Julius-Maximilians-Universität Würzburg
Fakultät für Mathematik und Informatik

Informationstechnik für Luft- und Raumfahrt
Lehrstuhl für Informatik 8 Prof. Dr. Sergio Montenegro

Bachelorarbeit

Entwicklung und Evaluierung eines Systems zur Bestimmung der Orientierung und Position eines Objektes durch inertiale und magnetische Sensoren

Vorgelegt von

Lukas Iffländer

Matr.-Nr.: 1717856

Prüfer: Prof. Dr. Sergio Montenegro

Betreuender wissenschaftliche Mitarbeiter: Dipl.-Ing. Nils Gageik

Würzburg, 07. 12. 2012

Erklärung

Ich versichere, dass ich die vorliegende Arbeit einschließlich aller beigefügter Materialien selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.
Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Werken entnommen sind, sind in jedem Einzelfall unter Angabe der Quelle deutlich als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form noch nicht als Prüfungsarbeit eingereicht worden.
Mir ist bekannt, dass Zuwiderhandlungen gegen diese Erklärung und bewusste Täuschungen die Benotung der Arbeit mit der Note 5.0 zur Folge haben kann.

Würzburg, 07. 12. 2012

Lukas Iffländer

Aufgabenstellung

Implementierung einer Datenfusion von Inertial- und Magnetsensor für einen Quadrocopter

Die Fortschritte im Bereich Sensorik und Mikrotechnik ermöglichen heutzutage den kostengünstigen Bau kleiner unbemannter Luftfahrzeuge (UAV, unmanned airial vehicle, Drohne) wie Quadrocopter. Die Forschung und Entwicklung dieser Systeme wurde in den letzten Jahren aufgrund der vielfältigen Anwendungsmöglichkeiten stark vorangetrieben. Wenngleich im Bereich UAV viel geforscht wurde, ist das Thema Autonomes Flugobjekt längst noch nicht vollständig behandelt. Insbesondere der Indoor-Betrieb ist aufgrund fehlender absoluter Positionsstützung durch GPS problematisch. Der Aufbau eines eigenen autonomen Systems wird daher am Lehrstuhl Aerospace Information Technology der Uni Würzburg erforscht und erprobt. Im Rahmen dieses Forschungsvorhabens ist ein System zur Datenfusion von Beschleunigungs-, Drehraten- und Magnetsensoren zu entwickeln.
Hauptaugenmerk ist die Entwicklung eines Algorithmus zur Datenfusion, um die Lage im dreidimensionalen Raum zu bestimmen. Der entwickelte Algorithmus soll in Echtzeit ausgeführt und in das bestehende System integriert werden, woran er zu evaluieren ist. Zur Aufgabe gehört eine ausführliche Übersicht bestehender Lösungen aus dem Stand der Technik, auf die aufgebaut werden soll.
Aufgabenstellung (Stichpunktartig):

- Überblick Stand der Technik - Datenfusion - Lagebestimmung
- Implementierung Datenfusion: Beschleunigungssensor, Gyroskop, Magnetfeldsensor
- Positionsbestimmung (optional)
- Integration in bestehendes System
- Evaluierung am fliegenden System
- Dokumentation

Zusammenfassung

In dieser Arbeit wird ein System entwickelt, implementiert und evaluiert, das eine möglichst präzise Orientierungserfassung auf Basis von Inertial- und Magnetsensoren ermöglicht. Zur Orientierungsbestimmung wird über die Sensorwerte eines Drei-Achsen-Gyroskops integriert, welches durch ein Drei-Achsen-Magnetometer sowie durch 3D-Accelerometer gestützt wird. Zur Reduzierung von Fehlern wird ein erweiterter Kalmanfilter (EKF) implementiert. Das System soll nach seiner Fertigstellung in den Quadrocopterprojekten der Universität Würzburg zur Anwendung kommen, aber auch andere Anwendungsgebiete wie z.B. die Positionierung von Industrierobotern oder die Verfolgung von Bewegungen im medizinischen Einsatz sind denkbar. Um die Kompatibilität mit dem Quadrocoptersystem zu gewährleisten, wird das Konzept in der Programmiersprache C für die AVR32-Architektur implementiert. Erwartet wird, dass zumindest kurzfristig (z.B. beim Ausfall der absoluten Sensoren) auf das Sensorsystem als alleinige Stützung der Navigation des Quadrocopters zurückgegriffen werden kann.
Die Evaluierungsergebnisse der Arbeit zeigen die grundsätzliche Eignung des vorgestellten Systems zur Orientierungsbestimmung. Erfüllt die gestellten Ansprüche an Reaktionszeit und Genauigkeit. Das System ist noch fehleranfällig. In der abschließenden Diskussion werden Lösungsansätze dieses Problems vorgestellt und es wird auf Möglichkeiten der Weiterentwicklung eingegangen.

Inhaltsverzeichnis

1. Einleitung

In den letzten Jahren wurden vermehrt Anwendungen für mobile Roboter entwickelt. Besonders Flugdrohnen, wie zum Beispiel Quadrocopter, erfreuen sich in der Wissenschaft, sowohl im zivilen, als auch im militärischen Bereich, steigender Beliebtheit. Für diese Systeme ist es essenziell, dass sie ihre Ausrichtung exakt bestimmen können, um ihre Aufgaben zuverlässig ausführen zu können. Bisher wird diese Orientierungsbestimmung meist über, von externen Faktoren abhängigen, Erfassungssystemen erreicht. Dabei werden sowohl externe Systeme wie zum Beispiel optische Trackingsysteme eingesetzt. Aber auch onboard Sensoren, die den Abstand zur Umgebung auf Basis von Infrarot-, Ultraschall- oder Lasertechnik messen und dann anhand einer bekannten Karte die Orientierung erkennen, werden genutzt. Diese Systeme haben ihre Schwachpunkte. So müssen die externen Systeme erst vorinstalliert werden und es muss eine ausreichende Signalqualität gegeben sein, um genaue Werte zu erhalten. Die onboard Sensoren verzeichnen Leistungseinbußen, sobald andere Systeme auf ihrer Frequenz agieren. Auch können beispielsweise Wärmequellen oder Rauchpartikel die Arbeitsweise beeinträchtigen. Um bei einer Nichtverfügbarkeit bzw. einem Versagen der obengenannten Systeme weiterhin die aktuelle Ausrichtung bestimmen zu können, muss folglich ein, von äußeren Einflüssen weitestgehend unabhängiges, Verfahren entwickelt werden. Dazu wird auf inertiale Systeme zurückgegriffen. Diese bestehen aus Gyroskopen, die die Drehrate liefern, sowie aus Accelerometern, die Beschleunigungswerte zur Verfügung stellen. Als weitere Stützung bieten sich Magnetometer an, falls die Umgebung nicht mit starken Störfeldern belastet ist. Bisher wurden für den Anwendungszweck der Orientierung meist sehr teure Sensoren eingesetzt. Aktuell wird daran geforscht, mit preiswerteren Sensoren unter Einsatz von Filtersystemen eine ähnliche Qualität zu erreichen. Damit beschäftigt sich auch die vorliegende Arbeit.

Die Orientierung wird dabei aus der einfachen Integration der Drehrate generiert. Durch die Integrationen kummulieren sich selbst kleinere Fehler in kürzester Zeit. Um dieses Driften des

Systems über die Zeit zu minimieren, müssen die Messwerte auf Plausibilität gefiltert werden. Auch eine Stützung durch andere Sensoren, die nicht von solch einem Drift betroffen sind, ist hilfreich.

Ein weiteres Problem ist die fehlende Kommutativität der Rotationen. Das heißt, werden die Drehungen um die einzelnen Achsen nacheinander verrechnet, tritt ein Orientierungsfehler auf.

An den beiden Punkten Nichtkommutativität und Filterung soll die vorliegende Arbeit ansetzen. Die Wahl der Filterungsmethode fiel auf einen Kalmanfilter, die Wahl der detaillierten Implementierung wird im weiteren Verlauf der Arbeit entschieden. Als Stützsensorik zur Orientierungsbestimmung wurde ein Magnetometer gewählt, außerdem wird bei Stillstand $(a \approx g)$ das 3D-Accelerometer zur Orientierungsbestimmung ausgewertet. Die Entscheidung für den Kalmanfilter fiel, da es sich bei diesem um ein erprobtes Konzept handelt, das mit der Wahrscheinlichkeitsgewichtung der Fehler genau das erwünschte Ergebnis ermöglicht. Ein besonderes Problem der Orientierungsbestimmung ist die Nichtlinearität des Systems. Da um bis zu drei Achsen rotiert wird, ist die Orientierung nicht in einem linearen Gleichunssystem darstellbar. Hierfür ist im Rahmen dieser Arbeit eine Lösung zu finden. Für die Sensorenwahl war entscheidend, dass keine externen Systeme vorhanden sind. Die Sensoren sind somit ortsunabhängig einsetzbar. Eine weitere Anforderung an die vorliegende Arbeit ist es, das System auf der Hardware, einem Atmel AVR Microprozessor, zu implementieren. Dabei muss das System in Echtzeit arbeiten. Die Zeit pro Durchgang muss also so kurz wie möglich gehalten werden.

Im Folgenden wird in der Arbeit zuerst tiefergehend auf die Problemstellung eingegangen. Anschließend folgen die Untersuchung unterschiedlicher Lösungsansätze zur Positions- und Orientierungsbestimmung, die Entscheidung für eine dieser Lösungen, die Aufstellung des detaillierten mathematischen Modells sowie die Implementierung der Orientierungsbestimmung. Im Anschluss wird dann anhand von Experimenten die Qualität der erarbeiteten Lösung bestimmt und die Ergebnisse im Anschluss diskutiert.

2. Stand der Wissenschaft

2.1. Inertialnavigation

Inertialnavigationssysteme, im Folgenden als INS (Inertial Navigation System) bezeichnet, dienen der Messung der Bewegung frei beweglicher Körper. Dabei werden sechs Freiheitsgrade betrachtet. Jeweils drei davon sind rotatorisch (Rotationen) bzw. translatorisch (Bewegung im Raum). Sie sind an drei orthogonalen Achsen ausgerichtet.
Der Hauptgrund für den Einsatz eines INS liegt in dessen weitestgehenden Unabhängigkeit von externen Faktoren bzw. Systemen und der damit einhergehenden hohen Ausfallsicherheit im Vergleich zu absoluten Systemen wie GPS oder Laser-Tracking.
Für ein INS sind folgende Voraussetzungen zu schaffen:

- Sensorik
- System zur Orientierungs- und Positionsdarstellung
- System zur Berechnung des Systemzustands aus den Sensorwerten

2.2. Sensorik

Um die Bewegungen entlang der oben genannten Freiheitsgrade (vgl. 2.1) messen zu können, bedarf es entsprechender Sensorik. Dazu werden im Folgenden zuerst die reinen Inertialsensoren (Gyroskop 2.2.1 und Accelerometer 2.2.2) und im Anschluss das Magnetometer (2.2.3) als umgebungsabhängiges Stützsystem vorgestellt.

2.2.1. Gyroskop

Ein Gyroskop beruht auf dem Prinzip des Drehmoments bzw. der Corioliskraft. Diese Scheinkraft tritt auf, wenn eine Masse innerhalb eines rotierenden Bezugssystems nicht ruht. Sie ist senkrecht zur Bewegungsrichtung des Körpers und der Rotationsachse des Bezugssystems angeordnet. Ihr Betrag ist dabei proportional zur Masse und Geschwindigkeit des Körpers sowie der Winkelgeschwindigket des Bezugssystems.
Die ursprünglichen mechanischen Gyroskope bestehen aus einem rotiernden, symmetrischen Kreisel mit hoher Drehrate. Nach den Stabilitätsgesetzen versucht dieser Kreisel die Richtung seiner Drehachse beizubehalten. Wirkt nun eine Kraft auf die Achsenrichtung des Kreisels, so weicht die Kreiselachse rechtwinklig zu dieser aus. Diese, Präzession genannte, Reaktion wird nun zur Messung der Lageregelung herangezogen[13]. Zur dreidimensionalen Messung werden drei Gyroskope senkrecht zueinander angeordnet.

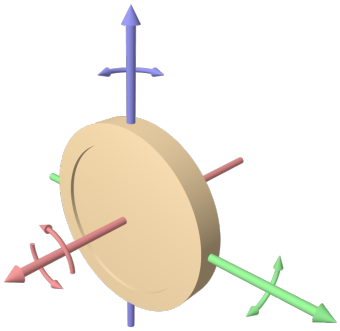

Abbildung 2.1.: Darstellung eines Gyroskopkreisels mit Drehachse (rot), Eingangskraft (grün) und Reaktion (blau) [13]

In dieser Arbeit wird ein MEMS-Gyroskop (Micro Electro-Mechanical System) [13] verwendet. Diese Entscheidung fällt aufgrund der geringen Masse und Abmessungen dieses Sensortyps.
Beim MEMS handelt es sich um einen stabförmigen Quader. Dieser wird in einer Ebene zur Schwingung angeregt. Dreht der Stab sich um seine z-Achse , induziert der Coriolis-Effekt ein Moment senkrecht zur anregenden Schwingung. Dies führt bei entsprechender Stab-Geometrie zu einer messbaren Schwingung in dieser Ebene, deren Amplitude ein Maß für die Drehrate ω ist[2].
Sensoren dieser Klasse erreichen nach einigen Jahren technischer Entwicklung inzwischen Fehler $< 10°/h$ [2] und erfüllen damit auch die grundsätzlichen Bedingungen, um zur Orientierungsbe-

stimmung eingesetzt werden zu können. Es ist allerdings eine starke Temperaturabhängigkeit zu berücksichtigen [7, Kap. 4]. .

2.2.2. Accelerometer

Bei den Accelerometern handelt es sich um Sensoren, welche Beschleunigungen messen. Dazu wird meist auf Masse-Feder-Systeme zurückgegriffen.

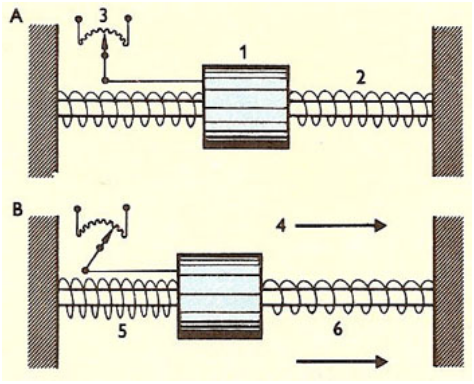

Abbildung 2.2.: Funktionsweise eines Accelerometers[1]

Auch hier wird, mit der selben Begründung wie beim Gyroskop, zu MEMS-Accelerometern gegriffen. Diese Sensoren sind Feder-Masse-Systeme, mit nur wenige μm breiten Siliziumstegen als Federn. Auch die Masse wird aus Silizium produziert. Durch die Auslenkung beim Beschleunigen kann eine Änderung der elektrischen Kapazität gemessen werden. [13] Beschleunigungssensoren dieses Typs leiden häufig, zusätzlich zu einem fertigungsbedingten konstanten Offset, an einem Drift, der nach jeder Bewegung variiert.

2.2.3. Magnetometer

Bei Magnetometern handelt es sich um Sensoren zur Messung einer magnetischen Feldstärke. Magnetometer können grundsätzlich in zwei Gruppen aufgeteilt werden [13]:

- Skalar-Magnetometer messen die Gesamtstärke des magnetischen Feldes, dem sie ausgesetzt sind.
- Vector-Magnetometer messen die Komponente des magnetischen Feldes in einer bestimmten Richtung.

Aufgrund des Einsatzzweckes der Orientierungsbestimmung fällt die Auswahl auf Vektor-Magnetometer. Um eine dreidimensionale Orientierungserfassung zu ermöglichen, sind drei orthogonal ausgerichtete Sensoren erforderlich.
Aus obigen Gründen wird auf ein MEMS-System zurück gegriffen. Dabei gibt es unterschiedliche Arten der Implementierung auf dem Microsystem (hier nur ein paar Beispiele) [13]:

- Halleffekt-Sensoren
- Magnetdioden
- Lorentzkraft-Sensoren
- Fluxgate-Magnetometer

Beim genutzten Sensor wird vom Hersteller keine Information über die Implementierung gegeben, es ist aber zu vermuten, dass es sich um einen Fluxgate-Magnetometer handelt, da dies der am häufigsten eingesetzte Typ ist [7, Kap. 4]. Dieser Typ liefert eine Richtungs-Messgenauigkeit $< 1°$, unterliegen aber einer hohen Temperaturabhängigkeit [7, Kap. 4] was zu einem starken Offset führen kann.
Zur Orientierungsberechnung wird aus dem Magnetfeldvektor die Orientierung im Erdmagnetfeldvektor errechnet.
Die Hauptfehlerquelle ist der Einfluss durch externe Magnetfelder. Diese können die Messung massiv verfälschen. Solcher Felder können sowohl passiver Herkunft (Metalle) als auch aktiver Herkunft (Ströme in den Leitungen, Motoren des Quadrocopters) sein. Daher ist zunächst ein Kompensationsvektor im Stillstand zu bestimmen.

