



Institut für Elektrotechnik und Informationstechnik
Universität Paderborn
Fachgebiet Nachrichtentechnik
Prof. Dr.-Ing. Reinhold Häb-Umbach

Studienarbeit

Verfahren zur Höhenschätzung in einem Navigationsfilter mit Hilfe eines barometrischen Sensors

von

Oliver Walter
Matr.-Nr.: 6238661

Betreuer: Maik Bevermeier
Nummer: 04/09
Abgabetermin: 08.02.2010

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und unter Verwendung der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.
Die Arbeit wurde bisher noch keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Paderborn, den

(Datum)

(Unterschrift)

Aufgabenstellung

- Einarbeitung in die Grundlagen der inertialen Navigation und die damit zusammenhängenden Filterverfahren
- Inbetriebnahme und Untersuchung einer low-cost inertialen Sensoreinheit
- Aufbau eines Navigationsfilters in Matlab
- Integration eines Verfahrens zur barometrischen Höhenschätzung in das Navigationsfilter
- Untersuchung der Filterstruktur anhand von Simulationen und realen Felddaten

Abstract

In this thesis a method for position, velocity and attitude estimation using inertial navigation will be implemented and tested for its function. The required sensors such as an inertial measurement unit, GPS receivers and barometers are introduced and tested for their suitability. In addition a Kalman filter allowing to support the inertial navigation using GPS measurements will be implemented and tested. Afterwards an improvement to the classic approach of supporting the inertial navigation using GPS, by using a barometer will be introduced. Here as well a Kalman filter will be implemented to reduce the errors occurring during barometric height estimation. A map-matching process will be used. To allow simple verification of the introduced methods a process for artificial data generation will be shown. Concluding, verifications using generated and real field data data will be done.

Inhaltsverzeichnis

Abbildungsverzeichnis	iii
Tabellenverzeichnis	v
Formelzeichen	vi
Abkürzungsverzeichnis	viii
1 Einleitung	1
2 Inertiale Navigation	2
2.1 Funktionsprinzip	2
2.2 Koordinatensysteme	4
2.3 Rechenregeln	9
2.4 Strapdown-Rechnung	11
2.4.1 Orientierungsberechnung	12
2.4.2 Geschwindigkeitsberechnung	13
2.4.3 Positions berechnung	14
3 Barometrische Höhenberechnung	16
3.1 Grundlagen	16
3.2 Herleitung der barometrischen Höhenformel	17
3.3 Fehlermodelle	19
3.4 Datenerzeugung	22
3.5 Höhenberechnung	23
3.6 Beispiele	24
4 Sensoren	26
4.1 Sensor typen	26
4.1.1 GPS-Empfänger	26
4.1.2 Inertiale Messeinheit	30
4.1.3 Barometer	33
4.2 Sensorfehler	35
4.2.1 GPS-Empfänger	35
4.2.2 Inertiale Messeinheit	37
4.2.3 Barometer	41
4.3 Korrektur der Messwerte	41

4.3.1	GPS-Empfänger	42
4.3.2	Inertiale Messeinheit	42
4.3.3	Barometer	50
5	Künstliche Datenerzeugung	52
5.1	Toolbox	52
5.2	Benötigte Messwerte	53
5.2.1	Profilgenerierung als Basis der Messwerterzeugung	53
5.2.2	Inertialnavigation	55
5.2.3	Generierung der GPS-Messwerte	58
5.2.4	Messwerterzeugung des Barometer	61
5.2.5	Messwerterzeugung des Magnetometer	61
6	GPS/INS-Integration	63
6.1	Grundlagen	63
6.2	Filterentwurf	64
6.2.1	Systemmodell	64
6.2.2	Messmodelle	71
6.2.3	Korrektur der totalen Größen	75
6.3	Implementierung	76
7	Höhenfilter	83
7.1	Grundlagen	83
7.2	Herleitung des Fehlermodells sowie der Kalman-Filter Gleichungen . . .	84
7.3	Referenzhöhe	88
7.4	Implementierung	90
7.5	Tests	96
8	Kombination GPS/INS und Barometer	101
8.1	Kombination	101
9	Auswertungen zum kombinierten Filter	105
9.1	Simulierte Daten	105
9.2	Reale Daten	112
10	Zusammenfassung und Ausblick	121
Literaturverzeichnis		123
A	Inhaltsübersicht der Begleit-DVD	127

Abbildungsverzeichnis

2.1	Vergleich Plattform und Strapdown	3
2.2	Verschiedene Koordinatensysteme	4
2.3	Erdellipsoid und LLH-Koordinaten	7
2.4	Vergleich: Ellipsoid und Geoid	8
2.5	Eulerwinkel	10
2.6	Strapdown-Algorithmus	11
3.1	Messwerte für Temperatur und Druck	19
3.2	Aufgezeichneter Druck	24
3.3	Berechnetes Höhenprofil	24
3.4	Referenztemperatur- und Referenzdruckänderung	25
3.5	Auswirkungen sich ändernder Referenzparameter	25
4.1	Navilock GPS-Empfänger	26
4.2	u-blox Konfiguration in u-center	27
4.3	Koordinatensystem des MotionNode	31
4.4	MotionNode Konfiguration	32
4.5	OAK-P Drucksensor	33
4.6	OAK-P Konfiguration und Datenaufnahme	34
4.7	Unterschiedliche Sensorfehler	38
4.8	Schiefe Ebene zur Kalibrierung des Beschleunigungsmessers	45
4.9	aufgezeichnete Beschleunigungen	46
4.10	korrigierte Beschleunigungen	47
4.11	Drehplattform zur Kalibrierung des Beschleunigungsmessers	49
4.12	Vergleich verschiedener Temperatursensoren	51
5.1	Datengenerierung	62
6.1	GPS/INS-Navigationsfilter	77
7.1	Unterschiedliche Karten mit Referenzhöhen	89
7.2	Filter zur Berechnung einer korrigierten Höhe	91
7.3	Test des Höhenfilters	96
7.4	Skalierungsfaktor und Bias	97
7.5	Höhenfehler vor/nach Korrektur	98
7.6	Barometrische und korrigierte Höhe	99
7.7	Skalierungsfaktor und geschätzter Bias	99
7.8	Teststrecke mit Referenzpunkten	100

8.1 Kombination GPS, INS, Höhenfilter	102
9.1 Vergleich unterschiedlicher Höhen	107
9.2 Absolute Fehler verschiedener Höhen	107
9.3 Verteilung der Höhenfehler	108
9.4 Absolute Positionsfehler	109
9.5 Verteilung der Positionsfehler	109
9.6 Absolute Geschwindigkeitsfehler	110
9.7 Verteilung der Geschwindigkeitsfehler	111
9.8 Teststrecke für die Datenaufzeichnung	113
9.9 Höhen über dem Ellipsoid	114
9.10 Ausschnitt aus dem Höhenprofil	114
9.11 Skalierungsfaktor und Bias	115
9.12 Geschätzter Bias in den Drehraten	115
9.13 Drehraten mit und ohne Bias	116
9.14 Geschätzter Bias in den Beschleunigungen	116
9.15 GPS-Ausfall 1 (mit Barometer)	117
9.16 GPS-Ausfall 2 (mit Barometer)	118
9.17 GPS-Ausfall 3 (mit Barometer)	119
9.18 GPS-Ausfall 2 und 3 (ohne Barometer)	120

Tabellenverzeichnis

3.1 Standardabweichung der T_0 und P_0 Änderung	20
4.1 Parameter der GPS-Empfänger	37
4.2 Rauschparameter des MotionNode	39
4.3 Rauschparameter des Barometers	41
4.4 Gemessene Drehraten und Skalenfaktoren	50
9.1 Parameter für die Datenerzeugung	106
9.2 Ergebnisse der Simulationen	112

Formelzeichen

\vec{a}	Beschleunigung
a	Skalierungsfaktor (in der Höhenfilterung)
b	Bias
C	Richtungskosinusmatrix
F	Übergangsmatrizen im Systemmodell
\vec{g}_l	Schwerebeschleunigung
g	Betrag der Schwerkraftbeschleunigung
g_0	Betrag der Schwerkraftbeschleunigung auf Meereshöhe
H	Messmatrizen im Messmodell
h	Höhe
h_0	Referenzhöhe
h_{mag}	Magnetfeld
k	zeitdiskrete Zählvariable
M	Fehlausrichtungs- und Skalierungsmatrix
M	Molare Masse
\vec{n}	vektorielle Zufallsvariable
n	Zufallsvariable
P	Druck
P_0	Referenzdruck
ΔP_0	Referenzdruckänderung
\vec{p}	Positionsvektor
$\Delta \vec{p}$	Positionssinkrement
q	Quaternionenvektor
R	Gaskonstante
R_0	Durchschnittlicher Krümmungsradius
R_e	Krümmungsradius in Ostrichtung
R_n	Krümmungsradius in Nordrichtung
\vec{r}	Ortsvektor
s	Skalierungsfaktor (Sensorfehler)
T	Temperatur
T_0	Referenztemperatur
ΔT_0	Referenztemperaturänderung
t	kontinuierliche Zeitvariable
Δt	Dauer eines Zeitschrittes
\vec{v}	Geschwindigkeit
$\Delta \vec{v}$	Geschwindigkeitsinkrement
x, y, z	Koordinatenachsen

α, β, γ	Winkel
γ	Temperaturgradient (in der barometrischen Höhenberechnung)
δ	Missalignmentfaktor
θ	Pitch-Winkel
λ	Breitengrad
ρ	Dichte
$\vec{\sigma}$	Orientierungsvektor
$\Delta\vec{\sigma}$	Orientierungsänderung
σ	Standardabweichung
ϕ	Roll-Winkel
φ	Längengrad
Ψ	kreuzproduktbildende Matrix des Orientierungsfehlers
$\vec{\psi}$	Orientierungsfehler
ψ	Yaw-Winkel
Ω	Erddrehrate
$\vec{\omega}$	Drehrate

Indizes

b	Körperfestes Koordinatensystem
e	Erdfestes Koordinatensystem
i	Inertialkoordinatensystem
n	Navigationskoordinatensystem
MN	MotionNode

Abkürzungsverzeichnis

6DOF	6 dregrees of freedom
CSV	C omma-Separated V alues
DCM	D irection C osine M atrix
DGPS	D ifferential G lobal P ositioning S ystem
DOP	D ilution of precision
ECEF	E arth C entered, E arth F ixed
EGNOS	E uropean G eostationary N avigation O verlay S ervice
ENU	E ast, NU p
GPS	G lobal P osition S ystem
IMU	I nertial M easurement U nit
INS	I nertialnavigationssystem
LLH	L atitude, L ongitude, H eight
NED	N orth, E ast, D own
NMEA	N ational M arine E lectronics A sociation
SBAS	S atellite B ased A ugmentation S ystem
USB	U niversal S erial B us
WGS-84	W orld G eodetic S ystem of 1984

1 Einleitung

In dieser Arbeit soll ein Verfahren zur Positions-, Geschwindigkeits- und Orientierungsschätzung durch inertiale Navigation vorgestellt und auf seine Funktion hin getestet werden. Außerdem sollen die dazu benötigten Sensoren wie eine inertiale Messeinheit, GPS-Empfänger und Barometer gezeigt und auf ihre Tauglichkeit hin überprüft werden. Des weiteren soll ein Kalman-Filter vorgestellt und getestet werden, welches es ermöglicht, mit Hilfe von GPS-Messungen die inertiale Navigation zu stützen. Anschließend soll eine Verbesserung des klassischen Ansatzes der Stützung durch GPS mit Hilfe eines Barometers vorgestellt werden. Auch hier wird ein Kalman-Filter vorgestellt, um Fehler in der Höhenbestimmung des Barometers zu korrigieren. Hierbei soll ein Map-Matching Verfahren genutzt werden. Um einfache Tests der vorgestellten Verfahren zu ermöglichen, wird schließlich noch eine Möglichkeit zur künstlichen Datenerzeugung gezeigt. Abschließend werden Tests mit simulierten und real aufgenommenen Daten durchgeführt. Diese Arbeit ist dazu wie Folgt aufgebaut:

- Kapitel 2 befasst sich mit den Prinzipien der Inertialen Navigation
- Kapitel 3 befasst sich mit dem Verfahren der barometrischen Höhenbestimmung
- Kapitel 4 befasst sich mit den eingesetzten Sensoren und ihren Fehlern
- Kapitel 5 stellt ein Verfahren zur künstlichen Datenerzeugen vor
- Kapitel 6 führt einen Kalman-Filter zur Erkennung und Korrektur von Fehlern in der Inertialen Navigation ein
- Kapitel 7 führt den Kalman-Filter zur Erkennung und Korrektur von Fehlern in der barometrischen Höhenberechnung ein
- Kapitel 8 zeigt die Kombination aus barometrischer Höhenberechnung und inertialer Navigation zusammen mit ihren jeweiligen Fehlerfiltern
- Kapitel 9 zeigt einige kurze Tests der vorgestellten Kombination auf Basis von simulierten und realen Daten

2 Inertiale Navigation

Bei der inertialen Navigation handelt es sich im Prinzip um ein Koppelnavigationsverfahren (engl. *dead reckoning*). Bei der Koppelnavigation wird fortlaufend die Bewegungsrichtung, die Geschwindigkeit und die vergangene Zeit seit der letzten Positionsbestimmung ermittelt. Mit diesen Informationen und der letzten bekannten Position, ist es nun möglich, die aktuelle Position zu bestimmen. Hierzu wird die in dem betrachteten Zeitintervall zurückgelegte Strecke berechnet oder direkt gemessen. Die Strecke wird anschließend unter Berücksichtigung der Bewegungsrichtung zur letzten bekannten Position hinzu addiert, um die aktuelle Position zu erhalten. Bei einem Inertialnavigationssystem (INS) besteht die Idee nun darin, die für die Koppelnavigation nötigen Informationen anhand der Messung von Beschleunigungen und Drehraten zu bestimmen. Die hierfür benötigten Sensoren werden im Kapitel 4 genauer beschrieben.

2.1 Funktionsprinzip

Wie bereits in der Einleitung zu diesem Kapitel erwähnt, werden in einem INS Beschleunigungen und Drehraten gemessen, um Geschwindigkeit, zurückgelegte Strecke und Orientierung des Sensors zu berechnen. Bei früheren Systemen wurden die Beschleunigungssensoren und Drehratensensoren dabei auf einer kardanisch gelagerten und stabilisierten Plattform angebracht. Diese Systeme werden Plattform-Systeme genannt. Durch die kardanische Aufhängung wird erreicht, dass die Orientierung der Sensoren von der Orientierung des Objektes (z. B. Autos) in dem sich die Plattform befindet, entkoppelt ist. Die Messewerte der Drehratensensoren werden dabei lediglich zur Stabilisierung der Plattform genutzt. Durch diese Vorgehensweise wird erreicht, dass die sensitiven Achsen des Sensors immer in Richtung raumfester Achsen wie z. B. Norden, Osten und Unten (NED) weisen. Anhand der gemessenen Beschleunigungen kann durch einfache Integration nun direkt die Geschwindigkeit in den jeweiligen Raumrichtungen bestimmt werden. Durch zweifache Integration erhält man die zurückgelegte Strecke. Die Orientierung des Objekts kann an der Stellung der Kardanrahmen abgelesen werden.

Eine Alternative zu den Plattform-Systemen sind sogenannte Strapdown-Systeme. Mit dem Aufkommen integrierter Schaltungen und leistungsfähiger Recheneinheiten wurde es möglich, Strapdown-Systeme zu realisieren. Hierbei kommt eine Inertialsensor-Einheit, auch Inertial Measurement Unit (IMU) genannt, zum Einsatz welche die Beschleunigungssensoren, Drehratensensoren und evtl. andere Sensoren in einer Einheit zusammenfasst. Für die Navigation im dreidimensionalen Raum sind jeweils drei Dreh-

atensensoren und drei Beschleunigungssensoren nötig. Idealerweise sind drei Paare aus Drehratensor und Beschleunigungssensor in drei zueinander orthogonalen Achsen angeordnet. Auch eine Recheneinheit zum Ausführen von bestimmten Operationen kann verbaut sein. Im Gegensatz zu einem Plattform-System werden hier die Sensoren fest mit dem Objekt verbunden. Hierdurch ist die Orientierung der Sensoren nicht von der Orientierung des Objektes entkoppelt. Im Gegensatz zu einem Plattform-System muss also in einem Strapdown-System die Orientierung der Sensoren und somit auch die Orientierung des Objektes im raumfesten Koordinatensystem berechnet werden. In einem Strapdown-System werden dazu die gemessenen Drehraten verwendet. Hiermit ist es dann wiederum möglich, die im Sensorkoordinatensystem gemessen Beschleunigungswerte, in Beschleunigungen in einem anderen festen Bezugssystem umzurechnen. Wie bei einem Plattform-System kann durch Integration nun die Geschwindigkeit und die zurückgelegte Strecke berechnet werden. Auf die genaue Rechenvorschrift wird später in Abschnitt 2.4 eingegangen.

Beide Systeme haben Vor- und Nachteile. Im Gegensatz zu einem Plattform-System, bei dem die Drehratensensoren nur für geringe Drehraten ausgelegt werden müssen, ist es bei einem Strapdown-System nötig, die Sensoren auf den gesamten Bereich der möglichen Drehraten des Objektes auszulegen. Außerdem muss in einem Strapdown-System die Orientierung berechnet werden, was Rechenaufwand bedeutet, und kann nicht direkt gemessen werden. Andererseits fällt bei einem Strapdown-System der mechanische Aufwand für die kardanische Aufhängung und Stabilisierung weg. Hierdurch lassen sich Strapdown-Systeme sehr kompakt und kostengünstig herstellen. Abbildung 2.1 zeigt den Vergleich zwischen einem Plattform-System wie es im Starfighter verwendet wurde [LN3NS] und einem Strapdown-System. In dieser Arbeit wird nur das Strapdown-System behandelt.



(a) Plattform INS (Quelle: jetmann91 (eBay))



(b) Strapdown INS (Quelle: MotionNode [GLI])

Abbildung 2.1: Vergleich Plattform und Strapdown (nicht maßstabsgerecht)

2.2 Koordinatensysteme

Bevor wir zur Strapdown-Rechnung übergehen, werden noch eine Reihe von Koordinatensystemen eingeführt. Dies ist nötig, da im vorherigen Abschnitt z. B. bereits zu erkennen gewesen ist, dass Sensoren in einer IMU Messwerte in einem körpereigenen Koordinatensystem liefern. Die Navigation selbst findet allerdings in einem raumfesten Koordinatensystem statt. Dies führt dazu, dass die Definition unterschiedlicher Koordinatensysteme und ihrer Lage zueinander benötigt wird. Abbildung 2.2 zeigt dabei die Lage der unterschiedlichen Koordinatensysteme zueinander.

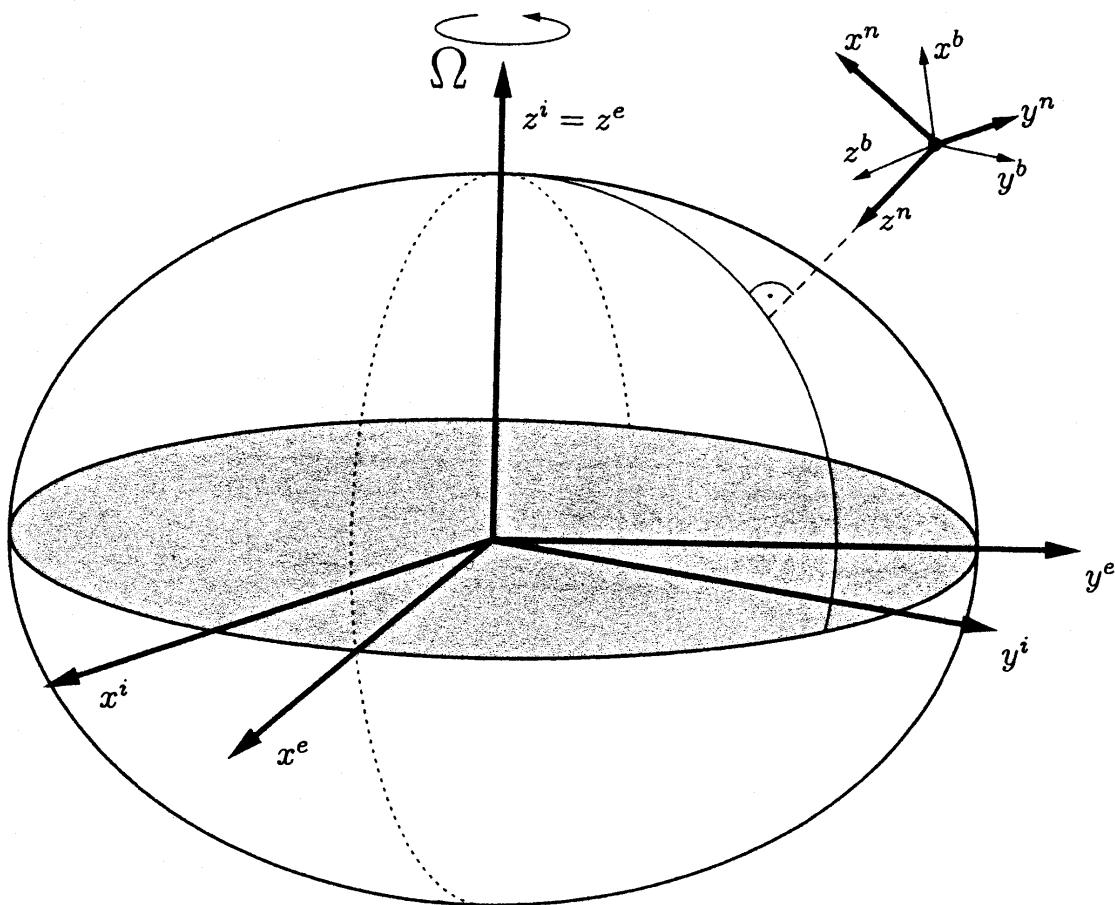


Abbildung 2.2: Verschiedenen Koordinatensysteme (Quelle: [WEN07])

Inertialkoordinatensystem (i-frame): Normalerweise ist ein Inertialsystem ein nicht rotierendes und unbeschleunigtes System. Dies trifft allerdings weder auf einen Ursprung in der Erde, der Sonne oder gar der Galaxie zu. Die Definition des Ursprungs des Inertialkoordinatensystems ist daher eine komplizierte Sache. Im Rahmen der Messgenauigkeit und dem geplanten Einsatzort auf der Erde, wird hier nur davon ausgegangen dass nur die Erdrotation einen signifikanten Einfluss hat. Der Ursprung des Inertialkoordinatensystems wird daher in den Mittelpunkt des Rotationsellipsoide gelegt, der die Form der Erde annähert. Die Achsen des

Inertialkoordinatensystems sind fest in Bezug auf die Fixsterne. Eine gängige Konvention für die Achsen ist folgende: z^i fällt mit der Rotationsachse der Erde zusammen, x^i und y^i liegen in der Äquatorebene.

Körperfestes Koordinatensystem (b-frame): Bei dem körperfesten Koordinatensystem sind die Achsen des Koordinatensystems fest in Bezug auf das Objekt. Der Ursprung befindet sich im Objekt. Eine gängige Konvention für die Achsen ist hierbei die Folgende: x^b weist nach vorne, y^b weist nach rechts und z^b weist nach unten. Wenn die Sensoren der IMU fest mit dem Objekt verbunden sind, und wenn die Sensoren exakt orthogonal ausgerichtet sind, fallen die Messwerte der Sensoren im körperfesten Koordinatensystem an. Eine eventuelle Verdrehung der Sensoren zum Objekt wäre konstant, müsste jedoch kompensiert werden. Im Folgenden werden daher b-frame und Sensorkoordinatensystem als äquivalent betrachtet (bzw. die IMU selber wird als das körperbezogene Objekt betrachtet). Die IMU misst Beschleunigungen und Drehraten des b-frames in Bezug auf das Inertialkoordinatensystem.

Je nach verwendetem Sensor kann diese Definition abweichen. Bei dem in dieser Arbeit verwendeten MotionNode weist x^{MN} nach vorne, y^{MN} nach oben und z^{MN} nach rechts. Soweit nichts anderes vermerkt, soll in dieser Arbeit die gängige Definition der Achsen des b-frame verwendet werden. Die Transformation eines Vektors \vec{x} in NM-Koordinaten zu b-Koordinaten ist durch

$$\vec{x}^b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \vec{x}^{NM} \quad (2.1)$$

gegeben.

Navigationskoordinatensystem (n-frame): Der Ursprung des Navigationskoordinatensystems fällt mit dem Ursprung des b-frames zusammen. Die Achsen sind wie folgt definiert: x^n weist nach Norden und y^n weist nach Osten. Sie liegen in der Tangentialebene an das Erdellipsoid. Die z^n Achse weist parallel zur Schwerebeschleunigung nach unten. Einzelne Komponenten eines Vektors in Koordinaten des Navigationskoordinatensystems werden mit n, e und d (north, east, down) bezeichnet. Dieses Koordinatensystem entspricht anschaulich einem Kartenausschnitt der einen kleinen räumlichen Ausschnitt der Erde als Ebene darstellt.

Eine weitere gebräuchliche Definition der Richtungen der Achsen ist Osten, Norden und Oben (ENU). Soweit nichts anderes vermerkt, soll in dieser Arbeit für den n-frame NED gelten. Die Transformation eines Vektors \vec{x} in ENU-Koordinaten zu NED-Koordinaten ist durch

$$\vec{x}^{NED} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \vec{x}^{ENU} \quad (2.2)$$

gegeben.

Erdfestes Koordinatensystem (e-frame): Der Ursprung des erdfesten Koordinatensystems liegt wie beim i-frame im Erdmittelpunkt. Die Achsen sind fest in Bezug auf die Erde. Eine gebräuchliche Definition ist dabei Folgende: z^e fällt wie z^i mit der Rotationsachse der Erde zusammen, x^e weist auf den Schnittpunkt von Äquator und Nullmeridian und y^e weist auf den Schnittpunkt des Äquators mit dem 90. Längengrad. Das erdfeste Koordinatensystem rotiert bezüglich des i-frame um die z^e -Achse mit der Winkelgeschwindigkeit Ω , der Erddrehrate. Dieses erdfeste Koordinatensystem wird auch als “Earth Centered, Earth Fixed” (ECEF) Koordinatensystem bezeichnet. Anschaulich entspricht die Erde in diesem Koordinatensystem einem Globus, der die Erdoberfläche als gekrümmte Ebene darstellt.

Notation

Aufgrund der unterschiedlichen Koordinatensysteme und der verschiedenen Möglichkeiten, die Beschleunigungen \vec{a} , Geschwindigkeiten \vec{v} und Drehraten $\vec{\omega}$ der Koordinatensysteme zueinander anzugeben, ist es nötig, eine eindeutige Nomenklatur einzuführen. Hierzu werden drei Indizes benötigt wie am Beispiel der Beschleunigung

$$\vec{a}_{ib}^n$$

gezeigt werden soll: Der obere Index gibt das Koordinatensystem an, in dem die Beschleunigung gegeben ist. Hier wird mit dem Index n z. B. angegeben, dass es sich um eine Beschleunigung im Navigationskoordinatensystem handelt. Die beiden unteren Indizes geben an, dass es sich hier um eine Beschleunigung des körperfesten Koordinatensystems (b-frame) bezüglich des Inertialkoordinatensystems (i-frame) handelt.

Positionsangabe

Die Angabe der Position des Ursprungs von b- und n-frame, also der Position des Objekts, ist nur in Koordinaten bezüglich des erdfesten Koordinatensystems sinnvoll. Normalerweise möchte man wissen, an welcher Position auf der Erde man sich befindet. Da außerdem das Navigationskoordinatensystem, an der jeweiligen Position, als Tangentialebene an die Erdoberfläche definiert ist, sind Koordinaten im Navigationskoordinatensystem nicht eindeutig. Eine Angabe in Koordinaten des Inertialkoordinatensystems ist auch nicht eindeutig, da dort die Erdrotation nicht berücksichtigt ist. Die Angabe der Position auf der Erde kann in ECEF Koordinaten erfolgen. Eine gängigere Angabe ist die Angabe von Breitengrad φ , Längengrad λ und Höhe h bezüglich des Erdellipsoids. Dies wird auch als LLH (Latitude, Longitude, Height) bezeichnet. Abbildung 2.3 zeigt den Erdellipsoid und die Koordinatenangabe in LLH-Koordinaten.

Um eine eindeutige mathematische Zuordnung von ECEF Koordinaten zu LLH Koordinaten zu ermöglichen, muss ein Erdmodell definiert werden. Ein solches Modell ist im “World Geodetic System of 1984” (WGS-84) angegeben. Auf eine genaue Beschreibung des Modells und der Umrechnungsformeln zwischen ECEF und LLH Koordinaten wird im Rahmen dieser Arbeit verzichtet. Hierzu sei auf die Literatur wie z. B. ([WEN07] S. 30ff) verwiesen. Das in dieser Arbeit verwendete Modell soll ebenfalls

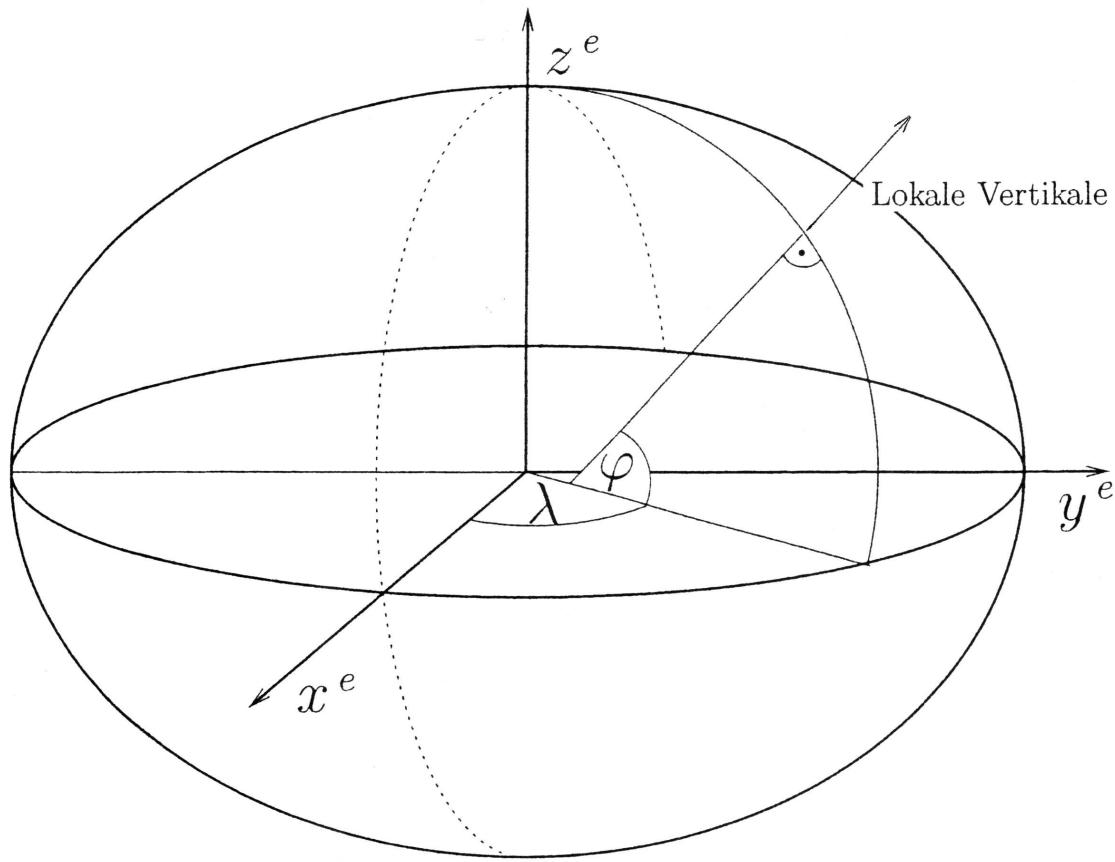


Abbildung 2.3: Erdellipsoid und LLH-Koordinaten (Quelle: [WEN07])

WGS-84 sein. Einige wichtige Parameter, die dieses Modell jeweils in Abhängigkeit vom Breitengrad φ und der Höhe h beschreiben seien dennoch genannt:

- R_n : Krümmungsradius in Nord-Süd Richtung
- R_e : Krümmungsradius in Ost-West Richtung
- R_0 : durchschnittlicher Krümmungsradius $\sqrt{R_n R_e}$
- \vec{g}_l^n : Schwerkraftbeschleunigung
- Erddrehrate: $\Omega = 7.292115 \times 10^{-5} \frac{\text{rad}}{\text{s}}$

Bei der Definition der Höhe h ist darauf zu achten, dass diese sich auf die Höhe über dem Ellipsoid bezieht und in dieser Arbeit positiv nach oben gezählt wird. Teilweise wird die Höhe auch positiv nach unten gezählt, so z. B. in [WEN07]. Außerdem existiert eine alternative Angabe der Höhe: die Höhe über dem Geoid, welche vereinfacht gesagt der Höhe über dem durchschnittlichen Meeresspiegel am jeweiligen Ort entspricht.

Höhendefinitionen

Zur Angabe einer Höhe existieren verschiedene Möglichkeiten. Abbildung 2.4 zeigt den Vergleich zwischen den unterschiedlichen Höhendefinitionen.

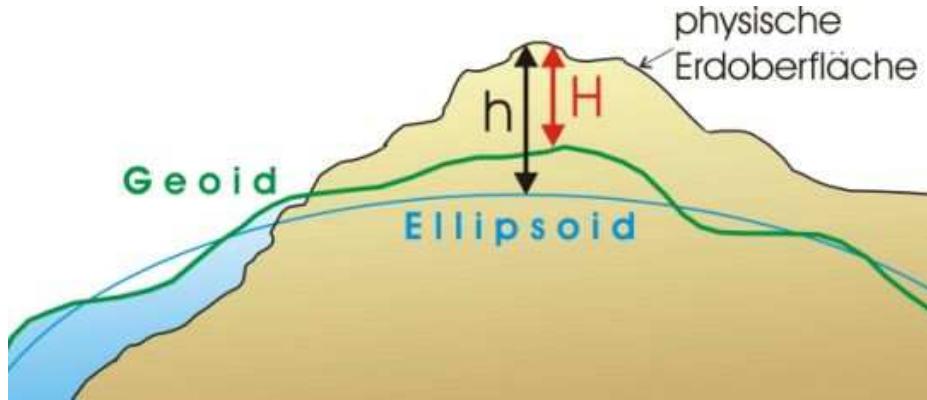


Abbildung 2.4: Vergleich: Ellipsoid und Geoid (Quelle: [TUM])

Höhe über dem Ellipsoid: Um eine einfache Positions berechnung durchführen zu können, ist nötig ein mathematisches Modell der Erde zu definieren, wie dies im WGS-84 der Fall ist. Hierbei wird die Form der Erde durch ein Ellipsoid approximiert, welches den idealen Meeresspiegel annähert. Die Höhe über dem Ellipsoid bezieht sich damit auf die Höhe über einem idealen Meeresspiegel. Ein Problem hierbei ist jedoch, dass dieses Modell den Meeresspiegel alleine durch Gravitation und Zentrifugalkraft, der rotierenden Erde, definiert. Eine ungleiche Massenverteilung auf der Erde beeinflusst den Meeresspiegel allerdings. Somit entspricht die Höhe über dem Ellipsoid im allgemeinen nicht der wahren Höhe über dem Meeresspiegel. Ein GPS-Empfänger bestimmt die Höhe über dem Ellipsoid.

Höhe über dem Geoid: Um eine genauere Angabe der Höhe zu ermöglichen, ist ein Modell nötig, welches die lokalen Abweichungen des Meeresspiegels berücksichtigt. Ein solches Modell ist das Geoid. Die Abweichungen zwischen dem Geoid und dem Ellipsoid betragen zwischen -106 m und +85 m. Ein gängiges Geoid ist das EGM96 [EGM96]. Die meisten GPS-Empfänger verfügen über ein Geoid-Modell und können somit auch eine angenähere Höhe über dem Meeresspiegel ausgeben. Ein Problem hierbei sind allerdings die unterschiedlich genau implementierten Modelle. Somit können sich die ausgegebenen Höhen über dem Meeresspiegel verschiedener GPS-Empfänger unterscheiden.

Geopotentielle Höhe: Die geopotentielle Höhe h_{geop} wird definiert durch den Quotienten der Energie, die benötigt wird, um ein Objekt von Meereshöhe auf die geometrische Höhe \tilde{h} zu heben und einer als konstant angenommenen Schwerkraftbeschleunigung ([USA76] S. 8), mit dem Betrag g_0 an der jeweiligen Position auf Meereshöhe:

$$h_{geop} = \frac{1}{g_0} \int_0^{\tilde{h}} g(h) dh. \quad (2.3)$$

Diese Definition erlaubt es z. B. in der Herleitung der barometrischen Höhenformel, die von der Höhe abhängige Schwerkraftbeschleunigung gegen einen konstanten Wert zu ersetzen. Wertet man die Gleichung (2.3) aus, erhält man

$$h_{geop} = \frac{R_0 \cdot \tilde{h}}{R_0 + \tilde{h}}, \quad (2.4)$$

mit R_0 als durchschnittlichem Krümmungsradius der Erde, an der jeweiligen Position.

2.3 Rechenregeln

Für die Strapdown-Rechnung werden zunächst einige Rechenregeln eingeführt. Eine ausführliche Auflistung kann ([WEN07] S. 34ff) entnommen werden.

Richtungsvektortransformation

Für die Transformation eines Richtungsvektors von einem Koordinatensystem in ein anderes wird eine Transformationsmatrix benötigt:

$$\vec{a}_{ib}^n = \mathbf{C}_b^n \cdot \vec{a}_{ib}^b. \quad (2.5)$$

\mathbf{C}_b^n ist hierbei die Rotationsmatrix, auch Richtungskosinusmatrix (engl. Direction Cosine Matrix (DCM)) genannt, welche die Verdrehung der Koordinatensysteme zueinander beschreibt. Der untere Index gibt an von welchem Koordinatensystem transformiert wird und der obere in welches Koordinatensystem transformiert wird. In diesem Fall vom b-frame in den n-frame.

