Der quantisierte Bias aus Gleichung 8.18 auf Seite 105 wird entsprechend hinzugefügt. Zur Modellierung des Rauschens wird dieses im grundsätzlichen Ansatz als normalverteilt, mit der Standardabweichung des Sensors σ_a, angenommen. Somit folgt:

$$\vec{\sigma}_a = \begin{pmatrix} 0.0186 \\ 0.0186 \\ 0.0186 \end{pmatrix} \mathrm{m/s^2}. \tag{8.22}$$

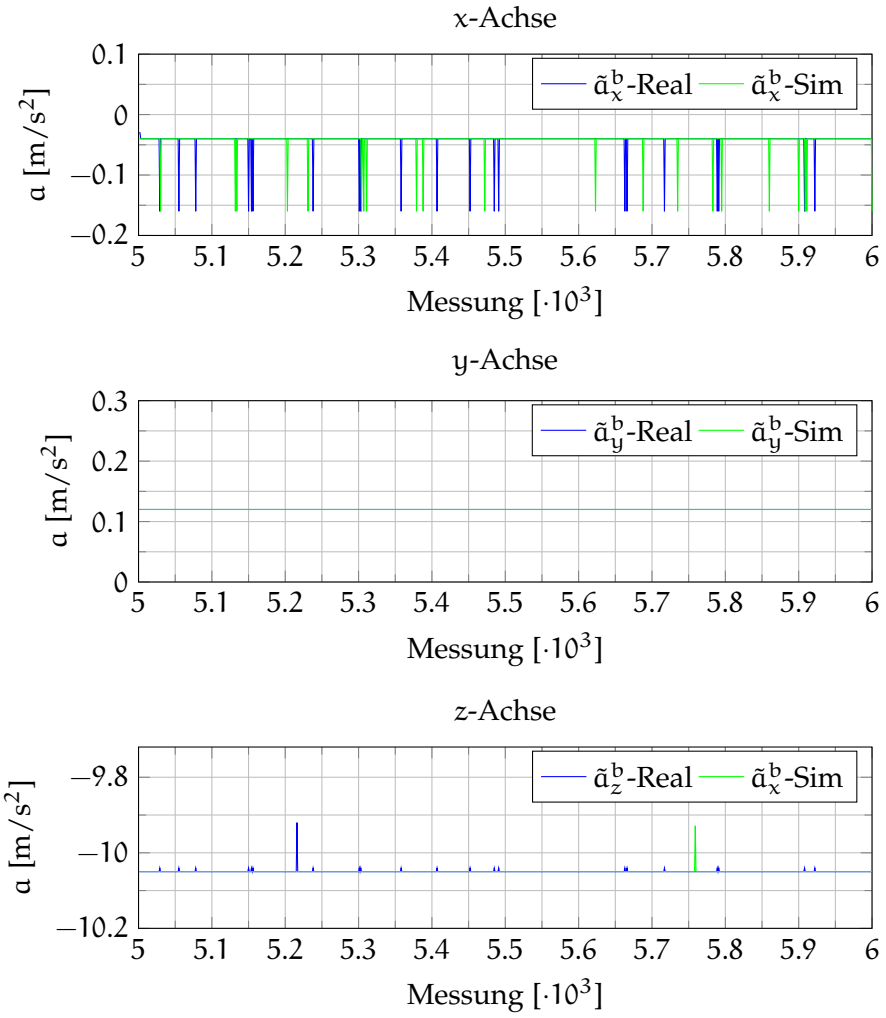


Abbildung 34: Gegenüberstellung der gemessenen Beschleunigungen des realen Sensors zu den simulierten Sensordaten

Um ein realistisches Abbild der analogen Signalstrecke zu erzeugen, wird, vergleichbar zum Modell des Drehratensensors, zusätzlich ein spezifischer Bias des Rauschterms modelliert. Dieser sorgt für eine Verschiebung des normalverteilten Rauschens an die Quantisierungsgrenzen des ADCs. Mit abschließender Quantisierung kann ein vergleichbares Rauschen reproduziert werden. Die Gegenüberstellung der Sensorsignale (siehe Abbildung 34 auf Seite 107) zeigt auf, dass die simulierten Sensordaten in guter Näherung den realen Sensordaten nachempfunden sind. So können die achsenspezifischen Nullpunktfehler sowie Ausrichtungsfehler optimal auf den virtuellen Sensor übertragen werden. Wird diesbezüglich der Fehler zwischen realen und simulierten Sensoren ausgewertet (siehe Abbildung 35), ist erkennbar, dass der Fehler ausschließlich durch das Rauschen gebildet wird. Dies ist, so wie beim Drehratensensor, nur konsequent, da die Rauschprozesse zeitlich unkorreliert sind. Wird das Rauschen über

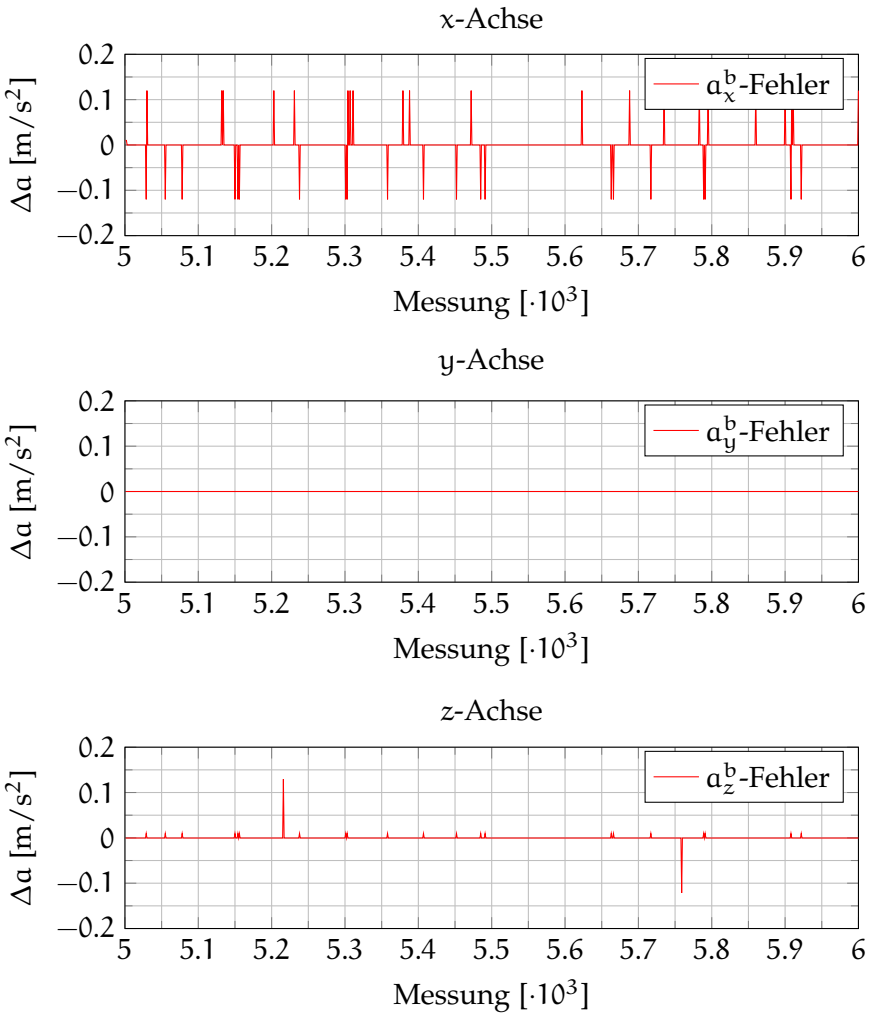


Abbildung 35: Fehler zwischen den realen und den simulierten Beschleunigungssensordaten

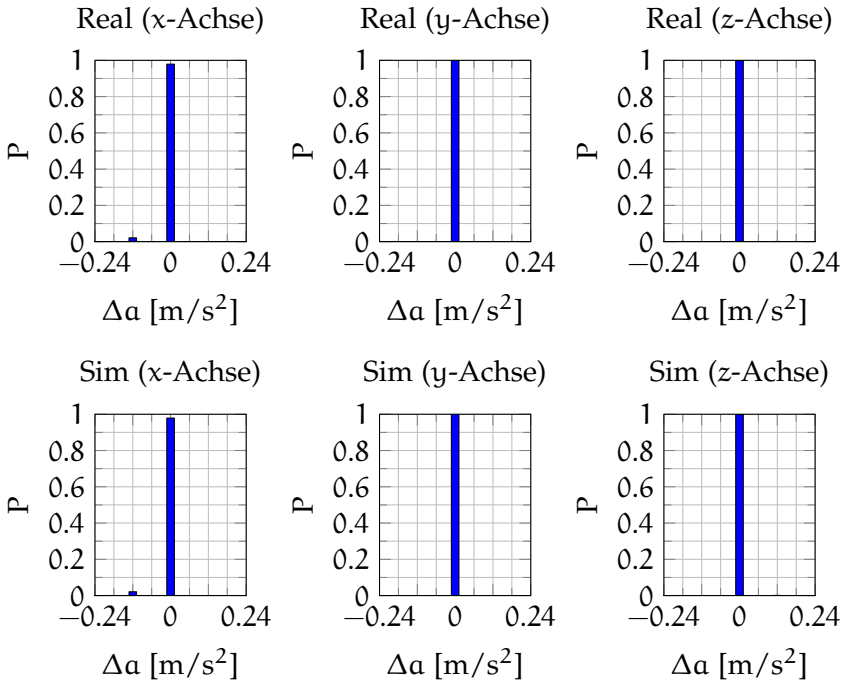


Abbildung 36: Histogramm der Rauschverteilung von realem und simuliertem Beschleunigungssensor

die bekannte Histogrammverteilung analysiert (siehe Abbildung 36), wird deutlich, dass auch der Rauschprozess mit dem hier vorgestellten Ansatz optimal nachempfunden werden konnte. Der Fehler zwischen der realen und der simulierten Rauschverteilung liegt bei jeder Achse unter 0.02 %. Der mittlere Fehler liegt bei jeder Sensorachse bei weniger als $0.025\,\mathrm{m/s^2}$ (RMSE). Zusammenfassend konnte gezeigt werden, dass die realen Sensordaten sehr gut durch das entwickelte Modell nachempfunden werden können.

Im Rahmen einer allgemeinen Charakterisierung des Sensorsystems ist jedoch darauf hinzuweisen, dass, analog zum Drehratensensor, auch das hier modellierte Beschleunigungssensorsystem eine sehr grobe Quantisierung aufweist. Es kann leicht über die Funktion

$$\Delta\phi = \arcsin\left(\frac{0.12\,\mathrm{m/s^2}}{9.8114\,\mathrm{m/s^2}}\right) = 0.70° \tag{8.23}$$

gezeigt werden, dass der kleinste aufzulösende Neigungswinkel $\Delta\phi$ mit der vom Flugboard des *AR100B* bereitgestellten Auflösung des Beschleunigungssensors mit $0.12\,\mathrm{m/s^2}$ lediglich 0.7° beträgt. Zudem ist zu beachten, dass es aufgrund der unterschiedlichen Charakteristik der Tiefpassfilter des Drehraten- sowie Beschleunigungssensorsystems zu unterschiedlichen Gruppenlaufzeiten der Sensorsignale kommt. Kombiniert mit dem nicht normalverteilten und mittelwertbehafteten Rauschen kann dies bei der Datenfusion zu suboptimalen Filterergebnissen führen. Diese systembedingten Einschränkungen werden am Ende dieses Kapitels nochmals aufgegriffen.

Magnetfeldsensoren

Im Anschluss an die Modellierung der IMU soll das Magnetometer untersucht und die entsprechenden Modellparameter bestimmt werden. Da analog zu den vorhergegangenen Sensorsystemen die tatsächlich verarbeiteten Messwerte der Flugplattform modelliert werden sollen, ist das Messsystem zwingend im eingebauten Zustand in Verbindung mit der werkseitigen Kalibrierung zu untersuchen. Deshalb wird die Flugplattform im Außenbereich um alle drei körperfesten Achsen rotiert. Im Idealfall ergibt sich damit eine kugelförmige Punktwolke mit dem Radius der ortsabhängigen, absoluten Intensität des Magnetfeldes $|\vec{h}^n|$ (siehe Abbildung 37 – *magenta* farbenes Netz). In *blau* sind die gemessenen Rohdaten des Magnetometers überlagert. In dieser Darstellung sind die in Kapitel 7.1.2 beschriebenen Fehlereinflüsse (*Hard- and Soft-Iron*-Effekte) eindeutig erkennbar. Um diese Fehler zu minimieren, werden diese Rohdaten systemintern durch die werkseitigen Kalibrierungsparameter des *AR100B* korrigiert. Zur Modellierung werden die korrigierten Sensordaten in Form der absoluten Intensität des Magnetfelds ausgewertet (siehe Abbildung 38). Dort wird deutlich, dass trotz Kalibrierung das gemessene Magnetfeld zum tatsächlichen einen Skalenfaktorfehler sowie einen Nullpunktfehler aufweist. Der mittlere Fehler beträgt hierbei 1.06 µT (RMSE). Der maximale Fehler beträgt 3.38 µT und im Mittel liegt das gemessene Magnetfeld 0.85 µT über dem tatsächlichen Wert. Da sich diese Fehler direkt auf die Genauigkeit der Ausrichtungsbestimmung auswirken, müssen diese in dem Sensormodell berück-

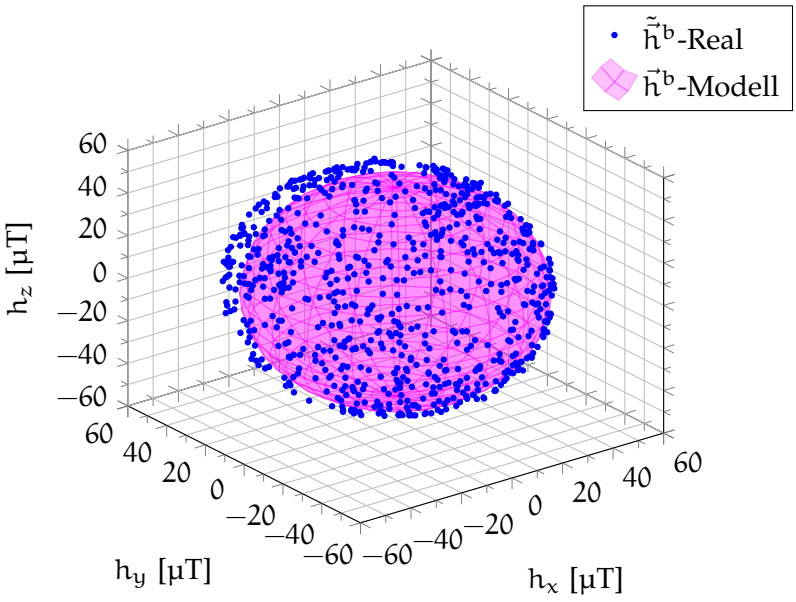


Abbildung 37: Vergleich der gemessenen Magnetfeldsensordaten (Real) mit den Referenzdaten des Erdmagnetfeldmodells (Modell)

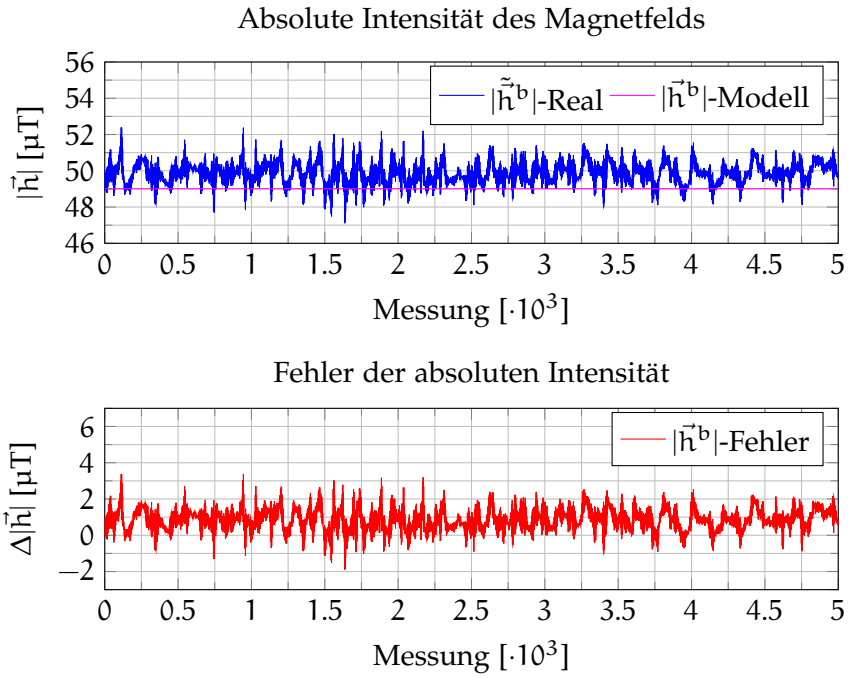


Abbildung 38: Vergleich der absoluten Intensität von gemessenem Magnetfeld gegenüber dem Referenzmagnetfeld

sichtigt werden. Dazu werden die systemintern korrigierten Sensordaten einer weiteren, genaueren Kalibrierroutine zugeführt. Hierbei werden die Daten aus allen drei körperfesten Achsen im dreidimensionalen Raum (siehe Abbildung 37) zusammengeführt. Daraufhin wird in diese Daten ein Ellipsoid mittels der Methode der kleinsten Fehlerquadrate (*engl. Least Squares*) eingepasst (siehe [111]). Danach werden die Parameter des Ellipsoiden zur Kugel der idealen, absoluten Intensität des Magnetfelds transformiert. Daraus ergeben sich die Parameter für den Nullpunktfehler zu:

$$\vec{b}_h = \begin{pmatrix} 0.43 \\ -0.25 \\ 0.19 \end{pmatrix} \mu T. \tag{8.24}$$

Des Weiteren folgt für die *Misalignment*-Matrix:

$$\mathbf{M} = \begin{pmatrix} 1.026105 & 0.002289 & -0.011022 \\ 0.002289 & 1.016847 & 0.002453 \\ -0.011022 & 0.002453 & 1.011821 \end{pmatrix}. \tag{8.25}$$

Eine detailliertere Beschreibung des Kalibrierverfahrens ist im Anhang B.3 ab Seite 228 zu finden. Wie bei den vorangegangenen Sensorsystemen werden auch in diesem Fall die Kalibrierparameter invers auf die Sensordaten angewendet, um ein Bestimmungsmaß für die Kalibriergenauigkeit zu erhalten. Daraus folgend konnte durch

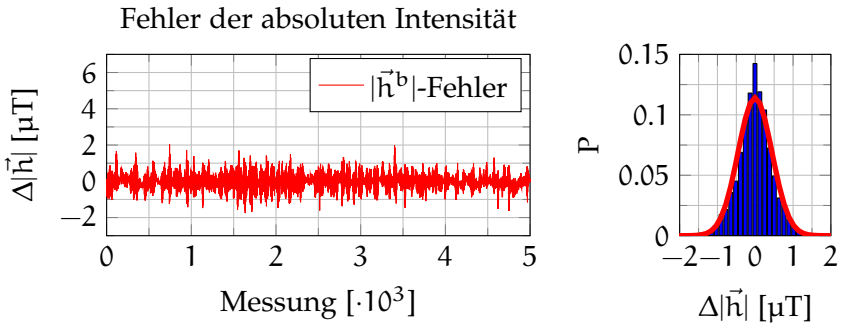


Abbildung 39: Sensorfehler (*links*) und Histogramm der Rauschverteilung (*rechts*) nach der Kalibrierung des Magnetfeldsensors

die in dieser Arbeit gezeigten Kalibrierungen der mittlere gemessene Fehler auf 0.45 µT (RMSE) halbiert werden. Zudem beträgt die mittlere Abweichung des gemessenen Magnetfelds jetzt nur −0.006 µT. Der maximale Fehler konnte auf 2.05 µT reduziert werden. Zusammenfassend ist der Fehler der absoluten Intensität auf Basis der korrigierten Messwerte in Abbildung 39 *links* dargestellt. Betrachtet man darin das Sensorrauschen, ist im Gegensatz zu den bisher untersuchten Sensoren auffällig, dass das Rauschen augenscheinlich deutlich zufälliger verteilt ist. Dies wird durch die in Abbildung 39 *rechts* dargestellte Histogrammanalyse der Rauschverteilung bestätigt. Des Weiteren ist in *rot* die Wahrscheinlichkeitsdichtefunktion überlagert. Diese ergibt sich aus dem zuvor errechneten Mittelwert und der Standardabweichung. Nach dieser Auswertung kann das Rauschen des hier genutzten Magnetfeldsensors als normalverteilt und mittelwertfrei angenommen und entsprechend modelliert werden. Zuletzt ist für die Modellierung der Sensordaten die Quantisierung zu ermitteln. Diese ist im Sensorsystem selbst zu konfigurieren und ergibt sich entsprechend [136] zu 0.128 µT. Die damit verbundene Abtastrate von 44 Hz wird somit in der Simulation umgesetzt. Mit diesen Modellierungsparametern wird das virtuelle Sensorsystem implementiert. Die simulierten Messwerte sind gemäß der absoluten Intensität des Magnet-

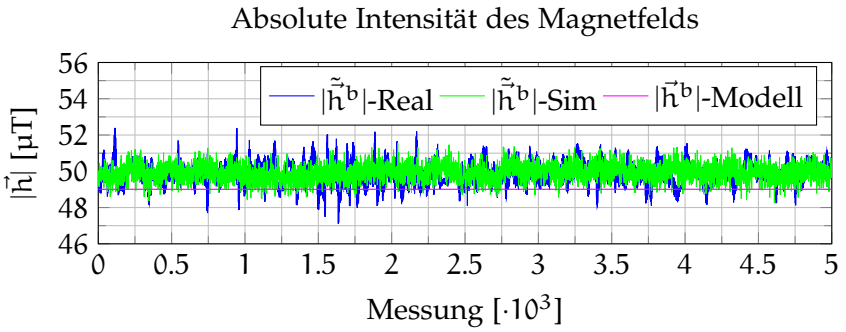


Abbildung 40: Gegenüberstellung der Messdaten des realen Magnetfeldsensors zum simulierten Sensor

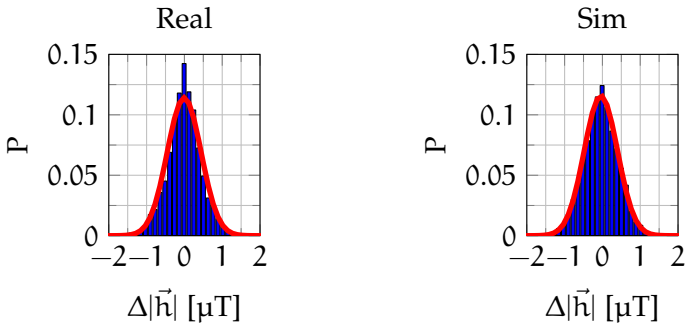


Abbildung 41: Histogramm der Rauschverteilung von realem und simuliertem Magnetfeldsensor

felds in Abbildung 40 dargestellt. Hier ist deutlich zu erkennen, dass die Skalenfaktorfehler sowie der Bias entsprechend dem originalen Sensor simuliert werden konnten. Im Gegensatz zu den vorherigen Sensordaten, bei denen die Bewegungsabläufe bei der Sensordatenaufnahme vollständig korreliert und synchronisiert waren, trifft dies in diesem Fall nicht zu. Daher treten die Einflüsse der Skalierungsfehler der einzelnen Sensorachsen zu unterschiedlichen Zeitpunkten auf und können somit nicht direkt miteinander verglichen werden. Aus diesem Grund wird an dieser Stelle darauf verzichtet, die Messwerte der einzelnen Sensorachsen von simuliertem und realem Sensor unmittelbar gegenüberzustellen. Stattdessen wird zur Bewertung der Modellierungsgenauigkeit die mittlere Abweichung vom simulierten Sensor sowie von dem realen Sensor zur errechneten absoluten Magnetfeldintensität auf Basis des Erdmagnetfeldmodells bestimmt. In beiden Fällen wird durchgehend eine um 0.855 µT höhere, absolute Intensität gemessen. Abschließend wird das Rauschverhalten von simuliertem zu realem Sensor untersucht. Hierzu wird ebenfalls eine Histogrammanalyse der vom Ausrichtungs-, Skalenfaktor- sowie Nullpunktfehler korrigierten Messungen vorgenommen. In Abbildung 41 sind die Ergebnisse des realen Sensors *links* und die des simulierten Sensors *rechts* einander gegenübergestellt. Zudem ist für beide Messungen die errechnete Wahrscheinlichkeitsdichtefunktion in *rot* überlagert. Die Standardabweichung beträgt $\sigma = 0.448\,\mu T$ für den realen Sensor sowie $\sigma = 0.442\,\mu T$ für den simulierten Sensor. Der mittlere Fehler zwischen der gemessenen absoluten Magnetfeldintensität des realen Sensors und der des simulierten Sensors beträgt lediglich 0.802 µT (RMSE). Somit ist der Sensor präzise simuliert.

Baro-Altimeter

Im Gegensatz zu den bisher modellierten Sensorsystemen ist der Aufbau des Ausgangsmodells für die barometrische Höhenmessung ab-

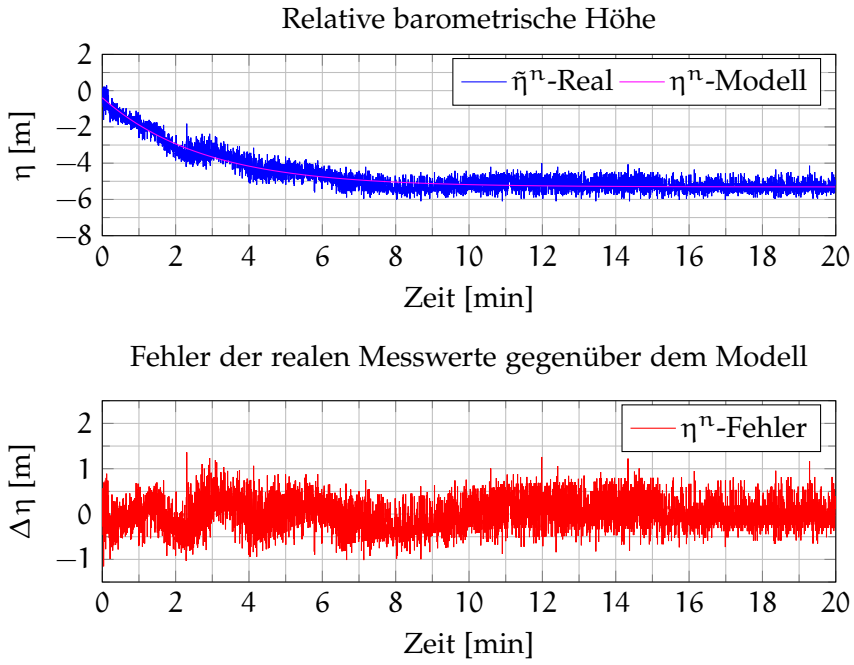


Abbildung 42: Sensordaten des Baro-Altimeters im stationären Zustand

weichend. Rückblickend auf Abschnitt 7.1.3 wird das Modell an dieser Stelle nochmals dargestellt:

$$\tilde{\eta}^n = \eta^n + S(1 - e^{\frac{-t}{T}}) + \vec{b}_\eta + \vec{n}_\eta \ . \tag{8.26}$$

Neben den bekannten Parametern, wie Bias ($\vec{b}_\eta$) und Sensorrauschen ($\vec{n}_\eta$), ist ein entscheidender Einflussfaktor die Drift des Messwertes nach dem Einschalten der Sensorsysteme. Diese Drift wird vorwiegend durch interne Erwärmungsprozesse hervorgerufen und kann als Exponentialfunktion modelliert werden. Im folgenden Abschnitt sollen diese Parameter bestimmt werden.

Das Messsystem wird hierzu stationär betrieben. Der gemessene Luftdruck wird im Sensorsystem direkt in die entsprechende Höhe nach Gleichung 4.22 auf Seite 44 umgerechnet. Direkt nach dem Einschalten des Systems erfolgt zudem die Bestimmung des relativen Nullpunkts. Die aufgenommenen Messwerte geben die Differenz zu diesem Nullpunkt wieder. In Abbildung 42 sind in *blau* die Messdaten über 20 Minuten dargestellt. Zur Modellbestimmung wird die Exponentialfunktion bestimmt, welche den kleinsten quadratischen Fehler zu den realen Messdaten aufweist (*Exponetial Curve Fitting*). Diese ist in *magenta* den Messdaten überlagert. Die dadurch bestimmten Parameter ergeben sich zu:

$$\tilde{\eta}^n = \eta^n + (-4.92)(1 - e^{\frac{-t}{165\,s}}) + \vec{b}_\eta + \vec{n}_\eta \ . \tag{8.27}$$

Trotz der Nullpunktbestimmung nach dem Einschalten ergibt sich durch den zeitlichen Versatz bis zur ersten Messung ein Bias von:

$$\vec{b}_\eta = -0.392\,m \ . \tag{8.28}$$

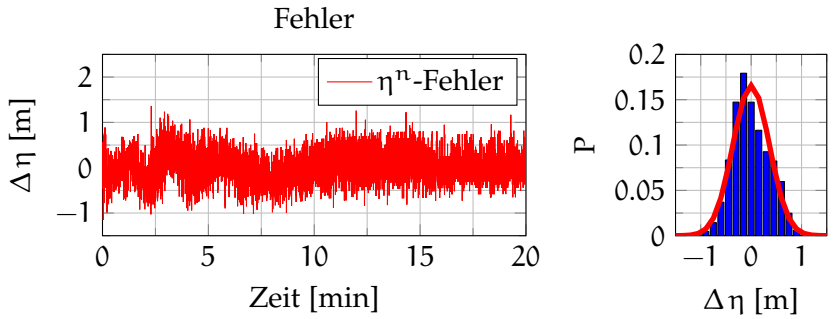


Abbildung 43: Sensorfehler (*links*) und Histogramm der Rauschverteilung (*rechts*) des Baro-Altimeters

In Abbildung 42 *unten* und in Abbildung 43 *links* ist in *rot* der Fehler zwischen realen Sensordaten und modellierter Exponentialfunktion dargestellt. Hier zeigt sich, dass die bestimmten Parameter sehr gut den realen Messwert approximieren. Der mittlere Fehler beträgt lediglich 0.36 m (RMSE). Der Sensor zeigt hingegen ein Rauschen von etwa ± 0.8 m. Zur Modellbildung ist eine nähere Untersuchung erforderlich. Dazu wird bekannterweise eine Histogrammanalyse durchgeführt (siehe Abbildung 43 *rechts*). Analog zum Magnetometer ist dem Histogramm die ermittelte Wahrscheinlichkeitsdichtefunktion überlagert. Die Standardabweichung des Rauschens beträgt nach dieser Analyse $\sigma = 0.36$ m. Obwohl das Rauschen entsprechend des Histogramms leicht nach links ausgelenkt ist, wird zur weiteren Model-

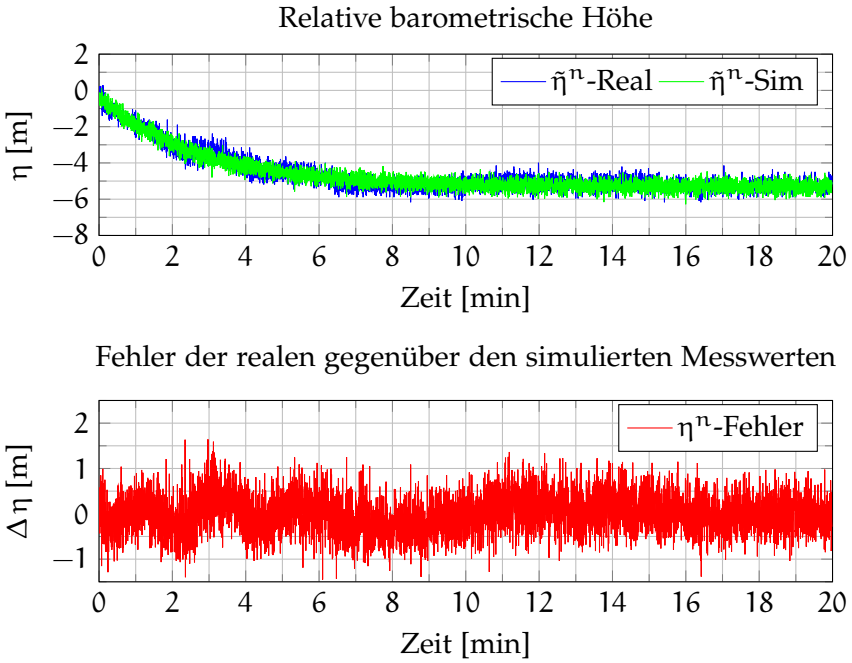


Abbildung 44: Gegenüberstellung der Messdaten des realen Baro-Altimeters zum simulierten Sensor

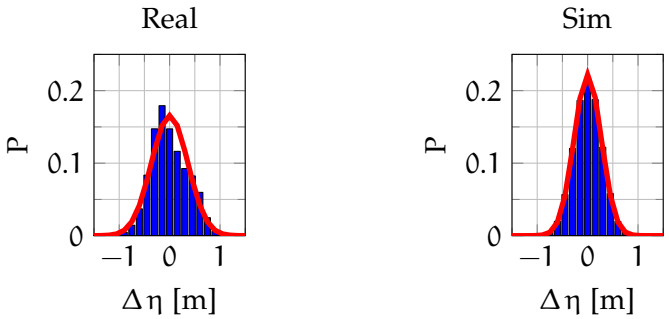


Abbildung 45: Histogramm der Rauschverteilung von realem und simuliertem Baro-Altimeter

lierung von einem normalverteilten, mittelwertfreien Rauschprozess ausgegangen. Es ist anzunehmen, dass die hier gezeigte Abweichung auf äußere Einflüsse zurückzuführen ist. Auch wenn die Messdaten im stationären Zustand aufgenommen worden sind, lassen sich von außen wirkende Luftdruckänderungen nicht vollständig vermeiden. Mit diesen Modellierungsparametern wird der virtuelle Sensor in der Simulation umgesetzt. Hierzu werden ebenfalls die effektive Quantisierungsschrittweite von 0.1 m und die systeminterne Abtastrate von 125 Hz des realen Sensors in die Simulation implementiert. Als Ergebnis des modellierten Baro-Altimeters sind die realen den simulierten Messdaten in Abbildung 44 auf Seite 115 gegenübergestellt. In der Simulation wird allerdings die Standardabweichung des Rauschens auf $\sigma = 0.27\,\mathrm{m}$ reduziert. Dieser Schritt ist naheliegend, da die Standardabweichung des Rauschens des realen Sensors, besonders in den ersten Minuten, zusätzlich durch leichte externe Störungen, wie bereits zuvor erörtert, beeinflusst worden ist. Dementsprechend zeigt sich in Abbildung 44 (*oben*) ab Minute 12 eine sehr gute Reproduktion des realen Rauschens. Abschließend wird die Verteilung des Rauschens in Abbildung 45 gegenübergestellt. In der rechten Abbildung ist an der in *rot* überlagerten Wahrscheinlichkeitsdichtefunktion die zuvor erläuterte reduzierte Standardabweichung des simulierten Rauschens erkennbar. In Kombination mit Abbildung 44 ist somit nachgewiesen, dass der reale Sensor in guter Näherung durch das hier hergeleitete Sensormodell abgebildet wird. Dies bestätigt ebenfalls der mittlere Fehler zwischen realem und simuliertem Sensor, der mit 0.43 m (RMSE) im Bereich des Sensorrauschens liegt.

GNSS

Wie bereits in Abschnitt 7.1.4 dargestellt, ist die Modellierung eines GNSS-Empfängers überaus komplex. Dies liegt einerseits an den unterschiedlichen äußeren sowie inneren Fehlereinflüssen des Satellitensystems selbst und andererseits an der internen Signalverarbeitung des in diesem Fall genutzten proprietären Empfängermoduls.

Dabei berechnet das Empfängermodul die aktuelle Position und Geschwindigkeit auf Basis voreingestellter Bewegungsmodelle. Deshalb ist die Bestimmung der tatsächlichen Modellierungsparameter nicht ohne Weiteres möglich. Da jedoch im Fall dieser Arbeit die Verarbeitung der globalen Positionsdaten nur eine untergeordnete Rolle spielt, wird ein vereinfachtes, lokales Positionsmodell angenommen. Hierbei wird die Position relativ zum Startpunkt bestimmt und als Fehler ein normalverteilter Rauschprozess mit der Standardabweichung von $\sigma = 0.6\,\mathrm{m}$ angenommen. Darüber hinaus wird die Abtastrate von 4 Hz des realen GNSS-Empfängers in die Simulation implementiert. Im später folgenden Abschnitt 8.2.2 wird gezeigt, dass dieses Modell durchaus ausreichend für das hier angestrebte Ziel ist.

8.1.2 *Antriebe*

Gemäß der Herleitung des Antriebsmodells aus Kapitel 7.2.1 sollen an dieser Stelle die Daten der realen Motoren analysiert und darauf aufbauend ein Antriebsmodell erstellt werden. Hierzu sind zwei Messaufbauten entworfen worden, mit denen der funktionale Zusammenhang zwischen Drehzahl und Schubkraft sowie Drehzahl und Drehmoment bestimmt werden kann. Dabei wird zuerst das stationäre und anschließend das dynamische Übertragungsverhalten der Antriebe untersucht.

Stationäres Antriebsmodell

Zur Modellierung des stationären Übertragungsverhaltens wird die Steuergröße s_M der Antriebe mit gleichmäßiger Schrittweite inkrementiert und dabei die Drehzahl N_M, die Schubkraft T_R und das Drehmoment M_R aufgezeichnet. Die Drehzahl folgt dabei entsprechend $N_M = 2\pi \cdot \omega_M \cdot 60$ und wird in Umdrehungen pro Minute angegeben. Die ermittelten Messkurven der vier Flugplattformantrie-

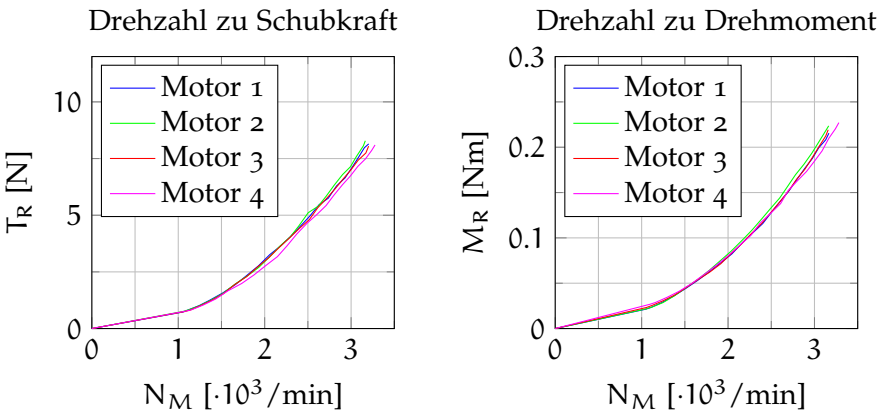


Abbildung 46: Antriebskennlinien im stationären Zustand

be sind in Abbildung 46 *links* auf Seite 117 als Zusammenhang von Drehzahl zu Schubkraft und *rechts* als Zusammenhang von Drehzahl zu Drehmoment dargestellt. Diese Messkurven zeigen anschaulich, dass die Antriebe ein sehr ähnliches Übertragungsverhalten haben, jedoch leichte Variationen zueinander aufweisen. Aus diesem Grund wird jeder Antrieb selbst modelliert. Durch die Ergebnisse der Herleitung aus Abschnitt 7.2.1 mit Gleichung 7.23 auf Seite 80 konnte festgestellt werden, dass zwischen Drehzahl und Schubkraft prinzipiell ein quadratischer Zusammenhang besteht. Deshalb ist auf die Messdaten ein polynomisches Ausgleichsverfahren (*Polynomial Curve Fitting*) zweiter Ordnung angewendet worden, das eine quadratische Funktion in der Form

$$T_R = a \cdot {N_M}^2 + b \cdot N_M + c \tag{8.29}$$

auf Basis der Methode der kleinsten Fehlerquadrate bestimmt. Hierbei wird die Modellierungsfunktion explizit für den Arbeitsbereich des Motors bestimmt, welcher bei $N_M \approx 1050\,1/\text{min}$ beginnt. In Abbildung 47 sind die vier Antriebe einzeln aufgetragen. In *blau* sind die ermittelten Messdaten dargestellt. Dabei ist jeder diskrete Messpunkt mindestens 250-mal gemessen worden. Die daraus bestimm-

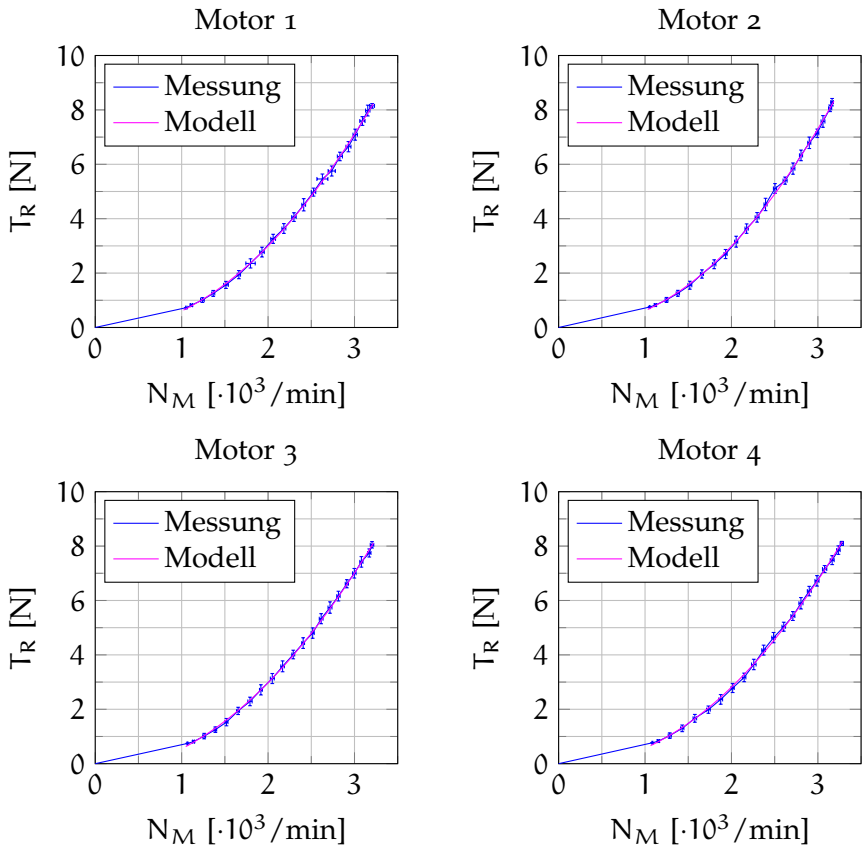


Abbildung 47: Stationäres Antriebsmodell: Drehzahl (N_M) zu Schubkraft (T_R)

ten Standardabweichungen sowohl für die Drehzahl als auch für die Schubkraft sind für jeden Messpunkt als x/y Fehlerbalken mit 1σ eingetragen. In *magenta* ist das berechnete Schubkraftmodell überlagert. Der Modellierungsfehler für die Schubkraft beträgt bei allen vier Antrieben weniger als 0.065 N (RMSE). Die Modellparametervektoren für das Schubkraftmodell lauten somit:

$$\begin{aligned}\begin{pmatrix} T_{R,1} \\ T_{R,2} \\ T_{R,3} \\ T_{R,4} \end{pmatrix} = & \begin{pmatrix} 8.60 \cdot 10^{-7} \\ 9.74 \cdot 10^{-7} \\ 8.06 \cdot 10^{-7} \\ 8.48 \cdot 10^{-7} \end{pmatrix} \begin{pmatrix} N_{M,1} \\ N_{M,2} \\ N_{M,3} \\ N_{M,4} \end{pmatrix}^2 \\ & + \begin{pmatrix} -1.93 \cdot 10^{-4} \\ -5.52 \cdot 10^{-4} \\ 7.33 \cdot 10^{-6} \\ -3.05 \cdot 10^{-4} \end{pmatrix} \begin{pmatrix} N_{M,1} \\ N_{M,2} \\ N_{M,3} \\ N_{M,4} \end{pmatrix} + \begin{pmatrix} -0.059 \\ 0.198 \\ -0.264 \\ 0.029 \end{pmatrix}.\end{aligned} \tag{8.30}$$

Analog zur Schubkraft wird das Drehmoment modelliert. Ebenfalls ist, entsprechend der Herleitung aus Kapitel 7.2.1 Gleichung 7.26 auf Seite 81, der Zusammenhang zwischen Drehzahl und Drehmoment

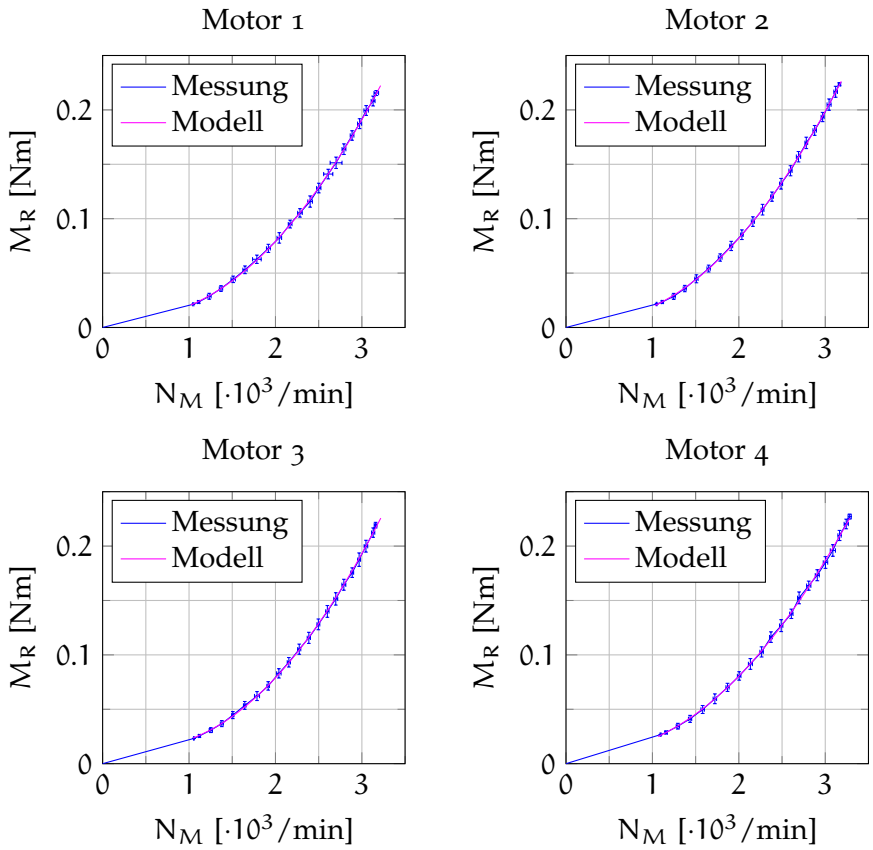


Abbildung 48: Stationäres Antriebsmodell: Drehzahl (N_M) zu Drehmoment (M_R)

quadratisch. Diese Ergebnisse sind in Abbildung 48 dargestellt. Auch hier sind in *blau* die Messdaten sowie deren Standardabweichungen mit 1σ als x/y Fehlerbalken aufgezeigt. Mit dem polynomischen Ausgleichsverfahren zweiter Ordnung ergibt sich die quadratische Funktion in der Form:

$$M_R = a \cdot {N_M}^2 + b \cdot N_M + c \,. \tag{8.31}$$

Das somit bestimmte Drehmomentmodell ist in *magenta* überlagert. Der Modellierungsfehler für das Drehmoment beträgt bei allen vier Antrieben weniger als 0.003 Nm (RMSE). Die Funktion des Drehmomentmodells lautet:

$$\begin{aligned}\begin{pmatrix} M_{R,1} \\ M_{R,2} \\ M_{R,3} \\ M_{R,4} \end{pmatrix} = & \begin{pmatrix} 2.64 \cdot 10^{-8} \\ 2.69 \cdot 10^{-8} \\ 2.88 \cdot 10^{-8} \\ 2.43 \cdot 10^{-8} \end{pmatrix} \begin{pmatrix} N_{M,1} \\ N_{M,2} \\ N_{M,3} \\ N_{M,4} \end{pmatrix}^2 \\ & + \begin{pmatrix} -2.02 \cdot 10^{-5} \\ -1.80 \cdot 10^{-5} \\ -2.97 \cdot 10^{-5} \\ -1.53 \cdot 10^{-5} \end{pmatrix} \begin{pmatrix} N_{M,1} \\ N_{M,2} \\ N_{M,3} \\ N_{M,4} \end{pmatrix} + \begin{pmatrix} 0.014 \\ 0.010 \\ 0.023 \\ 0.014 \end{pmatrix} .\end{aligned} \tag{8.32}$$

Allerdings ist der Eingabevektor des Antriebsmodells (siehe Abbildung 17 auf Seite 70) nicht die Drehzahl, sondern die Steuergröße s_M. Somit muss zusätzlich eine Modellierungsfunktion geschaffen werden, welche die eigentliche Steuergröße auf die Drehzahl der Antriebe abbildet. Hierzu ist neben der Schubkraft, dem Drehmoment und der Drehzahl zusätzlich die Steuergröße in allen Messreihen mit aufgezeichnet worden. Abbildung 49 zeigt den Zusammenhang zwischen Steuergröße und Drehzahl jedes einzelnen Antriebs. Analog dazu folgt auch hier durch Anwendung des polynomischen Ausgleichsverfahrens die Modellierungsfunktion in der Form:

$$N_M = a \cdot {s_M}^2 + b \cdot s_M + c \,. \tag{8.33}$$

Durch das Einsetzen der Parametervektoren ergibt sich für die vier Antriebe:

$$\begin{aligned}\begin{pmatrix} N_{M,1} \\ N_{M,2} \\ N_{M,3} \\ N_{M,4} \end{pmatrix} = & \begin{pmatrix} -0.020 \\ -0.020 \\ -0.017 \\ -0.023 \end{pmatrix} \begin{pmatrix} s_{M,1} \\ s_{M,2} \\ s_{M,3} \\ s_{M,4} \end{pmatrix}^2 \\ & + \begin{pmatrix} 17.924 \\ 17.769 \\ 16.821 \\ 18.901 \end{pmatrix} \begin{pmatrix} s_{M,1} \\ s_{M,2} \\ s_{M,3} \\ s_{M,4} \end{pmatrix} + \begin{pmatrix} -43.060 \\ -23.036 \\ 40.769 \\ -51.848 \end{pmatrix} .\end{aligned} \tag{8.34}$$

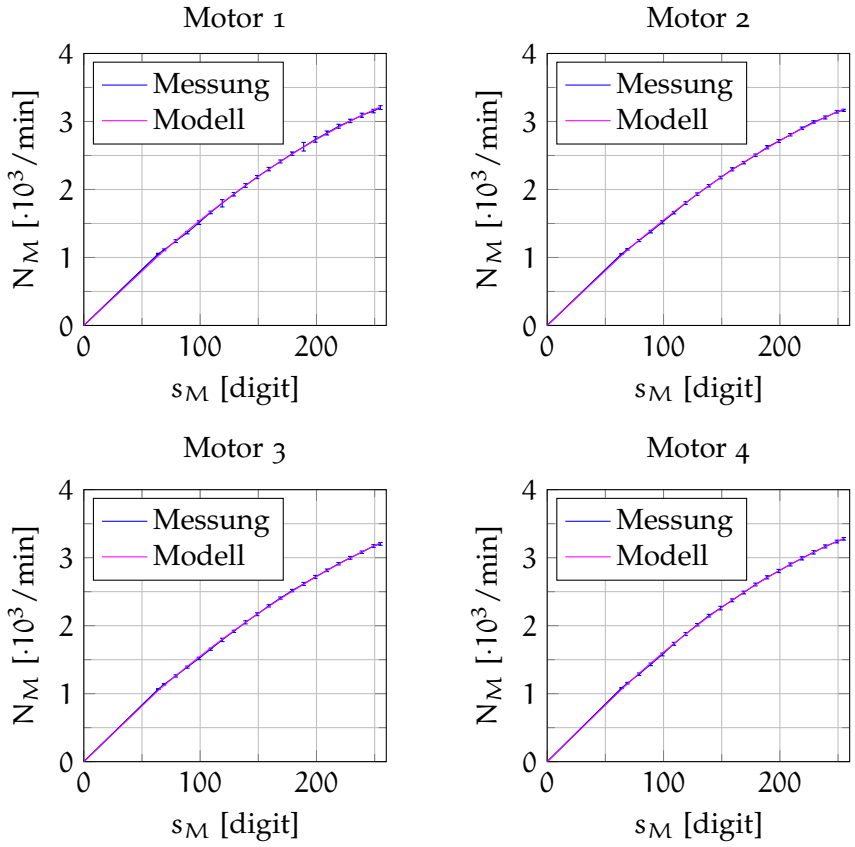


Abbildung 49: Stationäres Antriebsmodell: Steuergröße (s_M) zu Drehzahl (N_M)

In anderen Arbeiten, wie beispielsweise in [115], sind Schubkraft und Drehmoment direkt als Funktion der Steuergröße modelliert. Dies wird an dieser Stelle explizit nicht gemacht, da die Drehzahl des Motors nicht allein von der Steuergröße, sondern ebenfalls von der Motorspannung sowie der Motortemperatur abhängt. Hingegen sind die Modelle „Drehzahl zu Schubkraft“ und „Drehzahl zu Drehmoment“ von diesen Faktoren unabhängig. Das zuvor gezeigte Modell „Steuergröße zu Drehzahl“ ist bei der im Durchschnitt zu erwartenden mittleren Betriebsspannung der Flugplattform von 15.6 V aufgezeichnet worden. Exemplarisch ist in Abbildung 50 *links* auf Seite 122 dieselbe stationäre Antriebskennlinie bei 14.4 V aufgezeichnet worden. Darin ist zu sehen, dass bei geringerer Betriebsspannung bzw. Motorspannung dieselbe Steuergröße eine deutlich geringere Drehzahl erzeugt und somit auch eine geringere Schubkraft sowie ein geringeres Drehmoment zur Folge hat. Die Schubkraft- und Drehmomentmodelle bleiben jedoch gleich. Beispielhaft sei dazu die Schubkraft in Abhängigkeit der Drehzahl von Motor 1 bei unterschiedlicher Spannung in Abbildung 50 *rechts* dargestellt. Hier liegen die Messkurven nahezu übereinander. Lediglich die Messkurve bei geringerer Spannung erreicht proportional eine geringere Maximaldrehzahl. Somit

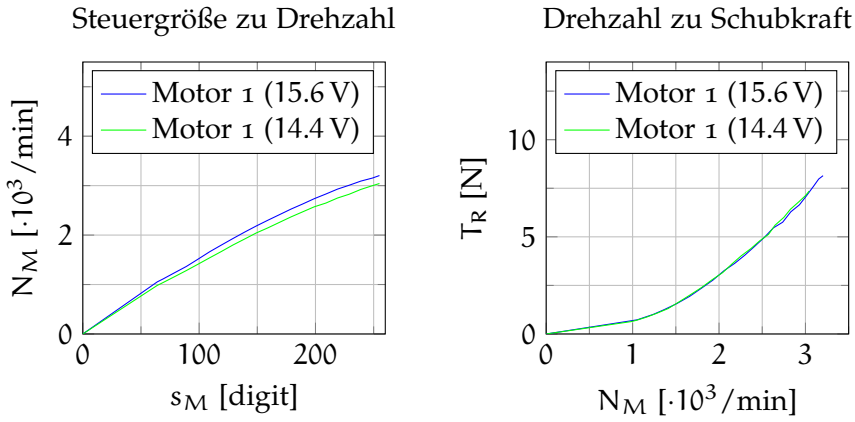


Abbildung 50: Spannungsabhängigkeit des Motormodells

ist bewiesen, dass das Verhältnis von Drehzahl zu Schubkraft spannungsunabhängig ist. Durch diese Aufteilung der Antriebsmodellierung kann, bezogen auf das Steuergrößen- zu Drehzahlmodell, zukünftig auch der Einfluss der Batterieentladung auf die Systemeigenschaften durch einen Korrekturterm implementiert werden.