2.3. Positions und Orientierungsbestimmung

Um die Bewegung entlang der oben genannten Freiheitsgrade (vgl. 2.1) und damit auch Orientierung und Position aus den Werten der in 2.2 eingeführten Sensoren bestimmen zu können, sind zuerst einige Rechenschritte nötig.

2.3.1. Orientierungsbestimmung

Für die Orientierungsbestimmung werden die Messwerte von Accelerometer, Magnetometer und Gyroskop genutzt.

Bei Accelerometer und Magnetometer wird der gemessene Wert direkt genutzt. Hier entstehen keine historiebedingten Fehler. Die gemessenen Werte müssen also nur auf ihre Eignung für die Orientierungsbestimmung geprüft werden. Das heißt, der Betrag der Beschleunigung darf nicht zu weit von der Erdbeschleunigung abweichen. Selbiges gilt für den Betrag der Magnetfeldstärke und das Erdmagnetfeld.

Um aus der vom Gyroskop errechneten Drehrate die Orientierung zu errechnen, muss über die Drehrate integriert werden.

$$\Omega(t) = \int_0^t \omega(\tau)\, d\tau$$

Dabei akkumuliert sich auch der Fehler linear, es entsteht ein sog. Drift [3][9][10]. Dies ist in der untenstehenden Grafik dargestellt.

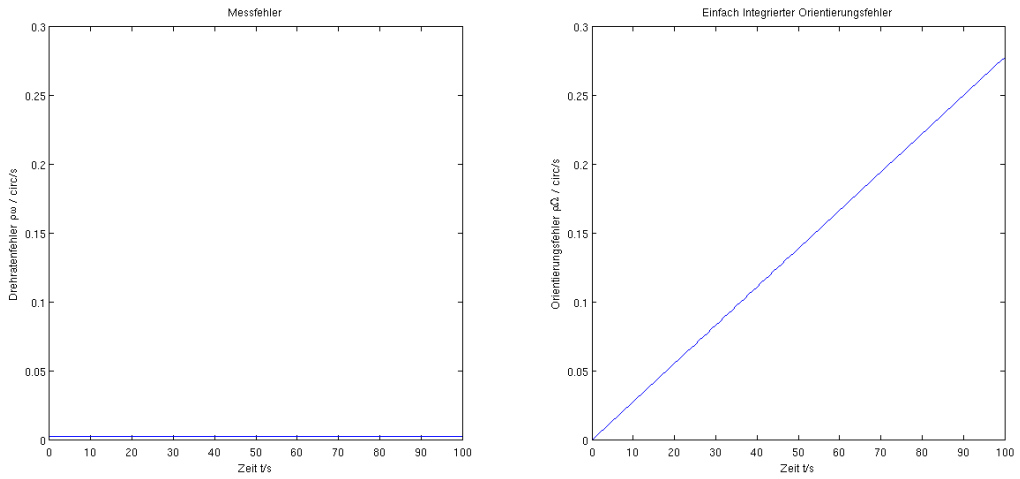

Abbildung 2.3.: Kumulation des Fehlers bei einfacher Integration

Ist die Abweichung im eindimensionalen Raum noch akzeptabel, so müssen bei der dreidimensionalen Betrachtung schon ordentliche Qualitätsabstriche hingenommen werden, da die Drehungen

voneinander abhängig sind. Dies liegt daran, dass die Matrizenmultiplikation nicht kommutativ ist. Vor allem im Zusammenhang mit der Positionsbestimmung wirkt sich dieser Fehler aus da die Bestimmung der Bewegungsrichtung auf die Ausrichtung aufbaut.
Bei der Ausweitung auf die dreidimensionale Betrachtung muss diese Nichtkommutativität der Rotationen beachtet werden. Dazu werden in 2.5 mehrere Ansätze vorgestellt.

2.3.2. Positionsbestimmung

Zur Positionsbestimmung werden Accelerometer verwendet.
In einem ersten Schritt wird die relative Beschleunigung errechnet. Dabei ist es wichtig auch die Gravitation zu beachten. Das heißt, dass $\vec{g}$ von $\vec{a}$ abegzogen werden muss. Dabei muss die Ausrichtung des Systems berücksichtig werden. $\vec{g}$ muss also vom externen System in das Körpersystem tranformiert werden.
Aus der errechneten Beschleunigung wird durch Integration die Geschwindigkeit errechnet:

$$v\left(t\right) = \int_0^t a\left(\tau\right) d\tau$$

Durch einen zweiten Integrationsdurchgang wird anschließend die Position bestimmt:

$$x\left(t\right) = \int_0^t v\left(\tau\right) d\tau = \int_0^t \int_0^t a\left(\tau\right) d\tau \; d\theta$$

Wegen der doppelten Integration kummuliert sich der Positionsfehler quadratisch. Wie in der folgenden Abbildung zu sehen ist, wird bereits bei einem Messfehler von $0.01 \frac{\mathrm{m}}{\mathrm{s}^2}$ nach 20 Sekunden ein Positionsfehler von 2 Metern erreicht:

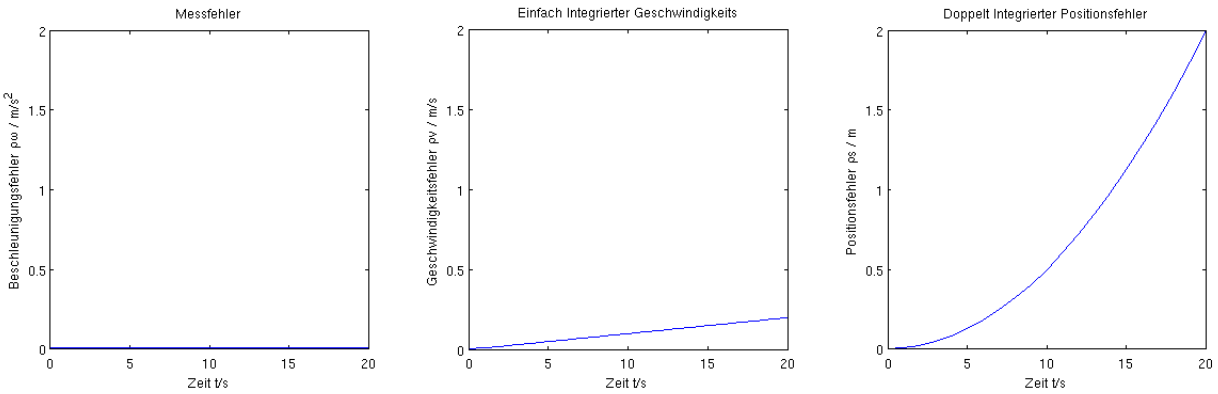

Abbildung 2.4.: Kummulation des Fehlers bei doppelter Integration

Dieses schnelle Ansteigen des Fehlers macht bereits im Eindimensionalen die Positionsbestimmung nach wenigen Sekunden wertlos. Im Dreidimensionalen kann man davon ausgehen, dass die Positionserfassung nach etwa einer Sekunde unbrauchbar wird. Insbesondere wenn noch der Orientierungsfehler eines nicht gefilterten Gyroskops dazu käme. Aus diesem Grund ist eine Filterung der Sensorwerte zwingend nötig um eine nutzbare Position zu erhalten. Auch hier wird auf einen Kalmanfilter zurück gegriffen. Es liegen liegen für die Beschleunigungsmessung keine alternativen inertialen Stützsensoren vor. Zur Steigerung der Messgenauigkeit kann auf zusätzliche parallel angeordnete Beschleunigungssensor zurückgegriffen werden, um durch die Fehlerreduzierung durch Redundanz den Standardfehler zu minimieren.
In einem zweiten Schritt wird nun die gemessene Positionsänderung rotiert und dadurch an die aktuelle Orientierung angepasst. Anschließend wird sie auf den vorherigen Positionswert aufaddiert.
Die Orientierungsbestimmung wirkt hier zweimal als potentielle Fehlerquelle. Sie wird sowohl bei der Berechnung der Beschleunigung als auch bei der Aufsummierung der Positionsänderung auf die vorherige Position genutzt.

2.4. Kalmanfilterung

Ein Kalmanfilter ermöglicht es, bei fehlerhaften Beobachtungswerten Rückschlüsse auf den wahren Zustand zu ziehen.
Im Gegensatz zu den einfacheren Filtertypen basiert der Kalmanfilter auf einer zeitdiskreten Zustandsraummodellierung (siehe [8]) eines linearen Systems.[7, Kap. 2]. Im Zustandsraummodell

wird die Veränderung des Zustandsvektors $\vec{x}$ mit der Eingangsgröße u und der Ausgangsgröße y modelliert. Für das Modell ergeben sich dann folgende Gleichungen:

$$\dot{\vec{x}} = A \cdot \vec{x} + \vec{b} \cdot \vec{u}$$
$$\vec{y} = \vec{c}^T \cdot x + d \cdot \vec{u}$$

mit der Systemmatrix A, dem Eingangsvektor b, dem transponierten Ausgangsvektor c und dem Durchgangsfaktor d.

2.4.1. Arbeitsweise

Ein Kalmanfilter arbeitet auf diesem System in zwei Schritten.

2.4.1.1. Prädikation

Bei diesem ersten Schritt wird aus dem, im letzten Durchgang geschätzten, Zustand, zum aktuellen Zeitpunkt eine Vorhersage des Zustands $\vec{x_k^-}$ sowie dessen Kovarianz P_k^-, errechnet.
Der Index $_k^-$ gibt dabei an, dass der Wert zum Zeitpunkt k auf Basis der Werte zum Zeitpunkt $k-1$ errechnet wird.

$$\vec{x}_{k+1}^- = \Phi_k \cdot \vec{x}_k + b \cdot u + w$$
$$P_{k+1}^- = \Phi P_k \Phi^T + Q_k$$

Wobei es sich bei Q um die Rauschkovarianzmatrix handelt.

2.4.1.2. Korrektur

Die zuvor geschätzten Vorhersagen werden anschließend mit den Informationen des aktuellen Messwertes korrigiert. Dazu werden die folgenden Hilfsgrößen errechnet:

Innovation $\tilde{y}_k$ gibt an, wie genau der aktuelle Messwert, mit Hilfe der Beobachtungsgleichung, beschreibbar ist.

Die nötige Korrektur kann demnach als zur Innovation direkt proportional angesehen werden.

Residualkovarianz S_k wird durch die aktuellen Beobachtungswerte, der Kovarianz aus der Prädikation und den Messunsicherheiten der Messwerte bestimmt.

Kalman-Matrix K_k gewichtet die Eingabewerte anhand ihrer Unsicherheiten (d.h unsichere Werte werden weniger gewichtet als sichere.)

Anschließend wird mit den Hilfswerten der korrigierte Zustandsvektor $\vec{x}_k$ und die zugehörige Kovarianzmatrix P_k errechnet.

$$
\begin{aligned}
K_{k+1} &= P^-_{k+1} F^T_{k+1} \left(F_{k+1} P^-_{k+1} F^T_{k+1} + R_{k+1} \right)^{-1} \\
\vec{x}_{k+1} &= \vec{x}^-_{k+1} + K_{k+1} \left[\vec{z}_{k+1} - F_k \vec{x}^-_{k+1} \right] \\
P_{k+1} &= P^-_{k+1} - K_{k+1} F_{k+1} P^-_{k+1}
\end{aligned}
$$

mit der Messkovarianzmatrix R und der Beobachtungsmatrix F.

2.4.2. Modellierungsvarianten

Grundsätzlich existieren zwei Basisvarianten, wie das System und der Kalmanfilter modelliert werden:

2.4.2.1. Direkte Modellierung

Hierbei wird direkt das System modelliert. Das heißt der Kalmanfilter wird der Berechnung vorgeschaltet, um im Voraus die Fehler zu eliminieren.

2.4.2.2. Indirekte Modellierung

Beim indirekten oder auch komplementären Kalmanfilter (Complementery KF, CKF) wird anstelle des Systems der Systemfehler modelliert und anschließend mit dem Ergebnis des Systems verrechnet. Dies hat mehrere Vorteile: Oft lässt sich das Fehlerverhalten durch ein lineares System beschreiben und man erspart sich die Verwendung von UKF (siehe 2.4.3.3) und EKF (siehe 2.4.3.2). Desweiteren wird eine Verbesserung des dynamischen Verhaltens durch das Wegfallen der Modellierungsfehler erreicht. Auch wird bei einem Ausfall des Filtersystems zumindest der ungefilterte Systemwert zurückgegeben.

2.4.3. Problem der Linearität

Der Kalmanfilter bedarf eines linearen Systems. Das heißt, er wurde darauf entwickelt, dass das zu filternde System als lineare Gleichung dargestellt werden kann. Dies ist bei der 3D Orientierung nicht der Fall.
Um dieses Problem zu umgehen, gibt es unterschiedliche Varianten:

2.4.3.1. Eindimensionale Filterung

Bei dieser einfachsten der vorgestellten Methoden werden die Messwerte bezogen auf die einzelnen Achsen eindimensional gefiltert und am Ende werden die drei erhaltenen Rotationen zusammengeführt. Der Vorteil dieser Methode ist eine einfache Implementierung mit vergleichsweise geringem Rechenaufwand. Nachteilig wirkt sich allerdings die Nichtkommutativität der Rotationen aus, da diese nacheinander und nicht gleichzeitig durchgeführt werden müssen.

2.4.3.2. Erweiterter Kalmanfilter (EKF)

Beim EKF wird eine Linearisierung des Systems durchgeführt. Dadurch kann das System dann mit linearen Gleichungen dargestellt werden. Dazu wird durch Taylorreihenentwicklung aus der Systemfunktion die Systemmatrix und aus der Messfunktion die Messmatrix errechnet. Dadurch entstehen Jacobi Matrizen. Derer ij-te Eintrag ist dabei die partielle Differentiation der i-ten Komponente der Messmatrix nach der j-ten Komponente der Systemmatrix.
Durch die Linearisierung entstehen Verluste. Diese Linearisierungsverluste können durch EKFs zweiter Ordnung (Second Order EKF, SOEKF) sowie UKF (Unscented Kalman Filter) minimiert werden. Ein weiterer Nachteil (der beim SOEKF noch einmal beträchtlich größer wird) ist der höhere Aufwand an Rechenleistung. Der Hauptvorteil liegt darin, dass das Problem der Nichtkommutativität umgangen werden kann.

2.4.3.3. Unscented Kalman-Filter (UKF)

Ein weiterer Ansatz zur Umgehung des Linearisierungsproblems ist der UKF. Im Wesentlichen unterscheiden sich EKF und UKF im Umgang mit den gaußverteilten Zufallsvariablen. Beim EKF werden vor der Schätzung der Linearisierung die Zufallsvariablen als gaußverteilt approximiert. Dadurch ist die Korrektheit von Mittelwert und Kovarianz dieser nicht mehr gesichert. Dieses

Problem umgeht der UKF durch die Modellierung der Zufallsvariablen mittels einer Menge von signifikanten Punkten, sog. Sigmapunkte[6]. Nennenswerte Unterschiede zu den Ergebnissen des EKF treten aber erst bei erheblich nicht-linearen Systemen auf. Ein möglicher geringer Genauigkeitsgewinn gegenüber dem EKF steht also in keinster Weise der komplexeren Modellierung und damit schwereren Wahl der Filterparameter sowie höherem Berechnungsaufwand gegenüber.

2.5. Systeme zur Ausrichtungsbestimmung

Um die Orientierung des Systems bestimmen zu können, ist eine mathematische Darstellung nötig. Dabei ist zu beachten, dass Rotationen nicht kommutativ sind, das heißt ihre Reihenfolge nicht variiert werden kann. Es bieten sich daher mehrere Varianten der Orientierungsdarstellung an, die jeweils spezifische Vor- und Nachteile haben.

2.5.1. Rotationswinkel

Beim Rotationswinkel arbeitet man mit der Drehung um eine Ursprungsgerade, die durch einen Einheitsvektor $\vec{n} = (n_1, n_2, n_3)^T$ gegeben ist. Dabei ergibt sich die folgende Rotationsmatrix mit $C = \cos\alpha$, und $S = \sin\alpha$ [12].

$$R_{\hat{n}}(\alpha) = \begin{pmatrix} n_1^2\,(1-C) + C & n_1 n_2\,(1-C) - n_3 S & n_1 n_3\,(1-C) + n_2 S \\ n_2 n_1\,(1-C) + n_3 S & n_2^2\,(1-C) + C & n_2 n_3\,(1-C) - n_1 S \\ n_3 n_1\,(1-C) - n_2 S & n_3 n_2\,(1-C) + n_1 S & n_3^2\,(1-C) + C \end{pmatrix}$$

Wie hier zu erkennen ist, liegen die Elemente redundant vor. Dadurch wird der Speicheraufwand erhöht. Im Ausgleich dazu ist pro Drehung allerdings nur eine Berechnung nötig.

2.5.2. Eulerwinkel

Eulersche Winkel oder auch kurz Eulerwinkel stellen eine Position durch drei Winkel Ψ, Θ, Φ dar, die jeweils eine Rotation um eine bestimmte Achse beschreiben[7, Kap. 4].