Eulerwinkel und Richtungskosinusmatrix

Um zu beschreiben, welche Drehungen nötig sind, um ein Koordinatensystem in ein anderes zu überführen, werden die drei Eulerwinkel roll, pitch und yaw, eingeführt. Der Yaw-Winkel ψ beschreibt hierbei die erste Drehung des Koordinatensystems um die z-Achse, der Pitch-Winkel θ beschreibt die zweite Drehung um die neue y-Achse nach der ersten Drehung und der Roll-Winkel ϕ beschreibt die dritte Drehung um die neue x-Achse nach der zweiten Drehung. Hierbei ist die Reihenfolge der Drehungen wichtig. Die DCM \mathbf{C}_b^n kann nun in Abhängigkeit der Eulerwinkel angegeben werden. Die Eulerwinkel beschreiben in diesem Fall die Drehung vom n-Frame in den b-frame wie in Abbildung 2.5 dargestellt.

$$\mathbf{C}_b^n = \begin{pmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta s\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix}. \quad (2.6)$$

Hierbei bedeuten: $sx = \sin(x)$ und $cx = \cos(x)$ mit $x \in [\psi, \theta, \phi]$. Bei der DCM handelt es sich um eine orthonormale Matrix. Daher lässt sich schreiben:

$$\mathbf{C}_n^b = \mathbf{C}_b^{n,-1} = \mathbf{C}_b^{n,T}. \quad (2.7)$$

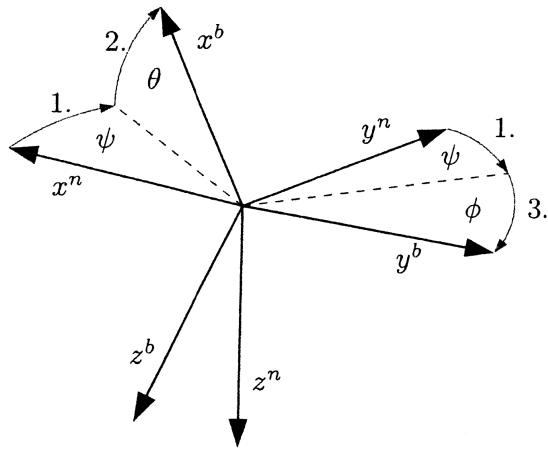


Abbildung 2.5: Eulerwinkel (Quelle: [WEN07])

Auch eine Verkettung von mehreren DCMs ist möglich, um aufeinanderfolgende Drehungen auszudrücken:

$$\mathbf{C}_b^n = \mathbf{C}_e^n \mathbf{C}_b^e. \quad (2.8)$$

Orientierungsvektor und Quaternionen

Eine weitere Möglichkeit die Orientierung zweier Koordinatensysteme zueinander zu beschreiben, ist der Orientierungsvektor

$$\vec{\sigma} = \begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{pmatrix}, \quad (2.9)$$

mit $\sigma = |\vec{\sigma}|$. Der Orientierungsvektor gibt dabei die Achse im Raum an, um die ein Koordinatensystem gedreht werden muss, um mit einer einzigen Drehung in das andere überführt zu werden. Der Betrag des Orientierungsvektors entspricht dabei dem Betrag des Winkels, um den das Koordinatensystem gedreht werden muss.

Der Orientierungsvektor wird meist als ein auf die Länge $|\mathbf{q}| = 1$ normiertes Quaternion [QUAT] gespeichert:

$$\mathbf{q}_b^n = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \cos(\sigma/2) \\ (\sigma_x/\sigma) \sin(\sigma/2) \\ (\sigma_y/\sigma) \sin(\sigma/2) \\ (\sigma_z/\sigma) \sin(\sigma/2) \end{pmatrix}. \quad (2.10)$$

Wie bei DCMs ist auch bei den Quaternionen eine Verkettung möglich, um aufeinanderfolgende Drehungen auszudrücken:

$$\mathbf{q}_b^n = \mathbf{q}_e^n \bullet \mathbf{q}_b^e. \quad (2.11)$$

Hierzu wird die Quaternionenmultiplikation benötigt, welche sich für zwei Quaternionen $\mathbf{q}_1(a, b, c, d)$ und $\mathbf{q}_2(e, f, g, h)$ als Multiplikation einer Matrix mit einem Vektor

darstellen lässt:

$$\mathbf{q}_1 \bullet \mathbf{q}_2 = \begin{pmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{pmatrix} \cdot \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}. \quad (2.12)$$

Quaternionen, DCMs und Eulerwinkel lassen sich ineinander umrechnen. Für die entsprechenden Umrechnungsformeln sei wieder auf die Literatur wie z. B. ([WEN07] S. 40ff) verwiesen.

2.4 Strapdown-Rechnung

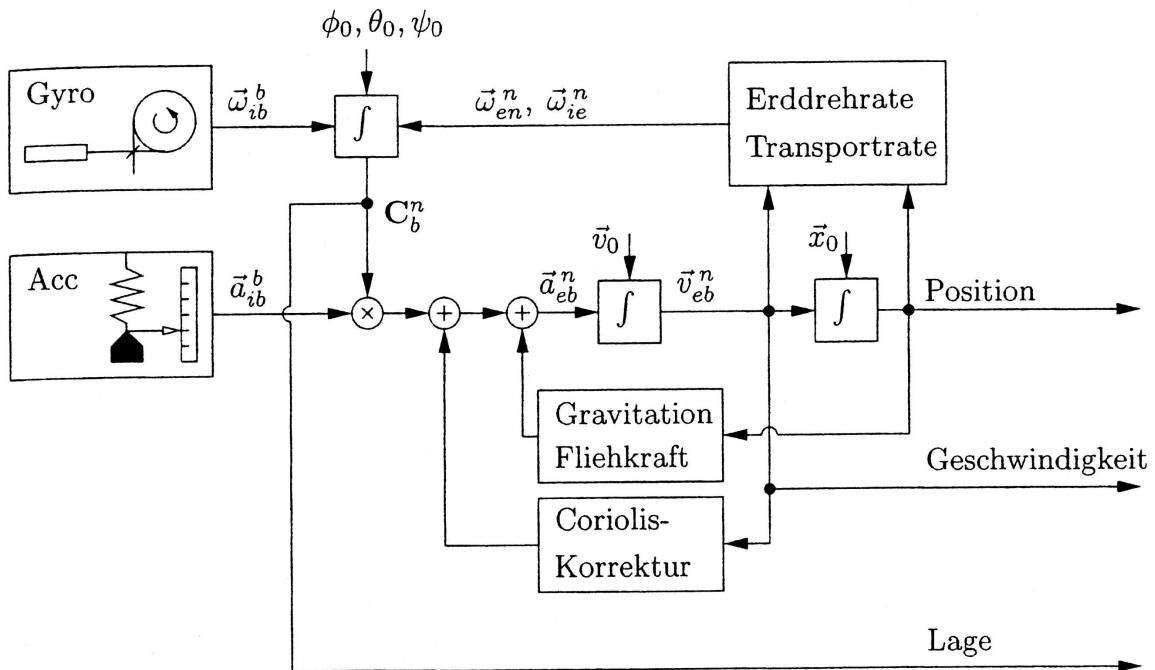


Abbildung 2.6: Strapdown-Algorithmus (Quelle: [WEN07])

Der Strapdown-Algorithmus ist eine iterative Rechenvorschrift, die beschreibt wie man aus den gemessenen Drehraten und Beschleunigungen der IMU sowie der Navigationslösung aus dem vorherigen Zeitschritt zu einer neuen Navigationslösung im aktuellen Zeitschritt kommt. Der Algorithmus lässt sich dabei in drei Schritte unterteilen: Im ersten Schritt wird mit den gemessenen Drehraten durch Integration die neue Orientierung berechnet. Im zweiten Schritt wird mit der berechneten Orientierung aus den gemessenen Beschleunigungen durch Integration die neue Geschwindigkeit berechnet. Schließlich wird durch Integration der Geschwindigkeit die neue Position berechnet. Abbildung 2.6 zeigt den prinzipiellen Aufbau des Strapdown-Algorithmus.

2.4.1 Orientierungsberechnung

Der erste Schritt in der Strapdown-Rechnung ist die Bestimmung der Orientierung. Hierzu ist es nötig, die Änderung der Orientierung in Abhängigkeit von den gemessenen Drehraten zu bestimmen. Möglich ist dabei die Beschreibung der Änderung der Eulerwinkel, des Orientierungsvektors oder des Quaternions ([WEN07] S. 38f und S. 42). Zur Bestimmung der Änderung soll hier der Orientierungsvektor betrachtet werden. Dazu ist es nötig die Bortz'sche Orientierungsvektordifferentialgleichung [BORTZ]

$$\dot{\vec{\sigma}} = \vec{\omega}_{nb}^b + \frac{1}{2}\vec{\sigma} \times \vec{\omega}_{nb}^b + \frac{1}{\sigma^2} \left(1 - \frac{\sigma \sin(\sigma)}{2(1 - \cos(\sigma))} \right) \vec{\sigma} \times (\vec{\sigma} \times \vec{\omega}_{nb}^b) \quad (2.13)$$

zu lösen.

Bei entsprechend hoher Update-Rate, so dass die Drehungen innerhalb eines Zeitintervalls entsprechend klein sind, ist es möglich, die Gleichung wie folgt zu vereinfachen ([TIT04] S. 315ff), ([WEN07] S. 45ff):

$$\dot{\vec{\sigma}} = \vec{\omega}_{nb}^b + \frac{1}{2}\vec{\sigma} \times \vec{\omega}_{nb}^b, \quad (2.14)$$

wobei $\vec{\omega}_{nb}^b$ die Drehrate des Objektes im n-frame ist, also die Drehrate des Objektes ohne Erdrehrate sowie Transportrate. Die Annahme einer ausreichend hohen Update-Rate ist sowieso notwendig, da Drehungen nicht kommutativ sind. Werden Drehungen verkettet, hängt die Orientierung auch von der Reihenfolge der Drehungen ab. Die Reihenfolge kann eine IMU allerdings nicht liefern. Durch eine hohe Update-Rate werden hierdurch verursachte Fehler vermindert, da die in einem Zeitschritt stattfindenden Drehungen klein sind.

In einer weiteren Vereinfachung wird auch der Kreuzterm vernachlässigt. Eine Berücksichtigung dieses Terms wäre möglich, wenn ein Update der Orientierung mit einer geringeren Update-Rate durchgeführt wird als die Daten von der IMU geliefert werden. Darauf wird hier aber verzichtet. Es ergibt sich nach der Integration nun folgende Näherung:

$$\Delta\vec{\sigma}_k \approx \vec{\omega}_{nb,k}^b \cdot \Delta t, \quad (2.15)$$

wobei $\Delta\vec{\sigma}_k$ die Änderung des Orientierungsvektors ist und Δt die Dauer des Zeitschrittes. k gibt den k -ten Zeitschritt an.

Mit dieser Änderung des Orientierungsvektors ist es nun möglich, ein Update des Orientierungsquaternion auszuführen. Hierzu wird ein Korrekturquaternion \mathbf{r}_k nach Gleichung (2.10) berechnet, welches die im betrachteten Zeitintervall erfolgte Drehung darstellt:

$$\mathbf{r}_k = \begin{pmatrix} \cos\left(\frac{\Delta\sigma_k}{2}\right) \\ \frac{\Delta\vec{\sigma}_k}{\Delta\sigma_k} \sin\left(\frac{\Delta\sigma_k}{2}\right) \end{pmatrix}, \quad (2.16)$$

wobei $\Delta\sigma_k = |\Delta\vec{\sigma}_k|$. Das aktuelle Quaternion erhält man durch Quaternionenmultiplikation nach Gleichung (2.12) anschließend mit:

$$\mathbf{q}_{b,k}^n = \mathbf{q}_{b,k-1}^n \bullet \mathbf{r}_k. \quad (2.17)$$

In der hier gewählten Formulierung ist für die Orientierungsberechnung die Drehrate $\vec{\omega}_{nb}^b$ nötig. Die IMU liefert allerdings $\vec{\omega}_{ib}^b$. Es werden also sowohl die Erddrehrate $\vec{\omega}_{ie}^n$ als auch die Drehung des n-frames bei der Bewegung über die gekrümmte Erdoberfläche, Transportrate $\vec{\omega}_{en}^n$ genannt, gemessen. Die benötigte Drehrate $\vec{\omega}_{nb}^b$ kann allerdings wie folgt berechnet werden:

$$\vec{\omega}_{nb}^b = \vec{\omega}_{ib}^b - \mathbf{C}_b^{n,T}(\vec{\omega}_{ie}^n + \vec{\omega}_{en}^n), \quad (2.18)$$

wobei \mathbf{C}_b^n die Orientierung des Objekts im n-Frame beschreibt. Hierbei berechnet sich die gemessene Erddrehrate $\vec{\omega}_{ie}^n$ in Abhängigkeit vom Breitengrad φ und dem Betrag der Erddrehrate Ω wie folgt:

$$\vec{\omega}_{ie}^n = \begin{pmatrix} \Omega \cos(\varphi) \\ 0 \\ -\Omega \sin(\varphi) \end{pmatrix}. \quad (2.19)$$

Die Transportrate $\vec{\omega}_{en}^n$ ist abhängig von der Geschwindigkeit $v_{eb,n}^n$ in Nord- und $v_{eb,e}^n$ Ostrichtung, der Höhe h sowie der Erdkrümmung R_e in Ost-West-Richtung und R_n in Nord-Süd-Richtung. $\vec{\omega}_{en}^n$ lässt sich wie folgt berechnen:

$$\vec{\omega}_{en}^n = \begin{pmatrix} \frac{v_{eb,e}^n}{R_e+h} \\ -\frac{v_{eb,n}^n}{R_n+h} \\ -\frac{v_{eb,e}^n \tan(\varphi)}{R_e+h} \end{pmatrix}. \quad (2.20)$$

Hiermit ist das Update der Orientierung abgeschlossen. Im nächsten Schritt kann nun zum Update der Geschwindigkeit übergegangen werden.

2.4.2 Geschwindigkeitsberechnung

Der zweite Schritt in der Strapdown-Rechnung ist nun die Berechnung der Geschwindigkeit \vec{v}_{eb}^n des Objektes. Die Geschwindigkeit soll hierbei in Koordinaten des Navigationskoordinatensystems berechnet werden. Hierzu ist es nötig, die Änderung der Geschwindigkeit in Abhängigkeit der gemessenen Beschleunigungen \vec{a}_{ib}^b zu beschreiben. Da die Geschwindigkeit in einem raumfesten Koordinatensystem bestimmt werden soll, muss die Abhängigkeit von der Orientierung berücksichtigt werden. Zusätzlich zu den eigentlichen Geschwindigkeitsänderungen misst der Beschleunigungssensor auch die Schwerbeschleunigung \vec{g}_l^n der Erde sowie Coriolis-Beschleunigungen, verursacht durch die Erddrehung und die Drehung des Navigationskoordinatensystems bei Bewegung über die gekrümmte Erdoberfläche. Eine Geschwindigkeitsdifferentialgleichung, die diese Effekte berücksichtigt lautet wie folgt:

$$\dot{\vec{v}}_{eb}^n = \mathbf{C}_b^n \vec{a}_{ib}^b - (2\vec{\omega}_{ie}^n + \vec{\omega}_{en}^n) \times \vec{v}_{eb}^n + \vec{g}_l^n. \quad (2.21)$$

Eine Herleitung dieses Zusammenhangs kann ([WEN07] S. 53f) entnommen werden. Um nun die Geschwindigkeitsänderung in einem Zeitintervall zu erhalten, wird die rechte Seite von Gleichung (2.21) integriert. Während der Kreuzterm und die Erdbeschleunigung in dem betrachteten Zeitintervall als konstant angenommen werden können, ist es nötig, den Term $\mathbf{C}_b^n \vec{a}_{ib}^b$ genauer zu betrachten. Ein vereinfachter Ausdruck

des Integrals über diesen Term ist Folgender:

$$\int_{t_{k-1}}^{t_k} \mathbf{C}_b^n(t) \cdot \vec{a}_{ib}^b dt \approx \mathbf{C}_{b,k-1}^n \left[\vec{a}_{ib,k}^b \cdot \Delta t + \frac{1}{2} \Delta \vec{\sigma}_k \times \vec{a}_{ib,k}^b \cdot \Delta t \right]. \quad (2.22)$$

Die genau Herleitung und ein Ausdruck für den Fall dass ein Update mit geringerer Rate gemacht wird als Daten von der IMU zur Verfügung stehen, kann ebenfalls ([WEN07] S. 55ff) und ([TIT04] S. 325ff) entnommen werden. Zusammen ergibt dies den folgenden Ausdruck:

$$\Delta \vec{v}_{eb,k}^n = \left(\mathbf{C}_{b,k-1}^n \left[\vec{a}_{ib,k}^b + \frac{1}{2} \Delta \vec{\sigma}_k \times \vec{a}_{ib,k}^b \right] - ((2\vec{\omega}_{ie,k-1}^n + \vec{\omega}_{en,k-1}^n) \times \vec{v}_{eb,k-1}^n + \vec{g}_{l,k-1}^n) \right) \cdot \Delta t. \quad (2.23)$$

Die aktuelle Geschwindigkeit erhält man nun mit

$$\vec{v}_{eb,k}^n = \vec{v}_{eb,k-1}^n + \Delta \vec{v}_{eb,k}^n. \quad (2.24)$$

Hiermit ist nun auch das Update der Geschwindigkeit abgeschlossen. Im letzten Schritt kann nun zum Update der Position übergegangen werden.

2.4.3 Positions berechnung

Der letzte Schritt der Strapdown-Rechnung ist schließlich das Update der Position. Hierzu wird nun die Geschwindigkeit integriert. Wie am Ende von Abschnitt 2.2 erwähnt, macht es eigentlich nur Sinn die Position in erdfesten Koordinaten anzugeben. Hier wird die Darstellung in LLH-Koordinaten gewählt. Eine Angabe der Position in Koordinaten des Navigationskoordinatensystems ist nicht sinnvoll, da sich die Orientierung des Navigationskoordinatensystems je nach Position ändert. Mögliche Gleichungen zum Update der Position \vec{p} sind Folgende:

$$\dot{\varphi} = \frac{v_{eb,n}^n}{R_n(\varphi) + h} \quad (2.25)$$

$$\dot{\lambda} = \frac{v_{eb,e}^n}{(R_e(\varphi) + h) \cos(\varphi)} \quad (2.26)$$

$$\dot{h} = -v_{eb,d}^n. \quad (2.27)$$

Im Zeitdiskreten können diese wie folgt implementiert werden:

$$\Delta \vec{p}_k^n = \vec{v}_{eb,k}^n \cdot \Delta t \quad (2.28)$$

$$\varphi_k = \varphi_{k-1} + \frac{\Delta p_{n,k}^n}{R_n(\varphi_{k-1}) + h} \quad (2.29)$$

$$\lambda_k = \lambda_{k-1} + \frac{\Delta p_{e,k}^n}{(R_e(\varphi_{k-1}) + h) \cos(\varphi_{k-1})} \quad (2.30)$$

$$h_k = h_{k-1} - \Delta p_{d,k}^n, \quad (2.31)$$

wobei $\Delta \vec{p}_k^n$ die Änderung der Position im Navigationskoordinatensystem darstellt, mit den Komponenten $\Delta p_{n,k}^n$, $\Delta p_{e,k}^n$ und $\Delta p_{d,k}^n$ in NED-Koordinaten. Für räumlich sehr

begrenzte Bereiche lässt sich auch näherungsweise eine Position \vec{p}_k^n im Navigationskoordinatensystem angeben. Hierzu kann folgende Darstellung verwendet werden:

$$\vec{p}_k^n = \vec{p}_{k-1}^n + \Delta \vec{p}_k^n. \quad (2.32)$$

Es ist hier aber auf jeden Fall darauf zu achten, dass diese Darstellung nur eine Näherung ist, da sie die Erdkrümmung nicht berücksichtigt. Hierbei handelt es sich um eine Tangentialebene an die Erde im Ursprung des verwendeten Navigationskoordinatensystems. Hierzu sei auch auf Abbildung 2.2 und die Definition der Koordinatenachsen des n-frame in Abschnitt 2.2 verwiesen. Für einen Bereich von wenigen Kilometern ist diese Näherung aber noch ausreichend. In dieser Arbeit werden die Positionen jedoch im erdfesten Koordinatensystem (e-frame) betrachtet.

Hiermit ist der letzte Schritt des Strapdown-Algorithmus abgeschlossen und die Berechnung kann für den nächsten Zeitschritt wieder von vorne beginnen. Der Strapdown-Algorithmus ist in der Matlab-Funktion “strapdown.m” implementiert worden.

3 Barometrische Höhenberechnung

Die barometrische Höhenberechnung oder Höhenmessung ist ein Verfahren zur Bestimmung der absoluten Höhe durch die Messung des Luftdrucks. Sie hat den Vorteil dass sie kostengünstig, einfach und schnell durchzuführen ist und trotzdem eine gute Genauigkeit liefert. Im Gegensatz zur Höhenmessung mit einem GPS-Empfänger liefert die barometrische Höhenberechnung meist genauere Höhenwerte. Verglichen mit einer Höhenbestimmung per Triangulation ist die barometrische Höhenmessung schneller und günstiger durchgeführt.

3.1 Grundlagen

Grundlage der barometrischen Höhenberechnung ist, wie bereits erwähnt, die Messung des Luftdrucks. Mit steigender Höhe nimmt dabei der Luftdruck ab, mit abnehmender Höhe steigt der Luftdruck an. Mit Hilfe eines Barometers ist es möglich diesen Luftdruck zu messen und daraus eine Höhe zu bestimmen. Der hierfür verwendete Luftdrucksensor wird im Kapitel 4 vorgestellt. Die benötigte Rechenvorschrift, die eine Umrechnung zwischen Höhe und Druck erlaubt, soll im nächsten Abschnitt hergeleitet werden.

Die folgenden Formelbuchstaben werden im Rahmen dieser Arbeit in Bezug auf die barometrische Höhenberechnung verwenden:

- M : Molare Masse der Luft ($M = 0,028964 \frac{\text{Kg}}{\text{mol}}$)
- R : Universelle Gaskonstante nach U.S. Standardatmosphäre ($R = 8,31432 \frac{\text{Nm}}{\text{mol}\cdot\text{K}}$)
- T : Absolute Temperatur in Kelvin
- g : Schwerkraftbeschleunigung in $\frac{\text{m}}{\text{s}^2}$ abhängig von der geographischen Breite und Höhe
- g_0 : Schwerkraftbeschleunigung in $\frac{\text{m}}{\text{s}^2}$ jeweils auf Meereshöhe
- P : Druck in Pa
- h : Höhe in Meter über dem lokalen mittleren Meeresspiegel
- γ : Temperaturgradient ($\gamma = -0,0065 \frac{\text{K}}{\text{m}}$)

3.2 Herleitung der barometrischen Höhenformel

Um eine Bestimmung der Höhe aus dem gemessenen Druck durchführen zu können soll in diesem Abschnitt der Zusammenhang zwischen gemessenem Druck und Höhe hergeleitet werden. Ausgangspunkt der Herleitung ist die hydrostatische Grundgleichung ([USA76] S. 6). Die hydrostatische Grundgleichung beschreibt die Änderung von Druck und Dichte mit der Höhe. Zur Herleitung der hydrostatischen Grundgleichung betrachtet man ein quaderförmiges Volumenelement dV mit der infinitesimalen Höhe dh und der Grundfläche dA . Von unten wirkt auf die Fläche dA nur eine, durch den von der Höhe abhängigen Atmosphärendruck P verursachte Kraft

$$F_u = P(h) \cdot dA. \quad (3.1)$$

Von oben wirkt auf die Fläche ein um den Betrag dP verschiedener Druck und damit die Kraft

$$F_o = (P(h) + dP) \cdot dA. \quad (3.2)$$

Zusätzlich hierzu wirkt von oben die Gewichtskraft F_g der in dem Volumen $dV = dA \cdot dh$ enthaltenen Luftmasse dm mit der Dichte ρ :

$$F_g = dm \cdot g(h) = \rho(h) \cdot dV \cdot g(h) = \rho(h) \cdot dh \cdot dA \cdot g(h). \quad (3.3)$$

Die Summe aller Kräfte muss im Gleichgewicht nun Null ergeben:

$$F_u + F_o + F_g = 0 \quad (3.4)$$

Einsetzen von (3.1), (3.2) und (3.3) sowie Kürzen und Umstellen liefert dabei

$$P(h) \cdot dA - \rho(h) \cdot dh \cdot dA \cdot g(h) - (P(h) + dP) \cdot dA = 0 \quad (3.5)$$

$$-\rho(h) \cdot dh \cdot g(h) - dP = 0 \quad (3.6)$$

$$\frac{dP}{dh} = -\rho(h)g(h). \quad (3.7)$$

Aus der idealen Gasgleichung ergibt sich außerdem für die Dichte ρ der Luft:

$$\rho(h) = \frac{P(h) \cdot M}{R \cdot T(h)}. \quad (3.8)$$

Auch wenn die ideale Gasgleichung an dieser Stelle nur eine gute Näherung darstellt und nicht die Luftfeuchtigkeit berücksichtigt, kann sie hier verwendet werden. Einsetzen von Gleichung (3.8) in Gleichung (3.7) liefert nun die Ausgangsgleichung zur Herleitung der barometrischen Höhenformel:

$$\frac{dP}{dh} = -\frac{P(h) \cdot g(h) \cdot M}{R \cdot T(h)}. \quad (3.9)$$

Für die weitere Herleitung ist es nötig, einige Annahmen zu treffen, um die Herleitung zu vereinfachen. Wie bereits in der Liste der Formelbuchstaben zu erkennen, sind die molare Masse der Luft und die universelle Gaskonstante unabhängig von der Höhe. Zusätzlich wird nun die Schwerkraft g , unabhängig von der Höhe, auf den

Wert g_0 auf Meereshöhe am jeweiligen Ort gesetzt. Diese Vereinfachung erlaubt es später, einfacher über die Höhe zu integrieren.

Die hiermit bestimmte Höhe wird als geopotentielle Höhe h_{geop} bezeichnet. Die Umrechnung in eine geometrische Höhe h_{geom} ist durch Umstellen von Gleichung (2.3) möglich ([USA76] S. 8):

$$h_{geom} = \left(\frac{R_0 \cdot h_{geop}}{R_0 - h_{geop}} \right). \quad (3.10)$$

Bis zu einer Höhe von 3000 m liegt der Unterschied zwischen der geometrischen Höhe und der geopotentiellen Höhe allerdings nahe der Messgenauigkeit von 1 m, der für diese Arbeit eingesetzten Sensoren. Außerdem ist die Änderung dieser Abweichung mit der Höhe nur gering, so dass die Abweichung im Gegensatz zu anderen Fehlern kaum einen Einfluss hat und durch die spätere Fehlerfilterung mit korrigiert wird. Aus diesem Grund wird auf die Umrechnung verzichtet und $h_{geop} = h_{geom} = h$ gesetzt.

Schließlich ist es noch nötig, für die Temperatur T passende Annahmen zu treffen, um die Abhängigkeit von der Höhe zu beschreiben. Eine gute Näherung für den Temperaturverlauf ist dabei ein linearer Verlauf. Ein solcher Verlauf wird auch in vielen Modellen für eine Standardatmosphäre angenommen. In ([USA76] S. 10) wird dabei folgender Zusammenhang angegeben:

$$T(h) = T(h_0) + \gamma \cdot (h - h_0), \quad (3.11)$$

wobei γ eine mit der Höhe abnehmende Temperatur beschreibt. Diese Annahme gilt für den Bereich von 0 km bis 11 km Höhe. $T(h_0)$ ist dabei die Temperatur auf Referenzhöhe. h_0 ist die Referenzhöhe und wird für den hier betrachteten Bereich von 0 km bis 11 km zu Null gesetzt. Für Bereiche oberhalb von 11 km existieren bis zu einer Höhe von 85 km weitere Bereiche für die jeweils der Temperaturgradient und die Referenzhöhe definiert sind. Diese Bereiche sollen in dieser Arbeit aber nicht weiter betrachtet werden.

Setzt man nun die oben gemachten Annahmen in die Gleichung (3.9) ein, ergibt sich Folgendes:

$$\frac{dP}{dh} = -\frac{P(h) \cdot g_0 \cdot M}{R \cdot (T(h_0) + \gamma \cdot (h - h_0))} \quad (3.12)$$

$$\Rightarrow \int_{P(h_0)}^{P(h_1)} \frac{dP}{P(h)} = - \int_{h_0}^{h_1} \frac{g_0 \cdot M}{R \cdot (T(h_0) + \gamma \cdot (h - h_0))} dh. \quad (3.13)$$

Nun gelte $P(h_1) := P(h)$ und $h_1 := h$. Damit folgt:

$$\Rightarrow \ln \left(\frac{P(h)}{P(h_0)} \right) = -\frac{M \cdot g_0}{R} \cdot \frac{1}{\gamma} \ln \left(\frac{T(h_0) + \gamma \cdot (h - h_0)}{T(h_0)} \right). \quad (3.14)$$

Das Anwenden der Exponentialfunktion auf beiden Seiten der Gleichung (3.14) und Umstellen nach $P(h)$ ergibt nun die barometrische Höhenformel, welche den Druck P

in Abhängigkeit von der Höhe h beschreibt:

$$P(h) = P_0 \left[\frac{T_0}{T_0 + \gamma \cdot (h - h_0)} \right]^{\frac{M \cdot g_0}{R \cdot \gamma}} \quad (3.15)$$

mit $P_0 = P(h_0)$ und $T_0 = T(h_0)$. Dieser Ausdruck lässt sich außerdem leicht nach h auflösen und man erhält einen Ausdruck, der die Höhe h in Abhängigkeit vom Druck P beschreibt. Es ergibt sich:

$$h(P) = h_0 + \left[\left(\frac{P_0}{P} \right)^{\frac{R \cdot \gamma}{M \cdot g_0}} - 1 \right] \cdot \frac{T_0}{\gamma}. \quad (3.16)$$

Mit den Gleichungen (3.11), (3.15) und (3.16) hat man nun zwei Möglichkeiten:

Berechnung von $T(h)$ und $P(h)$: Sind Werte für T_0 , h_0 , P_0 und h gegeben, ist es mit den Gleichungen (3.11) und (3.15) möglich, aus den gegebenen Werten ein Temperaturprofil $T(h)$ und das dazugehörige Druckprofil $P(h)$ für die gegebenen Höhen h zu erzeugen. Eine entsprechende Funktion ist in der Matlab-Datei “gen_T_P_profile.m” implementiert.

Berechnung von $h(P)$: Sind andererseits Werte für T_0 , h_0 , P_0 und $P(h)$ bekannt, ist es mit der Gleichung (3.16) möglich, aus dem Druck die Höhe h zu berechnen. Eine entsprechende Funktion ist in der Matlab-Datei “barheight.m” in leicht abgewandelter Form implementiert.

3.3 Fehlermodelle

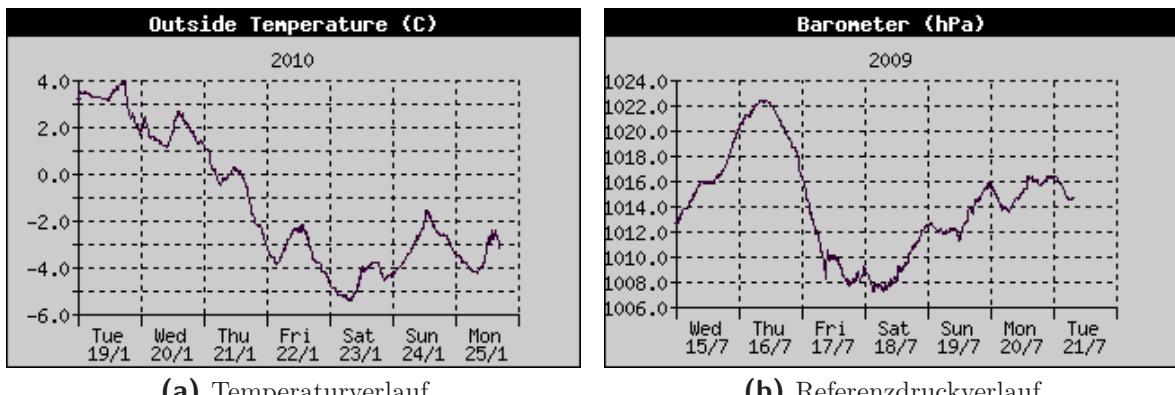


Abbildung 3.1: Messwerte für Temperatur und Druck (Quelle: [WETTR])

Mit den Gleichungen aus Abschnitt 3.2 ist jetzt die Möglichkeit gegeben, eine Höhe zu berechnen. Hier sollen nun die Eingangsgrößen, die für eine Höhenberechnung benötigt werden, genauer betrachtet werden. Zur Auswertung der Gleichung (3.16) werden zuerst eine Referenzhöhe h_0 sowie die Temperatur T_0 und der Druck P_0 auf der Referenzhöhe h_0 benötigt. Während es möglich ist, eine allgemein gültige und konstante

Referenzhöhe h_0 festzulegen, in diesem Fall $h_0 = 0$ m, ist es nicht möglich, konstante Werte für T_0 und P_0 anzugeben. Die Höhe eines Ortes ändert sich normalerweise nicht und ist an bekannten Referenzpunkten bestimmbar, allerdings ändert sich naturgegeben die Temperatur T an einem Ort mit der Zeit. Nachmittags ist es meist wärmer als früh am morgen. Auch der Druck P ändert sich über die Zeit. Während eines schönen Sommertages kann der Luftdruck höher sein als an einem schlechten Wintertag. Auch eine Ortsabhängigkeit der Temperatur und des Drucks kann gegeben sein. Schließlich herrschen nicht überall dieselben Wetterverhältnisse. Die Abbildungen 3.1 sollen den Effekt der zeitveränderlichen Parameter verdeutlichen. Sie zeigen den Temperaturverlauf und den Druckverlauf an einem festen Ort über einen Zeitraum von einer Woche.

Zur Modellierung der Änderung der Referenztemperatur und des Referenzdruckes wird nun ein Random-Walk-Modell gewählt. Temperatur und Druck sollen damit wie folgt dargestellt werden:

$$T_{0,k} = T_{0,k-1} + \sqrt{\Delta t_{baro}} \cdot n_{T_0,k} \quad (3.17)$$

$$P_{0,k} = P_{0,k-1} + \sqrt{\Delta t_{baro}} \cdot n_{P_0,k}, \quad (3.18)$$

wobei k den k -ten Zeitschritt angibt. $n_{T_0,k} \sim \mathcal{N}(0, \sigma_{T_0}^2)$ und $n_{P_0,k} \sim \mathcal{N}(0, \sigma_{P_0}^2)$ sind gaußverteilte Zufallsvariablen mit Mittelwert 0 und der jeweiligen Varianz $\sigma_{T_0}^2$ bzw. $\sigma_{P_0}^2$. Δt_{baro} ist hierbei das Abtastintervall des Barometers. Die Multiplikation mit $\sqrt{\Delta t_{baro}}$ wird durchgeführt um unabhängig vom Abtastintervall den gleichen zeitlichen Verlauf der Änderung, für eine gegebene Varianz, zu erhalten.

Die Varianzen $\sigma_{P_0}^2$ bzw. $\sigma_{T_0}^2$ sind nun so zu wählen dass sie die zu erwartenden Änderungen im beobachteten Zeitraum möglichst gut abbilden. Die Gleichung

$$\sigma_x = \frac{\Delta x}{\sqrt{\Delta t_b}}, \quad (3.19)$$

wobei Δt_b das Beobachtungsintervall ist und $x \in [T_0, P_0]$. Die Gleichung 3.19 gibt dabei den Zusammenhang zwischen der Standardabweichung des treibenden Rauschens σ_x des Random-Walk-Modells und der Standardabweichung der Änderung des Messwertes Δx bezogen auf den Anfangswert im Beobachtungsintervall, nach einer bestimmten Zeit Δt_b an. Für die in dieser Arbeit durchgeföhrten Betrachtungen wurden mit Gleichung (3.19) die in Tabelle 3.1 aufgelisteten Werte für ein Beobachtungsintervall von 100 Sekunden abgeschätzt. Als Quelle der Daten zum Temperatur- und Druckverlauf

Parameter	Änderung	Standardabweichung	Intervall Δt_b
T_0	$\Delta T_0 = 0.04$ K	$\sigma_{T_0} = 0.004 \frac{\text{K}}{\sqrt{\text{s}}}$	100 s
P_0	$\Delta P_0 = 4$ Pa	$\sigma_{P_0} = 0.4 \frac{\text{Pa}}{\sqrt{\text{s}}}$	100 s

Tabelle 3.1: Standardabweichung der T_0 und P_0 Änderung

an einem festen Ort wurde die Wetterstation der Universität Paderborn [WETTR] verwendet. Hierbei zeigten sich über ein Zeitraum von 7 Tagen Extremwerte in der Änderung des Druckes von 17 hPa in 12 Stunden und der Temperatur von 17 K in 12

Stunden. Umgerechnet auf 100 Sekunden, unter der Annahme dass sich Temperatur und Druck linear ändern, ergibt dies $0.04 \frac{\text{K}}{100 \text{ s}}$ für die Temperaturänderung und $4 \frac{\text{Pa}}{100 \text{ s}}$ für die Druckänderung.

Hierbei zeigt sich allerdings ein Widerspruch: Da hier versucht wird, einen als linear angenommenen Verlauf mit einem Random-Walk-Modell nachzubilden, ergeben sich je nach Beobachtungsintervall unterschiedliche Varianzen für das treibende Rauschen des Modells. Dies führt dazu, dass die mit dem Modell berechnete Varianz eines Wertes, bezogen auf den Startwert, für Werte an Zeitpunkten innerhalb des Beobachtungsintervalls, zu hoch und für Werte außerhalb des Beobachtungsintervalls zu klein ist. Die errechnete Varianz des treibenden Rauschens steigt mit größer werdendem Beobachtungsintervall an. Das Random-Walk-Modell kann daher in diesem Fall nur die Änderung von einem Beobachtungszeitpunkt zum nächsten korrekt darstellen und zu den anderen Zeitpunkten nur Näherungen liefern. Anders herum werden bei einer Simulation die Änderungen eines Wertes zu einem bestimmten Zeitpunkt innerhalb des Beobachtungsintervalls, bezogen auf den Startwert, „zu groß“ und die Änderungen außerhalb des Beobachtungsintervalls „zu klein“ erzeugt.

Dieser scheinbare Widerspruch stellt hier allerdings kein großes Problem dar, da in der späteren Fehlerrechnung nur die Werte zu festen Zeitpunkten und die Unterschiede an aufeinanderfolgenden Zeitpunkten betrachtet werden. In diesem Beispiel z. B. alle 100 Sekunden. Die Näherung innerhalb des Beobachtungsintervalls wird als hinreichend angenommen. Auch wird davon ausgegangen, dass ein erweitertes Modell, welches einen zusätzlichen Zustand ΔT_0 bzw. ΔP_0 zur Modellierung der Änderungsrate nutzt, keine Verbesserung bringt. Bei den in der Praxis vorkommenden Werten wird außerdem davon ausgegangen, dass variable Beobachtungsintervalle keinen allzu großen Einfluss haben. Hinzu kommt, dass für die Bestimmung eines Modells nur eine geringe Datenbasis zur Verfügung stand. Zusätzlich wird der Effekt der Ortsveränderlichkeit bei obiger Problembeschreibung noch außer Acht gelassen, da hierzu keine Untersuchungen gemacht werden konnten. Die Summe der bekannten und evtl. unbekannten Effekte ist mit dem Random-Walk-Modell daher einfacher zu modellieren.