Dynamisches Antriebsmodell

Nach der Betrachtung des stationären Übertragungsverhaltens muss im Weiteren die Antriebsdynamik untersucht werden. Idealerweise würde ein Antrieb eine Änderung der Steuergröße direkt in eine proportionale Kraft bzw. ein proportionales Drehmoment umsetzen. Durch das mechanische Trägheitsmoment und die Motorinduktivität kommt es jedoch systembedingt zu einem verzögerten, nichtlinearen Übertragungsverhalten. Um dieses Übertragungsverhalten zu modellieren, wird in der Regelungstechnik üblicherweise die Sprungantwort der Modellierungsstrecke untersucht. In diesem Fall wird die Steuergröße innerhalb eines Zeitschritts von 64 digit (minimale Ansteuerung des Arbeitsbereichs) auf 255 digit (maximale Ansteuerung des Arbeitsbereichs) geändert (siehe Abbildung 51 *links*). Idealerweise würde die Drehzahl des Motors von der Ruhedrehzahl ($N_M \approx 1050\,1/\text{min}$) auf die maximale Drehzahl direkt ansteigen. Dies ist exemplarisch in Abbildung 51 *rechts* in *rot* dargestellt. Die tatsächliche Sprungantwort des Antriebs, in *blau* dargestellt, sieht jedoch deutlich anders aus. Allgemein werden Gleichstrommotoren als PT_1-Glied modelliert. Dies hätte in diesem Fall jedoch zu erheblichen Abweichungen von der realen Drehzahl geführt. Daher wurde das präzisere PT_2-Übertragungsmodell genutzt. Hierbei ist jedoch wichtig, dass es sich um ein nicht schwingfähiges PT_2-Glied handelt. Dieses kann aus zwei

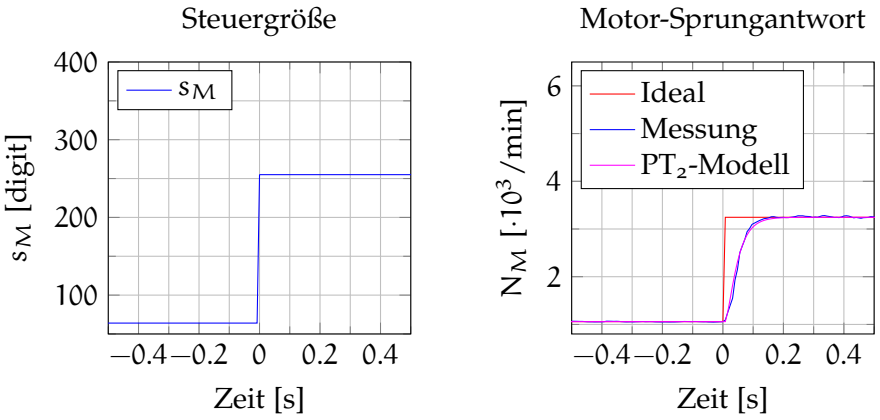


Abbildung 51: Dynamisches Antriebsmodell: Analyse durch Auswertung der Sprungantwort

in Reihe geschalteten PT_1-Gliedern aufgebaut werden. Entsprechend lautet die Übertragungsfunktion im Bildbereich

$$G_{(s)} = \frac{K}{(1 + T_1 s)(1 + T_2 s)} \tag{8.35}$$

mit der Verstärkung K und den beiden Zeitkonstanten des Systems T_1 und T_2. Mittels Minimierung der Fehlerquadrate zwischen realer Antriebsdrehzahl und Modell konnten die Parameter der Übertragungsfunktion bestimmt werden. Daraus ergibt sich:

$$G_{(s)} = \frac{2186.7}{(1 + 0.0112s)(1 + 0.028s)}. \tag{8.36}$$

Zusätzlich ist ein Totzeitglied in der Größe von einer Abtastperiode des Flugboards (8 ms) eingefügt worden. Diese Totzeit spiegelt das Ansprechverhalten der Motorsteuerungssoftware wieder. Das Übertragungsmodell ist in Abbildung 51 *rechts* in *magenta* dargestellt. Der mittlere quadratische Fehler zwischen den realen Antriebsdrehzahlen und dem PT_2-Modell beträgt 70 $^1/_{min}$ und hat damit einen Fehler von $\approx 2\,\%$ der Maximaldrehzahl.

8.1.3 *Flugplattform*

Neben den Sensoren und den Antrieben hat der mechanische Aufbau der Flugplattform einen bedeutenden Einfluss auf die Flugdynamik und auf das Regelverhalten des Autopiloten. Dabei spielen vor allem das Trägheitsmoment und der Strömungswiderstand der Flugplattform eine wichtige Rolle.

Trägheitsmoment und Massenmittelpunkt

Das Trägheitsmoment, auch Inertialmoment genannt, beschreibt in der Kinetik den Widerstand eines starren Körpers gegenüber einer

Änderung seiner Rotationsbewegung. Aufgrund dessen, dass die gesamte Flugplattform sehr detailliert als CAD-Modell von der Herstellerfirma erfasst ist, können aus diesem die Parameter für das Trägheitsmoment der Flugplattform $\mathbf{J}_F^b$ entnommen werden. Demzufolge ergibt sich:

$$\mathbf{J}_F^b = \begin{pmatrix} 0.021000 & -0.000062 & -0.000024 \\ -0.000062 & 0.020802 & -0.000045 \\ -0.000024 & -0.000045 & 0.037772 \end{pmatrix} \text{kg} \cdot \text{m}^2. \tag{8.37}$$

Der Massenmittelpunkt kann ebenfalls aus dem CAD-Modell abgeleitet werden und befindet sich, beschrieben durch den Vektor

$$\vec{l}_C^b = \begin{pmatrix} -0.107836 \\ -0.261584 \\ -21.343613 \end{pmatrix} \cdot 10^{-3}\,\text{m}\,, \tag{8.38}$$

vom geometrischen Mittelpunkt entfernt. Das Gesamtabfluggewicht der Flugplattform beträgt dabei 1.208 kg.

Strömungswiderstand

Der Strömungswiderstand ist hingegen deutlich aufwendiger zu bestimmen. Eine direkte Lösung konnte hierfür nicht erbracht werden, da ein Windkanal nicht zur Verfügung stand. Aus diesem Grund wurde auf die Dissertation [124] zurückgegriffen, in der auf Grundlage der Neigungswinkel der Flugplattform die Windgeschwindigkeit bestimmt werden sollte. Hierbei wurden Untersuchungen eines typgleichen MAVs im Windkanal vorgenommen, welche den Ausgangspunkt zur Lösung des Strömungswiderstandes in dieser Arbeit bilden. Hierzu wurden aus [124] die Versuchsergebnisse des Windkanalexperiments bezüglich der Windgeschwindigkeit zum Neigungswinkel in ein eigens entwickeltes Analysetool übertragen. Mit diesem können die Parameter des Strömungswiderstandskoeffizienten $\vec{C}_D$ und der wirkenden Flugplattformflächen $\vec{A}$ iterativ durch Fehlerminimierung bestimmt werden. In Abbildung 52 ist ein Vergleich der Windkanaldaten (in *magenta*) und der in dieser Arbeit simulierten Daten (in *grün* und in *blau*) dargestellt. Die Flugplattform wurde dabei in einem windkanaltypischen Szenario simuliert. Die Windstärke ist von 0.0 m/s bis auf 9.0 m/s mit einer Schrittweite von 0.5 m/s erhöht worden. Jeder Messpunkt ist hierbei 2 Sekunden gehalten worden. Zur Bestimmung der Neigungswinkel wurde das MAV in die automatische Positionshaltung mittels eines simulierten GPS-Signals versetzt. Somit regelte die Flugplattform die wirkende Widerstandskraft durch Anpassung des Neigungswinkels automatisch aus. Die Messungen sind jeweils bei 0° (Motor 1 zeigt nach vorne) und bei −45° (Kameranutzlast zeigt nach vorne) Ausrichtung (siehe Abbildung 19 auf Seite 83) zur wirkenden Windkraft durchgeführt worden. Somit konnte

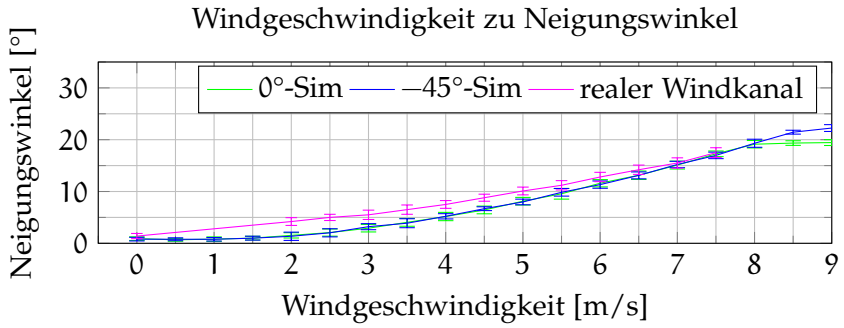


Abbildung 52: Modell des Strömungswiderstands: Vergleich von Windkanalmessungen mit Daten der Simulation

für das Produkt aus Strömungswiderstandskoeffizient und wirkender Fläche der jeweiligen körperfesten Raumachse der Vektor

$$\vec{C}_D \cdot \vec{A} = \begin{pmatrix} 0.112 \\ 0.112 \\ 0.099 \end{pmatrix} m^2 \tag{8.39}$$

bestimmt werden. Bedingt durch die mechanische Konstruktion der Flugplattform ist zusätzlich ein unterschiedlicher Strömungswiderstandskoeffizient $\vec{C}_D$ für die wirkenden Flächen angenommen worden. Für die x- und y-Ebene ist ein Halbzylinder mit Anströmung der konvexen Seite ($C_D = 1.20$) und für die z-Ebene ein Kreiszylinder ($C_D = 0.90$) zugrunde gelegt worden (siehe [85]). Dementsprechend ergeben sich rechnerisch die wirkenden Flächen zu:

$$\vec{A} = \begin{pmatrix} 0.093 \\ 0.093 \\ 0.110 \end{pmatrix} m^2. \tag{8.40}$$

Der mittlere Fehler zwischen dem hier modellierten Strömungswiderstand und den Ergebnissen von [124] aus dem Windkanal liegt bei 1.5° (RMSE) Abweichung im Neigungswinkel über den gesamten Modellierungsbereich hinweg. Dabei sind jedoch bei niedrigen Windgeschwindigkeiten höhere Abweichungen zwischen Simulation und realem Windkanal zu sehen. Hier wird vermutet, dass diese durch Messfehler entstanden sind. Zum einen stabilisierte das simulierte MAV die Position automatisch, wodurch reglerbedingte Fehler nicht verhindert werden konnten, zum anderen sind in [124] die Neigungswinkel des MAVs durch die flugplattformeigene IMU erfasst worden. Wie jedoch in Kapitel 8.1.1 gezeigt wurde, ist diese nicht unerheblichen systeminternen Fehlern unterworfen. Zudem liegt die minimale Winkelauflösung des verwendeten Beschleunigungssensorsystems bei lediglich 0.7°. In der Simulation wurden hingegen die idealen *Ground-Truth*-Daten verwendet. Trotz der Abweichungen zu den Er-

gebnissen aus dem Windkanal erscheinen die ermittelten Werte plausibel und bei einer rein qualitativen Bewertung verhält sich die simulierte Flugplattform bei Windeinfluss nahezu identisch zur realen. Der in Abbildung 52 auf Seite 125 dargestellte Unterschied zwischen den maximalen Windlasten der Flugplattform in Abhängigkeit von der Ausrichtung zum Windvektor ist durch die interne Regelung des Systems zu erklären. So kann die Flugplattform bei automatischer Positionshaltung um einen maximalen *Roll*- bzw. *Pitch*-Winkel von $\pm 20°$ verkippen (siehe Abbildung 52 *grüne* Kennlinie). Greift der Windvektor jedoch nicht parallel zur x- bzw. y-Achse, sondern um $\pm 45°$ um die z-Achse gedreht die Flugplattform an, kann diese die Gegenkraft sowohl durch ein Verkippen um die *Roll*- als auch um die *Pitch*-Achse mit maximal $\pm 20°$ ausgleichen. Der sich somit ergebende maximale Neigungswinkel entgegen der Windrichtung beträgt dann etwa $\pm 27°$, was zu einer höheren maximalen Windlast führt (siehe Abbildung 52 – *blaue* Kennlinie).

Abschließend ist zu den Abschnitten der Antriebs- und Flugplattformmodellierungen zu sagen, dass der Rotor, wie in Abschnitt 7.2.2 vorausgesetzt, starr mit dem Motor gekoppelt ist. Der Hersteller der Flugplattform stellt optional ein Schlaggelenk für die Rotoraufnahme zur Verfügung, wodurch der Anstellwinkel des Rotors bei Belastung verändert wird. Dieses Schlaggelenk wird im Rahmen dieser Arbeit nicht verwendet und daher auch nicht gesondert betrachtet.

8.2 ECHTZEIT-SIMULATION DER FLUGPLATTFORM

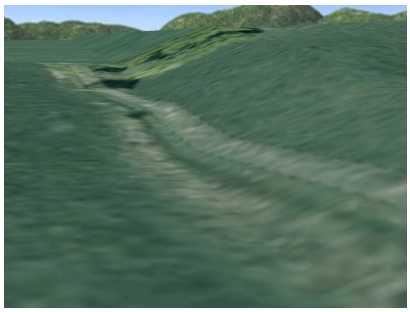
Nach der Implementierung der abgeleiteten Sensormodelle in die in Abbildung 17 auf Seite 70 gezeigte Simulationsstruktur wird der simulierte Quadrokopter seinem realen Pendant gegenübergestellt (siehe Abbildung 53). Zu diesem Zweck werden zwei Szenarien entwickelt. Im *Indoor*-Szenario erfolgt eine Bewertung der Simulationseigenschaften im direkt pilotierten Flug. Dementsprechend steuert der

(a) Reale Flugplattform beim *Hovern* im *Indoor*-Flugfeld

(b) Simulierte Flugplattform beim *Hovern* im *Indoor*-Flugfeld

Abbildung 53: Flugplattform *AR100B* – real zu simuliert

(a) Simuliertes *Indoor*-Flugfeld



(b) Simuliertes *Outdoor*-Flugfeld

Abbildung 54: Virtuelle Flugwelten

Pilot spezifische *Roll-*, *Pitch-* oder Höhenänderungsmanöver. Im *Outdoor*-Szenario erfolgt anhand einer automatisch abzufliegenden Wegpunkttrajektorie eine quantitative Analyse des simulierten Systems. Neben der realitätsgetreuen Abbildung aller Systemeigenschaften erfolgt zudem die photorealistische Simulation der Umgebung, d. h. der Flughalle oder des Außengeländes. Somit wird ein realistisches visuelles Feedback ermöglicht. In Abbildung 54 sind die modellierten Testgelände dargestellt.

8.2.1 *Indoor-Szenario*

Die Simulation der Flugplattform auf dem *Indoor*-Flugfeld ist aus verschiedenen Gründen sinnvoll. Einer der bestimmenden Faktoren ist beispielsweise, dass hier ohne die Einflüsse von Witterungsbedingungen unterschiedliche Flugtests mit den realen Systemen durchgeführt werden können. Zur Validierung der in den vorherigen Abschnitten hergeleiteten Modelle soll nun im *Indoor*-Flugtest das reale System dem simulierten gegenübergestellt werden. Dazu werden der Flugplattform vom Piloten mittels eines Joysticks spezifische Flugbewegungen vorgegeben. Sämtliche Steuerkommandos, Sensordaten und Motorsteuerungswerte werden in Echtzeit auf der Flugplattform mit-

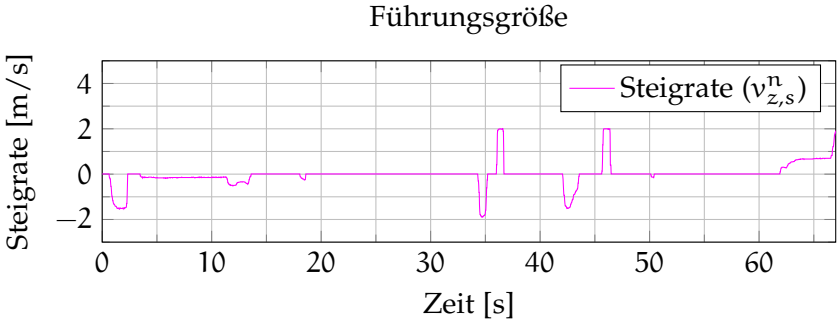


Abbildung 55: *Indoor*-Flugtest: Führungsgröße der Höhenregelung durch Vorgabe des Piloten

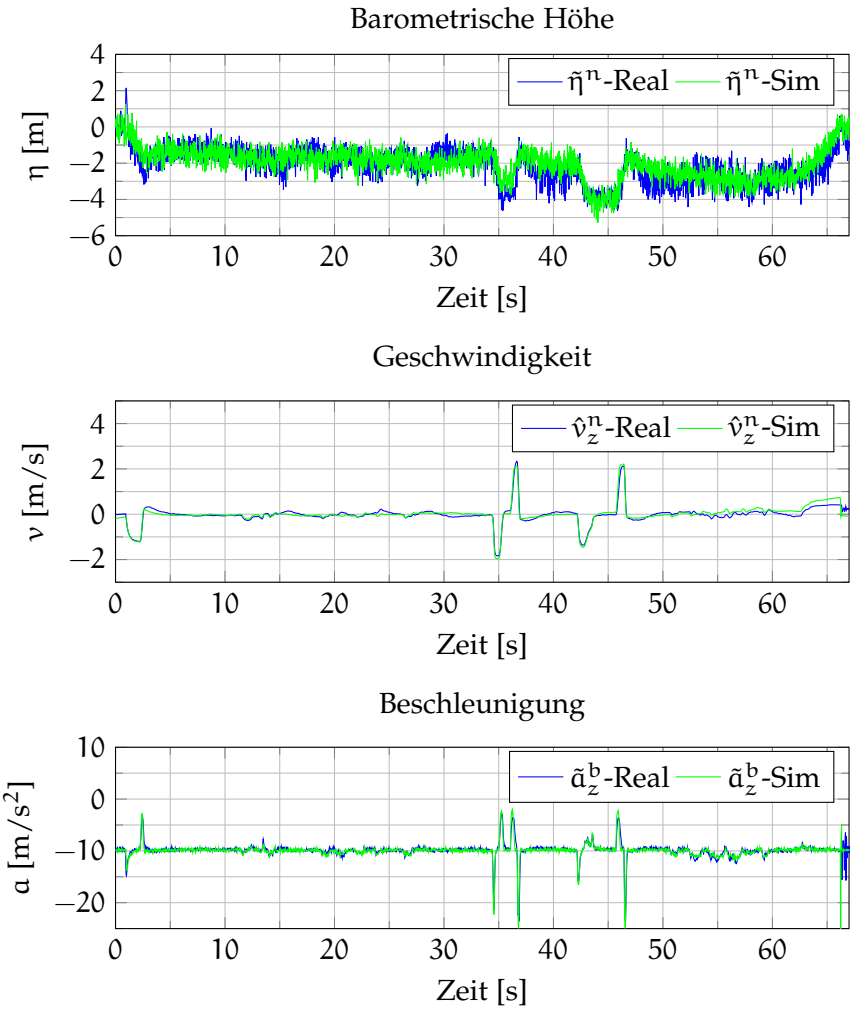


Abbildung 56: *Indoor*-Flugtest: Zustandsgrößen der Höhenregelung im Vergleich von der realen zu der simulierten Flugplattform

geschrieben. Daraufhin werden die gespeicherten Steuerungsbefehle an die simulierte Flugplattform zeitsynchronisiert gesendet und vom virtuellen System verarbeitet. Idealerweise sollte das virtuelle System äquivalente Sensordaten erzeugen und somit ein gleichwertiges Flugverhalten aufweisen. Zur Analyse werden vom Piloten die Steig- bzw. Sinkgeschwindigkeit, der *Roll*- sowie der *Pitch*-Winkel als Führungsgröße vorgegeben. Der gesamte Testflug hat eine Dauer von 67 Sekunden.

Die Analyse beginnt mit der Höhenregelung des Systems. Hierzu sind in Abbildung 55 auf Seite 127 die Eingaben des Piloten zur Höhensteuerung abgebildet. Analog zur Führungsgröße sind die jeweiligen Sensorinformationen in Abbildung 56 dargestellt. Darin sind, von *oben* nach *unten*, die Sensordaten des Barometers ($\tilde{\eta}^n$), die systemintern geschätzte Vertikalgeschwindigkeit ($\hat{v}_z^n$) und die gemessenen Werte des Beschleunigungssensors ($\tilde{a}_z^b$) in der z-Achse aufgezeigt. Zum Vergleich zwischen realem und simuliertem System sind jeweils

die Sensorinformationen des realen Systems in *blau* und die der simulierten Flugplattform in *grün* zueinander aufgetragen. Demnach gibt der Pilot bei Sekunde 1 eine Steiggeschwindigkeit von $-1.5\,\mathrm{m/s}$ vor (*Take-Off*). Die Flugplattform steigt entsprechend der barometrischen Höhe auf $\approx -2\,\mathrm{m}$ relativ zum Startpunkt. Des Weiteren steuert der Pilot die Flugplattform beispielsweise in den Sekunden 34 bis 38 mit maximaler Steig- bzw. Sinkrate von $\pm 2\,\mathrm{m/s}$. Wie abzulesen ist, steigt das System kurzzeitig auf $-3.8\,\mathrm{m}$ und sinkt danach wieder auf die Ausgangshöhe zurück. Ab Sekunde 62 beginnt der Pilot die Landung, so dass die Flugplattform wieder bis auf den Boden sinkt. Die Bodenberührung bei der Landung (*Touch-Down*) ist in den Beschleunigungssensorwerten bei Sekunde 66 deutlich erkennbar.

Darüber hinaus sind in Abbildung 57 und Abbildung 58 die Zustandsgrößen der Lageregelung dargestellt. Die Führungsgröße für

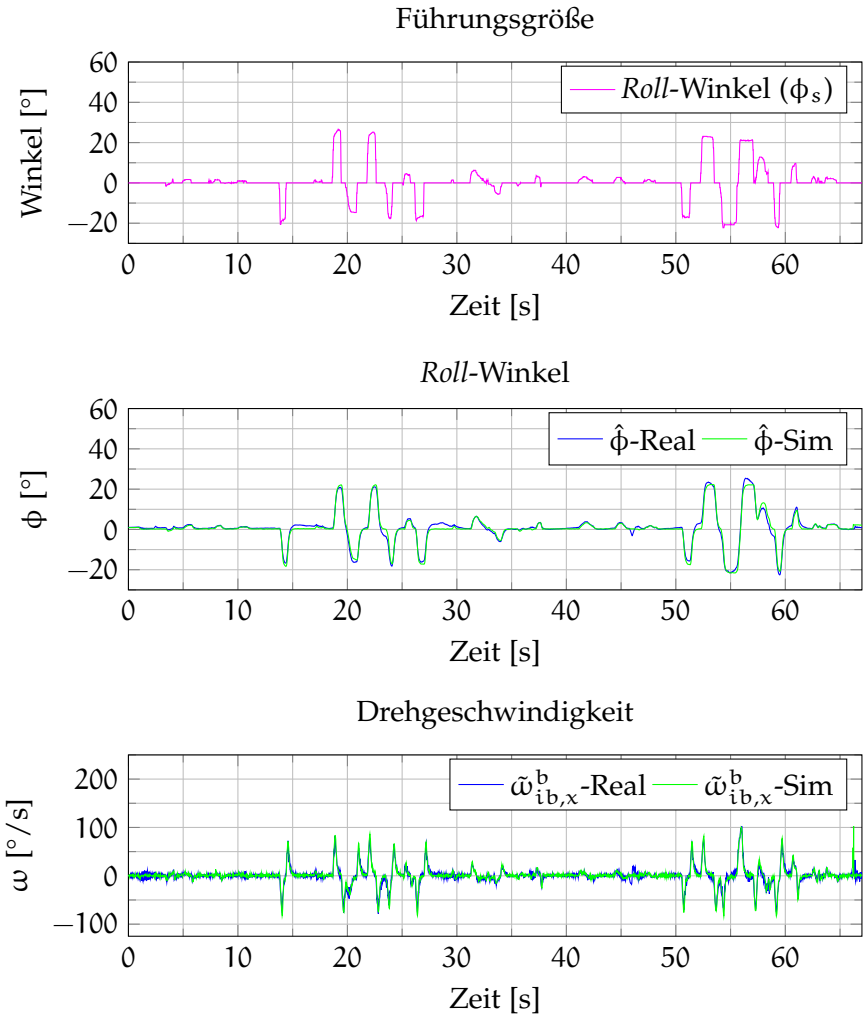


Abbildung 57: *Indoor*-Flugtest: Zustandsgrößen der Lageregelung im Vergleich von der realen zu der simulierten Flugplattform (*Roll*-Winkel)

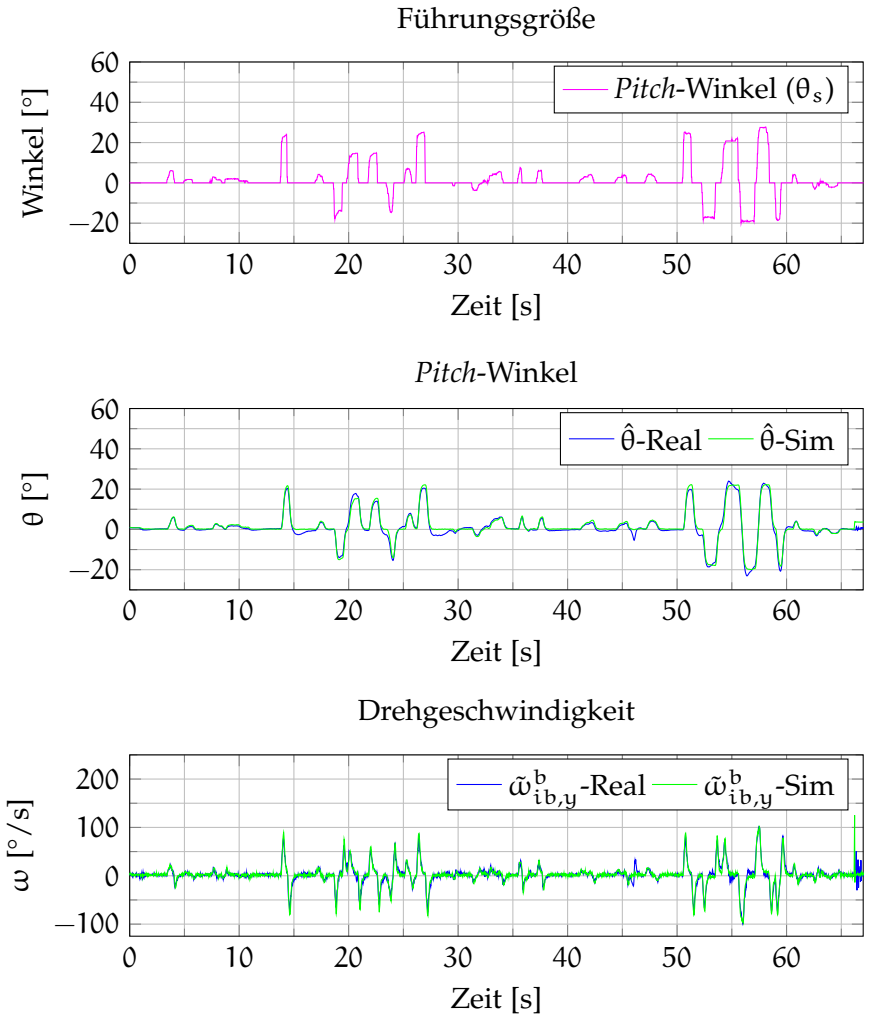


Abbildung 58: *Indoor*-Flugtest: Zustandsgrößen der Lageregelung im Vergleich von der realen zu der simulierten Flugplattform (*Pitch*-Winkel)

den *Roll*-Winkel des MAVs wird dabei in Abbildung 57 *oben* in *magenta* abgebildet. Der mittlere Plot zeigt, im Vergleich zu der vom Piloten gesteuerten Größe, die von der Flugplattform umgesetzte *Roll*-Bewegung anhand des systemintern geschätzten *Roll*-Winkels ($\hat{\phi}$). Im untersten Plot ist die gemessene Drehgeschwindigkeit ($\tilde{\omega}^{b}_{ib,x}$) um die körperfeste x-Achse aufgetragen. Korrespondierend zur *Roll*-Bewegung zeigt Abbildung 58 die entsprechenden Zustandsgrößen der Lageregelung für die *Pitch*-Bewegung. Die jeweiligen Führungsgrößen belegen, dass das MAV phasenweise sowohl mit geringer Bewegungsdynamik, beispielweise in der Zeit von Sekunde 40 bis zur Sekunde 50, als auch mit den maximalen Lagewinkeln von $\pm 22°$, beispielsweise in der Zeit von Sekunde 50 bis zur Sekunde 60, mit Drehraten bis zu $100°/s$ gesteuert worden ist. Analog zur Höhenregelung sind auch hier jeweils in *blau* und *grün* die Lageinformationen vom realen zum simulierten System gegenübergestellt worden.

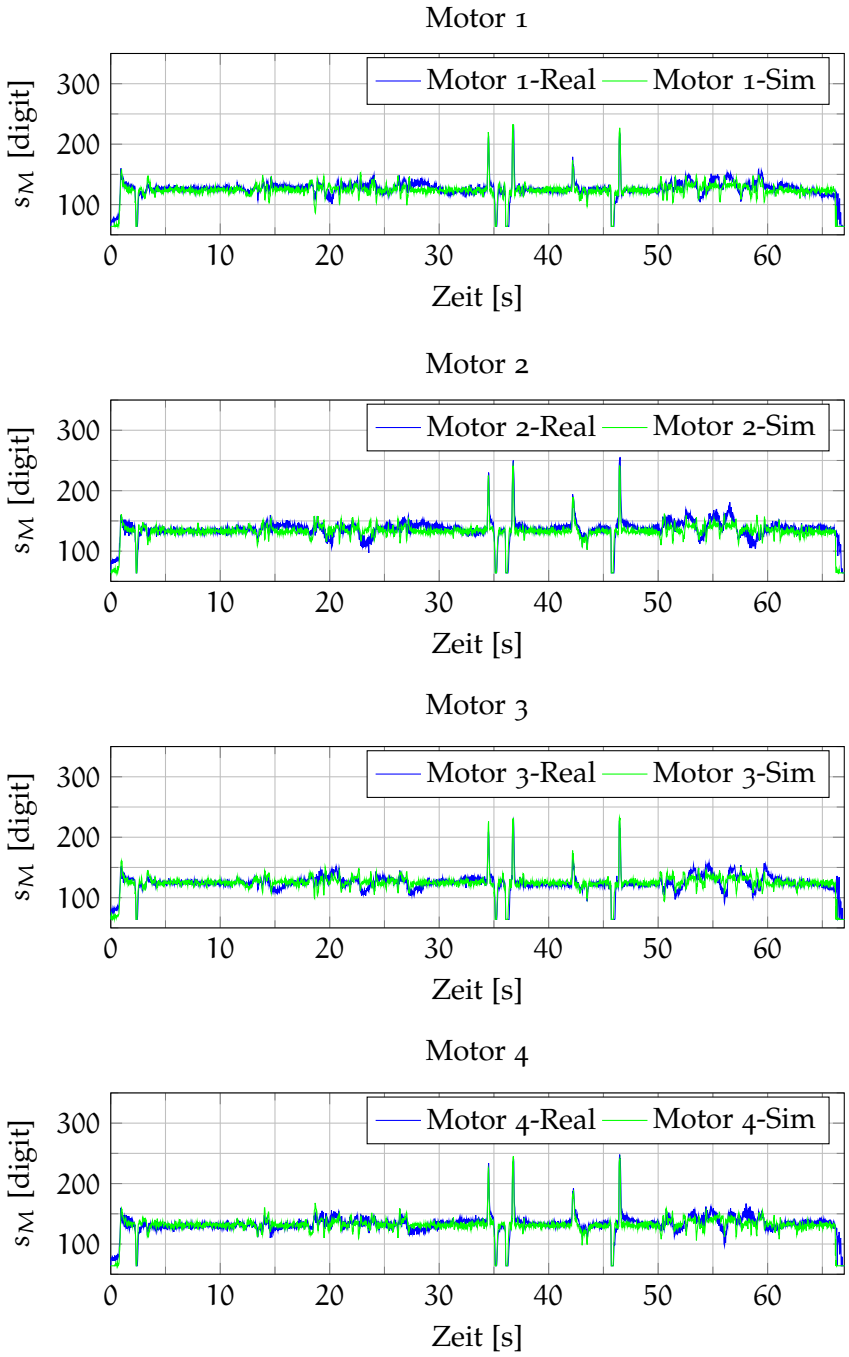


Abbildung 59: *Indoor*-Flugtest: Steuergrößen der Motoren im Vergleich von der realen gegenüber der simulierten Flugplattform

Als letzter Datensatz sind in Abbildung 59 die Motorsteuergrößen (s_M) des gesamten Fluges dargestellt. Die Steuerungswerte der realen Flugplattform sind in *blau* gehalten und die der simulierten in *grün* aufgetragen. Hierbei sind besonders die Steig- und Sinkphasen der Flugplattform in den Sekunden 34 bis 38 und 42 bis 48 deutlich in den Steuergrößen zu erkennen. In diesen Phasen werden die Motoren mit ihrem Maximalwert von 255 digit bzw. mit ihrem Minimalwert

von 64 digit angesteuert. Die *Roll*- bzw. *Pitch*-Bewegungen bewirken indes nur geringe Änderungen in den Steuergrößen der Motoren.

Bewertung der Ergebnisse

Die Auswertung der Sensorinformationen und der Motorsteuergrößen zeigt, dass die Modellierungseigenschaften der simulierten Flugplattform eindeutig den Eigenschaften der realen entsprechen. So ist in Abbildung 56 auf Seite 128 deutlich erkennbar, dass die kommandierten Höhenänderungen in nahezu identischer Art und Weise von beiden Systemen umgesetzt werden. Ebenso werden Lageänderungen der realen Flugplattform sehr präzise von der simulierten Flugplattform reproduziert (siehe Abbildung 57 und 58 auf den Seiten 129 und 130). Dies belegen nicht nur systemintern geschätzte Größen, wie die Vertikalgeschwindigkeit oder *Roll*- bzw. *Pitch*-Winkel,

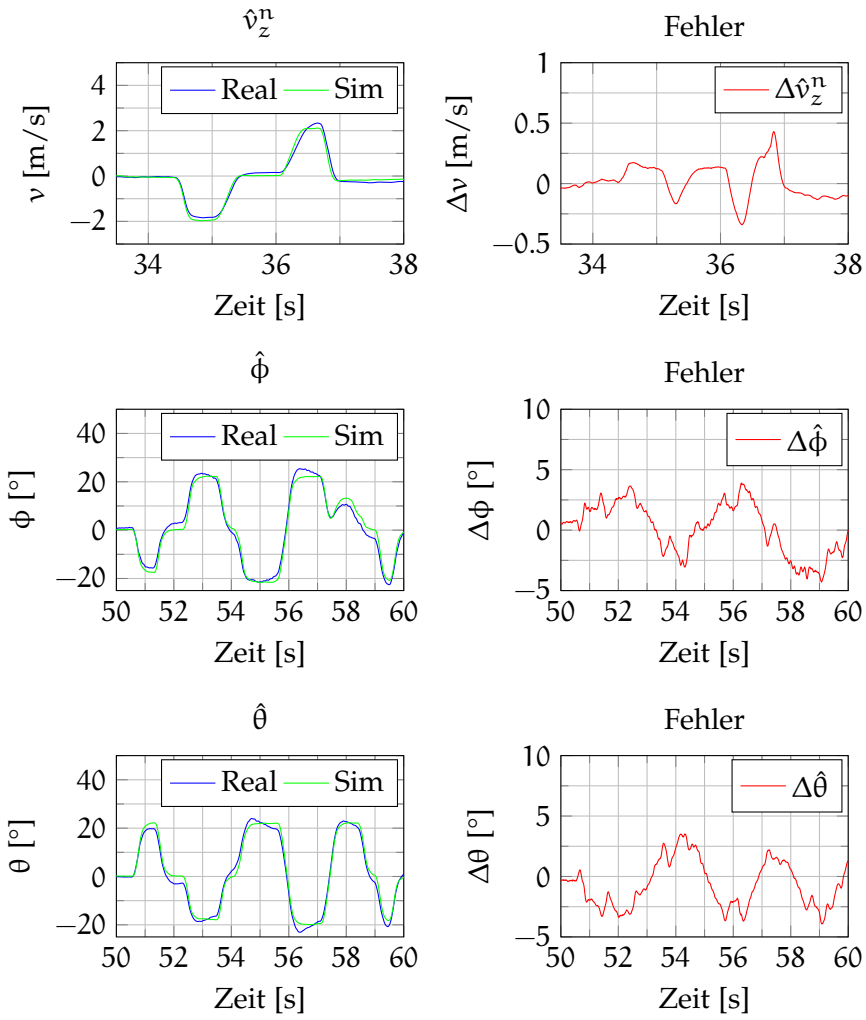


Abbildung 60: *Indoor*-Flugtest: Detaillierte Auswertung der signifikanten Zustandsgrößen

sondern ebenfalls die direkt gemessenen Sensorinformationen, wie die barometrische Höhe, Beschleunigungen oder Drehraten. Die bestimmten Motormodelle fügen sich zudem sehr gut in den simulierten Regelkreis ein. Belegt wird dies in Abbildung 59 auf Seite 131 bei Betrachtung der Motorsteuergrößen im Vergleich der Systeme (real zu simuliert). Leichte Abweichungen zwischen beiden Systemen sind dennoch erkennbar. Dazu werden exemplarisch drei Systemmanöver genauer untersucht. Zuerst wird die Höhenregelung im Bereich von Sekunde 34 bis 38 im Detail analysiert. Hier gibt der Pilot als Systemführungsgröße anfangs eine Steigrate von $v^n_{z,s} = -2\,\mathrm{m/s}$ und kurz darauf eine Sinkrate von $v^n_{z,s} = 2\,\mathrm{m/s}$ vor. Wird nun die systemintern geschätzte Vertikalgeschwindigkeit $\hat{v}^n_z$ vergrößert dargestellt (siehe Abbildung 60 *oben*), zeigt sich jeweils an den Maxima ein geringes Über- bzw. Unterschwingen mit $< 0.4\,\mathrm{m/s}$ bei der realen Flugplattform. Ansonsten weisen reale und simulierte Flugplattform ein nahezu identisches Verhalten auf, wodurch auf eine übereinstimmende Systemdynamik der MAVs geschlossen werden kann. Über den gesamten Flug hinweg betrachtet, beträgt der mittlere Fehler zwischen realem und simuliertem System lediglich $0.1\,\mathrm{m/s}$ (RMSE). Analog werden ebenfalls die Lagemanöver in den beiden unteren Abbildungen vergrößert. Auch hier wird die Führungsgröße für *Roll*- bzw. *Pitch*-Winkel innerhalb kurzer Zeit mehrfach in den Maximalbereich von $\pm 22°$ gesteuert. Beim Vergleich von realer und simulierter Flugplattform ist das dynamische Verhalten konvergent. Lediglich an den Extrema weisen die Messwerte geringe Abweichungen auf. Der Fehler ist in Abbildung 60 *rechts Mitte* und *rechts unten* abgebildet. Der mittlere Fehler über die Zeit des gesamten Flugs beträgt lediglich 1.21° (RMSE) für den *Roll*- und 1.24° (RMSE) für den *Pitch*-Winkel. Somit belegen die Messdaten beider Systeme, dass die modellierten Sensoren, Antriebe und flugdynamischen Kenngrößen der virtuellen Plattform sehr gut mit ihrem realen Pendant übereinstimmen. Ebenso beweisen die übereinstimmenden Steigungen in den geschätzten Sys-

(a) Realer *Indoor*-Flugtest

(b) Simulierter *Indoor*-Flugtest

Abbildung 61: *Indoor*-Flugtest: Qualitativer Vergleich von der realen (a) zu der simulierten (b) Flugplattform bei Lagewinkelvorgabe

temgrößen eine gleichwertige Systemdynamik. Dies ist nicht zuletzt der präzisen Modellierung des dynamischen Verhaltens der Antriebe geschuldet.

Abschließend ist in Abbildung 61 auf Seite 133 für den *Indoor*-Flugtest eine grafische Gegenüberstellung einer ausgewählten Flugsequenz zu finden. Hierbei ist die Flugplattform aus der Ruhelage (*1. Szene*) mit maximaler Lageführungsgröße nach rechts gesteuert worden (*2. und 3. Szene*). Danach kehrt die Flugplattform selbstständig in die Ruhelage zurück (*4. und 5. Szene*). In Abbildung 61a ist diese Testsequenz mit der realen und in 61b mit der simulierten Flugplattform aufgenommen worden. Somit ist auch qualitativ ein nahezu identisches Flugbild dargestellt.

Nachdem die *Low-Level*-Regelung des Systems untersucht wurde, soll im Folgenden das simulierte Systemverhalten in Bezug auf die *High-Level*-Funktionen durch das automatisierte Abfliegen einer vorgegebenen Route auf Basis von GPS-Informationen analysiert werden.

8.2.2 *Outdoor-Szenario*

Zur Analyse der *High-Level*-Funktionalität der Flugplattform in der Simulation wird eine 3D-Wegpunktmission geplant. Diese ist in Abbildung 62 dargestellt und besteht aus zehn Wegpunkten. Zu Beginn steigt das MAV – relativ zum Startpunkt – um 10 m. Danach fliegt es voll automatisch ein 30 m mal 30 m großes Quadrat, bestehend aus 4 Wegpunkten, ab. An jedem Wegpunkt stoppt das Fluggerät und

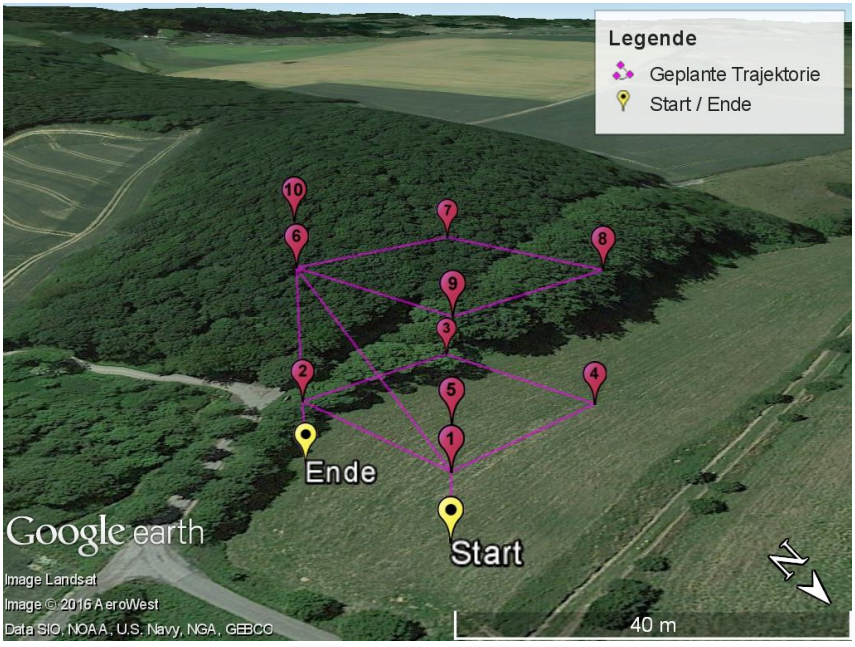


Abbildung 62: *Outdoor*-Flugtest: Planung der automatischen Wegpunktnavigation in 3D

verweilt dort eine Sekunde, bis es zum nächsten Punkt fliegt. Die Fluggeschwindigkeit von Wegpunkt zu Wegpunkt soll 3 m/s betragen. Am Ende des ersten Quadrats steigt das Fluggerät von Wegpunkt 5 nach 6 um 20 m auf 30 m über den Startpunkt. Von dort fliegt die Flugplattform abermals ein Quadrat mit den gleichen Parametern wie zuvor. Angekommen an Wegpunkt 10, sinkt das Fluggerät um 30 m zurück auf die Höhe des Startpunktes. Die Flugzeit beträgt ≈ 200 s. Diese Wegpunktmission wird dann in exakt gleicher Form sowohl in das reale als auch in das simulierte MAV geladen und nachfolgend von beiden Systemen vollautomatisch abgeflogen. Zum Vergleich der Systeme werden die Messgrößen der drei direkt abhängigen Sensorsysteme betrachtet: GNSS für die Position, Baro-Altimeter für die Höhe über Grund sowie Magnetometer für das *Heading*. Um kleinere witterungsbedingte Einflüsse in den Daten und daraus re-

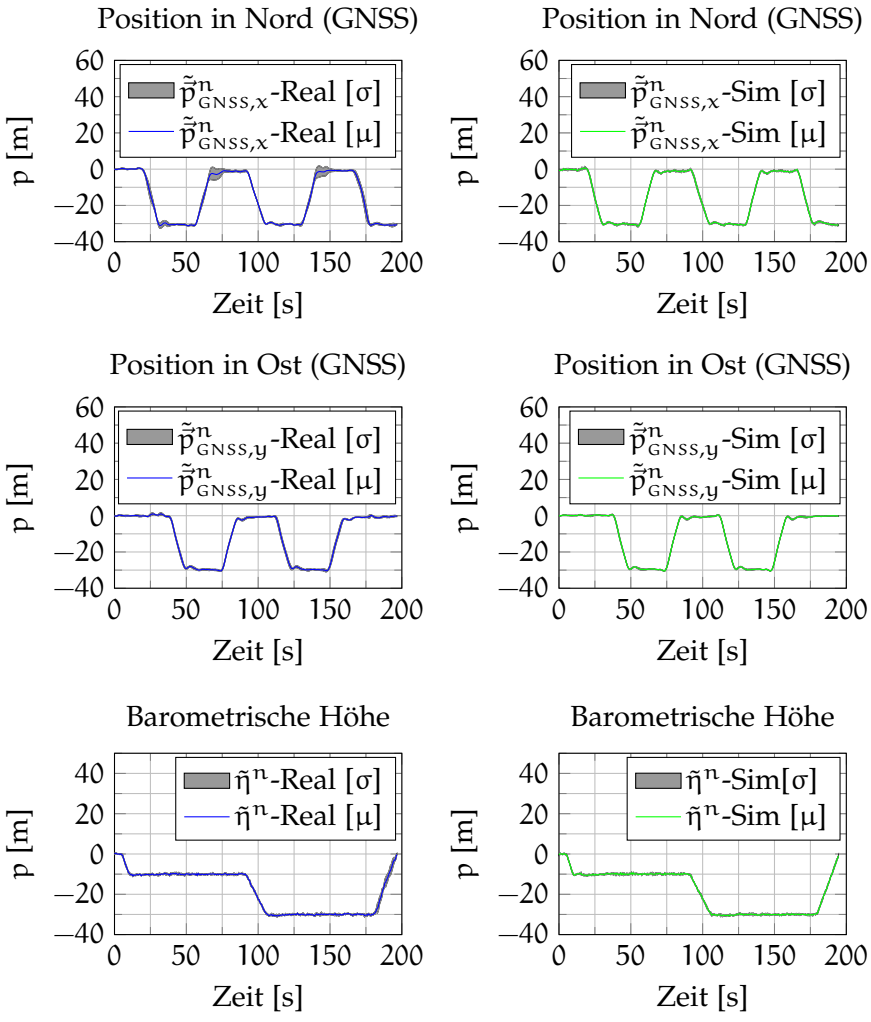


Abbildung 63: *Outdoor*-Flugtest: Vergleich der Positionsdaten aus GNSS- und Baro-Altimeter-Sensor von der realen zu der simulierten Flugplattform

sultierende Abweichungen zu reduzieren, ist die geplante Strecke, sowohl vom realen als auch vom simulierten System, am selben Tag jeweils dreimal hintereinander geflogen worden. Aus den drei Flügen ist daraufhin der Mittelwert und die Standardabweichung in jedem Datenpunkt ermittelt worden. Zuerst wird die systemintern berechnete lokale Positionslösung, basierend auf der Positionsangabe des GNSS-Empfängers und des Baro-Altimeters, untersucht. Die Darstellung erfolgt im Navigationskoordinatensystem entsprechend der Herleitung in Kapitel 4. Analog dazu sind in Abbildung 63 *oben/Mitte* auf Seite 135 die in das Navigationskoordinatensystem transformierte Positionslösung des GNSS-Empfängers und in Abbildung 63 *unten* die Sensordaten des Baro-Altimeters aufgezeigt. Es wird an dieser Stelle aus Gründen der Darstellbarkeit darauf verzichtet, die Standardabweichungen als Fehlerbalken zu zeichnen. Stattdessen wird eine *graue*, semitransparente Schattierung den Mittelwerten hinterlegt.

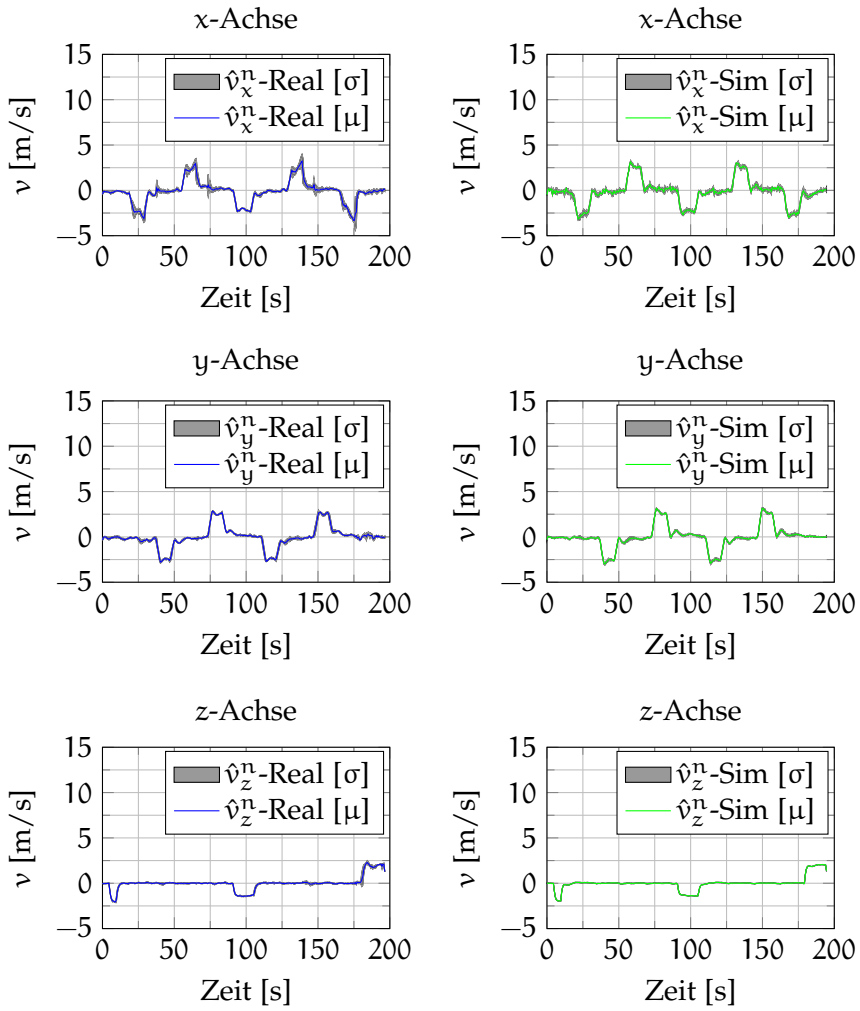


Abbildung 64: *Outdoor*-Flugtest: Vergleich der Geschwindigkeitsinformationen von der realen zu der simulierten Flugplattform

Zudem werden aufgrund der hohen Übereinstimmung der Datensätze in den folgenden Abbildungen die Daten der realen Flugplattform in *blau* und die Daten der simulierten in *grün* nebeneinander abgebildet. Den Darstellungen ist deutlich zu entnehmen, dass beide Systeme ein sehr äquivalentes Verhalten bezüglich der erfassten Position zeigen. Geringe Abweichungen sind in der x-Achse zu den Zeitpunkten $t = 30\,s$, $t = 70\,s$ sowie $t = 145\,s$ erkennbar. Ebenfalls zeigt sich bei der realen Plattform in diesen Zeitpunkten eine deutlich höhere Standardabweichung der Position. Dies lässt darauf schließen, dass sich zu diesen Zeitpunkten die von außen auf die Flugplattform wirkenden Faktoren während der Flüge geändert haben. Eine derartige Abweichung ist beispielsweise in der Simulation reproduzierbar, wenn sich der Windvektor kurzzeitig im Betrag ändert und demzufolge eine Windböe auf das System wirkt.