Ψ Rotation um die ursprüngliche z-Achse

Θ Rotation um die neue x-Achse

Φ Rotation um die neue z-Achse

Aus diesen Winkeln ergibt sich die Rotationsmatrix:

$$R = \begin{pmatrix} \cos\Psi\cos\Phi - \sin\Psi\cos\Theta\sin\Phi & -\cos\Psi\sin\Phi - \sin\Psi\cos\Theta\cos\Phi & \sin\Psi\sin\Theta \\ \sin\Psi\cos\Phi + \cos\Psi\cos\Theta\sin\Phi & \cos\Psi\cos\Theta\cos\Phi - \sin\Psi\sin\Phi & -\cos\Psi\sin\Theta \\ \sin\Theta\sin\Phi & \sin\Theta\cos\Phi & \cos\Theta \end{pmatrix}$$

Leistungstechnisch ergibt sich dabei das Problem, dass drei Koordinatenrotationen pro Drehung nötig sind [7, Kap. 4].
Auch besitzten die Eulerwinkel sog. Singularitäten, in denen die Zuordnung nicht lokal umkehrbar ist. Dies tritt grundsätzlich auf, wenn der zweite Drehvektor gleich null ist.

2.5.3. Drehvektor/-winkel

Beim Drehvektor handelt es sich um einen sog. Pseudovektor. Die Rotation wird dabei durch das Kreuzprodukt zweier Vektoren abgebildet. Das Kreuzprodukt leidet wie die Eulerwinkel an Singularitäten, außerdem ist die Portierung auf die konkrete Anwendung der Orientierungsbestimmung nur eingeschränkt bzw. unter hohem Konvertierungsaufwand möglich.[7, Kap. 4]

2.5.4. Quaternionen

Quaternionen bauen auf dem Konzept von Drehvektor und -winkel auf und sind ein eigenes 4D-Zahlensystem auf den realen Zahlen $q = [x_0, x_1, x_2, x_3]^t$. Ein Quaternion besteht aus einem Skalaranteil x_0 und einem Vektoranteil $\vec{x} = [x_1, x_2, x_3]$. Die Orientierung eines Körpers ist durch ein Quaternion eindeutig festlegbar. Zwar leidet die Anschaulichkeit unter der Minimierung der Systemanteile, im Gegenzug lässt sich aber dadurch die Berechnungsgeschwindigkeit deutlich steigern, da die nötigen Operationen gleichermaßen minimiert werden.[7, Kap. 4][5]

2.5.5. Vergleich

	Redundante Elemente	Anschaulichkeit	Singuläre Stellen	Performance
Rotationswinkel	6	gering	nein	gut
Eulerwinkel	0	gut	ja	schlecht
Drehvektor/-winkel	1	gering	ja	-
Quaternionen	1	sehr gering	nein	sehr gut

Tabelle 2.1.: Vergleich der Koordinatensysteme

Da bei der hier vorgelegten Arbeit eine Implementierung in einem Microcontroler erfolgen soll, liegt das Hauptaugenmerk auf der Berechnungsgeschwindigkeit. Daher werden bei der Wahl der Implementierung nur Rotationswinkel und Quaternionen weiter untersucht.

3. Konzept

3.1. Überblick

Ein System zur redundanten Orientierungs- und Positionserfassung besteht aus mehreren Sensoren und einem Filtersystem zur Datenfusion. Im Folgenden soll die Auswahl aus den oben beschriebenen Systemen dargelegt werden.
Bei der Sensorauswahl fiel die Entscheidung auf eine Kombination aus inertialen Sensoren und einem Gyroskop, zwei Accelerometern und einem Magnetometer. Das Gyroskop des Inertialsensors liefert zwar präzisere Daten und ist robust gegen Umwelteinflüsse, unterliegt aber einem Drift, der bei längerem Einsatz die Messwerte unzuverlässiger macht. Der Magnetometer liefert driftfreie Daten, ist aber durch umgebende Störfelder, leicht beeinflussbar. Mit dem vom Accelerometer bei Stillstand gelieferten Gravitationsvektor kann der Gyrodrift kompensiert werden. Dazu werden sowohl der Winkel der durch das Gyroskop durch Integration ermittelt wurde als auch der Winkel, welcher durch das Accelerometer durch die Ausrichtung des Gravitationsvektors bestimmt wurde, verglichen. Anschließend wird ein gewichteter Mittelwert gebildet. Die Sensoranordnung des Inertialsystems besteht aus zwei Accelerometern. Dies erhöht durch die zusätzliche Redundanz die Genauigkeit des Systems [4]. Damit steht die Sensorik, bestehend aus Magnetometer und Inertialsystem, fest (Abb.: 3.1, orange Blöcke). Im Folgenden wird die Wahl eines geeigneten Datenfusionssystems getroffen.

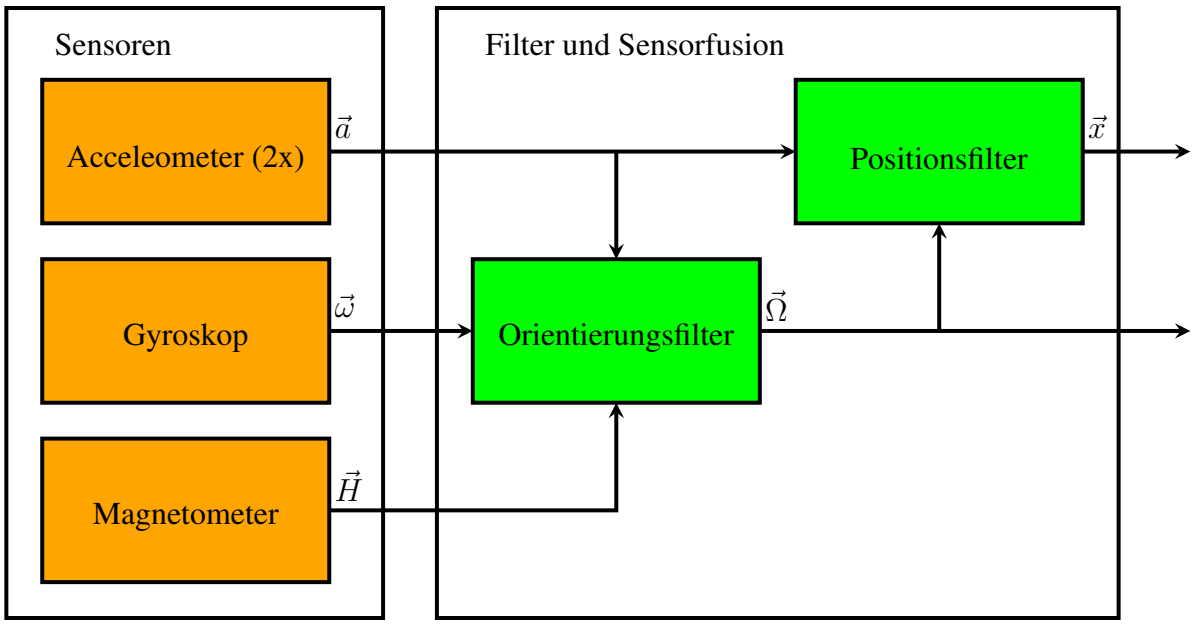

Abbildung 3.1.: Konzept des Gesamtsystems bestehend aus Sensorik und Datenfusionssystem

Bei der Auswahl des Datenfusionssystems gab es mehrere Optionen. Eine Zusammenlegung von Positions- und Orientierungsfilter ist offensichtlich keine optimale Lösung, da hier die Linearisierungsverluste zu groß werden. Daher wurde entschieden, die Datenfusion auf zwei Systeme aufzuteilen (Abb.: 3.1, grüne Blöcke).

3.2. Datenfusion

Im Folgenden wird auf zwei geeignete Lösungen zur Datenfusion gezielt eingegangen und im Anschluss die Entscheidung für eine der Lösungen getroffen.

3.2.1. Variante Haid

Die erste Lösung stammt von Markus Haid [4].
Das System von Haid basiert auf eindimensionalen Filtern, sowohl zur Orientierungs- als auch zur Positionsbestimmung.

3.2.1.1. Orientierungserfassung

Bei der Orientierungsbestimmung setzt Haid dabei auf einen eher unkonventionellen Ansatz: Dem Sensordrift wird ein Zustand zugeordnet. Das Messignal des Sensors kommt beim Kalman-

filter bereits integriert als Drehwinkel φ_D an. “Mit Hilfe der Zusatzinformation, in Form des Sensorsignals des Magnetfeldsensors φ_M, liefert der [Kalmanfilter-]Algorithmus einen Schätzwert des Drehwinkels φ_K” [4, Seite 110ff].

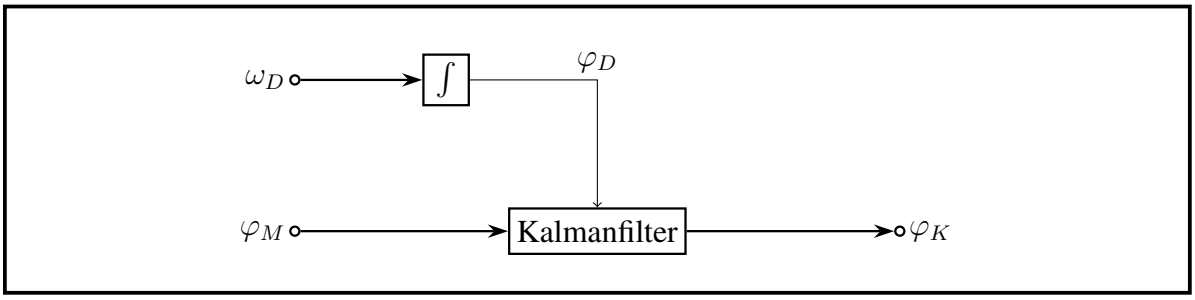

Abbildung 3.2.: Orientierungsbestimmung nach Haid

Für eine Adaption an die Ziele der vorliegenden Arbeit wäre zusätzlich eine Integration des Beschleunigungssensors in das System nötig.

3.2.1.2. Positionserfassung

Auch bei der Positionserfassung weicht Haid von den normalerweise eingesetzten Instrumenten ab. Der Kalmanfilter erhält nicht den Beschleunigungswert, sondern den Deltawert zweier redundanter Beschleunigungssensoren δe_a. Aus diesen berechnet er dann die geschätzten Fehlervektoren der Beschleunigung e_a, Geschwindikeit e_v sowie Position e_s. Diese fließen dann in die jeweilige Berechnung ein. So wird der Fehler an drei Stellen korrigiert [4, Seite 121ff].

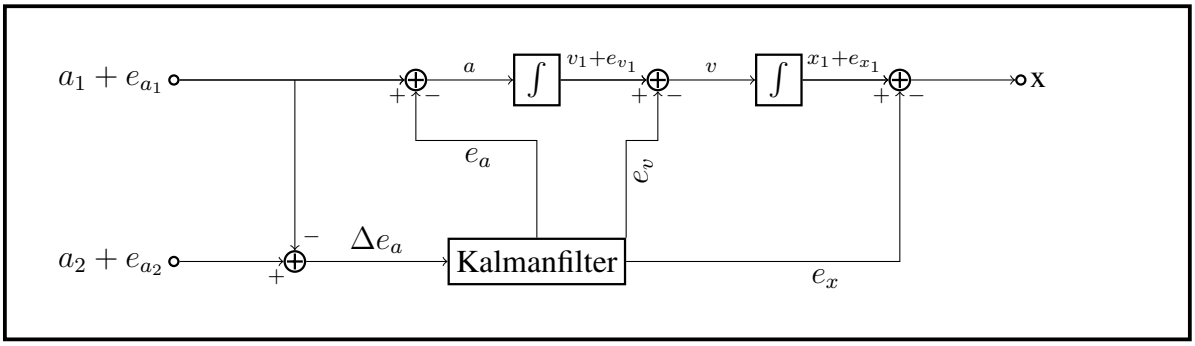

Abbildung 3.3.: Positionsbestimmung nach Haid

3.2.1.3. 3D-Zusammenführung

Für die 3D-Zusammenführung wertet Haid jeweils alle drei Messungen aus und führt die Rotationen aufeinanderfolgend aus. Zur Reduktion von Fehlern durchlaufen die Rotationen vor der Kalmanfilterung eine Orthogonalitätsausgleichsmatrix.

3.2.2. System Sabatini

Die zweite Lösung stammt von Angelo M. Sabatini [11].
Dieser stellt nur eine Lösung für die Orientierungsbestimmung vor. Eine Option zur Positionsbestimmung wird nicht angeboten. Dazu verwendet er einen Extended Kalman Filter auf Quaternionenbasis.
Als Sensorik werden Magnetometer, Accelerometer und Gyroskop genutzt. Hier ist also keine Integration weiterer Sensorik für die beabsichtigte Anwendung nötig.

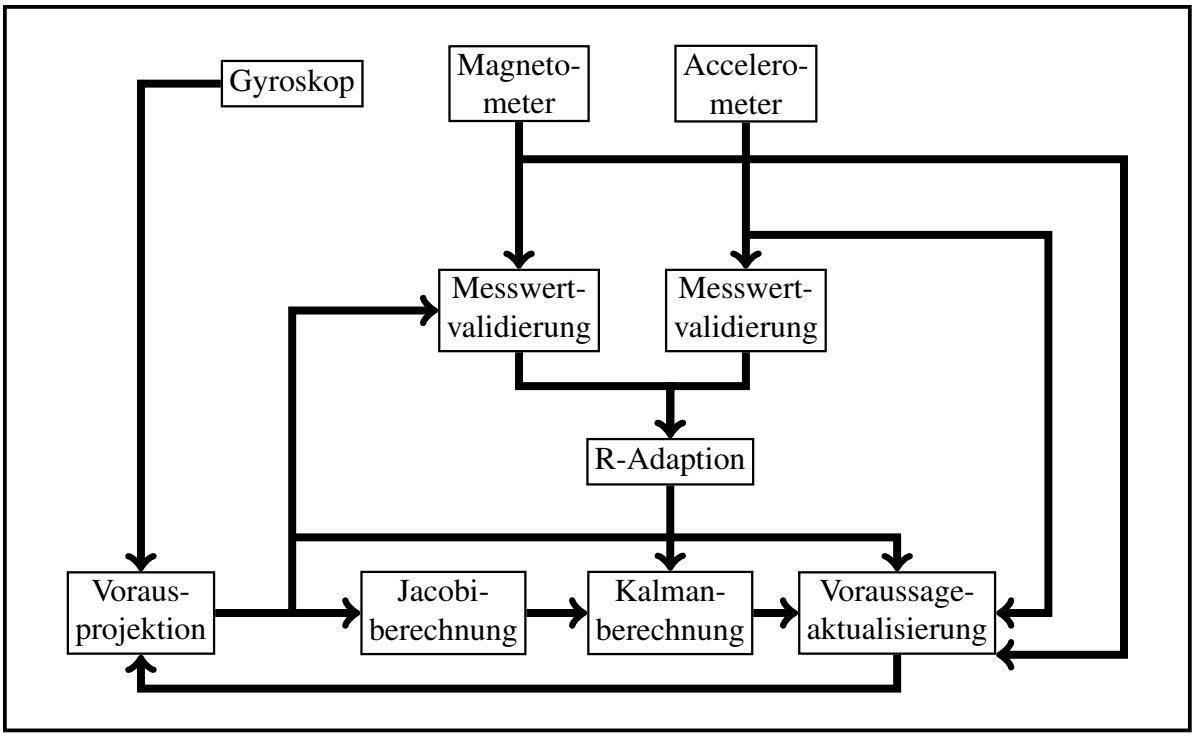

Abbildung 3.4.: Orientierungsbestimmung nach Sabatini

Wie man sieht, werden die Messwerte von Magnetsensor und Accelerometer auf Brauchbarkeit geprüft, bevor sie verwendet werden. Bei der R-Adaption wird die Kovarianzmatrix R einbezogen, welche die Kovarianzen des Beschleunigungssensors und des Magnetometers enthält.
Die Jacobiberechnung dient der Linearisierung des Zustandes. Dafür muss eine Taylor-Mac-Laurin-Expansion erster Ordnung durchgeführt werden [11, Seite 1349].
Aufgrund der Verwendung von Quaternionen stellt sich hier der Bedarf einer 3D-Zusammenführung nicht.

3.2.3. Auswahl des Verfahrens

Die Entscheidung des zu implementierenden Verfahrens fällt auf eine Mischung der beiden Systeme. Die Orientierung wird durch das Verfahren von Sabatini ermittelt, die Position nach Haid. Der Hauptgrund für die Wahl ist, dass nach allgemeinen wissenschaftlichen Erkenntinssen der Linearisierungsverlust durch den EKF geringer ist, als der Fehler durch die nicht kommutative Drehwinkelmultiplikation. Ein weiterer Grund ist die Tatsache, dass die Quaternionen in der Rechengeschwindigkeit und im Speicherverbrauch überlegen sind.

3.3. Mathematisches Konzept

Im Folgenden soll nun detailliert auf die mathematischen Gleichungen des System eingegangen werden.