Für die barometrische Höhenberechnung stellt die Veränderlichkeit der Referenzwerte für T_0 und P_0 ein Problem dar. Zum einen ist es nötig, vor der Berechnung der Höhe die Werte für T_0 und P_0 zu initialisieren und zum anderen ist es nötig, die Referenzwerte bei einer Änderung fortlaufend anzupassen. Ohne Anpassung würde das Ergebnis der Höhenberechnung nicht der tatsächlichen Höhe entsprechen. In klassischen Ansätzen wird dabei entweder auf eine Anpassung verzichtet, wie z. B. in der Luftfahrt auf Reiseflughöhe. Oder man führt eine manuelle Anpassung durch, wenn dies nötig ist, wie z. B. vor der Landung auf einem Flughafen. Im Kapitel 7 soll später ein neues Verfahren vorgestellt werden, welches es erlaubt, die Änderungen der Referenzgrößen fortlaufend und automatisch durchzuführen bzw. die dadurch verursachten Fehler zu korrigieren.

Neben den Änderungen der Referenzwerte sind auch die Messwerte für die Temperatur

T und den Druck P wie im Kapitel 4 gezeigt mit einem Rauschen überlagert.

$$\tilde{T}_k = T_k + n_{T,k} \quad (3.20)$$

$$\tilde{P}_k = P_k + n_{P,k} \quad (3.21)$$

stellt die Zusammenhänge zwischen Messung \tilde{x} und dem idealen Wert x mit $x \in [T, P]$ dar, wobei $n_x \sim \mathcal{N}(0, \sigma_x^2)$ gaußverteilte Zufallsvariablen mit Mittelwert 0 und der jeweiligen Varianz σ_x^2 sind. Um eine bessere Höhenschätzung zu erhalten, wird in Kapitel 8 ein Verfahren zur Kombination von einem Barometer mit einem Inertialnavigationssystem vorgestellt.

Die Gleichungen (3.17), (3.18), (3.20) und (3.21) sind in der Matlab-Datei "gen_T_P_err.m" implementiert.

3.4 Datenerzeugung

Für eine Simulationen ist es nötig, entsprechende Daten zu erzeugen. Das Vorgehen zur Erzeugung von Messwerten für Druck und Temperatur soll in diesem Abschnitt erklärt werden. Hierzu werden in Matlab die Dateien "gen_T_P_err.m" und "gen_T_P_profile.m" genutzt.

Für die spätere Simulation der barometrischen Höhenmessung für eine vorgegebene Trajektorie werden Informationen zum Temperatur und Druckverlauf benötigt. Zusätzlich muss die Veränderung der Referenzwerte simuliert werden. Es wird davon ausgegangen, dass sowohl die Trajektorie als auch die Anfangswerte der Referenzparameter vorgegeben sind. Zur Generierung der Daten wird wie folgt vorgegangen:

Initialisierung: Zuerst sind die Anfangswerte für die Referenzparameter zu setzen, sowie die Standardabweichungen zur Fehlermodellierung

Listing 3.1: Initialisierung

```

1 % Barometer: Referenz- und Fehlerparameter
2 H0_baro = 0; % Referenzhöhe in m
3 T0_baro = 288.15; % Absolute Temperatur auf Referenzhöhe in K
4 P0_baro = 101315; % Druck auf Referenzhöhe in Pa
5 std_T0_baro = 0.004; % Standardabweichung der Temperaturänderung
6 % in K/sqrt(s)
7 std_P0_baro = 0.4; % Standardabweichung der Druckänderung
8 % in Pa/sqrt(s)
9 std_P_baro = 10; % Standardabweichung des Messrauschen
10 % für den Druck
11 std_T_baro = 0.1; % Standardabweichung des Messrauschen
12 % für die Temperatur

```

Fehlergenerierung: Im nächsten Schritt werden die additiven Fehlerterme generiert und zu den Referenzparametern hinzugefügt, um die Wetterabhängigkeit zu simulieren. Die Variable `npts` gibt dabei die Anzahl der zu generierenden Werte an und `dt` das Abtastintervall.

Listing 3.2: Fehlergenerierung

```

1 % generiere additive Fehlerterme
2 [T0_offset, P0_offset, T_noise, P_noise] = ...
3     gen_T_P_err(std_T0_baro, std_P0_baro, std_T_baro, std_P_baro, dt, npts);
4
5 % generiere Referenztemperatur- und Druckprofile
6 T0_baro = T0_baro + T0_offset; % addiere Temperaturänderung
7 P0_baro = P0_baro + P0_offset; % addiere Druckänderung

```

Profilerzeugung: Im letzten Schritt werden aus dem vorgegebenen Höhenprofil `height_prof` und den generierten Referenzwerten, die Messwerte für Temperatur und Druck generiert. Zusätzlich wird das Messrauschen zu den idealen Messwerten hinzugaddiert

Listing 3.3: Profilerzeugung

```

1 % Profilerzeugung: Temperatur, Druck
2 [P_baro, T_baro] = gen_T_P_profile(height_prof, H0_baro, T0_baro, P0_baro);
3
4 % Messwerte verrauschen
5 est_P_baro = P_baro + P_noise; % Druck
6 est_T_baro = T_baro + T_noise; % Temperatur

```

Hiermit steht mit `est_P_baro` und `est_T_baro` nun ein Druck und Temperaturprofil für spätere Simulationen zur Verfügung.

3.5 Höhenberechnung

Zur Berechnung einer Höhe aus einem Druckprofil kann nun die Funktion "barheight.m" genutzt werden. Im Gegensatz zur Gleichung (3.16) kann dieser Funktion ein Wert h_1 , wobei $h_1 = h_0$ sein kann, für eine Anfangshöhe und der Wert für die Temperatur auf Anfangshöhe $T_1 = T(h_1)$ sowie der Wert für den Druck auf Anfangshöhe $p_1 = p(h_1)$ übergeben werden. Durch Umstellen von Gleichung (3.11) und (3.15) ist es mit diesen Werten dann möglich, die Werte für T_0 und P_0 zu bestimmen, was in der Funktion durchgeführt wird. Der Grund hierfür ist, dass anstelle von T_0 und P_0 im Allgemeinen an der bekannten Höhe h_1 nur T_1 und P_1 gemessen werden können. Eine Berechnung von T_0 und P_0 wäre also in jedem Fall nötig. Anschließend kann aus den Werten für p eine Höhe h bestimmt werden. Ein Beispiel zur Berechnung eines Höhenprofils aus vorgegebenen Werten kann dabei wie folgt aussehen:

Listing 3.4: Höhenberechnung

```

1 % Anfangshöhe h1 = 170m, Anfangstemperatur T1 = 19.6°C
2 % Anfangsdruck p1 = p(1)
3 h1 = 170;
4 T1 = 19.6 + 273.15;
5
6 % berechne Höhenprofil
7 h = barheight(p, h1, T1, p(1));

```

p ist dabei das vorgegebene Druckprofil und h die Ergebnisse der Höhenberechnung zu den angegebenen Referenzwerten und dem Druckprofil.

3.6 Beispiele

Dieser Abschnitt soll Beispiele zur barometrischen Höhenberechnung und Datenerzeugung zeigen. Dabei wurden die in diesem Kapitel vorgestellten Funktionen benutzt.

Druckprofil und Höhenberechnung

Zuerst sei hier in Abbildung 3.2 ein real aufgenommenes Druckprofil gezeigt. Dieses wurde unter Verwendung eines Barometers wie es in Kapitel 4 vorgestellt werden wird, aufgezeichnet. Zur besseren Darstellung wurden die Messwerte geglättet, daher ist in der Abbildung das Messrauschen des Barometers nicht zu erkennen.

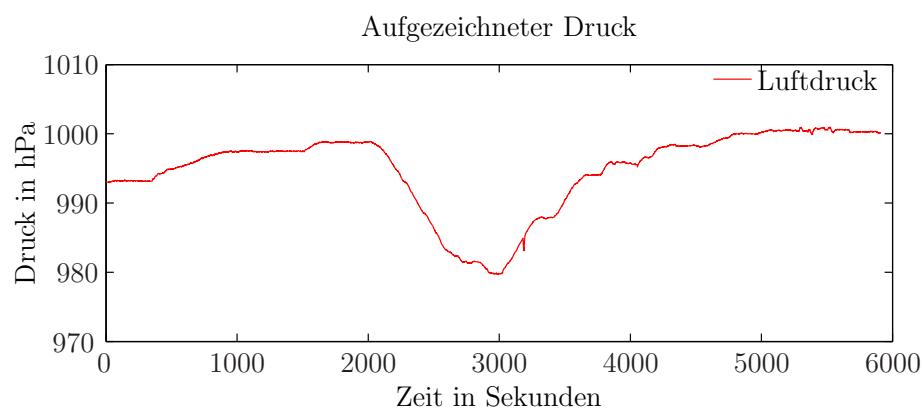


Abbildung 3.2: Aufgezeichneter Druck

Aus dem aufgenommenen Druckprofil kann mit Hilfe der Funktion "barheight.m" eine Höhenprofil berechnet werden, wie es hier in Abbildung 3.3 dargestellt ist.

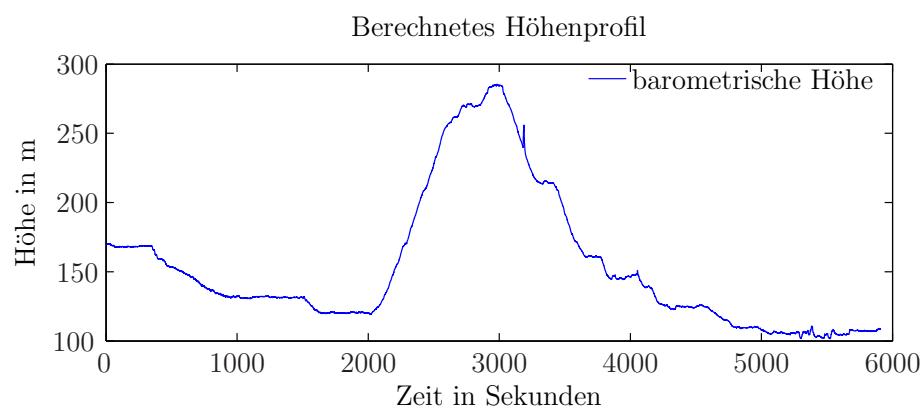


Abbildung 3.3: Berechnetes Höhenprofil

Änderung der Referenzwerte und dadurch verursachte Fehler

Mit Hilfe der Funktion “gen_T_P_err.m” können sich ändernde Referenzparameter erzeugt werden. Die folgenden beiden Abbildungen in 3.4 zeigen jeweils die sich ändernde Referenztemperatur und den sich ändernden Referenzdruck. Um den Effekt der sich ändernden Referenzwerte deutlicher darstellen zu können, wurden die Standardabweichungen des treibenden Rauschens der Random-Walk-Prozesse, der Referenzdruck- und Temperaturänderungen, um den Faktor 10 erhöht. Aus diesem Grund erscheinen die Änderungen in den Abbildungen in 3.4 deutlich größer als eigentlich erwartet.

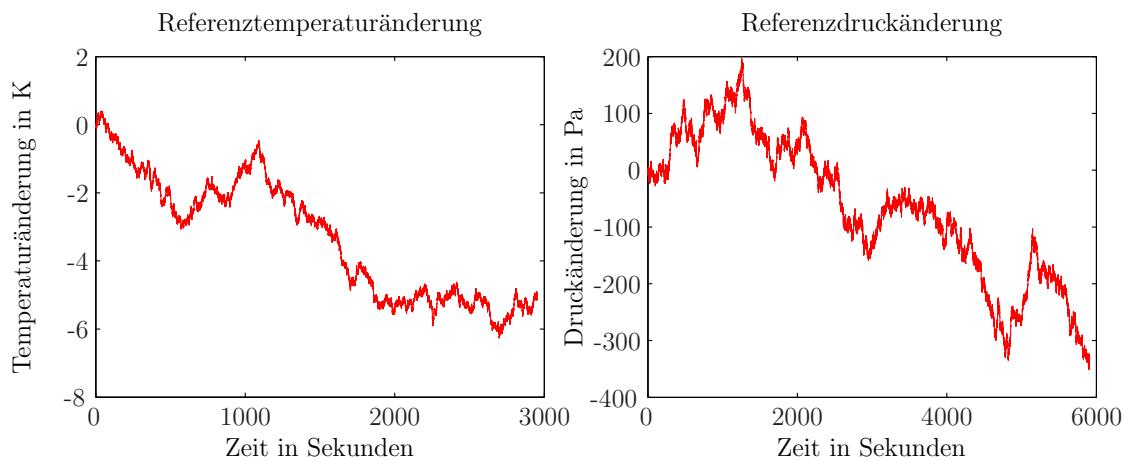


Abbildung 3.4: Referenztemperatur- und Referenzdruckänderung

Mit der Funktion “gen_T_P_profile.m” kann nun wiederum, passende zu einem vorgegebenen Höhenprofil, ein entsprechendes Druck- und Temperaturprofil generiert werden. Zur Demonstration der Auswirkungen der sich ändernden Referenzwerte wurde ein entsprechendes Druckprofil erzeugt. Basierend auf konstanten Referenzwerten wurde das Profil anschließend wieder in ein Höhenprofil umgerechnet. Die folgende Abbildung zeigt den Vergleich zwischen einem vorgegebenen Höhenprofil und dem berechneten Höhenprofil. Deutlich ist die größer werdende Abweichung zwischen vorgegebenem und berechnetem Profil zu erkennen. Außerdem ist die Unsicherheit, verursacht durch das simulierte Sensorrauschen, in der berechneten Höhe zu erkennen.

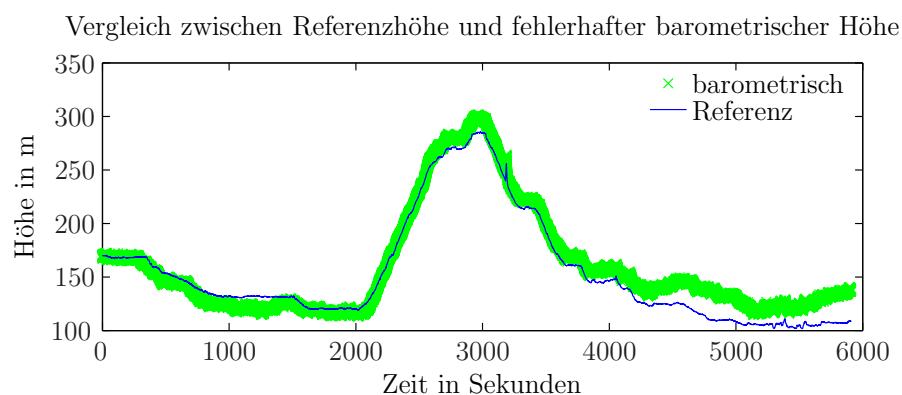


Abbildung 3.5: Auswirkungen sich ändernder Referenzparameter

4 Sensoren

Zur Aufnahme von realen Messdaten werden verschiedene Sensoren benötigt. Dazu sollen in diesem Abschnitt die in dieser Arbeit verwendeten Sensoren vorgestellt werden. Außerdem wird das Fehlerverhalten der Sensoren untersucht, und es werden entsprechende Modelle zur Modellierung der Fehler angeben. Soweit möglich, wird versucht, die Parameter der Fehlermodelle zu bestimmen. Für deterministische Fehler werden entsprechende Korrekturmöglichkeiten angegeben, um die gelieferten Messwerte zu verbessern.

4.1 Sensortypen

In dieser Arbeit wird auf verschiedene Messwerte zurück gegriffen. Hierzu gehören

- Zeitinformationen
- Positionsdaten und Geschwindigkeitswerte
- Beschleunigungswerte und Drehraten
- Luftdruckmessungen und Temperaturmessungen.

4.1.1 GPS-Empfänger



Abbildung 4.1: Navilock GPS-Empfänger (Quelle: [NL])

Zur Aufzeichnung von Positions- und Geschwindigkeitsdaten stehen zwei GPS-Empfänger der Firma NAVILOCK [NL] zur Verfügung. Der Empfänger NL-302U, welcher

Daten mit einer Datenrate von 1 Hz liefert und ein neuerer Empfänger, NL-402U, welcher Daten mit einer Datenrate bis zu 4 Hz liefert. Abbildung 4.1 zeigt den NL-402U Empfänger. In den Empfängern sind jeweils Chipsätze von SiRF [SIRF] (NL-302U: SiRF3) und u-blox [UBLOX] (NL-402U: u-blox5) verbaut. Die Empfänger werden per USB Anschluss an den Rechner angeschlossen und stehen im System über eine COM-Schnittstelle zur Verfügung. Zur Inbetriebnahme ist es gegebenenfalls nötig, vorher die Treiber von den jeweiligen Produktseiten [NL] zu installieren. Zur Konfiguration der Empfänger steht für den NL-302U die Software “SiRFDemo” [SFDEM] und für den NL-402U die Software “u-center“ [UCNTR] zur Verfügung.

Konfiguration

Auf die genaue Konfiguration der GPS-Empfänger soll hier nur oberflächlich eingegangen werden. Für den NL-302U ist nach dem Start von SiRFDemo lediglich die Einstellung ”Action → Switch to NMEA Protocol...“ wichtig, um die Daten in einem passenden Format zu erhalten.

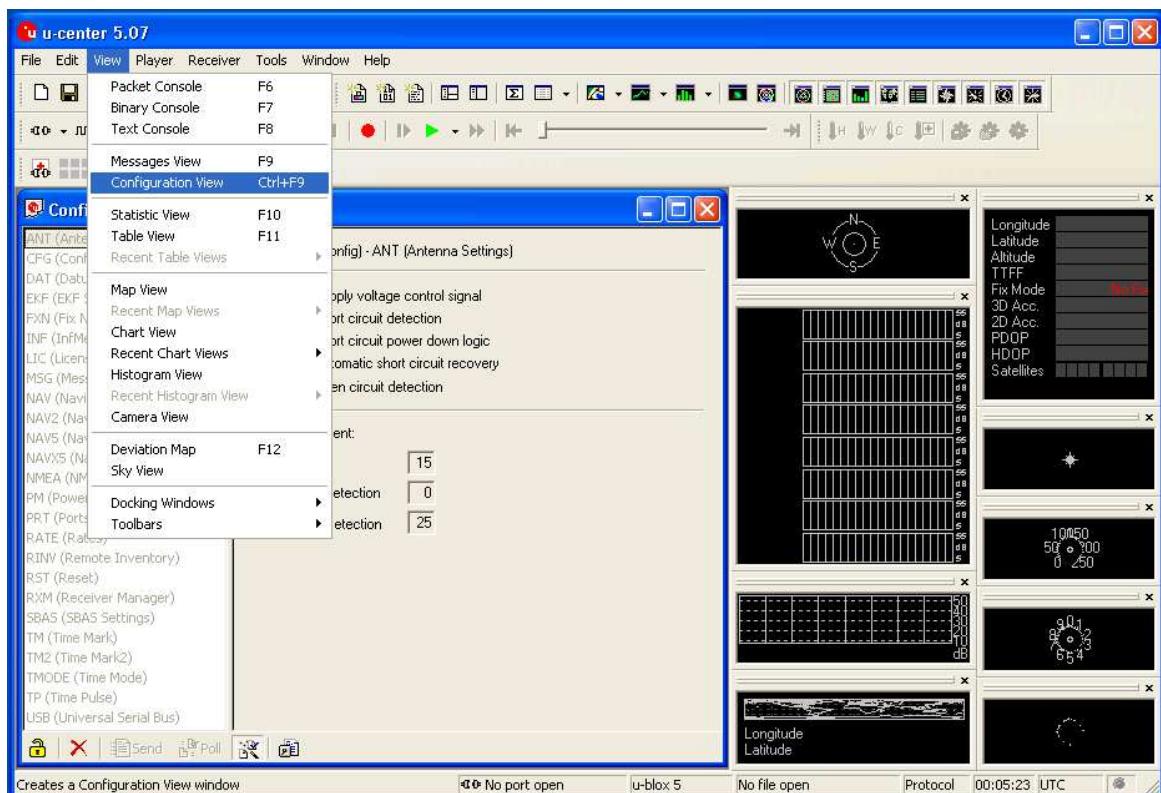


Abbildung 4.2: u-blox Konfiguration in u-center

Für den NL-402U sind die Einstellungsoptionen etwas umfangreicher. Nach dem Start vom u-center sind im Menü ”View → Configuration View“, wie in Abbildung 4.2 gezeigt, die Einstellungen veränderbar. Wichtige Optionen sind hier:

Updaterate: Im Unterpunkt ”RATE“ lassen sich sowohl das Messintervall als auch das

Ausgabeintervall, als Vielfaches des Messintervalls, einstellen. Hiermit lässt sich beeinflussen, mit welcher Update rate Daten vom Empfänger geliefert werden.

Systemmodell: In den Unterpunkten, die mit der Bezeichnung "NAV" beginnen, lassen sich die Systemmodelle wählen, welche der Empfänger zur Berechnung und Filterung der Positions und Geschwindigkeitsdaten nutzt. Die Profile beeinflussen die Genauigkeit der Schätzungen und die Verzögerung, mit der der Empfänger auf Änderungen reagiert. Eine kurze Beschreibung der Modelle ist in [UBPRT] im Abschnitt "Navigation Configuration Settings Description" auf Seite 32 zu finden. Empfehlenswert ist hier für die Verwendung in einem Auto die Einstellung "Automotive".

Datenausgabe: Im Unterpunkt "MSG" ist es möglich, unterschiedliche Ausgabedatensätze zu aktivieren. Wichtig sind hier vor allem die Datensätze "F0-00 NMEA GPGGA", "F0-04 NMEA GPRMC" und "F1-00 NMEA PUBX,00". Eine genaue Beschreibung dieser und weiterer Datensätze ist in [UBPRT] im Abschnitt "NMEA Protocol" ab Seite 47 zu finden. Ein kurzer Überblick über die wichtigsten Datensätze wird im nächsten Unterabschnitt gegeben.

Korrekturdaten: Der Empfänger bietet die Möglichkeit des Empfangs von Korrekturdaten eines satellitenbasierten Unterstützungssystems (SBAS [SBAS]), in Europa unter der Abkürzung EGNOS [EGNOS] bekannt. Die Daten dieses Systems werden genutzt, um die Positions und Geschwindigkeitsschätzungen des Empfängers zu verbessern. Im Unterpunkt "SBAS" bietet es sich an, die Option "Allow test mode use" zu setzen um die Daten von EGNOS im europäischen Bereich nutzen zu können.

Kartendatum: Im Unterpunkt "DAT" lässt sich das Kartendatum wählen. Hier ist WGS84 einzustellen.

Sichern der Konfiguration: Sollen die Einstellungen dauerhaft gespeichert werden, ist dazu im Unterpunkt "CFG" die Möglichkeit gegeben.

Ausgabeformat

Wie bereits erwähnt, liefern die GPS-Empfänger Daten zur aktuellen Position und Geschwindigkeit. Als Ausgabeformat wird dabei das sogenannte NMEA-0183 Datenformat [NMEA] genutzt. Dabei handelt es sich um eine zeilenweise Ausgabe von Datenblöcken im ASCII Format. Jeder Block beginnt dabei mit einem \$ gefolgt von einem Bezeichner für den Block. Direkt daran folgen die Datenfelder, jeweils getrennt durch ein Komma. Abgeschlossen wird der Block mit einer Prüfsumme, welche mit einem Stern beginnt. Die Datenblöcke haben dabei eine fest vorgegebene Struktur und Anzahl von Feldern. Sind keine Daten für ein Feld vorhanden, so wird ein leeres Feld ausgegeben. Diese Struktur macht es sehr einfach die ausgegebenen Daten weiterzuverarbeiten. Bei zeitkritischen Datensätzen ist zu beachten, dass nicht garantiert ist, dass die Daten zu dem Zeitpunkt ausgegeben werden an dem sie gültig sind. Hierbei kann es zu variablen

Verzögerungen von bis zu 500 ms und mehr kommen. Aus diesem Grund ist bei zeitkritischen Daten ein Zeitstempel in UTC-Zeit angegeben, der den Zeitpunkt der Gültigkeit angibt. Dabei ist eine Differenz zur lokalen Zeit bei Vergleichen zu beachten. Das GPS selber, basiert auf der sogenannten GPS-Zeit. Hierbei handelt es sich jedoch nur um eine, zur UTC-Zeit verschobene Basis. Die Verschiebung wird dabei bei der Ausgabe der UTC-Zeit berücksichtigt. Daher soll die GPS-Zeit hier weiter keine Rolle spielen. Im folgenden ist die Struktur der für diese Arbeit wichtigen Datenblöcke dargestellt. Eine ausführliche Liste kann [NMEA] entnommen werden.

GPGGA: Dieser Datenblock liefert Informationen über die 3D-Position.

\$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Wobei:

GPGGA	Bezeichner
123519	Messzeitpunkt 12:35:19 UTC
4807.038,N	Breite: 48° 07.038' N
01131.000,E	Länge: 11° 31.000' O
1	Modus
08	Anzahl verfolgter Satelliten
0.9	Horizontale Positionsverschlechterung
545.4,M	Höhe in m über dem Meeresspiegel (Geoid)
46.9,M	Höhe des Meeresspiegels (Geoids) über WGS84
(leeres Feld)	Zeit in Sekunden seit letzten DGPS Update
(leeres Feld)	DGPS Stations ID Nummer
*47	Prüfsumme beginnend mit *

GPRMC: Dieser Datenblock liefert Informationen über die 2D-Position und Geschwindigkeit.

\$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,,*6A

Wobei:

GPRMC	Bezeichner
123519	Messzeitpunkt 12:35:19 UTC
A	Status
4807.038,N	Breite: 48° 07.038' N
01131.000,E	Länge: 11° 31.000' O
022.4	Geschwindigkeit über Grund in Knoten
084.4	Bewegungsrichtung in Grad
230394	Datum: 23. März 1994
(leeres Feld)	Magnetische Ortsmissweisung (1)
(leeres Feld)	Magnetische Ortsmissweisung (2)
*6A	Prüfsumme beginnend mit *

PUBX,00: Dies ist ein spezieller Datenblock, der nach Aktivierung nur bei u-blox basierten Empfängern zur Verfügung steht. Der Block liefert 3D-Positionsinformationen und 3D-Geschwindigkeitsinformationen in einem Datenblock. Zusätzlich zu den vorherigen Blöcken steht hier auch die Vertikalgeschwindigkeit zur Verfügung. Der Aufbau des Blocks ist wie in [UBPRT] beschrieben:

```
$PUBX,00,123519,4807.038,N,01131.000,E,592.3,G3,2.1,2.0,41.485,
048.4,0.007,,0.9,1.19,0.77,8,0,0*5F
```

Wobei:

PUBX,00	Bezeichner
123519	Messzeitpunkt 12:35:19 UTC
4807.038,N	Breite: 48° 07.038' N
01131.000,E	Länge: 11° 31.000' O
592.3	Höhe in m über WGS84 Ellipsoid
G3	Modus
2.1	Horizontale Genauigkeit
2.0	Vertikale Genauigkeit
41.485	Geschwindigkeit über Grund in km/h
048.4	Bewegungsrichtung in Grad
0.007	Vertikale Geschwindigkeit nach unten in m/s
(leeres Feld)	Zeit in Sekunden seit letzten DGPS Update
0.9	Horizontale Positionsverschlechterung
1.19	Vertikale Positionsverschlechterung
0.77	Zeitverschlechterung
8	Anzahl verfolgter GPS Satelliten
0	Anzahl verfolgter GLONAS Satelliten
0	Dead Reckoning Status
*5F	Prüfsumme beginnend mit *

4.1.2 Inertiale Messeinheit

Zur Aufzeichnung von Beschleunigungen und Drehraten steht eine Inertiale Messeinheit (IMU) der Firma GLI Interactive LLC [GLI], wie in Abbildung 2.1 abgebildet, zur Verfügung. Verwendet wird der MotionNode [MTNDE], welcher Beschleunigungen bis 6 g und Drehraten bis $500 \frac{\circ}{s}$ in jeweils drei Raumrichtungen messen kann. Damit handelt es sich um einen sogenannten 6DOF-Sensor [6DOF]. 6DOF bedeutet dabei “sechs Freiheitsgrade”, in diesem Fall die Translationen in x, y und z-Richtung sowie die Rotationen um die x, y und z Achse des Sensors. Zusätzlich misst der MotionNode das lokale Magnetfeld in allen drei Raumrichtungen bis 100 μT und die Sensortemperatur in K. Die Orientierung der Achsen des MotionNode für die aufgezeichneten Daten ist dabei wie folgt: Die x-Achse ist vom MotionNode in Richtung Kabel gerichtet, die y-Achse ist nach oben gerichtet und die z-Achse folglich nach rechts. Die Orientierung der Achsen ist in Abbildung 4.3 gezeigt. Die Daten stehen mit einer Datenrate von 50 Hz bis 100 Hz, einstellbar in 10 Hz Schritten, zur Verfügung. Der Sensor wird per USB an den Rechner angeschlossen und wird im System über eine COM-Schnittstelle ange-

sprochen. Zur Inbetriebnahme ist es gegebenenfalls nötig, vorher die entsprechenden Treiber und die Software des Herstellers zu installieren [MNDRV]. Zur Konfiguration, sowie zum Aufzeichnen und Abspeichern der Daten muss die Software des Herstellers verwendet werden. Eine Schnellstartanleitung ist unter [MNGTS] zu finden. Auch das Ansprechen des Sensors über selbst entwickelte Software geschieht über Schnittstellen in der Software des Herstellers [MNSDK].

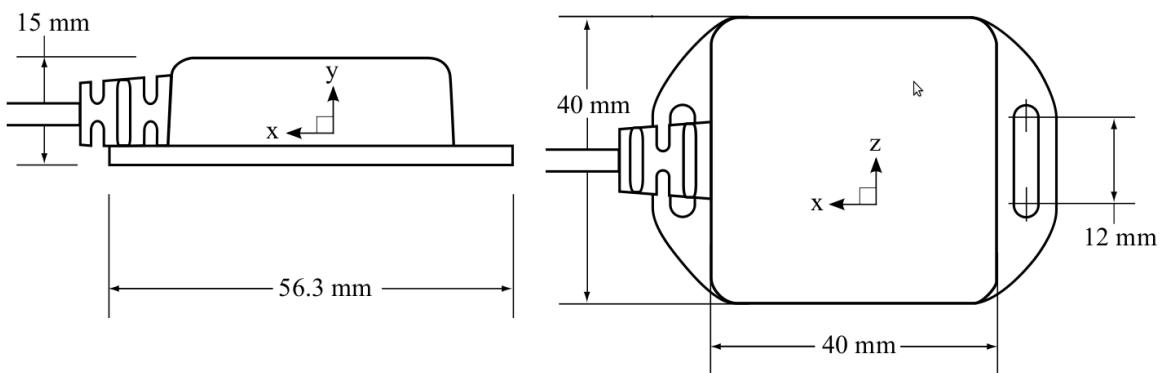


Abbildung 4.3: Koordinatensystem des MotionNode (Quelle: [MTNDE])

Inbetriebnahme und Konfiguration

Vor der Nutzung des MotionNode zur Aufzeichnung von Daten, sind einige Schritte zur Inbetriebnahme nötig. Ein ausführlicherer Überblick über die Software des Herstellers ist unter [MNTUT] zu finden. Als erstes wird der MotionNode per USB an den Rechner angeschlossen. Anschließend kann das “MotionNode User Interface” gestartet werden, welches sich in einem Webbrowser öffnet, wie in Abbildung 4.4 gezeigt. Hier wird der MotionNode nun per “Node → Scan” hinzugefügt. Vor der ersten Datenaufzeichnung sollte man den MotionNode etwa 15 min. auf Betriebstemperatur aufwärmen lassen. Auch im späteren Betrieb sollten extreme Temperaturschwankungen vermieden werden. Sollen die Magnetfeldaufzeichnungen später ebenfalls verwendet werden, ist es nötig, den MotionNode auf die jeweilige Umgebung, in der er verwendet wird, zu Kalibrieren. Hierzu sollte man zuerst im Punkt “Locations”, zu erreichen über den Pfeil neben “Log”, seinen aktuellen Standort einstellen. Die Kalibrierung geschieht dann wie unter [MNMAG] beschrieben. Abschließend sollten noch einige wichtige Einstellungen vorgenommen werden. Dazu wird per “Node → Close” die Verbindung zum MotionNode getrennt, falls diese besteht. Durch einen Klick auf “Detail” in der Zeile des entsprechenden MotionNodes öffnet sich die Konfigurationsseite. Standardeinstellungen für alle MotionNodes können durch Klick auf den Pfeil neben “Log” im Punkt “Defaults” abgespeichert werden. Wichtige Einstellungen sind:

Abtastrate: Die Abtastrate lässt sich bei “Sample Rate” von 50 Hz bis 100 Hz, in 10 Hz Schritten einstellen.

Tiefpassfilterung: Eine Tiefpassfilterung kann mit “Sensor Gain (Post)” eingestellt werden. Da unbekannt ist, wie sich dieser Parameter genau auswirkt, sollte hier

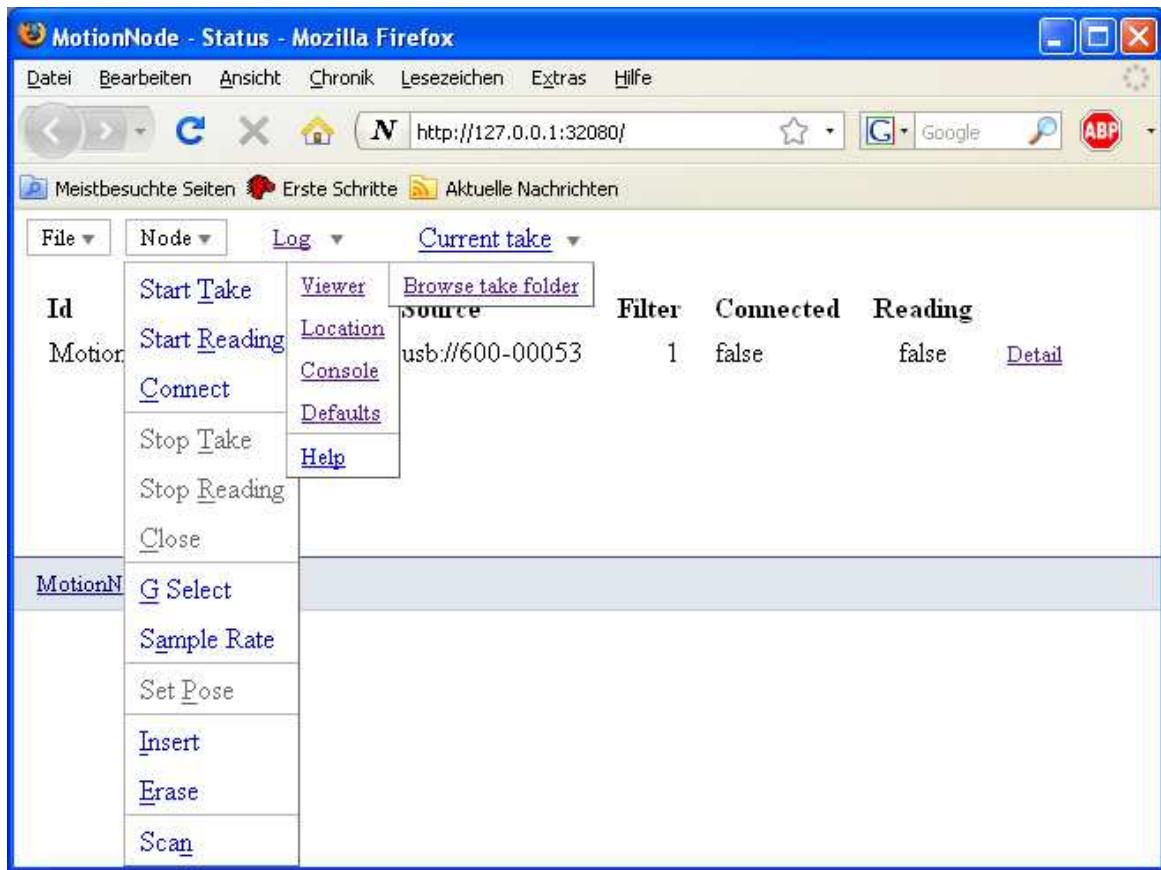


Abbildung 4.4: MotionNode Konfiguration

der Wert 1 eingestellt werden, um möglichst unveränderte Werte zu erhalten. Für die spätere Datenaufzeichnungen wurde hier daher ebenfalls der Wert 1 eingestellt.

Biaskompensation: Zur Vermeidung verfälschter Messwerte für die Drehraten, sollte bei "Track Gyroscope Bias" der Wert 0 eingestellt werden, um die Biaskompensation durch die Software zu deaktivieren. Bei anfänglichen Testfahrten hatte sich hier gezeigt, dass bei einer zu hohen Einstellungen auch reguläre Drehraten, z. B. bei einer konstanten Kreisfahrt, als Bias gewertet wurden. Für spätere Testfahren wurde hier daher der Wert 0 eingestellt.

Datenaufzeichnung und Ausgabeformat

Die Datenaufzeichnung wird über das Menü "Node → Start Take" gestartet. Über "Node → Stop Take" wird die Aufzeichnung wieder beendet. Werden Daten vom MotionNode gelesen wird dies durch den Wert "true" bei "Reading" angezeigt. Es ist dann über den Punkt "Viewer" möglich, erreichbar über den Pfeil neben "Log", die aktuellen Daten anzusehen. Nach der Aufzeichnung werden die Daten im "take folder" abgelegt. Dieser kann über den Pfeil neben "Current take" und dann mit "Browse take folder" eingesehen werden. Es stehen nun verschiedene Ausgabeformate zur Verfügung:

output: Die durch die Software des Herstellers berechnete Orientierung des Sensors steht in Form von 32 Bit Fließkommazahlen, als Quaternionen, im Unterordner “output” zur Verfügung.

raw: Die unveränderten Rohdaten der im MotionNode verbauten Sensoren stehen in Form von vorzeichenbehafteten, ganzzahligen 16 Bit Werten im Unterordner “raw” zur Verfügung. Hierbei werden jeweils die Werte der Beschleunigungssensoren, Magnetometer und Drehratensensoren in x, y und z-Richtung aufgezeichnet. Zusätzlich wird die Temperatur des Sensors aufgezeichnet.

sensor: Die nachbearbeiteten und auf physikalische Größen umgerechneten Werte stehen in Form von 32 Bit Fließkommazahlen im Unterordner “sensor” zur Verfügung. Hierbei werden ebenfalls jeweils die Werte der Beschleunigungssensoren, Magnetometer und Drehratensensoren in x, y und z-Richtung aufgezeichnet. Zusätzlich wird die Temperatur des Sensors aufgezeichnet. Die Beschleunigungswerte werden dabei in “g” Angegeben, wobei $1 \text{ g} = 9.80665 \frac{\text{m}}{\text{s}^2}$. Die Drehraten werden in $\frac{\circ}{\text{s}}$, die Magnetfeldwerte in μT und die Temperatur in K angegeben. Eine Nachbearbeitung der Messwerte kann durch eine Biaskompensation sowie Tiefpassfilterung erfolgt sein.