Analog zur Position ist in Abbildung 64 die systemintern geschätzte Geschwindigkeit in allen drei Achsen dargestellt. Auch hier ist eine sehr gute Übereinstimmung zwischen realer und simulierter Flugplattform zu sehen. Wie bei der Position sind in der x-Achse der Geschwindigkeit zu den genannten Zeitpunkten ebenfalls leichte Unterschiede wahrnehmbar. Dies ist aufgrund der vorherigen Erläuterung nur konsequent.

Bei der Wegpunktnavigation ist neben der Bestimmung der Position auch die Bestimmung des Kurses (*Heading*) eine elementare Bedingung. Dies wird bei der betrachteten Flugplattform durch ein Magnetometer vorgenommen. Die realen und simulierten Messwerte der drei körperfesten Sensorachsen sind in Abbildung 65 auf Seite 138 dargestellt. Auf derselben Seite ist in Abbildung 66 die absolute Magnetfeldintensität während des Testflugs aufgezeigt. Prinzipiell stehen die Sensormesswerte des simulierten Systems in guter Näherung zur realen Flugplattform. Abweichungen sind jedoch in der absoluten Magnetfeldintensität erkennbar. Hier ist besonders auffällig, dass die Abweichung achsenabhängig zu sein scheint. Dabei entspricht der Mittelwert der absoluten Magnetfeldintensität der realen Flugplattform in etwa dem theoretisch berechneten absoluten Feld von $|\vec{h}^n| = 49.0089\,\mu T$. Die positiven und negativen Abweichungen von diesem Mittelwert korrelieren dabei direkt mit den Minima und Maxima der y-Achse des Magnetometers. Deshalb ist im Besonderen die Abweichung der y-Achse nochmals durch die Simulation von weiteren Fehlereinflüssen untersucht worden. Dabei konnte ein zusätzlicher Biasfehler des realen Magnetometers während des Flugs gegenüber dem Modell aus Abschnitt 8.1.1 von $-2.368\,\mu T$ bezüglich der y-Achse und $-0.384\,\mu T$ bezüglich der z-Achse nachgewiesen werden. Dieser zusätzliche Bias wird als Vektor

$$\vec{b}_{h,I} = \begin{pmatrix} 0.0 \\ -2.368 \\ -0.384 \end{pmatrix} \mu T \tag{8.41}$$

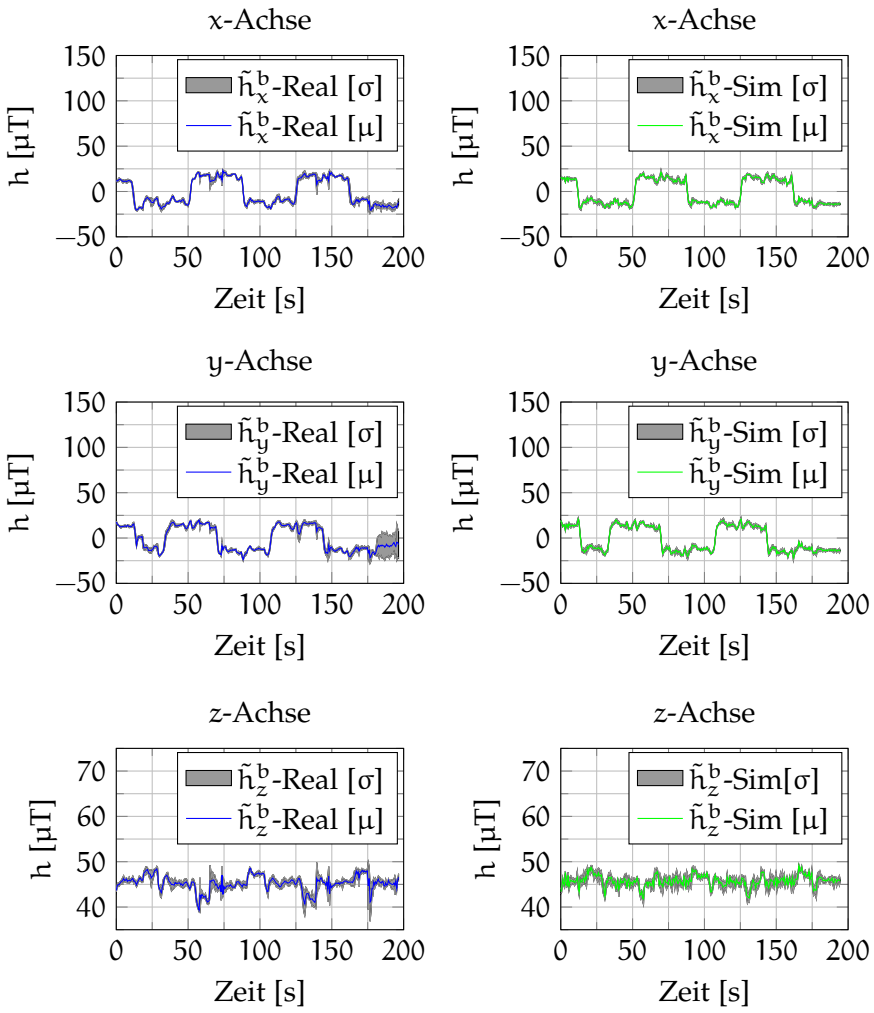


Abbildung 65: *Outdoor*-Flugtest: Gegenüberstellung der Magnetfeldsensordaten von der realen zu der simulierten Flugplattform

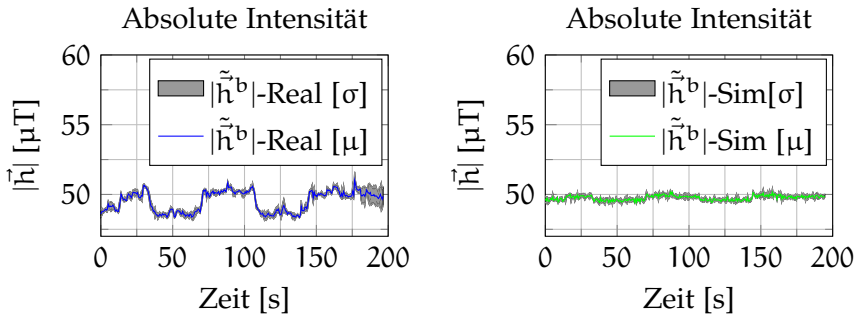


Abbildung 66: *Outdoor*-Flugtest: Gegenüberstellung der absoluten Magnetfeldintensität von der realen zu der simulierten Flugplattform

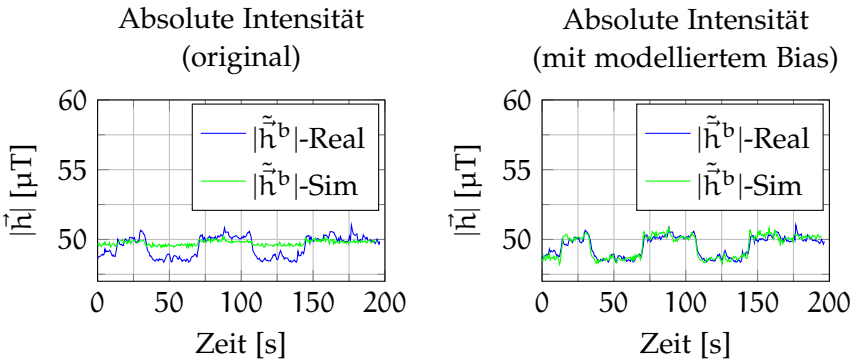


Abbildung 67: *Outdoor*-Flugtest: Gegenüberstellung der absoluten Magnetfeldintensität von der realen zu der simulierten Flugplattform nach der Modellierung des stromabhängigen Magnetfeldsensorbias

in das Magnetometermodell eingefügt und daraufhin auf die simulierten Messdaten angewendet. Im Vergleich des vorherigen simulierten absoluten Magnetfeldes (siehe Abbildung 67 *links*) zum entsprechend korrigierten Modell (siehe Abbildung 67 *rechts*) wird deutlich, dass nun ebenfalls die modellierten Magnetometerdaten sehr gut den realen entsprechen. Auch die Ursache des Biasfehlers konnte nachgewiesen werden. So werden bei der im Abschnitt 7.1 gezeigten Magnetometermodellierung die realen Magnetometermesswerte auf einen Ellipsoiden eingepasst (*Ellipsoid Fitting*) und, gemäß Kapitel B.3 ab Seite 228, die Parameter für das Magnetometermodell (Skalierung, Bias etc.) bestimmt. Die Messwerte des realen Systems mussten jedoch im quasistationären Zustand, d. h. bei nicht laufenden Antrieben, aufgenommen werden, da im anderen Fall die Gefahr von Verletzungen durch die drehenden Rotoren zu groß gewesen wäre. Deshalb sind in diesem Magnetometermodell zwar die Hart- bzw. Weicheisen (*Hard- and Soft-Iron*)-Effekte des Sensorsystems und der Flugplattform erfasst, jedoch nicht die Abweichungen aufgrund des externen elektromagnetischen Störfelds, das auf der elektrischen Leistungsaufnahme der Antriebe und dem damit verbundenen Stromfluss basiert. Dieser Störeinfluss wirkt sich als zusätzlicher Biasvektor $\vec{b}_{h,I}$ auf die Sensormesswerte aus. Somit konnte dieser Einflussfaktor nachträglich, auf Basis der realen Flugversuche, bestimmt und im Weiteren in das Modell implementiert werden. Mit den Sensormesswerten des Magnetometers wird folgend der *Yaw*-Winkel (*Heading*) der Flugplattform bestimmt. Hierzu wird das Magnetometer mit den Informationen aus der IMU zuerst lagekompensiert. Anschließend wird aus den kompensierten x- und y-Komponenten der *Yaw*-Winkel errechnet. Dieser ist auf Seite 140 in Abbildung 68 *oben links* für das reale in *blau* und *rechts* daneben für das simulierte System in *grün* dargestellt. Bezüglich des *Yaw*-Winkels verhalten sich beide Systeme annähernd äquivalent. Bei den realen Messdaten gibt es allerdings zu den Zeitpunkten $t = 14\,s$,

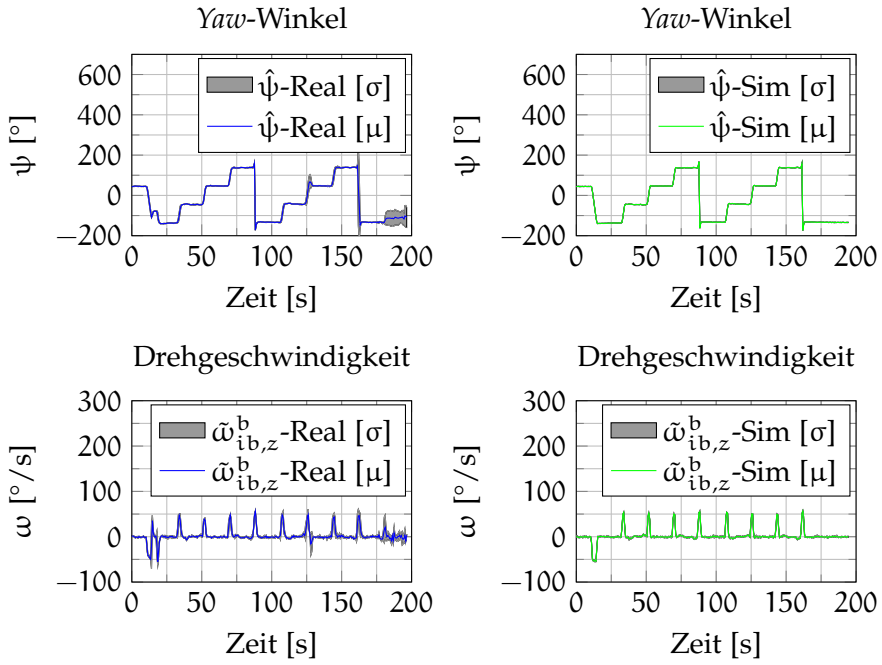


Abbildung 68: *Outdoor*-Flugtest: Gegenüberstellung des *Yaw*-Winkels und der Drehgeschwindigkeit der realen zu der simulierten Flugplattform

$t = 125\,s$ und $t = 180\,s$, sowohl im *Yaw*-Winkel als auch in der gemessenen Drehgeschwindigkeit der *z*-Achse, drei markante Stellen:

ZEITPUNKT $t = 14\,s$: Bei Ankunft des Systems im Wegpunkt 1 hat dieses einen *Yaw*-Winkel von 45°. Gemäß der Wegpunktplanung soll sich die Flugplattform daraufhin um 180° auf den *Yaw*-Winkel von −135° drehen und dann auf den zweiten Wegpunkt zufliegen. Dies belegen die Messdaten der simulierten Flugplattform. Das Verhalten der realen Flugplattform weist jedoch ein abweichendes Erscheinungsbild auf. So fängt die reale Flugplattform geplanterweise die Drehung zum zweiten Wegpunkt ab Sekunde 14 an, dreht jedoch lediglich bis zu einem *Yaw*-Winkel von −115° und dreht daraufhin mit einer Drehgeschwindigkeit von $\approx 45°/s$ in entgegengesetzter Richtung auf einen Winkel von −75° zurück. Dort verweilt sie für $\approx 3\,s$. Erst danach setzt die Flugplattform die Drehung zum geplanten *Yaw*-Winkel von −135° fort.

ZEITPUNKT $t = 125\,s$: Zum Zeitpunkt $t = 125\,s$ ereignet sich, ebenfalls nur bei der realen Flugplattform, eine weitere Abweichung von der geplanten Trajektorie. Hier hat das MAV den Wegpunkt 7 erreicht. Im nächsten Schritt soll eine Änderung des *Yaw*-Winkels von −45° um 90° auf 45° stattfinden. Die simulierte Flugplattform setzt dies entsprechend richtig um, wie die Messdaten zeigen. Die reale Flugplattform beginnt die Drehung ent-

sprechend der Planung, dreht jedoch deutlich über den geplanten Winkel hinweg und korrigiert diesen Fehler erst $\approx 2\,s$ später. Ebenfalls steigt die Standardabweichung sowohl beim *Yaw*-Winkel als auch bei der Drehgeschwindigkeit deutlich an. Um dies genauer zu analysieren, werden die drei Einzelflüge untersucht. Dabei weisen der erste und der dritte Flug jeweils das geplante, und damit richtige Flugverhalten auf. Beim zweiten Flug dreht das MAV jedoch auf einen *Yaw*-Winkel von 105° anstatt auf die geplanten 45°. Erst $\approx 2\,s$ später korrigiert die Flugplattform die fehlerhafte Ausrichtung.

ZEITPUNKT $t = 180\,s$: Die letzte Abweichung im Flugverhalten der realen Plattform ist am Ende des Fluges ab Sekunde 180 erkennbar. Hier sollte der *Yaw*-Winkel bei −135° liegen, wie es ebenfalls die simulierte Plattform zeigt. Die reale Flugplattform hat jedoch einen mittleren Winkel von −115°. Werden ebenfalls die Einzelflüge diesbezüglich ausgewertet, zeigt sich der Grund für die hohe Standardabweichung. Zwei der drei Flüge wiesen den richtigen Winkel von −135° auf. Der erste Flug jedoch weist lediglich einen *Yaw*-Winkel von −75° auf.

Die entscheidende Frage ist somit, warum die simulierte Flugplattform dieses Fehlverhalten der realen Flugplattform in keiner der Flüge aufweist, obwohl die Software des Autopiloten und somit die automatische Wegpunktnavigation identisch ist. Auffällig bei den Fehlern der realen Flugplattform ist allerdings der relative Winkelfehler.

Im ersten Zeitpunkt ($t = 14\,s$) beträgt die geplante Ausrichtung der Flugplattform −135°. Diese richtet sich jedoch in allen drei Flügen auf −75° aus. Das entspricht einer Abweichung von 60°.

Im zweiten betrachteten Zeitpunkt ($t = 125\,s$) absolviert die reale Flugplattform zwei der drei Flüge richtig. Im zweiten Flug jedoch steuert das MAV einen fehlerhaften *Yaw*-Winkel von 105° anstatt der geplanten 45° an. Die Differenz beträgt auch hier 60°.

Im letzten beschriebenen Zeitpunkt ($t = 180\,s$) weist nur einer der drei Flüge ein fehlerhaftes Verhalten auf. Hier beträgt der tatsächliche Kurswinkel −75° anstatt der geplanten −135°. Auffallend ist auch hier das Delta von 60°. Dieser jeweils identische Ausrichtungsfehler legt nahe, dass es sich jeweils um denselben Berechnungsfehler im Navigationsrechner des Autopiloten handelt.

Um diesen Sachverhalt zu untersuchen, werden mittels der Simulation verschiedene *Debug*-Informationen in den Quellcode des Autopiloten implementiert. Diese enthalten den vom Autopiloten geplanten Ausrichtungswinkel ψ_s, den tatsächlichen *Yaw*-Winkel $\hat{\psi}$ sowie die Phase $P_{j,\psi}$ der Abarbeitung im Navigationsrechner. Diese Informationen werden über den gesamten Flugverlauf mitgeschrieben. Daraufhin werden die Testflüge sowohl in der Simulation als auch mit der realen Flugplattform wiederholt. Exemplarisch ist in Abbildung 69

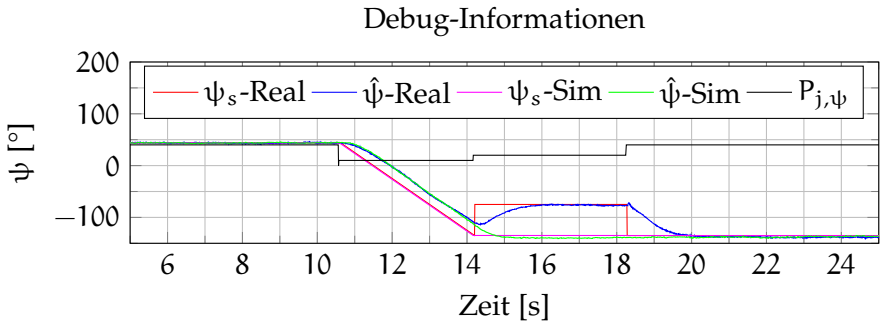


Abbildung 69: *Outdoor*-Flugtest: Vergleich der bestimmenden Zustandsgrößen der realen gegenüber der simulierten Flugplattform bei Auftreten des *Yaw*-Winkelfehlers

der Ausschnitt eines realen Fluges, der den Fehler aufweist, gegenüber einem simulierten Flug dargestellt. In *rot* ist die Führungsgröße, also der *Soll-Yaw*-Winkel ψ_s, und in *blau* der systemintern geschätzte *Ist-Yaw*-Winkel $\hat{\psi}$ der realen Flugplattform gezeigt. Diesen überlagert abgebildet ist in *magenta* die Führungsgröße des simulierten Systems sowie in *grün* dessen systemintern geschätzter *Yaw*-Winkel. Sehr deutlich ist kurz nach Sekunde 14 der Fehler des *Yaw*-Winkels $\hat{\psi}$ des realen Systems zu sehen. Dieser erreicht nicht, wie beim simulierten System, die geplante Ausrichtung von $-135°$, sondern konvergiert gegen $-75°$. Besonders interessant hierbei ist, dass der geplante Winkel ψ_s (in *rot*) ebenfalls zur selben Zeit einen Sprung von $-135°$ auf $-75°$ macht. Das bedeutet, dass der Fehler bereits in der Berechnung der Führungsgröße im Navigationsrechner passiert. Im Gegensatz dazu konvergieren geplanter und tatsächlicher *Yaw*-Winkel der simulierten Plattform entsprechend der Vorgabe von $-135°$. In *schwarz* ist die Abarbeitungsphase $P_{j,\psi}$ des Navigationsrechners gezeigt, die angibt, in welchem Abschnitt sich die Wegpunktverarbeitung der geplanten Trajektorie befindet. Die Codierung der Phasen entspricht dabei der Aufstellung in Tabelle 3. Vergleicht man die Verarbeitungsabschnitte des Navigationsrechners mit dem Sprung in der Führungsgröße des realen Systems, wird deutlich, dass der Sprung direkt mit der Änderung von Phase $P_{j,\psi} = 10$ auf Phase $P_{j,\psi} = 20$ beginnt und bei der

PHASE	BESCHREIBUNG
$P_{j,\psi} = 0$	Trajektorienberechnung zum nächsten Wegpunkt
$P_{j,\psi} = 10$	Drehung zum vorgegebenen *Yaw*-Winkel
$P_{j,\psi} = 20$	Abschluss der Drehung und Optimierung der *Yaw*-Winkelgenauigkeit
$P_{j,\psi} = 40$	Anflug des nächsten Wegpunktes

Tabelle 3: Phasen der Wegpunktverarbeitung

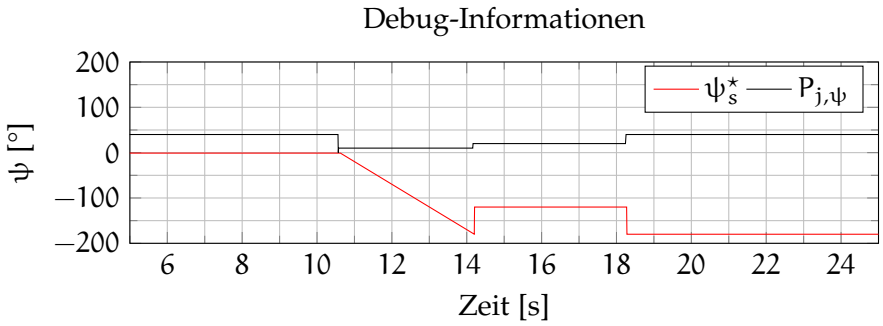


Abbildung 70: *Outdoor*-Flugtest: Auswertung des Winkelfehlers bei automatischer Wegpunktnavigation

Änderung von Phase $P_{j,\psi} = 20$ auf Phase $P_{j,\psi} = 40$ zum geforderten Wert zurückkehrt.

Um diesen Fehler weiter einzugrenzen, wird der Quellcode des Navigationsrechners eingehender analysiert. Dabei ist jedoch zu beachten, dass der Navigationsrechner den *Yaw*-Winkel bezüglich der Kamerarichtung bestimmt. Dieser ist um $-45°$ um die z-Achse des körperfesten Koordinatensystems gedreht (siehe Abbildung 19 auf Seite 83). In Abbildung 70 wird daher die systemintern berechnete Führungsgröße $\psi_s^\star = \psi_s - 45°$ gemäß dieser Rotation dargestellt. Grundsätzlich verarbeitet der Navigationsalgorithmus den geplanten *Yaw*-Winkel als trigonometrische Funktion auf dem Einheitskreis. Deshalb wird in der Trajektorienplanung der gewünschte *Yaw*-Winkel als Vektor $\vec{v}_{\psi_s^\star}$ in kartesische Koordinaten umgerechnet. In diesem Fall ist der geplante *Yaw*-Winkel $\psi_s^\star = (-135° - 45°) = -180°$. Die Navigationsplanung (Phase $P_{j,\psi} = 0$) rechnet diesen entsprechend des Einheitskreises in den Vektor $\vec{v}_{\psi_s^\star} = (-1.0,\, 0.0)^T$ um. Daraufhin rotiert der Navigationsrechner während Phase $P_{j,\psi} = 10$ den Vektor des bisherigen Winkels $\psi_s^\star = 0°$ mit $\vec{v}_{\psi_s^\star} = (1.0,\, 0.0)^T$ schrittweise, solange $\vec{v}_{\psi_s^\star} < (-1.0,\, 0.0)^T$ gilt. Die gewünschte Drehrate der Flugplattform entspricht der Rotationsgeschwindigkeit des Vektors. In Phase $P_{j,\psi} = 20$ hat der Navigationsrechner die Rotation abgeschlossen und wartet, bis die Flugplattform den vorgegebenen Winkel erreicht hat. Der Vektor beträgt erwartungsgemäß während dieser Zeit $\vec{v}_{\psi_s^\star} = (-1.0,\, 0.0)^T$. Zur Regelung des *Yaw*-Winkels wird der Vektor $\vec{v}_{\psi_s^\star}$ wieder in Polarkoordinaten mit dem Winkel $\psi_s^\star$ als Führungsgröße umgerechnet. Da es sich im Navigationscomputer um eine allgemeine Transformationsberechnung von kartesischen Koordinaten zu Polarkoordinaten handelt, wird der Eingangsvektor $\vec{v}_{\psi_s^\star}$ zuerst normiert zu:

$$\vec{n}_{\psi_s^\star} = \frac{\vec{v}_{\psi_s^\star}}{|\vec{v}_{\psi_s^\star}|}. \tag{8.42}$$

Der Winkel folgt entsprechend zu:

$$\psi_s^\star = \arccos(\frac{\vec{n}_{\psi_s^\star,x}}{1}). \tag{8.43}$$

Die Hypotenuse ist 1, da der Vektor zuvor normiert worden ist. Zusätzlich muss auf Basis des durch die Drehrichtung bekannten Quadranten das Vorzeichen des Winkels bestimmt werden.

Bei genaueren Untersuchungen zeigte sich, dass der Berechnungsfehler im Navigationscomputer reproduzierbar unter der Bedingung auftritt, wenn der Vektorbetrag $|\vec{v}_{\psi_s^\star}| = 0.99999990$ beträgt. In verschiedenen Validierungstests zeigte sich weiter, dass der bisher angenommene geplante *Yaw*-Winkel nicht exakt $\psi_s^\star = -180°$ beträgt, sondern geringfügig davon abweichen kann. In dem hier untersuchten Fall betrug der geplante *Yaw*-Winkel $\psi_s^\star = -179.97°$. Da die Berechnungen im Navigationscomputer mit einfacher Genauigkeit (*Single Precision*) durchgeführt werden, führt dieser geplante *Yaw*-Winkel zu einem Vektor von $\vec{v}_{\psi_s^\star} = (-0.99999980, -0.00052337)^T$. Der vom realen Autopiloten berechnete Vektorbetrag folgt damit genau zu $|\vec{v}_{\psi_s^\star}| = 0.99999990$. Wird weiter, entsprechend Gleichung 8.42, der Vektor normiert, folgt jedoch nicht das aufgrund der Simulation erwartete Ergebnis mit

$$\vec{n}_{\psi_s^\star} = \begin{pmatrix} -0.99999980 \\ -0.00052337 \end{pmatrix}, \tag{8.44}$$

sondern fälschlicherweise

$$\vec{n}_{\psi_s^\star} = \begin{pmatrix} -0.50000000 \\ -0.00052337 \end{pmatrix}. \tag{8.45}$$

Dieser fehlerhaft normierte Vektor führt basierend auf Gleichung 8.43 mit dem Vorzeichen aus der Quadrantenbetrachtung zu einem fehlerhaften Winkel von $\psi_s^\star = -120°$. Dies entspricht exakt dem Sprung in Abbildung 70 auf Seite 143. Unter Berücksichtigung der Rotation bezüglich der Kamerarichtung und dem körperfesten Koordinatensystem folgt für $\psi_s = \psi_s^\star + 45°$ und ergibt somit $\psi_s = (-120° + 45°) = -75°$. In Phase $P_{j,\psi} = 40$ wird der geplante *Yaw*-Winkel neu und an dieser Stelle richtig berechnet. Analog folgt für die Berechnung des Winkelfehlers im Zeitpunkt $t = 125\,s$ mit $\psi_s^\star \approx (45° - 45°)$ mit dem Vektor $\vec{v}_{\psi_s^\star} = (0.99999980, 0.00052337)^T$ ebenfalls ein fehlerhaft normierter Vektor mit $\vec{n}_{\psi_s^\star} = (0.50000000, 0.00052337)^T$. Dieser entspricht einem *Yaw*-Winkel von $\psi_s^\star = 60°$ und somit $\psi_s = (60° + 45°) = 105°$. Diese Winkelfehler stimmen mit den Abweichungen, welche in Abbildung 68 auf Seite 140 festgestellt werden konnten, exakt überein.

Zur Validierung des Fehlers ist ein Testprogramm auf dem Mikrocontroller des Autopiloten implementiert worden, welches eine vorgegebene Abfolge von Divisionen testet. Die Ausgabe in Abbildung 71

Ausgabe des Mikrocontrollertestprogramms

```
%-----------------------------------------------%
%         Compiler-Test bei Division            %
%-----------------------------------------------%
% Dividend       Divisor              Ergebnis %
%    (a)           (b)                  (a/b)  %
%-----------------------------------------------%
-0.99999900    0.99999990          -0.99999940
-0.99999910    0.99999990          -0.99999950
-0.99999920    0.99999990          -0.99999960
-0.99999930    0.99999990          -0.99999970
-0.99999940    0.99999990          -0.99999980
-0.99999950    0.99999990          -0.99999990
-0.99999960    0.99999990          -0.50000000
-0.99999970    0.99999990          -0.50000000
-0.99999980    0.99999990          -0.50000000
-0.99999990    0.99999990          -1.00000000
-1.00000000    0.99999990          -1.00000000
```

Abbildung 71: *Outdoor*-Flugtest: Veranschaulichung des compilerabhängigen Berechnungsfehlers bei unterschiedlichen Divisionen

zeigt eindeutig den Fehler. Somit kann dieses Problem auf einen Fehler innerhalb des genutzten *Compilers* zurückgeführt werden[3]. Zur kurzfristigen Beseitigung dieses Effekts ist eine Abfrage implementiert worden, welche den Vektorbetrag auf Gleichheit mit 0.99999990 prüft und gesetzt den Fall, diesen zu 1.0 setzt. Der durch diese Aufrundung entstehende Fehler ist vernachlässigbar klein. Zur Validierung sind weitere 12 Testflüge absolviert worden, in denen dieser Fehler nicht mehr aufgetreten ist. Damit ist auch die Ursache geklärt, warum dieses Verhalten nicht von der simulierten Plattform gezeigt worden ist, denn diese ist auf einer x86 Prozessor-Architektur kompiliert worden und dieser spezifische Compilerfehler tritt nachweislich dort nicht auf.

Bewertung der Ergebnisse

Beim Vergleich der *High-Level*-Algorithmen der Flugplattform, wie im dargestellten Fall der automatischen Wegpunktnavigation, zeigt sich ein absolut identisches Flugverhalten. Eingangsgrößen wie GNSS-Position, Magnetometerdaten oder die barometrische Höhenbestimmung sind bei der realen und der simulierten Flugplattform äquiva-

3 Es konnte in weiteren Untersuchungen nachgewiesen werden, dass dieser Fehler in der genutzten Entwicklungsumgebung *Keil uVision3*, in der Version V3.12a, reproduzierbar ist. Nach der Aktualisierung auf Version V3.20a erfolgt eine fehlerfreie Berechnung.

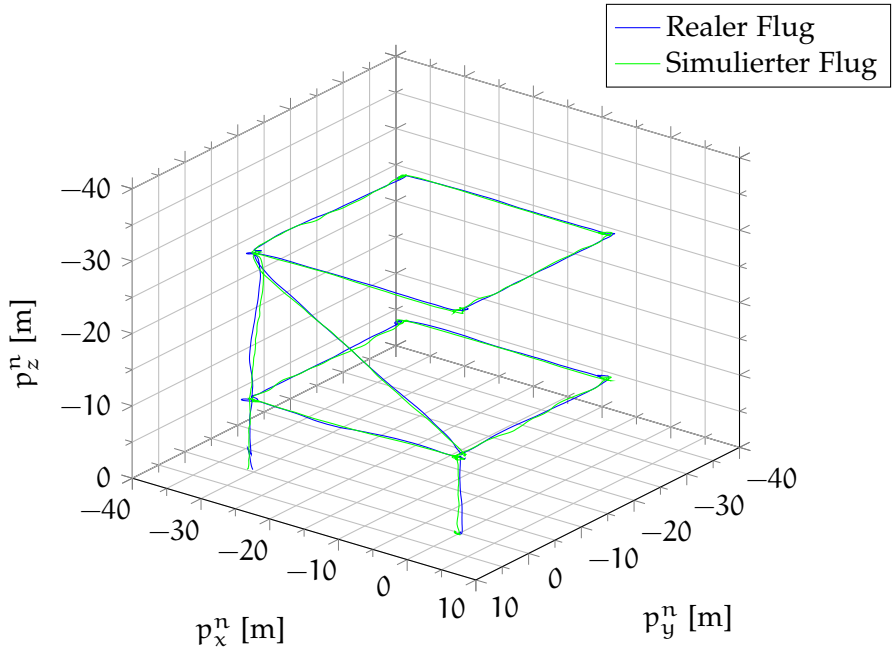


Abbildung 72: *Outdoor*-Flugtest: Abbildung der 3D-Trajektorie im Vergleich von der realen gegenüber der simulierten Flugplattform im lokalen Navigationskoordinatensystem

lent. Analog zum *Indoor*-Szenario zeigen die Systeme auch in diesem Fall eine identische Flugdynamik mit vergleichbar geringen Fehlertermen auf. Dies spiegeln die zuvor analysierten Darstellungen der Abbildungen 63 bis 68 auf den Seiten 135 bis 140 ebenso wieder wie Abbildung 72, in der die systemintern berechnete 3D-Position im Navigationskoordinatensystem von realer zu simulierter Plattform gegenübergestellt ist. Dabei ist in dieser Positionsdarstellung zu sehen, dass das simulierte System sehr präzise das Flugverhalten des realen MAVs nachempfindet.

Zudem konnte anhand der Magnetometerdaten gezeigt werden, dass die erarbeiteten Modelle erwiesenermaßen so präzise sind, dass nicht nur das Systemverhalten äquivalent in der Simulation abgebildet wird, sondern zusätzlich sogar kleinste Abweichungen von den Erwartungswerten detektiert werden können. In diesem Fall konnte nicht nur das stromabhängige elektromagnetische Störfeld als Biasfehler in der Magnetfeldmessung nachgewiesen werden, auch die quantitative Größe konnte in der Simulation bestimmt werden.

Des Weiteren konnte durch die Simulation ein Compilerfehler des Mikrocontrollers belegt werden. Hierbei war die Möglichkeit zur Simulation aller Daten, Parameter und Effekte besonders hilfreich. Ohne das in dieser Arbeit geschaffene Entwicklungswerkzeug zur präzisen und realistischen Simulation solcher Multirotorplattformen wäre die erfolgreiche Analyse bei derartigen Fehlern eine deutlich schwierigere Aufgabe, da keinerlei *Ground-Truth*-Daten zur Verfügung stün-

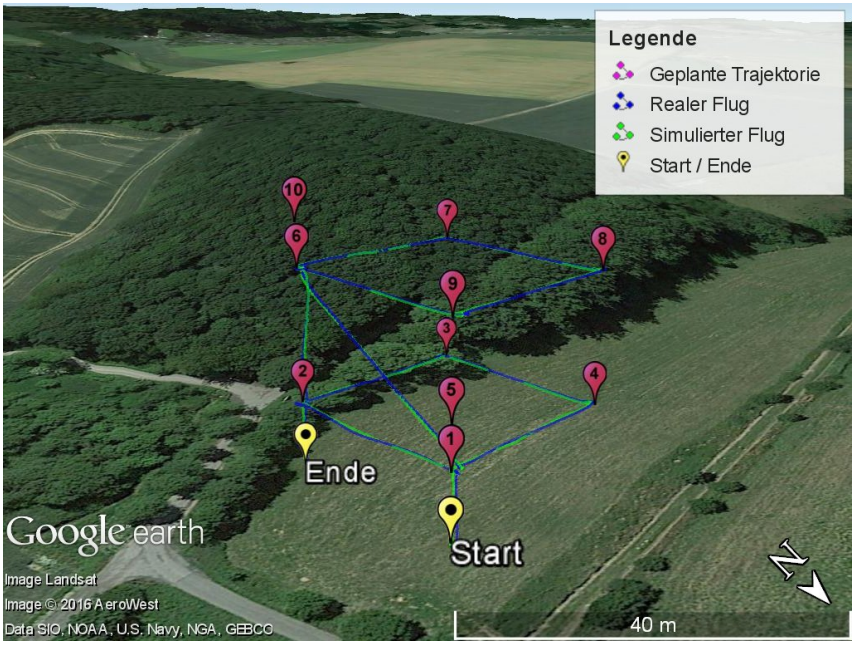


Abbildung 73: *Outdoor*-Flugtest: Abbildung der 3D-Trajektorie im Vergleich von der realen gegenüber der simulierten Flugplattform im WGS-84

den und somit ein *Soll-Ist*-Vergleich fast unmöglich wäre. Abschließend sind in Abbildung 73 die Daten der realen und der simulierten GNSS-Messung gekoppelt mit der relativen Höhenbestimmung dargestellt. Damit zeigt nicht zuletzt dieser Vergleich, dass die hier vorgestellte Annahme eines vereinfachten GPS-Fehlermodells zielführend ist.

8.2.3 *Ergebnisse und Diskussion*

In den letzten Abschnitten wurde die Modellierung der Flugplattform, ihre Sensoren, die Antriebe sowie flugdynamischen Kenngrößen hergeleitet und die Umsetzung in der vorgestellten Simulationsumgebung gezeigt. Zudem wurden zwei unterschiedliche Flugszenarien zur Analyse der Simulationsergebnisse dargestellt. Hierbei konnte in unterschiedlichen Flugsituationen nachgewiesen werden, dass die simulierte Flugplattform das Flugverhalten und die Messgrößen sehr präzise nachbildet. Minimalste Abweichungen, beispielsweise bei den Motorsteuergrößen in Abbildung 59 auf Seite 131 zu sehen, lassen sich auf unregelmäßige äußere Einflüsse auf die reale Plattform zurückführen. So zeigen die Antriebe bei unterschiedlichen Umgebungstemperaturen bereits geringfügige Abweichungen in ihren Kenndaten. Außerdem ist davon auszugehen, dass sich die Rotoreigenschaften ebenfalls durch eine unterschiedliche Luftanströmung in geringem Maße verändern.

Es ist festzuhalten, dass die in dieser Arbeit vorgestellte Simulation inklusive der erarbeiteten Modelle so präzise ist, dass sogar geringste Abweichungen von den Erwartungswerten detektiert werden können, wie es am Beispiel der Magnetometerdaten gezeigt worden ist. Dies zeigt vollständig neue Möglichkeiten in der Entwicklung von Multirotor-MAVs auf, beispielsweise beim Entwurf neuartiger Online-Kalibrierverfahren von Sensoren. Im Gegensatz zu bisherigen Arbeiten liegt der außerordentliche Vorteil dieser Simulationsumgebung in der Genauigkeit der Systemmodellierung. So ist es jetzt möglich, Algorithmen, zum Beispiel zur Sensorfusion, Systemregelung oder Navigation, in der virtuellen Welt zu entwickeln und diese mit realistischen Sensordaten sowie Systemeigenschaften der Plattform risikofrei zu evaluieren. Dies ist sowohl in *Indoor*- als auch in *Outdoor*-Szenarien möglich. Dabei stehen dem Entwickler zu jeder Zeit synchronisierte *Ground-Truth*-Daten zur Verfügung. Ebenfalls können Probleme oder Fehler direkt am virtuellen System diagnostiziert und eliminiert werden, ohne das Risiko der Zerstörung einer realen Flugplattform einzugehen. Dies konnte bei der Bestimmung des Compilerfehlers eindeutig gezeigt werden. Zusätzlich können Algorithmen mit unterschiedlichsten Startbedingungen und verschiedenen Sensorcharakteristika unter reproduzierbaren Bedingungen getestet und analysiert werden. Mit diesem innovativen Entwicklungswerkzeug wird im weiteren Verlauf der Arbeit gezeigt, dass zudem vollständige Systemneuentwicklungen möglich sind. Dabei werden alle Erweiterungen und Anpassungen des Autopiloten sicher und effektiv im virtuellen System erfolgen.

8.3 ENTWICKLUNG NEUER SYSTEME

Nachdem im vorherigen Abschnitt die Systemmodellierung analysiert und ein äquivalentes Flugverhalten zwischen realem und simuliertem MAV nachgewiesen worden ist, soll im Folgenden eine vollständig simulationsbasierte Entwicklung von neuen Multirotorsystemen gezeigt werden. So sind mit der zuvor entwickelten 3D-Echtzeit-Simulation insgesamt vier neue Flugplattformen in Kooperation mit der Firma AirRobot entstanden. Hierzu sind im Rahmen dieser Arbeit sämtliche Neu- und Weiterentwicklungen sowie Anpassungen und Optimierungen inneralb der Software des Autopiloten erarbeitet und analysiert worden. Von diesen vier Systemen werden zwei Systeme exemplarisch vorgestellt, die sich zur bisherigen Flugplattform zum einen in den mechanischen Dimensionen und dem Abfluggewicht unterscheiden und zum anderen in ihrer Anzahl an Antrieben und damit in der Konfiguration verschiedenartig sind. Dabei kommt bei den Systemen der zuvor modellierte Autopilot mit baugleichen Sensoren zum Einsatz.

8.3.1 *Der Hexakopter AR120*

Das erste System, bei dem sämtliche Algorithmen zur Sensorfilterung, -fusion und Systemregelung vollständig in dieser Simulation weiterentwickelt, analysiert und optimiert worden sind, ist der *AR120*. Dieses System ist ein VTOL-MAV mit sechs Antrieben (Hexakopter), das einen maximalen Durchmesser von Rotorblattspitze zu Rotorblattspitze von 120 cm hat. Grundsätzlich war dieses System als interne Studie geplant, um einerseits das Konzept und die Möglichkeiten von Hexakoptersystemen zu evaluieren und andererseits deutlich schwerere Nutzlasten mitführen zu können (600 g gegenüber den bisherigen 250 g des *AR100B*). Im Vergleich zur Ausgangsplattform *AR100B* sollten jedoch die Flugeigenschaften äquivalent sein. Aufgrund der sehr schnellen Entwicklungszeit von weniger als zwei Wochen und der überdurchschnittlichen Flugperformance ist dieses System daraufhin über die Studie hinaus für verschiedenste Anwendungen genutzt worden.

Durch die identische Hardware bei den Antrieben, den Sensoren und dem Autopiloten konnten die in Abschnitt 8.1 gezeigten Modelle vollständig weiter genutzt werden. Änderungen mussten jedoch in der Simulation im *Physik-Plug-In* gemacht werden, da nun sechs anstatt wie zuvor vier Antriebe modelliert und verarbeitet werden mussten. Ebenso mussten die Parameter für die Bewegungsmodellierung, das heißt für das geplante Abfluggewicht, das Massenträgheitsmoment und der Massenschwerpunkt des MAVs bestimmt werden. Da zum Projektbeginn kein detailliertes CAD-Modell zur Verfügung stand, wurden das Gewicht und der Massenschwerpunkt anhand der Einzelkomponenten abgeschätzt. Das gesamte Massenträgheitsmoment der Flugplattform ist durch die rechnerische Bestimmung der Trägheitsmomente von approximierten starren Körpern der einzelnen Komponenten bestimmt worden (siehe [115]). Der Strömungswiderstand wurde anhand der Kenndaten des *AR100B* abgeleitet.

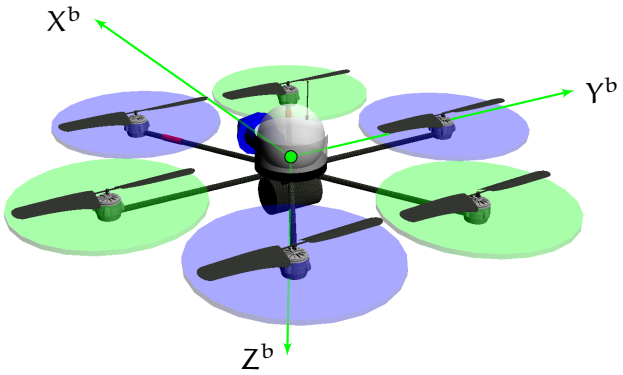


Abbildung 74: Modell der Flugplattform *AR120*

(a) Simulierte Flugplattform (b) Reale Flugplattform

Abbildung 75: Flugplattform *AR120* – simuliert zu real

Ein Schwerpunkt bei dieser Systementwicklung lag in der Adaption des körperfesten Koordinatensystems. Bei der Ausgangsplattform war dieses entsprechend einer +-Konfiguration definiert und musste daraufhin in die eigentlich gewünschte *X*-Konfiguration transformiert werden. Demgemäß musste bei allen Eingaben eine Drehung um die körperfeste Hochachse um −45° beachtet werden. Beim *AR120* wurde dies in der Planung berücksichtigt und das Koordinatensystem, wie Abbildung 74 auf Seite 149 zeigt, implementiert. Da hiervon sämtliche automatisierten Funktionen, beispielsweise die Wegpunktnavigation oder das automatische Heimkehren, betroffen sind, war es besonders hilfreich, innerhalb der virtuellen Realität die Richtigkeit und Vollständigkeit aller Transformationen und die darauf aufbauende Trajektorienplanung zu prüfen und die Ergebnisse mit den Erwartungswerten entsprechend zu validieren.

Eine weitere Herausforderung stellte die Anpassung und Optimierung der Reglerparameter für die Lage- und Positionsregelung dar. Hierzu konnten in der Simulation alle relevanten Zustandsgrößen, beispielsweise die Flughöhe, die Lagewinkel oder die Drehraten, analysiert und zur Bestimmung der Reglergüte die vorgegebene Führungsgröße (*Soll*-Wert) mit dem systemintern gemessenen bzw. geschätzten *Ist*-Wert verglichen werden. Darüber hinaus konnte auf Basis der *Ground-Truth*-Daten direkt die Qualität der Sensorfilterung und -fusion durch den Einfluss der neuen Systemdynamik bewertet und optimiert werden. In Abbildung 75 sind der simulierte und der daraufhin aufgebaute Prototyp dargestellt. Erst nach dem virtuellen Entwicklungsprozess ist der reale Prototyp gebaut worden. Die Software des Autopiloten ist anschließend direkt aus dem SiL-Modul der Simulation in den realen Prototypen ohne weitere Änderungen übertragen worden. Dabei zeigte das Fluggerät vom ersten Start an ein zur Simulation identisches und stabiles Flugbild. Durch diese neue Entwicklungsstrategie bei Multirotorsystemen sind bisherige softwarebedingte Fehler und daraus resultierende Abstürze der realen Systeme vollständig verhindert worden. Dies trägt gerade in der Entwicklungsphase zu einem erheblichen Sicherheits- und Zeitgewinn bei.

8.3.2 *Der Hexakopter AR200*

Wie eingangs beschrieben, war der *AR120* ursprünglich als Systemstudie geplant, um das Konzept und die Möglichkeiten von Hexakoptersystemen zu evaluieren. Das eigentliche Ziel stellte die Entwicklung einer weitaus größeren Flugplattform dar. Die Vorgabe bestand in der Entwicklung eines Multirotorsystems mit einer Nutzlastkapazität von 3000 g mit der Bedingung, dass trotzdem eine Flugzeit von $\geqslant 30$ Minuten erzielt werden sollte. Aufgrund dessen ist der *AR120* um knapp das Doppelte seiner mechanischen Abmessungen, von 120 cm auf 220 cm, gewachsen. Ebenfalls sollten für dieses System neue Antriebe genutzt werden, die demzufolge für die Simulation modelliert worden sind. Im Vergleich zu den bisherigen, welche einen maximalen Schub von ≈ 8 N erzeugen, erreicht jeder einzelne der neuen Antriebe des *AR200* bei einer Leistungsaufnahme von 540 W mehr als den fünffachen Schub mit maximal ≈ 42 N. Auf Basis eines vorhandenen CAD-Modells konnten die Modellparameter für das Massenträgheitsmoment sowie für den Massenschwerpunkt der Flugplattform bestimmt werden. Das Gewicht der Flugplattform beträgt ohne Nutzlast 9.10 kg. Mit der geplanten maximalen Nutzlast beläuft sich das maximale Abfluggewicht auf 12.10 kg.

Zu den besonderen Herausforderungen der Softwareentwicklung für den Autopiloten dieser Plattform gehörten die zuvor genannten außergewöhnlichen Systemparameter[4]. Im Vergleich zu den bisher entwickelten Plattformen waren das um eine Zehnerpotenz höhere Abfluggewicht und die grundsätzlich unterschiedliche Systemdynamik beim Filter- sowie Reglerentwurf entscheidend. Beispielhaft soll an dieser Stelle anhand der Flughöhenregelung ein derartiger Reglerentwurf diskutiert werden. So regelten die bisherigen MAVs die Höhe sehr direkt. Wenn der Pilot beispielsweise zuerst ein schnelles Sinken der Flugplattform kommandiert und dieses daraufhin abrupt beendet, „rastet“ die Flugplattform förmlich auf dieser Höhe ein. Dies führt regelungstechnisch dazu, dass die Antriebe von einer sehr geringen Drehzahl im Sinkflug sprunghaft auf die maximale Drehzahl gesteuert werden, um das Sinken unmittelbar zu beenden. Daraus resultiert eine sehr hohe Belastung der Antriebe. Durch deren maximale Leistungsaufnahme kommt es zu einer sehr hohen elektronischen Last der Energieversorgung. Bei dieser neuen Flugplattform wäre diese Spitzenlast jedoch so hoch, dass der mitgeführte Akku deutlich außerhalb seiner Spezifikation betrieben werden würde. Durch die hohen Spitzenströme bei diesem Flugmanöver wäre ein Einbrechen der Betriebsspannung der Antriebe und ein damit direkt verbundener Auftriebsverlust die Folge. Es käme somit zu einer instabilen Lagere-

4 Zum Zeitpunkt der Entwicklung dieser Flugplattform im Jahr 2012 war nach damaligem aktuellem Kenntnisstand kein System mit vergleichbaren Parametern bekannt.

(a) Simulierte Flugplattform (b) Reale Flugplattform

Abbildung 76: Flugplattform *AR200* – simuliert zu real

gelung. Aus diesem Grund ist in der Simulation für dieses Flugmanöver eine dynamische Regleranpassung entwickelt worden. Hierzu ist auf Basis des Systemmodells ein Regelungsalgorithmus entstanden, der in Abhängigkeit von der Sinkgeschwindigkeit eine systemspezifische Verzögerungsstrecke bestimmt, in welcher die Flugplattform selbstständig die Eigengeschwindigkeit reduziert und daraufhin in eine stabile Höhen- und Lagehaltung übergeht. Aufgrund des simulationsbasierten Ansatzes konnte ein optimales und durchgehend stabiles Reglerverhalten entwickelt werden; außerdem konnte der maximale Verzögerungsweg auf maximal 0.5 m begrenzt werden.

Neben dem eigentlichen Softwareentwurf stellt jedoch bei dieser Größe der Flugplattform die Evaluation selbst eine nicht unerhebliche Aufgabe dar. Fehler, die zum Divergieren des integrierten Filters und somit zur Nichtbeobachtbarkeit des Systemzustandes führen, oder auch fehlerhaft eingestellte Reglerparameter können jeweils zum Kontrollverlust über die Flugplattform und damit zum Absturz führen. Das hätte bei dieser Plattform nicht nur erhebliche Kosten von mehreren tausend Euro und einen entsprechenden Zeitverlust durch den Neuaufbau für den realen Prototypen zur Folgen, sondern würde ferner ein erhebliches Risiko für die Sicherheit aller Beteiligten darstellen. Die in dieser Arbeit entwickelte modellbasierte 3D-Echtzeit-Simulation bietet hierfür die entsprechende Lösung. So sind mit den für diese Flugplattform aktualisierten Modellen sämtliche Neu- und Weiterentwicklungen für die Systemzustandsschätzung und die Systemregelung innerhalb der virtuellen Welt vorgenommen worden. Erst nach der dortigen Analyse und Optimierung aller Systemparameter und Systemzustandsgrößen wurde, wie zuvor beim *AR120*, der erste reale Prototyp erprobt. Die Software des Autopiloten innerhalb des SiL-Moduls ist dazu ohne weitere Anpassungen bzw. Veränderungen in das reale *Automatic Flight Control System* (AFCS) eingespielt worden, so dass ein erster Testflug direkt folgen konnte. Dieser und weitere Testflüge zeigten im Vergleich zur vorherigen Simulation eine äquivalente Flugperformance. Systemausfälle oder Abstürze aufgrund von Fehlern in der Software des Autopiloten konnten durch diese Entwicklungsmethode verhindert werden. In Abbildung 76 sind der simulierte und der reale Prototyp gezeigt.

8.3.3 *Weitere Anwendungen der Systemsimulation*

Neben den beiden zuvor exemplarisch vorgestellten Systemen ist die Software des Autopiloten für zwei weitere Systemkonzepte innerhalb der Simulation entwickelt und ebenfalls bis zur Marktreife gebracht worden. Diese beiden Systeme basieren auf dem Konzept eines Quadrokopters in *X*-Konfiguration und können als Weiterentwicklung der gezeigten Ausgangsplattform *AR100B* gesehen werden. An dieser Stelle wird jedoch darauf verzichtet, diese Systeme bzw. deren Entwicklung explizit vorzustellen.