3.3.1. Orientierungsrepräsentaton und -bestimmung

Die Orientierung eines festen Körpers wird durch die Achsen-Orientierung eines am Körper angehefteten Koordinatensystems (Körperkoordinatensystem $\mathcal{B}$ (body-frame)) im Vergleich zu einem absoluten Koordinatensystem (Navigationskoordinatensystem $\mathcal{N}$) ermittelt.
Die Transformation eines zeitabhängigen 3×1 Spaltenvektors $\vec{x}\,(t)$ zwischen den Repräsentationen $\mathcal{B}$ und $\mathcal{N}$ ist wie folgt definiert [11, Seite 1347]:

$$\vec{x}^{b}\,(t) = C_{n}^{b}\left[q\,(t)\right]\vec{x}^{n}\,(t) \tag{3.1}$$

Im weiteren Verlauf wird das Argument t aus Gründen der Einfachheit ausgelassen.

Die Rotationsmatrix C_n^b (DCM) für die Transition kann aus dem Orientierungsquaternion $q = \left[\vec{e}^T, q_4\right]^T$ abgeleitet werden, wobei $\vec{e} = [q_1, q_2, q_3]^T$ den Vektorteil des Quaternions und q_4 den Skalarteil darstellt:

$$C_n^b = \frac{1}{\sqrt{\|\vec{e}\|^2}} \cdot \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2\left(q_1 q_2 + q_3 q_4\right) & 2\left(q_1 q_3 - q_2 q_4\right) \\ 2\left(q_1 q_2 - q_3 q_4\right) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2\left(q_2 q_3 + q_4 q_1\right) \\ 2\left(q_1 q_3 + q_2 q_4\right) & 2\left(q_2 q_3 - q_4 q_1\right) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{3.2}$$

Die Bewegung des Körpers unterliegt der folgenden Differentialgleichung:

$$\frac{d}{dt} q = \Omega\left[\vec{\omega}\right] q \tag{3.3}$$

wobei:

$$\Omega\left[\vec{\omega}\right] = \frac{1}{2} \begin{bmatrix} [\vec{\omega}\times] & \vec{\omega} \\ -\vec{\omega}^T & 0 \end{bmatrix} \tag{3.4}$$

$\vec{\Omega}\left[\vec{\omega}\right]$ ist eine 4×4 antisymmetrische Matrix mit der Winkelgeschwindigkeit $\vec{\omega} = [r, p, q]^T$ von $\mathcal{B}$ relativ zu $\mathcal{N}$ in $\mathcal{B}$ dargestellt

Der Operator $\times$ entspricht dem Vektorprodukt:

$$[\vec{\omega}\times] = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{3.5}$$

Das zeitdiskrete Modell in Korrespondenz zu 3.3 ist wie folgt beschrieben [11, Seite 1348]:

$$q_k = \begin{cases} \exp\left(\Omega_{k-1} T_s\right) q_{k-1} & k > 0 \\ q\left(0\right) & k = 0 \end{cases} \tag{3.6}$$

wobei T_s die Abtastrate und k der $\frac{t}{T_s}$-te Rechenschritt ist.

Diese Variante von Sabatini erweist sich als problematisch. Ω eignet sich zwar optimal zur Drehung von Matrizen, bei der Drehung des Quaternions tritt aber ein Fehler auf. Bei einer Drehung um eine Achse werden nicht nur die anderen Achsen, sondern auch die Drehachse selbst ge-

dreht. Dies führt auf Dauer zu einem sich akkumulierenden Fehler, der vermutlich von Sabatini aufgrund der starken Korrektur durch Accelerometer und Magnetometer nicht bemerkt wurde. Anstelle der Drehung von q via Ω wird ein neues Quaternion $q^{'}$ definiert. Mit

$$\Delta\vec{\phi} = \frac{1}{2}\vec{\omega} \cdot T_s \tag{3.7}$$

und

$$q^{'} = \begin{bmatrix} \frac{\sin\left|\Delta\vec{\theta}\right|}{\left|\Delta\vec{\theta}\right|} \cdot \Delta\vec{\theta} \\ \cos\left|\Delta\vec{\theta}\right| \end{bmatrix} \tag{3.8}$$

ergibt sich:

$$q_k = \begin{cases} q_k = q^{'} \cdot q_{k-1} & k > 0 \\ q\left(0\right) & k = 0 \end{cases} \tag{3.9}$$

Dadurch wurde der oben angesprochene Fehler korrigiert.
Gleichung 3.2 wird zur Aktualisierung des DCM-Ausdrucks genutzt, sobald die Lösung von 3.9 ins nächste Zeitintervall fortgeschritten ist.

3.3.1.1. Sensormodell

Gyroskop, Accelerometer und Magnetometer liefern dreidimensionale Vektoren in Abhängigkeit von der Winkelgeschwindigkeit $\vec{\omega}_{\texttt{true}}$, der Gesamtbeschleunigung (Gravitation $\vec{g}$ und Beschleunigung $\vec{a}_{\texttt{body}}$) und dem Erdmagnetfeld $\vec{h}$. Diese sind wie folgt ausgedrückt [11, Seite 1348]:

$$\begin{aligned} \vec{\omega} &= {}^{g}K\vec{\omega}_{\texttt{true}} + {}^{g}\vec{b} + {}^{g}\vec{v} \\ \vec{a} &= {}^{a}K\left[C_n^b\left(q\right)\left(\vec{g} + \vec{a}_{\texttt{body}}\right)\right] + {}^{a}\vec{b} + {}^{a}\vec{v} \\ \vec{m} &= {}^{m}KC_n^b\left(q\right)\vec{h} + {}^{m}\vec{b} + {}^{m}\vec{v} \end{aligned} \tag{3.10}$$

${}^{g}K$, ${}^{a}K$ und ${}^{m}K$ sind Skalierungsmatrizen (idealerweiser die Identitätsmatrix I); ${}^{g}\vec{b}$, ${}^{a}\vec{b}$ und ${}^{m}\vec{b}$ sind die Fehlervektoren der Messwerte (idealerweise sind diese gleich null); ${}^{g}\vec{v}$, ${}^{a}\vec{v}$ und ${}^{m}\vec{v}$ werden

als nichtkorelliertes, weißes gaußsches Messrauschen angenommen mit Kovarianzmatrix $\Sigma_g = \sigma_g^2 I$, $\Sigma_a = \sigma_a^2 I$, $\Sigma_m = \sigma_m^2 I$.

3.3.1.2. Filtermodell

Der Zustandsvektor besteht aus dem Rotationsquaternion, gestützt durch die Fehler von Accelerometer und Magnetometer, deren Komponenten als Zufallsbewegungen modelliert sind [11, Seite 1348].

$$\vec{x}_{k+1} = \begin{bmatrix} q_{k+1} \\ {}^a\vec{b}_{k+1} \\ {}^m\vec{b}_{k+1} \end{bmatrix} = \Phi\left(T_s, \vec{\omega}_k\right)\vec{x}_k + \vec{w}_k = \begin{bmatrix} \exp\left(\Omega_k T_s\right) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I \end{bmatrix} \begin{bmatrix} q_k \\ {}^a\vec{b}_k \\ {}^m\vec{b}_k \end{bmatrix} + \begin{bmatrix} {}^q\vec{w}_k \\ {}^a\vec{w}_k \\ {}^m\vec{w}_k \end{bmatrix} \tag{3.11}$$

Dabei ist $\mathbf{0}$ die 3×3 Null-Matrix und

$${}^q\vec{w}_k = -\frac{T_s}{2}\Xi_k {}^g\vec{v}_k = -\frac{T_s}{2}\begin{bmatrix} [\vec{e}_k\times] + q_{4k}I \\ -\vec{e}_k^T \end{bmatrix} {}^g\vec{v}_k \tag{3.12}$$

${}^a\vec{w}_k$ und ${}^m\vec{w}_k$ sind nullgemittelte weiße Rauschprozesse, mit Kovarianzmatrix ${}^a\Sigma_k = T_s^a\sigma_w^2$ bzw. ${}^m\Sigma_k = T_s^m\sigma_w^2$ [11, Seite 1349].

Aufgrund der Annahme, dass ${}^q\vec{w}_k$, ${}^a\vec{w}_k$ und ${}^m\vec{w}_k$ voneinander unabhängig sind, ergibt sich als Kovarianzmatrix für das Prozessrauschen [11, Seite 1349]:

$$Q_k = \begin{bmatrix} \left(\frac{T_s}{2}\right)^2 \Xi_k \Sigma_g \Xi_k^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & {}^a\Sigma_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & {}^m\Sigma_k \end{bmatrix} \tag{3.13}$$

Das Messmodell wird durch das Übereinanderschreiben der Accelerometer- und Magnetometermesswerte in einem Vektor erzeugt [11, Seite 1349]:

$$\begin{aligned}\vec{v}_{k+1} &= \begin{bmatrix}\vec{a}_{k+1}\\ \vec{m}_{k+1}\end{bmatrix} = f\left[\vec{x}_{k+1}\right] + \vec{v}_{k+1} \\ &= \begin{bmatrix}C_n^b\left(q_{k+1}\right) & \mathbf{0}\\ \mathbf{0} & C_n^b\left(q_{k+1}\right)\end{bmatrix}\begin{bmatrix}\vec{g}\\ \vec{h}\end{bmatrix} + \begin{bmatrix}{}^a\vec{b}_{k+1}\\ {}^m\vec{b}_{k+1}\end{bmatrix} + \begin{bmatrix}{}^a\vec{v}_{k+1}\\ {}^m\vec{v}_{k+1}\end{bmatrix}\end{aligned} \tag{3.14}$$

Die zugehörige Kovarianzmatrix errechnet sich wie folgt [11, Seite 1349]:

$$R_{k+1} = \begin{bmatrix}{}^aR_{k+1} & \mathbf{0}\\ \mathbf{0} & {}^mR_{k+1}\end{bmatrix} \tag{3.15}$$

Vorausgesetzt der Annahme, dass das Rauschen der Magnetometer- und Accelerometeraspekte ein weißer Rauschprozess ist, gilt [11, Seite 1349]:

$${}^aR_{k+1} = {}^R\sigma_a^2 \text{ und } {}^mR_{k+1} = {}^R\sigma_m^2 \tag{3.16}$$

Bevor die Messung $\vec{z}_{k+1}$ nun an den Filter weitergeleitet werden kann, muss die Kovarianzmatrix modifiziert werden:
In Gleichung 3.14 weicht die Beschreibung des Messmodells für den Accelerometer vom Sensormodell 3.10 ab, in welchem die Beschleunigungskomponente aus einer Bewegung des Körpers nicht auftauchen darf. Dazu wird der Betrag der gemessenen Beschleunigung auf signifikante Abweichungen von der Gravitationsbeschleunigung untersucht. Ist dies der Fall, wird die Varianz ${}^R\sigma_a^2$ auf einen extrem hohen Wert gesetzt, um den Filter zu zwingen, nicht mehr auf den Accelerometer zurückzugreifen [11, Seite 1349].

$${}^R\sigma_a^2 = \begin{cases}\sigma_a^2 & |||\vec{a}_{j+1}|| - ||\vec{g}||| < \varepsilon_a \forall j \in [k-k_a, k]\\ \infty & sonst\end{cases} \tag{3.17}$$

In magnetisch gestörten Umgebungen kann die Abweichung des gemessenen Magnetfeldes vom lokalen Erdmagnetfeld so groß sein, dass die Messung ihre Verlässlichkeit verliert und der Dip-Winkel

$$\hat{\theta}_{\texttt{dip}} = \arccos\left(\frac{C_b^n\left(\bar{q_{k+1}}\right)\vec{m}_{k+1}\cdot C_b^n\left(\bar{q_{k+1}}\right)\bar{\vec{a}_{k+1}}}{||\vec{m}_{k+1}||\left|\left|\bar{\vec{a}_{k+1}}\right|\right|}\right) \tag{3.18}$$

zu stark von seinem lokalen Wert $\theta_{\mathtt{dip}}$ abweicht [11, Seite 1349]. Der Term $C_b^n\left(q_{k+1}^-\right)\vec{a}_{k+1}^-$ in 3.18 ist die geschätzte Gravitation ($^-$ bedeutet, dass es sich um den apriori-Zustand handelt). Die folgende Validierung wird angewandt [11, Seite 1349]:

$$ {}^R\sigma_m^2 = \begin{cases} \sigma_m^2 & \left| ||\vec{m}_{k+1}|| - \left|\left|\vec{h}\right|\right| \right| < \varepsilon_m \cap \left|\hat{\theta}_{\mathtt{dip}} - \theta_{\mathtt{dip}}\right| < \varepsilon_{\mathtt{dip}} \\ \infty & sonst \end{cases} \tag{3.19} $$

Durch diese Vorselektion der genutzten Sensorwerte wird die Modellierung von magnetischen Interferenzen und Bewegungsbeschleunigungen als zeitvariable Komponenten des Störvektors vermieden.

Aufgrund der Nichtlinearität von 3.14 bedarf der EKF-Ansatz einer vorhergehenden Linearisierung. Dazu wird eine Taylor-Mac-Laurin-Erweiterung erster Ordnung über die aktuelle Zustandsschätzung durchgeführt, indem die Jacobi-Matrix berechnet wird [11, Seite 1349]:

$$ F_{k+1} = \left.\frac{\delta}{\delta\vec{x}_{k+1}}\vec{z}_{k+1}\right|_{\vec{x}_{k+1}=\vec{x}_{k+1}^-} \tag{3.20} $$

Als Beispiel, das Element $F_{k+1}\left(1,1\right)$ [11, Seite 1349]:

$$ F_{k+1}\left(1,1\right) = \frac{1}{\sqrt{||\vec{e}||^2 + q_4^2}} \begin{bmatrix} 2q_1 - C_n^b\left(1,1\right)q_1 \\ 2q_2 - C_n^b\left(1,2\right)q_1 \\ 2q_3 - C_n^b\left(1,3\right)q_1 \end{bmatrix}^T \cdot \vec{h} \tag{3.21} $$

Die komplette ausfaktorisierte Matrix ist unter A.2 zu finden.

Mit diesen Definitionen kann nun der Filtervorgang definiert werden [11, Seite 1350] (vgl. 2.4).

3.3.2. Positionsrepräsentation und -bestimmung

Im Positionsmodell wird der Sensorfehler modelliert, also ein CKF implementiert. “Dabei werden die beiden Integrationsschritte in die Modellierung aufgenommen. Man erhält also einen Schätzfehler für den Positionsfehler $\dot{e}_s$” [4, Seite 121]

Der Positionsfehler wird dabei durch doppelte Integration des Beschleunigungsfehlers $\dot{e}_a$ ermittelt:

$$\dot{e}_s(t) = \dot{e}_v(t) \tag{3.22}$$

$$\dot{e}_v(t) = \dot{e}_a(t) \tag{3.23}$$

$$\dot{e}_a(t) = -\beta \cdot \dot{e}_a(t) + \dot{w}_a(t) \tag{3.24}$$

Dann ergibt sich folgende Systemgleichung:

$$\dot{\vec{x}} = \Phi(T) \cdot x(t) + G \cdot \vec{w}(t) \tag{3.25}$$

$$\begin{bmatrix} \dot{e}_s(t) \\ \dot{e}_v(t) \\ \dot{e}_a(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\beta \end{bmatrix} \begin{bmatrix} e_s(t) \\ e_v(t) \\ e_a(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_s(t) \\ w_v(t) \\ w_a(t) \end{bmatrix} \tag{3.26}$$

Für das zeitdiskrete System ergibt sich dann:

$$\dot{\vec{x}}_{k+1}^{-} = \Phi(T) \cdot x_k + \cdot \vec{w}_k \tag{3.27}$$

und die Messgleichung

$$y(k) = \vec{C} \cdot \vec{x}_k + \vec{v}_k \tag{3.28}$$

$$y(k) = \vec{C} \cdot \begin{bmatrix} e_{s|k} \\ e_{v|k} \\ e_{a|k} \end{bmatrix} + \vec{v}_k \tag{3.29}$$

Der Kalmanfilter erhält als Eingabe die Differenz der Sensorsignale. In 3.29 eingesetzt folgt:

$$y(k) = \Delta e_{a|k} \tag{3.30}$$

$$= \left[a_{2|k} + e_{a_2|k}\right] - \left[a_{2|k} + e_{a_2|k}\right] \tag{3.31}$$

$$= e_{a_2|k} - e_{a_1|k} \tag{3.32}$$

$$= \begin{bmatrix} 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} e_{s_1|k} \\ e_{v_1|k} \\ e_{a_1|k} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ e_{a_2|k} \end{bmatrix} \tag{3.33}$$

$e_{a_1|k}$ und $e_{a_1|k}$ sind dabei als diskrete Fassung von 3.24 modelliert:

$$e^-_{a_x|k+1} = e^{-\beta T} \cdot e_{a_x|k} + w_{a_x|k} \tag{3.34}$$

Betrachten wir nun beide Beschleunigungsfehler als eigene Zustände, ergibt sich ein erweitertes Systemmodell:

$$\dot{\vec{x}} = \Phi(T) \cdot x(t) + G \cdot \vec{w}(t) \tag{3.35}$$

$$\begin{bmatrix} e^-_{s|k+1} \\ e^-_{v|k+1} \\ e^-_{a_1|k+1} \\ e^-_{a_2|k+1} \end{bmatrix} = \Phi \begin{bmatrix} e_{s|k} \\ e_{v|k} \\ e_{a_1|k} \\ e_{a_2|k} \end{bmatrix} + \vec{w}_k \tag{3.36}$$

$$\Phi(T) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\beta & 0 \\ 0 & 0 & 0 & e^{\beta_2 T} \end{bmatrix} \tag{3.37}$$

$$\vec{w} = \begin{bmatrix} \vec{w}_{a_1|k} \\ w_{a_2|k} \end{bmatrix} \tag{3.38}$$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{a_1} & 0 \\ 0 & 0 & 0 & \sigma_{a_2} \end{bmatrix} \tag{3.39}$$

$$y_k = \begin{bmatrix} 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} e_{s_1|k} \\ e_{v_1|k} \\ e_{a_1|k} \\ e_{a_2|k} \end{bmatrix} + \vec{v}\,(k) \tag{3.40}$$

Bei dieser Messung wird fehlerfrei gemessen. Daher ergibt sich die Kovarianzmatrix für das Messrauschen:

$$R_k = r_k = 0 \tag{3.41}$$

Die Kalmanfiltergleichungen ergben sich nach 2.4.