Zum Einlesen und Zeichnen der Daten in Matlab, stehen im Treiberpaket im Unterordner “tools/matlab/”, einige Skripte zur Verfügung. “plot_output_euler.m” wird zum Einlesen der “output-Daten” verwendet, “plot_raw.m” zum Einlesen der Rohdaten und “plot_sensor” zum Einlesen der nachbearbeiteten Daten.

Zur Synchronisation der aufgezeichneten Daten mit anderen aufgezeichneten Daten, stehen in der Datei “take.xml” Zeitstempel zum Beginn und Ende der Aufzeichnung, sowie die Anzahl der Samples zur Verfügung. Während der Aufzeichnung werden keine weiteren Zeitstempel erzeugt. Eine eventuelle Verzögerung zwischen Gültigkeitszeitpunkt und Aufzeichnungszeitpunkt dürfte zu vernachlässigen sein, da sich diese Verzögerung im Bereich < 1 ms bewegen dürften. Verlässliche Daten hierzu stehen zwar nicht zur Verfügung, aber vergleichbare Sensoren erreichen ähnliche Werte. Bei einem vergleichbaren Sensor, dem MTi von Xsens [XSMTI], sind z. B. Verzögerungen von 0.3 ms angegeben. Dieser Sensor steht dem Fachgebiet ebenfalls zur Verfügung, konnte für diese Arbeit aber nicht mehr verwendet werden.

4.1.3 Barometer

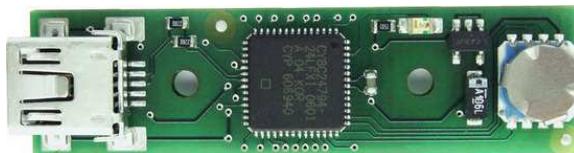


Abbildung 4.5: OAK-P Drucksensor (Quelle: [TORUB])

Zur Aufzeichnung des Luftdrucks steht ein Lufterucksensor der Firma Toradex [TORDX] aus der Produktreihe “Oak USB Sensoren” [TORUB] zur Verfügung. Verwen-

det wird hierbei der “Oak USB Sensor Atmospheric Pressure (OAK P)”. Die Abbildung 4.5 zeigt den Verwendeten Sensor. Der Sensor misst sowohl den Luftdruck im Bereich von 10 mbar bis 1100 mbar, als auch die Lufttemperatur im Bereich von 234 K bis 358 K. Die Daten stehen dabei mit einer Datenrate von bis zu 14 Hz zur Verfügung. Der Sensor wird per USB an den Rechner angeschlossen und steht im System als HID (Human Interface Device) [HID] zur Verfügung. Zur Inbetriebnahme ist es nicht nötig, spezielle Treiber zu installieren, da die entsprechenden HID-Treiber auf den meisten aktuellen Systemen vorinstalliert sind.

Konfiguration und Datenaufnahme

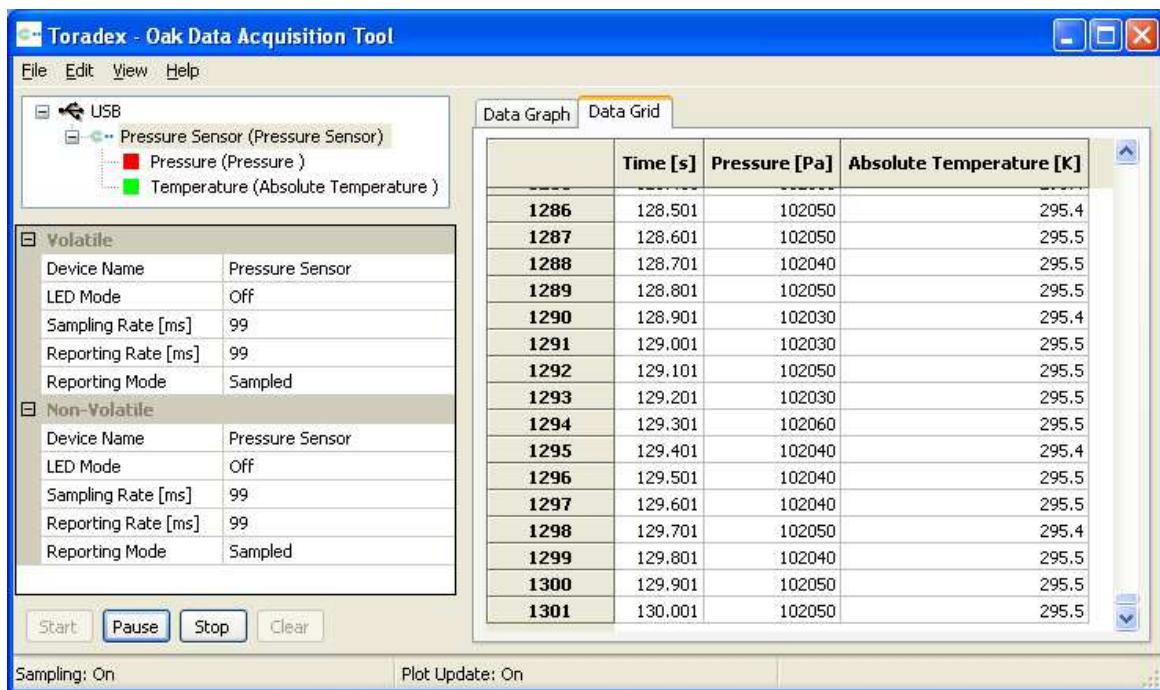


Abbildung 4.6: OAK-P Konfiguration und Datenaufnahme

Zur Konfiguration des Sensors und zur Datenaufnahme kann die Software des Herstellers, Oak DAT [TORUB] wie in Abbildung 4.6 dargestellt, verwendet werden. Dabei kann die Abtastrate auf den gewünschten Wert eingestellt werden. Zusätzlich sollte die Option “Reporting Mode” auf “Sampled” eingestellt werden damit die Daten mit möglichst geringer Verzögerung aufgezeichnet werden. Nach dem Klick auf “Start” beginnt die Aufzeichnung. Mit “Stop” kann die Aufzeichnung wieder beendet werden und mit “Clear” können bereits aufgezeichnete Daten wieder gelöscht werden. Speichern lassen sich die Daten mit “File → Export Data”. Dabei wird eine CSV-Datei [CSV] mit den aufgezeichneten Daten erstellt. Alternativ zur Verwendung der Software des Herstellers kann der Sensor auch direkt über das HID-Protokoll angesprochen werden. Zum vereinfachten Zugriff bietet der Hersteller hierzu eine Bibliothek, OAKLib [TORUB] an.

Eine Synchronisierung der aufgezeichneten Daten mit anderen Datensätzen gestaltet sich schwieriger, da sie Software lediglich einen Zähler mit der Dauer der Aufnahme-

zeit abspeichert. Eine absolute Zeit von Beginn oder Ende wird nicht aufgezeichnet. Die Verzögerung zwischen Gültigkeit und Aufnahme wird vom Hersteller mit 1 ms angegeben.

4.2 Sensorfehler

Im nächsten Schritt ist es nun nötig, sich mit der Qualität der Sensoren zu befassen, mögliche Fehlerquellen zu identifizieren und zu modellieren.

4.2.1 GPS-Empfänger

Aufgrund der komplexen Struktur des GPS-Systems, der empfängerinternen Verarbeitung und einer Vielzahl von Fehlerquellen ist es nur näherungsweise möglich, ein Fehlermodell anzugeben. Eine ausführlichere Auflistung und Beschreibung der Fehler kann dabei ([WEN07] S. 99ff) entnommen werden. Hier sollen die möglichen Fehlerquellen nur aufgezählt und deren Einfluss kurz genannt werden. Dabei kann zwischen zwei Fehlerarten unterschieden werden. Common-Mode-Fehler sind Fehler, die innerhalb einer bestimmten räumlichen Region bei allen Empfängern auftreten. Noncommon-Mode-Fehler treten dabei nur am jeweiligen Empfänger auf. Common-Mode-Fehler sind:

Ionosphären- und Troposphärenfehler: Fehler, verursacht durch variable Signallaufzeiten in der Atmosphäre vom Satelliten zum Empfänger

Ephemeridenfehler: Abweichung zwischen angenommener und tatsächlicher Satellitenbahn

Satellitenuhrenfehler: Abweichungen zwischen der GPS-Zeit und der Zeit der Satellitenuhr

Noncommon-Mode-Fehler sind:

Mehrwegeausbreitung: Fehler, verursacht durch den Empfang reflektierter Signale, z. B. von Gebäuden. Diese Fehler treten besonders stark in eng bebauten Gebieten, wie z. B. in einer Stadt auf.

Empfängeruhrenfehler: Abweichungen zwischen der GPS-Zeit und der Zeit im Empfänger.

Empfängerrauschen: Thermisches Rauschen und Fehler aufgrund von Nichtlinearitäten elektronischer Bauteile im Empfänger.

Positionsgenauigkeit

Zur einfachen Modellierung der Messfehler, bei einer Positionsmessung zu einem bestimmten Zeitpunkt k, werden sogenannte DOP-Werte (Dilution of precision) definiert

([WEN07] S. 102f). Diese Werte geben eine Abschätzung der zu erwartenden Verschlechterung der Messung, in Abhängigkeit einer minimalen Grundgenauigkeit σ_{p_0} an.

$$\sigma_{phor,k} = HDOP_k \cdot \sigma_{p_0} \quad (4.1)$$

$$\sigma_{pvert,k} = VDOP_k \cdot \sigma_{p_0}. \quad (4.2)$$

HDOP gibt dabei die horizontale Verschlechterung und VDOP die vertikale Verschlechterung an. σ_{phor} und σ_{pvert} sind die jeweiligen Standardabweichungen der Fehler in horizontaler und vertikaler Richtung. Stellt \vec{p}^n die Position in Nord, Ost und Unten-Richtung dar, kann die Positionsmessung im k-ten Zeitschritt wie folgt modelliert werden:

$$\tilde{\vec{p}}_k^n = \vec{p}_k^n + \vec{n}_{p,k}. \quad (4.3)$$

Wobei \vec{n}_p als mittelwertfreies, normalverteiltes und weißes Rauschen angenommen wird, mit

$$E[\vec{n}_{p,k} \vec{n}_{p,i}^T] = \begin{cases} \mathbf{R}_{p,k} & , k = i \\ 0 & , k \neq i. \end{cases} \quad (4.4)$$

Hierbei ist $\mathbf{R}_{p,k}$ eine zeitvariante Diagonalmatrix mit den Diagonalelementen $r_{p,k,1}$ bis $r_{p,k,3}$. Es gilt:

$$r_{p,k,1} + r_{p,k,2} = \sigma_{phor,k}^2 \quad (4.5)$$

$$r_{p,k,3} = \sigma_{pvert,k}^2 \quad (4.6)$$

Für ein sehr einfaches Modell können $\sigma_{phor,k}$ und $\sigma_{pvert,k}$ als konstant angenommen werden und $r_{p,k,1} = r_{p,k,2}$ gesetzt werden. Übliche Minimalwerte für σ_{phor} sind 3 m und 6 m für σ_{pvert} [NL]. DOP-Werte bis 20 sind möglich, und damit Maximalwerte bis zum 20-fachen des Minimalwertes. Üblich sind aber auch hier Faktoren zwischen 2 und 5 in schlechteren Empfangssituationen [DOP]. Zu beachten ist hier, dass die obige Modellierung nur eine Näherung darstellt, da die DOP-Werte nur die Konstellation der für die Positionsberechnung benutzten Satelliten berücksichtigen. Effekte aufgrund einer schlechten Signalqualität müssten in σ_{p_0} berücksichtigt werden. Informationen hierzu stehen aber im Allgemeinen nicht zur Verfügung, so dass man hier Annahmen treffen muss und den Wert häufig konstant auf die maximale Genauigkeit des jeweiligen Empfängers setzt.

Geschwindigkeitsgenauigkeit

Die Geschwindigkeitswerte lassen sich, analog zu den Positionsmessungen, mit

$$\tilde{\vec{v}}_k = \vec{v}_k + \vec{n}_{v,k} \quad (4.7)$$

modellieren, wobei \vec{n}_v als mittelwertfreies, normalverteiltes und weißes Rauschen angenommen wird, mit

$$E[\vec{n}_{v,k} \vec{n}_{v,i}^T] = \begin{cases} \mathbf{R}_{v,k} & , k = i \\ 0 & , k \neq i. \end{cases} \quad (4.8)$$

Hierbei ist $\mathbf{R}_{v,k}$ eine zeitvariante Diagonalmatrix mit den Diagonalelementen $r_{v,k,1}$ bis $r_{v,k,3}$. Es gilt analog wie bei den Positionsmessungen:

$$r_{v,k,1} + r_{p,k,2} = \sigma_{v_{hor},k}^2 \quad (4.9)$$

$$r_{p,k,3} = \sigma_{v_{vert},k}^2. \quad (4.10)$$

Da die GPS-Empfänger im Allgemeinen keine Informationen zur Genauigkeit der Geschwindigkeit ausgeben, werden die Varianzen $\sigma_{v_{hor,k}}$ und $\sigma_{v_{vert,k}}$ häufig als konstant angenommen. Übliche Minimalwerte für $\sigma_{v_{hor}}$ sind $0.1 \frac{\text{m}}{\text{s}}$ [NL]. In schlechteren Empfangssituationen dürften Werte bis $1 \frac{\text{m}}{\text{s}}$ und mehr möglich sein. Für $\sigma_{v_{vert}}$ werden für diese Arbeit Werte ab $1 \frac{\text{m}}{\text{s}}$ angenommen.

Daten der Verwendeten Empfänger

Für die in dieser Arbeit verwendeten GPS-Empfänger sind die Daten unter [NL], auf der jeweiligen Produktseite, bei den technischen Details zu finden. Die Daten werden in der Tabelle 4.1 noch einmal zusammengestellt. Nicht angegebene Werte werden mit NaN ausgefüllt. Die Werte in Klammern gelten bei Nutzung von SBAS. Zusätzlich ist

Empfänger	$\sigma_{p_{hor}}$ [m]	$\sigma_{p_{vert}}$ [m]	$\sigma_{v_{hor}}$ [$\frac{\text{m}}{\text{s}}$]	$\sigma_{v_{vert}}$ [$\frac{\text{m}}{\text{s}}$]
NL-302U	10	NaN	0.1	NaN
NL-402U	3 (2.4)	5.2 (2.7)	0.1	NaN

Tabelle 4.1: Parameter der GPS-Empfänger

der NL-402U in der Lage, Schätzungen für die Genauigkeit der jeweiligen Messung auszugeben. Informationen zur Genauigkeit der Position in NED-Richtung stehen im Datensatz “GBS” und Informationen zur Genauigkeit in horizontaler und vertikaler Richtung im Datensatz “UBX,00” zur Verfügung. Informationen zur Genauigkeit der 3D-Geschwindigkeit stehen nur in binären Datensätzen, wie z. B. NAV-VELNED zur Verfügung. Genaue Beschreibungen können [UBPRT] entnommen werden. In dieser Arbeit wurde auf die Nutzung dieser Daten vorerst verzichtet, da keine genauen Angaben zur Konsistenz der Daten gemacht werden können.

Anmerkung zu vorgefilterten Daten

Da in der Praxis oft vorgefilterte Werte von den GPS-Empfängern geliefert werden, wäre es für eine genauere Modellierung nötig, die Auswirkungen der Vorfilterung zu berücksichtigen oder eine Dekorrelation der Werte durchzuführen. Darauf soll in dieser Arbeit aber vorerst verzichtet werden.

4.2.2 Inertiale Messeinheit

Im Gegensatz zum GPS-Empfänger gestaltet sich die Fehlermodellierung bei der IMU etwas einfacher. Da sich die Fehler in den Beschleunigungssensoren und Drehratensensoren ähnlich verhalten, wird in der Modellierung nur bedingt zwischen den beiden Sensoren unterschieden. Die Modellierung soll anhand des Drehratensensoren gezeigt werden, gilt aber, soweit nichts anderes vermerkt ist, in gleicher Weise für den Beschleunigungssensoren. Da für einen späteren Filterentwurf die Gleichungen für den Drehratensor und den Beschleunigungssensor in zeitkontinuierlicher Form benötigt werden, werden auch die Fehlermodelle dieser beiden Sensoren im Zeitkontinuierlichen angegeben. Eine sehr genaue Betrachtung des Aufbaus verschiedener Sensoren und der

Fehlermodellierung sowie Fehlerkompensation kann ([TIT04] Kap. 4-8) entnommen werden. Auf die genaue Fehlermodellierung des Magnetometers soll verzichtet werden.

Fehlermodelle

Ein allgemeines Modell für die Messwerte vom Drehratensensoren ist mit

$$\tilde{\omega}_{ib}^b = \mathbf{M}(\vec{\omega}_{ib}^b, \vec{a}_{ib}^b, T) \cdot \vec{\omega}_{ib}^b + \vec{b}_\omega(\vec{\omega}_{ib}^b, \vec{a}_{ib}^b, T, t) + \vec{n}_\omega(t) \quad (4.11)$$

gegeben, wobei neben einer Abhängigkeit von den Anregungen auch Abhängigkeiten von der Temperatur T in der Zeit t gegeben sind.

$$\mathbf{M} = \begin{pmatrix} s_x & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & s_y & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & s_z \end{pmatrix} \quad (4.12)$$

wird als sogenannte Skalierungs- und Fehlausrichtungsmatrix bezeichnet. Die Faktoren $s_{(\dots)}$ auf der Hauptdiagonalen geben dabei einen Skalierungsfaktor an, der eine Abweichung von der idealen Kennlinie beschreibt. Eine Ideale Kennlinie wäre z. B. gegeben, wenn eine Anregung ω_x auf den Messwert $\tilde{\omega}_x = 1 \cdot \omega_x$ abgebildet wird. Die Faktoren $\delta_{(\dots)}$ auf den Nebendiagonalen beschreiben eine Fehlausrichtung der Sensoren, abweichend von einer idealen orthogonalen Anordnung. Normalerweise sollten die Sensoren orthogonal zueinander ausgerichtet sein. Ist dies nicht der Fall, so misst ein Sensor auch Ereignisse aus einer anderen Achse. Ist die x-Achse eines Beschleunigungssensors z. B. zur y-Achse geneigt, so werden in der x-Achse auch Beschleunigungen aus der y-Achse gemessen. Im Allgemeinen können besonders die Skalierungsfaktoren von der Temperatur T , den Drehraten $\vec{\omega}_{ib}^b$ und den Beschleunigungen \vec{a}_{ib}^b abhängen und zu einer nichtlinearen Kennlinie führen. Abbildung 4.7 zeigt dabei eine beispielhafte Kennlinie mit möglichen Sensorfehlern.

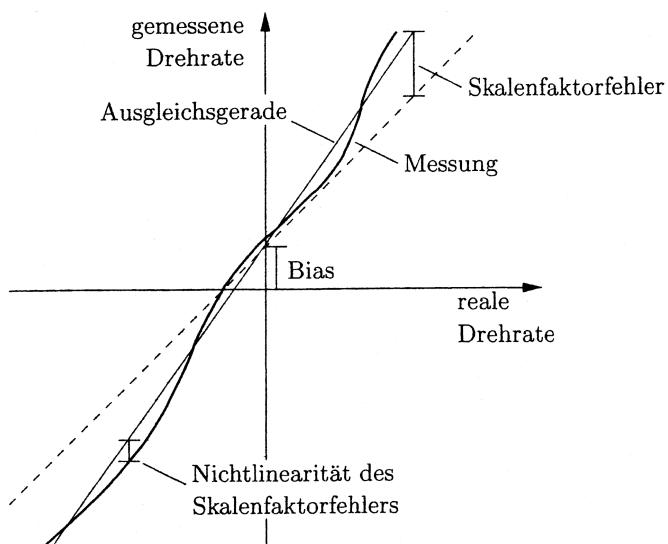


Abbildung 4.7: Unterschiedliche Sensorfehler (Quelle: [WEN07])

Neben der Skalierungs- und Fehlausrichtungsmatrix beschreibt \vec{b}_ω den Nullpunktfehler bzw. Offset oder Bias der Sensoren. Ein Bias führt dazu, dass Messwerte zu groß oder zu klein ausgegeben werden. Der Bias ist dabei oft von der Zeit t und Temperatur T abhängig, er variiert aber auch mit der Anregung der Sensoren. Eine Drehung des Sensors kann z. B. auch zu Beschleunigungsmesswerten führen.

Die meisten der oben genannten Fehler lassen sich nach dem Zusammenbau des Sensors bestimmen und kompensieren. Diese Kompensation wird im Abschnitt 4.3 beschrieben. Zufällige Effekte, abgesehen vom zeitveränderlichen Bias, sind im Rauschterm $\vec{n}_\omega(t)$ zusammengefasst. Dies betrifft insbesondere das sensorinhärente Rauschen.

Für diese Arbeit wird das obige Modell etwas vereinfacht. Es wird davon ausgegangen, dass die Skalierungs- und Fehlausrichtungsmatrix konstant ist und weder von der Temperatur, noch von der Anregung abhängt. Bei dem Bias wird davon ausgegangen, dass er sich aus einem konstanten Wert und einem zeitveränderlichen Teil zusammensetzt, aber unabhängig von der Anregung ist. Der Rauschterm wird als mittelwertfreies, weißes und normalverteiltes Rauschen angenommen. Damit ergibt sich nun folgendes Modell:

$$\tilde{\vec{\omega}}_{ib}^b = \mathbf{M} \cdot \vec{\omega}_{ib}^b + \vec{b}_\omega(t) + \vec{n}_\omega(t) \quad (4.13)$$

mit einer Modellierung des Bias durch einen Random-Walk Prozess

$$\dot{\vec{b}}_\omega(t) = \vec{n}_{b_\omega}(t), \quad (4.14)$$

wobei auch $\vec{n}_{b_\omega}(t)$ als mittelwertfreies, weißes und normalverteiltes Rauschen angenommen wird. Es wird außerdem davon ausgegangen dass die Rauschprozesse der einzelnen Komponenten von $\vec{n}_\omega(t)$ und $\vec{n}_{b_\omega}(t)$ jeweils unabhängig und unkorreliert sind. Damit ergibt sich für die einzelnen Komponenten:

$$E[n_x(t_1)n_x(t_2)] = \sigma_x^2 \cdot \delta(t_1 - t_2) \quad (4.15)$$

mit $x \in [\omega, b_\omega]$ für den Drehratensor und $x \in [a, b_a]$ für den Beschleunigungssensor. Hierbei können die Varianzen der einzelnen Komponenten natürlich verschieden voneinander sein. Für den MotionNode ergeben sich, unter der Annahme zeitinvarianter Größen, die in Tabelle 4.2 angegebenen Werte für die Rauschleistungsdichten [MTNDE]. Dabei wird die Rauschleistung in jeder Komponente als gleich angenom-

Sensor	inhärentes Rauschen	Bias-Rauschen
Beschleunigung	$\sigma_a = 0.5 \cdot 10^{-3} \frac{\text{m}}{\sqrt{\text{s}}}$	$\sigma_{b_a} = 0.3 \cdot 10^{-3} \frac{\text{m}}{\sqrt{\text{s}}}$
Drehrate	$\sigma_\omega = 0.014 \frac{\circ}{\sqrt{\text{s}}}$	$\sigma_{b_\omega} = 0.017 \frac{\circ}{\sqrt{\text{s}}}$

Tabelle 4.2: Rauschparameter des MotionNode

men, was für den MotionNode auch der Fall ist. Für Messungen zur Bestimmung der Parameter der Biasänderung sei z. B. auf [ALE09] verwiesen, wobei für diese Arbeit etwas optimistischere Werte für die Biasänderung des Drehratensors, mit einem Erwartungswert von $1 \frac{\circ}{\text{s}}$ nach einer Stunde angenommen werden. In [ALE09] zeigten sich

bereits nach 600 Sekunden Änderungen von $1 \frac{\circ}{\text{s}}$. Für die Beschleunigungsmesser wird ein Erwartungswert von 2 mg nach einer Stunde angenommen.

Häufig werden die Rauschleistungsdichten in der Einheit $\frac{1}{\sqrt{\text{Hz}}}$ angegeben. Zur Umrechnung der Werte in der Tabelle gilt:

$$\frac{1}{\sqrt{\text{s}}} \hat{=} \frac{\frac{1}{\text{s}}}{\sqrt{\text{Hz}}}, \quad 60 \frac{1}{\sqrt{\text{h}}} \hat{=} \frac{1}{\sqrt{\text{s}}} \quad \text{und} \quad 1 \text{ g} \hat{=} 9.80665 \frac{\text{m}}{\text{s}^2} \quad (4.16)$$

Modellierung im Zeitdiskreten

Da eine Datenerzeugung zur Simulation nur im Zeitdiskreten erfolgen kann, und auch Messwerte nur zu diskreten Zeitpunkten zur Verfügung stehen, ist eine Modellierung im Zeitdiskreten nötig. Zur Modellierung der Messwerte wird folgende Gleichung genutzt:

$$\tilde{\vec{\omega}}_{ib,k}^b = \mathbf{M} \cdot \vec{\omega}_{ib,k}^b + \vec{b}_{\omega,k} + \vec{n}_{\omega,k}. \quad (4.17)$$

Dabei wurde \mathbf{M} als konstant angenommen. Für weitere Betrachtungen kann für \mathbf{M} die Einheitsmatrix eingesetzt, wenn davon ausgegangen wird, dass diese Fehler vorher korrigiert werden. Für den Bias gilt:

$$\vec{b}_{\omega,k} = \vec{b}_{\omega,k-1} + \vec{n}_{b_{\omega},k}, \quad (4.18)$$

mit \vec{n}_{ω} und $\vec{n}_{b_{\omega}}$ als jeweils mittelwertfreiem, normalverteiltem und weißem Rauschen. Es gilt:

$$E[\vec{n}_{x,k} \vec{n}_{x,i}^T] = \begin{cases} \mathbf{R}_{x,k} & , k = i \\ 0 & , k \neq i. \end{cases} \quad (4.19)$$

Hierbei wird \mathbf{R}_x als unabhängig vom Abtastzeitpunkt angenommen und als Diagonalmatrix mit den Diagonalelementen $r_{x,1}$ bis $r_{x,3}$. Für die Diagonalelemente bei der Biasänderung gilt:

$$r_{x,[1-3]} = \Delta t \cdot \sigma_x^2 \quad (4.20)$$

mit $x \in [b_{\omega}, b_a]$. Dabei können die einzelnen Komponenten verschieden voneinander sein. Die Multiplikation der Varianz σ_x^2 mit Δt wird durchgeführt, um den Zusammenhang zwischen der Rauschleistung der Biasänderung im Zeitkontinuierlichen und der, mit der Abtastdauer steigenden Varianz des Bias im Zeitdiskreten zu beschreiben. Für das sensorinhärente Rauschen gilt:

$$r_{x,[1-3]} = \frac{\sigma_x^2}{\Delta t} \quad (4.21)$$

mit $x \in [\omega, a]$. Die einzelnen Komponenten können auch hier verschieden voneinander sein. Dabei beschreibt die Division durch Δt den Zusammenhang zwischen dem weißen Rauschen im Zeitkontinuierlichen und dem bandbegrenzten und abgetasteten weißen Rauschen im Zeitdiskreten ([WEN07] S. 69ff und S. 124f).

Die Generierung des sensorinhärenten Rauschens und des zeitveränderlichen Bias ist in den Matlab-Funktionen “gentherr2.m” für die Winkelinkremente der Drehratensensoren und in “gendverr2.m” für die Geschwindigkeitsinkremente der Beschleunigungssensoren implementiert.

Magnetometer

Die Fehler in den Messungen des Magnetometers werden als einfaches mittelwertfreies, normalverteiltes und weißes Rauschen angenommen. Die Rauschprozesse in den einzelnen Komponenten werden des Weiteren als unkorreliert angenommen. Ein Modell ist mit der folgenden Gleichung gegeben:

$$\tilde{\vec{h}}_{mag,k}^b = \vec{h}_{mag,k}^b + \vec{n}_{h_{mag},k}. \quad (4.22)$$

Für die einzelnen Komponenten von $\vec{n}_{h_{mag}}$ gilt:

$$n_{h_{mag},k} \sim \mathcal{N}(0, \sigma_{mag,k}^2). \quad (4.23)$$

Die einzelnen Komponenten können dabei verschieden voneinander sein. Für die Simulation wird $\sigma_{mag} = 110 \mu\text{T}$ für alle drei Komponenten gleich und konstant angenommen. [XSMTI] gibt $50 \mu\text{T}$ für den Sensor von Xsens an. In der Realität sind die Messwerte für das Magnetfeld allerdings noch weiteren Fehlern unterworfen. Hierzu gehören z. B. elektromagnetische Felder, die das Rauschen erhöhen können. Metallische Objekte beeinflussen außerdem Stärke, und evtl. auch Richtung, des Magnetfeldes. Hierzu wurden im Rahmen der Arbeit aber keine Untersuchungen angestellt.

4.2.3 Barometer

Bei der Modellierung der Fehler der barometrischen Messung wird von einem einfachen Modell ausgegangen, bei dem die Fehler lediglich als mittelwertfreies, normalverteiltes und weißes Rauschen angenommen werden. Die Modelle für die Druck- und Temperaturmessungen sind dabei wie in den Gleichungen (3.20) und (3.21) angegeben. Die Werte für die Varianzen in den Temperatur- und Druckmessungen wurden wie in Tabelle 4.3 bestimmt.

Sensor	Inhärentes Rauschen
Temperatur	$\sigma_T = 0.1 \text{ K}$
Druck	$\sigma_P = 10 \text{ Pa}$

Tabelle 4.3: Rauschparameter des Barometers

4.3 Korrektur der Messwerte

Weisen Messwerte deterministische Fehler auf, so ist es möglich, das entsprechende Fehlverhalten vorab zu bestimmen und vor der Messwertverarbeitung eine Korrektur durchzuführen. In diesem Abschnitt soll daher untersucht werden, bei welchen der in Abschnitt 4.1 genannten Sensoren eine Korrektur möglich ist. Daraufhin soll entsprechend versucht werden, zugehörige Korrekturterme zu bestimmen.

4.3.1 GPS-Empfänger

Bei einem einzelnen GPS-Empfänger kann aufgrund seiner komplexen Struktur davon ausgegangen werden, dass systematische Fehler beim Systementwurf vermieden oder bereits korrigiert wurden. Eine nachträgliche Bestimmung systematischer Fehler wäre ohnehin nur erschwert möglich und ein mögliches Fehlermodell relativ komplex. Es muss also davon ausgegangen werden, dass GPS-Empfänger, auf dem Stand der jeweils aktuellen Technik, die bestmöglichen Messwerte liefern.

Stehen mehrere entsprechende GPS-Empfänger zur Verfügung, ist es allerdings möglich, die in Abschnitt 4.2.1 genannten Common-Mode-Fehler zu korrigieren. Hierbei werden einer oder mehrere Empfänger an genau bekannten Positionen betrieben, und durch den Vergleich von zu erwartenden Daten mit gemessenen Werten, die jeweiligen Fehler bestimmt. Ein solches Verfahren nennt sich DGPS (Differential Global Positioning System) [DGPS] und es basiert darauf, dass sich Common-Mode-Fehler räumlich und zeitlich nur langsam ändern. Hiermit sind je nach Qualität der Referenzdaten und des Empfängers Genauigkeiten im Zentimeterbereich möglich [SAPOS]. Da der Betrieb entsprechender Referenzstationen aufwendig ist, und für die Verteilung der Informationen eine ständige Datenverbindung zur Verfügung stehen muss, ist die Verwendung von DGPS relativ teuer. Ein, dem DGPS System ähnliches System, ist SBAS. Dieses System steht jedem Nutzer frei zur Verfügung, allerdings bietet es nur eine geringe Menge an Korrekturinformationen und damit auch nicht die Genauigkeit von DGPS. Der Vorteil bei SBAS ist allerdings die Verfügbarkeit für große Bereiche und der geringe Aufwand für den Empfang.

4.3.2 Inertiale Messeinheit

Wie in Abschnitt 4.2.2 gezeigt, können inertiale Messeinheiten eine Vielzahl von Fehlern aufweisen. Da es kompliziert, zeitaufwendig und teuer ist, eine solche Messeinheit vollständig zu testen, wird eine genaue Fehlerbestimmung nicht von jedem Hersteller durchgeführt. Normalerweise wird eine IMU aus den Sensorchips einzelner Zulieferer hergestellt. Einige Hersteller verlassen sich dabei auf die Angaben in den Datenblättern der gelieferten Chips. Fehler des Gesamtsystems, die durch den Zusammenbau entstehen, bleiben dabei unberücksichtigt. Dies ist z. B. bei dem verwendeten MotionNode der Fall. Einige Hersteller führen nach dem Zusammenbau ausführliche Tests durch und bestimmen die systematischen Fehler, um die Messwerte vor der Ausgabe zu korrigieren. Dies ist z. B. bei dem MTi von Xsens der Fall. Die Sensoren werden individuell getestet und die Testergebnisse durch ein beigelegtes Zertifikat bescheinigt. Der Aufwand für die Kalibrierung führt natürlich zu einem höheren Preis der Sensoreinheit. Nicht-kalibrierte Sensoreinheiten sind günstiger zu bekommen. Da im Rahmen dieser Arbeit festgestellt wurde, dass der MotionNode teils ein erhebliches Fehlverhalten aufweist, wurde versucht, einige Fehlerterme des Modells (4.11) zu bestimmen. Auf den Sensor von Xsens soll hier nicht weiter eingegangen werden, da von einer entsprechend guten Kalibrierung ausgegangen wird. Zur Fehlerbestimmung wird zunächst ein vereinfachtes Modell nach (4.13) angenommen.

Kalibrierung der Beschleunigungssensoren

Für die Beschleunigungssensoren wurde versucht, sowohl die Skalierungs- und Fehlausrichtungsmatrix \mathbf{M}_a , wie in Gleichung (4.12) definiert, als auch die Biaswerte \vec{b}_a zu bestimmen. Bei Tests zeigte sich, dass es eine relative starke Kopplung zwischen den Messwerten der x-Achse und den Beschleunigung in der z-Achse gibt, was auf eine Fehlausrichtung schließen lässt. Da zusätzlich Skalierungsfehler nicht ausgeschlossen werden können, werden auch diese bestimmt. Außerdem weist die Ausgabe eines von Null verschiedenen Messwertes in der z-Achse bei Nulllage auf einen Bias hin. Damit ist hier, unter Vernachlässigung des Sensorrauschens, das vollständige, vereinfachte Modell

$$\tilde{\vec{a}}_{ib}^b = \mathbf{M}_a \cdot \vec{a}_{ib}^b + \vec{b}_a \quad (4.24)$$

gegeben. Der Bias wird zusätzlich als unabhängig von der Zeit angenommen und das Rauschen durch eine genügend lange Messzeit vernachlässigt. Die Gleichung (4.24) lässt sich als drei unabhängige lineare Gleichungen schreiben. Dies sei hier für die erste Komponente gezeigt. Die Indizes $(.)_{ib}^b$ werden dabei zur Vereinfachung weggelassen:

$$\tilde{a}_x = s_{a,x} \cdot a_x + \delta_{a,xy} \cdot a_y + \delta_{a,xz} \cdot a_z + b_{a,x}. \quad (4.25)$$

Wie zu erkennen ist, ist es nun mit vier geschickt gewählten Messungen möglich, die vier Parameter $s_{a,x}$, $\delta_{a,xy}$, $\delta_{a,xz}$ und $b_{a,x}$ zu bestimmen. Für alle drei Komponenten würden 12 Messungen genügen. Da eine Messreihe mit 80 Einzelmessungen und jeweils drei Messwerten durchgeführt wurde, stehen deutlich mehr Messungen zur Verfügung. Zur Lösung des Gleichungssystems wurde die Methode der kleinsten Fehlerquadrate genutzt. Dazu wurde die Gleichung 4.25 in Matrix-Form geschrieben:

$$\tilde{\vec{a}}_x = \mathbf{A} \cdot \vec{M}_1^T, \quad (4.26)$$

mit

$$\tilde{\vec{a}}_x = \begin{pmatrix} \tilde{a}_{x,1} \\ \vdots \\ \tilde{a}_{x,n} \end{pmatrix} \text{ und } \mathbf{A} = \begin{pmatrix} \vec{a}_1^T & 1 \\ \vdots & \vdots \\ \vec{a}_n^T & 1 \end{pmatrix}, \quad (4.27)$$

wobei $\tilde{a}_{x,n}$ die n-te Messung in der x-Achse ist, \vec{a}_n die n-te Anregung und \vec{M}_1 die erste Zeile der Matrix \mathbf{M}_a erweitert um $b_{a,x}$. Mit

$$\mathbf{A}^+ \cdot \tilde{\vec{a}}_x = \vec{M}_1^T \quad (4.28)$$

erhält man nun das Ergebnis für die erste Zeile von \mathbf{M}_a mit angehängtem Bias. \mathbf{A}^+ ist dabei die Pseudoinverse von \mathbf{A} . Für die weiteren Zeilen wurden jeweils die Messungen der y-Achse und der z-Achse auf die selbe Weise ausgewertet. Eine Auswertung in einem Schritt ist mit

$$\mathbf{A}^+ \cdot \tilde{\mathbf{A}} = \left[\mathbf{M}_a \quad \vec{b}_a \right]^T \quad (4.29)$$

möglich. Dabei ist $\tilde{\mathbf{A}} = [\tilde{\vec{a}}_x \quad \tilde{\vec{a}}_y \quad \tilde{\vec{a}}_z]$.

Die Kalibrierung ist im Matlab-Skript “accalibrate.m” bzw. den von dort aufgerufenen Skripten implementiert.

Alternatives Verfahren zur Bestimmung der Skalierungsfaktoren

Möchte man nur die Skalierungsfaktoren der Beschleunigungssensoren bestimmen, bietet sich die Annahme

$$g_l^2 = (s_{a,x} \cdot (a_x - b_{a,x}))^2 + (s_{a,y} \cdot (a_y - b_{a,y}))^2 + (s_{a,z} \cdot (a_z - b_{a,z}))^2 \quad (4.30)$$

an. Das Betragsquadrat der einzelnen Beschleunigungen entspricht, im unbeschleunigten Fall, der Erdbeschleunigung zum Quadrat. Dieses Vorgehen hat den Vorteil das hier die Orientierung des Sensors keine Rolle spielt. Andererseits lassen sich hiermit nicht die Fehlausrichtungsfaktoren $\delta_{(...)}$ bestimmen. Ein entsprechendes Verfahren ist in ([WEN07] S. 304ff) zur Implementierung als Kalman-Filter beschrieben. Dieses Verfahren wurde ebenfalls implementiert und am MotionNode getestet. Aufgrund der großen Fehlausrichtung der x-Achse lieferte es aber keine sinnvollen Werte und wurde daher nicht weiter verwendet.