Eine weitere Entwicklung, bei der die in dieser Arbeit gezeigte Simulation den Ausgangspunkt bildet, ist ein realistischer Ausbildungssimulator für MAV-Piloten. Das Konzept von Ausbildungssimulatoren ist nicht neu. So gehören diese in der bemannten Luftfahrt zum Stand der Technik. Sie werden aufgrund ihrer realistischen Nachbildung der Flugzeugeigenschaften zur Pilotenausbildung bei Verkehrs- und Überschallflugzeugen (siehe [106]) oder auch bei Raumfähren (siehe [5]) genutzt. Für unbemannte kleine Flugsysteme, wie den hier betrachteten MAVs, sind derartige Simulatoren jedoch nur vereinzelt vorhanden und kommen hauptsächlich aus dem Modellbau bzw. Hobbybereich. Meistens sind für die verschiedenen Fluggeräte jedoch nur rudimentäre Modelle der physikalischen Eigenschaften hinterlegt (siehe hierzu Abschnitt 6.1). Die Integration der realen Autopilotensoftware und deren Funktionsumfang wird bei diesen Simulatoren in der Regel jedoch vollständig vernachlässigt. Um diese Lücke zu schließen, ist die in dieser Arbeit entwickelte Simulation auf die Anforderungen des Ausbildungssimulators angepasst worden. Dazu wurde der SiL-Kern der Ausgangsplattform *AR100B* in einer ersten *Hardware in the Loop* (HiL)-Implementierung auf der realen Hardware des Autopiloten, in Kombination mit dem originalen Handbedienteil zur Flugplattformsteuerung, integriert. Das *Backend*, welches vorwiegend der Darstellung der Flugplattform und deren virtueller Umgebung dient, ist in Zusammenarbeit mit einem externen Partner (siehe [51]) auf ein kommerzielles *Framework* (siehe [17]) portiert worden, da dieses auf eine ausbildungsorientierte Gestaltung von unterschiedlichen Szenarien ausgelegt ist. Elementarer Kern des Ausbildungssimulators bleibt jedoch die in dieser Arbeit vorgestellte modellbasierte Simulationsstruktur und deren spezifische Bestandteile. Für zukünftige Erweiterungen sind die Schnittstellen zwischen der hier entwickelten Flugplattformsimulation und einem grafischen *Backend* so ausgelegt, dass andere Computerspiel-Engines bzw. Simulationstools ebenfalls genutzt werden könnten.

Neben der bisherigen Fokussierung dieser Simulation auf die Weiterentwicklung eines Autopiloten für bestehende oder für neue Flugplattformen ist auch die Umsetzbarkeit von bildbasierten Sensorsystemen auf der virtuellen Flugplattform untersucht worden. Der Schwer-

punkt lag hierbei auf der Modellierung eines Kamerasystems und der Implementierung eines musterbasierten Verfahrens zur Positions- und Lagebestimmung eines MAVs. Ziel war die Realisierung eines geschlossenen Regelkreises zum automatischen Starten, einer eigenständigen Positionshaltung und dem selbstständigen Landen. Die Abgrenzung zu bisherigen Arbeiten auf diesem Gebiet liegt im simulationsbasierten Ansatz der gesamten Entwicklung. Dies bedeutet, dass alle Algorithmen zur Bildaufnahme, Sensorverarbeitung, -fusion und Regelung in der Simulation entworfen und erst nach erfolgreicher Umsetzung und Evaluierung auf ein reales System portiert wurden. Dabei ist anhand des musterbasierten Ansatzes durch eine 2D-3D-Punktkorrespondenzanalyse die Position und Orientierung der Kamera bestimmt worden. Um diese Positionsbestimmung robust gegen Störeinflüsse zu halten, sind die optisch gewonnenen Daten mit den IMU-Daten in einem *Extended Kalman-Filter* (EKF) fusioniert worden. Die Herleitung, Umsetzung und daraus resultierende Ergebnisse dieser Arbeit sind in dem begutachteten Beitrag [102] und ebenfalls in [101] zu finden.

8.4 ZUSAMMENFASSUNG UND DISKUSSION

Die in diesem Kapitel vorgestellte Simulationsumgebung zeigt ein innovatives und maßgebendes Werkzeug zur Entwicklung und Analyse von Multirotor-MAVs. Dabei liegt die herausragende Eigenschaft in der modellbasierten, realistischen Echtzeit-Simulation des gesamten Flugsystems. So können durch die originalgetreue Abbildung der Sensoren, der Antriebe und der flugdynamischen Kenngrößen nicht nur vorhandene Algorithmen unter realistischen Bedingungen analysiert werden, sondern ebenfalls unterschiedlichste äußere Einflussfaktoren, wie z. B. Windlasten, in die Gesamtbetrachtung mit einbezogen werden. Im Gegensatz zu mathematischen Analysetools, beispielsweise *MATLAB & Simulink*, ist es in dieser neuartigen Simulation möglich, den vollständigen Quellcode des Autopiloten *Software in the Loop* (SiL) einzubinden und neben den prozessierten Daten der Sensoren, Filter oder Regler ein realistisches, visuelles Feedback des Fluggerätes in Echtzeit zu erhalten. Dabei konnte gezeigt werden, dass die Simulationsergebnisse nahezu identisch mit den Daten der realen Flugplattform sind. Darauf aufbauend können sogar komplexe Fehler, wie der gezeigte Kompilierungsfehler, auf der realen Flugplattform nachgewiesen werden, da durch die Simulation ein idealer Vergleichswert für alle internen Variablen und externen Parameter errechnet werden kann. Nur durch diese neue Möglichkeit der Softwareanalyse können nun derartig komplexe Fehler schnell, sicher und effizient detektiert und eliminiert werden. Darüber hinaus können durch die vollständige Parametrierbarkeit aller Sensormodelle die Algorithmen zur Datenfilterung oder Datenfusion entwickelt

und untersucht werden. Schließlich bietet die Umsetzung von zeitsynchronen *Ground-Truth*-Daten ein optimales Werkzeug für diese Untersuchungen. Spezifische Abhängigkeiten und weitere Einflussfaktoren können erkannt und die Genauigkeit der Algorithmen optimiert werden. Dies spielt vor allem in Szenarien eine wichtige Rolle, in denen gängige *Motion-Capture*-Systeme nicht eingesetzt werden können, beispielsweise bei der gezeigten Wegpunktnavigation im großflächigen Außenbereich.

Die derzeitige Simulation bietet Potenzial für zukünftige Arbeiten. So ist die weiterführende Untersuchung von Strömungsmodellen ein interessanter Bereich. Hier könnten Rotormodellierungsverfahren, wie beispielsweise die *Blade Element Momentum Theory* (BEMT) (siehe [94]), Erweiterungsmöglichkeiten bieten. Die Überlegung der Implementierung einer detaillierten Strömungssimulation liegt ebenfalls nahe. So könnten Effekte, wie die „Saugeffekte“ in Ecken oder der diskutierte Bodeneffekt, präziser nachempfunden werden. In diesem Kontext stellt die Untersuchung detaillierterer Windmodelle, zum Beispiel das *Dryden-* oder das *Von Kármán-Wind Turbulence Model* (siehe [37] und [158]), eine weitere zukünftige Herausforderung dar. Damit einhergehend wäre die Evaluierung anderer Simulations-Engines sowie die Untersuchung von neuesten Physik-Engines, beispielsweise die NVIDIA *PhysX* (siehe [128]), die eine umfangreiche Strömungssimulation unterstützen, eine Aufgabenstellung. Nicht zuletzt bieten die erarbeiteten Sensormodelle ein eigenes weiteres Forschungsfeld. Hierbei wäre ein interessanter Aspekt die Modellierung sowohl von systeminternen als auch von externen Einflussfaktoren bei GNSS-Empfängern.

Eines der aktuellsten Themengebiete umfasst die autonome Navigation und Exploration sowie deren Integration auf VTOL-MAVs. Ein Interessenschwerpunkt liegt dabei auf bildbasierten Methoden. Für diese visuellen Verfahren gäbe es für die Umsetzung in der Simulation spezifische Vorteile. Neben der stetigen und synchronisierten Verfügbarkeit von *Ground-Truth*-Daten können unterschiedliche Parameter, beispielsweise Verzeichnungen, Brennweiten oder die Bewegungsunschärfen im virtuellen Kamerasystem, schnell und direkt variiert werden. Wie im vorherigen Abschnitt konturiert vorgestellt, ist eine grundsätzliche Umsetzung von bildbasierten Navigationsverfahren innerhalb der Simulation durchaus möglich.

Im weiteren Verlauf dieser Arbeit wird die simulationsbasierte Entwicklung eines vollständigen MAVs, das heißt von den Algorithmen des Autopiloten, der benötigten Hardware über eine Bodenstation bis hin zu einem kompletten Flugsystem verfolgt. Dies hat vielfältige Gründe: Wie schon eingangs bei der Sensormodellierung der bisher genutzten IMU nachgewiesen, haben die Drehraten- und Beschleunigungssensoren durch eine suboptimale Anbindung an den derzeitig verwendeten Autopiloten nur eine unzureichende Auflösung. Die

Berechnung des *Strapdown*-Algorithmus (siehe Abschnitt 4.1.2) würde ohne externe Stützung schon bei einem Biasfehler des Beschleunigungssensors von lediglich 1 LSB, was bei dieser Sensorik jedoch $0.12\,\mathrm{m/s^2}$ entspricht (siehe Gleichung 8.18 in Abschnitt 8.1.1), bei der Kurzzeitnavigation von nur 5 s zu einem Positionsfehler von 1.5 m führen. Nach 10 s liegt dieser schon bei 6 m. Analog dazu führt ein Bias im Drehratensensor von ebenfalls 1 LSB $= 1.00°/\mathrm{s}$ (siehe Gleichung 8.8 in Abschnitt 8.1.1) beispielsweise zu einem *Yaw*-Winkelfehler von 5.0° nach 5 s und schon nach 6 min zu einem Fehler von 360.0°. Zudem muss das nicht normalverteilte Rauschspektrum dieser IMU, besonders bei der Datenfusion durch ein Kalman-Filter, gesondert betrachtet werden. Die sehr grobe Auflösung und das nicht normalverteilte Rauschspektrum würden sich bei zukünftigen Arbeiten wiederkehrend negativ auswirken. Neben der Steigerung der Sensorgüte, sollen bei der Entwicklung des neuen MAV-*Frameworks* universelle Schnittstellen geschaffen werden, um zum einen den neuen Autopiloten auf vielfältigen Flugplattformen integrieren zu können und zum anderen das gesamte MAV-*Framework* zukünftig mit weiteren Entwicklungssystemen, beispielsweise dem *Robot Operating System* (ROS), zu koppeln. Nicht zuletzt ist die Größe der zu entwickelnden Flugplattform ein wichtiger Gesichtspunkt. Hier steht vor allem der universelle Einsatzbereich im Vordergrund. Dabei soll das MAV sowohl in kleinen Räumen (*Indoor*) als auch im Außenbereich (*Outdoor*) eingesetzt werden können. Der grundlegende Aspekt des im Anschluss entwickelten *FireFly* MAV-*Frameworks* ist die vollständige Kopplung aller Komponenten mit der 3D-Echtzeit-Simulation. Dabei werden sämtliche Algorithmen des neuen Autopiloten von Grund auf in der Simulation geplant und umgesetzt. Die virtuelle Flugplattform ist darüber hinaus direkt mit der parallel enwickelten Bodenstation gekoppelt. So können Parameter wie die Kommunikationsbandbreite oder verschiedenste Systemfunktionalitäten von Beginn an mit berücksichtigt werden. Erst nach entsprechenden Validierungstests wird der reale Prototyp entwickelt.

9

DAS FIREFLY MAV-FRAMEWORK[1]

Wie einleitend in dieser Arbeit beschrieben, erfreuen sich VTOL-MAVs besonders bei wissenschaftlichen Anwendern zunehmender Beliebtheit. Zentrale Forschungsthemen sind Verfahren zur autonomen Navigation, Exploration (siehe [33, 68]) oder auch der Entwurf neuartiger Regelungsstrategien für hochdynamische Flugmanöver (siehe [107, 116]). Den Kern aller Flugplattformen bildet dabei der Autopilot. Sollen nun neue oder erweiterte Anwendungen in das MAV integriert werden, ist ein partieller Eingriff in dieses unabdingbar. Dabei treten jedoch erhebliche Hindernisse auf. Kommerzielle UAV-Hersteller, wie z. B. Microdrones (siehe [119]), bieten zwar ein professionell entwickeltes System für unterschiedlichste Anwendungen an, jedoch ist der Autopilot vollständig proprietär. Es werden Schnittstellen bereitgestellt, um eine *High-Level*-Steuerung zu ermöglichen. Allerdings kann über diese nur innerhalb vorgegebener Grenzen in die Kontrolle des MAVs eingegriffen werden. Die zuvor in dieser Arbeit untersuchte und simulierte Flugplattform mit dem Autopiloten der Firma AirRobot gehört ebenfalls zu dieser Kategorie. Einen anderen Weg beschreitet zum Beispiel Ascending Technologies (siehe [13]). Hier ist, speziell für externe Integratoren, eine Trennung zwischen *High-Level*- und *Low-Level*-Regelung eingeführt worden. Dies ermöglicht dem Nutzer, die Kontrolle über die Flugplattform zu übernehmen, ohne direkt in den *Low-Level*-Kern eingreifen zu müssen. Dabei besteht für den Nutzer die Einschränkung, dass ausschließlich mit diesen MAVs gearbeitet werden kann. Eine Alternative bildet das MikroKopter-Plattformkonzept (siehe [72]). Wichtige Elemente der Plattform sind hier quelloffen gehalten. So ist es dem Nutzer möglich, Schaltpläne und Quellcodes des Autopiloten einzusehen und zu bearbeiten. Jedoch ergeben sich deutliche Einschränkungen im Funktionsumfang der Software. Grundlegende Methoden sind zwar eingebettet, jedoch fehlen im Gegensatz zu den hochpreisigen Plattformen signifikante Algorithmen zur Sensorfusion oder zur präzisen Regelung.

Um diese Lücke zwischen *Low-Level*-Regelung verschiedener UAVs und *High-Level*-Algorithmen der wissenschaftlichen Anwendungen zu schließen, wird in diesem Abschnitt eine neuartige Entwicklungsplattform (siehe Abbildung 77 auf Seite 158) inklusive Demonstrator-MAV vorgestellt. Ziel ist es, einen Systementwurf darzustellen, welcher auf vielfältigen VTOL-MAVs eingesetzt werden kann. Hierbei steht

1 Überarbeitete Fassung des veröffentlichten Beitrags: *The FireFly MAV-Framework – Closing the Gap in Micro Air Vehicle (MAV) Development* (siehe [99, 103]).

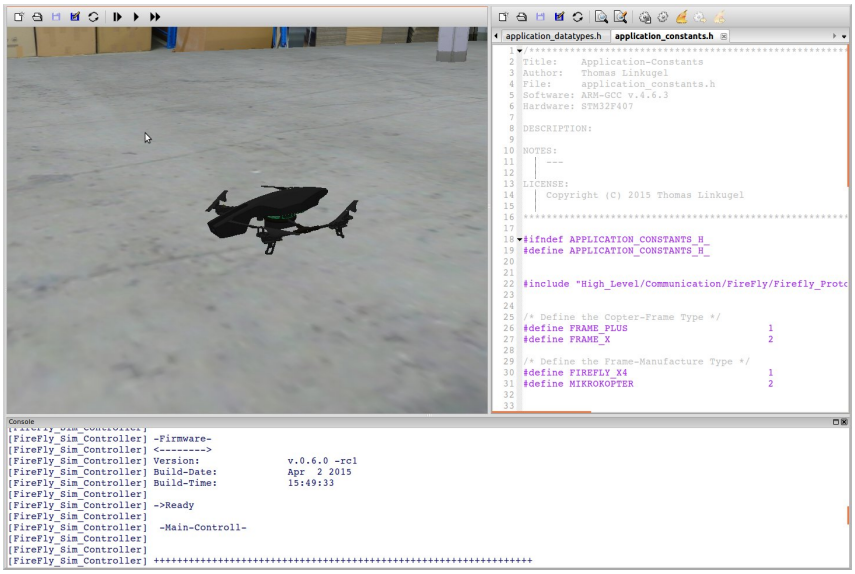


Abbildung 77: Entwicklungsplattform *FireBird-X4* in der Simulationsumgebung

jedoch nicht allein ein plattformunabhängiger Autopilot im Vordergrund, sondern ein vollständiges *Framework* zur MAV-Entwicklung. Ein besonderer Fokus wird deswegen auf das zuvor hergeleitete, echtzeitfähige Simulationssystem gelegt. Dieses ermöglicht das Design von neuen Algorithmen sowie die Optimierung von Filter- und Reglerparametern. Dabei können alle Entwicklungsschritte vollständig am virtuellen MAV erarbeitet und evaluiert werden. Durch diesen Entwurf in der virtuellen Realität wird die Entwicklungszeit maßgeblich verkürzt sowie, auf Basis der präzisen *Ground-Truth*-Daten, die Genauigkeit von Algorithmen maximiert. Nach der Entwicklungsphase in der Simulation können sämtliche Algorithmen direkt (*Software in the Loop* (SiL)) auf die reale Plattform übertragen werden. Das Risiko, eine Flugplattform während der Entwicklungsphase ungewollt zu zerstören, wird dadurch so gut wie ausgeschlossen.

9.1 VERWANDTE ARBEITEN

Durch die Verfügbarkeit von günstigen Sensoren und MAV-Komponenten wurden in den letzten zehn Jahren eine Vielzahl von Autopiloten entwickelt (siehe [96]). Eines der frühesten Projekte war das *Paparazzi*-UAV-System (siehe [26] und [69]), welches an der *Ecole Nationale de l'Aviation Civile* (ENAC) seinen Ursprung hat und in unterschiedlichen professionellen Anwendungen eingesetzt wird, dies zeigen z. B. [27] und [164]. Ein vergleichbares Projekt, der *Pixhawk* (*PX4*)-Autopilot (siehe [114]), wird federführend von der ETH Zürich entwickelt. Auch dieser wird bei wissenschaftlichen Anwendungen verbreitet genutzt, wie [56] und [179] belegen.

Parallelen zu dieser Arbeit sind vor allem in den Projekten zu finden, deren Zielsetzung es ist, einen plattformübergreifenden und kostengünstigen Autopiloten für professionelle und wissenschaftliche Anwendungen zu entwickeln. Dabei liegt der grundlegende Unterschied des hier vorgestellten *Frameworks*, gegenüber den anderen, in der echtzeitfähigen Simulation des MAVs in direkter Verbindung mit dem Autopiloten. Der Fokus liegt auf der Umsetzbarkeit und Evaluierung aller Softwarekomponenten, wie z. B. *Closed Loop Control*, Datenfilterung oder Navigationsberechnung, innerhalb des virtuellen MAVs. Dabei werden sämtliche Algorithmen des neuen Autopiloten von Grund auf in der Simulation geplant und umgesetzt. Die virtuelle Flugplattform ist darüber hinaus direkt mit der parallel enwickelten Bodenstation gekoppelt. So können Parameter wie die Kommunikationsbandbreite oder verschiedenste Systemfunktionalitäten von Beginn an mit berücksichtigt werden.

9.2 KONZEPT UND INTEGRATION

Um einen effizienten und möglichst einfachen Entwurf von neuen oder das Modifizieren bzw. Erweitern bestehender Flugplattformen zu ermöglichen, soll im Folgenden das Konzept einer neuartigen Entwicklungsumgebung von VTOL-MAVs vorgestellt werden. Grundgedanke dieses *Frameworks* ist der vollständig virtuelle Entwurf sowie die Evaluation in einer modellbasierten 3D-Echtzeit-Simulation. Hierzu wird ein neuer Softwarekern entwickelt, welcher besonders auf die Herausforderungen der simulationsbasierten Entwicklung eines effizienten und universellen *Automatic Flight Control System* (AFCS) ausgelegt ist. Dabei geht der Umfang dieses MAV-*Frameworks* über die reine Softwareentwicklung deutlich hinaus. So stellt der Hardwareentwurf des Autopiloten für den plattformunabhängigen Einsatz ebenfalls einen wichtigen Entwicklungsschritt dar. Aus diesem Grund wird neben dem Softwareentwurf eine vollständig neue Flugplattform entwickelt. Ein kostengünstiger Ansatz und die Nutzung von *Commercial-Off-The-Shelf* (COTS)-Komponenten, wie Motoren und Rotoren, stehen hierbei im Mittelpunkt, um eine attraktive und vielseitige Basis für zukünftige Arbeiten zu schaffen. Das Ziel ist es, eine vollständige Flugplattform in einem Kostenrahmen von unter 1000 € zu entwickeln. Die Grundlage für einen erfolgreichen Entwurf dieses *Framworks* ist dabei der strukturierte und detaillierte Softwareentwurf.

9.2.1 *Das Softwarekonzept*

Beim Entwurf des Softwarekonzeptes sind zwei Faktoren von entscheidender Bedeutung. Zum einen müssen die Algorithmen zur Sensordatenakquisition, -fusion und Maschinenregelung direkt zwischen

realem MAV und Simulation portierbar sein, zum anderen soll die Software mit geringstem Migrationsaufwand auf andere MAV-Plattformen angepasst werden können. Zu diesem Zweck werden für diesen Autopiloten verschiedene Ebenen (*System-Level*) eingeführt (siehe Abbildung 78). Im CPU-*Level* sind prozessorspezifische Funktionalitäten, z. B. *Universal Asynchronous Receiver and Transmitter* (UART), *Direct Memory Access* (DMA) oder *Interrupt Service Routines* (ISR), implementiert. Aufsetzend auf diesem *Layer* wird das *Low-Level* implementiert, welches die CPU-spezifischen Schnittstellen durch einheitliche Ein-/Ausgabe (*engl. Input/Output* (I/O))-Funktionen abbildet. Durch die Abstraktion der *Low-Level*-Funktionen wird eine einfache Überführung der *High-Level*-Funktionen auf unterschiedliche Prozessorarchitekturen geschaffen. Hierbei ist lediglich eine Anpassung des CPU-*Levels* notwendig, um den Autopiloten vollständig auf zukünftige, möglicherweise leistungsfähigere Prozessoren zu portieren. Der nächste *Layer* (*Board-Level*) basiert auf den generalisierten I/O-Funktionen. Er dient vordringlich der Implementierung aller Kommunikationsroutinen der benötigten externen Module, wie Sensoren, Motorendstufen oder Batterie-Management. Dieser *Layer* ist vergleichbar mit der Treiberebene moderner Betriebssysteme und erlaubt einen einfachen und unproblematischen Wechsel zwischen unterschiedlichen Sensormodulen, Motorendstufen und weiteren Komponenten. Hardwareelemente können so von verschiedenen MAVs mit dem Autopiloten kombiniert werden, indem einzig der spezifische Treiber geladen wird. Der oberste *Layer* (*Application-Level*) bildet die eigentliche

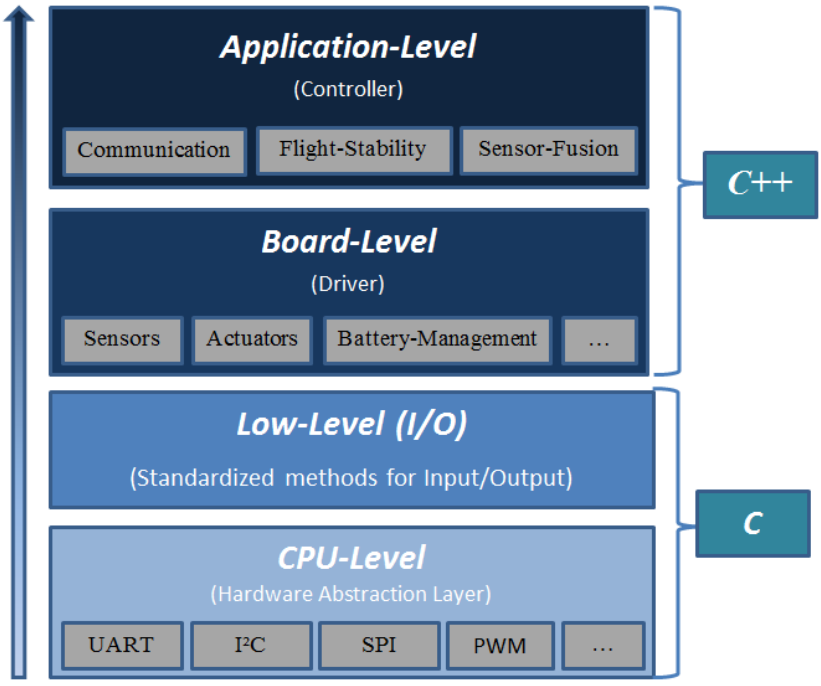


Abbildung 78: Softwarekonzept des *FireFly*-Autopiloten

Applikation. In diesem werden die Algorithmen zur Sensordatenfilterung, -fusion und Fluglageregelung umgesetzt. Ebenfalls sind darin die Kommunikations- und Navigationsroutinen eingebettet. In diesen *Layern* wird ein Ausgleich zwischen effizientem *Code* und abstrahierbaren Funktionen gesucht. Aus diesem Grund sind die unteren beiden Ebenen in effizientem C-*Code* programmiert. Die *High-Level* werden jedoch in objektorientiertem C++ implementiert, was der Treiberstruktur zugute kommt. Hierzu ist ein Klassenmodell mit festgelegten Methoden definiert, was einen Austausch von Algorithmen, Sensoren und/oder Kommunikationsmodulen vereinfacht. Durch dieses Schichtenmodell aus vier interagierenden Ebenen ist der SiL-Import in die Simulation unproblematisch. Hierzu wird ein simulationsspezifisches CPU-*Level* implementiert. Im *Board-Level* werden anstatt der Treiber für die realen Sensoren, Antriebe und weiteren Komponenten deren virtuelle Pendants geladen. Sämtliche Algorithmen, beispielsweise zur Datenfilterung, -fusion oder Navigation, bleiben auf dem realen und virtuellen System vollständig identisch.

9.2.2 *Die Hardwareauswahl*

Neben dem Softwarekonzept ist die Auswahl geeigneter Hardware für den Autopiloten und das Demonstrator-MAV ein ebenso wichtiger Bestandteil des Gesamtkonzepts. Einer der Schwerpunkte dieses neuen Entwicklungsframeworks liegt auf der Kosteneffizienz des späteren Gesamtsystems. Somit gilt es beispielsweise bei den Sensoren, einen optimalen Kompromiss zwischen Sensorgüte und Anschaffungskosten zu finden.

Inertialsensorik

Bei den Inertialsensorsystemen wird entsprechend ihrer Güte zwischen vier Kategorien unterschieden (siehe [167]):

- *Marine & Navigation Grade*
- *Tactical Grade*
- *Industrial Grade*
- *Automotive & Consumer Grade*

Dabei sind diejenigen der Kategorie *Marine & Navigation Grade* die genauesten und teuersten Sensorsysteme. Diese basieren in der Regel auf *Fiber Optic Gyroscope* (FOG)- oder *Ring Laser Gyroscope* (RLG)-Kreiselsystemen. Neben dem Preis von $\approx$ 100 000 € sind diese Systeme allein schon durch ihre mechanischen Dimensionen und ihr Gewicht für die hier betrachteten Multirotorsysteme nicht von Relevanz. Im Gegensatz dazu stehen die zu den Kategorien der *Tactical Grade* oder *Industrial Grade* gehörenden Sensoren. Aufgrund der

MEMS-Technologie beträgt beispielsweise bei der *ADIS16485* (siehe [9]) von Analog Devices die Baugröße lediglich 47 mm × 44 mm × 14 mm bei einem Gewicht von 48 g. Somit wäre dieses Sensorsystem durchaus interessant für MAV-Anwendungen. Jedoch liegen auch die Messsysteme dieser Kategorien mit einem Anschaffungspreis von 300 € bis über 1500 € deutlich außerhalb der anvisierten Gesamtkosten der Flugplattform. Aufgrund der angestrebten Kosteneffizienz wird daher der Fokus auf die kostengünstigen *Automotive & Consumer Grade*-Sensoren gelegt. Zudem sind neben den Anschaffungskosten zwei weitere Parameter von Bedeutung. So sollte der Drehratensensor mindestens einen Messbereich von ±800°/s bis ±1000°/s haben, um

Tabelle 4: Aufstellung möglicher MEMS-IMUs

Hersteller	Bosch	ST	Invensense
Bezeichnung	*BMI055* [21]	*LSM9DS0* [153]	*MPU-9150* [83]
Schnittstelle	SPI/I^2C	SPI/I^2C	SPI/I^2C
	Drehratensensor		
Messbereich (FS)	±2000°/s	±2000°/s	±2000°/s
Auflösung (ADC)	16 bit	16 bit	16 bit
Quantisierung	0.007 $\frac{°/s}{LSB}$	0.008 $\frac{°/s}{LSB}$	0.007 $\frac{°/s}{LSB}$
Bias	±1°/s	±25°/s	±20°/s
Rauschdichte	0.014 $\frac{°/s}{\sqrt{Hz}}$	N/A	0.005 $\frac{°/s}{\sqrt{Hz}}$
Nichtlinearität	±0.05 % FS	N/A	±0.2 % FS
Updaterate	2000 Hz	760 Hz	8000 Hz
	Beschleunigungssensor		
Messbereich (FS)	±16 g	±16 g	±16 g
Auflösung (ADC)	12 bit	16 bit	16 bit
Quantisierung	0.9 $\frac{mg}{LSB}$	0.06 $\frac{mg}{LSB}$	0.06 $\frac{mg}{LSB}$
Bias	±80 mg	±60 mg	±80 mg
Rauschdichte	150 $\frac{\mu g}{\sqrt{Hz}}$	N/A	400 $\frac{\mu g}{\sqrt{Hz}}$
Nichtlinearität	±0.5 % FS	N/A	±0.5 % FS
Updaterate	2000 Hz	760 Hz	1000 Hz
	Allgemein		
Abmessungen	3.0 × 4.5 × 1.0 mm^3	4.0 × 4.0 × 1.0 mm^3	4.0 × 4.0 × 1.0 mm^3
Preis	< 10 €	< 10 €	< 10 €

der späteren Flugplattform ebenfalls hochdynamische Flugmanöver zu ermöglichen. Des Weiteren sollte der Messbereich des Beschleunigungssensors nicht kleiner als ±4 g betragen. In Tabelle 4 sind drei den Anforderungen entsprechende IMUs gegenübergestellt. Alle diese Sensoren bestehen aus einem 3-achsigen Beschleunigungssensor sowie einem 3-achsigen Gyroskop. Die Kenndaten sind bei allen drei Sensoren sehr ähnlich. Demnach können die Messbereiche der Drehratensensoren auf bis zu ±2000°/s und die der Beschleunigungssensoren auf bis zu ±16 g eingestellt werden. Ebenfalls liegen sie preislich alle unter 10 € und unterscheiden sich damit nur minimal. Der Sensor von Bosch ist hierbei mit rund 6 € der günstigste. Jedoch liegt dieser Sensor mit einer Auflösung der Beschleunigung von 12 bit unter der Auflösung der beiden anderen, die eine Auflösung von 16 bit bereitstellen. Im Gegensatz hierzu schneidet der *BMI055* (siehe [21]) am besten in dem Bias (*Zero-Rate Offset*) des Drehratensensors mit lediglich ±1°/s ab. Der *LSM9DS0* (siehe [153]) weist hingegen bei der Updaterate mit 760 Hz den schlechtesten Wert auf. Rauschdichte und Nichtlinearität sind zudem im Datenblatt nicht angegeben. Der *MPU9150* (siehe [83]) zeichnet sich durch die kleinste Rauschdichte der betrachteten Drehratensensoren aus. Darüberhinaus hat dieser Sensor jeweils eine Auflösung von 16 bit für die Gyroskope und für die Beschleunigungssensoren. Die Updaterate berägt 8000 Hz bzw. 1000 Hz. Des Weiteren ist im *MPU-9150* und im *LSM9DS0* zusätzlich ein 3-achsiger Magnetfeldsensor integriert. Aufgrund der geringen Unterschiede zwischen dem *MPU-9150* und dem *BMI055* Sensorsystem ist die Entscheidung zu Gunsten des *MPU-9150* gefallen. Die Kenndaten dieses Sensorsystems passen sehr gut zu den hier gestellten Anforderungen. Nicht zuletzt gab die sehr gute und vielfältige Verfügbarkeit von Evaluierungsboards (*engl. Breakout Boards*) den Ausschlag für den Sensor von Invensense.

Magnetometer

Analog zu der Inertialsensorik beschränkt sich die Auswahl der Magnetfeldsensorik auf die *Consumer Grade*-Sensoren im Preissegment unter 100 €. Die Magnetfeldsensorik dient vordringlich der Bestimmung des lokalen Erdmagnetfelds und soll als elektronischer Kompass zur Navigationsstützung dienen. Infolgedessen ergibt sich der geforderte Messbereich von ±67 µT (siehe Abschnitt 4.2). Bei dieser Sensorauswahl werden drei unterschiedliche Systeme betrachtet, wie Tabelle 5 auf Seite 164 zeigt. Dabei ist jedes Sensorsystem 3-achsig ausgelegt und entspricht mit einem Messbereich von ±800 µT bzw. ±1200 µT den gestellten Anforderungen. Das Sensorsystem *RM3100* (siehe [137]) von PNI Sensor Corporation zeigt dabei von allen betrachteten Sensoren die höchste Quantisierung mit 0.013 $\mu T/_{LSB}$ mit gleichzeitig größtmöglicher Abtastrate von bis zu 530 Hz. Nachteilig

Tabelle 5: Aufstellung möglicher Magnetometer

Hersteller	Honeywell	PNI Sensor Corporation	AsahiKASEI
Bezeichnung	*HMC5883L* [75]	*RM3100* [137]	*AK8975* [12]
Schnittstelle	SPI/I²C	SPI/I²C	SPI/I²C
Messbereich (FS)	±810 µT	±800 µT	±1200 µT
Auflösung (ADC)	12 bit	N/A	13 bit
Quantisierung	0.073 $\frac{\mu T}{LSB}$	0.013 $\frac{\mu T}{LSB}$	0.3 $\frac{\mu T}{LSB}$
Rauschen	0.2 µT	0.015 µT	N/A
Nichtlinearität	±0.1 % FS	±0.5 % FS	N/A
Updaterate	160 Hz	147 – 530 Hz	110 Hz
		Allgemein	
Abmessungen	3.0 × 3.0 × 0.9 mm³	25.4 × 25.4 × 9.6 mm³	4.0 × 4.0 × 0.8 mm³
Preis	< 3 €	≈ 50 €	< 3 €

wirken sich bei diesem Sensor die nicht minder wichtigen Faktoren Größe und Preis aus, welche jeweils etwa um eine Zehnerpotenz größer sind als bei dem *HMC5883L* (siehe [75]) von Honeywell sowie dem *AK8975* (siehe [12]) von AsahiKASEI. Daher fiel die Entscheidung auf den *HMC5883L* Magnetfeldsensor. Dieser überzeugte durch den einstellbaren Messbereich von ±88 µT bis ±810 µT (*Full Scale* (FS)). So kann der Sensor auf einen Messbereich von ±88 µT (FS) voreingestellt werden, womit sich eine Quantisierung von 0.073 µT/LSB ergibt. Die Updaterate von 160 Hz ist für diese Applikation ausreichend. Besonders vorteilhaft ist bei diesem Sensor die sehr kleine Bauform von 3.0 mm × 3.0 mm × 0.9 mm und der Preis von weniger als 3 €. Obwohl der *AK8975* schlechter als der *HMC5883L* abschneidet, kann der Sensor auch genutzt werden, da er als Magnetfeldsensor in dem ausgewählten Inertialsensorsystem der *MPU9150* zusätzlich werkseitig integriert ist.

Barometer

Das Barometer dient dem zu entwickelnden Autopiloten zur Stützung der Flughöhe in Relation zum Startpunkt. Besonders wichtig ist die Quantisierung des gemessenen Luftdrucks, da eine Höhenänderung von 1 cm lediglich eine Luftdruckänderung von 0.0012 hPa erzeugt. Die in Tabelle 6 betrachteten Sensoren haben somit etwa ei-

Tabelle 6: Aufstellung möglicher Barometer

Hersteller	Bosch	Freescale / NXP	Measurement Specialties
Bezeichnung	*BMP180* [20]	*MPL3115A2* [130]	*MS5611-01 BA03* [113]
Schnittstelle	I^2C	I^2C	SPI/I^2C
Messbereich (FS)	300 – 1100 hPa	500 – 1100 hPa	10 – 1200 hPa
Auflösung (ADC)	N/A	24 bit	24 bit
Quantisierung	0.01 $\frac{\text{hPa}}{\text{LSB}}$	0.015 $\frac{\text{hPa}}{\text{LSB}}$	0.012 $\frac{\text{hPa}}{\text{LSB}}$
absoluter Fehler	±4.0 hPa	±4.0 hPa	±3.5 hPa
relativer Fehler	±0.12 hPa	±1.0 hPa	±0.5 hPa
Updaterate	20 – 200 Hz	100 Hz	120 – 2000 Hz
		Allgemein	
Abmessungen	$3.0 \times 3.0 \times 0.9\,\text{mm}^3$	$5.0 \times 3.0 \times 1.1\,\text{mm}^3$	$5.0 \times 3.0 \times 1.0\,\text{mm}^3$
Preis	< 5 €	< 5 €	< 10 €

ne Quantisierung von ≈ 10 cm. Obwohl der Sensor von Bosch (siehe [20]) bei einem geringeren Anschaffungspreis einen etwas geringeren relativen Fehler aufweist, fiel die Entscheidung trotzdem auf den *MS5611-01BA03* (siehe [113]) von Measurement Specialties, da dieser deutlich höhere Updateraten zulässt. Somit hat dieser Sensor selbst bei der höchsten Genauigkeit immer noch eine Aktualisierungsrate von 120 Hz. Der Preis von etwas weniger als 10 € liegt ebenfalls im hier angestrebten Kostenbereich. Im Vergleich zum *MPL3115A2* (siehe [130]) von NXP schneidet der *MS5611-01BA03* in allen Kategorien, bis auf den Preis, gleichfalls besser ab.

Somit ist die Auswahl der grundlegenden Sensorelemente mit der IMU, einem Magnetfeldsensor sowie einem Barometer abgeschlossen. Insgesamt liegen die Anschaffungskosten des gesamten Sensorsystems bei deutlich unter 25 €.

Antriebe und Mikrocontroller

Neben dem eigentlichen Autopiloten wird auch ein MAV-Prototyp zur Validierung entwickelt. Inspiriert wurde dieses Konzept durch die *AR.Drone* (siehe [134]), vor allem durch deren niedrigen Preis und die gute Verfügbarkeit von einzelnen Komponenten. Daher werden in dem folgend entwickelten Funktionsprototypen die Motoren, Roto-

ren und Motortreiber der *AR.Drone* verwendet. Außerdem zeigte sich in ersten Voruntersuchungen eine sehr gute Effizienz dieser Antriebe. Der Kostenumfang für die genannte Antriebskonfiguration des geplanten Quadrokopters liegt bei rund 200 €.

Es wird an dieser Stelle noch keine Auswahl eines Mikrocontrollers getroffen, da zuvor die Software für den Autopiloten in der Simulation entwickelt wird, um darauf aufbauend die Leistungsparameter des Prozessors zu bestimmen.

9.2.3 *Die Simulationsintegration*

Prinzipiell erfolgt die Simulationsintegration nach dem selben Verfahren wie schon in Kapitel 6 vorgestellt. Konzeptionell gilt deshalb auch an dieser Stelle als Grundlage der Aufbau aus Abbildung 17 auf Seite 70. Im Gegensatz zu der in Kapitel 8 vorgestellten Integration und Modellierung eines vorhandenen Autopiloten, liegt der Fokus in diesem Abschnitt darauf, innerhalb der Simulation ein vollständig neues *Flight Management System* (FMS) zu entwickeln. Nachdem vorab die zu nutzenden Sensoren ausgewählt worden sind, müssen diese modelliert werden. Als Rahmenparameter werden im ersten Schritt die in Tabelle 7 dargestellten konservativen Werte gewählt. Daraufhin werden die Sensoren in gleicher Weise, wie schon in Abschnitt 8.1 gezeigt, modelliert. Es wird jedoch darauf verzichtet, die vollständigen Messmodelle und Datenplots aufzuzeigen, da die grundlegende Vorgehensweise zuvor schon ausführlich erläutert worden ist. Aber ein anderer wichtiger Aspekt soll erneut aufgegriffen werden; dies betrifft die Verteilung des Sensorrauschens. Wie in Kapitel 8.1.1 gezeigt, ist bei der IMU des bisher genutzten Autopiloten durch die suboptimale externe Verschaltung und eine zu geringe Auflösung des *Analog-to-Digital-Converters* (ADCs) ein nicht normalverteiltes Rauschspektrum nachzuweisen. Dies ist besonders bei der folgenden Entwicklung eines Kalman-Filters zur Sensordatenfusion eine problematische Eigenschaft, welche gesondert betrachtet werden muss. Aus diesem Grund ist die hier ausgewählte Sensorik dahingehend untersucht worden.

Tabelle 7: Gewählte Parameter der Sensorik

Sensor	Gyroskop	Beschleunigung	Magnetometer	Barometer
Messbereich	$\pm 1000\,°/s$	$\pm 4\,g$	$\pm 130\,\mu T$	10 – 1200 hPa
Quantisierung	$0.0305\,\frac{°/s}{LSB}$	$0.12\,\frac{mg}{LSB}$	$0.092\,\frac{\mu T}{LSB}$	$0.012\,\frac{hPa}{LSB}$
Updaterate	1000 Hz	1000 Hz	160 Hz	120 Hz

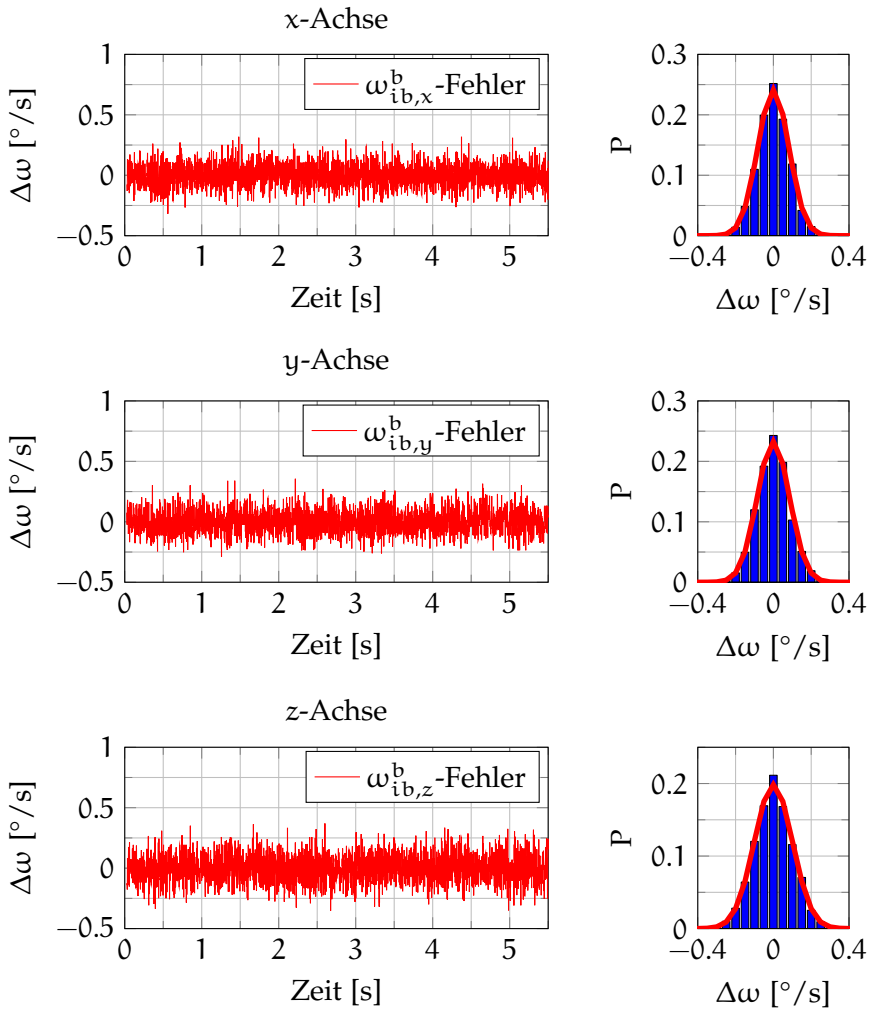


Abbildung 79: Sensorfehler (*links*) und Histogramm der Rauschverteilung (*rechts*) nach der Kalibrierung des Drehratensensors

In Abbildung 79 ist der Fehler der ausgewählten Drehratensensorik von der bekannten Messtrajektorie (siehe Abschnitt 8.1.1 auf Seite 91 in Abbildung 21) nach der Sensorkalibrierung dargestellt. Hierbei zeigt sich, dass sich bei den gewählten Sensoren der Fehler durch das Rauschen des Sensors selbst ergibt. Untersucht man die Verteilung des Rauschens, resultiert daraus der normalverteilte Histogrammplot (*rechts* dargestellt). Zur Verdeutlichung wird über dem Histogrammplot die Wahrscheinlichkeitsdichtefunktion aus dem errechneten Mittelwert und der Standardabweichung in *rot* gelegt. Durch diese Auswertung bestätigt sich die Annahme, dass das Rauschen des hier genutzten Drehratensensors normalverteilt und mittelwertfrei ist. Die Standardabweichung beträgt dabei $\sigma = 0.08°/s$. Des Weiteren ist in Abbildung 80 auf Seite 168 der Messfehler des Beschleunigungssensors nach der Sensorkalibrierung dargestellt. Auch hier bestimmt sich der Fehler hauptsächlich durch das Sensorrauschen. Wird mit diesen Daten eine Verteilungsanalyse durchgeführt, folgt auch hier

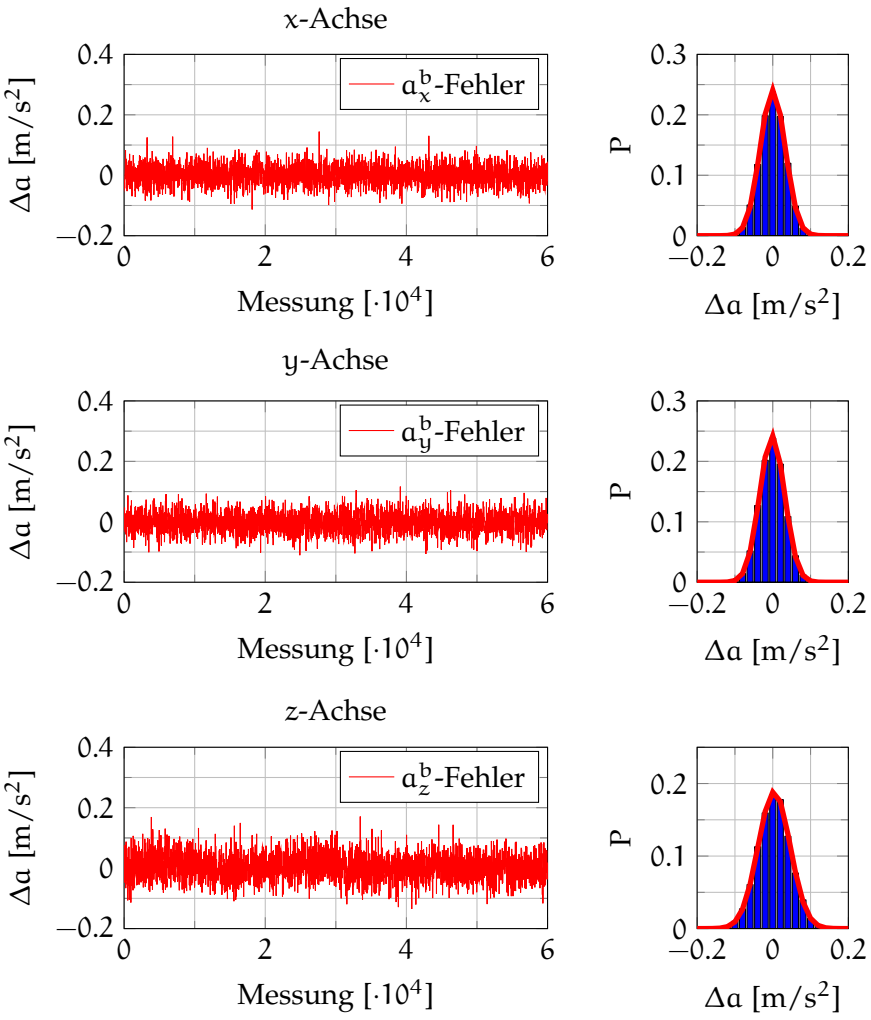


Abbildung 80: Sensorfehler (*links*) und Histogramm der Rauschverteilung (*rechts*) nach der Kalibrierung des Beschleunigungssensors

die erwartungsgetreue Normalverteilung des Rauschens (in Abbildung 80 *rechts* aufgeführt). Die Standardabweichung beträgt dabei $\sigma = 0.03\,m/s^2$. Ähnliche Normalverteilungen gelten für das genutzte Magnetometer mit einer Standardabweichung von $\sigma = 0.147\,\mu T$ sowie für das Baro-Altimeter mit einer Standardabweichung von $\sigma = 0.114\,m$.

Gemäß der in Abschnitt 7 dargelegten Modellierung, werden diese Sensormodelle in die Simulation integriert. Wie zuvor im Softwarekonzept (siehe Abschnitt 9.2.1) vorgestellt, werden hierzu im *Board-Level* anstatt der Treiber für die realen Sensoren, deren virtuelle Pendants auf Basis der ermittelten Sensoreigenschaften und Modellierungsfunktionen implementiert. Analog der bisherigen Flugplattformmodellierung werden die Modelle der Antriebe ermittelt und im *Board-Level* simuliert. Dabei konnte bei der Modellierung der Antriebe der *AR.Drone* eine Besonderheit festgestellt werden: Im Gegensatz

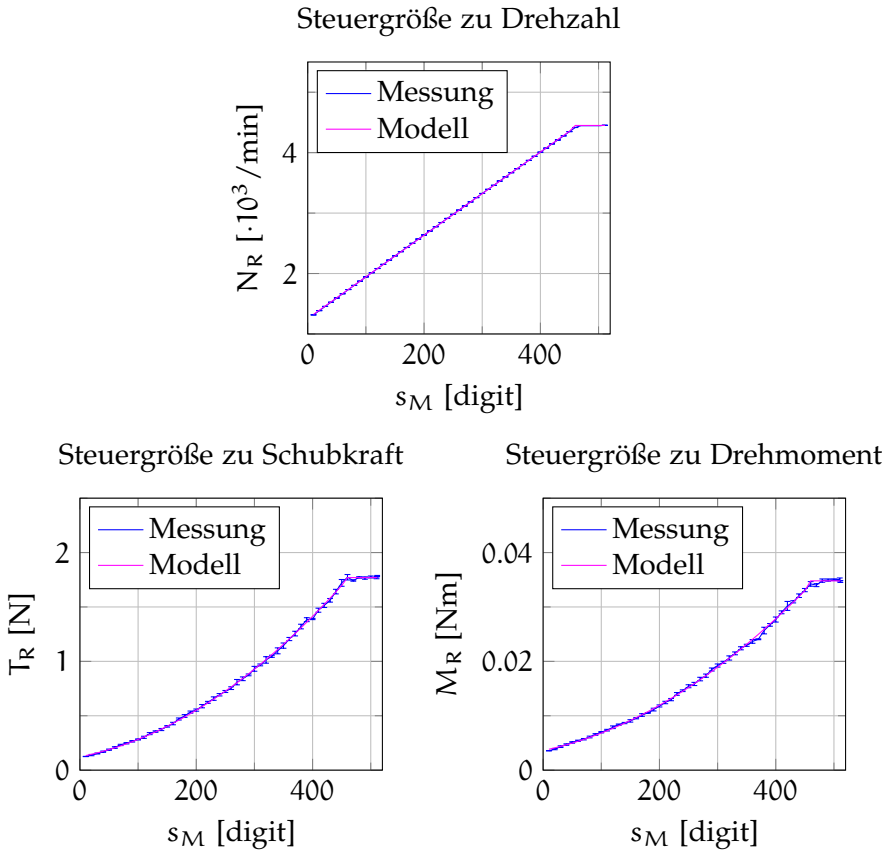
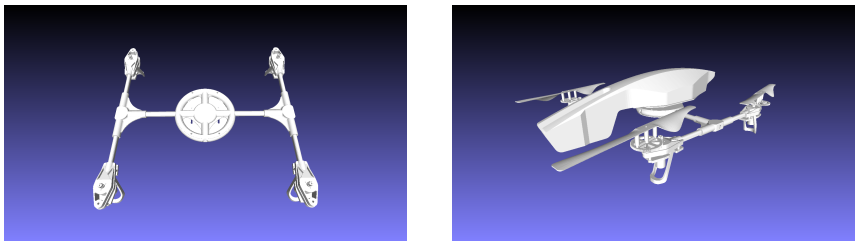


Abbildung 81: Stationäres Modell der *AR.Drone*-Antriebe: Steuergröße (s_M) zu Drehzahl (N_R), Schubkraft (T_R) und Drehmoment (M_R)

zu den bisher genutzten Motorendstufen, welche einen quadratischen Zusammenhang zwischen Steuergröße und Drehzahl aufweisen, zeigen die Antriebe der *AR.Drone* eine lineare Abhängigkeit (siehe Abbildung 81 *oben*). Das lässt darauf schließen, dass die Motorelektronik eine integrierte Drehzahlregelung hat. Dies hat für die Modellierung der Schubkräfte und der Drehmomente, aber auch für die spätere Systemregelung einen besonderen Vorteil, da durch diese integrierte Regelung der Zusammenhang zwischen Steuergröße und Drehzahl nicht von der Versorgungsspannung abhängig ist. Außerdem kann somit ein direktes Modell zwischen Steuergröße und Schubkraft sowie zwischen Steuergröße und Drehmoment abgeleitet werden (siehe Abbildung 81 *unten links* und *unten rechts*). Dabei zeigt dieses Modell die selbe quadratische Abhängigkeit wie das zuvor in Abschnitt 7.2.1 hergeleitete Modell zwischen Drehzahl und Schubkraft sowie zwischen Drehzahl und Drehmoment. Das Abknicken der Kennlinie im oberen Bereich ist auf eine aktive Begrenzung der Motorelektronik zurückzuführen. Diese wird im Modell berücksichtigt. Hierbei ist zu beachten, dass in diesem Fall die Rotordrehzahl N_R gemessen worden ist, da der Antrieb der *AR.Drone* ein Getriebe mit dem Übersetzungsverhältnis $N_R = {}^1/_{8.625} \cdot N_M$ zwischen Motor und Rotor besitzt.