4. Implementierung

Aufgrund des Zeitrahmens der Arbeit wurde auf eine Implementierung der Orientierungsbestimmung verzichtet.

4.1. Aufbau

Im Folgenden wird der Aufbau der Testhardware beschrieben.
Der Grobaufbau ist folgendem Blockdiagramm zu entnehmen:

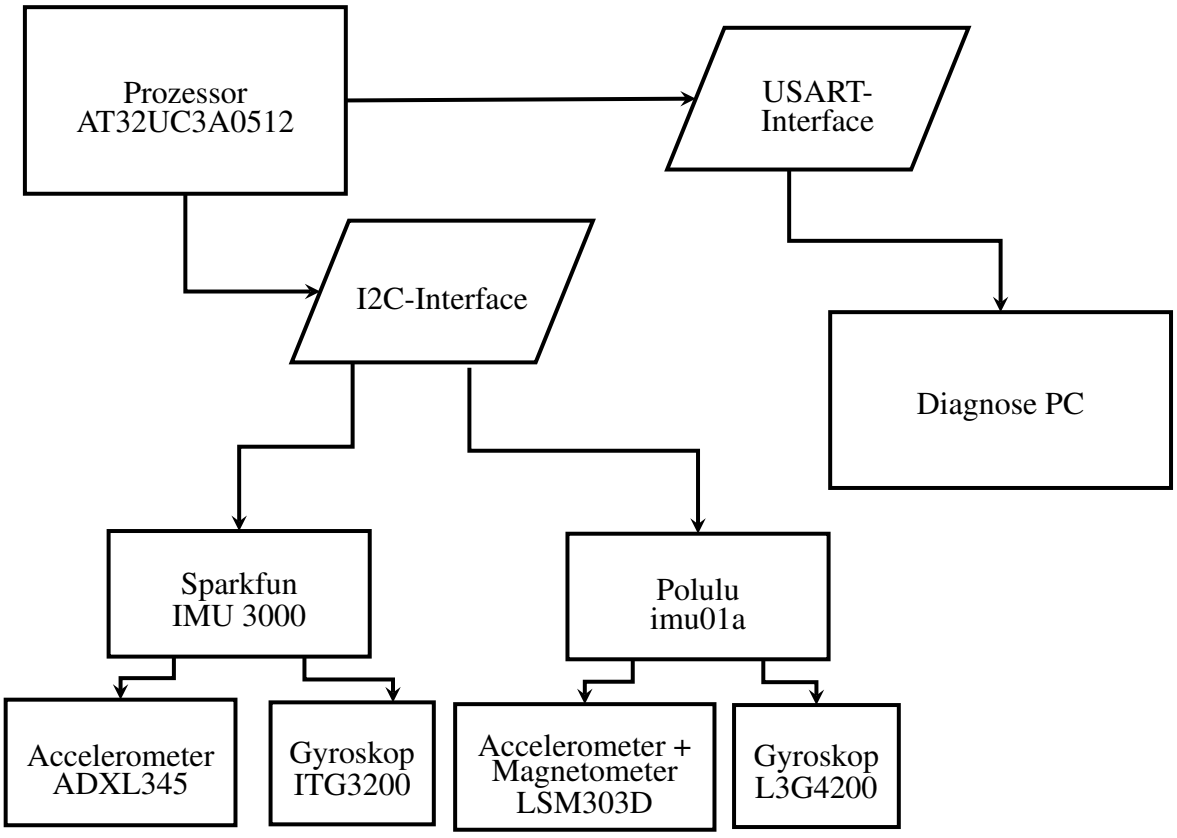

Abbildung 4.1.: Aufbau des Evaluierungssystems

Auf der nächsten Seite ist eine Photographie des Evaluierungssystems zu sehen. Der Prozessor befindet sich auf dem Evaluationsboard (Modell EVK 1100) oben im Bild. Er verfügt über 512KB Flash-Speicher und ist mit 60 MHz getaktet. Von diesem Board geht eine serielle Schnittstelle aus. Diese führt zu einem Diagnose PC und dient unter anderem der Übertragung der Orientierungsinformationen. Unten im Bild befinden sich die beiden IMUs. Diese sind über den I2C-Bus mit dem Prozessor verbunden. Bei der oberen IMU handelt es sich um das Modell *imu01a* von Polulu. Dieses Modell enthält ein Gyroskop *L3G4200* und eine Kombination aus Accelerometer und Magnetometer *LSM303D*. Bei der unteren, *IMU 3000* von Sparkfun, besteht aus einem Gyroskop *ITG3200* und einem Accelerometer *ADXL345*.

Abbildung 4.2.: Photographie des Evaluierungssystems

4.2. Matrizenmathematik

Zur Realisierung des Hauptprogrammes mussten erst die zugrundeliegenden Matrizen, Vektor- und Quaternionenoperationen implementiert werden. Der Lehrstuhl verfügt zwar über eine funktionsreiche C++-Bibliothek, diese steht aber für C nicht zur Verfügung.

4.2.1. Datendarstellung

Matrizen werden wie folgt als C-Struct dargsestellt:

```
typedef struct
{
        int h, w;
        double *x;
} matrix_t, *matrix;
```

Die Höhe und Breite der Matrix werden dabei durch die Variablen h und w angegeben. Die Werte der Matrix werden durch einen eindimensionalen double-Pointer repräsentiert. Zur Einsparung von Speicher (die Heapgröße der CPU beschränkt sich auf 16 Kilobyte) werden die Matrizen auf den minimal nötigen Speicher initialisiert. Das Matrix-Struct ist vom Typ matrix_t, zur Übergabe von Matrizen an Funktionen wird ein Pointer auf ein matrix_t Element genutzt. Zur einfacheren Lesart wurde diese Pointerklasse als Typ “matrix” definiert.
Zur Vereinfachung der Implementierung werden Quaternionen als 4x1 Matrizen und Vektoren als 3x1 Matrizen dargestellt.

4.2.2. Operationen

Es wurden folgende Funktionalitäten implementiert:

Punktprodukt Berechnet das Punktprodukt zweier Vektoren.

Multiplikation zweier Matrizen Multipliziert zwei Matrizen unter Zuhilfenahme des Punktprodukts von je einer Reihe und Spalte.

Multiplikation einer Matrix mit einer Konstanten Multipliziert eine Matrix elementweise mit einem festen Faktor.

Addition zweier Matrizen Addiert zwei Matrizen elementweise.

Transposition einer Matrix Spiegelung einer Matrix an ihrer Diagonalen.

Exponential einer Matrix Hierfür wurde die “matrix_exponential”-Bibliothek von Cleve Moler und Charles Van Loan verwendet, welche zur Berechnung des Exponentials eine Taylor-Reihe verwendet.

Änderung der Matrixgröße Wird die Matrix vergrößert, wird der Restbereich mit Nullen gefüllt. Wird die Matrix verkleinert, wird der Restbereich abgeschnitten.

Normalisierung eines Quaternions Dividiert alle Bestandteile eines Quaternions durch dessen Betrag.

Invertierung einer Matrix Invertiert eine Matrix. Dazu wird das Gauss-Jordan-Eliminationsverfahren verwendet.

4.3. Implementierung des eigentlichen Programms

Im Anschluss war die Implementierung des Hauptprogrammes unter Verwendung der oben genannten Matrixoperationen relativ einfach möglich. In den meisten Fällen konnten die festgelegten Formeln genutzt werden. In Einzelfällen war eine Änderung der Matrizengröße nötig, bevor die Operationen korrekt ausgeführt werden konnten. Teilweise waren Vereinfachungen nötig, um die Samplezeit niedrig zu halten.
Die Hauptherausforderungen waren dabei, das zu erstellende Programm möglichst generisch zu halten, keine Speicherlecks zu verursachen und vor allem effizient zu arbeiten. Das Effizienzproblem wurde weiter dadurch verstärkt, dass der bisher eingesetzte Quadrocoptercode mit keinerlei Optimierungen durch den Compiler kompiliert. Dies forderte eine möglichst hardwarenahe Programmierstrategie, welche sich vor allem durch den Einsatz von Pointern, der Minimierung von Iterationen und der Vermeidung mehrdimensionaler Arrays auszeichnet. Ziel war es eine möglichst niedrige Samplezeit zu erreichen. Dies ist die Zeit in der alle nötigen Operationen des Systems ausgeführt werden können. Samplezeit sollte hierbei $10ms$ oder niedriger zu erreichen.

4.4. Parametrierung

Für den Einsatz des Programmes ist eine Parametrierung der Fehlerwerte nötig, um die Matrizen R und Q berechnen zu können. Dazu wurden folgende Werte gewählt:

Sensor	Achse	Fehler	Standardabweichung
Gyroskop	x	$2.5\mathrm{mrad/s}$	$10\mathrm{mrad/s}$
	y	$2.5\mathrm{mrad/s}$	$10\mathrm{mrad/s}$
	z	$2.5\mathrm{mrad/s}$	$10\mathrm{mrad/s}$
Accelerometer	x	$0.25\mathrm{m/s^2}$	$0.05\mathrm{m/s^2}$
	y	$0.25\mathrm{m/s^2}$	$0.05\mathrm{m/s^2}$
	z	$0.25\mathrm{m/s^2}$	$0.05\mathrm{m/s^2}$
Magnetometer	x	$60\mathrm{mGauss}$	$1\mathrm{mGauss}$
	y	$20\mathrm{mGauss}$	$1\mathrm{mGauss}$
	z	$-30\mathrm{mGauss}$	$1\mathrm{mGauss}$

Tabelle 4.1.: Parametrierung der Sensoren

5. Evaluierung

5.1. Überblick

Die Evaluierung wurde in zwei Schritten durchgeführt.

1. Simulation
2. Tests auf der realen Hardware

5.2. Simulation

Die Simulation wird zur Validierung des mathematischen Konzeptes genutzt.

5.2.1. Simulation eines starken Gyroskopfehlers

Die Simulation soll die Fähigkeit des Systems untersuchen, auf fehlerhafte Sensoren zu reagieren. Dazu wird an allen drei Achsen des Gyroskopes der doppelte Standardfehler angelegt. Accelerometer und Magnetometer werden als fehlerfrei angenommen.

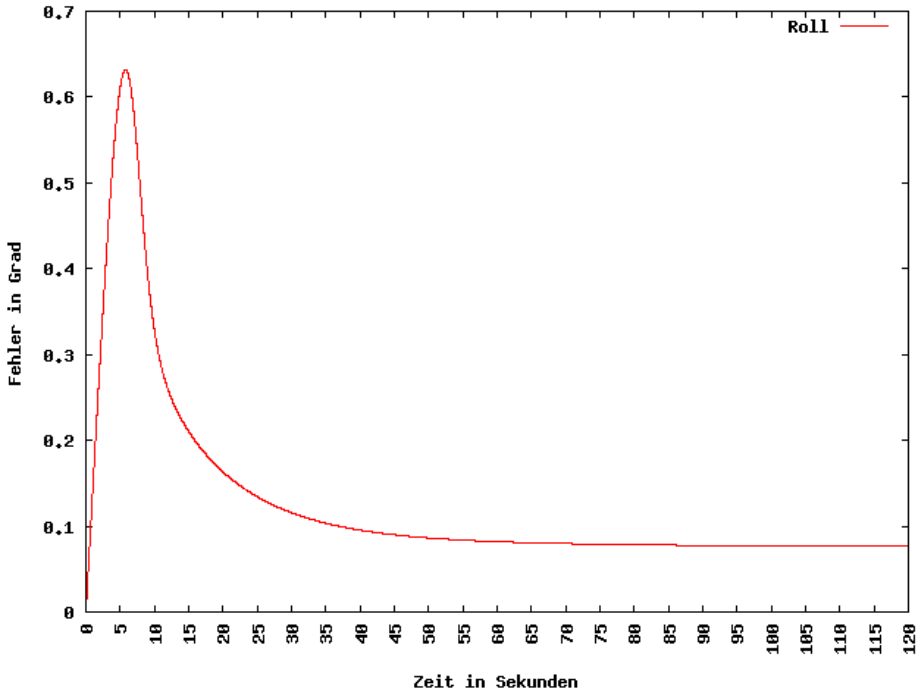

Abbildung 5.1.: Darstellung des Fehlers des Roll-Winkels

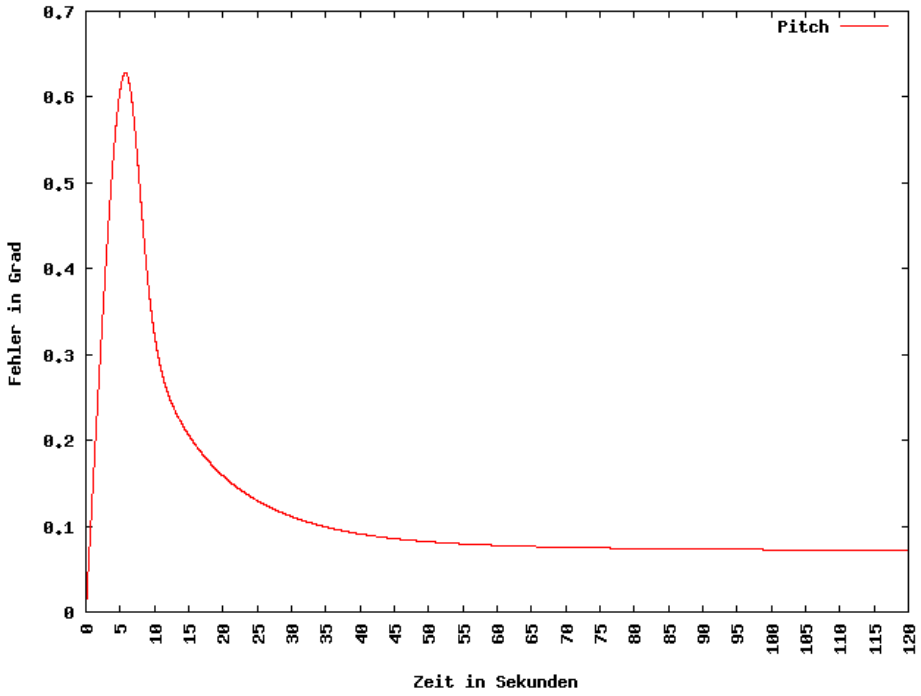

Abbildung 5.2.: Darstellung des Fehlers des Pitch-Winkels

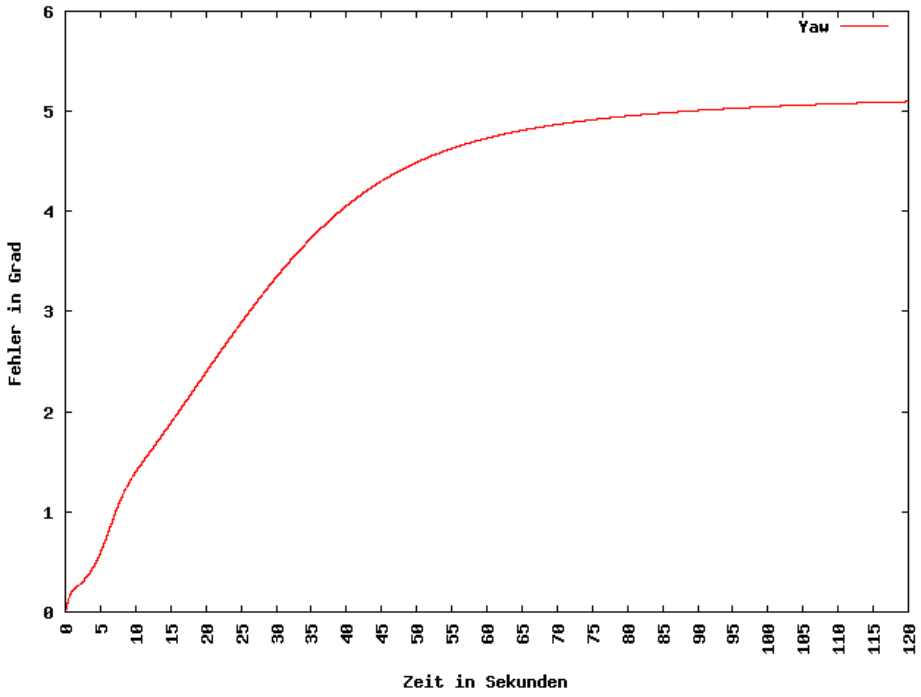

Abbildung 5.3.: Darstellung des Fehlers des Yaw-Winkels

Die Simulation belegt die grundsätzliche Fähigkeit des Systemes, einen dauerhaft großen Gyroskopfehler zu korrigieren. Roll- und Pitch-Winkel werden schnell korrigiert, während der Yaw-Winkel unter der Quaternionkorrektur leidet. Dies liegt daran, dass bei einem Quaternion eine Korrektur der Ausrichtung immer auf zwei Arten möglich ist. Dem Magnetometer gelingt es anfangs nur schwer, dies zu korrigieren. Dies liegt daran, dass die Beschleunigung im Vergleich zum Magnetfeld dreifach gewichtet ist.