Das Verfahren ist im Matlab-Skript “acc_calibrate.m” und in den von dort aufgerufenen Skripten implementiert.

Aufnahme der Werte für die Beschleunigungssensoren

Ein Problem bei der Kalibrierung von Sensoren ist, eine genau definierte Anregung zu generieren, um anschließend die Antwort des Sensors auszuwerten. Um einen Beschleunigungssensor zu kalibrieren, ist es also nötig, verschiedene bekannte Beschleunigungen auf den Sensor auszuüben. Die einfachste Möglichkeit wäre hier z. B. eine Zentrifuge zu verwenden. Da eine Zentrifuge nicht zur Verfügung stand, wurde nach einer anderen Möglichkeit gesucht. Eine noch einfachere Möglichkeit, bekannte Beschleunigungen auf den Sensor wirken zu lassen, ist die Schwerkraftbeschleunigung \vec{g}_l der Erde. Wird die Schwerkraftbeschleunigung in unterschiedlichen Winkeln in einen Beschleunigungssensor eingekoppelt, ist es möglich Beschleunigungen im Bereich von etwa -1 g bis 1 g zu messen. Um dies zu erreichen, wurde eine sogenannte schräge Ebene [SCHEB] konstruiert. Hiermit ist es möglich, den Sensor in genau definierte Positionen zu bringen und für jede Position jeweils den zu erwartenden Messwert zu berechnen, sowie den gemessenen Messwert aufzunehmen. Die Messungen wurden in 5° Schritten von 0° bis 90° Neigung durchgeführt. Nach jeder Messung wurde der Winkel erhöht und nach einer kurzen Pause die nächste Messung durchgeführt. Die Messdauer beträgt etwa 60 Sekunden. Durch die Pause wurden mögliche Störungen durch das Verändern des Winkels vermieden und durch die Länge der Messzeit wird der Einfluss des Rauschens vermindert. Nach jeder Messung wurde über die Werte in den jeweiligen Achsen gemittelt. Damit stehen nun 3 Messwerte für die drei Achsen des Sensors zur Verfügung. Zusätzlich lässt sich aus der Stellung der Ebene die tatsächliche Beschleunigung für die drei Achsen bestimmen. Die Messreihe sollte so ausgelegt sein, dass bei jedem Sensor Beschleunigungen von -1 g bis 1 g eingekoppelt werden, um eine vollständige Abdeckung zu erreichen. Die Abbildung 4.8 zeigen den Messaufbau.

Der Zusammenhang zwischen der Neigung der Ebene sowie der Anfangsposition des

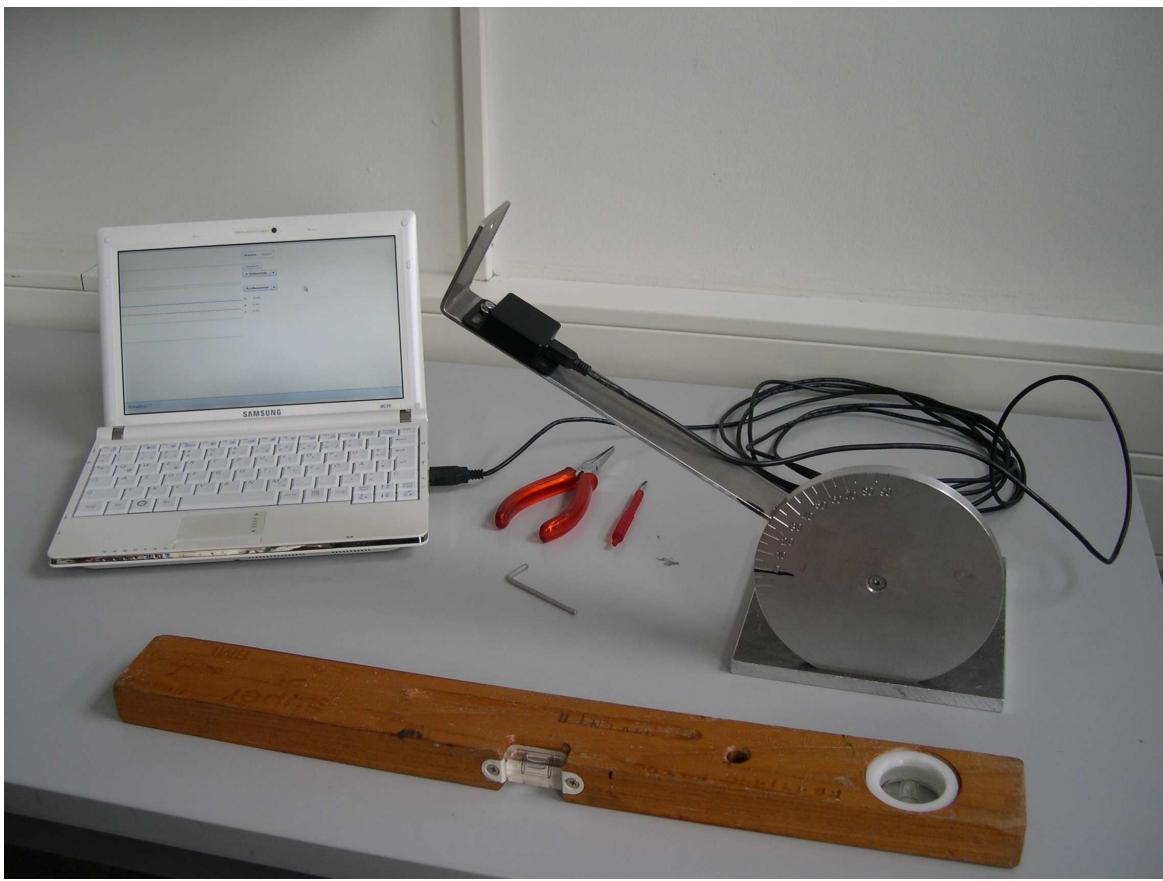


Abbildung 4.8: Schiefe Ebene zur Kalibrierung des Beschleunigungsmessers

Sensors und den Messwerten in den einzelnen Achsen ist durch

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \begin{pmatrix} \sin \alpha \cdot \cos \gamma + \cos \alpha \cdot \sin \beta \cdot \sin \gamma \\ \cos \alpha \cdot \cos \beta \\ \sin \alpha \cdot \sin \gamma - \cos \alpha \cdot \sin \beta \cdot \cos \gamma \end{pmatrix} \cdot g_l \quad (4.31)$$

gegeben. Dabei ist für g_l der Betrag der Schwerkraftbeschleunigung am jeweiligen Ort einzusetzen. Für Paderborn gilt hier $g_l = 9.8119 \frac{\text{m}}{\text{s}^2}$ [GIS]. Die Anfangsposition des Sensors auf der schiefen Ebene ist dabei wie folgt: Die x-Achse zeigt vom Drehpunkt der Ebene weg, die y-Achse weist nach oben und die z-Achse weist, vom Drehpunkt zum Sensor gesehen, nach rechts. Der Winkel α gibt die Neigung der schiefen Ebene an. Der Winkel β gibt die Verkipplung des Sensors auf der schiefen Ebene, nach rechts oder links, an. Der Winkel γ gibt die Verdrehung des Sensors auf der schiefen Ebene an. Im Prinzip beschreiben die Winkel eine Drehung des Sensorkoordinatensystems um seine Achsen in der Reihenfolge z-x-y. Die Gleichung (4.31) lässt sich durch Multiplikation von g_l , mit der zweiten Zeile der entsprechenden Rotationsmatrix [RTMAT] herleiten, hier die "yzx"-Matrix wobei 1 und 3 vertauscht sind. In diesem Fall ist darauf zu achten, dass für g_l in Vektorform gilt:

$$\vec{g}_l^{MN} = \begin{pmatrix} 0 \\ g_l \\ 0 \end{pmatrix}. \quad (4.32)$$

Die Anordnung der Komponenten von g_l^{MN} liegt in dem Koordinatensystem des MotionNode begründet. Die y-Achse weist hierbei nach oben.

Ergebnisse der Kalibrierung der Beschleunigungssensoren

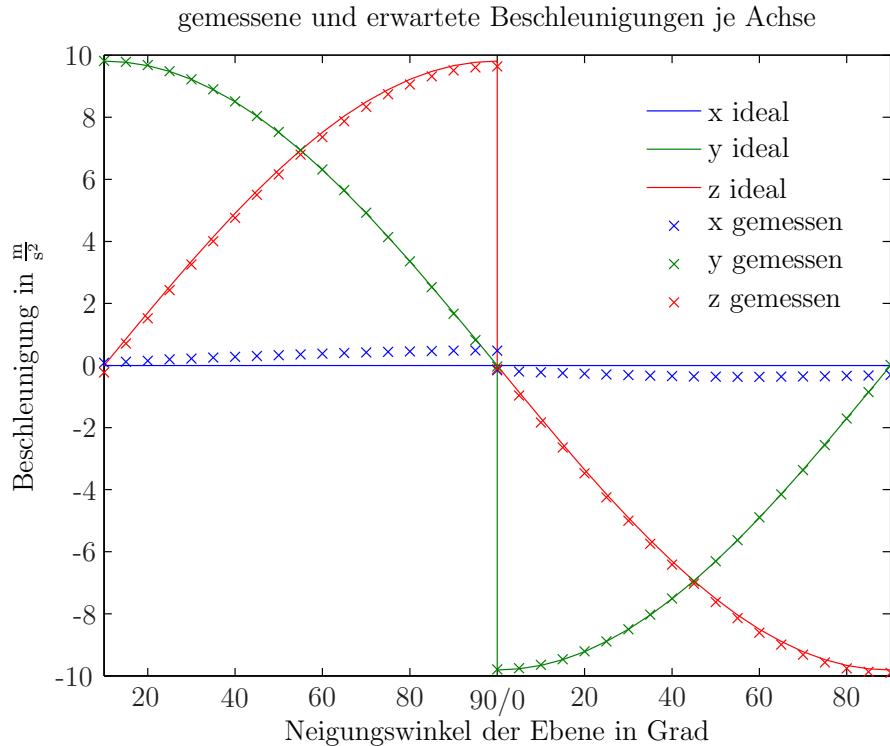


Abbildung 4.9: aufgezeichnete Beschleunigungen

Die Abbildung 4.9 zeigt beispielhaft das Ergebnis zweier Messdurchläufe. Aufgetragen sind jeweils die gemessenen Beschleunigungen und die erwarteten Messwerte, in Abhängigkeit von der Neigung der schiefen Ebene. Es ist deutlich der Effekt der Fehlausrichtung der x-Achse zu erkennen, es werden von Null verschiedene Messwerte ausgegeben, obwohl sie Null sein müssten. Auch der Effekt des Bias in der z-Achse ist zu erkennen. Aus allen Messwerten wurden nun die Parameter für \mathbf{M}_a und \vec{b}_a bestimmt.

$$\mathbf{M}_a^{MN} = \begin{pmatrix} 1.0036 & 0.0085 & 0.0417 \\ -0.0069 & 1 & 0 \\ 0 & 0 & 0.9961 \end{pmatrix}, \quad \vec{b}_a^{MN} = \begin{pmatrix} 0 \\ 0 \\ -0.114 \end{pmatrix} \frac{\text{m}}{\text{s}^2}. \quad (4.33)$$

Mit diesen Werten ist es nun möglich, die Messwerte zu korrigieren. Dazu wird die Gleichung (4.24) nach \tilde{a}_{ib}^b aufgelöst. Auch hier ist zu beachten, dass es sich um Messwerte im Koordinatensystem des MotionNode handelt.

$$\tilde{a}_{ib}^{MN} = (\mathbf{M}_a^{MN})^{-1} \cdot (\tilde{a}_{ib}^{MN} - \vec{b}_a^{MN}) \quad (4.34)$$

Die Abbildung 4.10 zeigt, für den aufgezeichneten Datensatz aus Abbildung 4.9, die korrigierten Werte. Es ist deutlich zu erkennen, dass der Effekt der Fehlausrichtung der

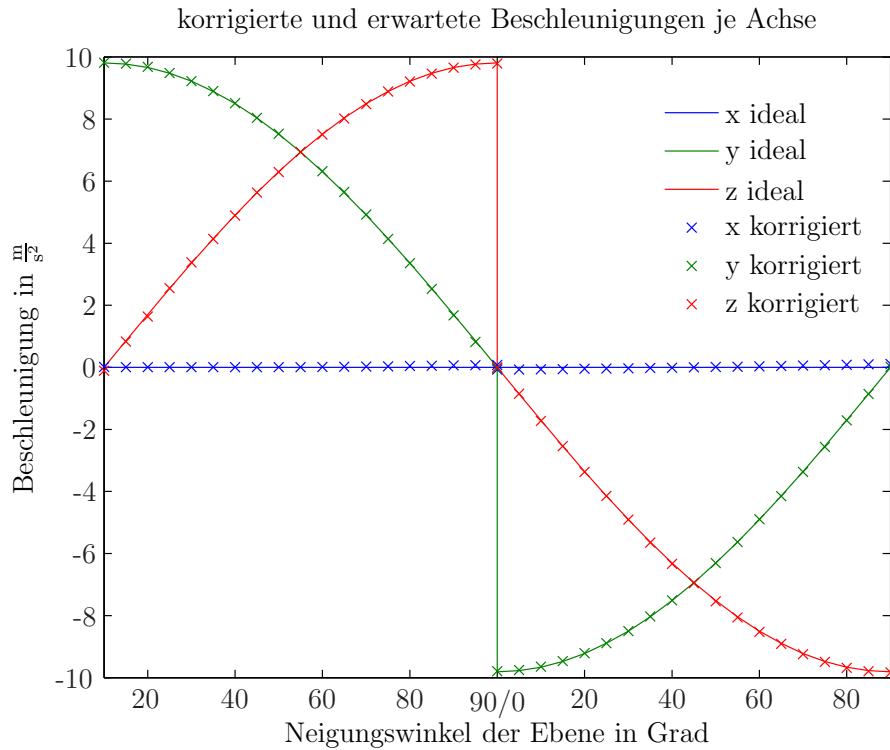


Abbildung 4.10: korrigierte Beschleunigungen

x-Achse korrigiert wurde. Auch der Bias in der z-Achse wurde korrigiert. Dies konnte auch mit weiteren Testmessungen belegt werden.

Kalibrierung der Drehratensensoren

Da sich bei den Messungen der Drehraten ebenfalls deutliche Fehler zeigten, wurde auch hier versucht, diese zu bestimmen. Ein wesentlicher Fehler ist hierbei eine falsche Skalierung der Drehrate, sowie ein deutlicher Bias. Der Betrag der Messwerte der Drehraten wird deutlich zu gering ausgegeben. Dies zeigt sich, wenn man den MotionNode um einen bekannten Winkel, z. B. 360° , dreht. Ein Aufintegrieren, der ungefilterten Messwerte, ergibt anschließend einen kleineren Winkel zwischen 335° und 334° . Da die Skalierungsfehler den größeren Teil der Fehler ausmachen, und bei der Durchführung der Messung keine ausreichend genaue Positionierung gewährleistet werden konnte, wurde auf die Bestimmung der Fehlausrichtungsfaktoren verzichtet. Dies führt zu einer vereinfachten Form der Gleichung (4.25). Für die einzelnen Komponenten, hier am Beispiel der ersten Komponente, resultiert, unter Vernachlässigung des Sensorrauschens, dabei:

$$\tilde{\omega}_x = s_{\omega,x} \cdot \omega_x + b_{\omega,x}. \quad (4.35)$$

Mit zwei Messungen lassen sich nun $s_{\omega,x}$ und $b_{\omega,x}$ bestimmen. Auch hier wurden mehr Messungen als nötig durchgeführt, jeweils vier Mal zwei Messungen für jede Komponente. Eine Messung wurde in Ruhelage durchgeführt, um den Bias zu bestimmen. Anschließend wurde eine Messung durchgeführt, bei der sich der Sensor mit konstanter Drehrate drehte, um den Skalierungsfaktor zu bestimmen. Die Messungen wurden bei

vier verschiedenen Drehraten durchgeführt, um die Skalierungsfaktoren bei unterschiedlichen Drehraten zu vergleichen. Für ein lineares Modell sollten die Faktoren bei jeder Drehrate gleich sein. Für jede Messreihe wurde $s_{\omega,x,i}$, $s_{\omega,y,i}$ und $s_{\omega,z,i}$ bestimmt, dabei bezeichnet i die i -te Messung. Im Anschluss wurde für jede Komponente, mit

$$s_{\omega,x} = \frac{1}{4} \sum_{i=1}^4 s_{\omega,x,i} \quad (4.36)$$

der Skalierungsfaktor der jeweiligen Komponente ermittelt, wie hier an der ersten Komponente gezeigt ist.

Das Verfahren ist im Matlab-Skript “gyro.m” implementiert.

Aufnahme der Werte für die Drehratensensoren

Auch bei den Drehratensensoren ergibt sich das Problem genau definierte Anregungen zu generieren. Aufgrund des einfachen Modells (4.35), genügen für die Berechnung des Skalierungsfaktors allerdings zwei Messungen je Komponente. Eine einfache Möglichkeit wäre hier z. B. eine erste Ruhemessung zur Bestimmung und Eliminierung des Bias und ein anschließendes Drehen des Sensors um einen bekannten Winkel. Die Aufintegration der gemessenen Drehraten sollte dann den bekannten Winkel ergeben. Das Verhältnis zwischen bekanntem Winkelinkrement $\Delta\alpha_x$ und berechnetem Winkelinkrement bestimmt den Skalierungsfaktor.

$$\Delta\alpha_x = \frac{1}{s_{\omega,x}} \cdot (\tilde{\omega}_x - b_{\omega,x}) \cdot \Delta t \quad (4.37)$$

In einem ersten Versuch wurde der Sensor dazu jeweils 100 Mal, in beide Richtungen, um jede seiner drei Achse rotiert. Als Winkelinkrement ergibt sich dann jeweils 36000° . Die auf diese Weise bestimmten Skalierungsfehler liegen schon nahe ($+/- 15\%$) an den später bestimmten Werten. Ein Nachteil an dieser Methode ist allerdings, dass man die Drehraten nicht direkt vergleichen kann, um eventuelle Nichtlinearitäten zu bestimmen. Um eine genauere Bestimmung der Skalierungsfaktoren durchzuführen, ist ein Drehteller nötig, bei dem man feste Drehraten einstellen kann. Da ein solcher Drehteller nicht zur Verfügung stand, wurde ein Schallplattenspieler verwendet. Die meisten Schallplattenspieler bieten die Möglichkeit, zwei verschiedene Drehraten mit einer relativ hohen Genauigkeit einzustellen. Der hier verwendete Schallplattenspieler bietet die Möglichkeit Drehraten von $200 \frac{\circ}{s}$ und $270 \frac{\circ}{s}$ einzustellen. Die Abbildung 4.11 zeigt den Versuchsaufbau. Bei den Messungen wurde versucht, den Sensor so zu positionieren, dass eine Drehung jeweils in nur einer Achse durchgeführt wird. Zur Bestimmung des Bias wurde als Erstes für 60 Sekunden eine Ruhemessung durchgeführt. Danach wurde der Schallplattenspieler eingeschaltet und die gewünschte Drehrate eingestellt und stabilisiert. Anschließend folgte eine Messung für 60 Sekunden bei konstanter Drehrate. Abschließend wurde der Schallplattenspieler wieder gestoppt und die Messwerte vom Ende der Messung genutzt, um den vorher geschätzten Bias noch einmal zu überprüfen. Im nächsten Schritt wurde auf die selbe Weise eine Messung mit der zweiten Drehrate durchgeführt. Um auch negative Drehraten zu messen, wurde der Sensor nach den



Abbildung 4.11: Drehplattform zur Kalibrierung des Beschleunigungsmessers

beiden Messreihen “auf den Kopf” gedreht und die selben Messreihen noch einmal durchgeführt. Auf diese Weise konnten, jeweils für alle drei Achsen, die Skalierungsfaktoren bestimmt werden. Prinzipiell liefert jede Messreihe Messwerte für alle drei Achsen. Aufgrund der ungenauen Positionierung konnte aber nicht zweifelsfrei festgestellt werden, ob die Drehraten in den nicht angeregten Achsen von einer Fehlausrichtung der Sensorachsen verursacht werden oder von einer Fehlpositionierung des Sensors selbst. Es wurde daher von einer Bestimmung der Fehlausrichtungsfaktoren abgesehen.

Ergebnisse der Kalibrierung der Drehratensensoren

Tabelle 4.4 zeigt die Ergebnisse der einzelnen Messungen sowie die berechneten Skalierungsfaktoren. Es ist deutlich zu erkennen, dass die gemessenen Drehraten kleiner sind als die tatsächlichen. Im Mittel ergeben sich die Faktoren: $s_{\omega,x}^{-1} = 1.045$, $s_{\omega,y}^{-1} = 1.075$ und $s_{\omega,z}^{-1} = 1.073$. Außerdem ist zu erkennen, dass die Sensoren große Biaswerte ausweisen. Es ist allerdings auch zu erkennen, dass die Biaswerte teilweise größeren Schwankungen unterworfen sind. Daher werden die Biaswerte nicht weiter beachtet und sollten vor jeder Messreihe erneut bestimmt werden. In der Praxis zeigt sich auch, dass die Biaswerte des MotionNode noch größeren Schwankungen unterworfen sind als hier zu erkennen, daher bringt es keinen Gewinn die hier bestimmten Biaswerte als Initialwerte

zu nutzen. Eine Korrektur der einzelnen Komponenten der Messwerte kann nun mit

$$\omega_x = \frac{1}{s_{\omega,x}} \cdot (\tilde{\omega}_x - b_{\omega_x}) \quad (4.38)$$

durchgeführt werden. Es ist auch hier darauf zu achten dass die Werte jeweils im Koordinatensystem des MotionNode gegeben sind.

Achse	$b_\omega [\frac{\circ}{s}]$	$\tilde{\omega} [\frac{\circ}{s}]$	$\tilde{\omega} - b_\omega [\frac{\circ}{s}]$	$\omega [\frac{\circ}{s}]$	$\frac{1}{s_\omega}$
x	-1.8173	-192.3054	-190.4880	-200	1.0499
x	-0.8468	-259.1525	-258.3057	-270	1.0453
x	-4.0998	187.3951	191.4949	200	1.0444
x	-2.1337	257.0690	259.2027	270	1.0417
y	6.1973	-179.3517	-185.5490	-200	1.0779
y	6.4343	-246.1574	-252.5917	-270	1.0689
y	6.3370	190.4241	184.0872	200	1.0864
y	6.6974	259.6878	252.9904	270	1.0672
z	0.0260	187.0676	187.0416	200	1.0693
z	0.2085	252.3746	252.1661	270	1.0707
z	-1.5401	-187.4085	-185.8684	-200	1.0760
z	-1.5199	-252.9188	-251.3989	-270	1.0740

Tabelle 4.4: Gemessene Drehraten und Skalenfaktoren

Kalibrierung der Magnetometer

Eine genaue Betrachtung der Fehler des Magnetometers wurde nicht durchgeführt. Zur Kalibrierung des Magnetometers sei hier auf den Abschnitt 4.1.2 und den Punkt “Inbetriebnahme und Konfiguration” verwiesen.

4.3.3 Barometer

Typischerweise weisen Barometer systematische Fehler auf, die von der Temperatur und dem Druck abhängen können. Da es allerdings relativ einfach ist, die Sensorchips im Herstellungsprozess in großen Stückzahlen schnell zu testen, werden entsprechende Korrekturinformationen mit den Chips mitgeliefert. Dies ist z. B. auch bei dem für diese Arbeit verwendeten Sensor der Fall. Auf dem verbauten Chip sind Korrekturinformationen gespeichert, welche nach der Vorschrift aus dem Datenblatt [INSMA] zur Korrektur der Messdaten verwendet werden sollten. Eine solche Korrektur findet vor der Datenausgabe im Sensormodul statt. Es kann also davon ausgegangen werden, dass die ausgegebenen Daten, ausgenommen von geringen Restfehlern, frei von systematischen Fehlern sind.

Eine Anmerkung sei noch zu den Temperaturmessungen gemacht: Da der Temperatursensor zusammen mit dem Drucksensor und der Elektronik auf einem Chip verbaut ist, werden die Temperaturmessungen in geringem Maße von der Chiptemperatur beeinflusst. Abbildung 4.12 zeigt den Temperaturverlauf des verwendeten Sensors (grüne

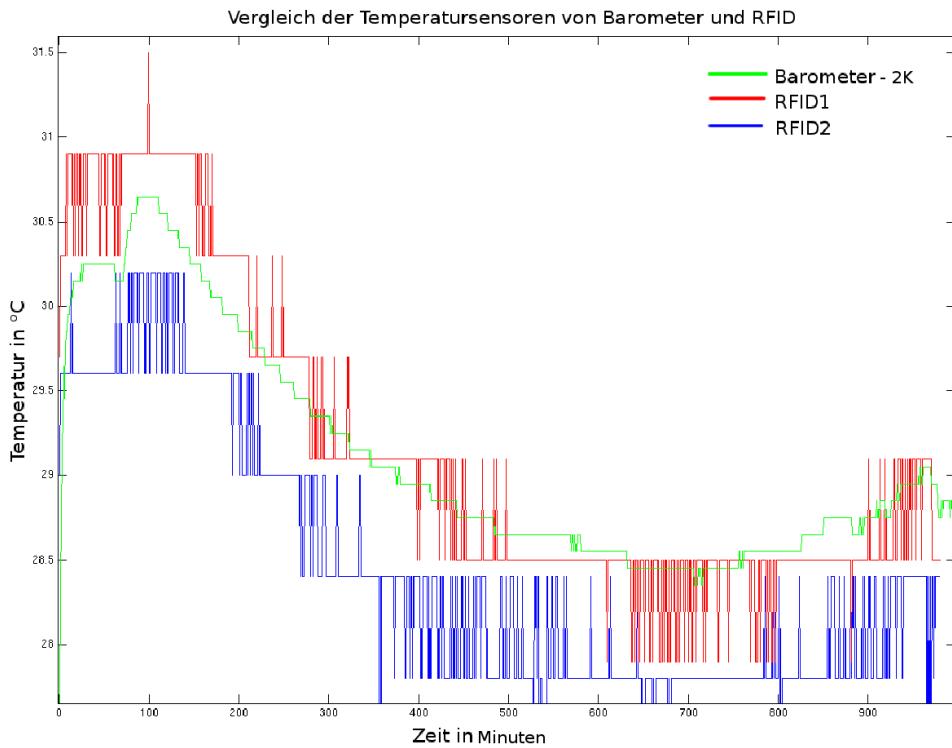


Abbildung 4.12: Vergleich verschiedener Temperatursensoren

Kurve), bereinigt um einen Offset von 2 K, im Vergleich zu zwei weiteren Sensoren (Al bis RFID Temperature Sensor Tags). Die Aufzeichnungsdauer beträgt etwa 1000 min. und die Messwerte werden in Grad Celsius angegeben. Die Temperaturerhöhung durch den Aufbau sei hiermit also zu etwa 2 K bestimmt. Im Rahmen der Messgenauigkeit von 0.5 K der Vergleichssensoren, ändert sich dieser Offset mit der Zeit nicht. Da die Vergleichsmessung nur für einen relativ kurzen Zeitraum und nur für einen kleinen Temperaturbereich durchgeführt wurde, sollte der bestimmte Wert nur als Anhaltspunkt gesehen werden. Außerdem ist die Qualität der Vergleichssensoren nicht bekannt. Die Messung zeigt allerdings, dass der Temperatursensor die Umgebungstemperatur aufzeichnet. Für genauere Messungen sollte allerdings ein separater Sensor verwendet werden. Ein solcher stand für diese Arbeit allerdings nicht zur Verfügung.

5 Künstliche Datenerzeugung

Mit den, in den Kapiteln 2, 3 und 4 vorgestellten Algorithmen und Sensoren, stehen nun Werkzeuge zur Verfügung, um eine Orientierungs-, Positions- und Geschwindigkeitsberechnung durchführen zu können. Um eine einfache Untersuchung der vorgestellten Verfahren zu ermöglichen, sind entsprechende Referenz- und Sensordaten nötig. Da in ausreichendem Maße entsprechende Datensätze nicht zur Verfügung standen, war es im Rahmen dieser Arbeit nötig, zusätzliche Daten in einer Simulationsumgebung zu erzeugen. Ein Vorteil der künstlichen Datenerzeugung ist dabei, dass direkt entsprechende Referenzen und ungestörte Daten zur Verfügung stehen und für spätere Vergleiche genutzt werden können. Für reale Daten kann es relativ aufwendig werden, entsprechende Referenzen aufzuzeichnen, sofern dies überhaupt möglich ist. Eine Schwierigkeit der Datenerzeugung ist dabei, die physikalischen Gegebenheiten möglichst genau zu simulieren. In diesem Kapitel soll die künstliche Datenerzeugung vorgestellt und auf die verwendeten Toolboxen, Funktionen und Codefragmente eingegangen werden.

5.1 Toolbox

Messwerterzeugung der Interralsensoren und GPS-Empfänger

Zur Erzeugung einer künstlichen Trajektorie und der entsprechenden Inertialsensordaten wird die “Inertial Navigation System (INS) Toolbox for MATLAB” [GPINS] von GPSSoft LLC genutzt. Diese Toolbox bildet dabei die Grundlage der Simulationsumgebung. Zum einen basiert die weitere Datenerzeugung auf der, von der Toolbox generierten Trajektorie und zum anderen werden einige, von der Toolbox zur Verfügung gestellte Funktionen, in weiteren Skripten verwendet. Die Toolbox bietet unter anderem die Möglichkeit die folgenden Daten zu generieren:

Positionsdaten: Positionsinformationen können im Navigationskoordinatensystem (n-frame), in NED- und ENU-Koordinaten und im erdfesten Koordinatensystem (e-frame), in ECEF- und LLH-Koordinaten (GPS-Koordinaten) erzeugt werden. Außerdem können sie ineinander umgerechnet werden. Es ist dabei ebenfalls möglich, Positionsmessungen zu erzeugen die denen entsprechen, die in einer Umgebung mit schlechten Bedingungen empfangen werden würden.

Geschwindigkeitsdaten: Auch die Geschwindigkeitsdaten können, wie die Positionsdaten, in Koordinaten des n-frame erzeugt werden. Hier stehen ebenfalls fehlerbehaftete und fehlerfreie Informationen zur Verfügung.

Orientierungsinformationen: Die Orientierung des Körperkoordinatensystems (b-frame) zum n-frame wird in Form von Richtungskosinusmatrizen, Eulerwinkel und Quaternionen erzeugt. Außerdem kann für Simulationen im Erdkoordinatensystem die Orientierung des n-frame zum e-frame berechnet werden. Auch hier stehen entsprechende Funktionen zur Umrechnung der Werte ineinander zur Verfügung.

Beschleunigungswerte: Beschleunigungswerte werden in Koordinaten des n-frame sowie des b-frame erzeugt. Außerdem ist es möglich, die jeweiligen Coriolisbeschleunigungen und die Schwerbeschleunigung zu erzeugen. Somit ist es möglich, Beschleunigungsmesswerte zu Erzeugen, wie sie annähernd in einer realen Umgebung gemessen werden könnten. (Beschleunigungen des b-frame, in Koordinaten des b-frame, bezüglich des Inertialkoordinatensystems (i-frame)). Auch die Erzeugung der jeweiligen Sensorfehler ist möglich, um realistische Sensordaten zu erhalten.

Drehraten: Die Drehraten werden in Koordinaten des b-frame erzeugt. Auch hier können sowohl Messungen der Erddrehrate wie auch der Transportrate erzeugt werden. Hiermit stehen, wie bei den Beschleunigungen, Drehraten des b-frame, in Koordinaten des b-frame, bezüglich des i-frame zur Verfügung. Zusammen mit der Erzeugung der jeweiligen Sensorfehler, sind auch hier realistische Sensordaten vorhanden, wie sie in einer realen Umgebung gemessen werden können.

5.2 Benötigte Messwerte

5.2.1 Profilgenerierung als Basis der Messwerterzeugung

Um die Drehraten, Beschleunigungen, Magnetfeldwerte und GPS-Messwerte erzeugen zu können, ist ein Profil aus Orientierungs-, Geschwindigkeits- und Positionsinformationen nötig. Ein solches Profil wird von der Funktion “progen.m” generiert. Die Informationen werden dabei im n-frame, in ENU-Koordinaten erzeugt. Startwerte für die Position, Geschwindigkeit und Orientierung sowie Parameter, für das zu erzeugende Profil, können beim Aufruf der Funktion übergeben werden. Diese Funktion ist in der Lage, ein Profil für unterschiedliche Flugsituationen im 3D-Raum, zu generieren. Zur Simulation des INS spielt es dabei keine Rolle, dass hier ein Flugpfad generiert wird. Eine Funktion “progencar.m”, zur Generierung eines Profils für ein Auto, steht zwar zur Verfügung, hat aber den Nachteil, dass sie nur ein Profil in einer 2D-Ebene erzeugt. Das mit “progen.m” erzeugte Profil kann dabei aus folgenden Segmenten von variabler Dauer bestehen:

- 1: Geradlinige Bewegung mit konstanter Geschwindigkeit bei konstanter Höhe, zunehmender Höhe oder abnehmender Höhe
- 2: Geradlinige Beschleunigung mit konstanter Beschleunigung bei konstanter Höhe, zunehmender Höhe oder abnehmender Höhe.
- 3: Kurve mit konstantem Radius auf konstanter Höhe.

- 4: Übergänge zwischen den vorherigen Segmenten mit folgenden Möglichkeiten: nach rechts/links Kippen/Rollen, für eine Kurve oder nach oben/unten Neigen, zum Auf- und Absteigen.

Die Parameter für die einzelnen Segmente werden dabei als $[N \times 9]$ -Matrix an die Funktion übergeben, um N Segmente zu erzeugen. Die Bedeutung der Parameter ist dabei wie folgt:

- 1: Typ (1-4) aus obiger Liste.
- 2: Dauer der Segmente vom Typ 1 und 2 (in Sekunden)
- 3: Betrag der Beschleunigung in g für ein Segment vom Typ 2.
- 4: Winkeländerung in Grad für ein Segment vom Typ 3, also der Winkel der Kurve. Die Richtung der Kurve wird durch das Vorzeichen der Verkippung/Seitenneigung (Typ 4) angegeben.
- 5: Zu erreichender Verkippungswinkel in Grad für ein Segment vom Typ 4.
- 6: Drehrate für die Verkippungsänderung (in Grad pro Sekunde) für ein Segment vom Typ 4.
- 7: Absolut zu erreichender Neigungswinkel in Grad für ein Segment vom Typ 4.
- 8: Drehrate für die Neigungsänderung (in Grad pro Sekunde) für ein Segment vom Typ 4.
- 9: Abtastdauer für die generierten Werte des Segments (in Sekunden).

Für eine genaue Beschreibung der Funktion sei auf die Beschreibungen in den Funktionsköpfen und die gedruckte Anleitung zur Toolbox verwiesen. Der folgende Programmcode zeigt die Verwendung der Funktion. `initpos` und `initvel` sind dabei die Startwerte in ENU-Koordinaten. `initdcm` ist die initiale Orientierung und `segparam` enthält die Parameter für die Profilgenerierung.

Listing 5.1: Profilgenerierung

```

1 %% Einige konstante Parameter und Werte
2 earthflg = 1; % (WGS-84 Ellipsoid)
3 deg2rad = pi/180; % Grad zu Radian
4
5 %% Profilerzeugung
6 % Initiale Position und Geschwindigkeit (in ENU-Koordinaten)
7 initpos = [0 0 0]; % Initiale Position
8 initvel = [0 0.2 0.1]; % Initiale Geschwindigkeit
9
10 % Initiale Lage in Eulerwinkel
11 phi = 0*deg2rad; % Initiale Verkippung
12 theta = 0*deg2rad; % Initiale Neigung
13 psi = 0*deg2rad; % Initialer Nordwinkel (0° := Norden)
14
15 % Berechne initiale Richtungskosinusmatrix der Objektorientierung (n- zu b-frame)
16 initdcm = eulr2dcm([phi theta psi]);
17
18 % Profil Definition

```

```

19 %      1 2 3      4 5 6      7 8 9
20 segparam = [2 4 0.01 0 -999 0 -999 0 dt; % 1 - Beschleunigung: 0.01g
21      4 NaN NaN NaN -999 0 10 1 dt; % 2 - Schräg nach oben: 10°
22      1 2 NaN NaN -999 0 -999 0 dt; % 3 - Aufwärts fahren
23      4 NaN NaN NaN -999 0 0 1 dt; % 4 - Gerade stellen: 0°
24      4 NaN NaN NaN -2 0.02 -999 0 dt; % 5 - Um -2° (links) kippen
25      3 NaN NaN 90 -999 0 -999 0 dt; % 6 - 90° (links) Kurve
26      4 NaN NaN NaN 0 2 -999 0 dt]; % 7 - Zurück kippen (rechts)
27
28 profile = progen(initpos, initvel, initdcm, segparam); % generiere Profil
29 time = profile(:,19); % Laufzeit in Sekunden
30 npts = size(profile, 1); % Anzahl der Werte
31
32 % Extrahiere benötigte Werte
33 DCMnb_prof = profile(:,10:18); % Ideale DCMnb Matrix [M x 9]-Matrix
34 vel_prof_L = profile(:,4:6); % Geschwindigkeitsprofil in ENU-Koordinaten
35 pos_prof_L = profile(:,1:3); % Positionsprofil in ENU-Koordinaten
36
37 %% Erzeugung der idealen GPS-Werte
38 % Initiale Position in LLH-Koordinaten
39 PB_deg = [51.7067 8.7711];
40 PB_rad = PB_deg * deg2rad;
41 latlonstart = [PB_rad(1) PB_rad(2)];
42 heightstart = 165;
43
44 % Umwandlung des Profil in ideale LLH-Koordinaten
45 [lat_prof, lon_prof, height_prof] = profconv(pos_prof_L, [latlonstart heightstart]);
46
47 for k=1:npts
48     % Erzeuge DCMel Rotationsmatrix vom e-frame zum n-frame (nach Norden Ausgerichtet)
49     % Diese Matrix wird nur zur Berechnung der Erddrehrate benutzt, auch wenn sie
50     % bei der Berechnung der Transportrate mit angegeben ist (wegen vertmech = 0).
51     DCMel = llw2dcm([lat_prof(k) lon_prof(k) 0]);
52     DCMel_prof(k, :) = [DCMel(1,1:3), DCMel(2,1:3), DCMel(3,1:3)];
53 end;

```

`vel_prof_L` und `pos_prof_L` sind $[M \times 3]$ -Matrizen. Sie enthalten M Geschwindigkeits- und M Positionsvektoren in ENU-Koordinaten. `DCMnb_prof` ist eine $[M \times 9]$ -Matrix. Sie enthält M DCM-Matrizen, die eine Transformation vom n-frame in den b-frame beschreiben. Eine DCM-Matrix ist dabei als Zeilenvektor wie folgt in einer Zeile der Matrix abgelegt:

```
1 DCMnb_prof(m,1:9) = [DCMnb(1,1:3,m) DCMnb(2,1:3,m) DCMnb(3,1:3,m)]
```

`lat_prof`, `lon_prof` und `height_prof` enthalten die idealen LLH-Werte der Position des b- und n-frame im e-frame. `DCMel_prof` enthält DCM Matrizen, die eine Transformation vom e-frame ins n-frame beschreiben. Sie werden später zur Berechnung der Messwerte für die Erddrehrate benötigt.