(a) CAD-Modell des neuen Frames in *H*-Konfiguration

(b) CAD-Modell des neuen FireBird-X4

Abbildung 82: CAD-Modelle der neuen *FireBird-X4*-Entwicklungsplattform

Die Modellierungsparameter des Trägheitsmoments und des Massenschwerpunkts werden wie bisher aus dem entwickelten CAD-Modell entnommen. Für eine erste Abschätzung der Parameter war es hilfreich, dass einzelne Baugruppen der *AR.Drone*, beispielsweise die Antriebe oder die Außenhülle, als CAD-Modell bereits bei der Onlineplattform *GrabCad* (siehe [61]) zur Verfügung standen. Zur flexibleren Auslegung der Entwicklungsplattform wurde jedoch ein eigenes *Frame* aus Leichtbaukomponenten in *H*-Form entworfen (siehe Abbildung 82a). Das Gesamtkonzept ist in Abbildung 82b dargestellt.

Nach erfolgreicher Integration der beschriebenen Modelle erfolgt die Softwareentwicklung des neuen Autopiloten. Dabei werden sämtliche Algorithmen zur Datenfilterung, -fusion, Navigation etc. auf dem virtuellen System realitätsgetreu entwickelt und evaluiert.

9.3 DATENFILTERUNG UND SENSORFUSION

Ein elementarer Aspekt bei der Entwicklung des Autopiloten ist die Datenfilterung. Hierbei müssen verschiedene Fehler und externe Einflussfaktoren betrachtet und möglichst optimal kompensiert werden.

9.3.1 *IIR-Filterdesign*

Bei der Modellierung der Antriebe ist neben den Antriebseigenschaften der Einfluss von Vibrationen, verursacht durch die Drehzahl der Rotoren, untersucht worden. Dabei konnte festgestellt werden, dass diese Vibrationen einen erheblichen Einfluss auf die Inertialsensorik ausüben und folglich die Messwerte verfälschen können. Zur Bestimmung dieser Störeinflüsse wurden Messdaten der IMU mit einer Abtastrate von 1000 Hz aufgenommen und mittels einer diskreten Fourier-Transformation (*engl. Discrete Fourier Transform* (DFT)) untersucht. Zur Bestimmung der Störfrequenzen wurden die Antriebe in drei stationären Zuständen betrieben. Die Motorsteuergrößen betrugen dabei 50 % $\widehat{=}$ 3037 $^1/_{\text{min}}$ (entspricht $\approx$ 50.6 Hz), 100 % $\widehat{=}$ 4290 $^1/_{\text{min}}$ (entspricht $\approx$ 71.5 Hz) und 64 % $\widehat{=}$ 3512 $^1/_{\text{min}}$ (entspricht

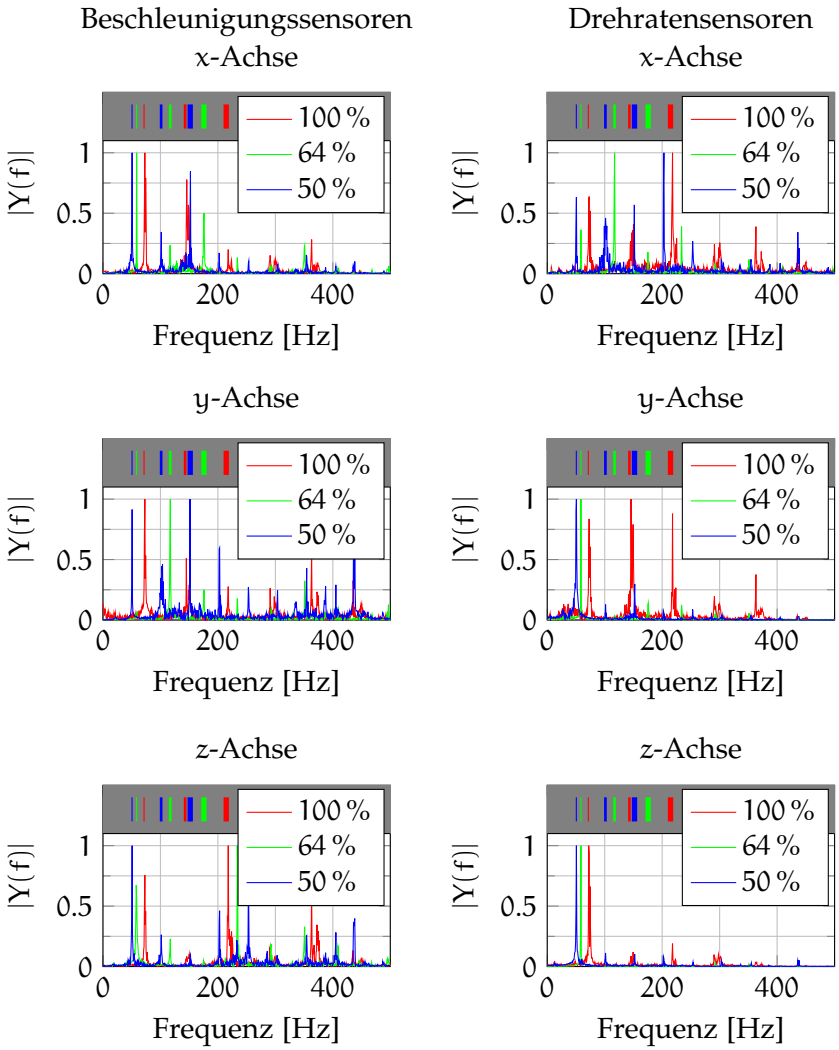


Abbildung 83: FFT-Untersuchung der Motorvibrationen

≈ 58.5 Hz), dabei entsprechen die 64 % der errechneten Steuergröße für den Schwebezustand. Die Ergebnisse der schnellen Fourier-Transformation (*engl. Fast Fourier Transform* (FFT)) für alle drei Achsen sind in Abbildung 83, jeweils *links* für den Beschleunigungssensor und *rechts* für den Drehratensensor, dargestellt. Im oberen *grau* hinterlegten Bereich ist zudem die errechnete Erregerfrequenz der gemessenen Rotordrehzahl und deren erste und zweite Oberwelle gekennzeichnet. Es ist deutlich zu erkennen, dass das Störspektrum direkt mit der Rotordrehzahl korreliert. Um einen optimalen Filter zu designen, ist diese Einflussgröße ebenfalls in der Simulation modelliert worden. Zur Kompensation wird ein *Butterworth*-Filter zweiter Ordnung entwickelt, da dieses sich besonders durch den monotonen Amplitudengang im Durchlass- und im Sperrbereich auszeichnet. Dieses wird als IIR-Filter in den Autopiloten implementiert. Die Ergebnisse des IIR-Filters sind in Abbildung 84 auf Seite 172 dargestellt. Dabei

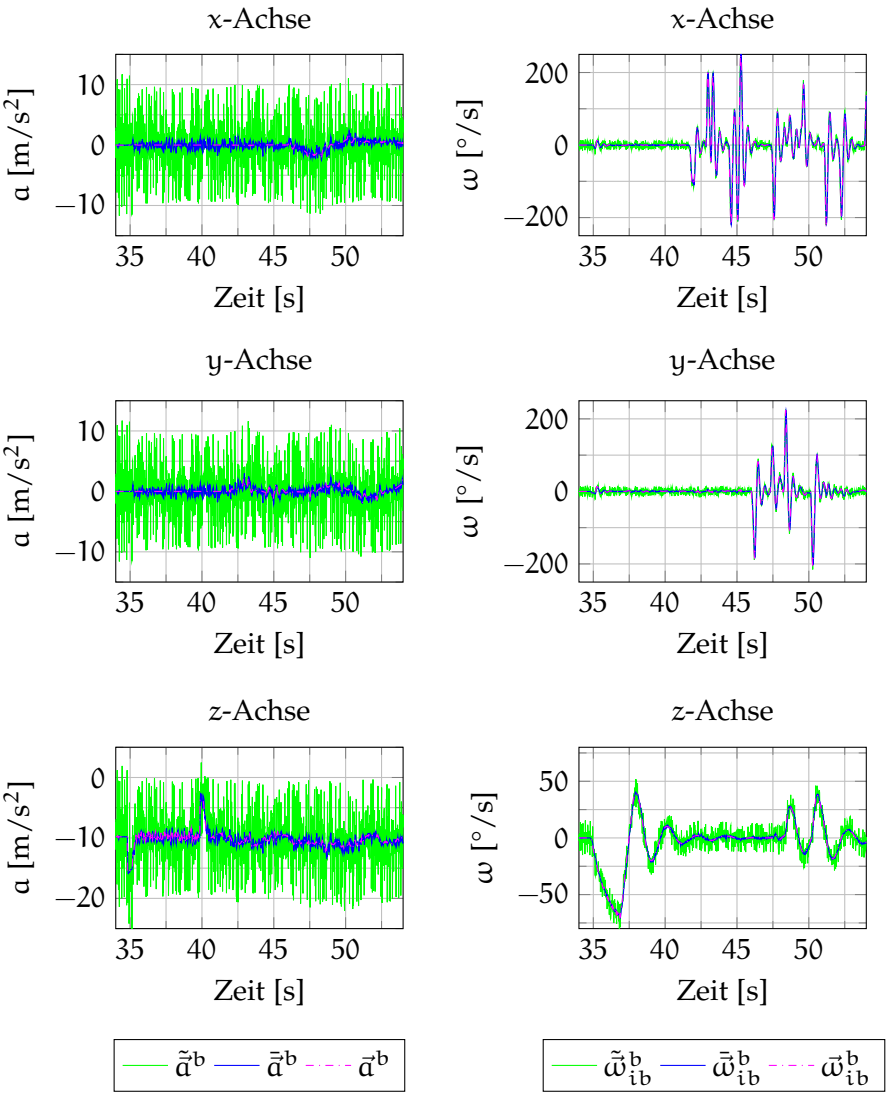


Abbildung 84: Ergebnisse des IIR-Filters

sind in *grün* jeweils die verrauschten Sensordaten, in *blau* die Sensordaten nach der Filterung und in *magenta* die *Ground-Truth*-Werte der Sensoren abgebildet. Die sehr gute Filterwirkung ist deutlich erkennbar. Außerdem werden Signalspitzen, sowohl bei den Beschleunigungswerten als auch bei den Drehraten, präzise vom Filter wiedergegeben. Gerade bei derartigen Filterdesigns ist eine besondere Stärke der hier gezeigten simulationsbasierten Entwicklung erkennbar. So können Filterparameter mit realistischen Messwerten optimiert werden, wobei ein Vergleich mit den tatsächlichen Zustandsgrößen zu jeder Zeit möglich ist. So kann sogar während der Signalverarbeitung die filterspezifische Gruppenlaufzeit in Echtzeit bestimmt werden.

9.3.2 *Lage-Kalman-Filter*

Mit den Messdaten der IMU soll im Folgenden die Orientierung des MAVs berechnet werden. Grundsätzlich geschieht dies auf Basis des *Strapdown*-Algorithmus (siehe Abschnitt 4.1.2). Dabei wird die Orientierung durch Integration der Drehraten bestimmt. Da bei diesem Autopiloten jedoch eine *Low-Cost Consumer Grade*-MEMS-Inertialsensorik verwendet wird, welche, im Gegensatz zu hochwertiger *Marine & Navigation Grade*-Sensorik, einen nicht zu vernachlässigenden Bias bzw. Bias-Drift besitzt, würde die Lageberechnung ohne weitere Stützinformationen sehr schnell mit deutlichen Fehlern behaftet sein. Aus diesem Grund wird das in [115] und [172] vorgestellte *Error State Space* Kalman-Filter zur Lagestützung implementiert[2]. Die folgende Herleitung ist an diese Arbeiten angelehnt, jedoch sind in den genannten Arbeiten abweichende Definitionen für $\Delta\vec{x}(t)$ genutzt worden, was zu unterschiedlichen Vorzeichen in der Betrachtung führt. In dieser Darstellung wird, wie in Abschnitt 5.3.2 hergeleitet, die einheitliche Definition $\Delta\vec{x}(t) = \vec{x}(t) - \hat{\vec{x}}(t)$ verwendet.

Das Zustandsraummodell

Ausgangspunkt ist das zeitkontinuierliche Zustandsraummodell (siehe Herleitung in Abschnitt 5.3.2) in der Form:

$$\Delta\dot{\vec{x}}(t) = \mathbf{F}(t)\Delta\vec{x}(t) + \mathbf{G}(t)\vec{w}(t). \tag{9.1}$$

Im Zustandsvektor werden die drei Fehler der Winkelfehler ($\Delta\vec{\Psi}$) sowie die drei Fehler des Drehratenbias ($\Delta\vec{b}_\omega$) geschätzt. Somit gilt:

$$\Delta\vec{x}(t) = \begin{pmatrix}\Delta\vec{\Psi} & \Delta\vec{b}_\omega\end{pmatrix}^T \tag{9.2}$$

$$= \begin{pmatrix}\alpha, & \beta, & \gamma, & \Delta b_{\omega,x}, & \Delta b_{\omega,y}, & \Delta b_{\omega,z}\end{pmatrix}^T. \tag{9.3}$$

Zur Herleitung wird zunächst die Differentialgleichung der Winkelfehler bestimmt. Entsprechend [172] ergibt sich:

$$\dot{\vec{\Psi}} = -\vec{\omega}^n_{in} \times \vec{\Psi} + \Delta\vec{\omega}^n_{in} - \hat{\mathbf{C}}^n_b \Delta\vec{\omega}^b_{ib}. \tag{9.4}$$

Dabei kann dieser Term, analog der Ausführung in Abschnitt 4.1.2, aufgrund der Güte der verwendeten Sensoren vereinfacht werden zu:

$$\dot{\vec{\Psi}} = -\hat{\mathbf{C}}^n_b \Delta\vec{\omega}^b_{ib}. \tag{9.5}$$

Mit dem Fehlermodell des Drehratensensors aus Abschnitt 7.1.1 und unter der Bedingung, dass die *Misalignment*-Matrix $\mathbf{M}$ als ideal angenommen wird ($\mathbf{M} = \mathbf{I}$), folgt:

$$\tilde{\vec{\omega}}^b_{ib} = \mathbf{I}\vec{\omega}^b_{ib} + \vec{b}_\omega + \vec{n}_\omega. \tag{9.6}$$

2 An dieser Stelle stand nicht der Entwurf eines Kalman-Filters im Vordergrund, sondern die simulationsbasierte Optimierung.

Der Drehratenfehler ergibt sich damit zu:

$$\Delta\vec{\omega}_{ib}^{b} = (\tilde{\vec{\omega}}_{ib}^{b} - \vec{\omega}_{ib}^{b}) \tag{9.7}$$

$$= \vec{b}_{\omega} + \vec{n}_{\omega}. \tag{9.8}$$

Eingesetzt in Gleichung 9.5 folgt:

$$\dot{\vec{\Psi}} = -\hat{\mathbf{C}}_{b}^{n}\left(\vec{b}_{\omega} + \vec{n}_{\omega}\right). \tag{9.9}$$

Dieser Term kann umformuliert werden zu:

$$\dot{\vec{\Psi}} = -\hat{\mathbf{C}}_{b}^{n}\vec{b}_{\omega} + \underbrace{\left(-\hat{\mathbf{C}}_{b}^{n}\vec{n}_{\omega}\right)}_{\mathbf{G}\vec{w}}. \tag{9.10}$$

Somit wird klar, dass die Winkelfehler ($\dot{\vec{\Psi}}$) eine Funktion des Drehratensensorbias ($\vec{b}_{\omega}$) sind. Wird diese Funktion entsprechend einer Taylor-Reihenentwicklung linearisiert (siehe Gleichung 5.39 und 5.40 auf Seite 58), dann folgt:

$$\dot{\vec{\Psi}} \approx \vec{f}\left(\bar{\vec{b}}_{\omega}\right) + \left.\frac{\partial\vec{f}\left(\vec{b}_{\omega}\right)}{\partial\vec{b}_{\omega}}\right|_{\vec{b}_{\omega}=\bar{\vec{b}}_{\omega}}\left(\vec{b}_{\omega} - \bar{\vec{b}}_{\omega}\right) + \mathbf{G}\vec{w} \tag{9.11}$$

sowie

$$\dot{\hat{\vec{\Psi}}} \approx \vec{f}\left(\bar{\vec{b}}_{\omega}\right) + \left.\frac{\partial\vec{f}\left(\vec{b}_{\omega}\right)}{\partial\vec{b}_{\omega}}\right|_{\vec{b}_{\omega}=\bar{\vec{b}}_{\omega}}\left(\hat{\vec{b}}_{\omega} - \bar{\vec{b}}_{\omega}\right). \tag{9.12}$$

Mit der Definition $\Delta\dot{\vec{\Psi}} = \dot{\vec{\Psi}} - \dot{\hat{\vec{\Psi}}}$ führt dies zu:

$$\underbrace{\dot{\vec{\Psi}} - \dot{\hat{\vec{\Psi}}}}_{\Delta\dot{\vec{\Psi}}} = \underbrace{\left.\frac{\partial\vec{f}\left(\vec{b}_{\omega}\right)}{\partial\vec{b}_{\omega}}\right|_{\vec{b}_{\omega}=\bar{\vec{b}}_{\omega}}}_{-\hat{\mathbf{C}}_{b}^{n}}\underbrace{\left(\vec{b}_{\omega} - \hat{\vec{b}}_{\omega}\right)}_{\Delta\vec{b}_{\omega}} + \underbrace{\mathbf{G}\vec{w}}_{-\hat{\mathbf{C}}_{b}^{n}\vec{n}_{\omega}}. \tag{9.13}$$

Die Bias-Drift selbst wird indes als *Random-Walk*-Prozess modelliert. Ergebnis dessen ist:

$$\Delta\dot{\vec{b}}_{\omega} = \vec{n}_{b_{\omega}}. \tag{9.14}$$

Somit kann das vollständige zeitkontinuierliche Zustandsraummodell beschrieben werden als:

$$\underbrace{\begin{pmatrix}\Delta\dot{\vec{\Psi}}\\ \Delta\dot{\vec{b}}_{\omega}\end{pmatrix}}_{\Delta\dot{\vec{x}}(t)} = \underbrace{\begin{pmatrix}0 & -\hat{\mathbf{C}}_{b}^{n}\\ 0 & 0\end{pmatrix}}_{\mathbf{F}(t)}\underbrace{\begin{pmatrix}\Delta\vec{\Psi}\\ \Delta\vec{b}_{\omega}\end{pmatrix}}_{\Delta\vec{x}(t)} + \underbrace{\begin{pmatrix}-\hat{\mathbf{C}}_{b}^{n} & 0\\ 0 & \mathbf{I}\end{pmatrix}}_{\mathbf{G}(t)}\underbrace{\begin{pmatrix}\vec{n}_{\omega}\\ \vec{n}_{b_{\omega}}\end{pmatrix}}_{\vec{w}(t)}. \tag{9.15}$$

Das Messmodell

Zur Korrektur des geschätzten Systemzustands werden die Beschleunigungssensorwerte herangezogen. Entsprechend Abschnitt 5.3.2 (siehe Gleichung 5.49 auf Seite 60) folgt das Messmodell in der Form:

$$\Delta\vec{z}(t) = \mathbf{H}(t)\Delta\vec{x}(t) + \vec{v}(t). \tag{9.16}$$

Die Beschleunigungswerte können wie in Gleichung 4.11 auf Seite 39 und den dort gezeigten Vereinfachungen beschrieben werden durch:

$$\vec{a}^n = \mathbf{C}_b^n \vec{a}^b + \vec{g}_l^n. \tag{9.17}$$

Wird zudem der stationäre Fall des Systems betrachtet, folgt aus Gleichung 9.17:

$$-\vec{g}_l^n = \mathbf{C}_b^n \vec{a}^b. \tag{9.18}$$

Darüber hinaus gilt mittels des Sensormodells aus Gleichung 7.2 auf Seite 73, unter den Annahmen, dass die *Misalignment*-Matrix $\mathbf{M} = \mathbf{I}$ und der Bias $\vec{b}_a = 0$ ist, der Zusammenhang:

$$\tilde{\vec{a}}^b = \mathbf{I}\vec{a}^b + \vec{n}_a \ . \tag{9.19}$$

Wird Gleichung 9.18 nach $\vec{a}^b$ umgestellt und in Gleichung 9.19 eingesetzt, dann ergibt sich:

$$\tilde{\vec{a}}^b = -\mathbf{C}_b^{n,T} \vec{g}_l^n + \vec{n}_a \ . \tag{9.20}$$

Dabei ist die reale Richtungskosinusmatrix $\mathbf{C}_b^{n,T}$ nicht bekannt. Nach [172] gilt jedoch zwischen der realen und der geschätzten Richtungskosinusmatrix $\hat{\mathbf{C}}_b^{n,T}$ der Zusammenhang:

$$\mathbf{C}_b^{n,T} = \hat{\mathbf{C}}_b^{n,T} (\mathbf{I} - \boldsymbol{\Psi}) . \tag{9.21}$$

Darin stellt $\boldsymbol{\Psi}$ die schiefsymmetrische Matrix der Winkelfehler $\vec{\Psi}$ dar:

$$\boldsymbol{\Psi} = \begin{pmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{pmatrix} . \tag{9.22}$$

Wird weiter Gleichung 9.21 in Gleichung 9.20 eingesetzt, dann folgt:

$$\tilde{\vec{a}}^b = -\hat{\mathbf{C}}_b^{n,T} (\mathbf{I} - \boldsymbol{\Psi}) \vec{g}_l^n + \vec{n}_a \tag{9.23}$$

$$\tilde{\vec{a}}^b + \hat{\mathbf{C}}_b^{n,T} \vec{g}_l^n = \hat{\mathbf{C}}_b^{n,T} \boldsymbol{\Psi} \vec{g}_l^n + \vec{n}_a \ . \tag{9.24}$$

Wird schließlich noch $\boldsymbol{\Psi}$ ausmultipliziert

$$\tilde{\vec{a}}^b + \hat{\mathbf{C}}_b^{n,T} \vec{g}_l^n = \hat{\mathbf{C}}_b^{n,T} \begin{pmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ g_0 \end{pmatrix} + \vec{n}_a \tag{9.25}$$

$$= \hat{\mathbf{C}}_b^{n,T} \begin{pmatrix} \beta g_0 \\ -\alpha g_0 \\ 0 \end{pmatrix} + \vec{n}_a \ , \tag{9.26}$$

zeigt sich, dass anhand der Beschleunigungssensormessung die *Roll*- und *Pitch*-Winkel gestützt werden können, nicht aber der *Yaw*-Winkel. Die vollständige Messgleichung lautet somit:

$$\underbrace{\tilde{\vec{a}}^b + \hat{\mathbf{C}}_b^{n,T}\vec{g}_l^n}_{\Delta\vec{z}(t)} = \hat{\mathbf{C}}_b^{n,T}\begin{pmatrix} 0 & g_0 & 0 \\ -g_0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} + \vec{n}_a \,. \tag{9.27}$$

Korrektur der Lage

Aufgrund der Tatsache, dass das gezeigte *Error State Space* Kalman-Filter lediglich die Winkelfehler und die Drehratenbiasfehler schätzt, muss anschließend die Lagelösung korrigiert werden. Da die Lage des MAVs als Lagequaternion ($\vec{q}^n$) dargestellt ist, wird, in Anlehnung an [115] und [172], ein Korrekturquaternion ($\vec{q}_c^n$) aus den Winkelfehlern berechnet. Dazu werden zuerst die Winkelfehler einem Orientierungsvektor zugewiesen:

$$\vec{\sigma}_c = \Delta\hat{\vec{\Psi}} \,. \tag{9.28}$$

Das Korrekturquaternion errechnet sich wie folgt:

$$\vec{q}_c = \begin{pmatrix} \cos\frac{\sigma_c}{2} \\ \frac{\sigma_{c,x}}{\sigma_c}\sin\frac{\sigma_c}{2} \\ \frac{\sigma_{c,y}}{\sigma_c}\sin\frac{\sigma_c}{2} \\ \frac{\sigma_{c,z}}{\sigma_c}\sin\frac{\sigma_c}{2} \end{pmatrix} \tag{9.29}$$

mit $\sigma_c = |\vec{\sigma}_c|$. Aus Effizienzgründen können die trigonometrischen Terme, gemäß der in [172] gezeigten Reihenentwicklung, vereinfacht werden zu:

$$\vec{q}_c \approx \begin{pmatrix} 1 - \frac{1}{8}\sigma_c^2 \\ \sigma_{c,x}\left(\frac{1}{2} - \frac{1}{48}\sigma_c^2\right) \\ \sigma_{c,y}\left(\frac{1}{2} - \frac{1}{48}\sigma_c^2\right) \\ \sigma_{c,z}\left(\frac{1}{2} - \frac{1}{48}\sigma_c^2\right) \end{pmatrix} . \tag{9.30}$$

Somit kann das Lagequaternion korrigiert werden[3]:

$$\hat{\vec{q}}^{n,+} = \vec{q}_c \bullet \hat{\vec{q}}^{n,-} . \tag{9.31}$$

Wie in [115] gezeigt wird, ist daraufhin das korrigierte Lagequaternion zu normieren, da dies durch Rundungsfehler nicht mehr gewährleistet ist:

$$\hat{\vec{q}}^{n,+} = \frac{\hat{\vec{q}}^{n,+}}{|\hat{\vec{q}}^{n,+}|} \,. \tag{9.32}$$

3 Die Quaternionenmultiplikation wird durch den Operator $\bullet$ angegeben (vergleiche [172]).

Danach werden noch die geschätzten Biasfehler der Drehratensensoren verarbeitet durch:

$$\hat{\vec{b}}_{\omega}^{+} = \hat{\vec{b}}_{\omega}^{-} + \Delta\vec{b}_{\omega} \; . \tag{9.33}$$

Zuletzt muss der Zustandsvektor zu 0 gesetzt werden, da die geschätzten Fehler zuvor in die Lagelösung eingeflossen sind.

$$\Delta\hat{\vec{x}}^{+} = 0 \; . \tag{9.34}$$

Implementierung

Mit diesen Filtergleichungen ist das Lage-Kalman-Filter in der Simulation mit Hilfe der Eigen-Bibliothek (siehe [46]) entwickelt und evaluiert worden. Um bei der späteren Ausführung auf dem realen Autopiloten jedoch eine hohe Aktualisierungsrate von 1 kHz realisieren zu können, liegt ein besonderer Fokus auf der effizienten Umsetzung und Optimierung hinsichtlich der Berechnungszeit. Dazu wird im ersten Schritt angenommen, dass das Messrauschen $\vec{n}_{\omega}$ in allen drei Achsen identisch ist. Somit kann die Einflussmatrix $\mathbf{G}(t)$ zu einer Einheitsmatrix vereinfacht werden. Zudem ist gezeigt worden, dass das dargestellte Kalman-Filter die *Roll*- und *Pitch*-Winkelfehler schätzen kann, nicht jedoch die *Yaw*-Winkelfehler. Somit kann der Zustandsvektor von sechs auf vier Einträge reduziert werden zu:

$$\Delta\vec{x}_k = \begin{pmatrix} \alpha \\ \beta \\ \Delta b_{\omega,x} \\ \Delta b_{\omega,y} \end{pmatrix} . \tag{9.35}$$

Des Weiteren ist das reduzierte Zustandsraummodell zu diskretisieren (siehe Gleichung 5.51 auf Seite 60). Somit ist

$$\mathbf{\Phi}_{k-1} = \begin{pmatrix} 1 & 0 & -\hat{C}^{n}_{b,0/0}\Delta t & -\hat{C}^{n}_{b,0/1}\Delta t \\ 0 & 1 & -\hat{C}^{n}_{b,1/0}\Delta t & -\hat{C}^{n}_{b,1/1}\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{9.36}$$

und

$$\mathbf{G}_{k-1} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Delta t \; . \tag{9.37}$$

Wie erkennbar ist, sind eine Vielzahl von Einträgen der Matrizen mit 0 besetzt, was bei der effizienten Implementierung des Filters ebenso

ausgenutzt wird. So sind diese unnötigen Multiplikationen bei der Optimierung identifiziert worden, wodurch nur diejenigen Operationen ausgeführt werden, welche notwendig sind. Weiter wird die Symmetrie der Kovarianzmatrix $\mathbf{P}_k$ genutzt, womit sich der Rechenaufwand zusätzlich erheblich reduzieren lässt. Zuletzt ist bei der Auswahl des Mikrocontrollers darauf geachtet worden, dass dieser eine eigene Gleitkommaeinheit (*Floating Point Unit* (FPU)) besitzt.

Zum Vergleich ist die Implementierung mittels der Eigen-Bibliothek sowie die optimierte Version auf dem verwendeten Mikrocontroller, einem *STM32F407* (siehe [154]), ausgeführt worden. Zur Validierung wurden 10 000 Filterdurchläufe berechnet und daraufhin die Mittelwerte bestimmt. Dabei benötigt die Implementierung mittels der Eigen-Bibliothek und den üblichen Matrixoperationen im Schnitt 6.00 ms pro Filterdurchlauf. Die optimierte Version, die lediglich die benötigten Multiplikationen sowie Additionen vornimmt, kann hingegen in nur 200 µs prozessiert werden. Somit ist eine Lageberechnung mit einer Updaterate von 5 kHz auf dem gewählten Mikrocontroller möglich.

Umsetzung und Diskussion

Die Umsetzung, Parametrierung und Evaluierung des Lage-Kalman-Filters fand vollständig in der Simulation mit der modellierten Sensorik statt. Dabei entscheidet die Parametrierung des Kalman-Filters direkt über die Filtergüte. Bei diesem Lage-Kalman-Filter muss zudem besonders berücksichtigt werden, dass die Drehratensensoren durch die Messungen der Beschleunigungssensoren gestützt werden. Jedoch wird hierbei von einem quasistationären Flugzustand ausgegangen, was bedeutet, dass außer der Erdbeschleunigung g_0 keine zusätzlichen Beschleunigungen auf das MAV wirken. Im Schwebezustand (*Hover*) ist diese Bedingung hinreichend erfüllt. Jedoch ist in den Beschleunigungsphasen des MAVs diese Bedingung nicht erfüllt, was zur Folge hat, dass es zu einer fehlerhaften Schätzung der Fluglage sowie des Drehratensensorbias kommt. In der Simulation konnte jedoch nachgewiesen werden, dass diese Beschleunigungsphasen nur kurzfristig wirken und der Einfluss durch eine günstige Parameterwahl des Prozessrauschens ($\mathbf{Q}(t)$) minimiert werden kann. Hierzu sind exemplarisch in Abbildung 85 zwei Filterergebnisse mit unterschiedlich gewähltem Prozessrauschen (in *grün* und in *blau*) den idealen *Ground-Truth*-Daten der Simulation (in *magenta*) gegenübergestellt. Darin ist deutlich zu sehen, dass ein ungünstig gewähltes Prozessrauschen (Lage-Kalman-Filter 2 (L-KF 2) in *grün* dargestellt) zu deutlichen Fehlern in den geschätzten Lagefehlern führt und damit eine falsche Fluglage ermittelt wird. Der mittlere Fehler (RMSE) liegt hier bei 1.38° für den *Roll*-Winkel und 1.66° für den *Pitch*-Winkel. Die selben Winkel hingegen schätzt das Filter mit den in der Simulation

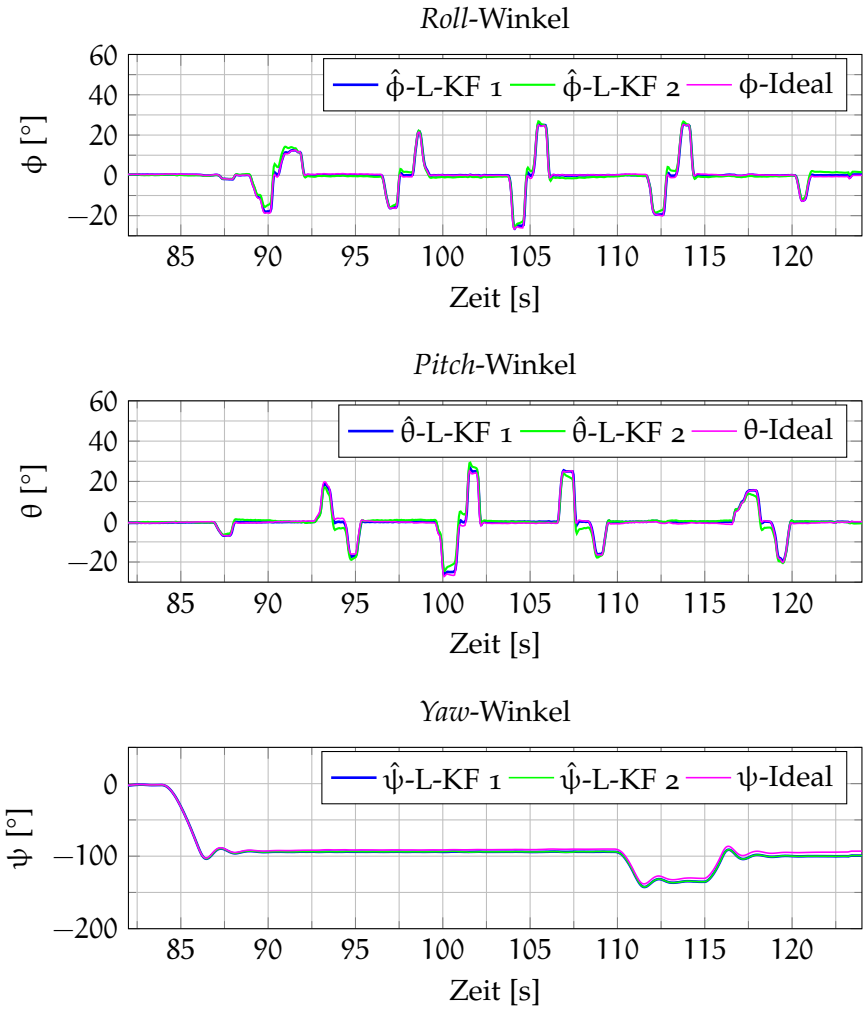


Abbildung 85: Fluglageschätzung des Lage-Kalman-Filters

optimierten Parametern (L-KF 1) deutlich besser. Der mittlere Fehler ist um Faktor zwei geringer und liegt lediglich bei 0.61° für den *Roll*-Winkel und 0.77° für den *Pitch*-Winkel. Analog dazu ist eine fehlerhafte Schätzung des Bias der Drehratensensorachsen zu beobachten (siehe Abbildung 86 auf Seite 180). Modelliert wurde der Bias anhand der realen Sensorik für die x-Achse mit 0.81°/s, für die y-Achse mit −0.19°/s sowie für die z-Achse mit −0.12°/s. Auch hier ist der Fehler durch ein ungünstig gewähltes Prozessrauschen sehr deutlich erkennbar. Der mittlere Fehler (RMSE) für den Bias liegt beim L-KF 2 bei 0.3°/s für die x-Achse und bei 0.33°/s für die y-Achse. Nach der Optimierung verringert sich der Fehler um eine Zehnerpotenz und liegt beim L-KF 1 bei 0.034°/s für die x-Achse und bei 0.035°/s für die y-Achse. Wie schon in der Filterherleitung diskutiert wurde, kann der Bias der z-Achse des Drehratensensors nicht gestützt werden. Die Schätzung ist daher 0°/s. Dieser Bias wirkt sich somit direkt in einem aufsummierenden *Yaw*-Winkelfehler aus. Bei dieser Betrachtung ist jedoch eines besonders zu beachten: Bei dem dargestellten Flug

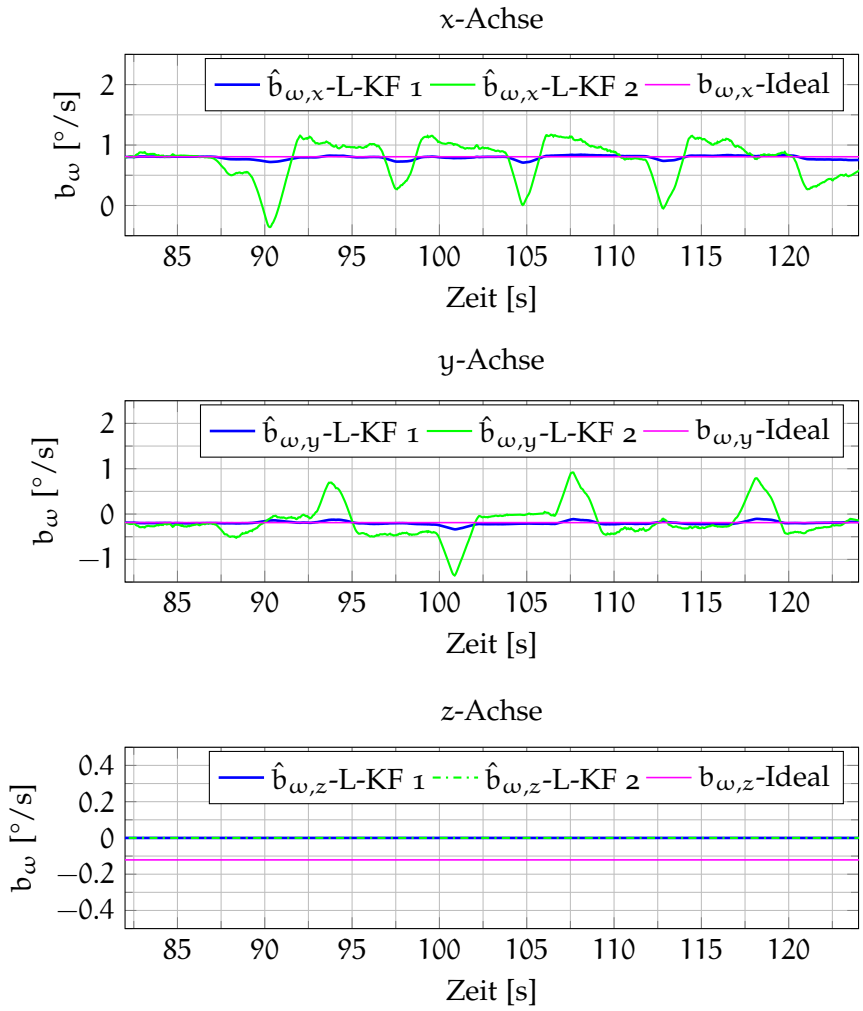


Abbildung 86: Biasschätzung des Lage-Kalman-Filters

würden beide Kalman-Filter ein akzeptables Ergebnis für die Stabilisierung der Flugplattform liefern, da die Phasen der einwirkenden, trajektorienbedingten Beschleunigung ausreichend kurz sind. Ohne *Ground-Truth*-Daten wäre der Fehler nur sehr schwer identifizierbar.

Daraus ableitend wurde ein weiterer Test in der Simulation durchgeführt, bei dem die trajektorienbedingte Beschleunigung über einen längeren Zeitraum auf das System wirkte. Dies wurde realisiert, indem mehrere (fünf) Kreise hintereinander geflogen worden sind. Somit wirkte über den gesamten Zeitraum die Zentripetalkraft auf die Flugplattform. In Abbildung 87 sind hierzu die geschätzten Lagewinkel dargestellt. Es ist erkennbar, dass bei beiden Filtern die Lagebestimmung fehlerhaft ist. Die Situation eskaliert jedoch ab Sekunde 61, in der die Flugplattform wieder in die Nulllage zurückkehren soll. Beim ungünstig parametrierten L-KF 2 ist der Biasfehler zu diesem Zeitpunkt bis auf 21°/s angestiegen, so dass innerhalb von 6 Sekunden der *Roll*-Winkelfehler auf $\approx 65°$ steigt. Das System wäre an dieser Stelle unkontrollierbar, da der maximale Winkel zum Gegensteuern

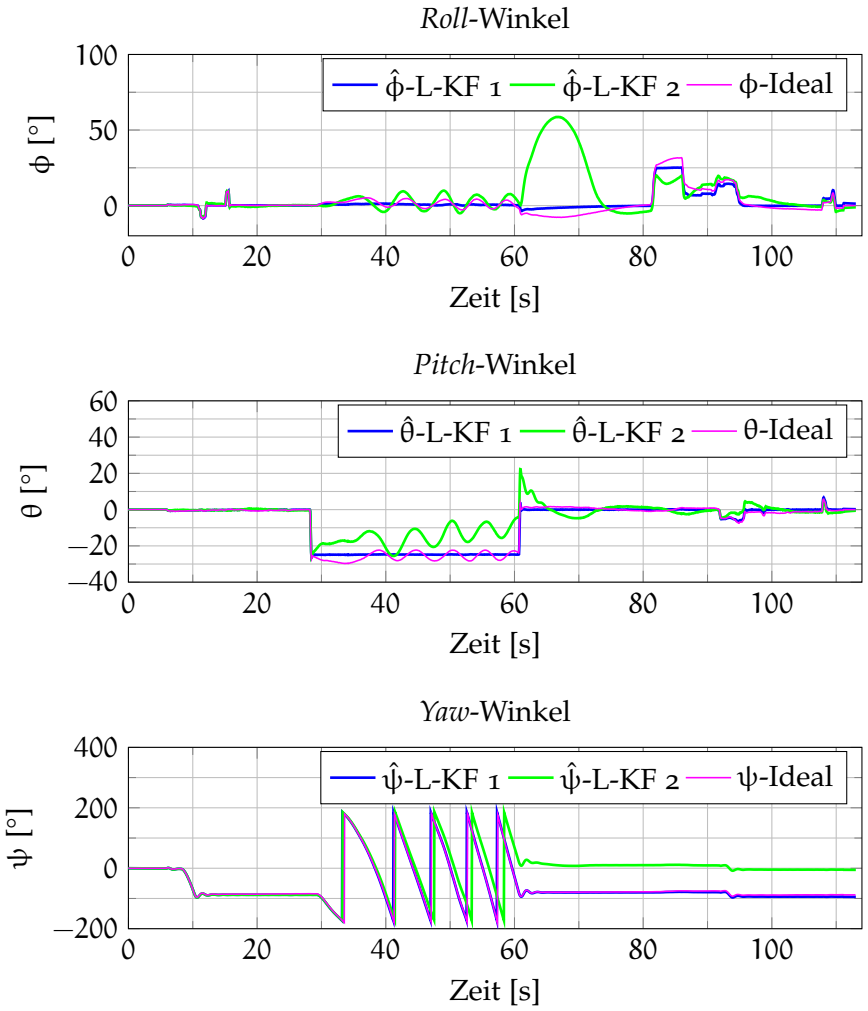


Abbildung 87: Einfluss der trajektorienbedingten Beschleunigung auf das Lage-Kalman-Filter

auf $\pm 25°$ begrenzt ist. Bei dem optimierten Filter steigt der maximale Winkelfehler zwar auch an, liegt aber nur bei $\approx 7°$. Das sorgt ebenfalls für ein Wegdriften der Flugplattform, jedoch bleibt sie weiterhin sicher kontrollierbar. Nicht zuletzt wird durch so deutliche Winkelfehler in der *Roll*- oder *Pitch*-Achse auch der bestimmte *Yaw*-Winkel negativ beeinflusst. Zur Lagestabilisierung der Flugplattform wurde bei dem Flug der Lagewinkel aus dem L-KF 1 genutzt. Dies erklärt, warum der Winkel in der *Pitch*-Achse in Abbildung 87 bei $-25°$ liegt, da das der kommandierte Winkel war. Hierbei muss angemerkt werden, dass der hier gezeigte Fehler ohne die Simulation wahrscheinlich erst in einem Freifeldtest mit dem realen Prototypen erkannt worden wäre und dies voraussichtlich zu einem Absturz der Flugplattform geführt hätte. Zudem ist ohne die Verfügbarkeit von *Ground-Truth*-Daten ein solcher Fehler deutlich schwerer nachvollziehbar und nur mit hohem Zeit- und Materialaufwand und den damit verbundenen Kosten reproduzierbar.

9.3.3 *Höhen-Kalman-Filter*

Neben der Fluglagebestimmung ist die Bestimmung der Flughöhe und der Vertikalgeschwindigkeit ein weiteres elementares Glied zur eigenständigen Stabilisierung des MAVs . Dazu wird ein virtueller Beschleunigungssensor eingeführt, der die Vertikalbeschleunigung im Navigationskoordinatensystem (a_d^n) des MAVs ermittelt. Durch den bekannten Zusammenhang

$$\vec{a}_k^n = \mathbf{C}_{b,k}^n \vec{a}_k^b + \vec{g}_l^n \tag{9.38}$$

wird die zum Zeitpunkt t_k im körperfesten Koordinatensystem bestimmte Beschleunigung ($\vec{a}_k^b$) zuerst ins Navigationskoordinatensystem transformiert. Dabei entspricht $a_{d,k}^n$ genau

$$a_{d,k}^n = \begin{pmatrix} 0, & 0, & 1 \end{pmatrix} \begin{pmatrix} a_x^n \\ a_y^n \\ a_z^n \end{pmatrix}_k . \tag{9.39}$$

Die gesuchten Parameter ergeben sich entsprechend des *Strapdown*-Algorithmus (siehe Abschnitt 4.1.2) für die Vertikalgeschwindigkeit zu:

$$v_{d,k}^n = v_{d,k-1}^n + a_{d,k}^n \cdot \Delta t \tag{9.40}$$

und analog für die Flughöhe zu:

$$p_{d,k}^n = p_{d,k-1}^n + v_{d,k-1}^n \cdot \Delta t + \frac{1}{2} \cdot a_{d,k}^n \cdot \Delta t^2. \tag{9.41}$$

Aufgrund dessen, dass die reale Richtungskosinusmatrix ($\mathbf{C}_{b,k}^n$) nicht bekannt ist, muss die durch das Lage-Kalman-Filter geschätzte Richtungskosinusmatrix ($\hat{\mathbf{C}}_{b,k}^n$) verarbeitet werden. Somit ist die berechnete Vertikalbeschleunigung ($a_{d,k}^n$) direkt abhängig von den Winkelfehlern des Lage-Kalman-Filters. Um diesen Fehlern Rechnung zu tragen, wird die virtuelle Messung der vertikalen Beschleunigung ($\tilde{a}_{d,k}^n$) modelliert als:

$$\tilde{a}_{d,k}^n = a_{d,k}^n + b_{a,d} . \tag{9.42}$$

Jedoch kann der als konstant angenommene Bias $b_{a,d}$ nicht ohne eine Stützgröße geschätzt werden. Damit wird klar, dass sich Fehler in der bestimmten vertikalen Beschleunigung direkt auf die Flughöhe auswirken und über die Zeit aufsummieren.

Auf Basis dieser Erkentnisse wird ein erweitertes Kalman-Filter (EKF siehe Abschitt 5.3.2) mit dem linearen, zeitdiskreten Zustandsraummodell

$$\vec{x}_k = \mathbf{\Phi}_{k-1} \vec{x}_{k-1} + \mathbf{B}_{k-1} \vec{u}_{k-1} + \mathbf{G}_{k-1} \vec{w}_{k-1} \tag{9.43}$$

eingesetzt, welches die barometrische Höhenbestimmung, wie sie in Abschnitt 4.3 beschrieben ist, als Stützinformation verwendet.

Das Zustandsraummodell

Wie bereits in [115] gezeigt, wird der Zustandsvektor entsprechend

$$\vec{x}_k = \begin{pmatrix} p_d^n \\ v_d^n \\ b_{a,d} \end{pmatrix}_k \tag{9.44}$$

definiert. Werden nun die zuvor hergeleiteten Zusammenhänge aus Gleichung 9.40, 9.41 und 9.42 zum Zustandsraummodell formuliert, ergibt sich:

$$\begin{aligned} \begin{pmatrix} p_d^n \\ v_d^n \\ b_{a,d} \end{pmatrix}_k = & \begin{pmatrix} 1 & \Delta t & -\frac{1}{2}\Delta t^2 \\ 0 & 1 & -\Delta t \\ 0 & 0 & 1 \end{pmatrix}_{k-1} \begin{pmatrix} p_d^n \\ v_d^n \\ b_{a,d} \end{pmatrix}_{k-1} \\ & + \begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \\ 0 \end{pmatrix}_{k-1} \underbrace{\left(a_d^n + b_{a,d}\right)_k}_{\tilde{a}_d^n} + \begin{pmatrix} n_{p_d} \\ n_{v_d} \\ 0 \end{pmatrix}_{k-1} . \end{aligned} \tag{9.45}$$

Dieses muss im Folgenden durch Verarbeitung der gemessenen barometrischen Höhe gestützt werden.

Das Messmodell

Zur Verarbeitung des Messwertes wird das in Abschnitt 7.1.3 hergeleitete Modell des Baro-Altimeters gewählt. Hierbei wird der Messfehler, bedingt durch den sensorinternen Erwärmungsprozess, der bei jedem Sensor verschieden ausgeprägt ist, in diesem Messmodell nicht berücksichtigt. Deshalb folgt:

$$\tilde{\eta}^n = \eta^n + b_\eta + n_\eta \, . \tag{9.46}$$

Da der Bias durch dieses Kalman-Filter nicht zu bestimmen ist, wird dieser zu $b_\eta = 0$ gesetzt[4]. Somit ergibt sich:

$$\tilde{\eta}_k^n = \eta_k^n + n_{\eta,k} . \tag{9.47}$$

Hier sei nochmal auf die Herleitung der barometrischen Höhe in Abschnitt 4.3 verwiesen, aus der hervorgeht, dass η^n die Höhe relativ zum Startpunkt in Richtung *unten* (NED) bereitstellt. Für das vollständige Messmodell gilt daher:

$$\underbrace{\tilde{\eta}_k^n}_{z_k} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \end{pmatrix}}_{H_k} \underbrace{\begin{pmatrix} p_d^n \\ v_d^n \\ b_{a,d} \end{pmatrix}_k}_{\vec{x}_k} + \underbrace{n_{\eta,k}}_{\nu_k} . \tag{9.48}$$

4 Zur Biasbestimmung bzw. zur Bestimmung der absoluten Höhe ist ein Sensorsystem zur absoluten Positionsbestimmung, wie etwa ein GNSS-System, notwendig.

Implementierung

Die Implementierung erfolgt entsprechend der Herleitung des EKFs in Abschitt 5.3.2 mit den Gleichungen 5.67 bis 5.72 auf Seite 61. Dabei werden, analog zum Lage-Kalman-Filter, die Filtergleichungen optimiert und unnötige Rechenoperationen eliminiert. Wie beim Lage-Kalman-Filter werden auch hier Symmetriebedingungen ausgenutzt. Hier ist zu beachten, dass bei dieser Sensorfusion die Prädiktion mit den Daten des virtuellen Beschleunigungssensors mit der vollen Updaterate der IMU von 1000 Hz durchgeführt wird, die Korrektur erfolgt jedoch lediglich mit der Updaterate des Baro-Altimeters mit 120 Hz. Daher werden die Berechnungszeiten unterteilt ermittelt. Der Prädiktionsschritt benötigt hier 14 µs und der Korrekturschritt 3 µs. Ein vollständiges Filterupdate kann somit in 17 µs durchgeführt werden.