5.2.2. Korrektur einer falschen Orientierung

In der nächsten Simulation wird jeweils ein um 90 Grad abweichender Orientierungswinkel angegeben und untersucht, wie lange das System zur Korrektur braucht.

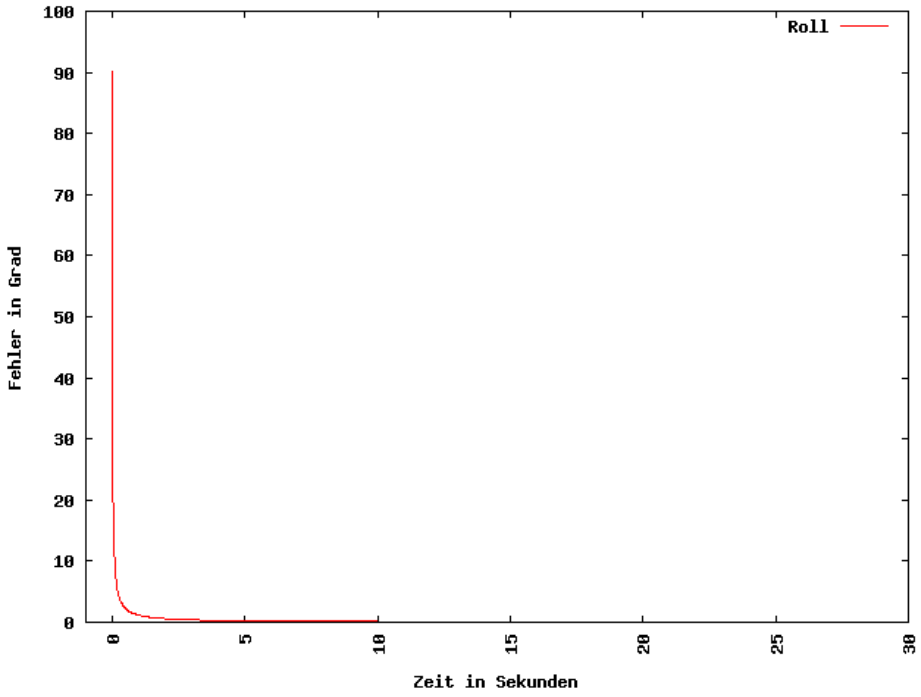

Abbildung 5.4.: Korrektur des Fehlers des Roll-Winkels

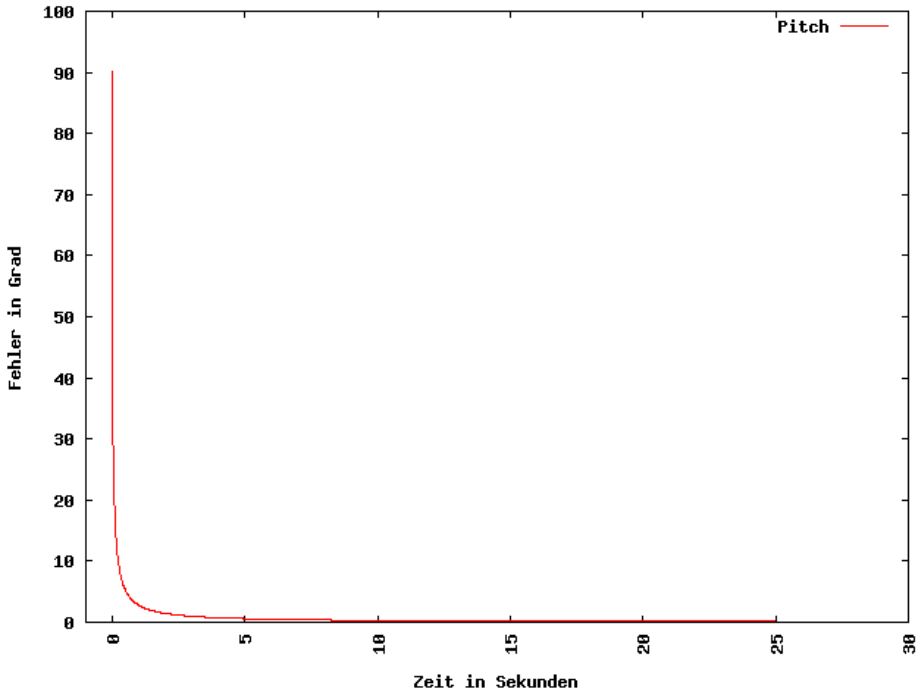

Abbildung 5.5.: Korrektur des Fehlers des Pitch-Winkels

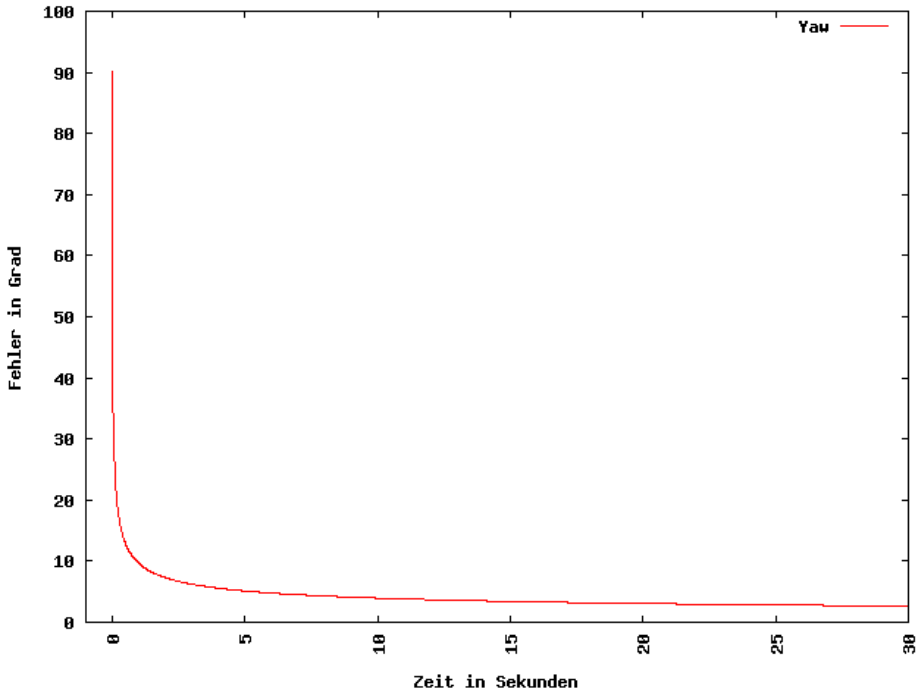

Abbildung 5.6.: Korrektur des Fehlers des Yaw-Winkels

Die Simulation belegt erfolgreich die Fähigkeit des Systems zur zügigen Korrektur eines Orientierungsfehlers. Die langsamere Korrektur des Yaw-Winkels liegt wieder an der höheren Gewichtung der Beschleunigung.

5.3. Tests auf der realen Hardware

Im Folgenden werden Validierungsexperimente mit der realen Einsatzhardware durchgeführt. Leider stand zum Zeitpunkt der Durchführung noch keine entsprechend ausgearbeitete Magnetometerimplementierung für den Quadrocopter zur Verfügung. Daher wurde die Stützsensorik auf das Accelerometer beschränkt.

5.3.1. Samplezeit

Bei Einsatz des Systems wird eine Samplezeit von 10 Millisekunden erreicht. Leider genügt aber die Rechenleistung nicht mehr, sobald das Diagnosetool aktiv ist. Dann ist eine Erhöhung der Samplezeit auf 20 Millisekunden erforderlich, um realistische Berechnungen zu erhalten.

5.3.2. Drehung um 90 Grad

Im ersten Test wird eine Drehung des Systemes um 90 Grad jeweils um eine Achse durchgeführt.

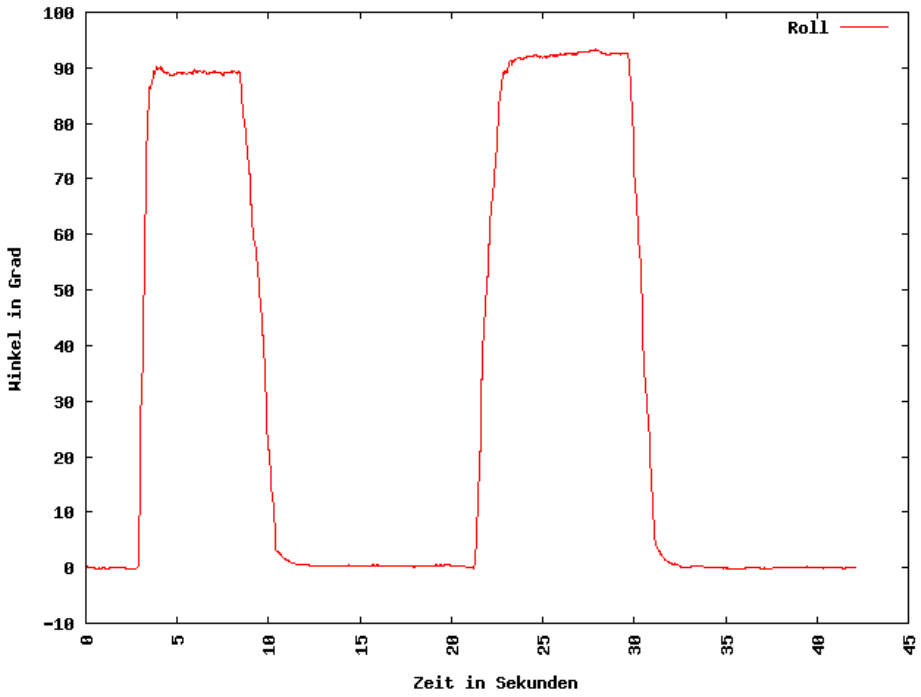

Abbildung 5.7.: Drehung um die Roll-Achse

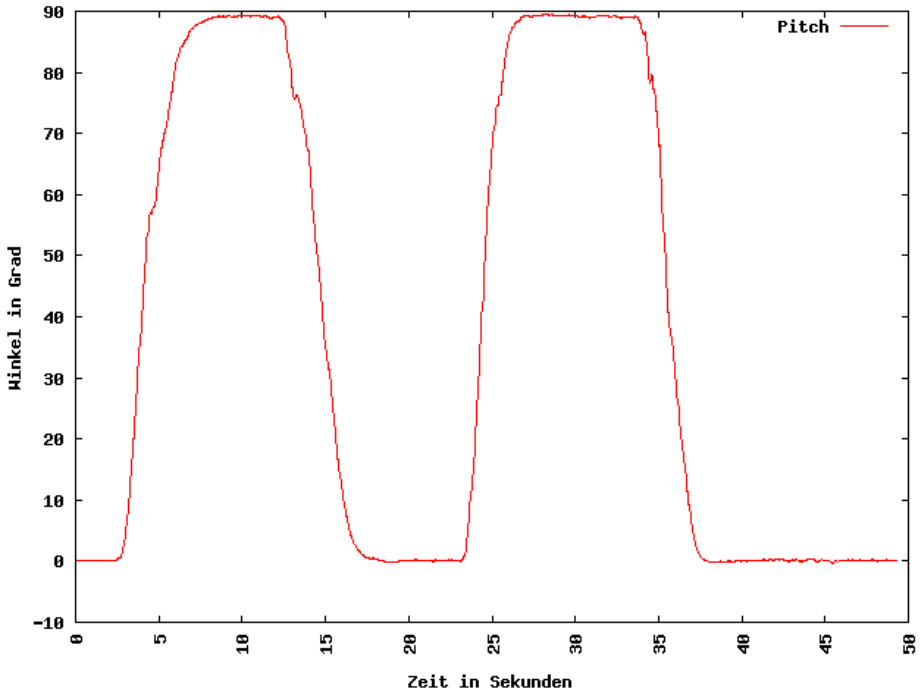

Abbildung 5.8.: Drehung um die Pitch-Achse

Bei Roll- und Pitch-Achse ist eine schnelle Annahme des ungefähren Orientierungswinkel festzustellen. Die anschließende Feinkorrektur auf einen stabilen Wert erfolgt dann innerhalb von weniger als fünf Sekunden.

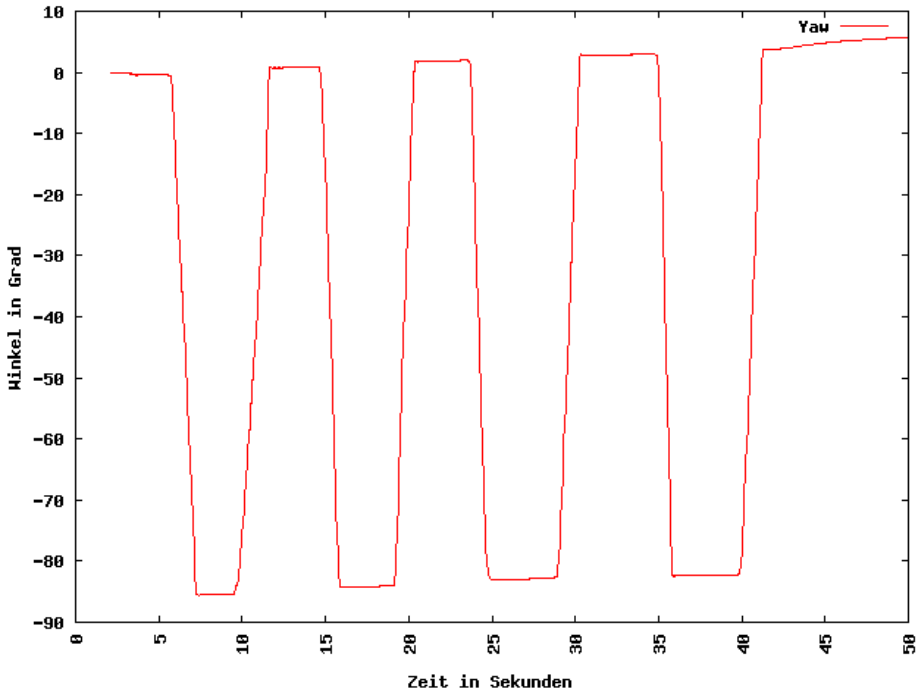

Abbildung 5.9.: Drehung um die Yaw-Achse

Die Yaw-Achse reagiert etwas schneller auf Änderungen, was an der fehlenden Korrektur durch das Magnetometer liegt. Diese fehlende Korrektur ist auch die Ursache für den ersichtlichen Drift, sowie das nicht korrekte Erreichen von 90 Grad.

5.3.2.1. Korrektur einer falschen Orientierung

Im nächsten Test wird dem System eine falsche Orientierung übergeben. Anschließend wird betrachtet, wie lange das System zur Korrektur des Fehlers braucht. Es wird jeweils ein Winkel auf 90 Grad gesetzt. Das Experiment wird nur bei Roll- und Pitch-Winkel durchgeführt, da wegen des fehlenden Magnetometers eine Korrektur des Yaw-Winkels nicht realisierbar ist. Die Aufzeichnung beginnt entlang der Zeitachse nicht bei 0 Sekunden. Dies wird durch die Notwendigkeit eines manuellen Zeitresets beim vom Lehrstuhl angewandten Kommunikationstool verursacht. Als Nullpunkt der Messung ist daher der Zeitpunkt des ersten auftretenden Wertes anzusehen.