5.2.2 Inertialnavigation

Für die Strapdown-Rechnung werden zur Auswertung der Gleichungen (2.15) und (2.23), jeweils Messwerte für die Drehraten $\vec{\omega}_{ib}^b$ und die Beschleunigungen \vec{a}_{ib}^b benötigt. Es handelt sich um Messwerte des b-frame, bezüglich des i-frame, in Koordinaten des b-frame. Die verwendete Toolbox generiert dabei jeweils die entsprechenden Inkremente in einen Zeitschritt, also

$$\Delta\vec{\omega}_{ib}^b = \vec{\omega}_{ib}^b \cdot \Delta t \text{ und } \Delta\vec{a}_{ib}^b = \vec{a}_{ib}^b \cdot \Delta t. \quad (5.1)$$

Drehraten

Die gemessene Drehrate $\vec{\omega}_{ib}^b$ setzt sich dabei aus der Summe verschiedener Drehraten, gemessen in Koordinaten des b-frame, zusammen:

- Die tatsächliche Drehrate $\vec{\omega}_{nb}^b$ des b-frame, bezüglich des n-frame, gibt die relevanten Drehungen des Objektes an. Die Matlab-Funktion “gendthet.m” wird zur Generierung der entsprechenden Winkelinkrementen genutzt.
- Die Transportrate $\vec{\omega}_{en}^b$ gibt die Drehung des n-frame, bezüglich des e-frame an. Die Transportrate beschreibt den Effekt, dass der n-frame, bei der Bewegung über die gekrümmte Erdoberfläche, immer tangential zur Erdoberfläche ausrichtet bleibt. Die Matlab-Funktion “gendetcr2.m” generiert die entsprechenden Winkelinkrementen.
- Die Erddrehrate $\vec{\omega}_{ie}^b$ gibt die Drehung des e-frame, bezüglich des i-frame an. Die Erddrehrate beschreibt damit die gemessene Erdrotation. Die Matlab-Funktion “earthrot.m” kann zur Generierung der entsprechenden Winkelinkrementen genutzt werden.

Zusammen ergibt dies dann:

$$\vec{\omega}_{ib}^b = \vec{\omega}_{nb}^b + \vec{\omega}_{en}^b + \vec{\omega}_{ie}^b. \quad (5.2)$$

Der folgende Programmcode zeigt die Generierung der jeweiligen Messwerte für die Drehraten:

Listing 5.2: Generierung der Drehraten

```

1 %% Erzeuge delta-theta (omega_ib_b*dt) Profile
2 fprintf(1, 'Erzeuge delta-theta Profile\n')
3
4 % Anteil von delta-theta verursacht durch die Erddrehrate
5 deltaer = earthrot(time, DCMel_prof, DCMnb_prof);
6
7 % Anteil von delta-theta verursacht durch die Transportrate
8 deltacr = gendetcr2(lat_prof, vel_prof_L, height_prof, ...
9 time, DCMnb_prof, DCMel_prof, earthflg);
10
11 % Anteil von delta-theta verursacht durch die Körperbewegung im n-frame
12 dthetbody = gendthet(DCMnb_prof);
13
14 % dthetbody := omega_nb_b * tdint
15 % deltaer := omega_ie_b * tdint
16 % deltacr := omega_en_b * tdint
17 % deltheta := omega_ib_b * tdint
18 deltheta = dthetbody + deltaer + deltacr;    % Ideale (Fehlerfreie) Winkelinkrementa

```

Beschleunigungen

Die gemessene Beschleunigung \vec{a}_{ib}^b setzt sich dabei ebenfalls aus der Summe verschiedener Beschleunigungen, gemessen in Koordinaten des b-frame, zusammen:

- Die eigentliche Beschleunigung \vec{a}_{nb}^b des b-frame, bezüglich des n-frame, gibt die relevante Beschleunigung des Objektes an. Die Matlab-Funktion “gendv.m” wird zur Generierung der entsprechenden Geschwindigkeitsinkrementen genutzt.

- Die Coriolisbeschleunigung \vec{a}_c^b setzt sich zusammen aus der Beschleunigung, verursacht durch die Bewegung auf der rotierenden Erde und der Bewegung im rotierenden n-frame. Außerdem wird in der Realität die lokale Schwerebeschleunigung \vec{g}_l^b von den Sensoren gemessen. Die Matlab-Funktion “gendvcor2.m” wird genutzt, um jeweils für die drei genannten Beschleunigungen die Geschwindigkeitssinkamente zu generieren.

Zusammen ergibt dies wiederum:

$$\vec{a}_{ib}^b = \vec{a}_{nb}^b + \vec{a}_c^b + \vec{g}_l^b. \quad (5.3)$$

Der folgende Programmcode zeigt die Generierung der jeweiligen Messwerte für die Beschleunigungen:

Listing 5.3: Generierung der Beschleunigungen

```

1 %% Erzeuge delta-v (a_ib_b*dt) Profile
2 fprintf(1, 'Erzeuge delta-v Profile\n')
3
4 % Anteil von delta-v verursacht durch die Körperbewegung im n-frame
5 deltav_b = gendv(vel_prof_L, DCMnb_prof);
6
7 % Anteil von delta-v verursacht durch Coriolisbeschleunigungen und Schwerebeschleunigung
8 dvcor = gendvcor2(lat_prof, vel_prof_L, height_prof, ...
9 time, DCMnb_prof, DCMel_prof, earthflg);
10
11 % deltav_b := a_nb_b * tdint
12 % dvcor := a_c * tdint
13 % dvtot := a_ib_b * tdint
14 dvtot = deltav_b + dvcor;      % Ideale (Fehlerfreie) Messwerte

```

Verrauschen der generierten Werte

Nach Gleichung (4.17) werden nun noch die Fehler, verursacht durch die Skalenfehler in \mathbf{M}_x und die Fehler, bestehend aus Bias \vec{b}_x sowie Rauschen \vec{n}_x , hinzugaddiert, wobei $x \in [\omega, a]$. Dazu werden die Funktionen “gentherr2.m” zur Generierung der Fehleranteile der Winkelinkremente und “gendverr2.m” zur Generierung der Fehleranteile der Geschwindigkeitsinkremente genutzt. Die Ausrichtungsfehler werden nicht modelliert. Der folgende Programmcode zeigt die Generierung der jeweiligen additiven Fehlerterme und die Erzeugung der Messwerte.

Listing 5.4: Generierung der Fehler in den Inkrementen

```

1 %% Erzeuge additive Fehler für die Winkelinkremente
2 % Gyroskop Fehlerparameter
3 thxbias = 1;          % 1 deg/root-hr Gyroskop Biasänderung
4 thybias = 1;
5 thzbias = 1;
6 thxserr = 0;          % 0% Gyroskop Skalenfehler
7 thyserr = 0;
8 thzserr = 0;
9 thxstdev = 0.9;        % 0.9 deg/root-hour Gyroskop Rauschleistungsdichte
10 thystdev = 0.9;
11 thzstdev = 0.9;
12
13 dthparam = [thxbias  thybias  thzbias]; % Erzeuge Matrix für

```

```

14         thxsferr thysferr thzsferr; % GENTHERR2
15         thxstdev thystdev thzstdev];
16
17 [dtherr, bias_dtheta] = gentherr2(deltheta, time, dthparam, 98765); % Fehlererzeugung
18
19 est_dtheta = deltheta + dtherr;           % Erzeuge Profil mit Messwerten
20
21 %% Erzeuge additive Fehler für die Geschwindigkeitsinkremente
22 % Beschleunigungssensor Fehlerparameter
23 vxbias = 0.02;      % 0.02 m/s per root-hour Beschleunigungssensor Biasänderung
24 vybias = 0.02;
25 vzbias = 0.02;
26 vxsferr = 0;        % 0% Beschleunigungssensor Skalenfehler
27 vysferr = 0;
28 vzsferr = 0;
29 vxstdev = 0.03;     % 0.03 m/s per root-hour Beschleunigungssensor Rauschleistungsdichte
30 vystdev = 0.03;
31 vzstdev = 0.03;
32
33 dvparam = [vxbias vybias vzbias;          % Erzeuge Matrix für
34             vxsferr vysferr vzsferr;          % GENDVERR2
35             vxstdev vystdev vzstdev];
36
37 [dverr, bias_dv] = gendverr2(dvtot, time, dvparam, 76542); % Fehlererzeugung
38
39 est_dv = dvtot + dverr;           % Erzeuge Profil mit Messwerten

```

5.2.3 Generierung der GPS-Messwerte

Da von den Algorithmen in den folgenden Kapiteln, auch GPS-Messwerte benötigt werden, müssen GPS-Messwerte erzeugt werden. Die Messwerte bestehen dabei aus Positions- und Geschwindigkeitsmessungen. Als Basis zur Simulation der GPS-Messwerte werden der ENU-Pfad und die ENU-Geschwindigkeiten genutzt, die von der Funktion “progen.m” generiert werden. Hierbei ist darauf zu achten, dass die Position der GPS-Antenne an der selben Position angenommen wird, wie der Ursprung des b- und n-frames. Möchte man eine Verschiebung der GPS Antenne berücksichtigen so, sind die Positionen und Geschwindigkeiten entsprechend anzupassen ([WEN07] S. 207 und S. 209).

“Verrauschen” der Werte: Zuerst werden die Werte für Position und Geschwindigkeit, gemäß den Funktionen (4.3) und (4.7) verrauscht. Dabei werden die Varianzen jeweils auf konstante Werte, unabhängig vom Zeitpunkt, gesetzt.

Verringern der GPS-Datenrate: Da ein realer GPS-Empfänger Daten meist nur mit einer Rate von 1 Hz liefert, wird die Datenrate durch Verwerfen überflüssiger Werte verringert.

Filtern der Werte: Optional werden die verrauschten und in der Datenrate angepassten Positions- und Geschwindigkeitswerte mit einem Kalman-Filter, der die Newton’sche Bewegungsgleichung als Systemmodell enthält, gefiltert. Dies ist hier für

jeweils ein Komponentenpaar gezeigt:

$$\begin{pmatrix} p^n \\ v^n \end{pmatrix}_k = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p^n \\ v^n \end{pmatrix}_{k-1} + \begin{pmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{pmatrix} a_k^n \quad (5.4)$$

$$\begin{pmatrix} \tilde{p}^n \\ v^n \end{pmatrix}_k = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p^n \\ v^n \end{pmatrix}_k + \begin{pmatrix} n_p \\ n_v \end{pmatrix}_k \quad (5.5)$$

Der Systemzustand ist dabei die GPS-Schätzung für Position und Geschwindigkeit und (\cdot) die Messung für Position und Geschwindigkeit. Δt ist hier das GPS-Abtastintervall und $a_k^n \sim \mathcal{N}(0, \sigma_{a,GPS}^2)$ das Systemrauschen. $\sigma_{a,GPS}^2$ ist abhängig von den in der Trajektorie maximal vorkommenden Beschleunigungen. $\sigma_{a,GPS}$ wird auf einen konstanten Wert gesetzt, der der maximalen Beschleunigung entspricht. Prinzipiell ließe sich $\sigma_{a,GPS}$ auch anhand des Systemmodells ermitteln. Das ist hier aber nicht sinnvoll, da die so bestimmte Varianz der Beschleunigung in Situationen mit starken Beschleunigungen zu niedrig ist und dann zu großen Fehlern in der Schätzung führen würde. Im Mittel mag dies zu einem größeren Schätzfehler führen aber da Situationen mit höherer Fahrtdynamik von größerer Bedeutung sind ist dieses Vorgehen hier sinnvoller. Das Filter wurde mit Hilfe einer Kalman-Filter-Toolbox [KMFLT] implementiert. Dieses Filter soll die in einem realen Empfänger durchgeführte Filterung nachbilden.

Umwandeln in LLH-Messungen: Im Normalfall liefert ein GPS-Empfänger Messwerte in Form von LLH-Messwerten. Daher werden im nächsten Schritt die verrauschten und in der Datenrate verringerten sowie optional gefilterten ENU-Positionswerte, in LLH-Werte umgewandelt. Hierzu wird die Funktion “provconf.m” verwendet. Der Funktion wird dazu das ENU-Profil und eine Startposition in LLH-Koordinaten, übergeben. Eine Schwierigkeit bei der Umwandlung besteht darin, den im n-frame erzeugten Pfad auf die gekrümmte Erdoberfläche zu projizieren. Dies wird von der Funktion “provconf.m” durchgeführt.

Der folgende Programmcode zeigt die Generierung und Filterung der jeweiligen GPS-Messwerte:

Listing 5.5: Erzeugung von GPS-Messungen

```

1 %% Erzeuge GPS Messungen
2 fprintf(1, 'Erzeuge GPS Messungen\n');
3
4 %% Erzeuge additive Fehler für Positions- und Geschwindigkeitsmessungen
5 % GPS Positions Fehlerparameter
6 posxbias = 0;
7 posybias = 0;
8 poszbias = 0;
9 posxsferr = 0;
10 posysferr = 0;
11 poszsferr = 0;
12 posxstdev = 10/dt;      % 10m Standardabweichung je horizontaler Komponente
13 posystdev = posxstdev;
14 poszstdev = 20/dt;       % 20m Standardabweichung in der Höhenmessung
15
16 posparam = [posxbias posybias poszbias;    % Erzeuge Matrix für
17           posxsferr posysferr poszsferr;   % GENDVERR

```

```

18         posxstdev posystdev poszstdev];
19
20 % GPS Geschwindigkeits Fehlerparameter
21 velxbias = 0;
22 velybias = 0;
23 velzbias = 0;
24 velxsferr = 0;
25 velysferr = 0;
26 velzsferr = 0;
27 velxstdev = 0.5/dt;      % 0.5m/s Standardabweichung je horizontaler Komponente
28 velystdev = velxstdev;
29 velzstdev = 15/dt;        % 15m/s Standardabweichung in der Vertikalen Komponente
30
31 % Erzeuge Fehler
32 posdelta = zeros(npts, 3);
33 poserr = gendverr(posdelta, [0; time], posparam, 76543);
34 velerr = gendverr(posdelta, [0; time], velparam, 76544);
35
36 %% Taste Werte ab und Erzeuge Profil mit Messwerten
37 dt_gps = 1;                % GPS Abtastintervall
38 idx_gps_abt = 1:dt_gps:dt;npts;          % Erzeuge Abtastzeitpunkte
39 pos_gps_abt = pos_prof_L(idx_gps_abt,:);    % Taste Positionen ab
40 vel_gps_abt = [diff(est_pos_gps_abt); 0 0 0]; % Bestimme Geschwindigkeitsmesswerte
41
42 % Erzeuge Profil mit Messwerten
43 est_pos_gps_abt = pos_gps_abt + poserr(idx_gps_abt,:);
44 est_vel_gps_abt = vel_gps_abt + velerr(idx_gps_abt,:);
45
46 %% GPS Kalman-Filterung
47 fprintf(1, 'GPS Kalman-Filterung\n')
48
49 F_GPS = [1 0 0 dt_gps 0      0;           % Systemmodell
50          0 1 0 0      dt_gps 0;
51          0 0 1 0      0      dt_gps;
52          0 0 0 1      0      0;
53          0 0 0 0      1      0;
54          0 0 0 0      0      1       ];
55
56
57 H_GPS = [1 0 0 0 0 0;     % Messmodell
58          0 1 0 0 0 0;
59          0 0 1 0 0 0;
60          0 0 0 1 0 0;
61          0 0 0 0 1 0;
62          0 0 0 0 0 1];
63
64 % Systemrauschen
65 Q_GPS = [eye(3,3)*dt_gps^2/2 zeros(3,3); zeros(3,3) eye(3,3)*dt_gps] * ...
66 blkdiag(3.7, 2.5, 0.7, 3.7, 2.5, 0.7) * ...
67 [eye(3,3)*dt_gps^2/2 zeros(3,3); zeros(3,3) eye(3,3)*dt_gps]';
68
69 % Messrauschen
70 R_GPS = blkdiag(posxstdev.^2, posystdev.^2, poszstdev.^2, ...
71                  velxstdev.^2, velystdev.^2, velzstdev.^2)*dt^2;
72
73 initx = profile(1,1:6)';    % initialer Systemzustand
74
75 % Initiale Kovarianz
76 initV = ...
77 [blkdiag(posxstdev/3, posystdev/3, poszstdev)*dt, ...
78  blkdiag(posxstdev*velxstdev, posystdev*velystdev, poszstdev*velzstdev)/1e3*dt;
79  blkdiag(posxstdev*velxstdev, posystdev*velystdev, poszstdev*velzstdev)/1e3*dt, ...
80  blkdiag(velxstdev, velystdev, velzstdev/8)*dt].^2;
81
82 y_GPS = [est_pos_gps_abt, est_vel_gps_abt]';    % Messwerte
83
84 % Filterung
85 [xfilt_GPS, Vfilt_GPS, VVfilt_GPS, loglik] = ...
86 kalman_filter(y_GPS, F_GPS, H_GPS, Q_GPS, R_GPS, initx, initV);
87

```

```

88 xfilt_GPS = xfilt_GPS'; % gefilterte Werte
89
90 %% Umwandlung in Geschwindigkeits und LLH-Messungen
91 % Erzeuge LLH-Positionsmesswerte (NaN wenn kein Wert gemessen)
92 [est_lat_prof, est_lon_prof, est_height_prof] = ...
93 profconv(xfilt_GPS(:,1:3), [latlonstart heightstart]);
94 est_llh_gps = ones(npts, 3)*NaN;
95 est_llh_gps(idx_gps_abt,:) = [est_lat_prof; est_lon_prof; est_height_prof]';
96
97 % Erzeuge Geschwindigkeitsmesswerte (NaN wenn kein Wert gemessen)
98 estfilt_vel_gps = ones(npts, 3)*NaN;
99 estfilt_vel_gps(idx_gps_abt,:) = xfilt_GPS(:,4:6);

```

Alternativ zur obigen Vorgehensweise steht auch die “Satellite Navigation (SatNav) ToolBox 3.0” [GPSAT] von GPSSoft LLC zur Erzeugung von GPS-Messungen und die “Navigation System Integration and Kalman Filter Toolbox 2.0” [GPFLT] zur Filterung von GPS-Messungen zur Verfügung. Die erste Toolbox bietet dabei die Möglichkeit, realistische GPS-Messungen auf Basis der Simulation eines kompletten GPS-Empfängers zu erzeugen. Beide Toolboxen wurden allerdings aufgrund ihrer Komplexität im Rahmen dieser Arbeit nicht verwendet. Eine Verwendung in nachfolgenden Arbeiten ist jedoch vorgesehen.

5.2.4 Messwerterzeugung des Barometer

Da die Erzeugung der Messwerte für die Simulation der barometrischen Höhenmessung bereits in Abschnitt 3.4 gezeigt wurde ist es nicht nötig hier noch einmal darauf einzugehen. Zur Erzeugung der für die barometrische Höhenmessung benötigten Daten werden die in Abschnitt 3.4 aufgeführten Funktionen genutzt. Als Basis wird das generierte Höhenprofil verwendet. Die Datenerzeugung wird dabei gemäß den Gleichungen (3.11), (3.15), (3.17), (3.18), (3.20) und (3.21) durchgeführt.

5.2.5 Messwerterzeugung des Magnetometer

Zur Erzeugung der Messwerte für das Magnetometer wird Gleichung (4.22) an der entsprechenden Stelle, im Matlab-Skript zur Datenerzeugung, ausgewertet. $\tilde{\vec{h}}_{mag,k}^b$ wird in Koordinaten des b-frame benötigt. Als Basis werden hierbei die generierte Orientierungen verwendet. Damit ergibt sich:

$$\tilde{\vec{h}}_{mag,k}^b = (\mathbf{C}_{b,k}^n)^T \cdot \vec{h}_{mag_0}^n + \vec{n}_{h_{mag,k}}, \quad (5.6)$$

wobei $\vec{h}_{mag_0}^n$ das Magnetfeld am jeweiligen Ort, in Koordinaten des n-frame, ist. Für Paderborn gilt hier [CALMG]:

$$\vec{h}_{mag_0}^n = \begin{pmatrix} 19148.7 \\ 418 \\ 45077.9 \end{pmatrix} \text{nT.} \quad (5.7)$$

Diese Gleichung stellt zwar nur eine Näherung dar, da das Magnetfeld im Allgemeinen von der Position abhängt. Aber aufgrund der langsamen Veränderlichkeit des Magnetfeldes mit der Position, genügt dies hier für die Simulation im Bereich von mehreren 100 km. Der folgende Programmcode zeigt die Generierung der jeweiligen Magnetometer-Messwerte:

```

1 %% Erzeugung der Magnetometerwerte
2 h_n = [19148.7 418 45077.9];      % Magnetisches Feld bei [latlonstart heightstart]
3
4 % Magnetometer Fehlerparameter
5 magxstdev = 110;      % 110 nT Standardabweichung der Messwerte
6 magystdev = 110;
7 magzstdev = 110;
8 magparam = [magxstdev magystdev magzstdev];
9
10 for k=1:npts
11     % Berechne magnetisches Feld in NED Koordinaten:
12     dcmnb = [DCMnb_prof(k,1:3); DCMnb_prof(k,4:6); DCMnb_prof(k,7:9)]; % bestimme DCMnb
13     h_b(k,:) = h_n * dcmnb' + magparam .* randn(1, 3);    % (dcmnb * h_n') = h_n * dcmnb'
14 end;

```

Das folgende Blockdiagramm zeigt alle Blöcke der Datengenerierung:

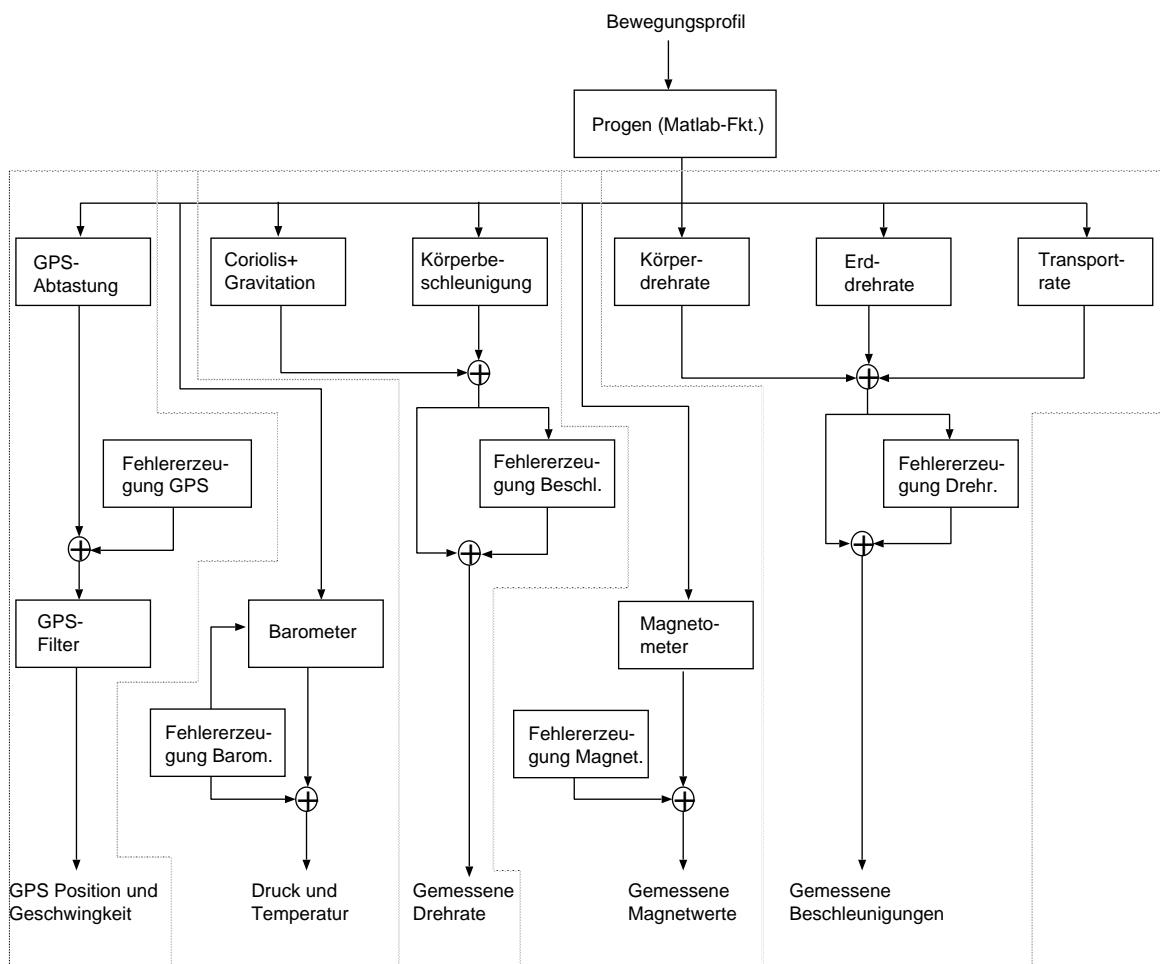


Abbildung 5.1: Datengenerierung

6 GPS/INS-Integration

In diesem Kapitel soll ein Verfahren zur Kombination von GPS und inertialer Navigation vorgestellt werden. Eine solche Kombination führt zu einem sogenannten integrierten Navigationssystem. Das Ziel einer solchen Kombination ist es, eine bessere und robustere Navigationslösung zu erhalten als es die beiden einzeln Systeme bieten würden. In diesem Kapitel sollen kurz die Gründe und Vorteile einer solchen Kombination genannt werden. Als nächstes sollen die benötigten Gleichungen und Algorithmen des Verfahrens vorgestellt werden um anschließend implementiert und getestet zu werden.

Im Wesentlichen werden sich die hier gemachten Ausführungen an das Kapitel 8 “Anwendungsbeispiel GPS/INS-Integration” aus dem Buch “Integrierte Navigationssysteme” von Jan Wendel [WEN07] anlehnen. Für ausführlichere Ausführungen, speziell zu den komplexeren Filterverfahren, sei auf die genannte Literatur verwiesen. Bei den hier gemachten Herleitungen sollen die Zusammenhänge verständlich aber kurz dargestellt werden. Für die ausführliche Herleitung sei auch hier auf die Literatur verwiesen. Es gilt dabei aber unter anderem zu beachten, dass es geringe Unterschiede in der Herleitung geben kann, z. B. wird hier die Höhe positiv nach oben gezählt, während sie in der genannten Literatur positiv nach unten gezählt wird.

6.1 Grundlagen

Im Abschnitt 4.2.2 wird gezeigt, dass die verwendeten Messwerte für die Inertialnavigation mit Fehlern behaftet sind. Diese Fehler führen dazu, dass auch das in Kapitel 2 vorgestellte Verfahren zu einer fehlerhaften Lösung kommt. Aufgrund der mehrfachen Integration der Messwerte über die Zeit, wachsen diese Fehler mit der Zeit immer weiter an. Die Navigationslösung des Inertialnavigationssystems ist also nur kurzzeitstabil. Für kurze Zeit ist das INS aber in der Lage eine relativ genaue Navigationslösung mit hoher Datenrate von typischerweise 50 Hz bis 100 Hz oder mehr zu liefern. Die Verfügbarkeit der Messungen für das INS hängt außerdem kaum von externen Einflüssen ab, wodurch die Verfügbarkeit der Messungen für das INS fast immer gewährleistet ist um eine Navigationslösung zu berechnen.

Auf der anderen Seite bietet das GPS langzeitstabile Messungen. Diese Messungen liegen meist in Form von Geschwindigkeitsmessungen und Positionsmessungen vor. Im Gegensatz zu einem INS, liefert ein GPS-Empfänger die Messwerte meist nur mit einer Datenrate von 1 Hz bis 4 Hz. Außerdem können die Messwerte, wie in Abschnitt 4.2.1 gezeigt, stark fehlerbehaftet sein. Zusätzlich ist eine Verfügbarkeit der GPS-Messungen

nicht immer gewährleistet, da es je nach Umgebungsbedingungen zu Ausfällen beim Empfang der GPS-Satelliten kommen kann.

Mit Hilfe der GPS-Messungen ist es möglich, die anwachsenden Fehler in der Navigationslösung des INS zu erkennen und zu korrigieren. Auf der anderen Seite ist es mit der Navigationslösung des INS möglich, zwischen zwei GPS-Messungen trotzdem Positions- und Geschwindigkeitswerte auszugeben. Durch die geschickte Kombination beider Verfahren ist es außerdem möglich, eine Navigationslösung zu bestimmen, die besser sein kann als die jeweiligen Einzellösungen von GPS und INS. Die Kombination der Daten des GPS und INS wird meist mit einem Kalman-Filter durchgeführt. Oft kommen dabei sogenannte Error State Space Kalman-Filter zum Einsatz, die mit Hilfe der GPS-Messungen die Fehler in der Lösung des INS schätzen. Daraufhin können die Fehler korrigiert werden. Zusätzlich ist es unter Umständen möglich, deterministische Fehler wie z. B. einen Bias der Inertialsensoren zu schätzen und zu korrigieren. Auch die Schätzung weiterer Fehler wie Skalierungsfehler und Ausrichtungsfehler wäre denkbar. Dadurch wird die Navigationslösung des INS weiter verbessert. Es existieren unterschiedliche Ansätze, eine solche Kombination durchzuführen:

Schwach gekoppelte Systeme (Loosely Coupled System): Bei einem Loosely Coupled System werden GPS-Positions- und Geschwindigkeitsmessungen verwendet, um die Fehler des INS zu bestimmen und zu korrigieren. Der Vorteil hierbei ist, dass die Kombination mit relativ geringem Aufwand realisiert werden kann. Andererseits ist für eine vollständige Stützung des INS immer der Empfang von mindestens 4 Satelliten nötig, da sonst keine GPS-Werte zur Verfügung stehen.

Stark gekoppelte Systeme (Tightly Coupled System): Bei einem Tightly Coupled System werden zusätzlich noch Pseudorange- und Deltarangemessungen zu den einzelnen Satelliten verwendet. Der Vorteil hierbei ist, dass eine eingeschränkte Stützung auch mit weniger als 4 Satelliten durchgeführt werden kann. Allerdings ist der Aufwand der Implementierung höher. Zusätzlich kann hier manchmal auch der GPS-Empfänger durch das INS bzw. die Inertialsensoren unterstützt werden, da diese Informationen zur Trajektoriendynamik liefern können.

Ultra-Tight und Deep Integration: Bei einem Deep Integration System kann die Integration bis zur Verarbeitung der eigentlichen Samples der Signalempfänger führen. Ein solches System ist noch einmal deutlich komplexer zu implementieren, bietet aber wiederum eine gesteigerte Genauigkeit und Robustheit.

In dieser Arbeit soll ein schwach gekoppeltes System betrachtet und ein Error State Space Kalman-Filter entworfen werden.

6.2 Filterentwurf

6.2.1 Systemmodell

Für den Entwurf des Filters ist zuerst zu überlegen wie dieser aufgebaut werden soll. Es wird eine “Error State Space”-Formulierung gewählt. Dabei sollen die Fehler der Navigationslösung des INS, wie der Positionsfehler $\Delta \vec{p}$, der Geschwindigkeitsfehler $\Delta \vec{v}$ und

dem Orientierungsfehler $\Delta\vec{\psi}$ geschatzt werden. Die Fehler sollen dabei in Koordinaten des n-frame geschatzt werden. Außerdem sollen jeweils die Fehler in den Biaswerten des Beschleunigungssensors \vec{b}_a und des Drehratensensors \vec{b}_ω , in Koordinaten des b-frame, geschatzt werden. Die Messwerte der Beschleunigungssensoren und Drehratensensoren werden dabei als bekannte Werte behandelt und direkt im Propagationsschritt des Filters verarbeitet, obwohl sie eigentlich Messwerte sind. Das Rauschen in den Messwerten wird dabei dem Systemrauschen hinzugerechnet. Dieses Vorgehen bringt kaum Nachteile, vereinfacht aber den Filterentwurf und verringert spater die Rechenlast. Bei einem Error State Space Kalman-Filter dient der Propagationsschritt lediglich zur Fortschreibung der Kovarianzmatrix des Schatzfehlers. Eine Propagation des Zustandsvektors ist nicht notig, da die Fehler nach einem Messschritt korrigiert werden und der Zustandsvektor zu Null gesetzt wird. Die eigentliche Berechnung der INS-Navigationslosung findet dabei im Strapdown-Algorithmus statt.

Hieraus resultiert für das Kalman-Filter also ein Zustandsvektor der Dimension $[15 \times 1]$:

$$\Delta \vec{x} = \begin{pmatrix} \underbrace{\Delta x_n, \Delta x_e, \Delta x_d}_{\text{Positionsfehler } \Delta \vec{p}^T}, \quad \underbrace{\Delta v_{eb,n}^n, \Delta v_{eb,e}^n, \Delta v_{eb,d}^n}_{\text{Geschwindigkeitsfehler } \Delta \vec{v}^T}, \quad \underbrace{\Delta \alpha, \Delta \beta, \Delta \gamma}_{\text{Orientierungsfehler } \Delta \vec{\psi}^T} \\ \text{Fehler in } \vec{b}_a^T \\ \text{Fehler in } \vec{b}_\omega^T \end{pmatrix}^T \quad (6.1)$$

$\Delta\vec{p} = (\Delta x_n, \Delta x_e, \Delta x_d)^T$ gibt dabei die Positionsfehler in m und $\Delta\vec{v} = (\Delta v_{eb,n}^n, \Delta v_{eb,e}^n, \Delta v_{eb,d}^n)^T$ die Geschwindigkeitsfehler in $\frac{m}{s}$, jeweils in Nord- Ost- und Unten-Richtung im n-frame, an. $\Delta\vec{\psi} = (\Delta\alpha, \Delta\beta, \Delta\gamma)^T$ gibt die Eulerwinkel an, die eine Drehung vom realen n-frame in den geschätzten n-frame beschreiben. $\Delta b_a = (\Delta b_{a,x}, \Delta b_{a,y}, \Delta b_{a,z})^T$ und $\Delta\vec{b}_\omega = (\Delta b_{\omega,x}, \Delta b_{\omega,y}, \Delta b_{\omega,z})^T$ geben jeweils die Abweichungen zu den Biasen der Beschleunigungssensoren, in $\frac{m}{s^2}$ und den Biasen der Drehratensensoren in $\frac{rad}{s}$ an.

Das Systemmodell muss, neben den deterministischen Inertialsensorfehlern, auch die Auswirkungen der Fehler auf die Navigationslösung beschreiben. Die Gleichungen hierzu können aus den Gleichungen für die inertiale Navigation aus Kapitel 2 gewonnen werden. Da diese Gleichung nichtlinear sind, ist es nötig, den Systemzustand zu linearisieren. Im Folgenden soll nun die Herleitung für die einzelnen Fehlerterme durchgeführt werden. Die Herleitung basiert dabei auf den Zeitkontinuierlichen Gleichungen und liefert folglich auch ein Systemmodell welches im Zeitkontinuierlichen gegeben ist.

Vorab einige kurze Anmerkungen zur Nomenklatur und einfachen Umrechnungen: Für kleine Differenzen in der Position gilt für die Umrechnung von LLH-Differenzen in

NED-Differenzen:

$$\Delta x_n = (R_n + h) \cdot \Delta \varphi \quad (6.2)$$

$$\Delta x_e = (R_e + h) \cdot \cos(\varphi) \cdot \Delta \lambda \quad (6.3)$$

$$\Delta x_d = -\Delta h. \quad (6.4)$$

Für die Richtungskosinusmatrix $\mathbf{C}_{\hat{n}}^n$, die die Transformation von einem geschätzten n-frame (\hat{n}) in den idealen n-frame (n) beschreibt, gilt bei kleinen Eulerwinkel $\vec{\psi} = (\alpha, \beta, \gamma)^T$

$$\mathbf{C}_{\hat{n}}^n \approx \begin{pmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{pmatrix} = \mathbf{I} + [\vec{\psi} \times]. \quad (6.5)$$

Hierbei ist \mathbf{I} die Einheitsmatrix und $[\vec{\psi} \times]$ die kreuzproduktbildende Matrix von $\vec{\psi}$. Die kreuzproduktbildende Matrix wird im Folgenden auch in anderen Zusammenhängen verwendet.