Umsetzung und Diskussion

Analog zum Lage-Kalman-Filter wird das zuvor vorgestellte Höhen-Kalman-Filter (H-KF) in der Simulation umgesetzt und optimiert. Dabei ist der schon aus dem Lage-Kalman-Filter bekannte Fall mit anhaltender, trajektorienbedingter Beschleunigung besonders von Interesse. Aus diesem Grund werden in der Simulation zwei Höhen-Kalman-Filter parallel implementiert. Beide haben exakt gleiche Filterparame-

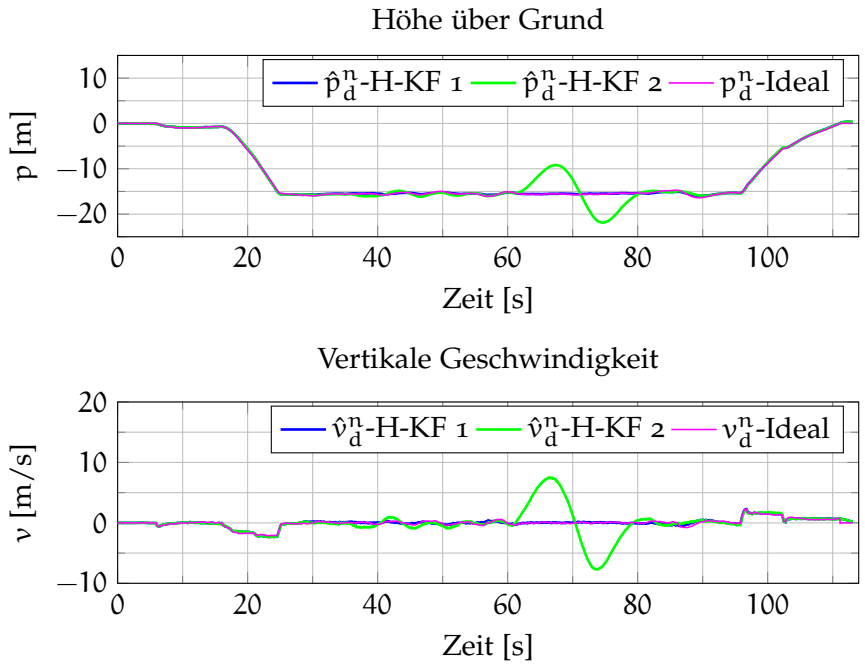


Abbildung 88: Einfluss der trajektorienbedingten Beschleunigung auf das Höhen-Kalman-Filter.

ter. Der einzige Unterschied ist, dass das H-KF 1 die ins Navigationskoordinatensystem transformierte Beschleunigung aus dem L-KF 1 nutzt und das H-KF 2 selbige jedoch aus dem L-KF 2. Somit gehen die Lagefehler der Lage-Kalman-Filter durch den fehlerhaft bestimmten Beschleunigungsvektor $\vec{a}^n$ direkt in das Höhen-Kalman-Filter ein. Das Ergebnis ist in Abbildung 88 dargestellt. So ist zu sehen, dass bis zu Sekunde 28 beide Filter die tatsächliche Flughöhe annähernd optimal schätzen. Dabei hat die kurzzeitige, trajektorienbedingte Beschleunigung durch das Steigen von Sekunde 17 – 25 nur einen unwesentlichen Einfluss auf die Filterergebnisse. Allerdings erkennt man den zunehmenden Fehler während der Kreisflüge ab Sekunde 30. Ab Sekunde 61 maximiert sich der Schätzfehler beim H-KF 2 analog zum L-KF 2. Der Höhenfehler beträgt hier $\pm 6\,\mathrm{m}$. Dies ist anhand der zuvor gezeigten Herleitung des Höhen-Kalman-Filters nur konsequent. Das Höhen-Kalman-Filter 1, welches den genaueren Schätzwert für die Beschleunigung aus dem optimierten L-KF 1 erhält, kann auch in dieser Phase die Höhe sowie die vertikale Geschwindigkeit deutlich besser schätzen. Ein unkontrollierbares Sinken bzw. Steigen kann verhindert werden. Der maximale Fehler für die geschätzte Flughöhe liegt lediglich bei $0.38\,\mathrm{m}$. So konnte durch die Simulation wiederholt gezeigt werden, dass trotz einer vermeintlich bestmöglichen Wahl der Filterparameter sowie einer hinreichend genauen barometrischen Höhenmessung zur Stützung, das Filterergebnis trotzdem durch die direkte Abhängigkeit zum Lage-Kalman-Filter kritisch sein kann. Die hier entwickelte Echtzeit-Simulation bietet dabei ein optimales Validierungstool, um derartige Einflüsse und Abhängigkeiten zu bestimmen. Dabei liegt ein weiterer Vorteil in der Möglichkeit, mehrere Filter parallel zu berechnen und die optimale Konfiguration zu bestimmen.

Abschließend sei noch festzuhalten, dass durch den sehr hohen Optimierungsgrad der implementierten Kalman-Filter ein vollständiges Update der Fluglage sowie der Höhe in $217\,\mu\mathrm{s}$ berechnet werden kann. Somit wäre eine theoretische Aktualisierungsrate von $4.6\,\mathrm{kHz}$ möglich. Die limitierenden Faktoren liegen hierbei in der möglichen Bandbreite des Sensorkommunikationsbusses und in der maximal möglichen sensorinternen Aktualisierungsrate.

9.4 HARDWAREENTWICKLUNG

Nachdem die gesamte Software des Autopiloten, beispielsweise die Datenvorfilterung, die Sensorfusion und die Systemregelung, innerhalb der Simulation entwickelt und evaluiert worden ist, soll diese jetzt auf ein reales MAV überführt werden. Ebenso wie für die Software gilt auch hier die Anforderung der MAV-Plattformunabhängigkeit. Aus diesem Grund wurde die Hardware in unterschiedliche Baugruppen (*Abstraktions-Level*) unterteilt. Die Hardware des Autopiloten

wird dabei hochintegriert und dennoch systemunabhängig geplant und entwickelt.

9.4.1 *Der Autopilot*

Um zukünftig eine unkomplizierte und effektive Integration in verschiedene MAV-Plattformen zu gewährleisten, werden die wichtigsten Sensoren und Funktionen auf einem Board, der *Flight Management Unit* (FMU), zusammengefasst (siehe Abbildung 89). Um auch dem Anspruch eines kostengünstigen Entwurfs gerecht zu werden, kommt die zuvor beschriebene und ausgewählte *Low-Cost Consumer Grade*-Sensorik zum Einsatz. So vereint die FMU eine IMU (*MPU9150*), zwei jeweils 3-achsige Magnetometer (*HMC58883L* und *MPU9150*), ein Barometer (*MS5611-01BA03*) und einen 12 bit Temperatursensor (*TMP102*). Weiterhin bietet sie ein 128 k *EEPROM* und einen microSD-Karten-Slot für Kalibrierungsdaten bzw. Datenlogging. Wie bereits vorgestellt, wird als Mikrocontroller der *STM32F407* von ST-Microelectronics bestückt. Dieser 32 bit *ARM-CortexM4* Mikrocontroller mit 168 MHz Taktfrequenz und integrierter *Floating Point Unit* (FPU) erlaubt die Berechnung des Lage- und Höhen-Kalman-Filters in weniger als 217 µs, wie bereits in Abschnitt 9.3 erläutert. Das gesamte *Automatic Flight Control System*, inklusive der Sensorvorverarbeitung, der optimierten Kalman-Filter sowie der Regler zur Fluglage und Höhenstabilisierung, wird mit einer Updaterate von 1000 Hz ausgeführt. Eine Aufstellung der weiteren Leistungsdaten ist in Tabelle 8 dargestellt. Damit dieser Autopilot auf unterschiedlichsten Flugplattformen eingesetzt werden kann, wird eine stapelbare Architektur verfolgt. Dazu werden sämtliche Schnittstellen, beispielsweise fünf UARTs, jeweils zwei I^2C- und SPI-Busse, ein CAN-Bus und zusätzlich sechs bzw. acht PWM-Kanäle, auf zwei jeweils 50-poligen Steckerleisten herausgeführt. Somit können unkomplizierte, aber anwendungsorientierte Mainboards entwickelt werden, welche auf die speziellen Bedürfnisse des MAVs bzw. dessen Applikation ausgerichtet sind.

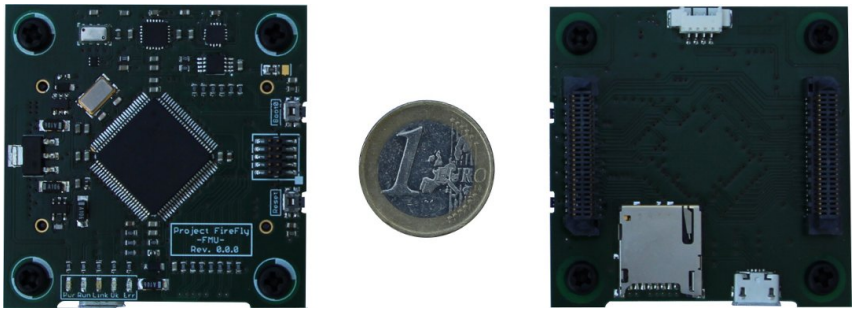


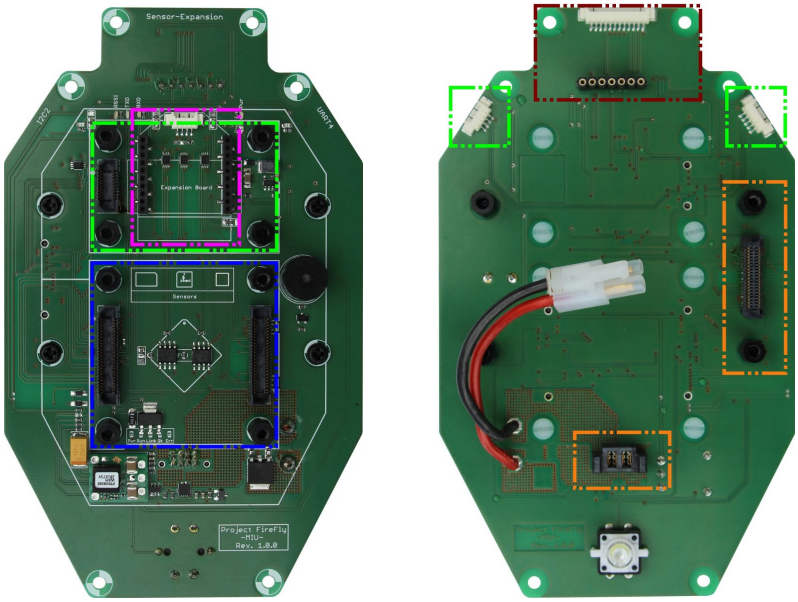
Abbildung 89: *Flight Management Unit* (FMU), Vorder- und Rückansicht

Tabelle 8: FMU-Leistungsdaten

Baugruppe	Bezeichnung
Prozessor	*STM32F407VGT6*
IMU	*MPU9150*
Magnetometer	*HMC5883L*
Barometer	*MS5611-01BA03*
EEPROM	*24FC128*
Schnittstellen	5×UART, 2×I²C, 2×SPI, 1×CAN, 16×GPIO, 6(8)×PWM, 5×ADC, 1×USB
Statusanzeige	5×LED
Datenschreiber	1×microSD-Karte (SDIO)
	Allgemein
Größe und Gewicht	50 mm × 50 mm, 12.8 g, 4-lagige PCB
Leistungsaufnahme	72 mA @ 5 V (0.36 W)

9.4.2 *Das Mainboard*

Das Mainboard (*Main Interface Unit* (MIU)) für den Flugplattformprototypen *FireBird-X4* ist darauf ausgelegt worden, eine Vielzahl an Erweiterungsmöglichkeiten für zukünftige Entwicklungen zu bieten. Prinzipiell ist das Konzept vergleichbar mit einem Mainboard bei einem *Desktop-PC*, bei dem unterschiedliche Stecksockel für vielfältige Erweiterungskarten vorgesehen sind. In Abbildung 90 auf Seite 188 ist die Ober- und Unterseite der MIU dargestellt. Auf der Oberseite ist neben dem eigentlichen Autopiloten (*blauer* Bereich) ebenfalls ein Steckplatz für ein *XBee*-Datenfunkmodul (siehe [41]) (*magenta* Bereich) zur MAV-Steuerung und für Telemetriedaten vorgesehen. Außerdem ist ein definierter Steckplatz für Erweiterungskarten integriert (*grüner* Bereich). Hier wird beispielsweise das in Abbildung 91b auf Seite 189 gezeigte GNSS-Modul eingesteckt. Zusätzlich befindet sich auf dieser Seite das Energiemanagement der Plattform. Neben der Ein-/Ausschaltelektronik und den Sensoren zur Spannungs- sowie Stromerfassung bzw. -überwachung werden hier die benötigten Betriebsspannungen erzeugt. Auf der Mainboardunterseite sind weitere Erweiterungssteckplätze angeordnet. So können im vorderen Bereich (*roter* Bereich) zusätzliche Sensoren integriert werden, beispielsweise der *TeraRanger One* (siehe [157]) – ein *Time of Flight* (ToF) basierter Entfernungssensor zur Stützung der Flughöhe über Grund. Außerdem sind ein weiterer dedizierter I²C-Bus sowie ein zusätzlicher Hochgeschwindigkeits-UART an den vorderen Eckpunkten vorgesehen (*grüner* Bereich), um zukünftig einen leistungsstarken *Sin-*



(a) Vorderansicht (b) Rückansicht

Abbildung 90: Die *Main Interface Unit* (MIU) des *FireBird-X4*

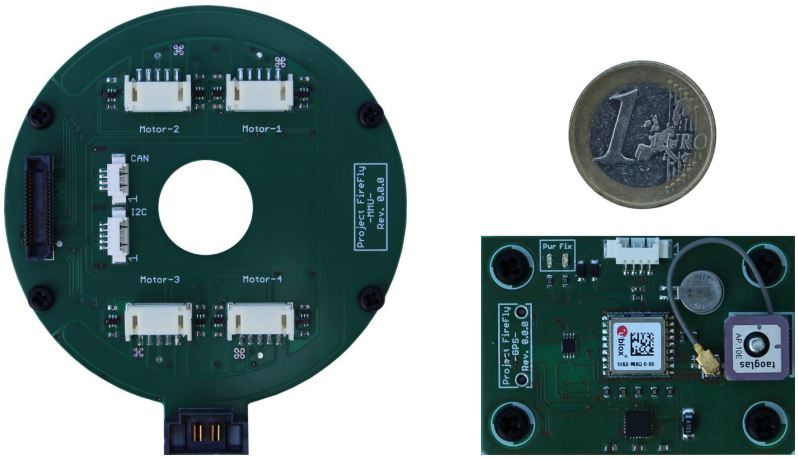
gle Board Computer (SBC), wie zum Beispiel das NVIDIA *Jetson TX1*-Modul (siehe [127]), direkt mit dem Autopiloten zu verbinden. Nicht zuletzt wird an den *orange* gekennzeichneten Steckplätzen das Modul zur Motorkommunikation eingehängt. Die Form und Größe des gesamten Mainboards ist direkt von der *Outdoor Hull* der *AR.Drone 2.0* (siehe [134]) abgeleitet, damit alle Komponenten und Module vor äußeren Einflüssen geschützt sind.

9.4.3 *Das Motorinterfaceboard*

Um auch in Zukunft bei der Wahl der Antriebe flexibel bleiben zu können, ist die Kommunikationshardware ebenfalls modular ausgelegt. Beispielsweise ist die *Motor Management Unit* (MMU) zur Kommunikation mit den Antrieben der *AR.Drone* in Abbildung 91a dargestellt. Darüber hinaus werden an der Steckverbindung vom Mainbaord zur MMU zusätzliche Kommunikationsschnittstellen, wie UART, I^2C-, SPI- oder ein CAN-Bus, zur Verfügung gestellt. Auch können die aus dem Modellbaubereich bekannten Motorendstufen (ebenfalls als *Electronic Speed Controler* (ESC) bezeichnet) über bis zu acht separate PWM-Kanäle angesteuert werden.

9.4.4 *Das GNSS-Modul*

Ein letztes Modul, das an dieser Stelle beschrieben werden soll, ist das entwickelte GNSS-Modul. Dieses umfasst einen *Ublox MAX-M8Q*



(a) *Motor Management Unit* (MMU)

(b) *Global Navigation Satellite System* (GNSS)-Modul

Abbildung 91: Erweiterungsboards: (a) MMU und (b) GNSS-Modul

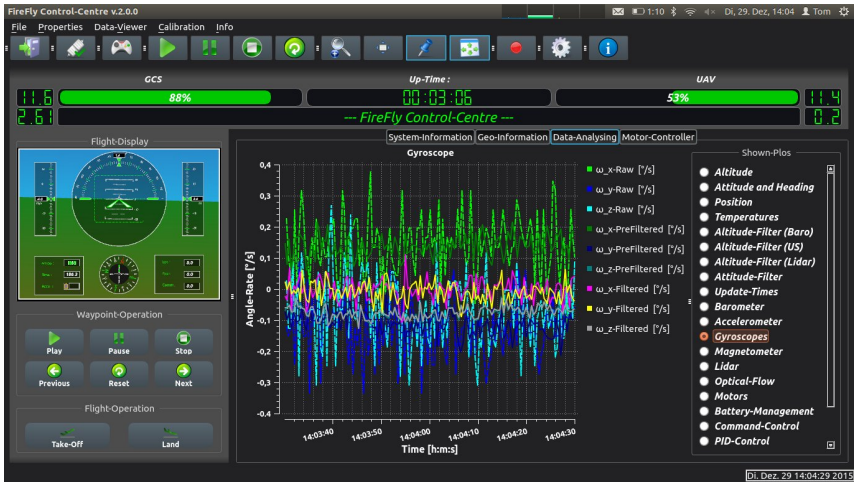
Empfänger (siehe [165]), welcher gleichzeitig bis zu drei unterschiedliche Satellitendienste empfangen und auswerten kann. Dazu gehören neben dem bekannten GPS das GLONASS, das *BeiDou*- und das noch im Aufbau befindliche *Galileo*-System. Dabei bietet das Modul Updateraten von bis zu 18 Hz. Zusätzlich ist auf dem Board eine weitere IMU und ein 3-achsiges Magnetometer (siehe [153]) integriert, was zukünftig als Redundanz zur Sensorik des Autopiloten prozessiert werden soll. Integriert ist die gesamte Sensorik auf der Bauform des definierten Erweiterungsmoduls mit der Größe 50 mm × 35 mm, so dass es direkt auf das Mainboard (MIU) gesteckt und verschraubt werden kann. Zusätzliche Verbindungskabel sind so nicht erforderlich. Das Gewicht, inklusive Antenne, beträgt lediglich 10 g.

9.5 DIE BODENKONTROLLSTATION

Neben den zuvor vorgestellten Elementen, wie der *embedded* Software des Autopiloten, der Hardware und der Simulation, soll in diesem Abschnitt der vierte und letzte Part des gesamten MAV-*Frameworks* vorgestellt werden. Dies ist die Software der Bodenkontrollstation (*engl. Ground Control Station* (GCS)), welche die Schnittstelle zwischen Operator (Pilot) bzw. Entwickler und dem MAV bildet. Dabei ist es für die GCS-Software unerheblich, ob diese mit dem realen oder dem simulierten MAV kommuniziert, da die Datenstrukturen und das Systemverhalten identisch sind. Neben der Darstellung globaler Systeminformationen, z. B. dem Software-Revisionsstand, der Art der Flugplattform, dem Energiemanagement, wie Strom- und Spannungsdarstellung, sowie der Systemstatusinformationen (siehe Abbildung 92a), kann der Operator über diese Software das MAV auch direkt steuern. Zu diesem Zweck ist eine Schnittstelle implementiert, die jeden

Standard-Spielecontroller zur Fernsteuerung der Flugplattform umwandelt. Aufgrund der Stabilisierung der Flugplattform in allen drei Lagewinkeln und in der Höhe, ist die Steuerung sehr intuitiv gestaltet. Mit dem linken Joystick kontrolliert der Pilot die Steig- bzw. Sinkrate sowie die Drehgeschwindigkeit in der *Yaw*-Achse. Über den rechten Joystick wird die *Roll*- bzw. *Pitch*-Winkelverstellung vorgegeben. Lässt der Pilot beide Sticks los, richtet sich die Flugplattform in Nulllage aus und stabilisiert Höhe und Lagewinkel. Außerdem kann durch Darstellung der Telemetriedaten des MAVs in Echtzeit (siehe Abbildung 92b)[5] das Verhalten und die Funktion unterschiedlichster Subsysteme analysiert sowie zur Postprozessierung in Logdateien gespeichert werden. Zusätzlich können *Ground-Truth*-Daten den Sensor-

(a)



(b)

Abbildung 92: Bodenkontrollstationssoftware: (a) Allgemeine Systeminformationen und (b) Echtzeitdatenplot

5 Zur Darstellung des Datenplots wird eine angepasste Version der freien Bibliothek *Qwt* (siehe [142]) verwendet.

und Filterdaten überlagert werden, wenn diese, wie beispielsweise in der Simulation, zur Verfügung stehen. In der Entwicklungsphase können ebenso verschiedene Steuerungs- oder Regelungsfunktionen, wie z. B. die der Motoren, direkt aus der Software über das Motorkontrollinterface angesprochen werden (siehe Abbildung 93a). Zukünftig kann über das Karteninterface[6] (siehe Abbildung 93b) eine Wegpunktplanung erstellt werden. Derzeit dient dieses zur Darstellung der MAV-Position, die vom GNSS-Modul berechnet wird. Nicht zuletzt umfasst die GCS ein zusätzliches Interface, um die Prüfstände zur Bestimmung der Modellparameter, beispielsweise von Antriebsschubkraft und Drehmoment, aufzunehmen. Somit deckt diese Software sämtliche Entwicklungsschritte ab.

(a)

(b)

Abbildung 93: Bodenkontrollstationssoftware: (a) Motorkontrollinterface sowie (b) Karteninterface

6 Das Karteninterface basiert auf einer angepassten Version des Karteninterfaces der freien GCS-Software des OpenPilot-Projektes (siehe [131]).

Wie bei der Software des Autopiloten gilt auch hier die Maßgabe der Plattformunabhängigkeit, weshalb die gesamte Software in C++ und mittels der Qt-Bibliothek programmiert ist. Derzeit ist die GCS-Software sowohl unter Windows und Linux (Ubuntu) als auch auf verschiedenen Prozessorarchitekturen wie x86, x64 und ARM® getestet.

9.6 ERGEBNISSE

Im letzten Schritt soll die reale Funktionsfähigkeit des gesamten MAV-*Frameworks* verifiziert werden. Hierzu werden drei exemplarische Szenarien entwickelt und vorgestellt. Im ersten Verifikationsschritt wird die bereits in Teilen in Abschnitt 9.2.3 vorgestellte Demonstratorplattform gefertigt und mit allen vorgestellten Hard- und Softwarekomponenten umgesetzt. Anschließend soll die angestrebte Plattformunabhängigkeit untersucht werden. Hierzu wird größtenteils eine kommerziell erhältliche Flugplattform genutzt, jedoch ist der integrierte Autopilot durch das hier gezeigte FMS ersetzt worden. Zuletzt werden die vielfältigen Möglichkeiten dieses MAV-*Frameworks* anhand eines neuartigen Algorithmus vorgestellt, welcher derartigen Flugsystemen das Starten aus extrem dynamischen Situationen heraus (Abwurf mit mehr als dem achtfachen der Erdbeschleunigung) erlaubt.

9.6.1 *FireBird-X4*

Um die Fähigkeiten dieses neuartigen Autopiloten zu überprüfen, ist eine neue, kleine und vielseitige Flugplattform aufgebaut worden. Der Preisrahmen wurde hierbei bewusst niedrig gehalten, um gerade für wissenschaftliche Anwendungen eine attraktive und vor allem kostengünstige Basis zu schaffen. Aus diesen Gründen nutzt

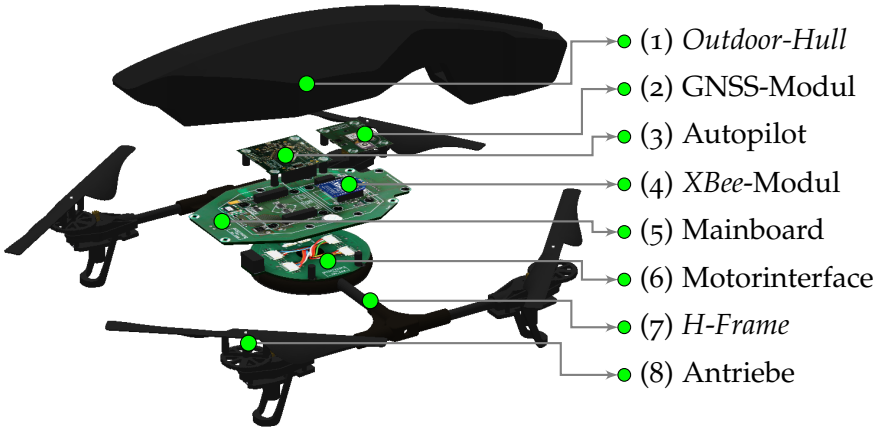


Abbildung 94: Explosionsdarstellung des *FireBird-X4*

(a) Simulierter FireBird-X4

(b) Realer FireBird-X4

Abbildung 95: Flugplattform *FireBird-X4* – simuliert zu real

der in Abbildung 94 dargestellte *FireBird-X4* die Motoren, Rotoren und Motortreiber (siehe Abbildung 94 (8)) der *AR.Drone*, die über das entwickelte Motorinterfaceboard (MMU) (6) integriert werden. Zur flexibleren Auslegung der Plattform wird jedoch ein eigenes *Frame* aus Leichtbaukomponenten in *H*-Form entwickelt (7). Der eigentliche Autopilot (FMU) (3) wird über das vorgestellte Mainboard (MIU) (5) in die Flugplattform integriert, welches weitere Funktionen übernimmt, zum Beispiel das Batterie-Management, die Energieversorgung oder den Funk-Datenlink, in diesem Fall ein *XBee*-Modul (4). Zusätzliche Erweiterungen, beispielsweise das für den *FireBird-X4* entwickelte GNSS-Modul (2), können über den Expansions-Slot integriert werden. Zum Schutz aller Komponenten wird ebenfalls auf die *AR.Drone* zurückgegriffen und deren *Outdoor-Hull* (1) genutzt.

Nachdem alle Algorithmen zur Sensordatenfilterung und -fusion und die Parameter zur Systemregelung mit dem virtuellen MAV (siehe Abbildung 95a) innerhalb der Simulation entworfen, evaluiert und optimiert worden sind, ist der gesamte Softwarekern des Autopiloten ohne jedwede Veränderung auf das reale MAV (siehe Abbildung 95b) portiert worden. Dabei sind vom ersten realen Flug an alle bisherigen Testflüge (> 50) erfolgreich und ohne einen Systemausfall absolviert worden.

9.6.2 *MikroKopter MK QuadroXL*

Eine weitere Flugplattform, auf der das hier vorgestellte Autopilotensystem integriert wurde, ist der kommerziell verfügbare MikroKopter *MK QuadroXL*. Im Gegensatz zu den bisher vorgestellten Plattformmodellierungen sind in diesem Fall jedoch die Modellierungsparameter der Struktur und der Antriebe nicht vorher präzise analysiert und bestimmt worden. Die Schubkraft der Antriebe ist beispielsweise anhand der Herstellerangaben modelliert worden. Das Antriebsdrehmoment sowie die Modellierungsparameter des Trägheitsmoments und des Massenschwerpunkts sind auf Basis der bisher erarbeiteten Erfahrungen abgeschätzt worden. Mit diesem lediglich näherungs-

(a) Simulierter MK QuadroXL (b) Realer MK QuadroXL

Abbildung 96: Flugplattform MikroKopter *MK QuadroXL* – simuliert zu real

weise bestimmten Flugplattformmodell sind die Parameter der Filteralgorithmen und Regler optimiert worden. Ziel war es, qualitativ zu evaluieren, ob die somit ermittelten Parameter für den Autopiloten ausreichend genau für ein automatisches Stabilisieren der realen Flugplattform sind. Bei der Integration des neuen FMS zeichnet sich das vorgestellte MAV-*Framework* besonders dadurch aus, dass sämtliche Kernkomponenten, beispielsweise die zur Sensorverarbeitung, -filterung und Datenfusion sowie die grundsätzlichen Regelungsalgorithmen, vollständig weiter genutzt werden können. Analog zum *FireBird-X4* sind daraufhin innerhalb der Simulation die Parameter in den Bereichen Datenfilterung und Systemregelung angepasst und optimiert worden. Folglich konnte die Software auch in diesem Fall nach der simulationsbasierten Integration auf der virtuellen Plattform (siehe Abbildung 96a) ohne weitere Änderungen auf die reale Flugplattform (siehe Abbildung 96b) portiert werden. Um ebenfalls bei der Hardware eine möglichst einfache Integration des neuen Autopiloten in das bestehende System zu erreichen, werden sämtliche Originalteile, bis auf den Autopiloten, beibehalten. Dementsprechend ist für die original verbauten Motortreiber (*BL-Crtl V2.0*) des *MK QuadroXL* ein für diesen optimierter Softwaretreiber im *Board-Level* implementiert worden. Das Integrationskonzept der Hardware (siehe Abbildung 97) folgt demselben Ansatz wie bei dem zuvor vorgestellten *FireBird-X4*. Somit wird hier die Autopilotenhardware (FMU) (siehe Abbildung 97 (2)) über ein speziell für diese Flugplattform optimiertes Mainboard (MIU-MK4) (5) in das originale *Frame* (8) integriert. Das Mainboard übernimmt auch hier die Kommunikation mit verschiedensten Peripherien, wie z. B. dem Funk-Datenlink (3). Kabelverbindungen zu den originalen Motortreibern (ESCs) (6) können, wie auch die originale Haube (1) und die originalen Antriebe (7), weiter genutzt werden. Bei diesem Konzept besteht ebenfalls über die vorgesehenen verschiedenen Expansions-Interfaces (4) die Anbindung weiterer (externer) Peripherien, wie beispielsweise die Integration eines GNSS-Empfängers oder einer zusätzlichen Rechenplattform. Bei den darauf folgenden realen Flugtests war besonders interessant, dass

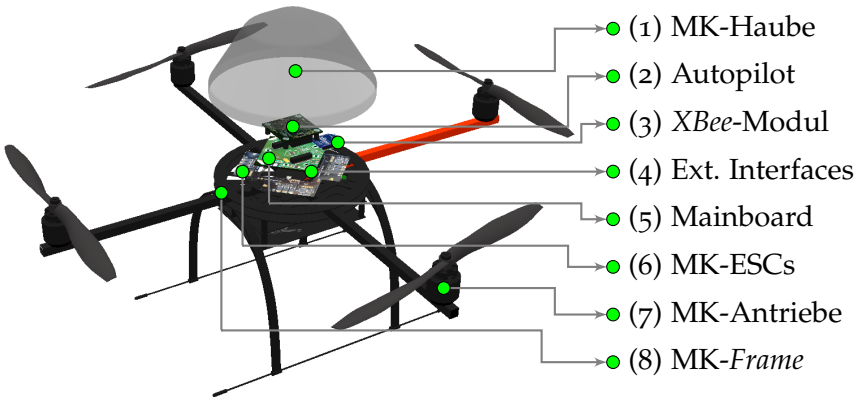


Abbildung 97: Explosionsdarstellung des *MK QuadroXL* mit dem *FireFly*-Autopiloten

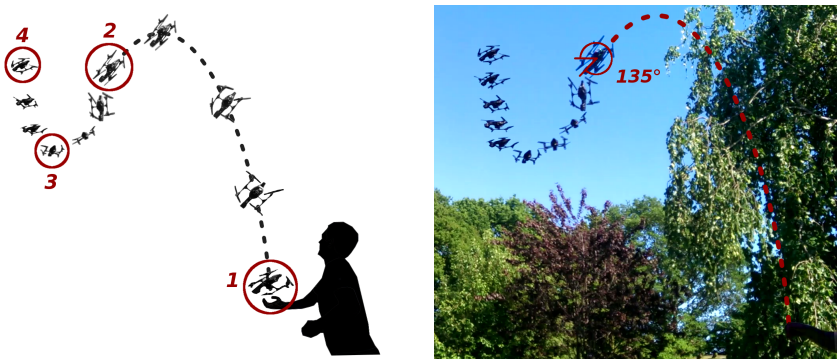
trotz des nur näherungsweise bekannten Flugplattformmodells ein sicheres Manövrieren der Flugplattform vom ersten Flug an möglich war. So war es möglich, die gesuchten Parameter iterativ zu optimieren, wodurch gleichermaßen die Genauigkeit der simulierten Flugplattform als auch das reale Flugverhalten verbessert wurde. Dies zeigt, dass auch bei nur näherungsweise bekannten Modellen diese simulationsbasierte Entwicklung bzw. Integration zielführend ist, da weiterhin verschiedene Fehler- sowie Einflussmöglichkeiten zuvor am virtuellen MAV eruiert werden können. Gleichwohl sind die Simulationsergebnisse mit deutlich präziseren Modellierungsparametern, wie am Beispiel des *AR100B* oder des *FireBird-X4* gezeigt wurde, wesentlich genauer, so dass sämtliche Optimierungen vollständig am virtuellen System stattfinden können.

9.6.3 *Automatisches Starten*

Auf Basis der erfolgreichen Integration des neuen MAV-*Frameworks* auf beiden zuvor dargestellten Flugplattformen wird im Folgenden ein weiterer Aspekt des Funktionsumfangs dieses *Toolkits* vorgestellt. Hierbei handelt es sich um die Möglichkeit, neue Algorithmen zu entwickeln, bei denen das Systemverhalten sowie die Regler- bzw. Filterparameter im Vorfeld nicht eindeutig bzw. vollständig bekannt sind und deshalb ein simulationsbasierter Entwicklungsansatz zielführend ist. Als Beispiel hierfür wird ein neues Verfahren zum automatischen Starten aus hochdynamischen Situationen vorgestellt[7].

Funktionen zum automatischen Starten (*Auto Take-Off*) gehören bei aktuellen VTOL-MAVs zur Standardfunktionalität. Dabei wird in der Regel vorausgesetzt, dass sich die Flugplattform stehend (unbewegt) am Boden befindet, daraufhin eingeschaltet wird und erst nach der

7 Zum gleichen Zeitpunkt der Veröffentlichung dieses Verfahrens (siehe [99]) ist ein ähnlicher Ansatz in [52] vorgestellt worden.



(a) (b)

Abbildung 98: Automatisches Starten aus hochdynamischen Situationen: Konzept (a) und realer Test (b)

Initialisierung von Sensoren, Filtern etc. das Startkommando erfolgt. Im Anschluss steigt das MAV automatisch auf eine vorgegebene Höhe und verweilt dort, bis weitere Kommandos erfolgen. Dies schränkt jedoch die Möglichkeiten dieser Funktion erheblich ein, beispielsweise bei Offshore-Anwendungen, bei denen das Starten und Landen von einem (kleinen) Schiff vorausgesetzt wird. Dabei schlagen traditionelle Sensorinitialisierungen fehl, weil sich die Startplattform, z. B. durch den natürlichen Wellengang, ständig in Bewegung befindet. Ein ähnliches Szenario stellt sich bei kooperativen Missionen dar, bei denen von fahrenden Bodenfahrzeugen gestartet werden soll. Aus diesem Grund ist im Rahmen dieser Arbeit ein Verfahren entwickelt worden, welches es ermöglicht, das System auch aus unterschiedlichsten, hochdynamischen Situationen automatisch zu starten und in einen stabilen Flugzustand zu überführen. In Abbildung 98a ist das Konzept dieses Startvorgangs in vier diskreten Sequenzen dargestellt. Zu Beginn wird die Flugplattform (*FireBird-X4*) in der Hand des Piloten eingeschaltet und kann daraufhin verzögerungsfrei zum Starten in die Luft geworfen werden (Sequenz 1). Hierbei spielt es keine Rolle, in welcher Ausrichtung das System eingeschaltet bzw. abgeworfen wird. Nach Durchquerung des Scheitelpunktes der – in diesem Fall – parabelförmigen Wurfbahn erfolgt der zweite Schritt der Startsequenz, bei dem das System das Startkommando erhält. Dieses kann manuell vom Piloten ausgelöst werden oder zukünftig auch automatisch durch eine *Freefall-Detection* erfolgen (Sequenz 2). Ab diesem Zeitpunkt beginnt das MAV mit der automatischen Stabilisierung. Hierzu ist der Lageregler für *Roll-* und *Pitch*-Winkel höher priorisiert als der Höhen- und *Yaw*-Regler. Hat sich das Flugsystem in der Lage stabilisiert (Sequenz 3), folgt die Stabilisierung und Korrektur der Höhe und der Ausrichtung, bis die zum Startzeitpunkt vorgegebene Höhe, Lage und Ausrichtung vollständig erreicht ist (Sequenz 4). Ab diesem Zeitpunkt kann das Flug-

(a) t = 0.0 s (b) t = 2.0 s (c) t = 2.4 s (d) t = 2.8 s

(e) t = 3.0 s (f) t = 3.1 s (g) t = 3.4 s (h) t = 4.5 s

Abbildung 99: Bildsequenzen des hochdynamischen automatischen Starts

system mit den gewohnten Funktionalitäten vom Piloten gesteuert werden. In Abbildung 98b ist ein realer Startvorgang dargestellt. Dabei kann dieser *Auto Take-Off*-Algorithmus das System auch problemlos aus unkonventionellen Fluglagen, beispielsweise kopfüber, vollständig automatisch stabilisieren. Eine Neuheit bei diesem Verfahren ist, dass dieser Autopilot weder eine vorherige Sensorkalibrierung noch eine Sensor- oder Filter-Initialisierung benötigt, da bis zu drei Kalman-Filter parallel mit einer Updaterate von 1000 Hz den Systemzustand prädizieren. Die Besonderheit an diesem Algorithmus ist neben der eigentlichen Funktionalität jedoch der Entwicklungsprozess, da das gesamte Verfahren in der Simulation des hier vorgestellten *FireFly* MAV-*Frameworks* erfolgt ist. Erst nach der virtuellen Evaluierung des Algorithmus wurde der erste Test unter realen Bedingungen durchgeführt. Nach den ersten Versuchsreihen, beispielhaft dargestellt in Abbildung 98b, sind innerhalb der Simulation die Kalman-Filter sowie die Regler- und Sensorparameter weiter optimiert worden, damit auch äußerst aggressive Starts mit sehr hohen Beschleunigungen optimal berechnet und vollautomatisch stabilisiert werden können. So ist beispielsweise der Messbereich jeder Sensorachse des Beschleunigungssensors von ±4 g auf ±8 g erhöht worden. In den Abbildungen 99a bis 99h ist dazu exemplarisch ein weiterer hochdynamischer Start dargestellt, bei dem das MAV auf über 10 m/s mit mehr als der achtfachen Erdbeschleunigung beschleunigt wird. Parallel zu den Bildsequenzen sind in Abbildung 100 auf Seite 198 die Sensor- bzw. Filterdaten für die Flughöhe über Grund, für die Vertikalgeschwindigkeit und der Beschleunigungsvektorbetrag aufgezeichnet. Außerdem zeigt Abbildung 101 auf Seite 199 die berechnete Orientierung des MAVs. Die diskreten Zeitpunkte der Bildsequenzen sind dabei durch *gepunktete* bzw. *gestrichelte* Linien in den Gra-

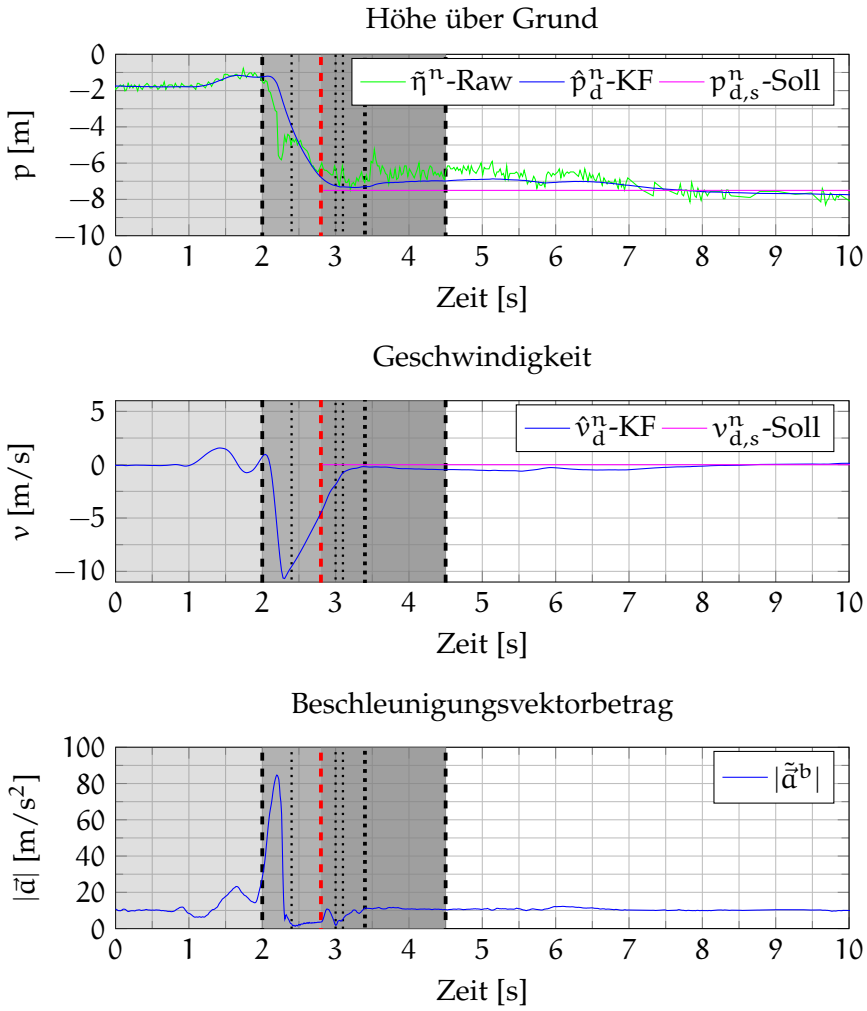


Abbildung 100: Sensor- und Kalman-Filterdaten der Flughöhenschätzung während des hochdynamischen automatischen Starts

phen gekennzeichnet. Abbildung 99a auf der vorhergehenden Seite zeigt zu Beginn die Aktivierung der Flugplattform. Daraufhin erfolgt in den Sequenzen 99b und 99c der Abwurf des MAVs, wobei dieses mit einem Beschleunigungsvektorbetrag von $84.7\,m/s^2$ (siehe Abbildung 100 *unten*) auf eine Vertikalgeschwindigkeit von $10.7\,m/s$ (siehe Abbildung 100 *Mitte*) beschleunigt wird. Dabei ist in den gemessenen Sensordaten des Barometers (siehe Abbildung 100 *oben* in *grün* dargestellt) besonders in der Beschleunigungsphase ein deutlicher Messfehler zum Zeitpunkt $t = 2.25\,s$ erkennbar. Jedoch kann durch das in Abschnitt 9.3.3 vorgestellte Fusions-Kalman-Filter aus Beschleunigung und barometrischer Höhe über die gesamte Zeit eine plausible Prädiktion der Flughöhe erreicht werden (siehe Abbildung 100 *oben* in *blau* dargestellt). Zum Zeitpunkt $t = 2.8\,s$ erfolgt das Kommando zum automatischen Starten bzw. Stabilisieren der Flugplattform (*rot-gestrichelte* senkrechte Linie). Wie in der Bildsequenz in Abbildung 99d ersichtlich, ist die Flugplattform zu diesem Zeitpunkt voll-

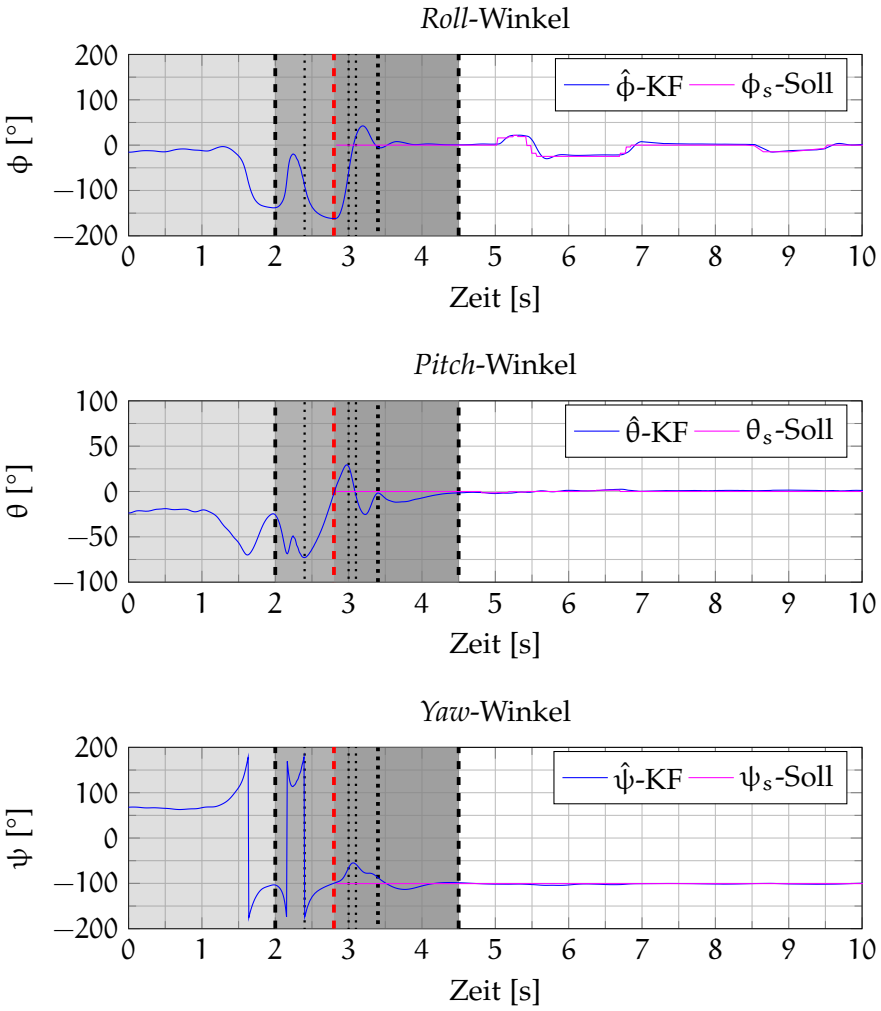


Abbildung 101: Orientierungsschätzung des Lage-Kalman-Filters während des hochdynamischen automatischen Starts

ständig kopfüber ausgerichtet. Der *Roll*-Winkel beträgt $\approx 162°$ (siehe Abbildung 101 *oben*). Ab diesem Zeitpunkt beginnt die vollautomatische Stabilisierung des Systems. Dazu sind in *magenta* die Vorgabewerte der Systemregelung dargestellt. Der Autopilot setzt die Flughöhe 0.8 m über der Starthöhe von $-6.7\,\mathrm{m}$ auf $p^n_{d,s} = -7.5\,\mathrm{m}$, um ebenfalls bei flachen Startsequenzen eine sichere Flughöhe zu garantieren. Die Steig- bzw. Sinkrate wird mit $v^n_{d,s} = 0\,\mathrm{m/s}$ und die Lagewinkel mit $\phi_s = \theta_s = 0°$ vorgegeben. Der *Yaw*-Winkel wird auf den Wert zum Startzeitpunkt von $\psi_s = -100°$ festgelegt. In den folgenden 600 ms bis zum Zeitpunkt $t = 3.4\,\mathrm{s}$ (*dick-gepunktete* senkrechte Linie) stabilisiert die Flugplattform zunächst die Fluglage für *Roll*- und *Pitch*-Winkel, wie in Abbildung 99e bis 99g erkennbar. Hierbei werden Drehraten von bis zu $900°/\mathrm{s}$ erreicht. Daraufhin erfolgt die Stabilisierung des *Yaw*-Winkels bis zum Zeitpunkt $t = 4.5\,\mathrm{s}$. Die Orientierung des MAVs ist nun vollständig stabilisiert. Die Abweichung der tatsächlichen Flughöhe zur Vorgabe beträgt an dieser Stelle ledig-

lich 0.5 m. Diese Differenz wird in den folgenden ≈ 4 s bewusst langsam ausgeregelt, um ein ruhiges und gleichmäßiges Flugverhalten zu erzeugen. Dabei ist ab dem Stabilisierungszeitpunkt bei $t = 4.5$ s das System vollständig pilotierbar, wie es im weiteren Verlauf, beispielsweise bei der *Roll*-Winkelvorgabe, ersichtlich ist. Hervorzuheben ist, dass alle Optimierungen, besonders in den Filtern als auch bei den Reglern, in vollem Umfang in der Simulation implementiert und evaluiert worden sind. Aufgrund dieser simulationsbasierten Herangehensweise sind alle durchgeführten Tests, bisher wurden mehr als 50 unterschiedliche dynamische Starts durchgeführt, erfolgreich verlaufen. Einen Absturz oder eine Zerstörung aufgrund von Software- bzw. Systemfehlern gab es nicht.

9.7 ZUSAMMENFASSUNG UND DISKUSSION

In diesem Kapitel ist eine neuartige Entwicklungsumgebung zur effizienten und plattformunabhängigen Arbeit mit MAVs vorgestellt worden. Dabei verbindet dieses *Framework* die herausragenden Vorteile der in Abschnitt 6 vorgestellten 3D-Echtzeit-Simulationsumgebung mit einem innovativen Autopiloten, dessen neuartige Softwarestruktur vollständig *Software in the Loop* (SiL) in die Simulation eingebunden ist. Somit ist eine einfache Integration bzw. Portierbarkeit dieses neuen Autopiloten auf unterschiedliche Plattformen einfach und effizient. Alle Parameter für die Filter und die Regler können in der virtuellen Umgebung bestimmt und optimiert werden. Durch die Verfügbarkeit von *Ground-Truth*-Daten und einer realistischen Flugumgebung können Fehler und Probleme frühzeitig in der Simulation diagnostiziert und eliminiert werden. Das Risiko, eine reale Plattform zu zerstören, verringert sich drastisch. Durch das vorgestellte Hardwarekonzept können die erzielten Ergebnisse aus der Simulation durch ein minimalistisches Mainboard auf reale Plattformen überführt werden. Zudem zeigt die neu entwickelte Hardware des Autopiloten gegenüber dem anfänglich genutzten kommerziellen Autopiloten deutliche Vorteile. So konnte die Auflösung bei den Drehratensensoren von 1.00°/s pro LSB um den Faktor 30 auf 0.03°/s pro LSB gesteigert werden. Ähnliches gilt für die Beschleunigungssensorik, bei der die Auflösung im ursprünglichen Autopiloten bei $0.12\,\mathrm{m/s^2}$ pro LSB lag und bei dem nun neu entwickelten um den Faktor 100 auf $0.0012\,\mathrm{m/s^2}$ pro LSB erhöht werden konnte[8]. Nicht zuletzt zeigt die Kombination aus dem neuen Softwarekonzept des Autopiloten und der direkten Integration in der 3D-Echtzeit-Simulationsumgebung einen erheblichen Mehrwert, das konnte beispielsweise anhand der Optimierung der Kalman-Filter gezeigt werden. In Summe führt dies zu einem äußerst verlässlichen MAV-Autopiloten mit optimal parametrierten Sensoren, Filtern und Reglern.

8 Entsprechend Tabelle 7 auf Seite 166 gilt: $0.12\,\frac{\mathrm{mg}}{\mathrm{LSB}} \approx 0.0012\,\frac{\mathrm{m/s^2}}{\mathrm{LSB}}$.

Zur Evaluation dieser neuen Entwicklungs- und Simulationsumgebung ist der hier vorgestellte Autopilot in zwei unterschiedliche Flugplattformen integriert worden. Erstere ist eine vollständig selbst entwickelte *Low-Cost*-Plattform, die klein und leicht und darüber hinaus auch für den *In-* sowie für den *Outdoor*-Bereich einsetzbar ist. Zudem bietet das Konzept des *FireBird-X4* größtmögliche Ausbaufähigkeiten an. Für die Simulation sind zuerst die Struktur, die Antriebe sowie die flugdynamischen Kenngrößen modelliert worden, um daraufhin die Software des Autopiloten zu entwickeln sowie die Parameter für die Filteralgorithmen und Regler zu optimieren. Erst nach erfolgreicher Softwareevaluierung und -optimierung wurde diese auf das realen Flugsystem portiert. Dabei zeigte die Flugplattform vom ersten realen Flug an exzellente Flugeigenschaften, so dass die Software und die Parameter des realen Flugsystems keine Modifikationen oder Änderungen zur simulierten Version benötigten. Die zweite Flugplattform, in die der hier entwickelte Autopilot integriert wurde, ist der kommerziell von MikroKopter vertriebene *MK QuadroXL*. Im Gegensatz zum *FireBird-X4* ist jedoch beim *MK QuadroXL* nur ein näherungsweise bestimmtes Flugplattformmodell in der Simulation genutzt worden. Trotzdem konnte auch in diesem Fall bewiesen werden, dass die simulationsbasierte Entwicklung bzw. Integration zielführend ist.

Darüber hinaus ist gezeigt worden, dass auch Algorithmen, bei denen das Systemverhalten sowie die Sensor- und Filterparameter im Vorfeld nicht eindeutig bzw. vollständig bekannt sind, präzise und effizient innerhalb der hier gezeigten Simulation zu entwickeln, zu evaluieren und zu optimieren sind. In diesem Kontext ist ein Verfahren zum automatischen Starten von MAVs aus hochdynamischen Situationen vorgestellt worden.

Neben der Entwicklung und Optimierung von vergleichbaren komplexen Algorithmen zur *Low-Level*-Steuerung dieser Flugplattformen eröffnet dieses *FireFly* MAV-*Framework* ebenfalls neue Möglichkeiten zur Umsetzung von *High-Level*-Funktionalitäten. Beispielsweise wäre die Entwicklung von Verfahren zur visuellen Odometrie, zum simultanen Lokalisieren und Kartieren (*engl. Simultaneous Localization and Mapping* (SLAM)) oder auch die Entwicklung von komplexen und hochdynamischen Bahnplanungsalgorithmen zur Hindernisvermeidung denkbar. So bildet ebenfalls hier die Kombination von MAV, Autopilot und Simulation einen entscheidenden Mehrwert, um zukünftig derart innovative Applikationen effizient und vor allem risikoarm zuerst in der virtuellen Realität zu entwickeln und zu optimieren, bevor sie auf der realen Flugplattform eingesetzt werden. Hierbei können zu jeder Zeit Abhängigkeiten zwischen *Low-Level-* und *High-Level*-Steuerung untersucht werden, da in der Simulation verschiedenste Einflussfaktoren variiert und so alle Parameter für das optimalen Ergebnis angepasst werden können.

Teil III

ZUSAMMENFASSUNG

10 ZUSAMMENFASSUNG UND AUSBLICK

Die Entwicklung von neuen und innovativen MAVs oder intelligenten Algorithmen zur automatischen Regelung dieser Systeme ist ein aktueller Forschungsschwerpunkt, obgleich sich schon verschiedenste Systeme und Plattformen am Markt befinden. Eine wiederkehrende Herausforderung ist die Integration in eigene Projekte. Eine Anpassung von Filter- und Reglerparametern auf die genutzte Plattform ist beliebig komplex und oft nur mittels „*Try and Error*“ möglich.

Aus diesem Grund ist in dieser Arbeit eine neuartige Simulationsumgebung erarbeitet worden, welche ein neues und maßgebendes Werkzeug zur Entwicklung und Analyse von Multirotor-MAVs bietet. Dabei liegt die herausragende Eigenschaft in der modellbasierten, realistischen Echtzeit-Simulation des gesamten Flugsystems. So können durch die originalgetreue Abbildung der Sensoren, der Antriebe und der flugdynamischen Kenngrößen nicht nur vorhandene Algorithmen unter realistischen Bedingungen analysiert werden, sondern ebenfalls unterschiedlichste äußere Einflussfaktoren, unter anderem Windlasten, in die Gesamtbetrachtung mit einbezogen werden. Im Gegensatz zu mathematischen Analysetools, beispielsweise *MATLAB & Simulink*, ist es in dieser neuartigen Simulation möglich, den vollständigen Quellcode des Autopiloten *Software in the Loop* (SiL) einzubinden und neben den prozessierten Daten der Sensoren, Filter und Regler ebenfalls ein realistisches visuelles Feedback des Fluggerätes in Echtzeit zu erhalten.