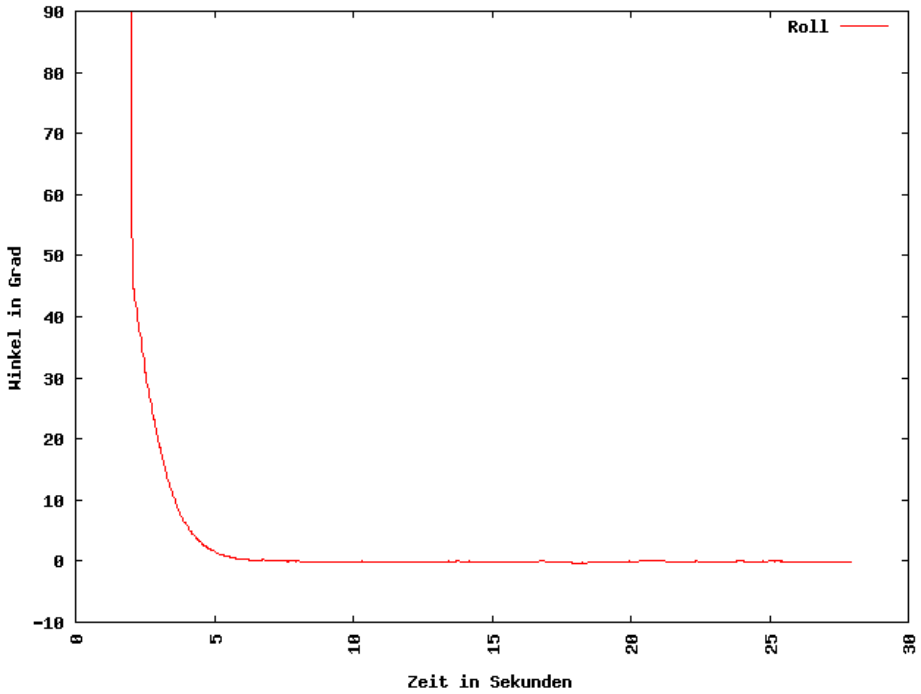

Abbildung 5.10.: Korrektur des Roll-Winkels

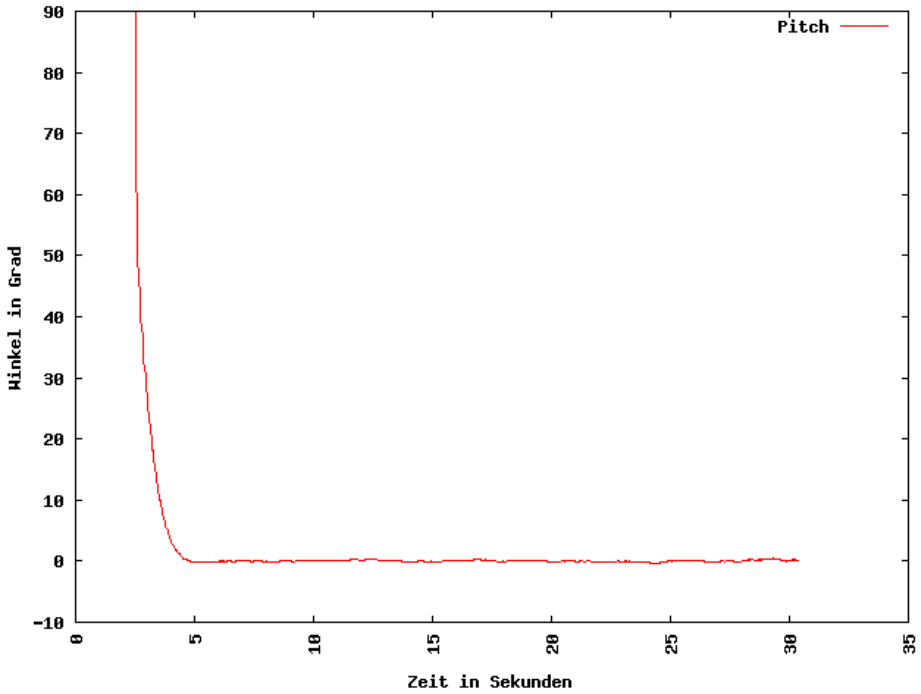

Abbildung 5.11.: Korrektur des Pitch-Winkels

Die Ergebnisse zeigen eine äußerst genaue und schnelle Korrekturfähigkeit des Systems. Bereits nach circa 2,5 Sekunden ist der Fehler unter 5 Grad gesunken. Nach 5 Sekunden liegt der Fehler bereits unter einem Grad.

Im nächsten Versuch wird das obige Experiment wiederholt. Allerdings wird nicht von 90 auf 0 Grad korrigiert, sondern von 90 auf 45 Grad.

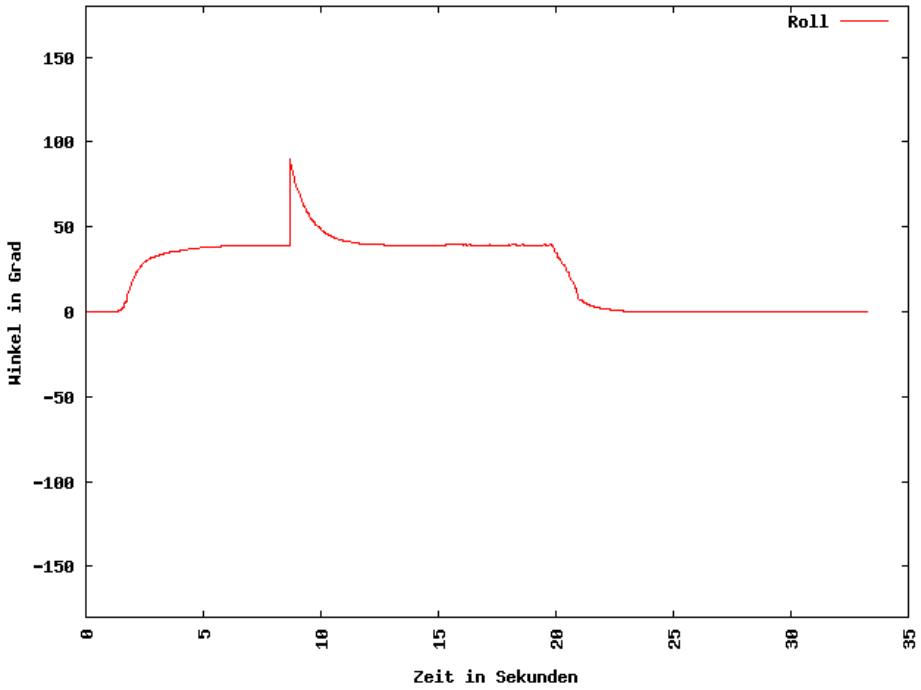

Abbildung 5.12.: Korrektur des Roll-Winkels

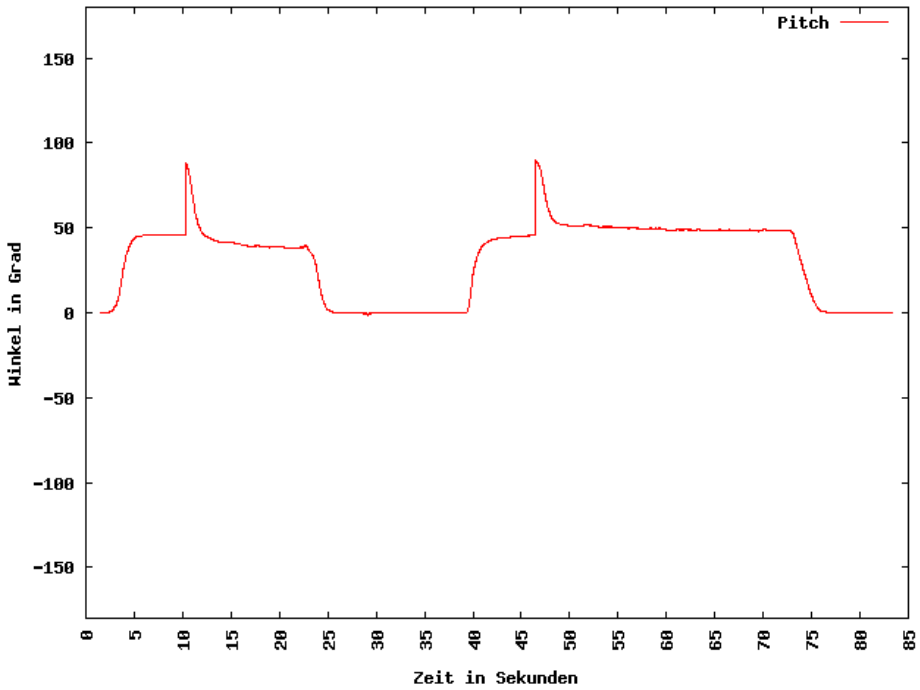

Abbildung 5.13.: Korrektur des Pitch-Winkels

Auch hier dauert es weniger als 5 Sekunden, bis die korrekte Orientierung wiederhergestellt ist.

5.3.3. Drehung um mehrere Achsen

In diesem Test soll überprüft werden, wie das System eine Drehung um verschiedene Achsen verarbeitet. Dazu wurde um alle Achsen gedreht und am Ende die Ausgangshaltung wieder eingenommen.

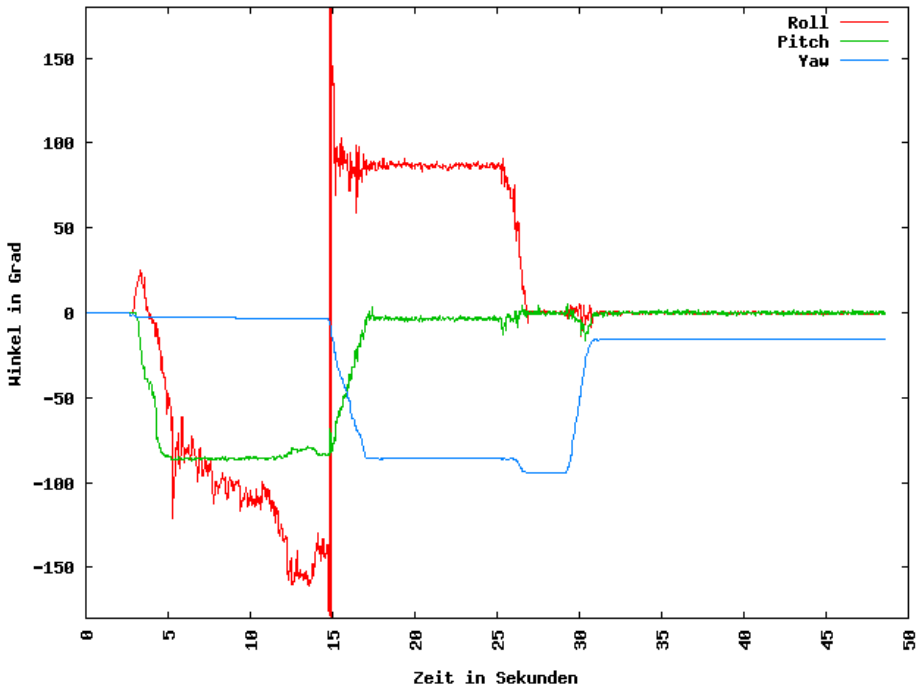

Abbildung 5.14.: Drehung um mehrere Achsen

Die errechnete Orientierung nimmt wie erwünscht wieder die Nullstellung ein. Dies erfolgt wie in den vorherigen Experimenten in relativ kurzer Zeit. Die starken Ausschläge des Roll-Winkels liegen daran, dass dieser sich sowohl um 180 Grad, als auch um -180 Grad drehen könnte, um die Zielposition einzunehmen. Der Fehler des Yaw-Winkels ist durch den Drift zu erklären.

5.3.4. Test auf dem vibrierenden Quadrocoptergerüst

Im letzten Test wird die implementierte Lösung auf dem vibrierenden Quadrocopter getestet, um die Toleranz gegenüber der Vibrationen zu überpfrüfen. Dazu wird dieser per Regler in der Nullausrichtung gehalten. Die Aufgabe dieses Testes ist es, die Eignung des implementierten Systems in einer realitätsnahen Einsatzumgebung zu evaluieren.

Dabei traten gelegentlich unerklärliche Fehler auf:

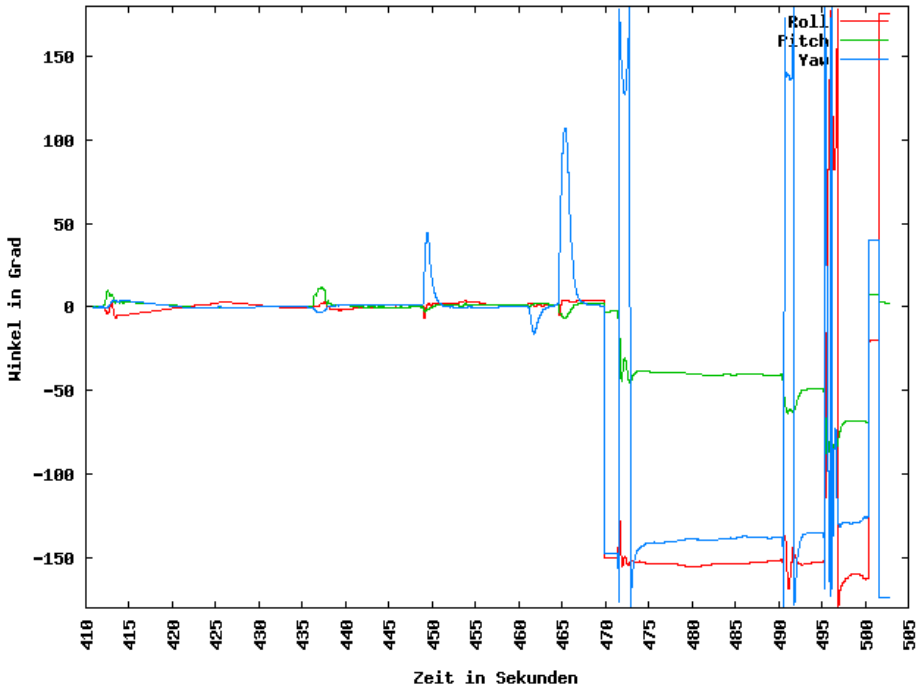

Abbildung 5.15.: Plötzlich auftretender Fehler

Ohne erkenntlichen Grund (System bewegt sich nicht) springen die Werte um bis zu 180 Grad. Teilweise schlägt die Berechnung komplett fehl. Es ist zu vermuten, dass das Problem im Bereich der Matrixinversion zu finden ist.

Im ersten Test wird untersucht, wie gut das System die Erschütterungen der Rotoren verarbeiten kann.

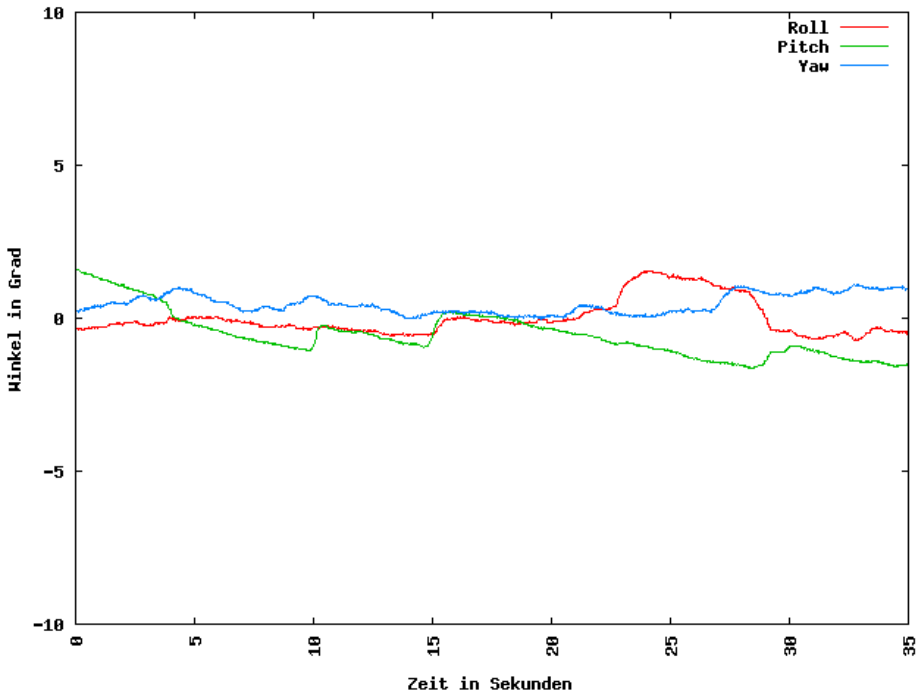

Abbildung 5.16.: Ruhelage mit eingeschalteten Rotoren

Die Messung zeigt, dass das System von den Vibrationen in der Ruhelage nicht betroffen ist. Die minimalen Ausschläge sind durchgängig mit Regelungsaktivitäten erklärbar.

Im nächsten Versuch werden mehrere Auslenkungen des Systems durchgeführt. Anschließend wird die Rückstellung dem Regler überlassen.