Positionsfehlerdifferentialgleichungen

Die Änderungen der Positionsfehler lassen sich aus den Gleichungen (2.25) bis (2.27) gewinnen, also

$$\dot{\varphi} = \frac{v_{eb,n}^n}{R_n + h} \quad (6.6)$$

$$\dot{\lambda} = \frac{v_{eb,e}^n}{(R_e + h) \cos(\varphi)} \quad (6.7)$$

$$\dot{h} = -v_{eb,d}^n. \quad (6.8)$$

Dabei ist zu erkennen, dass die Änderungen des Positionsfehlers im Wesentlichen von den Geschwindigkeitsfehlern, Höhenfehlern und Breitengradfehlern abhängen. Der Krümmungsradius der Erde wird als konstant angenommen, da ein Fehler im Breitengrad kaum Einfluss darauf hat und auch die Änderung mit dem Breitengrad gering ist. Für die Änderung des Fehlers in Nordrichtung ergibt sich durch Taylorreihenentwicklung von Gleichung (6.6):

$$\begin{aligned} \Delta \dot{\varphi} &= \frac{\partial \dot{\varphi}}{\partial h} \cdot \Delta h + \frac{\partial \dot{\varphi}}{\partial v_{eb,n}^n} \cdot \Delta v_{eb,n}^n \\ &= \frac{-v_{eb,n}^n}{(R_n + h)^2} \cdot \Delta h + \frac{1}{R_n + h} \cdot \Delta v_{eb,n}^n. \end{aligned} \quad (6.9)$$

Hiermit ergibt sich mit den Umrechnungen (6.2) und (6.4) für die Änderung des Fehlers in Nordrichtung, in Abhängigkeit von den geschätzten Fehlern:

$$\Delta \dot{x}_n = \frac{v_{eb,n}^n}{R_n + h} \cdot \Delta x_d + \Delta v_{eb,n}^n. \quad (6.10)$$

Für die Änderung des Fehlers in Ostrichtung ergibt sich durch Taylorreihenentwicklung von (6.7):

$$\begin{aligned}\Delta \dot{\lambda} &= \frac{\partial \dot{\lambda}}{\partial \varphi} \cdot \Delta \varphi + \frac{\partial \dot{\lambda}}{\partial h} \cdot \Delta h + \frac{\partial \dot{\lambda}}{\partial v_{eb,e}^n} \cdot \Delta v_{eb,e}^n \\ &= \frac{v_{eb,e}^n \cdot \sin(\varphi)}{(R_e + h) \cdot \cos^2(\varphi)} \cdot \frac{\Delta x_n}{R_n + h} + \frac{v_{eb,e}^n}{(R_e + h)^2 \cdot \cos(\varphi)} \cdot \Delta x_d \\ &\quad + \frac{1}{(R_e + h) \cdot \cos(\varphi)} \cdot \Delta v_{eb,e}^n.\end{aligned}\quad (6.11)$$

Mit der Umrechnung (6.3) ergibt sich für die Änderung des Fehlers in Ostrichtung, in Abhängigkeit von den geschätzten Fehlern:

$$\Delta \dot{x}_e = \frac{v_{eb,e}^n \tan(\varphi)}{R_n + h} \cdot \Delta x_n + \frac{v_{eb,e}^n}{R_e + h} \cdot \Delta x_d + \Delta v_{eb,e}^n. \quad (6.12)$$

Für die Änderung des Fehlers in der Höhe ergibt sich durch Taylorreihenentwicklung von (6.8) schließlich:

$$\Delta \dot{h} = \frac{\partial \dot{h}}{\partial v_{eb,d}^n} \Delta v_{eb,d}^n = -\Delta v_{eb,d}^n. \quad (6.13)$$

Mit der Umrechnung (6.4) ergibt sich für die Änderung des Fehlers in der Höhe, in Abhängigkeit von den geschätzten Fehlern:

$$\Delta \dot{x}_d = \Delta v_{eb,d}^n. \quad (6.14)$$

Geschwindigkeitsfehlerdifferentialgleichungen

Die Änderungen der Geschwindigkeitsfehler lassen sich aus der Gleichung (2.21) gewinnen, also

$$\dot{\vec{v}}_{eb}^n = \mathbf{C}_b^n \vec{a}_{ib}^b - (2\vec{\omega}_{ie}^n + \vec{\omega}_{en}^n) \times \vec{v}_{eb}^n + \vec{g}_l^n. \quad (6.15)$$

Dabei ist zu erkennen, dass die Änderung des Geschwindigkeitsfehlers im Wesentlichen von den Orientierungsfehlern, Geschwindigkeitsfehlern und Fehlern in den Beschleunigungswerten z. B. Beschleunigungsmesserbiasen beeinflusst wird. Als erstes ist es nötig die Zusammenhänge zu den im Zustandsvektor des Kalman-Filters enthaltenen Größen herzustellen.

Die Orientierung \mathbf{C}_b^n kann in Abhängigkeit der Orientierungsfehler und der geschätzten Orientierung durch

$$\mathbf{C}_b^n = (\mathbf{I} + \boldsymbol{\Psi}) \hat{\mathbf{C}}_b^n \quad (6.16)$$

mit Hilfe der Gleichung (6.5) dargestellt werden. $\boldsymbol{\Psi}$ ist dabei die kreuzproduktbildende Matrix der Orientierungsfehler. $\hat{\mathbf{C}}_b^n$ ist die geschätzte Orientierung.

Die gemessenen Beschleunigungen $\tilde{\vec{a}}_{ib}^b$ werden, analog zu Gleichung (4.13), als Summe aus wahrer Beschleunigung \vec{a}_{ib}^b , den Biasen \vec{b}_a und dem Rauschen \vec{n}_a dargestellt:

$$\tilde{\vec{a}}_{ib}^b = \vec{a}_{ib}^b + \vec{b}_a + \vec{n}_a \quad (6.17)$$

$$\Rightarrow \vec{a}_{ib}^b = \tilde{\vec{a}}_{ib}^b - \vec{b}_a - \vec{n}_a. \quad (6.18)$$

Hiermit ergibt sich für die Taylorreihenentwicklung nun die Ausgangsgleichung

$$\begin{aligned}\dot{\vec{v}}_{eb}^n &= (\mathbf{I} + \boldsymbol{\Psi}) \hat{\mathbf{C}}_b^n (\tilde{\vec{a}}_{ib}^b - \vec{b}_a - \vec{n}_a) - (2\vec{\omega}_{ie}^n + \vec{\omega}_{en}^n) \times \vec{v}_{eb}^n + \vec{g}_l^n \\ &\approx (\mathbf{I} + \boldsymbol{\Psi}) \hat{\mathbf{C}}_b^n (\tilde{\vec{a}}_{ib}^b - \vec{b}_a) - (2\vec{\omega}_{ie}^n + \vec{\omega}_{en}^n) \times \vec{v}_{eb}^n + \vec{g}_l^n - \hat{\mathbf{C}}_b^n \vec{n}_a.\end{aligned}\quad (6.19)$$

In einer ersten Vereinfachung wurde hier das Produkt aus Rauschen und kreuzproduktbildender Matrix des Orientierungsfehlers vernachlässigt. Es ist nun noch zu beachten, dass der Coriolis-Term von der Geschwindigkeit \vec{v}_{eb}^n abhängt. Mit Gleichung (2.19) und (2.20) sowie Auswerten des Kreuzproduktes ergibt sich:

$$-(2\vec{\omega}_{ie}^n + \vec{\omega}_{en}^n) \times \vec{v}_{eb}^n = \begin{pmatrix} \frac{v_{eb,n}^n v_{eb,d}^n}{R_n+h} - \left(2\Omega \sin \varphi + \frac{v_{eb,e}^n \tan \varphi}{R_e+h}\right) v_{eb,e}^n \\ \left(2\Omega \sin \varphi + \frac{v_{eb,e}^n \tan \varphi}{R_e+h}\right) v_{eb,n}^n + \left(2\Omega \cos \varphi + \frac{v_{eb,e}^n}{R_e+h}\right) v_{eb,d}^n \\ - \left(2\Omega \cos \varphi + \frac{v_{eb,e}^n}{R_e+h}\right) v_{eb,e}^n - \frac{v_{eb,n}^n v_{eb,d}^n}{R_n+h} \end{pmatrix}. \quad (6.20)$$

Die Abhängigkeiten von den Fehlern in φ und h werden nun vernachlässigt. Diese Vereinfachung ist zulässig, da kleine Änderungen in den Werten kaum einen Einfluss auf die Gleichung (6.20) haben. Man nimmt weiterhin \vec{g}_l^n sowie die Krümmungsradien der Erde als konstant an, da auch hier die Fehler nur sehr geringe Änderungen zur Folge haben. Es ergibt sich die Geschwindigkeitsfehlerdifferentialgleichung damit zu

$$\Delta \dot{\vec{v}}_{eb}^n = \frac{\partial \dot{\vec{v}}_{eb}^n}{\partial \vec{v}_{eb}^n} \Delta \vec{v}_{eb}^n + \frac{\partial \dot{\vec{v}}_{eb}^n}{\partial \vec{\psi}} \Delta \vec{\psi} + \frac{\partial \dot{\vec{v}}_{eb}^n}{\partial \vec{b}_a} \Delta \vec{b}_a. \quad (6.21)$$

Die Ableitung nach der Geschwindigkeit ergibt sich zu

$$\frac{\partial \dot{\vec{v}}_{eb}^n}{\partial \vec{v}_{eb}^n} = \begin{pmatrix} \frac{v_{eb,d}^n}{R_n+h} & -2\Omega \sin \varphi - 2\frac{v_{eb,e}^n \tan \varphi}{R_e+h} & \frac{v_{eb,n}^n}{R_n+h} \\ 2\Omega \sin \varphi + \frac{v_{eb,e}^n \tan \varphi}{R_e+h} & \frac{v_{eb,n}^n \tan \varphi}{R_e+h} + \frac{v_{eb,d}^n}{R_e+h} & 2\Omega \cos \varphi + \frac{v_{eb,e}^n}{R_e+h} \\ -2\frac{v_{eb,n}^n}{R_n+h} & -2\Omega \cos \varphi - 2\frac{v_{eb,e}^n}{R_e+h} & 0 \end{pmatrix}. \quad (6.22)$$

Die Ableitung nach den Orientierungsfehlern ergibt

$$\frac{\partial \dot{\vec{v}}_{eb}^n}{\partial \vec{\psi}} = -\frac{\partial}{\partial \vec{\psi}} \left(\left[\hat{\mathbf{C}}_b^n (\tilde{\vec{a}}_{ib}^b - \vec{b}_a) \times \right] \vec{\psi} \right) = - \left[\hat{\vec{a}}_{ib}^n \times \right]. \quad (6.23)$$

Dabei ist $\hat{\vec{a}}_{ib}^n = \hat{\mathbf{C}}_b^n (\tilde{\vec{a}}_{ib}^b - \vec{b}_a)$ die gemessene Beschleunigung des Körpers bezüglich des i-frame, in Koordinaten des n-frame, verringert um den geschätzten Bias der Beschleunigungssensoren.

Die Ableitung nach den Beschleunigungsmesserbiasen liefert

$$\frac{\partial \dot{\vec{v}}_{eb}^n}{\partial \vec{b}_a} = \frac{\partial}{\partial \vec{b}_a} \left((\mathbf{I} + \boldsymbol{\Psi}) \hat{\mathbf{C}}_b^n (-\vec{b}_a) \right) \approx -\hat{\mathbf{C}}_b^n. \quad (6.24)$$

Auch hier wurde das Produkt aus Bias und kreuzproduktbildender Matrix des Orientierungsfehlers vernachlässigt, da beide Terme als klein angenommen werden.

Orientierungsfehlerdifferentialgleichungen

Auch die Änderung der Orientierungsfehler muss modelliert werden, da z. B. ein nicht kompensierter Bias in den Drehratensensoren eine Orientierungsfehler verursacht. Auch die fehlerhafte Kompensation von Transport- und Erddrehrate führt zu einem Orientierungsfehler. Hierzu wird die Gleichung (6.16) nach der kreuzproduktbildenden Matrix des Orientierungsfehlers Ψ umgestellt und nach der Zeit abgeleitet. Umstellen führt zu:

$$\dot{\Psi} = -\hat{\mathbf{C}}_b^n \mathbf{C}_b^{nT} + \mathbf{I}. \quad (6.25)$$

Für die Ableitung von Ψ nach der Zeit ergibt sich nach zahlreichen Umformungen, Vereinfachungen und Näherungen nun in vektorieller Form:

$$\dot{\vec{\psi}} = -\vec{\omega}_{in}^n \times \vec{\psi} + \Delta\vec{\omega}_{in}^n - \hat{\mathbf{C}}_b^n \Delta\vec{\omega}_{ib}^b. \quad (6.26)$$

Dabei ist $\Delta\vec{\omega}_{in}^n$ der Fehler in der angenommenen Summe aus Transportrate und Erd-drehrate und $\Delta\vec{\omega}_{ib}^b$ der Fehler in den gemessenen Drehraten. Auf die genaue Herleitung sei hier verzichtet und stattdessen auf die Literatur ([WEN07] S. 199ff) verwiesen. Durch Taylorreihenentwicklung von Gleichung (6.26) ergibt sich nun:

$$\dot{\Delta\vec{\psi}} = -\vec{\omega}_{in}^n \times \Delta\vec{\psi} + \Delta(\Delta\vec{\omega}_{in}^n) - \hat{\mathbf{C}}_b^n \Delta(\Delta\vec{\omega}_{ib}^b). \quad (6.27)$$

Für das Sensorfehlermodell wird nun angenommen, dass sich die gemessene Drehrate $\tilde{\vec{\omega}}_{ib}^b$ analog zu Gleichung (4.13) aus der Summe von realer Drehrate $\vec{\omega}_{ib}^b$, dem Bias \vec{b}_ω und dem Rauschterm \vec{n}_ω zusammensetzt:

$$\tilde{\vec{\omega}}_{ib}^b = \vec{\omega}_{ib}^b + \vec{b}_\omega + \vec{n}_\omega. \quad (6.28)$$

Somit lässt sich $\Delta\vec{\omega}_{ib}^b$ mit

$$\Delta\vec{\omega}_{ib}^b = \tilde{\vec{\omega}}_{ib}^b - \vec{\omega}_{ib}^b = \vec{b}_\omega + \vec{n}_\omega \quad (6.29)$$

berechnen und mit

$$\frac{\partial \Delta\vec{\omega}_{ib}^b}{\partial \vec{b}_\omega} = 1 \cdot \Delta\vec{b}_\omega \quad (6.30)$$

ergibt sich schließlich

$$\Delta(\Delta\vec{\omega}_{ib}^b) = \Delta\vec{b}_\omega + \vec{n}_\omega. \quad (6.31)$$

$\Delta(\Delta\vec{\omega}_{in}^n)$ wird durch die Taylorreihenentwicklung der Summe aus Transportrate (2.20) und Erddrehrate (2.19) bestimmt:

$$\vec{\omega}_{in}^n = \vec{\omega}_{ie}^n + \vec{\omega}_{en}^n = \begin{pmatrix} \Omega \cos \varphi + \frac{v_{eb,e}^e}{R_e + h} \\ -\frac{v_{eb,n}^n}{R_n + h} \\ -\Omega \sin \varphi - \frac{v_{eb,e}^e \tan \varphi}{R_e + h} \end{pmatrix}. \quad (6.32)$$

Es ergibt sich mit der Annahme von konstanten Krümmungsradien und den entsprechenden Umrechnungen für die Positionen:

$$\Delta(\Delta\vec{\omega}_{in}^n) = \frac{\partial\vec{\omega}_{in}^n}{\partial\varphi, \lambda, h} \begin{pmatrix} \Delta\varphi \\ \Delta\lambda \\ \Delta h \end{pmatrix} + \frac{\partial\vec{\omega}_{in}^n}{\partial\vec{v}_{eb}^n} \Delta\vec{v}_{eb}^n \quad (6.33)$$

$$= \begin{pmatrix} -\frac{\Omega \sin \varphi}{R_n+h} & 0 & \frac{v_{eb,e}^n}{(R_e+h)^2} \\ 0 & 0 & \frac{v_{eb,n}^n}{(R_n+h)^2} \\ -\frac{\Omega \cos \varphi}{R_n+h} - \frac{v_{eb,e}^n}{(R_e+h)(R_n+h) \cos^2 \varphi} & 0 & \frac{v_{eb,e}^n \tan \varphi}{(R_e+h)^2} \end{pmatrix} \begin{pmatrix} \Delta x_n \\ \Delta x_e \\ \Delta x_d \end{pmatrix} \quad (6.34)$$

$$+ \begin{pmatrix} 0 & \frac{1}{R_e+h} & 0 \\ -\frac{1}{R_n+h} & 0 & 0 \\ 0 & -\frac{\tan \varphi}{R_e+h} & 0 \end{pmatrix} \Delta\vec{v}_{eb}^n$$

Inertialsensorfehlerdifferentialgleichungen

Zur Modellierung der Inertialsensorbiase wird ein Random-Walk Prozess nach Gleichung (4.14) gewählt:

$$\dot{\vec{b}}_a = \vec{n}_{ba} \quad , \quad \dot{\vec{b}}_\omega = \vec{n}_{b\omega}. \quad (6.35)$$

Somit folgt auch hier:

$$\Delta\dot{\vec{b}}_a = \vec{n}_{ba} \quad , \quad \Delta\dot{\vec{b}}_\omega = \vec{n}_{b\omega}. \quad (6.36)$$

Zusammenfassung der Gleichungen

Der Übersichtlichkeit halber sollen die Gleichungen und das Systemmodell hier noch einmal zusammengefasst werden. Als Untermatrizen des Systemmodells ergeben sich die folgenden Gleichungen:

$$\mathbf{F}_{11} = \begin{pmatrix} 0 & 0 & \frac{v_{eb,n}^n}{R_n+h} \\ \frac{v_{eb,e}^n \tan \varphi}{R_n+h} & 0 & \frac{v_{eb,e}^n}{R_e+h} \\ 0 & 0 & 0 \end{pmatrix} \quad (6.37)$$

$$\mathbf{F}_{22} = \begin{pmatrix} \frac{v_{eb,d}^n}{R_n+h} & -2\Omega \sin \varphi - 2\frac{v_{eb,e}^n \tan \varphi}{R_e+h} & \frac{v_{eb,n}^n}{R_n+h} \\ 2\Omega \sin \varphi + \frac{v_{eb,e}^n \tan \varphi}{R_e+h} & \frac{v_{eb,n}^n \tan \varphi}{R_e+h} + \frac{v_{eb,d}^n}{R_e+h} & 2\Omega \cos \varphi + \frac{v_{eb,e}^n}{R_e+h} \\ -2\frac{v_{eb,n}^n}{R_n+h} & -2\Omega \cos \varphi - 2\frac{v_{eb,e}^n}{R_e+h} & 0 \end{pmatrix} \quad (6.38)$$

$$\mathbf{F}_{23} = \begin{pmatrix} 0 & \hat{a}_{ib,d}^n & -\hat{a}_{ib,e}^n \\ -\hat{a}_{ib,d}^n & 0 & \hat{a}_{ib,n}^n \\ \hat{a}_{ib,e}^n & -\hat{a}_{ib,n}^n & 0 \end{pmatrix} = - \left[\hat{\vec{a}}_{ib}^n \times \right] \quad (6.39)$$

$$\mathbf{F}_{31} = \begin{pmatrix} -\frac{\Omega \sin \varphi}{R_n+h} & 0 & \frac{v_{eb,e}^n}{(R_e+h)^2} \\ 0 & 0 & \frac{v_{eb,n}^n}{(R_n+h)^2} \\ -\frac{\Omega \cos \varphi}{R_n+h} - \frac{v_{eb,e}^n}{(R_e+h)(R_n+h) \cos^2 \varphi} & 0 & \frac{v_{eb,e}^n \tan \varphi}{(R_e+h)^2} \end{pmatrix} \quad (6.40)$$

$$\mathbf{F}_{32} = \begin{pmatrix} 0 & \frac{1}{R_e+h} & 0 \\ -\frac{1}{R_n+h} & 0 & 0 \\ 0 & -\frac{\tan \varphi}{R_e+h} & 0 \end{pmatrix} \quad (6.41)$$

$$\mathbf{F}_{33} = \begin{pmatrix} 0 & -\Omega \sin \varphi - \frac{v_{eb,e}^n \tan \varphi}{R_e+h} & \frac{v_{eb,n}^n}{R_n+h} \\ \Omega \sin \varphi + \frac{v_{eb,e}^n \tan \varphi}{R_e+h} & 0 & \Omega \cos \varphi + \frac{v_{eb,e}^n \tan \varphi}{R_e+h} \\ -\frac{v_{eb,n}^n}{R_n+h} & -\Omega \cos \varphi - \frac{v_{eb,e}^n \tan \varphi}{R_e+h} & 0 \end{pmatrix} = -[\vec{\omega}_{in}^n \times] \quad (6.42)$$

Als Systemmodell ergibt sich mit den obigen Untermatrizen nun zu:

$$\begin{pmatrix} \Delta \vec{p} \\ \Delta \vec{v} \\ \Delta \vec{\psi} \\ \Delta \vec{b}_a \\ \Delta \vec{b}_\omega \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{11} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{22} & \mathbf{F}_{23} & -\hat{\mathbf{C}}_b^n & \mathbf{0} \\ \mathbf{F}_{31} & \mathbf{F}_{32} & \mathbf{F}_{33} & \mathbf{0} & -\hat{\mathbf{C}}_b^n \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Delta \vec{p} \\ \Delta \vec{v} \\ \Delta \vec{\psi} \\ \Delta \vec{b}_a \\ \Delta \vec{b}_\omega \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\hat{\mathbf{C}}_b^n & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{C}}_b^n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \vec{n}_a \\ \vec{n}_\omega \\ \vec{n}_{b_a} \\ \vec{n}_{b_\omega} \end{pmatrix} \quad (6.43)$$

6.2.2 Messmodelle

Um nun die gelieferten Messungen vom GPS-Empfänger verarbeiten zu können, sind entsprechende Messmodelle nötig. Die Zusammenhänge zwischen den vom GPS-Empfänger gelieferten Messwerten, dem zu schätzenden Systemzustand und der vom INS gelieferten Navigationslösung sind nichtlinear. Daher wird auch die im Messschritt benötigte Messmatrix durch Linearisierung gewonnen. In diesem Abschnitt sollen die Zusammenhänge zwischen den Messwerten und den Zuständen gezeigt werden, und die benötigte Messmatrix sowie die Messgleichungen sollen hergeleitet werden. Die Herleitung wird dabei im Zeitdiskreten betrachtet. Bei den betrachteten Werten handelt es sich daher um die jeweiligen Werte zum Zeitpunkt k . Der Index k wird zur Verbesserung der Übersichtlichkeit weg gelassen. Lediglich in den Gleichungen zur Bestimmung der zu verarbeitenden Differenzen wird der Index k zur Verdeutlichung der zeitdiskreten Messwerte hinzugefügt.

Positionsmessung

Als erstes werden Positionsmessungen verarbeitet. Der GPS-Empfänger liefert eine Messung für die Position der Antenne \vec{r}_A^n , welche nicht zwangsweise dieselbe sein muss wie die des INS \vec{r}_U^n . Der konstante Vektor zwischen GPS und INS sei in Koordinaten des b-frame bekannt und mit $\vec{l}^b = (l_x, l_y, l_z)^T$ bezeichnet. Hiermit und über den Zusammenhang (6.16) lässt sich die Antennenposition in Zusammenhang zu den geschätzten Werten darstellen:

$$\begin{aligned} \vec{r}_A^n &= \vec{r}_U^n + (\mathbf{I} + \boldsymbol{\Psi}) \hat{\mathbf{C}}_b^n \vec{l}^b \\ &= \vec{r}_U^n + (\mathbf{I} + \boldsymbol{\Psi}) \hat{\vec{l}}^n \\ &= \vec{r}_U^n + \hat{\vec{l}}^n - [\hat{\vec{l}}^n \times] \boldsymbol{\psi}. \end{aligned} \quad (6.44)$$

Die Messmatrix ergibt sich damit aus der Jacobi-Matrix der rechten Seite der Gleichung (6.44) zu

$$\mathbf{H}_{pos,k} = (\mathbf{H}_{p1} \ \mathbf{H}_{p2} \ \mathbf{H}_{p3} \ \mathbf{H}_{p4} \ \mathbf{H}_{p5}) \quad (6.45)$$

mit

$$\mathbf{H}_{p1} = \frac{\partial \vec{r}_A^n}{\partial \vec{r}_U^n} = \mathbf{I} \quad (6.46)$$

$$\mathbf{H}_{p2} = \frac{\partial \vec{r}_A^n}{\partial \vec{v}_{eb}^n} = \mathbf{H}_{p4} = \frac{\partial \vec{r}_A^n}{\partial \vec{b}_a} = \mathbf{H}_{p5} = \frac{\partial \vec{r}_A^n}{\partial \vec{b}_\omega} = \mathbf{0} \quad (6.47)$$

$$\mathbf{H}_{p3} = \frac{\partial \vec{r}_A^n}{\partial \vec{\psi}} = - [\hat{\vec{l}}^n \times] \quad (6.48)$$

Das Error State Space Kalman-Filter verarbeitet als Messwerte $\Delta \vec{y}_k$ die Differenzen zwischen erwartetem Messwert $\hat{\vec{y}}_k$, der vom INS geliefert wird, und der eigentlichen Messung $\tilde{\vec{y}}_k$. Für die Positionen ist dies in diesem Fall die Differenz zwischen der vom INS geschätzten Position, bzw. der geschätzten Position der Antenne, und der vom GPS-Empfänger gemessenen Position. Die Differenz wird dabei in [m] verarbeitet. $\Delta \vec{y}_{pos,k}$ ist also gegeben durch:

$$\Delta \vec{y}_{pos,k} = \begin{pmatrix} (R_n + \hat{h}) \cdot (\tilde{\varphi} - \hat{\varphi}) \\ (R_e + \hat{h}) \cos \hat{\varphi} \cdot (\tilde{\lambda} - \hat{\lambda}) \\ -(\tilde{h} - \hat{h}) \end{pmatrix}_k. \quad (6.49)$$

Geschwindigkeitsmessung

Neben den Positionsmessungen werden auch Geschwindigkeitsmessungen verarbeitet. Durch die möglicherweise unterschiedlichen Positionen von INS und GPS können sich auch die Geschwindigkeiten von INS \vec{v}_{eb}^n und GPS \vec{v}_{eA}^n unterscheiden. Eine Gleichung die den Zusammenhang zwischen den beiden Geschwindigkeiten und den geschätzten Zuständen näherungsweise beschreibt ist durch

$$\vec{v}_{eA}^n \approx \vec{v}_{eb}^n + (\mathbf{I} + \boldsymbol{\Psi}) \hat{\mathbf{C}}_b^n \left((\tilde{\vec{\omega}}_{eb}^b - \vec{b}_\omega - \vec{n}_\omega) \times \vec{l}^b \right) \quad (6.50)$$

gegeben ([WEN07] S. 209). In den Drehraten kann die mitgemessene Erddrehrate vernachlässigt werden, da der Einfluss im Vergleich zur Drehrate des Körpers gering ist. Auch das Rauschen in den Drehraten kann in der Regel vernachlässigt werden, und es wird $\hat{\vec{\omega}}_{ib}^b = \tilde{\vec{\omega}}_{ib}^b - \vec{b}_\omega$ gesetzt. Außerdem wird $\hat{\Omega}_{ib}^b = [\hat{\vec{\omega}}_{ib}^b \times]$ gesetzt. Damit ergibt sich:

$$\begin{aligned} \vec{v}_{eA}^n &\approx \vec{v}_{eb}^n + \hat{\mathbf{C}}_b^n \tilde{\vec{\omega}}_{ib}^b \times \vec{l}^b + \hat{\mathbf{C}}_b^n [\vec{l}^b \times] \vec{b}_\omega - [\hat{\mathbf{C}}_b^n \hat{\Omega}_{ib}^b \vec{l}^b \times] \vec{\psi} \\ &= \vec{v}_{eb}^n + \hat{\mathbf{C}}_b^n \tilde{\vec{\omega}}_{ib}^b \times \vec{l}^b + \hat{\mathbf{C}}_b^n [\vec{l}^b \times] \vec{b}_\omega - [\vec{v}_{rot}^n \times] \vec{\psi}. \end{aligned} \quad (6.51)$$

Auch hier ist die für die Verarbeitung der Geschwindigkeitsmessungen nötige Messmatrix durch die Jacobi-Matrix der rechten Seite von Gleichung (6.51) gegeben:

$$\mathbf{H}_{velo,k} = (\mathbf{H}_{v1} \ \mathbf{H}_{v2} \ \mathbf{H}_{v3} \ \mathbf{H}_{v4} \ \mathbf{H}_{v5}) \quad (6.52)$$

mit

$$\mathbf{H}_{v1} = \frac{\partial \vec{v}_{eA}^n}{\partial \vec{r}_U^n} = \mathbf{H}_{v4} = \frac{\partial \vec{v}_{eA}^n}{\partial \vec{b}_a} = \mathbf{0} \quad (6.53)$$

$$\mathbf{H}_{v2} = \frac{\partial \vec{v}_{eA}^n}{\partial \vec{v}_{eb}^n} = \mathbf{I} \quad (6.54)$$

$$\mathbf{H}_{v3} = \frac{\partial \vec{v}_{eA}^n}{\partial \vec{\psi}} = -[\vec{v}_{rot}^n \times] \quad (6.55)$$

$$\mathbf{H}_{v5} = \frac{\partial \vec{v}_{eA}^n}{\partial \vec{b}_\omega} = \hat{\mathbf{C}}_b^n [\vec{l}^b \times]. \quad (6.56)$$

Als Messwert wird die Differenz zwischen der mit Hilfe des INS bestimmten Geschwindigkeit der Antenne $\hat{\vec{v}}_{eA,k}^n$ und der vom GPS gemessenen Geschwindigkeit $\tilde{\vec{v}}_{eA,k}^n$ verarbeitet:

$$\Delta \vec{v}_{eA,k}^n = \tilde{\vec{v}}_{eA,k}^n - \hat{\vec{v}}_{eb,k}^n - \hat{\mathbf{C}}_{b,k}^n (\hat{\vec{\omega}}_{ib,k}^b \times \vec{l}^b). \quad (6.57)$$

Magnetfeldmesswerte

Ein Problem des GPS/INS-Filter Entwurfs in seiner jetzigen Form ist, dass nur eine schwache Stützung des Headings, also der Ausrichtung in Nordrichtung, gegeben ist. Zur Bestimmung der Neigung nach Vorne sowie zur Seite hat die Gravitation einen großen Einfluss. Dabei ist die Messung unabhängig vom Heading. Auf demselben Prinzip basiert eine Wasserwaage. Zur Stützung des Headings sind jedoch ausreichende Beschleunigungen nach vorne/Norden oder zur Seite/Osten nötig, welche aber nicht immer gegeben sind. Neben den Positions- und Geschwindigkeitsmessungen ist es aber auch möglich, Messungen des Erdmagnetfeldes zur Orientierungsbestimmung zu nutzen. Auf demselben Prinzip basiert ein Kompass. Für den GPS/INS-Filter soll hier nun der Zusammenhang zwischen dem gemessenen Erdmagnetfeld, der Orientierung und dem Orientierungsfehler in Nordrichtung, dem Heading, bestimmt werden. Mit Hilfe der Magnetfeldmessungen wäre auch eine komplette Orientierungsberechnung möglich. Aufgrund möglicher Beeinflussungen des Magnetfeldes durch äußere Einflüsse wird die Stützung aber nur auf das Heading beschränkt. Eine genaue Betrachtung des Problems, sowie der folgenden Herleitung kann ([WEN07] S. 283ff) entnommen werden. Eine Beeinflussung des Magnetfeldes durch das Fahrzeug konnte nicht festgestellt werden bzw. lässt sich durch die Kalibrierung der Magnetfeldsensoren beheben.

Alternativ zur Nutzung der Magnetfeldmessungen wäre es auch möglich, aus den gemessenen Geschwindigkeiten das Heading zu bestimmen, wie es ein GPS-Empfänger tut. Der Nachteil hierbei ist aber, dass die Ausrichtung des INS zum Objekt bekannt sein muss und sich nicht ändern darf. Außerdem darf sich das Objekt nur entlang einer vorgegebenen Achse bewegen, wie dies z. B. bei einem Auto der Fall ist. Zu Beachten ist aber auch hier, dass sich das berechnete Heading bei Vorwärts- und Rückwärtsfahrt bereits unterscheiden würden. Bei einem Menschen wären die Bedingen nicht erfüllt, da er sich in alle Richtungen frei bewegen kann.

Der Zusammenhang zwischen dem Magnetfeld \vec{h}^b im b-frame und dem Erdmagnetfeld \vec{h}^n im n-frame, in Abhängigkeit von den Systemzuständen und der geschätzten Orientierung, ist gegeben durch:

$$\begin{aligned}\vec{h}^b &= \hat{\mathbf{C}}_b^{n,T}(\mathbf{I} - \boldsymbol{\Psi})\vec{h}^n \\ &= \hat{\mathbf{C}}_b^{n,T}\vec{h}^n + \hat{\mathbf{C}}_b^{n,T}[\vec{h}^n \times] \vec{\psi}.\end{aligned}\quad (6.58)$$

Eine Messgleichung ist damit durch

$$\vec{h}^b - \hat{\mathbf{C}}_b^{n,T}\vec{h}^n = \hat{\mathbf{C}}_b^{n,T}[\vec{h}^n \times] \vec{\psi} \quad (6.59)$$

gegeben. Da hier allerdings nur der Fehler im Heading bestimmt werden soll, muss $\vec{\psi} = (0, 0, \gamma)^T$ gesetzt werden. Daraus folgt dann die Messgleichung

$$\Delta \vec{y}_{mag,k} = \vec{h}_k^b - \hat{\mathbf{C}}_{b,k}^{n,T}\vec{h}^n = \hat{\mathbf{C}}_b^{n,T} \begin{pmatrix} h_e \\ -h_n \\ 0 \end{pmatrix} \gamma. \quad (6.60)$$

Die Messmatrix ist wiederum durch die Jacobi-Matrix der rechten Seite der Gleichung (6.60) gegeben und ergibt sich zu:

$$\mathbf{H}_{h,k} = (\mathbf{H}_{h1} \ \mathbf{H}_{h2} \ \mathbf{H}_{h3} \ \mathbf{H}_{h4} \ \mathbf{H}_{h5}) \quad (6.61)$$

mit

$$\mathbf{H}_{h1} = \frac{\partial \vec{h}^b}{\partial \vec{r}_U} = \mathbf{H}_{h2} = \frac{\partial \vec{h}^b}{\partial \vec{v}_{eb}^n} = \mathbf{H}_{h4} = \frac{\partial \vec{h}^b}{\partial \vec{b}_a} = \mathbf{H}_{h5} = \frac{\partial \vec{h}^b}{\partial \vec{b}_\omega} = \mathbf{0} \quad (6.62)$$

$$\mathbf{H}_{h3} = \frac{\partial \vec{h}^b}{\partial \vec{\psi}} = \begin{bmatrix} 0 & 0 & \hat{\mathbf{C}}_b^{n,T} \begin{pmatrix} h_e \\ -h_n \\ 0 \end{pmatrix} \\ 0 & 0 & \end{bmatrix} \quad (6.63)$$

Zusammenfassung der Gleichungen

Wie bei dem Systemmodell sollen auch hier die Gleichungen noch einmal kurz zusammengefasst werden. Für die Fehler des INS ergibt sich mit den obigen Gleichungen nun die folgende Messgleichung:

$$\Delta \vec{y}_k = \tilde{\vec{y}}_k - \hat{\vec{y}}_k = \begin{pmatrix} \Delta \vec{y}_{pos,k} \\ \Delta \vec{v}_{eA,k} \\ \Delta \vec{y}_{mag,k} \end{pmatrix} + \begin{pmatrix} \vec{n}_{pos,k} \\ \vec{n}_{vel,k} \\ \vec{n}_{mag,k} \end{pmatrix}, \quad (6.64)$$

wobei $\vec{n}_{pos,k}$ und $\vec{n}_{vel,k}$ jeweils das Rauschen der GPS Messungen für Position und Geschwindigkeit wie in Abschnitt 4.2.1 dargestellt, sind. $\vec{n}_{mag,k}$ ist das Rauschen des Magnetometers, wie in Abschnitt 4.2.2 dargestellt.

Insgesamt ergibt sich damit nun folgendes linearisiertes Messmodell:

$$\begin{pmatrix} \Delta \vec{y}_{pos,k} \\ \Delta \vec{v}_{eA,k} \\ \Delta \vec{y}_{mag,k} \end{pmatrix} = \begin{pmatrix} \mathbf{H}_{pos,k} \\ \mathbf{H}_{velo,k} \\ \mathbf{H}_{h,k} \end{pmatrix} \begin{pmatrix} \Delta \vec{p} \\ \Delta \vec{v} \\ \Delta \vec{\psi} \\ \Delta \vec{b}_a \\ \Delta \vec{b}_\omega \end{pmatrix}_k + \begin{pmatrix} \vec{n}_{pos,k} \\ \vec{n}_{eA,k} \\ \vec{n}_{mag,k} \end{pmatrix}. \quad (6.65)$$

Je nach Verfügbarkeit der Messwerte ist es hier außerdem möglich, das Messmodell durch Hinzufügen oder Weglassen der jeweiligen Zeilen dynamisch anzupassen. So wäre es z. B. denkbar, nur die Messewerte des Magnetometers zu verwenden, auch wenn keine GPS-Messwerte zur Verfügung stehen oder eine Höhenmessung mit einer höheren Datenrate zu verwenden als Positionsinformationen in der Ebene vorhanden sind.

6.2.3 Korrektur der totalen Größen

Sind, in einem Zeitschritt k , nach einem Messschritt die Fehler der Navigationslösung des INS durch das Kalman-Filter bestimmt worden, ist eine Korrektur der INS-Werte möglich. Die Fehler sollten nun anhand der im Zustandsvektor des Kalman-Filters gegebenen Werte korrigiert werden. \hat{x}^- bezeichnet dabei die Schätzungen des INS vor der Korrektur und \hat{x}^+ die Schätzungen nach der Korrektur. Wie bei den Messwerten handelt es sich auch hier um zeitdiskrete Werte. Auch hier wird zur Verbesserung der Übersichtlichkeit der Index k weggelassen.