Hierzu ist zuerst ein kommerziell verfügbares Flugsystem vollständig modelliert und in die virtuelle Realität überführt worden. Mit diesem System wurden zwei unterschiedliche Flugszenarien zur Analyse der Simulationsergebnisse umgesetzt. Hierbei konnte in unterschiedlichen Flugsituationen nachgewiesen werden, dass die simulierte Flugplattform sehr präzise das Flugverhalten sowie die Messgrößen des realen MAVs nachbildet. Auch konnte gezeigt werden, dass die vorgestellte Simulation inklusive der erarbeiteten Modelle so präzise ist, dass sogar geringste Abweichungen von den Erwartungswerten detektiert werden können, wie es im Beispiel der Magnetometerdaten gezeigt worden ist. Das ermöglicht vollständig neue Ansätze in der Entwicklung von Multirotor-MAVs. Zudem können sogar komplexe Fehler, wie der gezeigte Kompilierungsfehler, auf der realen Flugplattform nachgewiesen werden, da mit dieser Simulation ein idealer Vergleichswert für alle internen Variablen und externen Parameter errechnet werden kann. Nur durch diese neue Möglichkeit der Softwareanalyse können nun derartig komplexe Fehler schnell, sicher

und effizient ausfindig gemacht und eliminiert werden. Nicht zuletzt konnte durch diese Simulationsumgebung und den damit verbundenen Entwicklungsmöglichkeiten der vorhandene Autopilot erweitert und optimiert werden, so dass innerhalb kürzester Entwicklungszeit insgesamt vier neue Flugsysteme fertiggestellt werden konnten. Ein wichtiger Entwicklungsschritt stellt zum damaligen Zeitpunkt der erfolgreiche Reglerentwurf für ein Hexakoptersystem mit einem maximalen Abfluggewicht von 12 kg und einem Durchmesser von 220 cm dar.

Trotz dieser guten Ergebnisse wurde die Entscheidung getroffen, den bisher genutzten, kommerziellen Autopiloten zu ersetzen. Dies hat vielfältige Gründe: Wie bei der Sensormodellierung der bisher genutzten IMU nachgewiesen wurde, haben die Drehraten- und Beschleunigungssensoren aufgrund einer suboptimalen Anbindung an den Autopiloten nur eine bedingte Auflösung, die mit dem neuen Design um mindestens den Faktor 30 gesteigert werden konnte. Zudem zeigte sich, dass das Rauschspektrum der alten IMU nicht einer Normalverteilung folgt, was hinsichtlich der Datenfusion in einem Kalman-Filter besonders betrachtet werden muss. Diese und weitere Nachteile hätten sich bei zukünftigen Arbeiten wiederkehrend negativ ausgewirkt. Mit der neuen Hardware konnten diese Unwegbarkeiten erfolgreich abgestellt werden.

Im Verlauf dieser Arbeit ist so ein vollständig neues *Framework* zur effizienten und plattformunabhängigen Arbeit mit MAVs entwickelt worden. Dieses MAV-*Framework* besteht explizit nicht allein aus dem neuen Autopiloten, sondern verbindet eine neuartige Softwarestruktur mit der zuvor entwickelten 3D-echtzeitfähigen Simulationsumgebung. Das neue Softwarekonzept für das *Flight Management System* ist optimal auf die Anforderungen der SiL-Struktur für die simulationsbasierte Entwicklung ausgelegt. Damit der Entwicklungsprozess jedoch auf Basis realistischer Daten und Modelle aufbaut, ist vorab eine Auswahl der zu nutzenden Sensorik und Aktorik getroffen worden, die anschließend modelliert worden sind. Somit ist die gesamte Autopilotensoftware innerhalb der Simulation anhand eines virtuellen Prototypen entworfen und evaluiert worden. Durch die vollständige Parametrierbarkeit aller Sensormodelle konnten dementsprechend Algorithmen zur Datenfilterung und -fusion entwickelt und genauestens untersucht werden, wie es beispielsweise bei den Kalman-Filtern für die Fluglage und die Höhe gezeigt worden ist. Die in der Simulation zur Verfügung stehenden zeitsynchronen *Ground-Truth*-Daten bilden ein optimales Werkzeug für solche Untersuchungen. Spezifische Abhängigkeiten können damit erkannt und die Genauigkeit der Algorithmen optimiert werden.

Der Umfang dieses MAV-*Frameworks* geht jedoch über die reine Softwareentwicklung deutlich hinaus. So war der Hardwareentwurf des Autopiloten für den plattformunabhängigen Einsatz ein wichtiger

Entwicklungsschritt. Neben der Plattformunabhängigkeit standen zudem die Forderungen nach einer möglichst geringen Abmessung und einem geringen Gewicht im Vordergrund. Darüber hinaus war ein insgesamt kostengünstiger Entwurf das Ziel. Mit einer Baugröße von 50 mm × 50 mm, einem Gewicht von 12.8 g und Gesamtkosten für den Autopilotprototypen von unter 100 € konnten auch diese Ziele realisiert werden. Um den entwickelten Autopiloten herum ist daraufhin eine komplette Flugplattform neu gestaltet worden. Der *FireBird-X4* ist dabei das Bindeglied zwischen dem virtuellen Entwurf und der realen Anwendbarkeit. Er vereint dabei eine für den *In-* und *Outdoor*-Bereich universell einsetzbare Fluglattform mit möglichst großen Erweiterungsoptionen für zukünftige Entwicklungen. Mit Gesamtkosten von etwa 700 € für diesen Entwicklungsprototypen konnte zudem der angestrebte Kostenrahmen von 1000 € erfolgreich eingehalten werden. Zur Demonstration der Plattformunabhängigkeit ist zudem der selbe Autopilot erfolgreich in ein weiteres kommerzielles MAV integriert worden, den MikroKopter *MK QuadroXL*.

Besonders in Szenarien, bei denen das Systemverhalten sowie die Sensor- und Filterparameter im Vorfeld nicht eindeutig bzw. vollständig bekannt sind, beispielsweise bei dem gezeigten automatischen Starten aus hochdynamischen Situationen, zeigte dieses neue MAV-*Framework* einen außergewöhnlichen Mehrwert. Durch die simulationsbasierte Entwicklung und Optimierung aller Softwarekomponenten sowie die Möglichkeit der Simulation derartig komplexer Situationen ist eine risikoarme Entwicklung mit derartigen VTOL-MAVs möglich. Der so entwickelte *Auto Take-Off*-Algorithmus kann dabei das System problemlos aus unkonventionellen Fluglagen, zum Beispiel kopfüber, vollständig automatisch stabilisieren, sogar wenn das MAV im Startvorgang auf über 10 m/s mit mehr als der achtfachen Erdgravitation beschleunigt wird. Nur durch die konsequente und vollständige Portierbarkeit des neuen FMS zwischen virtuellem und realem Flugsystem konnte ein Verlust oder die Zerstörung der Flugplattform durch Software- bzw. Systemfehler verhindert werden.

Ausblick

Die hier dargestellte Arbeit kann in vielfältiger Art und Weise weitergeführt werden. Allein die Simulation bietet Potenzial für zukünftige Arbeiten. So sei die weiterführende Untersuchung von Strömungsmodellen beispielhaft als ein interessanter Bereich genannt. Naheliegend sind auch Überlegungen für eine Implementierung einer differenzierten Strömungssimulation und die Untersuchung detaillierterer Windmodelle. Aber auch die Modellierung weiterer Flugplattformen wäre ein interessanter Arbeitsschwerpunkt. So müssen die Modelle zur Zeit noch mit großem zeitlichen Aufwand mit unterschiedlichen Messapparaturen erstellt werden. Die Umsetzung am Beispiel

des MikroKopter *MK QuadroXL* zeigt jedoch, dass auch mit rudimentären Modellen schon akzeptable Ergebnisse erzielt werden können. Zukünftig wäre vorstellbar, ein *Motion-Capture*-System und das reale MAV mit dessen virtuellem Pendant zu koppeln und so gleichzeitig das Modell und die Filter- und Reglerparameter iterativ zu verbessern. Ein hochinteressanter Ansatz wäre ein zukünftiges „Analyse durch Synthese"-Verfahren (siehe [95]), in dem die realen Daten des MAVs mit der Simulation gekoppelt werden, um daraus weitere, vielfältigere Informationen abzuleiten (siehe [98]). Mit den derzeitig erstellten Modellen der Flugplattform lassen sich schon jetzt grundlegende Informationen im realen Flug, wie beispielsweise die Windgeschwindigkeit und die Windrichtung, bestimmen.

Ein besonderer Fokus liegt jedoch in der Erweiterung für die simulationsbasierte Entwicklung von *Computer Vision*-Applikationen. Während der Fertigstellung dieser Arbeit ist von Microsoft Research eine Simulationsumgebung auf Basis der *Unreal Engine 4* (siehe [90]) veröffentlicht worden (siehe [147]), bei der der Fokus auf der photorealistischen Visualisierung der virtuellen Welt liegt. Im Unterschied zu dem in dieser Arbeit entwickelten Simulationsframework werden jedoch signifikante Vereinfachungen bei den Modellen gemacht und bisher nur zwei Autopilotsysteme – *Hardware in the Loop* (HiL) – unterstützt. Da die Simulation von Microsoft Research jedoch quelloffen (*Open Source*) ist, wäre die Integration des *FireFly* MAV-*Frameworks* mit dem neu entwickelten Autopiloten und der *FireBird-X4*-Flugplattform möglich. Dabei kann mit den in dieser Arbeit entwickelten detaillierten Modellen der realen Sensoren und Sensorsysteme sowie der flugplattformspezifischen Kenngrößen und Antriebe ein nahezu identisches, aerodynamisches Verhalten der Flugplattform erzeugt werden. Die damit einhergehende Verfügbarkeit realistischer Sensordaten in Kombination mit einer umfangreichen photorealistischen Visualisierung der virtuellen Welt würde somit neue, innovative Möglichkeiten bei der Entwicklung von *Computer Vision*- und *Machine Learning*-Applikationen für MAVs aufzeigen.

Dass die zur Verfügung stehende Technologie und die Applikationsbandbreite dieser fliegenden Agenten auch in Zukunft ein außergewöhnliches Potenzial bietet, zeigen gerade die aktuellsten Marktentwicklungen. So hat Intel Anfang des Jahres 2016 den bekannten MAV-Hersteller Ascending Technologies übernommen (siehe [168]). DJI bringt nahezu quartalsweise neue Flugsysteme auf den Markt, wie aktuell die *Mavic Pro* (siehe [42]), die mit zwei Stereokameras und einer hochperformanten *OnBoard*-Rechenkapazität ausgestattet ist. Dieses System kann schon heute in vorgegebenen Grenzen Kollisionen vermeiden und Objekte verfolgen.

Trotz dieser schnellen Marktentwicklung bzw. gerade wegen dieser ist die Weiterentwicklung und der Ausbau des hier vorgestellten MAV-*Frameworks* besonders sinnvoll, denn im Gegensatz zu den

auf dem Markt befindlichen Systemen ist dieses Entwicklungssystem vollständig auf die Forschung und Weiterentwicklung ausgelegt. Besonders durch die tiefe Kopplung des Autopiloten mit der Simulation schafft dieses *Framework* eine substantielle Basis für zukünftige Arbeiten. Vor allem durch die Hardwareunabhängigkeit ergeben sich neue Entwicklungsfelder. So wäre die Integration dieses Autopiloten auf das von Qualcomm entwickelte *Snapdragon Flight-Kit* (siehe [140]) ebenfalls ein interessanter Aspekt. Auch das von Intel vorgestellte *Aero-Kit* (siehe [79]), bestehend aus einem leistungsstarken Prozessorboard und einem *Computer Vision-Kit*, würde in Verbindung mit diesem MAV-*Framework* den Weg zu einer autonomeren Flugplattform ermöglichen. Nicht zuletzt kündigte Parrot zum Ende des Jahres 2016 das *S.L.A.M. dunk*-Entwicklungskit (siehe [133]) an, das eine Stereokamerakonfiguration mit einem leistungsstarken NVIDIA *Tegra K1* Prozessor vereint.

Kurzfristig ist die Integration des Autopiloten in die Robotikplattform ROS (siehe [145]) angestrebt, um so einen breiteren Zugang auf vielfältige Algorithmen zu gewährleisten und in Verbindung mit weiterer Umgebungssensorik Untersuchungen für die automatische Lokalisierung und Kartierung zu bewerkstelligen.

Mit der in dieser Arbeit vorgestellten ganzheitlichen Lösung, bestehend aus Software und Hardware, für die Forschung an MAVs ergibt sich eine beeindruckende Zahl an zukünftigen Anwendungs- und Entwicklungsmöglichkeiten. Der jetzt zur Verfügung stehende vollständige Entwicklungskreis aus simulationsbasierter Algorithmenentwicklung und einem verlässlichen und erprobten Autopiloten kann ab sofort mit neuester *OnBoard*-Rechenkapazität gekoppelt werden.

Teil IV

ANHANG

A

ANHANG: GNSS

A.1 GNSS/INS-KOPPLUNGSKONZEPTE

Aufgrund der komplementären Eigenschaften des *Global Navigation Satellite Systems* (GNSS) und des *Inertial Navigation Systems* (INS) ist eine Kopplung zielführend und in einer Vielzahl von Applikationen gängige Praxis. Dazu sind in der Literatur vier Integrationskonzepte zu finden, die sich zum einen in den genutzten Messgrößen und zum anderen im Integrationsumfang erheblich unterscheiden.

A.1.1 *Uncoupled*

Beim ungekoppelten System bestimmen das GNSS und das INS unabhängig voneinander ihre Position. Dabei wird lediglich im Navigationsrechner zwischen INS- und GNSS-Position gewechselt, wenn eine valide Lösung des GNSS zur Verfügung steht. Ist keine GNSS-Position verfügbar, wird die Position aus dem INS genutzt. Diesen Zusammenhang zeigt Abbildung 102. Prinzipiell gehört dieser Ansatz nicht zu den klassischen Kopplungsmethoden, jedoch soll er der Vollständigkeit halber trotzdem hier dargestellt werden.

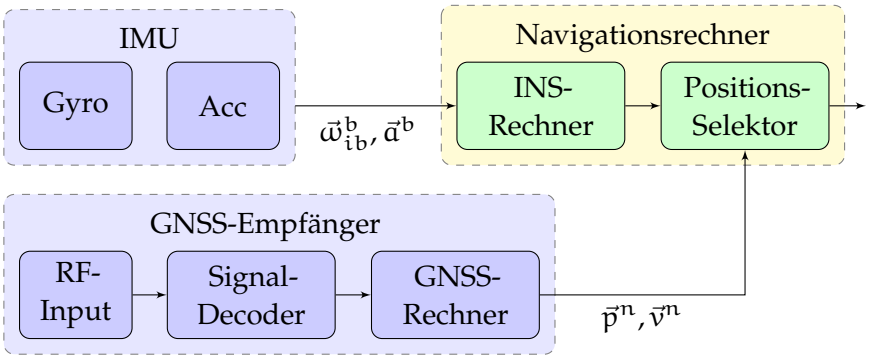


Abbildung 102: Prinzip eines *Uncoupled*-GNSS/INS

A.1.2 *Loosely Coupled*

Das *Loosely Coupled*-Verfahren basiert auf der direkten Nutzung des Positions- und Geschwindigkeitsvektors der GNSS-Lösung zur Stützung des inertialen Navigationssystems. Dabei kann ein Kalman-Filter zur Fusion der unabhängigen Größen aus INS und GNSS dienen. Der Zustandsvektor $\hat{\vec{x}}$ des Filters kann beispielsweise in Form der

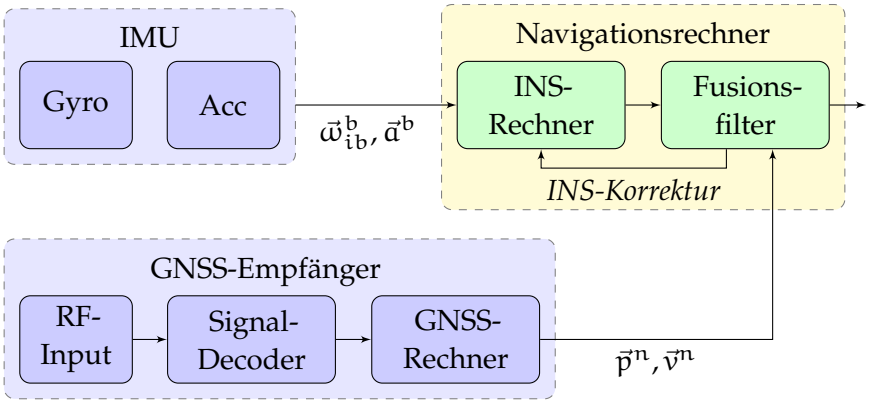


Abbildung 103: Prinzip eines *Loosely Coupled*-GNSS/INS

Gleichung A.1, bestehend aus Position ($\vec{p}^{n}$), Geschwindigkeit ($\vec{v}^{n}$), Lagequaternion ($\vec{q}$), Beschleunigungsbias ($\vec{b}_a$) sowie Drehratenbias ($\vec{b}_\omega$) beschrieben werden:

$$\hat{\vec{x}} = \left(\vec{p}^{n},\ \ \vec{v}^{n},\ \ \vec{q},\ \ \vec{b}_a,\ \ \vec{b}_\omega\right)^{T}. \tag{A.1}$$

Das Prinzip dieser Kopplungsmethode ist in Abbildung 103 dargestellt. Der Vorteil des *Loosely Coupled*-Konzeptes liegt in der verhältnismäßig einfachen Integration, da die Werte der IMU und jedes herkömmlichen GNSS-Empfängers verarbeitet werden können. Nachteilig wirkt sich jedoch der Umstand aus, dass in vielen GNSS-Empfängern ein gesondertes Kalman-Filter zur Positions- und Geschwindigkeitsschätzung zum Einsatz kommt. Durch die Kaskadierung dieser beiden Filter kann es zu möglichen Fehlschätzungen des Fusionsfilters kommen. So wird es auch in [161] und [172] beschrieben. Ein weiterer Nachteil liegt in dem Fusionsansatz begründet, da die Stützung auf der Positionslösung des GNSS-Empfängers beruht. Deshalb kann auch nur eine Stützung erfolgen, wenn mindestens vier oder mehr Satelliten gleichzeitig und störungsfrei empfangen werden können.

A.1.3 *Tightly Coupled*

Beim *Tightly Coupled*-Integrationsansatz wird auf die Positions- und Geschwindigkeitsauswertung durch den GNSS-Empfänger verzichtet. Stattdessen werden die errechneten Entfernungen direkt zu den Satelliten, bezeichnet als *Pseudorange* ($\tilde{\rho}$), und dessen Ableitung ($\dot{\tilde{\rho}}$) verarbeitet (siehe auch [70]). Der Vorteil dieser Methode liegt in der Möglichkeit, das INS ebenfalls zu stützen, wenn weniger als vier Satelliten sichtbar sind. Nachteilig wirkt sich jedoch ein erheblich höherer Rechenaufwand des Navigationscomputers aus, da dieser den Zeitfehler zwischen GNSS-Empfängeruhr und der Satellitenuhr schätzen muss (siehe [60]). Ebenfalls wachsen hiermit die Anforderungen

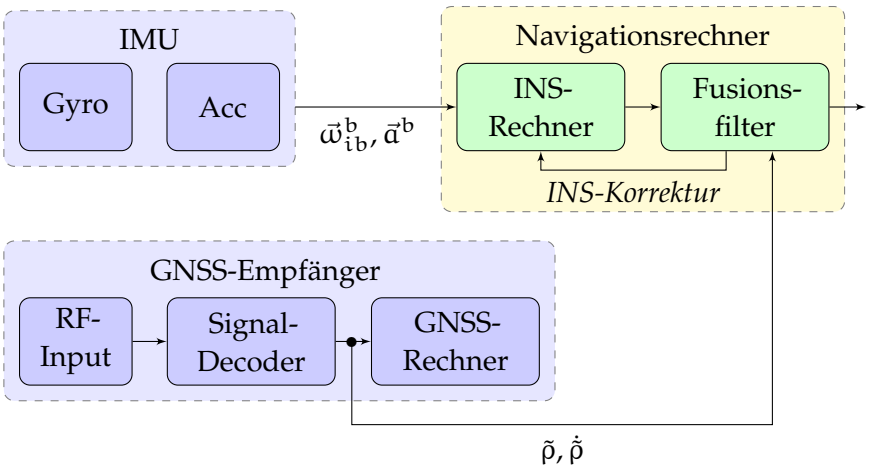


Abbildung 104: Prinzip eines *Tightly Coupled*-GNSS/INS

an den GNSS-Empfänger, da dieser die benötigten Daten bereitstellen muss. Das Systemkonzept ist in Abbildung 104 dargestellt. Umfangreichere Betrachtungen dieses Konzeptes werden in [161] vorgenommen.

A.1.4 *Ultra-Tightly/Deeply Coupled*

Den komplexesten Ansatz zeigt das *Ultra-Tightly* bzw. *Deeply Coupled*-Integrationskonzept. Hierbei nutzt das Fusionsfilter das INS zur direkten Stützung der Korrelationsverfahren und der sogenannten *Tracking Loops* der Satelliten. Durch die Rückkopplung in den Signalpfad des GNSS-Empfängers kann die Genauigkeit und die Verlässlichkeit der Navigationslösung gegenüber den vorherigen Verfahren nochmals gesteigert werden. Hierzu wird jedoch ein sehr umfangreicher Zugriff

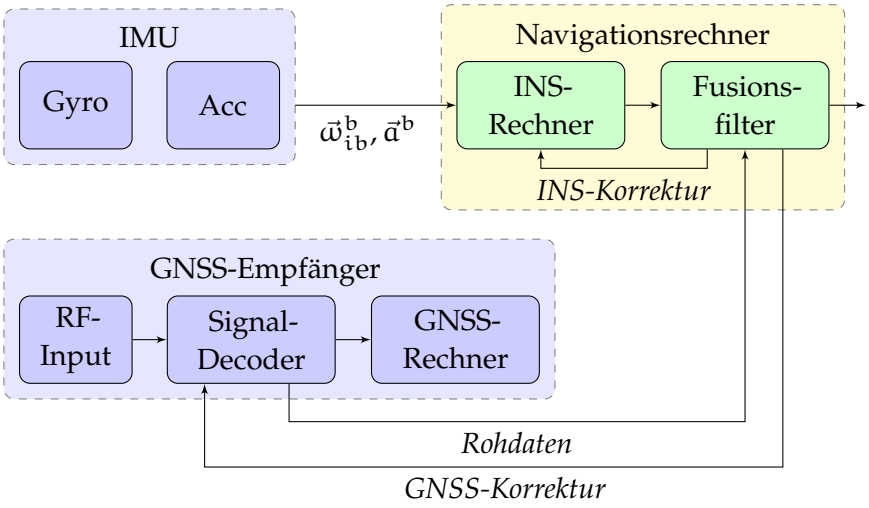


Abbildung 105: Prinzip eines *Deeply Coupled*-GNSS/INS

auf den GNSS-Empfänger benötigt. Zusätzlich sollte zur Berechnung ein leistungsstarker Navigationsrechner zur Verfügung stehen. Das grundlegende Systemkonzept wird in Abbildung 105 gezeigt. Auf eine detaillierte Betrachtung dieses sehr komplexen Verfahrens soll in dieser Arbeit verzichtet werden, da zum Verständnis wesentliche Kenntnisse in der Signalverarbeitung sowie in der Decodierung von GNSS-Satellitensignalen notwendig sind. Eine umfangreiche Diskussion zur Theorie der Satellitennavigation und deren Integrationskonzepte sind beispielsweise in [67, 70, 172] oder in [180] zu finden.

A.2 FEHLERABSCHÄTZUNG BEI GPS

Die Genauigkeit der GPS-Lösung ist multiplen Fehlern unterlegen. Konventionelle, kostengünstige Ein-Frequenz-Empfänger werten lediglich den C/A-Code für die Positions- und Geschwindigkeitslösung aus. Nach [70] können die unterschiedlichen Fehlerquellen entsprechend Tabelle 9 quantitativ beschrieben werden.

FEHLERURSACHE	MITTLERER FEHLER
Uhrenfehler des Satelliten	3.0 m
Bahnstörungen	1.0 m
Vereinfachungen in der Störungstheorie	4.2 m
Bahnmanöver des Satelliten	0.9 m
Refraktion in der Ionosphäre	5.0 m
Refraktion in der Troposphäre	1.5 m
Empfängerrauschen	1.5 m
Mehrwegeempfang	2.5 m
Sonstiges	0.5 m
Summe der Fehlerquadrate	$64.45\,\mathrm{m}^2$
Gesamt RMSE	8.03 m

Tabelle 9: Fehlerabschätzung für das GPS nach [70]

Dementsprechend ist bei derartigen Empfängern mit einem durchschnittlichen Fehler von 8.0 m zu rechnen. Um die Positionsgenauigkeit zu steigern, können unterschiedliche Methoden eingesetzt werden. So kann beispielsweise der Fehler durch Refraktion in der Ionosphäre durch Nutzung von Zwei-Frequenz-Empfängern (L1 und L2) kompensiert werden. Ebenso kann die Genauigkeit durch die Nutzung von *Differential-GPS* (DGPS)[1] bis in den cm-Bereich gesteigert werden.

1 Beim DGPS handelt es sich um ein Korrekturverfahren, bei dem Korrekturdaten von ortsfesten Referenzstationen ausgestrahlt und zur Genauigkeitssteigerung im Empfänger des Nutzers verarbeitet werden.

B

ANHANG: KALIBRIERUNG VON SENSOREN

B.1 KALIBRIERUNG VON DREHRATENSENSOREN

Die Kalibrierung von Sensoren ist in unterschiedlichen Anwendungen eine wichtige Voraussetzung für die Güte der Messwerte. Dabei können fertigungsbedingte Fehler ebenso wie systembedingte Fehler deutlichen Einfluss auf die Genauigkeit der Messwerte haben. Allgemein gilt für Drehratensensoren das Fehlermodell:

$$\tilde{\vec{\omega}}_{ib}^{b} = \mathbf{M}\vec{\omega}_{ib}^{b} + \vec{b}_{\omega} + \vec{n}_{\omega}. \tag{B.1}$$

Darin entspricht die Matrix $\mathbf{M}$ der *Misalignment*-Matrix, in der die Parameter für Ausrichtungs- und Skalenfaktorfehler der Sensorachsen abgebildet werden. Die Nullpunktfehler (*Bias*) werden vom Vektor $\vec{b}_{\omega}$ repräsentiert und das Sensorrauschen (*Noise*) ist im Vektor $\vec{n}_{\omega}$ abgebildet. Diese Fehlerterme wirken auf die ideale Drehrate $\vec{\omega}_{ib}^{b}$ ein, wodurch sich die gemessene Drehrate $\tilde{\vec{\omega}}_{ib}^{b}$ ergibt. Durch eine Sensorkalibrierung sollen nun die Parameter für den Bias und das Sensor-*Misalignment* gefunden werden. Beim Rauschen wird von einem normalverteilten, weißen Rauschen ausgegangen.

B.1.1 *Messaufbau und theoretischer Hintergrund*

Die grundlegende Herausforderung bei der Sensorkalibrierung ist die Generierung von Referenzwerten. Zunächst wird daher ein Teststand entwickelt, welcher Referenzdaten mit mindestens der doppelten angestrebten Kalibriergenauigkeit erzeugt. Hierzu wird auf der vorhergegangenen Masterarbeit [97] aufgebaut, in der ein Präzisionsdrehtisch zur Vermessung eines 1-achsigen Gyroskops genutzt wird.

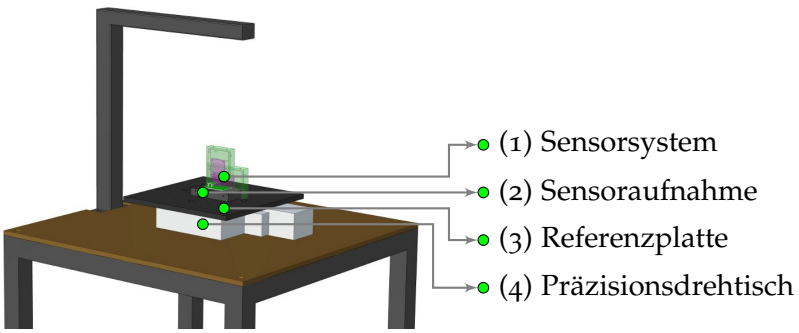


Abbildung 106: Aufbau für die Kalibrierung der Drehratensensoren

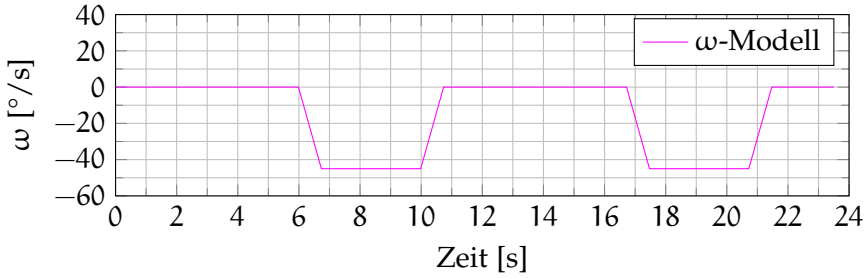


Abbildung 107: Referenzdatentrajektorie zur Kalibrierung des Drehratensensors

Der dort gezeigte Aufbau wird zu diesem Zweck weiterentwickelt, um im Folgenden ein 3-achsiges Sensorsystem zu vermessen (siehe Abbildung 106 auf Seite 219).

Ausgangspunkt ist der Präzisionsdrehtisch *x-Act RT170ST* der Firma Linos (siehe [139]). Dieser wird vom Hersteller mit einer Auflösung von 0.001°, einer absoluten Positionierbarkeit von ±0.05° und einer reproduzierbaren Positionierbarkeit von ±0.01° angegeben. Montiert wird der Präzisionsdrehtisch (siehe Abbildung 106 (4)) auf einem starren und präzise auszurichtenden Kalibriergestell. Die Referenzplatte (3) für die Sensoraufnahme (2) wird mit einer Ausrichtungsgenauigkeit von ±0.05° eingemessen. Durch die beigestellte Interfacekarte (3-*Kanal Schrittmotorkarte M50 c.PCI*) kann der Drehtisch von einem PC aus gesteuert werden. Um den hohen Anforderungen der Kalibrierung gerecht zu werden, ist eine Software in C++ entwickelt worden, welche die Sensorkalibrierung synchronisiert in Echtzeit steuert. Mit dieser kann vom Drehtisch eine spezifische Trajektorie mit vorgegebener Drehrate und Drehratenbeschleunigung abgefahren werden. Des Weiteren erfasst die Modellierungssoftware synchron die Sensordaten des Drehratensensors sowie die Referenztrajektorie des Drehtisches mit $\geqslant 66\,\mathrm{Hz}$ und speichert diese.

Die Referenztrajektorie muss zwei wesentliche Zustände abbilden. Zum einen muss eine statische Phase implementiert sein, in welcher die Nullpunktfehler der drei Sensorachsen bestimmt werden können. Zum anderen muss eine dynamische Phase implementiert werden, um die Skalierungsfehler berechnen zu können. Wird weiter festgelegt, dass jeweils nur um eine körperfeste Achse mit bekannter Drehrate gedreht wird, können ebenfalls Ausrichtungsfehler aufgrund der Beeinflussung der Querachsen bestimmt werden. Aufbauend auf den Ergebnissen aus [97] ist in Abbildung 107 die den geschilderten Anforderungen entsprechend entwickelte Referenztrajektorie dargestellt. Somit befindet sich das Sensorsystem (siehe Abbildung 106 (1)) bis zur Sekunde 5.5 in Ruhelage. In dieser Zeit wird der Bias der je-

weiligen Sensorachse (hier durch ein □ gekennzeichnet) durch den arithmetischen Mittelwert bestimmt, entsprechend:

$$b_{\omega,\square} = \frac{1}{N}\sum_{j=0}^{N} \tilde{\omega}^{b}_{ib,\square,j} \tag{B.2}$$

mit

$$N = \frac{5.5\,s}{\Delta t} + 1. \tag{B.3}$$

Darauf folgt eine 180° Drehung. Hierzu wird das System mit $60°/s^2$ auf eine konstante Drehrate von $-45°/s$ beschleunigt[1]. Kurz vor Erreichen des angestrebten Drehwinkels wird das System mit $60°/s^2$ verzögert, um bei exakt 180° in die Ruhelage zurückzukehren. Zur Validierung der Ergebnisse wird dieser Messablauf ein zweites Mal absolviert. Zur Bestimmung der Skalenfaktor- sowie Ausrichtungsfehler folgt zuerst eine Biaskorrektur der Messwerte der Sensorachsen. So können, analog zu [104], die Messwerte $\tilde{\omega}^{b}_{ib,\square}$ als Funktion der Skalenfaktor- sowie Ausrichtungsfehler und der tatsächlichen Drehrate formuliert werden zu:

$$\begin{aligned}\left(\tilde{\omega}^{b}_{ib,x} - b_{\omega,x}\right) = \omega^{b}_{ib,x} S_x + \omega^{b}_{ib,y}\sin(\Phi_{xy}) \\ + \omega^{b}_{ib,z}\sin(\Phi_{xz})\,,\end{aligned} \tag{B.4}$$

$$\begin{aligned}\left(\tilde{\omega}^{b}_{ib,y} - b_{\omega,y}\right) = \omega^{b}_{ib,x}\sin(\Phi_{yx}) + \omega^{b}_{ib,y} S_y \\ + \omega^{b}_{ib,z}\sin(\Phi_{yz})\end{aligned} \tag{B.5}$$

sowie

$$\begin{aligned}\left(\tilde{\omega}^{b}_{ib,z} - b_{\omega,z}\right) = \omega^{b}_{ib,x}\sin(\Phi_{zx}) + \omega^{b}_{ib,y}\sin(\Phi_{zy}) \\ + \omega^{b}_{ib,z} S_z\,.\end{aligned} \tag{B.6}$$

Darin entspricht $S_\square$ dem Skalenfaktorfehler der Sensorachsen und $\Phi_{\square\square}$ der jeweiligen Achsenkomponente des Ausrichtungswinkelfehlers ($\delta\vec{\Psi}$). So wird beispielsweise der Ausrichtungswinkelfehler bezüglich der x-Achse ($\delta\Psi_x$) in die Komponente, welche auf die y-Achse einwirkt (Φ_{xy}) und in die Komponente, welche auf die z-Achse einwirkt (Φ_{xz}), unterteilt (siehe [104]). Zusammengefasst ergibt sich:

$$\begin{pmatrix} \tilde{\omega}^{b}_{ib,x} - b_{\omega,x} \\ \tilde{\omega}^{b}_{ib,y} - b_{\omega,y} \\ \tilde{\omega}^{b}_{ib,z} - b_{\omega,z} \end{pmatrix} = \begin{pmatrix} S_x & \sin(\Phi_{xy}) & \sin(\Phi_{xz}) \\ \sin(\Phi_{yx}) & S_y & \sin(\Phi_{yz}) \\ \sin(\Phi_{zx}) & \sin(\Phi_{zy}) & S_z \end{pmatrix} \cdot \begin{pmatrix} \omega^{b}_{ib,x} \\ \omega^{b}_{ib,y} \\ \omega^{b}_{ib,z} \end{pmatrix}. \tag{B.7}$$

1 Obwohl die gewählte Drehrate 3°/s außerhalb der Spezifikationen liegt, haben eingehende Untersuchungen und Rückfragen beim Hersteller gezeigt, dass der Drehtisch ohne weitere Einschränkungen in dieser Konfiguration betrieben werden darf.

Die *Misalignment*-Matrix $\mathbf{M}$ ergibt sich demgemäß zu:

$$\mathbf{M} = \begin{pmatrix} S_x & \sin(\Phi_{xy}) & \sin(\Phi_{xz}) \\ \sin(\Phi_{yx}) & S_y & \sin(\Phi_{yz}) \\ \sin(\Phi_{zx}) & \sin(\Phi_{zy}) & S_z \end{pmatrix}. \tag{B.8}$$

Wird die *Misalignment*-Matrix $\mathbf{M}$ mit ihren Elementen Skalenfaktorfehler ($S_\square$) und Missweisung ($M_{\square\square}$) dargestellt, dann folgt:

$$\begin{pmatrix} \tilde{\omega}^b_{ib,x} - b_{\omega,x} \\ \tilde{\omega}^b_{ib,y} - b_{\omega,y} \\ \tilde{\omega}^b_{ib,z} - b_{\omega,z} \end{pmatrix} \begin{pmatrix} S_x & M_{xy} & M_{xz} \\ M_{yx} & S_y & M_{yz} \\ M_{zx} & M_{zy} & S_z \end{pmatrix} \begin{pmatrix} \omega^b_{ib,x} \\ \omega^b_{ib,y} \\ \omega^b_{ib,z} \end{pmatrix}. \tag{B.9}$$

Um die neun Parameter der *Misalignment*-Matrix zu lösen, wird das Sensorsystem jeweils um eine festgelegte Achse mit der Modelltrajektorie rotiert. Hierzu wird das Sensorsystem auf der Referenzplatte so angebracht (siehe Abbildung 106 – *grüne* Darstellung – auf Seite 219), dass die Drehbewegung des Drehtisches auf eine spezifische, körperfeste Achse wirkt. Beginnend mit der Rotation um die x-Achse ergibt sich:

$$\begin{pmatrix} \omega^b_{ib,x} \\ \omega^b_{ib,y} \\ \omega^b_{ib,z} \end{pmatrix} = \begin{pmatrix} \omega\text{-Modell} \\ 0 \\ 0 \end{pmatrix}. \tag{B.10}$$

Somit kann die erste Spalte der Matrix gelöst werden zu:

$$S_x = \frac{\tilde{\omega}^b_{ib,x} - b_{\omega,x}}{\omega\text{-Modell}} \tag{B.11}$$

$$M_{yx} = \frac{\tilde{\omega}^b_{ib,y} - b_{\omega,y}}{\omega\text{-Modell}} \tag{B.12}$$

$$M_{zx} = \frac{\tilde{\omega}^b_{ib,z} - b_{\omega,z}}{\omega\text{-Modell}}. \tag{B.13}$$

Daraufhin wird das Sensorsystem auf der Referenzplatte gedreht und damit um die körperfeste y-Achse rotiert; das führt zu:

$$\begin{pmatrix} \omega^b_{ib,x} \\ \omega^b_{ib,y} \\ \omega^b_{ib,z} \end{pmatrix} = \begin{pmatrix} 0 \\ \omega\text{-Modell} \\ 0 \end{pmatrix}. \tag{B.14}$$

Es folgt:

$$M_{xy} = \frac{\tilde{\omega}^b_{ib,x} - b_{\omega,x}}{\omega\text{-Modell}} \tag{B.15}$$

$$S_y = \frac{\tilde{\omega}^b_{ib,y} - b_{\omega,y}}{\omega\text{-Modell}} \tag{B.16}$$

$$M_{zy} = \frac{\tilde{\omega}^b_{ib,z} - b_{\omega,x}}{\omega\text{-Modell}}. \tag{B.17}$$

Abschließend wird diese Messprozedur ein drittes Mal absolviert und um die körperfeste z-Achse rotiert:

$$\begin{pmatrix} \omega^{b}_{ib,x} \\ \omega^{b}_{ib,y} \\ \omega^{b}_{ib,z} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \omega\text{-Modell} \end{pmatrix}. \tag{B.18}$$

Die letzten drei Parameter ergeben sich zu:

$$M_{xz} = \frac{\tilde{\omega}^{b}_{ib,x} - b_{\omega,x}}{\omega\text{-Modell}} \tag{B.19}$$

$$M_{yz} = \frac{\tilde{\omega}^{b}_{ib,y} - b_{\omega,y}}{\omega\text{-Modell}} \tag{B.20}$$

$$S_{z} = \frac{\tilde{\omega}^{b}_{ib,z} - b_{\omega,z}}{\omega\text{-Modell}}. \tag{B.21}$$

Mit der hier gezeigten neuen Sensorkalibrierung auf Basis einer präzisen Referenztrajektorie, ist eine Sensorkalibrierung mit einer theoretischen Genauigkeit von $\leqslant \pm 0.1°$ für die Winkelfehler möglich. Die tatsächlich mögliche Genauigkeit hängt zudem von der Auflösung sowie Messgenauigkeit des Sensorsystems ab.

B.2 KALIBRIERUNG VON BESCHLEUNIGUNGSSENSOREN

Analog zum Gyroskopmodell kann das Sensormodell von Beschleunigungssensoren anhand der gemessenen Beschleunigung $\tilde{\vec{a}}^b$ und der tatsächlichen Beschleunigung $\vec{a}^b$ grundsätzlich formuliert werden zu:

$$\tilde{\vec{a}}^b = \mathbf{M}\vec{a}^b + \vec{b}_a + \vec{n}_a. \tag{B.22}$$

Im Folgenden wird daher eine Kalibriermethode vorgestellt, mit deren Hilfe die Parameter für Sensorbias sowie *Misalignment*-Matrix bestimmt werden können. Das Rauschen wird auch in diesem Fall als normalverteilt und mittelwertfrei angenommen.

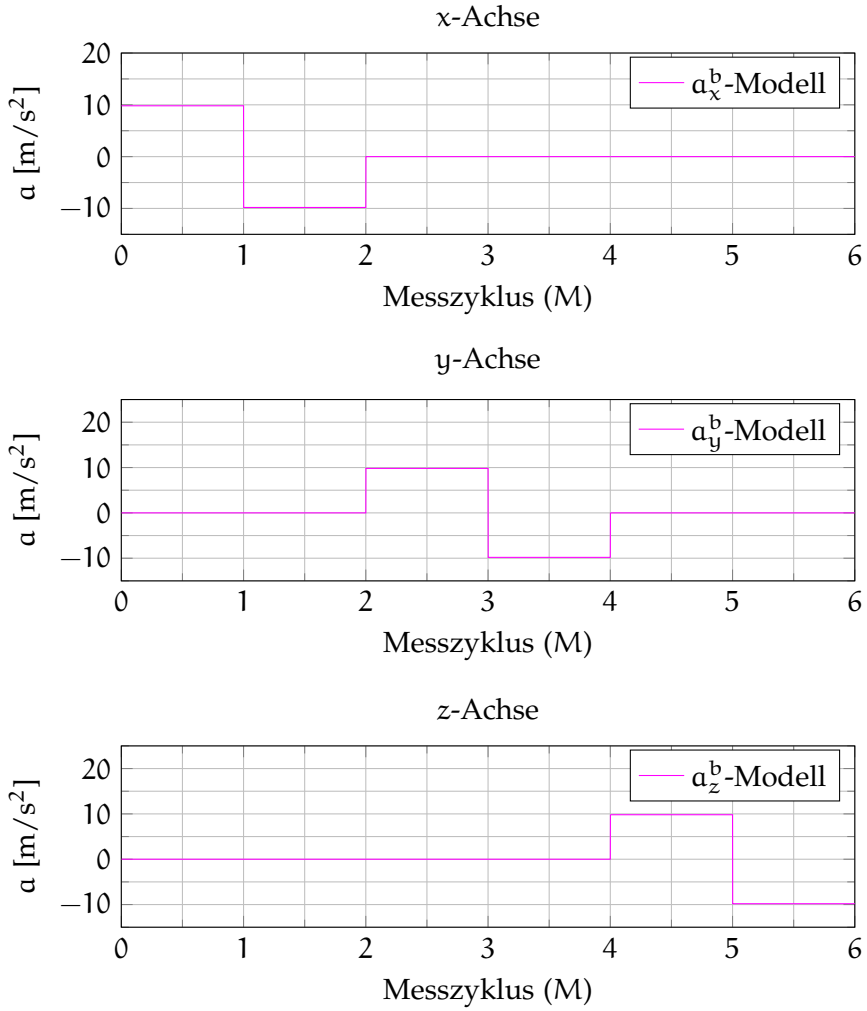


Abbildung 108: Referenzdatentrajektorie zur Kalibrierung des Beschleunigungssensors

B.2.1 *Messaufbau und theoretischer Hintergrund*

Ebenso wie bei der Kalibrierung von Drehratensensoren ist der Messaufbau ein entscheidendes Kriterium für die Kalibriergenauigkeit. Die mit einer Ausrichtungsgenauigkeit von $\pm 0.05°$ eingemessene Referenzplatte des Messaufbaus zur Drehratensensorkalibrierung (siehe Abbildung 106 (3) auf Seite 219) bildet den Ausgangspunkt zur hier gezeigten Kalibrierung des Beschleunigungssensors. Auf dieser Referenzebene wird das Beschleunigungssensorsystem in sechs diskreten Messzyklen mit jeder der drei körperfesten Sensorachsen parallel zur Erdgravitation ausgerichtet. Somit wirkt auf jede Sensorachse die Erdgravitation mit $|\vec{g}_l^n| = 9.8114\,\mathrm{m/s^2}$, einmal in positiver und einmal in negativer Richtung. Die Wirkrichtung wird durch den Vorzeichenparameter s beschrieben. Im Idealfall ergibt sich ein Messmuster, wie in Abbildung 108 dargestellt. Zur Bestimmung der Nullpunktfehler wird über jeden Messzyklus (M) der Mittelwert jeder Sensorachse gebildet. Mit der Annahme, dass das Rauschen des Sensors normalverteilt und mittelwertfrei ist, sollte bei entsprechend gewählter Datensatzlänge (N) jedes Messzyklusses dieses vernachlässigbar sein. Der Bias der x-Achse berechnet sich damit zu:

$$x_{max} = \frac{1}{N}\sum_{M=0}^{M=1} \tilde{a}^b_{x,j} \qquad x_{min} = \frac{1}{N}\sum_{M=1}^{M=2} \tilde{a}^b_{x,j} \tag{B.23}$$

$$b_{a,x} = \frac{(x_{max} + x_{min})}{2}. \tag{B.24}$$

Entsprechend gilt für die y-Achse:

$$y_{max} = \frac{1}{N}\sum_{M=2}^{M=3} \tilde{a}^b_{y,j} \qquad y_{min} = \frac{1}{N}\sum_{M=3}^{M=4} \tilde{a}^b_{y,j} \tag{B.25}$$

$$b_{a,y} = \frac{(y_{max} + y_{min})}{2}. \tag{B.26}$$

Abschließend ergibt sich der Bias für die z-Achse zu:

$$z_{max} = \frac{1}{N}\sum_{M=4}^{M=5} \tilde{a}^b_{z,j} \qquad z_{min} = \frac{1}{N}\sum_{M=5}^{M=6} \tilde{a}^b_{z,j} \tag{B.27}$$

$$b_{a,z} = \frac{(z_{max} + z_{min})}{2}. \tag{B.28}$$

Analog der Herleitung zur Bestimmung der Parameter für die *Misalignment*-Matrix der Drehratensensoren wird die Gleichung B.22 der Beschleunigungssensoren entsprechend umgestellt zu:

$$\begin{pmatrix} \tilde{a}^b_x - b_{a,x} \\ \tilde{a}^b_y - b_{a,y} \\ \tilde{a}^b_y - b_{a,z} \end{pmatrix} = \begin{pmatrix} S_x & M_{xy} & M_{xz} \\ M_{yx} & S_y & M_{yz} \\ M_{zx} & M_{zy} & S_z \end{pmatrix} \begin{pmatrix} a^b_x \\ a^b_y \\ a^b_z \end{pmatrix}. \tag{B.29}$$

Die Skalenfaktor- und Ausrichtungsfehler können bestimmt werden, indem im jeweiligen Messzyklus die Wirkung der Erdgravitation auf die Sensorachsen ausgewertet wird. Bei einer Ausrichtung der x^b-Achse parallel zur Erdgravitation, gilt für den ersten Messzyklus:

$$\begin{pmatrix} a_x^b \\ a_y^b \\ a_z^b \end{pmatrix} = \begin{pmatrix} s \cdot |\vec{g}_l^n| \\ 0 \\ 0 \end{pmatrix}. \tag{B.30}$$

Somit kann die erste Spalte der Matrix gelöst werden zu:

$$S_x = \frac{\tilde{a}_x^b - b_{a,x}}{s \cdot |\vec{g}_l^n|} \tag{B.31}$$

$$M_{yx} = \frac{\tilde{a}_y^b - b_{a,y}}{s \cdot |\vec{g}_l^n|} \tag{B.32}$$

$$M_{zx} = \frac{\tilde{a}_z^b - b_{a,z}}{s \cdot |\vec{g}_l^n|}. \tag{B.33}$$

Bei Drehung der Ausrichtung der y^b-Achse parallel zur Erdgravitation ergibt sich:

$$\begin{pmatrix} a_x^b \\ a_y^b \\ a_z^b \end{pmatrix} = \begin{pmatrix} 0 \\ s \cdot |\vec{g}_l^n| \\ 0 \end{pmatrix}. \tag{B.34}$$

Demgemäß wird die zweite Spalte der Matrix gelöst zu:

$$M_{xy} = \frac{\tilde{a}_x^b - b_{a,x}}{s \cdot |\vec{g}_l^n|} \tag{B.35}$$

$$S_y = \frac{\tilde{a}_y^b - b_{a,y}}{s \cdot |\vec{g}_l^n|} \tag{B.36}$$

$$M_{zy} = \frac{\tilde{a}_z^b - b_{a,z}}{s \cdot |\vec{g}_l^n|}. \tag{B.37}$$

Abschließend wird die z^b-Achse des Beschleunigungssensorsystems parallel zur Erdgravitation ausgerichtet:

$$\begin{pmatrix} a_x^b \\ a_y^b \\ a_z^b \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ s \cdot |\vec{g}_l^n| \end{pmatrix}. \tag{B.38}$$

Daraus ergeben sich die Parameter der letzten Spalte der Matrix zu:

$$M_{xz} = \frac{\tilde{a}_x^b - b_{a,x}}{s \cdot |\vec{g}_l^n|} \tag{B.39}$$

$$M_{yz} = \frac{\tilde{a}_y^b - b_{a,y}}{s \cdot |\vec{g}_l^n|} \tag{B.40}$$

$$S_z = \frac{\tilde{a}_z^b - b_{a,z}}{s \cdot |\vec{g}_l^n|}. \tag{B.41}$$

Analog zur Biasbestimmung werden auch bei der Bestimmung der *Misalignment*-Matrix die Mittelwerte der gemessenen N-Sensormesswerte der jeweiligen Sensorachse genutzt. Des Weiteren kann die *Misalignment*-Matrix sowohl in positiver als auch in negativer z^n-Richtung bestimmt werden. Idealerweise sollten die Parameter in beide Richtungen identisch sein. In praktischen Tests hat sich jedoch gezeigt, dass bei „*Low-Cost*"-Beschleunigungssensoren die Sensitivität nicht identisch ist und sich daraus folgend abweichende Matrizen ergeben können.

Mit dem hier dargestellten Kalibrierverfahren für Beschleunigungssensoren auf Basis einer präzise ausgerichteten Referenzfläche ist eine Sensorkalibrierung mit einer theoretischen Genauigkeit von $\leqslant \pm 0.1°$ für die Winkelfehler möglich. Somit kann ein Skalenfaktorfehler mit $0.017\,\mathrm{m/s^2}$ bestimmt werden. Die in der Praxis erreichbare Genauigkeit hängt, analog zu den Drehratensensoren, von der Auflösung und der Messgenauigkeit des Sensorsystems ab.

B.3 KALIBRIERUNG VON MAGNETFELDSENSOREN

Die Kalibrierung von Magnetometern hat einen besonderen Stellenwert bei der Navigationskopplung mit einem GNSS-System. Hier führen schon geringe Fehler in der Ausrichtungsbestimmung zu deutlichen Abweichungen in der Navigationslösung. Dabei sind die auf dieses Sensorsystem wirkenden Fehler in der Regel relativ groß. Sie hängen einerseits von äußeren Faktoren, wie der Umgebung, und andererseits von designtechnischen Faktoren der Flugplattform ab. So erzeugt der den einzelnen Verbrauchern, beispielsweise den Antrieben, zugeführte elektrische Strom ein elektromagnetisches Feld, welches zu Messfehlern im Magnetometer und damit zu Missweisungen in der Ausrichtungsbestimmung (*Heading*) führt. Allgemein werden diese Einflussfaktoren als Hart- bzw. Weicheisen (*Hard- and Soft-Iron*)-Effekte bezeichnet. Den Ausgangspunkt der Kalibrierung bildet das ideale, nicht verzerrte Magnetfeld, das als Kugel mit dem Radius der absoluten Intensität $|\vec{h}^n| = 49.0089\,\mu T$ im Koordinatenursprung $(0,0,0)$ (siehe Abbildung 109 – *magenta* Kugel) dargestellt werden kann[2]. Zur Kalibrierung empfiehlt es sich, die Messwerte im eingebauten Zustand in der Flugplattform aufzunehmen. So werden die plattformspezifischen Einflüsse direkt mit abgebildet. Dementsprechend wird das MAV im Außenbereich um alle drei körperfesten Achsen gleichmäßig rotiert. Als Ergebnis erhält man eine Punktwolke in Form eines Ellipsoiden, vergleichbar dem Modell in Abbildung 109

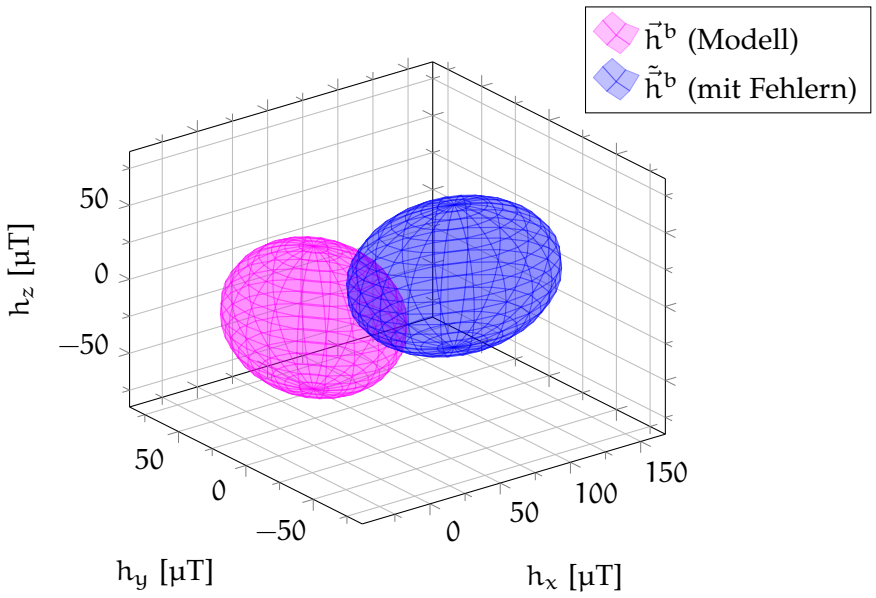


Abbildung 109: *Hard-* und *Soft-Iron*-Effekte

2 $|\vec{h}^n| = 49.0089\,\mu T$ entspricht der absoluten Intensität des Erdmagnetfeldes in Arnsberg nach [123].