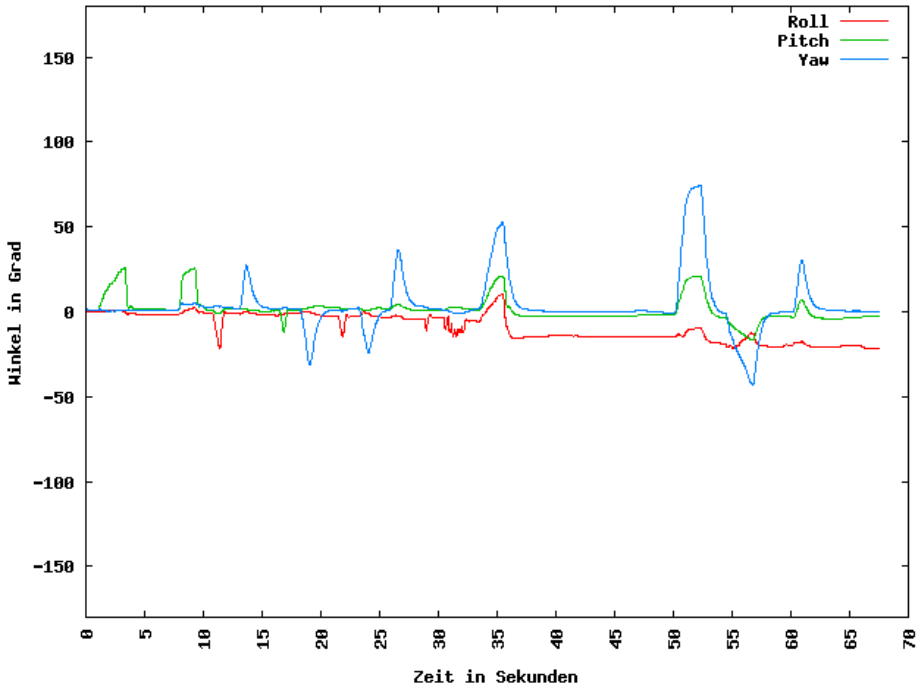

Abbildung 5.17.: Systemverhalten bei Auslenkungen

Wie gut erkennbar ist, kehren die Winkel schnell wieder zum Ausgangswert zurück. Der Fehler des Roll-Winkels lässt sich dabei durch ein Offset des Reglers erklären.

Im letzten Test wurde ein Vergleich zwischen dem bisher eingesetzten System und dem hier entwickelten System gezogen:

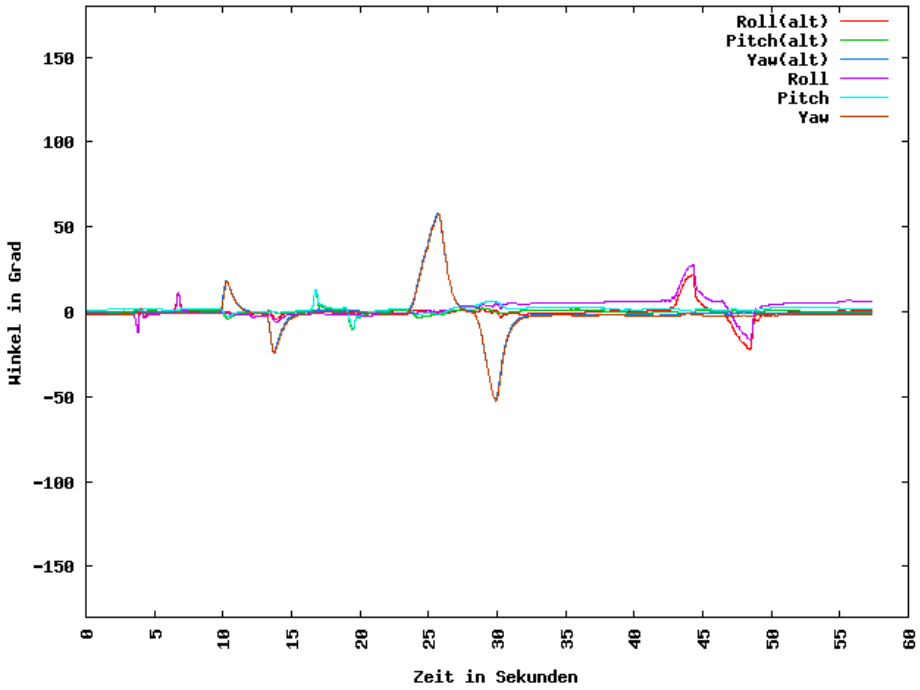

Abbildung 5.18.: Vergleich zum Altsystem

Die Winkel stimmen großteils überein. Damit ist auch die Korrektheit der Messwerte bestätigt.

6. Diskussion und Ausblick

6.1. Ergebnisse

In Simulationen wurde die Korrektheit des mathematischen Konzeptes erfolgreich belegt. Auch in den stationären Tests zeigte sich das System als sehr präzise und reaktionsschnell. Zuletzt belegte ein Test auf dem Quadrocopter mit eingeschalteten Motoren die Einsatzfähigkeit auf dem realen System, sobald der noch nicht identifizierte Fehler, der zu einem plötzlichen Totalausfall führen kann, eliminiert ist.
Das entwickelte System erfüllt damit die gestellten Ansprüche an die Orientierungsbestimmung und eignet sich für den fliegenden Einsatz. Der gestellte Anspruch an die Samplezeit des Systems wurde knapp verfehlt. Zumindest im Zusammenspiel mit der USART-Kommunikation sind 10ms nicht realisierbar. Dieses Problem ließe sich allerdings relativ einfach durch Einsatz einer schnelleren CPU lösen.
Die Plausibilität der Orientierungsbestimmung wurde durch einen Vergleich mit dem bisher eingesetzten System bestätigt. Im Vollausbau ist zu vermuten, dass durch den Einsatz des EKFs der Linearisierungsfehler geringer ausfällt.

6.2. Verbesserungspotential

Eine Steigerung der Systemqualität ist zu erwarten, sobald eine Einbindung des Magnetsensors erfolgt, da dann durch diesen eine Korrektur des Yaw-Winkels ermöglicht wird. Zudem lässt sich mit diesem - je nach Lage im Raum - der Roll- und/oder der Pitch-Winkel zusätzlich stützen. Dies dürfte zu einer Steigerung der Orientierungsqualität führen.
Sobald das Magnetometer eingerichtet ist, kann eine deutlich genauere Feinabstimmung der Kalibrierungswerte durchgeführt werden. Dies dürfte nochmals zu besseren Ergebnissen führen.

Auch waren teilweise Vereinfachungen des mathematischen Konzepts nötig, um die Leistungsfähigkeit der AVR-CPU nicht zu sprengen. Diese gehen möglicherweise zu Kosten der Genauigkeit. Der Lehrstuhl arbeitet zur Zeit an der Migration des Quadrocopters auf leistungsfähigere Hardware. Mit dieser ließe sich das vollständige Konzept umsetzen.
Auch innerhalb der Implementierung besteht Verbesserungspotential. So arbeitet der verwendete Gauß-Jordan-Algorithmus bei der Matrixinversion in $O\left(n^3\right)$ Laufzeit. Es existieren Algorithmen, die dies in einer Laufzeit, die näher an $O\left(n^2\right)$ liegt, erreichen. Diese Algorithmen sind allerdings äußerst komplex und würden den Rahmen dieser Arbeit sprengen. Hier würde es sich vom Arbeitsaufwand anbieten, dies in eine eigene Arbeit auszulagern.

6.3. Fazit

Das in der vorliegenden Arbeit präsentierte System zeigt Potential für den Einsatz im Flugbetrieb. Das System eignet sich auch als gute Basis für eine spätere Implementierung einer Positionserfassung.

Literaturverzeichnis

[1] DARLING, David: *The Encyclopedia of Science*. 04/12/2012. – URL `http://www.daviddarling.info/encyclopedia/A/accelerometer.html`

[2] DEPPNER, Heinz G.: Drehratenmessgeber. In: *STN Atlas Elektronik, Bremen* (1999)

[3] FOXLIN, E.: Inertial head-tracker sensor fusion by a complimentary separate-bias kalman filter. (1996), S. 185

[4] HAID, Markus: *Verbesserung der referenzlosen inertialen Objektverfolgung zur Low-cost Indoor-Navigation durch Anwendung der Kalman-Filterung*, Universität Siegen, Dissertation, 2005

[5] HAMILTON, William R.: *Elements of Quaternions*. Longmans Green Co., 1866

[6] HAYKIN, Simon: *Kalman Filtering and Neural Networks*. John Wiley and Sons, 2001

[7] LEY, Wilfried ; WITTMAN, Klaus ; HALLMANN, Willi: *Handbuch der Raumfahrttechnik*. Carl Hanser Verlag München, 2008. – ISBN 978-3-446-41185-2

[8] LUNZE, Jan: *Regelungstechnik 1: Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen*. Springer-Verlag, 2010. – ISBN 978-3642138072

[9] ROETENBERG, Daniel ; SLYCKE, Per J. ; VELTINK, Peter H.: Ambulatory Position and Orientation Tracking Fusing Magnetic and Inertial Sensing. In: *IEEE Transactions on Biomedical Engineering* 54 (2007)

[10] ROUMELIOTIS, Stergios I. ; SUKHATME, Gaurav S. ; BEKEY, George A.: Circumventing Dynamic Modeling: Evaluation of the Error-State Kalman Filter applied to Mobile Robot Localization. In: *IEEE International Conference on Robotics and Automation* (1999)

[11] SABATINI, Angelo M.: Quaternion-Based Extended Kalman Filter for Determining Orientation by Inertial and Magnetic Sensing. In: *IEEE Transactions on biomedical Engineering, Vol. 53, No. 7,* (2006)

[12] SCHILLING, Prof. K.: *Robotik 1*. 18.10.2012

[13] WIKIPEDIA: *Wikipedia*. 01/10/2012. – URL `http://de.wikipedia.org`

A. Anhang

A.1. zusätzliche Filtersysteme

Eine Filterung der Eingangsdaten der Sensoren ist nötig, um Abweichungen vom Realwert weitestgehend zu eliminieren. Dafür stehen zusätzlich zum Kalmanfilter verschiedene einfachere Methoden zur Verfügung:

A.1.1. Mittelwertfilter

Ein sog. Mean-Filter ist ein einfacher Filter. Dabei wird über die letzten n gemessenen Werte ein Mittelwert gebildet. Die Werte werden aus praxistechnischen Gründen meist in einem Ringbuffer gespeichert. Der Filter erkennt allerdings keine extremen Ausreißer bzw. er kann die Plausibilität der Messungen nicht prüfen.

A.1.2. Intervallfilter

Der Range-Filter ist ebenfalls relativ einfach. Ist ein Messwert nicht innerhalb eines bestimmten Bereiches, so wird er verworfen. Der Range-Filter bietet sich als Vorschaltung für den Mean-Filter an.

A.2. Ausformulierte Jacobimatrix

Die ersten 4 Spalten werden spaltenweise angegeben, aus Platzgründen werden diese untereinander geschrieben. Sie restlichen Spalten sind mit Nullen gefüllt und werden hier nicht angegeben.

$$\begin{pmatrix}
\frac{g1q1\left(q1^2+3q2^2+3q3^2+q4^2\right)+2\left(g3\left(q2^2q3+q3^3+q1q2q4+q3q4^2\right)+g2\left(q2^3-q1q3q4+q2\left(q3^2+q4^2\right)\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{-g2q1\left(q1^2+3q2^2+q3^2+3q4^2\right)+2\left(g1\left(q2^3+q1q3q4+q2\left(q3^2+q4^2\right)\right)+g3\left(-q1q2q3+q4\left(q2^2+q3^2+q4^2\right)\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{-g3q1\left(q1^2+q2^2+3\left(q3^2+q4^2\right)\right)-2\left(-g1\left(q2^2q3+q3^3-q1q2q4+q3q4^2\right)+g2\left(q1q2q3+q4\left(q2^2+q3^2+q4^2\right)\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{h1q1\left(q1^2+3q2^2+3q3^2+q4^2\right)+2\left(h3\left(q2^2q3+q3^3+q1q2q4+q3q4^2\right)+h2\left(q2^3-q1q3q4+q2\left(q3^2+q4^2\right)\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{-h2q1\left(q1^2+3q2^2+q3^2+3q4^2\right)+2\left(h1\left(q2^3+q1q3q4+q2\left(q3^2+q4^2\right)\right)+h3\left(-q1q2q3+q4\left(q2^2+q3^2+q4^2\right)\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{-h3q1\left(q1^2+q2^2+3\left(q3^2+q4^2\right)\right)-2\left(-h1\left(q2^2q3+q3^3-q1q2q4+q3q4^2\right)+h2\left(q1q2q3+q4\left(q2^2+q3^2+q4^2\right)\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}}
\end{pmatrix}$$

$$\begin{pmatrix}
\frac{-g1q2\left(3q1^2+q2^2+q3^2+3q4^2\right)-2g3\left(q1q2q3+q1^2q4+q3^2q4+q4^3\right)+2g2\left(q1^3-q2q3q4+q1\left(q3^2+q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{g2q2\left(3q1^2+q2^2+3q3^2+q4^2\right)+2g3\left(q1^2q3+q3^3-q1q2q4+q3q4^2\right)+2g1\left(q1^3+q2q3q4+q1\left(q3^2+q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{-g3q2\left(q1^2+q2^2+3\left(q3^2+q4^2\right)\right)+2\left(g2\left(q1^2q3+q3^3+q1q2q4+q3q4^2\right)+g1\left(-q1q2q3+q1^2q4+q3^2q4+q4^3\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{-h1q2\left(3q1^2+q2^2+q3^2+3q4^2\right)-2h3\left(q1q2q3+q1^2q4+q3^2q4+q4^3\right)+2h2\left(q1^3-q2q3q4+q1\left(q3^2+q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{h2q2\left(3q1^2+q2^2+3q3^2+q4^2\right)+2h3\left(q1^2q3+q3^3-q1q2q4+q3q4^2\right)+2h1\left(q1^3+q2q3q4+q1\left(q3^2+q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{-h3q2\left(q1^2+q2^2+3\left(q3^2+q4^2\right)\right)+2\left(h2\left(q1^2q3+q3^3+q1q2q4+q3q4^2\right)+h1\left(-q1q2q3+q1^2q4+q3^2q4+q4^3\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}}
\end{pmatrix}$$

$$\begin{pmatrix}
\frac{-g1q3\left(3q1^2+q2^2+q3^2+3q4^2\right)+2g2\left(-q1q2q3+q1^2q4+q2^2q4+q4^3\right)+2g3\left(q1^3+q2q3q4+q1\left(q2^2+q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{2g3\left(q1^2q2+q2^3-q1q3q4+q2q4^2\right)-q3\left(2g1(q1q2-q3q4)+g2\left(q1^2+3q2^2+q3^2+3q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{g3q3\left(3q1^2+3q2^2+q3^2+q4^2\right)+2g2\left(q1^2q2+q2^3+q1q3q4+q2q4^2\right)+2g1\left(q1^3-q2q3q4+q1\left(q2^2+q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{-h1q3\left(3q1^2+q2^2+q3^2+3q4^2\right)+2h2\left(-q1q2q3+q1^2q4+q2^2q4+q4^3\right)+2h3\left(q1^3+q2q3q4+q1\left(q2^2+q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{2h3\left(q1^2q2+q2^3-q1q3q4+q2q4^2\right)-q3\left(2h1(q1q2-q3q4)+h2\left(q1^2+3q2^2+q3^2+3q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{h3q3\left(3q1^2+3q2^2+q3^2+q4^2\right)+2h2\left(q1^2q2+q2^3+q1q3q4+q2q4^2\right)+2h1\left(q1^3-q2q3q4+q1\left(q2^2+q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}}
\end{pmatrix}$$

$$
\begin{pmatrix}
\frac{2g2\left(q1^2q3+q2^2q3+q3^3-q1q2q4\right)-2g3\left(q1^2q2+q2^3+q2q3^2+q1q3q4\right)+g1q4\left(q1^2+3q2^2+3q3^2+q4^2\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{2g3\left(q1^3+q1\left(q2^2+q3^2\right)-q2q3q4\right)+q4\left(2g1(-q1q2+q3q4)+g2\left(3q1^2+q2^2+3q3^2+q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{2g1\left(q1^2q2+q2^3+q2q3^2-q1q3q4\right)-2g2\left(q1^3+q1\left(q2^2+q3^2\right)+q2q3q4\right)+g3q4\left(3q1^2+3q2^2+q3^2+q4^2\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{2h2\left(q1^2q3+q2^2q3+q3^3-q1q2q4\right)-2h3\left(q1^2q2+q2^3+q2q3^2+q1q3q4\right)+h1q4\left(q1^2+3q2^2+3q3^2+q4^2\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{2h3\left(q1^3+q1\left(q2^2+q3^2\right)-q2q3q4\right)+q4\left(2h1(-q1q2+q3q4)+h2\left(3q1^2+q2^2+3q3^2+q4^2\right)\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}} \\
\frac{2h1\left(q1^2q2+q2^3+q2q3^2-q1q3q4\right)-2h2\left(q1^3+q1\left(q2^2+q3^2\right)+q2q3q4\right)+h3q4\left(3q1^2+3q2^2+q3^2+q4^2\right)}{\left(q1^2+q2^2+q3^2+q4^2\right)^{3/2}}
\end{pmatrix}
$$