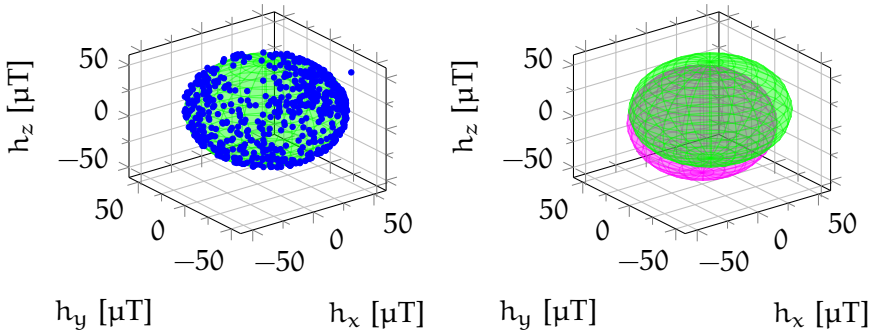


Abbildung 110: Kalibrierung von Magnetfeldsensoren: *links*: eingepasstes Ellipsoid-Modell (*grün*) in die realen Magnetometerrohdaten (*blau*); *rechts*: eingepasstes Ellipsoid-Modell (*grün*) gegenüber dem Erdmagnetfeld-Modell (*magenta*)

(*blauer* Ellipsoid). Der Zusammenhang zwischen dem gemessenen $\tilde{\vec{h}}^b$ und dem tatsächlichen Magnetfeld $\vec{h}^b$ kann als Funktion

$$\tilde{\vec{h}}^b = \mathbf{M}\vec{h}^b + \vec{b}_h + \vec{n}_h \tag{B.42}$$

angegeben werden. Die *Hard-Iron*-Effekte erzeugen dabei einen konstanten additiven Wert zum lokalen Erdmagnetfeld und werden als Bias im Parameter $\vec{b}_h$ des Sensormodells erfasst. Die Folge ist eine Translation des Mittelpunkts der Kugel. *Soft-Iron*-Effekte verzerren die Messung des Erdmagnetfelds selbst. Das führt zu einer Verzerrung der idealen Kugel zu einem Ellipsoid. Folglich haben diese Effekte einen direkten Einfluss auf die sensitiven Achsen und können als Skalenfaktorfehler des Magnetometers modelliert werden. Somit werden diese der *Misalignment*-Matrix $\mathbf{M}$ zugeschlagen.

Zur Bestimmung der Parameter der *Misalignment*-Matrix und des Nullpunktfehlers wird in die Punktwolke der aufgenommenen Messwerte ein Ellipsoid-Modell eingepasst. Entsprechend [111] wird dies mittels der Methode der kleinsten Fehlerquadrate vorgenommen. Beispielhaft ist dieses anhand realer Messdaten in Abbildung 110 *links* dargestellt. Stellt man dieses eingepasste Sensormodell dem realen Erdmagnetfeld gegenüber, ist eine Verschiebung in y- sowie z-Richtung feststellbar (siehe Abbildung 110 *rechts*). Die Parameter des eingepassten Ellipsoids werden anschließend in die ideale Kugel der absoluten Intensität des Erdmagnetfelds transformiert. Als Ergebnis erhält man die inverse *Misalignment*-Matrix sowie den Vektor für den Nullpunktfehler. Nach Umformung der Gleichung B.42 zu

$$\bar{\vec{h}}^b = \mathbf{M}^{-1}(\tilde{\vec{h}}^b - \vec{b}_h) \tag{B.43}$$

können die Fehler im gemessenen Magnetfeld $\tilde{\vec{h}}^b$ kompensiert werden, woraus der korrigierte Messvektor $\bar{\vec{h}}^b$ folgt. Wendet man diese Prozedur auf die zuvor dargestellten Messdaten an, ergibt sich die korrigierte Punktwolke entsprechend Abbildung 111. Im Gegensatz

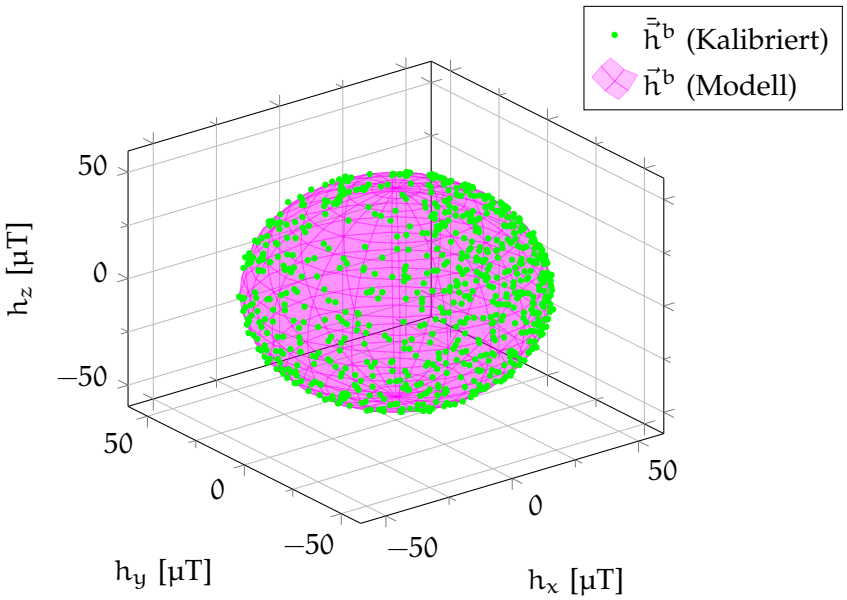


Abbildung 111: Kalibrierung von Magnetfeldsensoren: korrigierte Magnetometerdaten (*grün*) und Erdmagnetfeld-Modell (*magenta*)

zu Extremwert-Verfahren, welche die gemessenen Minima und Maxima der jeweiligen Sensorachse zur Kalibrierparameterbestimmung nutzen, ist das hier vorgestellte Verfahren deutlich stabiler gegenüber Ausreißern. In der realen Anwendung auf einer mobilen Plattform kann dieses Verfahren ebenfalls durch das in Kapitel 4.2 gezeigte *World Magnetic Model* (WMM) erweitert werden, da bei einer bekannten Position die lokale Intensität des Magnetfelds bestimmt werden kann. Somit können in Echtzeit Ausreißer bzw. Fehlmessungen durch äußere Einflüsse während des Flugs bestimmt und direkt kompensiert werden. Dies führt zu einer deutlich höheren Verlässlichkeit bei der Ausrichtungsbestimmung.

ABKÜRZUNGEN UND SYMBOLE

ABKÜRZUNGEN

ADC	Analog-to-Digital-Converter
AFCS	Automatic Flight Control System
ARW	Angular Random Walk
API	Application Programming Interface
BGS	British Geological Survey
BLDC	Brushless DC
BEMT	Blade Element Momentum Theory
CAD	Computer-Aided Design
CFK	Carbonfaserverstärkter Kunststoff
CFD	Computational Fluid Dynamics
CW	Clockwise
CCW	Counterclockwise
CDMA	Code Division Multiple Access
C/A	Coarse/Acquisition
COTS	Commercial-Off-The-Shelf
CPU	Central Processing Unit
CAN	Controller Area Network
DARPA	Defense Advanced Research Projects Agency
DGPS	Differential-GPS
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
ESC	Electronic Speed Controler
EU	Europäische Union
ESA	European Space Agency
EKF	Extended Kalman-Filter
ENAC	Ecole Nationale de l'Aviation Civile
FAA	Federal Aviation Administration
FAI	Fédération Aéronautique Internationale
FFT	Fast Fourier Transform
FOC	Full Operational Capability
FOG	Fiber Optic Gyroscope
FMS	Flight Management System
FMU	Flight Management Unit
FIR	Finite Impulse Response
FS	Full Scale
FPU	Floating Point Unit
GNSS	Global Navigation Satellite System
GPS	Global Positioning System

GLONASS	Globalnaja Nawigazionnaja Sputnikowaja Sistema
GCS	Ground Control Station
GPIO	General-Purpose Input/Output
ECEF	Earth-Centered Earth-Fixed
ETH	Eidgenössische Technische Hochschule
ICAO	International Civil Aviation Organization
IGRF	International Geomagnetic Reference Field
INS	Inertial Navigation System
IMU	Inertial Measurement Unit
ISA	International Standard Atmosphere
IOV	In-Orbit Validation
IIR	Infinite Impulse Response
IGE	In Ground Effect
ISR	Interrupt Service Routine
I/O	Input/Output
I^2C	Inter-Integrated Circuit
HD	High Definition
HiL	Hardware in the Loop
Li-Po	Lithium-Polymer
LTI	Linear Time-Invariant
LSB	Least Significant Bit
LED	Light-Emitting Diode
MAV	Micro Air Vehicle
MEMS	Micro-Electro-Mechanical System
MIU	Main Interface Unit
MMU	Motor Management Unit
NASA	National Aeronautics and Space Administration
NAVSAT	Navigation Satellite System
NAVSTAR	Navigation System with Timing and Ranging
NED	North-East-Down
NGDC	National Geophysical Data Center
ODE	Open Dynamics Engine
OGRE3D	Object-Oriented Graphics Rendering Engine 3D
P	Precise/Encrypted
PRN	Pseudo Random Noise
PWM	Pulsweitenmodulation
PCB	Printed Circuit Board
ROS	Robot Operating System
RPAS	Remotely Piloted Aircraft System
RLG	Ring Laser Gyroscope
RWTH	Rheinisch-Westfälische Technische Hochschule
RMSE	Root Mean Squared Error
SBIR	Small Business Innovation Research
SDK	Software Development Kit
SD	Standard Definition
SLAM	Simultaneous Localization and Mapping

SiL	Software in the Loop
Spice	Simulation Program with Integrated Circuit Emphasis
SPI	Serial Peripheral Interface
SBC	Single Board Computer
SDIO	Secure Digital Input/Output
ToF	Time of Flight
UA	Unmanned Aircraft
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
UART	Universal Asynchronous Receiver and Transmitter
USB	Universal Serial Bus
VSG	Vibrating Structure Gyroscope
VTOL	Vertical Take-Off and Landing
VRW	Velocity Random Walk
VRML97	Virtual Reality Modeling Language 97
WGS	World Geodetic System
WMM	World Magnetic Model

LATEINISCHE BUCHSTABEN

$\vec{a}$	Beschleunigungsvektor
a_C	Coriolisbeschleunigung
a	Temperaturgradient oder Funktionsparameter
a_N , b_M	Filterkoeffizienten
A	Fläche
b	Funktionsparameter
$\vec{b}$	Biasvektor
$\mathbf{B}$	Steuermatrix
c	Funktionsparameter
$\vec{c}_b$	Koeffizientenvektor zur Temperaturkalibrierung
C_D , C_W	Koeffizient des Strömungswiderstands
C_M	Koeffizient des Rotordrehmoments
C_T	Koeffizient der Rotorschubkraft
$\mathbf{C}$	Richtungskosinusmatrix
e	Regeldifferenz oder Exzentrizität des Ellipsoids
f	Abflachung des Ellipsoids oder Frequenz
$\vec{F}$	Kraftvektor
$\vec{F}_g$	Gewichtskraft
$\vec{F}_{R,\Sigma}$	Schubkraft der Antriebe
$\vec{F}_D$	Strömungswiderstandskraft
$\vec{F}_W$	Strömungswiderstandskraft aufgrund von Wind
$\mathbf{F}$	Systemmatrix
$\vec{g}$, $\vec{g}_l^n$, g_0	Schwerebeschleunigung
$\mathbf{G}$	Einflussmatrix
h	Höhe

$\vec{h}$	Magnetische Flussdichte des Erdmagnetfeldes
$\mathbf{H}$	Messmatrix
I	Elektrischer Strom
I_A	Ankerstrom
$\mathbf{I}$	Einheitsmatrix
$\mathbf{J}$	Massenträgheitsmoment
J_A	Massenträgheitsmoment des Ankers
$\mathbf{J}_F$	Massenträgheitsmoment der Flugplattform
$\mathbf{J}_R$	Massenträgheitsmoment des Rotors
k_1 , k_2	Maschinenkonstanten
k_M	Proportionalitätskoeffizient des Rotordrehmoments
k_T	Proportionalitätskoeffizient der Rotorschubkraft
K_U , K_E	Spezifische Motorkonstanten
$\mathbf{K}$	Kalman-Gain
$\vec{l}_C$	Abstand des Massenmittelpunkts zum geometrischen Mittelpunkt
$\vec{l}_D$	Abstand des Massenmittelpunkts zum Angriffspunkt des Strömungswiderstands
$\vec{l}_M$	Länge der Motorarme
$\vec{L}$	Drehimpuls
L_A	Induktivität der Ankerwicklung
m	Masse
$\dot{m}$	Massenstrom
M	Mittlere molare Masse
M	Drehmoment
M_E	Elektromagnetisches Drehmoment
M_L	Lastmoment
M_R	Rotordrehmoment
$\mathbf{M}$	*Misalignment*-Matrix
N_M	Motordrehzahl
N_R	Rotordrehzahl
N	Krümmungsradius des ersten Vertikals
$\vec{n}$	Normalverteiltes Rauschen
$\vec{n}$	Normierter Vektor
O	Koordinatenursprung
p	Luftdruck
p_0	Luftdruck auf Meereshöhe
$\vec{p}$	Positionsvektor
P_j	Phase der Wegpunktverarbeitung
P	Wahrscheinlichkeit
$\mathbf{P}$	Kovarianzmatrix der Systemzustandsschätzung
$\vec{q}$	Lagequaternion
$\mathbf{Q}$	Kovarianzmatrix des System- bzw. Prozessrauschens
R	Universelle Gaskonstante
R	Radius des Rotors

R_a	große Halbachse des Ellipsoids
R_b	kleine Halbachse des Ellipsoids
R_A	Ankerwiderstand
$\mathbf{R}$	Kovarianzmatrix des Messrauschens
s	Eingangssignal, Steuergröße oder Vorzeichenparameter
$\vec{s}$	Vorzeichenparametervektor
$\vec{s}_M$	Motorsteuergrößenvektor
S	Skalenfaktor oder Verstärkungsfaktor
t	Zeit
t_k	Zeitpunkt
Δt	Abtastzeit
T	Zeitraum oder Zeitkonstante
T_ν	Verzögerungszeitkonstante
T_0	Temperatur auf Meereshöhe
T_R	Rotorschubkraft
$\vec{u}$	Eingangsgrößenvektor
U_A	Ankerspannung
$\vec{v}$	Geschwindigkeitsvektor oder Messrauschen
v_i	Einströmungsgeschwindigkeit der Luft
$\vec{v}_W$	Windgeschwindigkeit
$\vec{v}$	Vektor in kartesischen Koordinaten
w	Maximale Strömungsgeschwindigkeit der Luft
$\vec{w}$	System- bzw. Prozessrauschen
X , Y , Z	Koordinatenachsen
x	Ausgangsgröße oder Systemzustand
$\vec{x}$, $\Delta\vec{x}$	Systemzustandsvektor
y	Ausgangssignal
$\lvert Y(f)\rvert$	Amplitudenspektrum in Abhängigkeit der Frequenz f
$\vec{z}$	Messgrößenvektor

GRIECHISCHE BUCHSTABEN

α, β, γ	Lagewinkelfehler
Δ , δ	Fehler, Abweichung, Änderung, Intervall oder Fehlausrichtung
$\Delta\vec{\sigma}$	Lageänderung
$\delta\vec{\Psi}$	Vektor der Ausrichtungswinkelfehler des Sensors (*Misalignment*)
η	Barometrische Höhe
θ	*Pitch*-Winkel
λ	Längengrad
λ_h	Proportionalitätswert (*engl. Induced Inflow Ratio*)
ρ	Luftdichte
$\tilde{\rho}$	Pseudorange

σ , $\vec{\sigma}$	Standardabweichung bzw. Streuung oder Orientierungsvektor
τ	Gruppenlaufzeit
φ	Breitengrad
ϕ	*Roll*-Winkel
Φ_E	Elektromagnetischer Fluss
$\Phi_{\square\square}$	Achsenkomponente des Ausrichtungswinkelfehlers
$\mathbf{\Phi}$	Transitionsmatrix
ψ	*Yaw*-Winkel
$\vec{\Psi}$	Vektor der Lagewinkelfehler α, β und γ
$\mathbf{\Psi}$	Schiefsymmetrische Matrix der Lagewinkelfehler
$\vec{\omega}$	Winkel-/Drehgeschwindigkeitsvektor
ω_M	Motordrehgeschwindigkeit
Ω	Erddrehrate

INDIZES

b	Körperfestes Koordinatensystem (*b-Frame*)
e	Erdfestes Koordinatensystem (*e-Frame*)
i	Inertialkoordinatensystem (*i-Frame*)
n	Navigationskoordinatensystem (*n-Frame*)
$\vec{\square}^b$	Vektor gegeben im *b-Frame*
$\vec{\square}^n$	Vektor gegeben im *n-Frame*
$\vec{\square}^{nf}$	Vektor gegeben im *flugplattformbasierten n-Frame*
$\vec{\square}^{wgs}$	Vektor bzgl. des WGS-84 Referenzellipsoids
$\vec{\square}^n_{ie}$	Vektor *e-Frame* zu *i-Frame*, gegeben im *n-Frame*
$\vec{\square}^b_{ib}$	Vektor *b-Frame* zu *i-Frame*, gegeben im *b-Frame*
$\square_k$	Wert zum Zeitpunkt t_k
$\square_0$	Startparameter
$\square_j$	Index
$\square_x$, $\square_y$, $\square_z$	Komponente in Richtung der *x*-, *y*- oder *z*-Achse
$\square_n$, $\square_e$, $\square_d$	Komponente in *North*-, *East*- oder *Down*-Richtung
$\vec{\square}_{GNSS}$	Vektor auf Basis der transformierten GNSS-Messung
$\square_s$	Führungsgröße (Soll-Wert)
$\square^\star$	Zwischenergebnis
$\square^+$	*a posteriori* Größe
$\square^-$	*a priori* Größe
$\hat{\square}$	Geschätzte Größe
$\tilde{\square}$	Gemessene Größe
$\bar{\square}$	Größe am Linearisierungspunkt oder gemittelte Größe
$[\vec{\square}\times]$	Kreuzproduktbildende Matrix von $\vec{\square}$

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LITERATURVERZEICHNIS

[1] Achtelik, M., Doth, K.-M., Gurdan, D. und Stumpf, J. (2009): *Drehflügelfluggerät*, Patent, DE 102008014853 A1.

[2] Achtelik, M., Doth, K.-M., Gurdan, D. und Stumpf, J. (2012): *Verfahren zur Verbesserung der Flugeigenschaften eines Multikopters in Ausfallsituationen*, Patent, DE 102010040770 B4.

[3] Acuvance Corporation (1990): *Keyence Gyrosaucer II*, Internetseite, URL: https://acuvance.co.jp/rc/dick/index.html, Abruf am 24. Febr. 2016.

[4] Acuvance Corporation (1990): *Keyence Gyrosaucer II - Operating Instructions*, Operating Instructions, Acuvance Corporation.

[5] Air and Space Magazine (2011): *NASA Space Shuttle Flight Simulator*, Internetseite, URL: http://aero.tamu.edu/nasasimulator, Abruf am 19. Juni 2016.

[6] AirRobot (2016): *AirRobot - Homepage*, Internetseite, URL: http://www.airrobot.de/, Abruf am 24. Febr. 2016.

[7] Amazon (2014): *Amazon Prime Air*, Internetseite, URL: http://www.amazon.com/b?ref_=tsm_1_tw_s_amzn_mx3eqp&node=8037720011, Abruf am 08. Dez. 2015.

[8] Analog Devices, Inc. (2007): *ADXL321 - Small and Thin ±18g Accelerometer*, Datenblatt, Rev. 0.

[9] Analog Devices, Inc. (2016): *ADIS16485 - Tactical Grade, Six Degrees of Freedom Inertial Sensor*, Datenblatt, Rev. F.

[10] Anschütz-Kaempfe, H. und von Schirach, F. (1904): *Kreiselapparat*, Patent, DE 000000182855 A.

[11] ArduPilot Autopilot Suite (2016): *Simulation*, Internetseite, URL: http://ardupilot.org/dev/docs/simulation-2.html, Abruf am 16. Okt. 2016.

[12] Asahi Kasei Microdevices Corporation (2010): *AK8975 / AK8975C 3-axis Electronic Compass*, Datenblatt, Rev. 2.

[13] Ascending Technologies (2016): *Ascending Technologies - Homepage*, Internetseite, URL: http://www.asctec.de/, Abruf am 24. Febr. 2016.

[14] Bar-Shalom, Y. und Fortman, T. (1988): *Tracking and Data Association*, Academic Press.

[15] Berliner Zeitung - Rest, J. (2014): *„Project Wing" Google entwickelt Paket-Drohnen*, Internetseite, URL: http://www.berliner-zeitung.de/wirtschaft/--project-wing--google-entwickelt-paket-drohnen,10808230,28256108,item,1.html, Abruf am 08. Dez. 2015.

[16] Bloomberg - Capaccio, T. (2011): *Northrop Drone Flies Over Japan Reactor to Record Data*, Internetseite, URL: http://www.bloomberg.com/news/articles/2011-03-16/, Abruf am 19. Juni 2016.

[17] Bohemia Interactive Simulations (2016): *Virtual Battlespace (VBS) 3*, Internetseite, URL: https://bisimulations.com/virtual-battlespace-3, Abruf am 19. Juni 2016.

[18] Bon Air Drone - Anya Lamb und DroneDeploy (2017): *A Perfect Match: Winning a Mining Client with an Accurate Contour Map*, Internetseite, URL: https://blog.dronedeploy.com/, Abruf am 03. Febr. 2017.

[19] Bortz, J. (1971): *A New Mathematical Formulation for Strapdown Inertial Navigation*, in: IEEE Transactions on Aerospace Electronic Systems, 7, S. 61–66.

[20] Bosch Sensortec GmbH (2012): *BMP180 - Digital pressure sensor*, Datenblatt, Rev. 2.4.

[21] Bosch Sensortec GmbH (2014): *BMI055 - Small, versatile 6DoF sensor module*, Datenblatt, Rev. 1.2.

[22] Bouabdallah, S. und Siegwart, R. (2007): *Advances in Unmanned Aerial Vehicles*, Kap. Design and Control of a Miniature Quadrotor, S. 171–210, Springer Press.

[23] Boykow, J. M. und Siemens Apperate und Maschinen (1938): *Einrichtung zum Messen von Wegstrecken*, Patent, DE 000000661822 A.

[24] Braunschweigische Wissenschaftliche Gesellschaft 2002 (2002): *Biographisches Lexikon zur Geschichte der Geophysik: Torricelli, Evangelista*, Internetseite, URL: http://www.geophys.tu-bs.de/geschichte/torricelli.html, Abruf am 11. Febr. 2016.

[25] Briod, A., Kornatowski, P. M., Zufferey, J.-C. und Floreano, D. (2014): *A Collision Resilient Flying Robot*, in: Journal of Field Robotics, 31(4), S. 496–509.

[26] Brisset, P., Drouin, A., Gorraz, M., Huard, P.-S. und Tyler, J. (2006): *The Paparazzi Solution*, in: MAV 2006, 2nd US-European Competition and Workshop on Micro Air Vehicles, Sandestin, United States.

[27] Bronz, M., Moschetta, J.-M. und Brisset, P. (2010): *Flying Autonomously to Corsica: A Long Endurance Mini-UAV System*, in: Proceedings of the International Micro Air Vechicle Competition and Conference (IMAV) 2010, Branschweig, Germany.

[28] Cadence Design Systems, Inc. (2016): *OrCAD PSpice Designer*, Internetseite, URL: http://www.orcad.com/products/orcad-pspice-designer/overview, Abruf am 23. März 2016.

[29] Cai, G., Chen, B. M. und Lee, T. H. (2011): *Unmanned Rotorcraft Systems*, Advances in Industrial Control, Springer London.

[30] Calafell, E. F. (2011): *CoaX Flight Simulator in Webots*, Master's Thesis, ETH Zürich.

[31] Carpin, S., Lewis, M., Wang, J., Balakirsky, S. und Scrapper, C. (2007): *USARSim: a robot simulator for research and education*, in: Proceedings 2007 IEEE International Conference on Robotics and Automation, S. 1400–1405.

[32] Caruso, M. J. (1997): *Applications of Magnetoresistive Sensors in Navigation Systems*, in: SAE Technical Paper 970602.

[33] Courbon, J., Mezouar, Y., Guenard, N. und Martinet, P. (2009): *Visual navigation of a quadrotor aerial vehicle*, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2009, IROS'09, S. 5315–5320, IEEE Press, Piscataway, NJ, USA.

[34] Crocoll, P. (2015): *Modellbasierte Quadrokopter-Navigation mit Laserstützung*, Dissertation, Karlsruher Institut für Technologie (KIT).

[35] CYBERBOTICS Ltd. (2011): *Webots User Guide*, User Guide, Cyberbotics Ltd.

[36] Deeg, C. (2006): *Modeling, Simulation, and Implementation of an Autonomously Flying Robot*, Dissertation, Technische Universität Berlin.

[37] Department of Defense (DoD) (1997): *Flying Qualities of Piloted Aircraft*, Handbook, Department of Defense (DoD).

[38] DHL (2013): *Forschungsprojekt DHL Paketkopter - vom Festland auf die Insel Juist*, Internetseite, URL: http://www.dhl.de/paketkopter, Abruf am 08. Dez. 2015.

[39] Die Welt (2014): *Flying Robot Rockstars*, Internetseite, URL: http://www.welt.de/wissenschaft/roboter/article127338374/Musikalische-Drohnen-ersetzen-das-Orchester.html, Abruf am 08. Dez. 2015.

[40] Die Welt - Trentmann, N. (2015): *Drohnen sollen endlich schlauer werden*, Internetseite, URL: http://www.welt.de/wirtschaft/article139665552/Drohnen-sollen-endlich-schlauer-werden.html, Abruf am 08. Dez. 2015.

[41] Digi International Inc. (2016): *XBee® DigiMesh® 2.4*, Datenblatt.

[42] DJI (2016): *Mavic Pro*, Internetseite, URL: http://www.dji.com/de/mavic, Abruf am 03. Okt. 2016.

[43] Draganfly Innovations Inc. (2016): *The story behind Draganfly Innovations*, Internetseite, URL: http://www.draganfly.com/our-story/, Abruf am 24. Febr. 2016.

[44] Driessens, S. und Pounds, P. (2013): *Towards a more efficient quadrotor configuration*, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2013, S. 1386–1392.

[45] Durban, J. W., Fearnbach, H., Barrett-Lennard, L. G., Perryman, W. L. und Leroi, D. J. (2015): *Photogrammetry of killer whales using a small hexacopter launched at sea*, in: Journal of Unmanned Vehicle Systems, 3(0), S. 1–5.

[46] eigen.tuxfamily (2016): *Eigen, a C++ template library* , Internetseite, URL: http://eigen.tuxfamily.org/, Abruf am 02. Sept. 2016.

[47] Eisenbeiss, H. (2009): *UAV Photogrammetry*, Dissertation, Institut für Geodäsie und Photogrammetrie an der ETH Zürich.

[48] ETH Zürich, Flying Machine Arena (2016): *The Flying Machine Arena (FMA) - Homepage*, Internetseite, URL: http://flyingmachinearena.org/, Abruf am 24. Febr. 2016.

[49] European Commission (2016): *The history of Galileo*, Internetseite, URL: http://ec.europa.eu/growth/sectors/space/galileo/history/index_en.htm, Abruf am 13. Febr. 2016.

[50] European GNSS Agency (2016): *Galileo-Programme*, Internetseite, URL: http://www.gsa.europa.eu/galileo/programme, Abruf am 13. Febr. 2016.

[51] Eurosimtec (2016): *eurosimtec - Homepage*, Internetseite, URL: http://www.eurosimtec.de, Abruf am 19. Juni 2016.

[52] Faessler, M., Fontana, F., Forster, C. und Scaramuzza, D. (2015): *Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor*, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), S. 1722–1729.

[53] Federal Aviation Administration (FAA) (2012): *Public Law 112-95, Title III, Subtitle B – Unmanned Aircraft Systems, Sec.331.Definitions. (8), (9)*, in: FAA Modernization and Reform Act of 2012.

[54] Federal Aviation Administration (FAA) (2015): *Grant of Exemption-BNSF Railway Company*, in: Regulatory Docket No. FAA–2014-0704.

[55] FlightGear (2016): *FlightGear Flight Simulator*, Internetseite, URL: http://www.flightgear.org/, Abruf am 16. Okt. 2016.

[56] Fraundorfer, F., Heng, L., Honegger, D., Lee, G., Meier, L., Tanskanen, P. und Pollefeys, M. (2012): *Vision-based autonomous mapping and exploration using a quadrotor MAV*, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2012, S. 4557–4564.

[57] Freese, M., Singh, S. P. N., Ozaki, F. und Matsuhira, N. (2010): *Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator.*, in: Proceedings of the International Conference on Simulation Modeling and Programming for Autonomous Robots (SIMPAR) 2010, Bd. 6472, S. 51–62, Springer.

[58] Furrer, F., Burri, M., Achtelik, M. und Siegwart, R. (2016): *Robot Operating System (ROS): The Complete Reference (Volume 1)*, Kap. RotorS—A Modular Gazebo MAV Simulator Framework, S. 595–625, Springer International Publishing.

[59] Fédération Aéronautique Internationale (FAI) (2015): *Rotorcraft World Records*, Internetseite, URL: http://www.fai.org/record-rotorcraft, Abruf am 08. Dez. 2015.

[60] George, M. und Sukkarieh, S. (2005): *Tightly Coupled INS/GPS with Bias Estimation for UAV Applications*, in: Proceedings of Australiasian Conference on Robotics and Automation (ACRA).

[61] GrabCAD (2016): *GrabCAD, a STRATASYS solution - Homepage*, Internetseite, URL: https://grabcad.com/, Abruf am 12. Aug. 2016.

[62] Grant, G. A. A. und Klinkert, J. (1970): *The ship's compass: including general magnetism, theory, practice and calculations relating to magnetic and gyro compasses*, Routledge and K. Paul.

[63] GRASMEYER, J. UND KEENNON, M. (2001): *Development of the Black Widow Micro Air Vehicle*, in: Aerospace Sciences Meetings, 39.

[64] GRASP LABORATORY (2016): *General Robotics, Automation, Sensing and Perception (GRASP) Laboratory - Homepage*, Internetseite, URL: https://www.grasp.upenn.edu/research-groups/kumar-lab, Abruf am 24. Febr. 2016.

[65] GREWAL, M. S. UND ANDREWS, A. P. (2008): *Kalman Filtering: Theory and Practice Using MATLAB*, Wiley.

[66] GREWAL, M. S. UND ANDREWS, A. P. (2010): *Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives]*, in: Control Systems, IEEE, 30(3), S. 69–78.

[67] GROVES, P. (2013): *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition:*, GNSS/GPS, Artech House.

[68] GRZONKA, S., GRISETTI, G. UND BURGARD, W. (2009): *Towards a Navigation System for Autonomous Indoor Flying*, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2009, Kobe, Japan.

[69] HATTENBERGER, G., BRONZ, M. UND GORRAZ, M. (2014): *Using the Paparazzi UAV System for Scientific Research*, in: Proceedings of the International Micro Air Vechicle Competition and Conference (IMAV) 2014, S. 247–252, Delft, Netherlands.

[70] HECKER, P. (2015): *Satellitennavigation - Technologien und Anwendungen*, Vorlesungsscript, Institut für Flugführung, Technische Universität Braunschweig.

[71] HELMHOLTZ-ZENTRUM POTSDAM - DEUTSCHES GEOFORSCHUNGS-ZENTRUM (GFZ) (2017): *Häufig gestellte Fragen zum Thema Erdmagnetismus*, Internetseite, URL: http://www.gfz-potsdam.de/magservice/faq/, Abruf am 03. März 2017.

[72] HISYSTEMS (2016): *MikroKopter UAV - Homepage*, Internetseite, URL: http://www.mikrokopter.de, Abruf am 30. Juni 2016.

[73] HOFFMANN, G. M., HUANG, H., WASL, S. L. UND TOMLIN, E. C. J. (2007): *Quadrotor helicopter flight dynamics and control: Theory and experiment*, in: In Proc. of the AIAA Guidance, Navigation, and Control Conference.

[74] HONEYWELL (2011): *Honeywell T-Hawk Aids Fukushima Daiichi Disaster Recovery*, Internetseite, URL: https://honeywell.com/News/, Abruf am 19. Juni 2016.

[75] Honeywell International Inc. (2013): *HMC5883L - 3-Axis Digital Compass IC*, Datenblatt, Rev. E.

[76] Hundley, R. und Gritton, E. C. (1994): *Future Technology-Driven Revolutions in Military Operations: Results of a Workshop*, RAND Corporation.

[77] Hybrid Systems Lab (2016): *Hybrid Systems Lab - Research Projects*, Internetseite, URL: http://hybrid.eecs.berkeley.edu/research.html, Abruf am 24. Febr. 2016.

[78] Ibenthal, A. (2009): *Systemtheorie*, Vorlesungsscript, Hochschule für angewandte Wissenschaft und Kunst (HAWK) Göttingen.

[79] Intel Corporation (2016): *Intel® Aero Platform Developer Kits*, Internetseite, URL: http://click.intel.com/intel-aero-platform-developer-kits.html, Abruf am 03. Okt. 2016.

[80] International Civil Aviation Organization (ICAO) (1993): *Manual of the ICAO standard atmosphere : extended to 80 kilometres (262 500 feet)*, 3. Aufl., Montreal, Quebec: International Civil Aviation Organization (ICAO).

[81] InvenSense Inc. (2009): *IDG-500 - Integrated Dual-Axis Gyro*, Datenblatt, Rev. 6.

[82] InvenSense Inc. (2009): *ISZ-500 - Single-Axis Z-Gyro*, Datenblatt, Rev. 3.

[83] InvenSense Inc. (2013): *MPU9150*, Datenblatt, Rev. 4.3.

[84] Jenkins, D. und Vasigh, B. (2013): *The economic impact of unmanned aircraft systems integration in the United States*, Economic Report, Association for Unmanned Vehicle Systems International (AUVSI).

[85] Jirka, G. H. (2007): *Einführung in die Hydromechanik*, Univ.-Verlag Karlsruhe.

[86] jMAVSim (2016): *jMAVSim: a simple and lightweight multirotor simulator*, Internetseite, URL: https://pixhawk.org/dev/hil/jmavsim, Abruf am 30. Dez. 2016.

[87] JSBSim (2016): *An open source, platform-independent, flight dynamics and control software library in C++*, Internetseite, URL: http://jsbsim.sourceforge.net/, Abruf am 16. Okt. 2016.

[88] Jugend-Forscht (2003): *Die UFOs kommen ... – selbstregelnde, fliegende Robotikplattform*, Internetseite, URL: http://www.jugend-forscht.de/projektdatenbank/, Abruf am 19. Juni 2016.

[89] KALMAN, R. E. (1960): *A new approach to linear filtering and prediction problems*, in: Transaction of the ASME-Journal of Basic Engineering, 82, S. 35–45.

[90] KARIS, B. UND EPIC GAMES (2013): *Real Shading in Unreal Engine 4*, in: SIGGRAPH 2013 Course: Physically Based Shading in Theory and Practice.

[91] KENNEL, R. (2010): *Elektrische Antriebe - Grundlagen und Anwendungen*, Vorlesungsscript / Übungen, Lehrstuhl für Elektrische Antriebssysteme und Leistungselektronik, Technische Universität München.

[92] KOENIG, N. UND HOWARD, A. (2004): *Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator*, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2004, Bd. 3, S. 2149–2154.

[93] LÄCHELE, J., FRANCHI, A., BÜLTHOFF, H. H. UND GIORDANO, P. R. (2012): *SwarmSimX: Real-time Simulation Environment for Multi-robot Systems*, in: Proceedings of the International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR) 2012, Tsukuba, Japan.

[94] LEISHMAN, J. G. (2006): *Principles of Helicopter Aerodynamics*, Cambridge Aerospace Series, Cambridge University Press.

[95] LIEFERS, F., PARYS, R. UND SCHILLING, A. (2012): *Analysis-by-Synthesis Texture Reconstruction*, in: Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission 2012, S. 571–578.

[96] LIM, H., PARK, J., LEE, D. UND KIM, H. J. (2012): *Build Your Own Quadrotor: Open-Source Projects on Unmanned Aerial Vehicles*, in: IEEE Robotics and Automation Magazine, S. 33–45.

[97] LINKUGEL, T. (2010): *Methoden und Algorithmen zur Positions- und Ausrichtungsbestimmung eines autonomen Systems*, Master's Thesis, Hochschule für angewandte Wissenschaft und Kunst (HAWK) Göttingen.

[98] LINKUGEL, T. UND SCHILLING, A. (2013): *Another Step towards Measuring the World from the Air: Model-based 3D Real-time Simulation of Micro-UAV*, in: Photogrammetric Week '13, S. 181–191.

[99] LINKUGEL, T. UND SCHILLING, A. (2015): *Simulation Based Development of Micro Air Vehicles - One step further towards a more efficient development*, in: Proceedings of the International Micro Air Vechicle Competition and Conference (IMAV), Aachen, Germany.

[100] LINKUGEL, T., SCHILLING, A. UND MALLOT, H. A. (2012): *Modellbasierte 3D-Echtzeit-Simulation von Micro-UAS*, in: Photogrammetrie, Laserscanning, Optische 3D-Messtechnik; Beiträge der Oldenburger 3D-Tage 2012, S. 245–254.

[101] LINKUGEL, T., SCHILLING, A. UND MALLOT, H. A. (2013): *Musterbasierte optische Positionsschätzung zur Regelung eines mico-UAV – ein simulationsbasieter Ansatz*, in: Photogrammetrie, Laserscanning und Optische 3D-Messtechnik; Beiträge der Oldenburger 3D-Tage 2013, S. 344–355.

[102] LINKUGEL, T., SCHILLING, A. UND MALLOT, H. A. (2013): *Musterbasierte optische Positionsschätzung zur Regelung eines mico-UAV: ein simulationsbasieter Ansatz*, in: Allgemeine Vermessungsnachrichten (AVN), (10/2013), S. 331–341.

[103] LINKUGEL, T., SCHILLING, A. UND MALLOT, H. A. (2016): *The FireFly MAV-Framework – Closing the Gap in Micro Air Vehicle Development*, in: Photogrammetrie, Laserscanning und Optische 3D-Messtechnik; Beiträge der Oldenburger 3D-Tage 2016, S. 178–192.

[104] LOONEY, M. (2015): *The Basics of MEMS IMU/Gyroscope Alignment*, Techn. Bericht, Analog Devices.

[105] LU, M. UND YAO, Z. (2014): *New Signal Structures for BeiDou Navigation Satellite System*, Stanford's 2014 PNT Symposium.

[106] LUFTHANSA FLIGHT TRAINING (2016): *Full Flight Simulatoren*, Internetseite, URL: https://www.lufthansa-flight-training.com/training/flight/full-flight-simulators, Abruf am 18. Juni 2016.

[107] LUPASHIN, S. UND D'ANDREA, R. (2011): *Adaptive Open-Loop Aerobatic Maneuvers for Quadrocopters*, in: Proceedings of the IFAC world congress 2011.

[108] MAJDIK, A., ALBERS-SCHOENBERG, Y. UND SCARAMUZZA, D. (2013): *MAV urban localization from Google street view data*, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 2013, Tokyo, Japan, S. 3979–3986.

[109] MANSFELD, W. (2013): *Satellitenortung und Navigation: Grundlagen und Anwendung globaler Satellitennavigationssysteme*, Vieweg+Teubner Verlag.

[110] MARTIN, S., BANGE, J. UND BEYRICH, F. (2011): *Meteorological profiling of the lower troposphere using the research UAV "M^{2}AV Carolo"*, in: Atmospheric Measurement Techniques (AMT), 4(4), S. 705–716.

[111] MATLAB FILE EXCHANGE, YURY (2009): *Ellipsoid fit*, Internetseite, URL: http://de.mathworks.com/matlabcentral/fileexchange/24693-ellipsoid-fit, Abruf am 08. Apr. 2016.

[112] MCLEAN, D. (1993): *A short History of Flight Control*, in: European Control Conference, 3, S. 1240 – 1245.

[113] MEASUREMENT SPECIALTIES (2012): *MS5611-01BA03 - Barometric Pressure Sensor*, Rev. 1.

[114] MEIER, L., TANSKANEN, P., HENG, L., LEE, G. H., FRAUNDORFER, F. UND POLLEFEYS, M. (2012): *PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision*, in: Autonomous Robots, 33, S. 21–39.

[115] MEISTER, O. (2010): *Entwurf und Realisierung einer Aufklärungsplattform auf Basis eines unbemannten Minihelikopters mit autonomen Flugfähigkeiten*, Dissertation, Karlsruher Institut für Technologie (KIT).

[116] MELLINGER, D., MICHAEL, N. UND KUMAR, V. (2012): *Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors*, in: International Journal of Robotics Research, 31(5), S. 664–674.

[117] MEYER, J., SENDOBRY, A., KOHLBRECHER, S., KLINGAUF, U. UND VON STRYK, O. (2012): *Comprehensive Simulation of Quadrotor UAVs using ROS and Gazebo*, in: Proceedings of the International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR) 2012, S. 400–411.

[118] MICHEL, O. (2004): *Webots: Professional Mobile Robot Simulation*, in: Journal of Advanced Robotics Systems, 1(1), S. 39–42.

[119] MICRODRONES (2016): *Microdrones - Homepage*, Internetseite, URL: http://www.microdrones.de, Abruf am 26. Juni 2016.

[120] MUELLER, M. UND D'ANDREA, R. (2014): *Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers*, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2014, S. 45–52.

[121] MUELLER, M., HEHN, M. UND D'ANDREA, R. (2013): *A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification*, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2013, S. 3480–3486.

[122] NATIONAL GEOGRAPHIC - RODGERS, A. I. (2013): *National Geographic, Education.: Compass*, Internetseite, URL: http://education.nationalgeographic.org/encyclopedia/compass/, Abruf am 11. Febr. 2016.

[123] National Oceanic and Atmospheric Administration (NOAA) (2015): *World Magnetic Model 2015*, Internetseite, URL: http://www.ngdc.noaa.gov/geomag/WMM/data/WMM2015, Abruf am 02. Sept. 2016.

[124] Neumann, P. P. (2013): *Gas Source Localization and Gas Distribution Mapping with a Micro-Drone*, Dissertation, Freie Universität Berlin.

[125] Niethammer, U. (2013): *UAV-basierte Fernerkundungsmethoden der Aerogeophysik für die hochauflösende Beobachtung von alpinen Rutschhängen*, Dissertation, Universität Stuttgart.

[126] Northrop Grumman Corporation (2015): *Northrop Grumman - Homepage*, Internetseite, URL: http://www.northropgrumman.com, Abruf am 08. Dez. 2015.

[127] NVIDIA Corporation (2016): *NVIDIA Jetson TX1*, Internetseite, URL: http://www.nvidia.com/object/jetson-tx1-module.html, Abruf am 03. Okt. 2016.

[128] NVIDIA Corporation (2017): *NVIDIA PhysX*, Internetseite, URL: http://www.nvidia.de/object/nvidia-physx-de.html, Abruf am 13. März 2017.

[129] NXP Semiconductors (2011): *LPC2131/32/34/36/38 - Single-chip 16/32-bit microcontrollers*, Datenblatt, Rev. 5.1.

[130] NXP Semiconductors (2015): *MPL3115A2 - I2C precision pressure sensor with altimetry*, Rev. 4.0.

[131] OpenPilot (2016): *OpenPilot Ground Control Station (GCS)*, Internetseite, URL: www.openpilot.org, Abruf am 22. Mai 2015.

[132] Parrot SA (2014): *Parrot MiniDrones Rolling Spider*, Datenblatt.

[133] Parrot SA (2016): *Parrot S.L.A.M. dunk*, Internetseite, URL: https://corporate.parrot.com/en/pressrelease/parrots.l.a.m.dunkturnadroneintoasmartrobot, Abruf am 03. Okt. 2016.

[134] Parrot SA (2017): *Parrot AR.Drone 2.0*, Internetseite, URL: https://www.parrot.com/us/drones/parrot-ardrone-20-elite-edition#reviews, Abruf am 24. März 2017.

[135] Physikalisch-Technische Bundesanstalt (PTB), Braunschweig (2007): *Local gravity*, Internetseite, URL: http://www.ptb.de/cartoweb3/SISproject.php, Abruf am 23. März 2016.

[136] PNI Sensor Corporation (2005): *MicroMag3 - 3-Axis Magnetic Sensor Module*, Datenblatt, Rev. 1.

[137] PNI Sensor Corporation (2012): *RM3100 - Geomagnetic Sensor*, Datenblatt, Rev. 5.1.

[138] PX4 (2016): *Software in the Loop (SITL) Simulation*, Internetseite, URL: http://dev.px4.io/simulation-sitl.html, Abruf am 16. Okt. 2016.

[139] Qioptiq Group (2016): *Die LINOS Positioniersysteme* , Produktinformation.

[140] Qualcomm Technologies, Inc. (2016): *Snapdragon Flight*, Internetseite, URL: https://developer.qualcomm.com/hardware/snapdragon-flight, Abruf am 03. Okt. 2016.

[141] Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R. und Ng, A. Y. (2009): *ROS: an open-source Robot Operating System*, in: ICRA Workshop on Open Source Software.

[142] Rathmann, U. und Wilgen, J. (2016): *Qwt - Qt Widgets for Technical Applications*, Internetseite, URL: http://qwt.sourceforge.net/, Abruf am 17. Nov. 2016.

[143] Raytheon Anschütz (2015): *A Pioneer in Navigation Technology*, Internetseite, URL: http://www.raytheon-anschuetz.com/company/history/, Abruf am 08. Dez. 2015.

[144] Roberts, P. H. und Glatzmaier, G. A. (2000): *Geodynamo theory and simulations*, in: Reviews of Modern Physics, 72, S. 1081–1123.

[145] ROS (2016): *Robot Operating System (ROS)*, Internetseite, URL: http://www.ros.org/, Abruf am 18. Nov. 2016.

[146] Schmidt, S. F. (1981): *The Kalman Filter: Its Recognition and Development for Aerospace Applications*, in: Journal of Guidance, Control, and Dynamics, 4(1), S. 4–7.

[147] Shah, S., Dey, D., Lovett, C. und Kapoor, A. (2017): *Aerial Informatics and Robotics Platform*, Techn. Bericht MSR-TR-2017-9, Microsoft Research.

[148] Silicon Sensing Systems Limited (2015): *The Gyro 100 years on*, Internetseite, URL: http://www.siliconsensing.com/technology/the-gyro-100-years-on/, Abruf am 08. Dez. 2015.

[149] Sperry, E. A. (1922): *Automatic pilot for aeroplanes*, Patent, US 1418335 A.

[150] Sperry, L. B. (1919): *Mechanical pilot for aeroplanes*, Patent, US 1324134 A.

[151] Sperry Product Innovation (2015): *History*, Internetseite, URL: http://sperryinc.com/who-we-are/history/, Abruf am 08. Dez. 2015.

[152] Spiegel Online (2015): *Drohnen-Crash bei Ski alpin: „Ein Wahnsinn, was da passiert ist“.* , Internetseite, URL: http://www.spiegel.de/sport/wintersport, Abruf am 08. März 2016.

[153] STMicroelectronics (2013): *LSM9DS0 - iNEMO inertial module*, Datenblatt, Rev. 2.

[154] STMicroelectronics (2016): *STM32F407xx - ARM Cortex-M4 32bit MCU+FPU*, Datenblatt, Rev. 7.

[155] Stockwell, W. (2003): *Angle Random Walk*, Techn. Bericht, Crossbow Technology Inc.

[156] Stumpf, A., Niethhammer, U., Rothmund, S., Mathieu, A., Malet, J., Kerle, N. und Joswig, M. (2013): *Advanced Image Analysis for Automated Mapping of Landslide Surface Fissures*, Landslide Science and Practice, S. 357–363, Springer Berlin Heidelberg.

[157] Terabee (2016): *TeraRanger One*, Datenblatt.

[158] The MathWorks, Inc. (2016): *Matlab Aerospace Blockset - Wind*, Internetseite, URL: http://de.mathworks.com/help/aeroblks/wind.html, Abruf am 26. Juni 2016.

[159] The MathWorks, Inc. (2016): *Matlab/Simulink R2016b*, Internetseite, URL: https://de.mathworks.com/, Abruf am 22. Jan. 2017.

[160] The Washington Post - Fung, B. (2015): *The future of train safety lies in drones*, Internetseite, URL: https://www.washingtonpost.com/news/the-switch/wp/2015/05/13/how-drones-could-make-train-travel-safer/, Abruf am 19. Juni 2016.

[161] Titterton, D. H. und Weston, J. L. (2004): *Strapdown Inertial Navigation Technology*, Progress in astronautics and aeronautics, American Institute of Aeronautics and Astronautics Stevenage, U.K. Institution of Electrical Engineers, Reston, VA.

[162] Tomayko, J. E. und US NASA History Office (2000): *Computers take flight: a history of NASA's pioneering digital fly-by-wire project*, The NASA history series, NASA.

[163] Trainer, M. (2008): *Albert Einstein's expert opinions on the Sperry vs. Anschütz gyrocompass patent dispute*, in: World Patent Information, 30(4), S. 320 – 325.

[164] Truong, T. V. A., Hattenberger, G. und Ronfle-Nadaud, C. (2013): *The cooperation between Unmanned Aerial Vehicles using a mission planner.*, in: Proceedings of the Industrial Informatics (INDIN) 2013, S. 797–803, IEEE.

[165] u-blox AG (2016): *MAX-M8 concurrent GNSS modules*, Datenblatt, Rev. 1.

[166] US Department of Defense Release (1997): *DARPA selects Micro Air Vehicle Contractor*, Internetseite, URL: http://fas.org/irp/news/1997/b12121997_bt676-97.html, Abruf am 24. Febr. 2016.

[167] VectorNav Technologies (2016): *VectorNav Support Library*, Internetseite, URL: http://www.vectornav.com/support/library/, Abruf am 24. Febr. 2016.

[168] Walden, J. (2016): *Intel Acquires Ascending Technologies*, Internetseite, URL: http://blogs.intel.com/technology/2016/01/intel-acquires-asctec/, Abruf am 03. Okt. 2016.

[169] Watts, A. C., Ambrosia, V. G. und Hinkley, E. A. (2012): *Unmanned Aircraft Systems in Remote Sensing and Scientific Research: Classification and Considerations of Use*, in: Remote Sensing, 4(6), S. 1671 – 1692.

[170] Weiss, S. M. (2012): *Vision Based Navigation for Micro Helicopters*, Dissertation, ETH Zürich.

[171] Welch, G. und Bishop, G. (2001): *An introduction to the Kalman Filter*, Techn. Bericht, University of North Carolina at Chapell Hill.

[172] Wendel, J. (2007): *Integrierte Navigationssysteme: Sensordatenfusion, GPS und Inertiale Navigation*, Oldenbourg Wissenschaftsverlag.

[173] Wiggerich, B. (2011): *Fluggerät*, Patent, DE 102009033821 A1.

[174] Wiggerich Warenwirtschaft - Video Inspektionssysteme (WIWA-VI) (2016): *WIWA-VI - Homepage*, Internetseite, URL: http://wiwa-vi.de/, Abruf am 24. Febr. 2016.

[175] Wildmann, N., Hofsäss, M., Weimer, F., Joos, A. und Bange, J. (2014): *MASC - a small Remotely Piloted Aircraft (RPA) for wind energy research*, in: Advances in Science and Research, 11, S. 55–61.

[176] Wilson, S. B. (2002): *SBIR Success Story for DARPA Tech 2002 AeroVironment Inc. Program Overview*, DARPA Tech 2002.

[177] Woodman, O. J. (2007): *An introduction to inertial navigation*, Techn. Bericht, University of Cambridge, Cambridge, UK.

[178] X-Plane (2016): *X-Plane*, Internetseite, URL: http://www.x-plane.com/, Abruf am 16. Okt. 2016.

[179] Yang, S., Scherer, S. A. und Zell, A. (2014): *Robust Onboard Visual SLAM for Autonomous MAVs*, in: Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13), S. 361–373, Padova, Italy.

[180] Zogg, J.-M. (2011): *GPS und GNSS: Grundlagen der Ortung und Navigation mit Satelliten*, User Guide, ublox.

[181] Zufferey, J.-C., Guanella, A., Beyeler, A. und Floreano, D. (2006): *Flying over the Reality Gap: From Simulated to Real Indoor Airships*, in: Autonomous Robots, 21(3), S. 243–254.