Die Korrektur der Positionen ist gegeben durch die Gleichungen

$$\hat{\varphi}^+ = \frac{\Delta x_n}{R_n + \hat{h}^-} + \hat{\varphi}^- \quad (6.66)$$

$$\hat{\lambda}^+ = \frac{\Delta x_e}{(R_e + \hat{h}^-) \cos \hat{\varphi}^-} + \hat{\lambda}^- \quad (6.67)$$

$$\hat{h}^+ = \hat{h}^- - \Delta x_d. \quad (6.68)$$

Außerdem erfolgt die Korrektur der NED Positionen mit

$$\hat{\vec{p}}^{n,+} = \Delta \vec{p} + \hat{\vec{p}}^{n,-}. \quad (6.69)$$

Die Korrektur der Geschwindigkeit erfolgt mit

$$\hat{\vec{v}}_{eb}^{n,+} = \Delta \vec{v}_{eb}^n + \hat{\vec{v}}_{eb}^{n,-}. \quad (6.70)$$

Zur Korrektur der Orientierung wird aus den Orientierungsfehlern $\Delta \vec{\psi}$, wie in Gleichung (2.10), ein Korrekturquaternion \mathbf{q}_c berechnet und per Quaternionenmultiplikation durchgeführt:

$$\mathbf{q}_c = \begin{pmatrix} \cos(\frac{\Delta\psi}{2}) \\ \frac{\Delta\vec{\psi}}{\Delta\psi} \sin(\frac{\Delta\psi}{2}) \end{pmatrix} \quad (6.71)$$

$$\hat{\mathbf{q}}_b^{n,+} = \mathbf{q}_c \bullet \hat{\mathbf{q}}_b^{n,-}. \quad (6.72)$$

Die Korrektur der Inertialsensorbiase erfolgt mit den Gleichungen

$$\hat{\vec{b}}_a^+ = \Delta \vec{b}_a + \hat{\vec{b}}_a^- \quad (6.73)$$

$$\hat{\vec{b}}_\omega^+ = \Delta \vec{b}_\omega + \hat{\vec{b}}_\omega^- \quad (6.74)$$

Nach der Korrektur der totalen Größen mit Hilfe der geschätzten Fehler müssen auch die geschätzten Fehler, also der Zustandsvektor, wieder auf Null gesetzt werden:

$$\Delta \vec{x} = \vec{0} \quad (6.75)$$

6.3 Implementierung

In diesem Abschnitt soll nun die Implementierung der Filterung allgemein und im speziellen in Matlab beschrieben werden. Neben der Durchführung der Filterung sind unter Anderem natürlich auch die Berechnung der Navigationslösung des INS sowie die Korrektur der Werte nötig. Der Aufbau eines kompletten Systems und der Ablauf der Prozesse sollen im Folgenden grob skizziert werden:

1. Im ersten Schritt sind die initialen Werte für das INS zu setzen. Hierzu gehören unter anderem die Position, Geschwindigkeit und Orientierung. Bereits hier kann eine erste Messwertaufnahme zur Initialisierung nötig sein oder es werden die Werte aus dem vorherigen Zeitschritt verwendet.
2. Anschließend ist es nötig, die Messwerte für das INS und für die Filterung aufzunehmen und in die entsprechenden Größen umzuwandeln.
3. Im nächsten Schritt müssen die Rauschwerte für das Systemmodell sowie für das Messmodell bestimmt und gesetzt werden. Dies kann sowohl statisch, über fest eingestellte Werte, als auch dynamisch, durch eine Schätzung der Werte, geschehen. Auch eine Initialisierung der Kovarianzmatrix des Kalman-Filters ist möglich.
4. Ist die Initialisierung abgeschlossen, kann mit der eigentlichen Berechnung der Navigationslösung und Filterung begonnen werden. Dazu werden zuerst die Messwerte der Beschleunigungssensoren und Drehratensensoren um einen evtl. geschätzten Bias korrigiert.
5. Anschließend kann die Navigationslösung des INS aus den initialen oder den Werten aus dem vorherigen Zeitschritt mit Hilfe des Strapdown-Algorithmus berechnet werden.
6. Ist nun die Lösung des INS bekannt und stehen Messwerte zur Fehlerfilterung zu Verfügung, kann das Kalman-Filter zur Bestimmung der Fehler aufgerufen werden.
7. Nach dem Durchlauf der Filterung sollten die Fehler bekannt sein und eine Korrektur durchgeführt werden, außerdem ist der Filterzustand nach einer Korrektur auf Null zu setzen.
8. Sind keine Messwerte für eine Fehlerfilterung vorhanden, so wird nur der Propagationsschritt des Kalman-Filters aufgerufen.
9. Mit der Fehlerkorrektur der Navigationslösung des INS und/oder der Propagation der Kovarianzmatrix des Systemzustandes ist die Berechnung abgeschlossen und es kann mit dem nächsten Zeitschritt wieder von vorne begonnen werden.

Die Abbildung 6.1 zeigt ein Blockschaltbild des Systems.

Eine Demonstration der Berechnung und Fehlerfilterung für die in Kapitel 5 erzeugten Daten ist im Matlab-Skript “gps_ins_demo.m” realisiert. Dabei werden die Rauschwerte

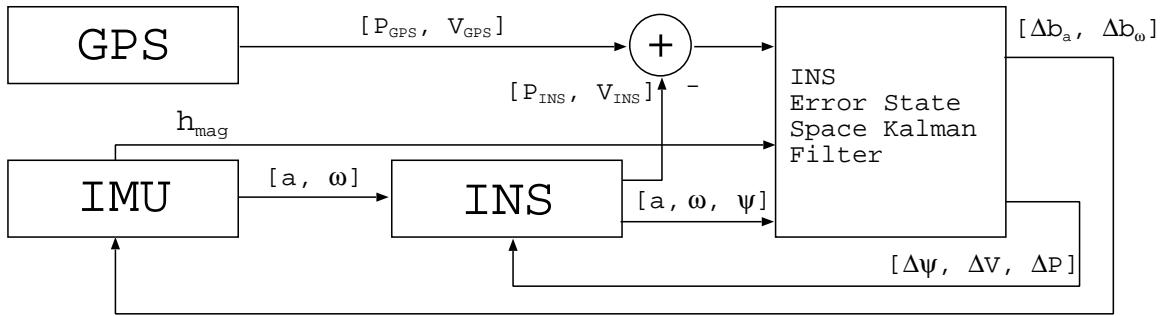


Abbildung 6.1: GPS/INS-Navigationsfilter

jeweils konstant gesetzt. Außerdem geschieht die “Messwertaufnahme” und Konvertierung für den kompletten Datensatz vor der Berechnung. Hierbei handelt es sich also um eine Offline-verarbeitung, bei der ein aufgezeichneter Datensatz nach der Datenaufzeichnung bzw. Datenerzeugung komplett verarbeitet wird. Die eigentliche Berechnung und Filterung geschieht im Matlab-Skript “gps_ins.m”

Listing 6.1: gps_ins_demo.m

```

1 %% Initialization specific to INS Toolbox %%
2 C = [0 1 0; 1 0 0; 0 0 -1]; % Conversion between NED and ENU
3 msprh2msprs = 1/sqrt(3600); % meter per second per root hour to
4 % meter per second per root second
5 dprh2rprs = (pi/180)/sqrt(3600); % degree per root hour to radians per root second
6
7 % get initial attitude in NED system relating body to nav system
8 DCMnb = [DCMnb_prof(1,1:3); DCMnb_prof(1,4:6); DCMnb_prof(1,7:9)];
9 DCMbn = DCMnb';
10
11 % get initial quaterinon in NED system
12 q0_b_n = dcm2qua(DCMbn);
13
14 % initial position an velocity in NED and LLH system
15 x0 = (C*pos_prof_L(1,:))';
16 v0 = (C*vel_prof_L(1,:))';
17 llh0 = llh_gps(1,:);
18
19 % accelerometer and gyroscope readings in body system
20 a_ib_b = est_dv/dt;
21 omega_ib_b = est_dtheta/dt;
22
23 % convert velocity readings from ENU to NED
24 vel_gps_f = (C*estfilt_vel_gps(2:end,:))';
25 llh_gps_f = est_llh_gps(2:end,:);
26
27 % measurement noise given by GPS Kalman Filter and magnetometer during data generation
28 Rins = blkdiag(3^2, 3^2, 10^2, 0.5^2, 0.5^2, 2^2, magxstdev^2, magystdev^2, magzstdev^2);
29 % cross correlations in GPS positions and velocities
30 Rins(1,4) = 0.15^2;
31 Rins(2,5) = 0.15^2;
32 Rins(3,6) = 4^2;
33 Rins(4,1) = 0.15^2;
34 Rins(5,2) = 0.15^2;
35 Rins(6,3) = 4^2;
36
37 % process noise
38 std_a = 0.03*msprh2msprs;
39 std_omega = 0.9*dprh2rprs;
40 std_bias_a = 0.02*msprh2msprs;
41 std_bias_omega = 1*dprh2rprs;

```

```

42 Q = blkdiag(std_a^2, std_a^2, std_a^2, ...
43             std_omega^2, std_omega^2, std_omega^2, ...
44             std_bias_a^2, std_bias_a^2, std_bias_a^2, ...
45             std_bias_omega^2, std_bias_omega^2, std_bias_omega^2)/dt;
46 %%% Initialization (END) %%%
47
48 %% call Strapdown + GPS/INS filtering
49 [q_sg, v_sg, x_sg, llh_sg, omega_ib_b_sg, a_ib_b_sg, stg, cvg] = ...
50 gps_ins(q0_b_n, x0, v0, llh0, dt, omega_ib_b, a_ib_b, h_b, llh_gps_f, vel_gps_f, ...
51 Rins, Q);

```

Das Matlab-Skript “gps_ins.m” führt dabei die Berechnung, Filterung und Korrektur für den kompletten Datenblock durch und gibt die berechneten und korrigierten Werte für Orientierung q_{sg} , Geschwindigkeit v_{sg} , Position in ENU x_{sg} und LLH llh_{sg} , Drehraten $\omega_{ib_b_\text{sg}}$ und Beschleunigungen $a_{ib_b_\text{sg}}$ zurück. Außerdem werden die einzelnen Zustände stg und Kovarianzen cvg des Kalman-Filters für jeden Zeitschritt ausgegeben. Für die Berechnung der Navigationslösung des INS wird die Matlab-Funktion “strapdown.m” genutzt. Das eigentliche Kalman-Filter ist in dem Matlab-Skript “gps_ins_filter.m” implementiert. Eine Filterung wird hierbei nur dann durchgeführt, wenn alle Messwerte zur Verfügung stehen.

Listing 6.2: gps_ins.m

```

1 function [q_s, v_s, x_s, llh_s, omega_ib_b, a_ib_b, st, cv] = ...
2     gps_ins(q0_b_n, x0, v0, llh0, dt, omega_ib_b, a_ib_b, h_b, llh_gps, vel_gps, R, Q)
3 %GPS_INS do combined strapdown calculation and INS/GPS Kalman filtering
4 %
5 %     function [q_s, v_s, x_s, llh_s, omega_ib_b, a_ib_b, st, cv] = ...
6 %         gps_ins(q0_b_n, x0, v0, llh0, dt, omega_ib_b, a_ib_b, h_b, llh_gps, vel_gps, R, Q)
7 %
8 % INPUTS
9 %     q0_b_n = inertial quaterinon
10 %    x0 = initial position in NED coordinates
11 %    v0 = initial velocity in NED coordinates
12 %    llh0 = initial llh coordinates
13 %    dt = time step
14 %    omega_ib_b = gyroscope readings in body coordinates
15 %    a_ib_b = accelerometer readings in body coordinates
16 %    h_b = magnetometer readings in body coordinates
17 %    llh_gps = GPS llh measurements/estimates
18 %    vel_gps = GPS velocity measurements/estimates in NED coordinates
19 %    R = measurement noise of GPS values: diag([std_pos.^2, std_vel.^2])
20 %    Q = measurement + process noise of INS values:
21 %        diag([std_a.^2, std_omega.^2, std_ba.^2, std_bomega.^2])
22 %
23 % OUTPUTS
24 %     q_s = new corrected quaternions
25 %     v_s = new corrected velocity
26 %     x_s = new corrected NED position
27 %     llh_s = new llhposition
28 %     omega_ib_b = est_omega_ib_b - bias_omega (corrected omega)
29 %     a_ib_b = est_a_ib_b - bias_a (corrected a)
30 %     st = kalman filter states
31 %     cv = kalman filter covariances
32
33 % number of values
34 npts = size(a_ib_b,1);
35
36 % initialization to zero
37 state = zeros(15,1);
38 bias_omega_ib_b = zeros(1,3);
39 bias_a_ib_b = zeros(1,3);
40
41 % initialization of covariance

```

```

42 covariance = zeros(15);
43 covariance(1:6,1:6) = R(1:6,1:6);
44 covariance(7:9,7:9) = blkdiag(0.5,0.5,0.5);
45 covariance(10:15,10:15) = Q(7:12,7:12)*100;
46
47
48 for i = 1:npts
49
50     %% Strapdown calculation %%
51     % bias correction
52     a_ib_b(i,:) = a_ib_b(i,:) - bias_a_ib_b;
53     omega_ib_b(i,:) = omega_ib_b(i,:) - bias_omega_ib_b;
54
55     % strapdown algorithm
56     [q_s(i,:), v_s(i,:), x_s(i,:)] = strapdown(a_ib_b(i,:), omega_ib_b(i,:), ...
57                                                 dt, q0_b_n, v0, x0, llh0);
58 %% (end) Strapdown calculation %%
59
60     %% GPS/INS filter calculation and correction %%
61     % transform acceleration from body to navigation frame
62     C_b_n = qua2dcm(q_s(i,:));
63     a_ib_n = C_b_n * a_ib_b(i,:)';
64
65     % estimated values
66     estimates = [llh_s(i,:)' omega_ib_b(i,:)' v_s(i,:)' a_ib_n C_b_n];
67
68     % measured values for gps + magnetometer
69     measurements = [llh_gps(i,:)' vel_gps(i,:)' h_b(i,:)'];
70
71     % do update if gps measurement is available
72     % else only do covariance propagation
73     if ~isnan(llh_gps(i,1))
74
75         [state, covariance] = gps_ins_filter(state, covariance, estimates, ...
76                                               measurements, dt, 1, R, Q);
77
78         % correction
79         esskf_delta_pos_n = state(1);
80         esskf_delta_pos_e = state(2);
81         esskf_delta_pos_d = state(3);
82         esskf_delta_vel_N = state(4:6)';
83         esskf_delta_Cbn_N = state(7:9)';
84         esskf_delta_bias_a_ib_b = state(10:12)';
85         esskf_delta_bias_omega_ib_b = state(13:15)';
86
87         % attitude correction
88         q_s(i,:) = quaupdat2(q_s(i,:),esskf_delta_Cbn_N);
89
90         % velocity correction
91         v_s(i,:) = v_s(i,:) + esskf_delta_vel_N;
92
93         % gyroscope and accelerometer bias update
94         bias_omega_ib_b = bias_omega_ib_b + esskf_delta_bias_omega_ib_b;
95         bias_a_ib_b = bias_a_ib_b + esskf_delta_bias_a_ib_b;
96
97         % xyz position correction
98         x_s(i,:) = x_s(i,:) + [esskf_delta_pos_n esskf_delta_pos_e esskf_delta_pos_d];
99
100        % llh position correction
101        % curvature of earth surface in north and east direction
102        [Rn Re] = radicurv(llh_s(i,1));
103        lat_cor = llh_s(i,1) + esskf_delta_pos_n/(Rn + llh_s(i,3));
104        lon_cor = llh_s(i,2) + esskf_delta_pos_e/((Re + llh_s(i,3))*cos(lat_cor));
105        height_cor = llh_s(i,3) - esskf_delta_pos_d;
106        llh_s(i,:) = [lat_cor lon_cor height_cor];
107
108        % save and reset state after correction
109        st(:,i) = state;
110        state = zeros(15,1);
111
112        % propagate/modify covariance matrix only

```

```

112     else
113         [state, covariance] = gps_ins_filter(state, covariance, estimates, ...
114                                         measurements, dt, 0, R, Q);
115     end
116 %% (end) GPS/INS filter calculation and correction %%%
117
118 % copy previous values
119 q0_b_n = q_s(i,:);
120 v0 = v_s(i,:);
121 x0 = x_s(i,:);
122 llh0 = llh_s(i,:);
123 cv(:,:,i) = covariance;
124 end

```

In dem Matlab-Skript “gps_ins_filter.m” sind die Gleichungen wie weiter oben in diesem Kapitel hergeleitet, implementiert. Dabei werden bei dem Aufruf des Filters jeweils immer alle Messgleichungen ausgewertet. Eine Fallunterscheidung findet hier noch nicht statt. Außerdem ist zu beachten dass die Herleitung des Systemmodells im Zeitkontinuierlichen durchgeführt wurde. Für die Implementierung des Systemmodells ist es aber nötig zum Zeitdiskreten überzugehen. Dies wird hier in Zeile 93 und 94, nach ([WEN07] S. 141f), unter Zuhilfenahme der Matrixexponentialfunktion, gemacht.

Listing 6.3: gps_ins_filter.m

```

1 function [state2, covariance2] = gps_ins_filter(state1, covariance1, estimates, ...
2                                         measurements, tdint, update, R, Q)
3 %GPS_INS_FILTER GSP/INS based Kalman filter to combine GPS and INS readings
4 %
5 % function [state2, covariance2] = gps_ins_filter(state1, covariance1, estimates, ...
6 %                                         measurements, tdint, update, R, Q)
7 %
8 % INPUTS
9 % state1 = last state: [dp dv dphi db_a db_omega]'
10 % covariance1 = last covariance matrix
11 % estimates = estimated values: [llh omega_ib v_eb a_ib C_bn]
12 % measurements = measured values: [llh v_eA]
13 % tdint = timestep
14 % update = do state update/estimation
15 %           0 = do covariance propagation only
16 %           1 = do state update/estimation
17 % R = measurement noise matrix: diag([std_pos.^2, std_vel.^2])
18 % Q = processnoisematrix: diag([std_a.^2, std_omega.^2, std_ba.^2, std_bomega.^2])
19 %
20 % OUTPUTS
21 % state2 = new state: [dp dv dphi db_a db_omega]'
22 % covariance2 = new covariance matrix
23
24 %% Initialization
25 % some constants for the moment
26 global Omega;
27 KF_update = 2; % KF update type
28 l_b = [0 0 0]'; % GPS antenna position in body coordinates
29 h_n = [19148.7 418 45077.9]'; % magnetic field as specified by data_generation
30 Omega = 7.292115e-05; % Earth rate
31
32 % measurements/estimates of GPS receiver
33 est_llh_gps = measurements(1:3,1);
34 est_v_eA = measurements(1:3,2);
35
36 % magnetic field measurements/estimates
37 est_h_b = measurements(1:3,3);
38
39 % measurements by INS and estimates by strapdown algorithm
40 est_llh = estimates(1:3,1);
41 est_omega_ib = estimates(1:3,2);

```

```

42 est_v_eb = estimates(1:3,3);
43 est_a_ib = estimates(1:3,4);
44 est_C_bn = estimates(1:3,5:7);
45
46 %% do filtering
47 % prediction
48 [state2, covariance2] = predict_EKF(state1, covariance1, Q, est_C_bn, ...
49                                     est_a_ib, est_llh, est_v_eb, tdint);
50
51 % update if specified
52 if update==1
53     [state2, covariance2] = update_EKF(state2, covariance2, R, KF_update, ...
54                                         est_llh_gps, est_llh, l_b, est_omega_ib, ...
55                                         est_C_bn, est_v_eb, est_v_eA, est_h_b, h_n);
56 end
57 end
58
59 %% Prediction
60 function [x, P] = predict_EKF(x, P, Q, est_C_bn, est_a_ib, est_llh, est_v_eb, tdint)
61     global Omega;
62     size_x = size(x,1);
63     lat = est_llh(1);
64     h = est_llh(3);
65     s_phi = sin(lat);
66     c_phi = cos(lat);
67     t_phi = tan(lat);
68     [R_n R_e] = radicurv(lat); % curvature of earth surface in north and east direction
69     R_nh = R_n+h;
70     R_eh = R_e+h;
71     v_ebn = est_v_eb(1);
72     v_ebe = est_v_eb(2);
73     v_ebd = est_v_eb(3);
74
75     I = eye(3);
76     Z = zeros(3);
77
78     % System matrix relating [dp dv dpsi db_a db_omaga]' between states
79     F = [F_11    I      Z      Z      Z
80           Z      F_22   F_23   F_24   Z
81           F_31   F_32   F_33   Z      F_35
82           Z      Z      Z      Z      Z
83           Z      Z      Z      Z      Z];
84
85     % Control input matrix relating noise [n_a n_omega n_ba n_bomega]' to next state
86     G = [Z      Z      Z      Z
87           est_C_bn Z      Z      Z
88           Z      est_C_bn Z      Z
89           Z      Z      I      Z
90           Z      Z      Z      I];
91
92     % covariance propagation/modification and state prediction
93     A = expm(F * tdint);
94     B = (eye(size_x) + A) * G * tdint/2;
95     Q2 = B * Q * B';
96     x = A * x;
97     P = A * P * A' + Q2;
98 end
99
100 %% Update
101 function [x, P] = update_EKF(x, P, R, type, est_llh_gps, est_llh, l_b, est_omega_ib, ...
102                               est_C_bn, est_v_eb, est_v_eA, est_h_b, h_n)
103
104     % Measurement submatrices
105     I = eye(3);
106     Z = zeros(3);
107     l_n = est_C_bn*l_b;
108
109     % Measurement matrix for position error
110     H_p = [H_p1   H_p2   H_p3   H_p4   H_p5];
111

```

```

112      % Measurement matrix for velocity error
113      H_v = [H_v1    H_v2    H_v3    H_v4    H_v5];
114
115      % Measurement matrix for magnetic field error
116      H_h = [H_h1    H_h2    H_h3    H_h4    H_h5];
117
118      H = [H_p; H_v; H_h];
119
120      % calculate difference between expected measurements given by INS and
121      % strapdown algorithm and measurements by GPS
122      z = input_measurement(est_llh_gps, est_llh, l_b, est_omega_ib, est_C_bn, ...
123                            est_v_eb, est_v_eA, est_h_b, h_n);
124
125      % do update of state and covariance matrix
126      zpred = H * x;
127      v = z - zpred;
128
129      switch type
130      case 1
131          [x, P] = KF_update_simple(x, P, v, R, H);
132      case 2
133          [x, P] = KF_update_joseph(x, P, v, R, H);
134      case 3
135          [x, P] = KF_update_cholesky(x, P, v, R, H);
136      otherwise
137          error('Invalid choice of KF update')
138      end
139  end
140
141 %% calculate difference between estimates and measurements
142 function d_y = input_measurement(est_llh_gps, est_llh, l_b, est_omega_ib, est_C_bn, ...
143                                   est_v_eb, est_v_eA, est_h_b, h_n)
144
145     % measured position
146     lat = est_llh_gps(1);
147     lon = est_llh_gps(2);
148     h = est_llh_gps(3);
149
150     % estimated position
151     lat_est = est_llh(1);
152     lon_est = est_llh(2);
153     h_est = est_llh(3);
154     % curvature of earth surface in north and east direction
155     [R_n R_e] = radicurv(lat_est);
156
157     % delta in position (meas - est)
158     d_y_p = [(lat-lat_est)*(R_n+h_est)
159               (lon-lon_est)*(R_e+h_est)*cos(lat_est)
160               -(h-h_est)];
161
162     % delta in velocity (meas - vrot - est)
163     d_y_v = est_v_eA - est_C_bn*skewsymm(est_omega_ib)*l_b - est_v_eb;
164
165     % delta in magnetic field
166     d_y_h = est_h_b - est_C_bn'*h_n;
167
168     d_y = [d_y_p; d_y_v; d_y_h];
169  end

```

7 Höhenfilter

Oft werden genaue Höheninformationen benötigt, um die Position eines Objektes zu bestimmen. Eine Möglichkeit zur Höhenbestimmung ist die barometrische Höhenschätzung wie sie in Kapitel 3 vorgestellt wird. Wie im Abschnitt 3.3 gezeigt, besteht bei der barometrischen Höhenschätzung allerdings das Problem, dass die Fehler mit der Zeit immer größer werden können. Der Grund dafür liegt darin, dass für die Höhenberechnung Referenzwerte für Temperatur und Luftdruck auf einer Referenzhöhe benötigt werden. Wie gezeigt, können sich diese Werte allerdings mit der Zeit ändern. Werden die verwendeten Referenzwerte für die barometrische Höhenberechnung nicht angepasst, liefert die Berechnung fehlerhafte Ergebnisse. Eine einfache Anpassung der Werte ist allerdings auch nicht ohne Weiteres möglich, da für die Bestimmung jeweils wiederum eine Referenzhöhe bekannt sein muss. In diesem Kapitel soll ein Verfahren vorgestellt werden, welches es in Anlehnung an [BARER] ermöglicht, die in der barometrischen Höhenschätzung auftretenden Fehler zu erkennen und zu korrigieren.

7.1 Grundlagen

Das hier vorgestellte Verfahren soll es nun ermöglichen, unter Verwendung von Referenzhöhen, die Fehler in der barometrischen Höhenberechnung, die durch veränderte Referenzparameter verursacht werden, zu bestimmen und zu korrigieren. Die Idee des Verfahrens ist es, an Punkten mit bekannter Höhe, sogenannten Referenzhöhen, die Abweichung zwischen der geschätzten Höhe und der bekannten Referenzhöhe zu messen. Um mit dieser Abweichung auf die Änderungen der Referenzwerte für Temperatur und Luftdruck zu schließen, soll im Folgenden ein Modell hergeleitet werden, welches den Einfluss der sich ändernden Referenztemperatur und des sich ändernden Referenzdruckes auf die Höhenschätzung darstellt. Außerdem soll gezeigt werden, auf welche Weise die jeweiligen Referenzhöhen gewonnen werden können.

7.2 Herleitung des Fehlermodells sowie der Kalman-Filter Gleichungen

Fehlermodell

Ausgangspunkt der Herleitung eines Fehlermodells ist die barometrische Höhenformel (3.16) wie sie in Abschnitt 3.2 hergeleitet wurde:

$$h(P, P_0, T_0) = h_0 + \left[\left(\frac{P_0}{P} \right)^{\frac{R \cdot \gamma}{M \cdot g_0}} - 1 \right] \cdot \frac{T_0}{\gamma}, \quad (7.1)$$

wobei γ , g_0 , R und M wie im Abschnitt 3.1 definiert sind. h_0 , P_0 und T_0 sind jeweils die Referenzhöhe für die barometrische Höhenberechnung, der initiale Referenzdruck und die initiale Referenztemperatur. P ist der gemessene Luftdruck. Von der Gleichung (7.1) wird nun eine Taylorreihenentwicklung um die Entwicklungspunkte $T_0 + \Delta T_0$ und $P_0 + \Delta P_0$ durchgeführt, um die Auswirkung der Änderung der Referenzwerte zu bestimmen. Die Taylorreihenentwicklung wird außerdem nach dem Glied erster Ordnung abgebrochen. Mit

$$\frac{\partial h}{\partial T_0} = \left[\left(\frac{P_0}{P} \right)^{\frac{R \cdot \gamma}{M \cdot g_0}} - 1 \right] \cdot \frac{1}{\gamma} \quad (7.2)$$

und

$$\frac{\partial h}{\partial P_0} = \frac{T_0 R \gamma}{\gamma M g_0} \left[\left(\frac{1}{P} \right)^{\frac{R \gamma}{M g_0}} \cdot P_0^{\frac{R \gamma}{M g_0} - 1} \right] = \frac{T_0 R}{M g_0 P_0} \left[\left(\frac{P_0}{P} \right)^{\frac{R \gamma}{M g_0}} \right] \quad (7.3)$$

ergibt sich nun

$$h(P, T_0 + \Delta T_0, P_0 + \Delta P_0) \approx h(P, T_0, P_0) + \frac{\partial h}{\partial T_0} \Delta T_0 + \frac{\partial h}{\partial P_0} \Delta P_0 \quad (7.4)$$

$$= h(P, T_0, P_0) + \left[\left(\frac{P_0}{P} \right)^{\frac{R \cdot \gamma}{M \cdot g_0}} - 1 \right] \cdot \frac{\Delta T_0}{\gamma} + \frac{T_0 R}{M g_0 P_0} \left[\left(\frac{P_0}{P} \right)^{\frac{R \gamma}{M g_0}} \right] \cdot \Delta P_0 \quad (7.5)$$

$$= h(P, T_0, P_0) + \left[\frac{h(P, T_0, P_0) - h_0}{T_0} \right] \Delta T_0 + \frac{T_0 R}{M g_0} \left(\frac{\Delta P_0}{P_0} \right) \left(\frac{P_0}{P} \right)^{\frac{R \gamma}{M g_0}}. \quad (7.6)$$

Schließlich wird nun noch

$$\left(\frac{P_0}{P} \right)^{\frac{R \gamma}{M g_0}} \approx 1 \quad (7.7)$$

angenommen, da sich dieser Term bei den zu erwartenden Werten von P und P_0 nur geringfügig von 1 unterscheidet. Außerdem wird

$$\frac{\Delta T_0}{T_0} = a \quad \text{und} \quad \frac{T_0 R}{M g_0} \left(\frac{\Delta P_0}{P_0} \right) = b \quad (7.8)$$

gesetzt. Hiermit ergibt sich nun ein sehr einfacher Ausdruck für ein Fehlermodell zur barometrischen Höhenberechnung:

$$h_b = h + a \cdot (h - h_0) + b. \quad (7.9)$$

Es ist zu erkennen, dass sich die fehlerhaft berechnete barometrische Höhe h_b aus der korrekten Höhe h , einer mit dem Skalierungsfaktor a skalierten Höhendifferenz aus Referenzhöhe h_0 und korrekter Höhe h und einem Bias b zusammensetzt. Der Skalierungsfaktor a hängt dabei nur von der Referenztemperaturänderung ΔT_0 sowie der initialen Referenztemperatur T_0 und der Bias b nur von der Referenzdruckänderung ΔP_0 sowie dem initialen Referenzdruck P_0 ab.

Modell für die Höhenmessung

Die Parameter a und b sollen nun mit einem Kalman-Filter geschätzt werden. Dazu ist es nötig, die gemessene Höhe in Abhängigkeit der zu schätzenden Fehler darzustellen. Für die barometrische Höhenberechnung wird zunächst eine Referenzhöhe $h_0 = 0$ m festgelegt. Die zum Zeitpunkt k gemessenen barometrische Höhe $h_{b,k}$ wird nach Gleichung (7.9) durch

$$h_{b,k} = (1 + a_k)h_k + b_k + n_{b,k} \quad (7.10)$$

dargestellt. $n_{b,k}$ ist dabei der zufällige Fehler in der Höhenberechnung, welcher durch das Sensorrauschen des Barometers nach (3.21) verursacht wird. Dabei wird die Näherung

$$n_{b,k} \sim \mathcal{N} \left(0, \left(0.1 \frac{\text{m}}{\text{Pa}} \cdot \sigma_P \right)^2 \right) \quad (7.11)$$

verwendet. σ_P^2 ist dabei die Varianz des Sensorrauschen des Barometers in Pa. Der Faktor $0.1 \frac{\text{m}}{\text{Pa}}$ gibt an, dass eine Druckänderung um 1 Pa etwa 0.1 m Höhenänderung entspricht. Die Höhenänderung, welche sich bei einer festen Druckänderung von 1 hPa ergibt, wird als barometrische Höhenstufe bezeichnet [BAHST]. Die barometrische Höhenstufe entspricht etwa 10 m für 1 hPa, wobei sich dieser Wert je nach Referenztemperatur und Referenzdruck ändern kann. Für den erwarteten Einsatzbereich sind nur geringfügige Änderungen von etwa 10 % zu erwarten, so dass dieser Wert als konstant betrachtet wird.

Modell für die Referenzhöhe

Des weiteren wird davon ausgegangen, dass zu bestimmten Zeitpunkten eine Vergleichs- oder Referenzhöhe $h_{t,k}$ durch

$$h_{topo,k} = h_k + n_{topo,k} \quad (7.12)$$

gegeben ist.

$$n_{topo,k} \sim \mathcal{N} (0, \sigma_t^2) \quad (7.13)$$

gibt dabei die Unsicherheit in der Referenzhöhe an, wobei σ_t die Standardabweichung des Fehlers in der Referenzhöhe angibt. Auf ein Verfahren zur Bestimmung einer Referenzhöhe soll später noch eingegangen werden.

Messgleichung für den Höhenfehler

Bildet man nun die Differenz aus barometrischer Höhe $h_{b,k}$ und Referenzhöhe $h_{t,k}$, erhält man eine Messgleichung für den Höhenfehler. Es ergibt sich

$$h_{b,k} - h_{topo,k} = a_k h_k + b_k + n_{b,k} - n_{topo,k}. \quad (7.14)$$

Aus Gleichung (7.12) ergibt sich die Annahme

$$h_k = h_{topo,k} - n_{topo,k}. \quad (7.15)$$

Nach Einsetzen von Gleichung (7.15) in Gleichung (7.14) folgt

$$h_{b,k} - h_{topo,k} = (h_{topo,k} - n_{topo,k}) a_k + b_k + n_{b,k} - n_{topo,k} \quad (7.16)$$

$$= a_k h_{topo,k} + b_k + n_{b,k} - (1 + a_k) n_{topo,k} \quad (7.17)$$

$$\Rightarrow z_k = a_k h_{topo,k} + b_k + n_{z,k} \quad (7.18)$$

mit

$$n_{z,k} \sim \mathcal{N}(0, \sigma_b^2 + (1 + a_k)^2 \sigma_{topo}^2). \quad (7.19)$$

Messgleichung für den Referenztemperaturfehler

Mit Hilfe der Gleichung (3.11)

$$T_k(h) = T_{0,k} + \gamma \cdot h_k, \quad (7.20)$$

lässt sich eine Messgleichung für den Referenztemperaturfehler ΔT_0 und mit (7.8) auch für den Skalierungsfaktor a angeben. Dazu wird Gleichung (7.20) nach $T_{0,k}$ umgestellt und die Differenz von $T_{0,0}$ und $T_{0,k}$ gebildet. Mit (7.15) ergibt sich nun eine Messgleichung für die Messgröße μ_k des Skalierungsfaktors a_k :

$$\Delta T_{0,k} = T_k - \gamma h_k - T_{0,0} \quad (7.21)$$

$$\Rightarrow \mu_k = \frac{1}{T_{0,0}} (T_k - \gamma \cdot (h_{topo,k} - n_{topo,k}) - T_{0,0} + n_{T,k}) \quad (7.22)$$

$$\Rightarrow \mu_k = \frac{T_k - \gamma \cdot h_{topo,k} - T_{0,0}}{T_{0,0}} + \frac{n_{T,k} + \gamma n_{topo,k}}{T_{0,0}} \quad (7.23)$$

$$\Rightarrow \mu_k = a_k + n_{\mu,k}, \quad (7.24)$$

wobei

$$n_{\mu,k} = \frac{n_{T,k} + \gamma n_{topo,k}}{T_{0,0}} \quad \text{und} \quad a_k = \frac{T_k - \gamma \cdot h_{topo,k} - T_{0,0}}{T_{0,0}}. \quad (7.25)$$

Auch hier soll gelten:

$$n_{\mu,k} \sim \mathcal{N}\left(0, \frac{\sigma_T^2 + \gamma \cdot \sigma_{topo}^2}{T_{0,0}^2}\right) \quad (7.26)$$

Dabei stellt $n_{T,k}$ das Rauschen in den Temperaturmessungen nach (3.20) dar.

Modellierung von Bias und Skalierungsfaktor

Da der Skalierungsfaktor a und der Bias b jeweils direkt von der Referenztemperaturänderung und der Referenzdruckänderung abhängen, werden a und b , wie die Referenzwerte in (3.17) und (3.18), als Random-Walk Prozess modelliert. Damit ergibt sich

$$a_k = a_{k-1} + \sqrt{\Delta t_{baro}} n_{\alpha,k} \quad \text{mit} \quad n_{\alpha,k} \sim \mathcal{N} \left(0, \left(\frac{\sigma_{T_0}}{T_{0,0}} \right)^2 \right) \quad (7.27)$$

und

$$b_k = b_{k-1} + \sqrt{\Delta t_{baro}} n_{\beta,k} \quad \text{mit} \quad n_{\beta,k} \sim \mathcal{N} \left(0, \left(\frac{T_0 R}{M g_0 P_0} \cdot \sigma_{P_0} \right)^2 \right) \quad (7.28)$$

σ_{T_0} und σ_{P_0} sind die jeweiligen Standardabweichungen der Referenzdruck- und Referenztemperaturänderung.

Systemmodell und Messmodell

Mit den obigen Gleichungen sind nun alle nötigten Grundlagen gegeben, um ein Systemmodell sowie Messmodell für eine Kalman-Filterung aufzustellen. Das Systemmodell und das Messmodell sollen hier nun noch einmal zusammengefasst werden. Mit den Gleichungen (7.27) und (7.28) lässt sich das Systemmodell wie folgt aufstellen:

$$\begin{pmatrix} a \\ b \end{pmatrix}_k = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}_{k-1} + \begin{pmatrix} \sqrt{\Delta t_{baro}} & 0 \\ 0 & \sqrt{\Delta t_{baro}} \end{pmatrix} \begin{pmatrix} n_\alpha \\ n_\beta \end{pmatrix}_k. \quad (7.29)$$

Mit den Gleichungen (7.18) und (7.24) lässt sich das Messmodell aufstellen:

$$\begin{pmatrix} z \\ \mu \end{pmatrix}_k = \begin{pmatrix} h_{topo,k} & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}_k + \begin{pmatrix} n_z \\ n_\mu \end{pmatrix}_k. \quad (7.30)$$

Hierbei wird h_{topo} als bekannte Größe behandelt. Die Unsicherheit in h_{topo} ist gemäß Gleichung (7.19) in n_z berücksichtigt.

Korrektur der berechneten Höhe

Sind nun die Fehlerparameter bekannt, kann für die berechnete Höhe $h_{b,k}$ eine korrigierte Höhe berechnet werden. Dazu wird die Gleichung (7.10) nach h_k umgestellt. Es ergibt sich für die korrigierte Höhe h_k^+ dann:

$$h_k^+ = \frac{h_{b,k} - \hat{b}_k}{1 + \hat{a}_k} \quad (7.31)$$

Abschätzung der Rauschwerte für die korrigierte Höhe

Zur Abschätzung der Rauschwerte der korrigierten barometrischen Höhe wird Gleichung (7.31) betrachtet. Hierzu wird Gleichung (7.10) in Gleichung (7.31) eingesetzt

und die Varianz des Höhenfehlers also der Differenz zu h_k bestimmt. Damit ergibt sich

$$h_k^+ - h_k = \frac{(1 + a_k)h_k + b_k + n_{b,k} - \hat{b}_k}{1 + \hat{a}_k} - h_k \quad (7.32)$$

$$= \frac{h_k(a_k - \hat{a}_k) + (b_k - \hat{b}_k) + n_{b,k}}{1 + \hat{a}_k} \quad (7.33)$$

Ersetzt man nun die Differenz aus dem Schätzwert und dem tatsächlichen Wert durch die Schätzfehlervarianz, ergibt sich

$$h_k^+ - h_k = \frac{h_k(n_{\alpha^+,k}) + (n_{\beta^+,k}) + n_{b,k}}{1 + a_k - n_{\alpha^+,k}} \quad (7.34)$$

Der Erwartungswert wird nun von

$$E[(h_k^+ - h_k)(h_k^+ - h_k)^T] = \dots \quad (7.35)$$

gebildet. Hierbei treten allerdings einige Schwierigkeiten auf. Der Quotient aus zwei gaußverteilten Zufallsvariablen führt zu einer Cauchy-Verteilung [CAUCY] dessen Parameter nicht ganz einfach zu Berechnen sind. Außerdem existieren aufgrund des Systemmodells Kreuzkorrelationen zwischen $n_{\alpha^+,k}$ und $n_{\beta^+,k}$. Aus diesem Grund soll hier eine Approximation angegeben werden für die folgende Annahmen gemacht werden: Der Term unter dem Bruch ist ungefähr 1; die Kreuzkorrelationen werden vernachlässigt. Hiermit ergibt sich dann:

$$h_k^+ - h_k = h_k(n_{\alpha^+,k}) + (n_{\beta^+,k}) + n_{b,k} \quad (7.36)$$

Die Schätzfehlervarianzen für a und b können dabei in erster Näherung den Diagonalelementen der Schätzfehlerkovarianz des Höhenfilters entnommen werden. Hierbei zeigt sich allerdings ein Nachteil des Filterentwurfs: Mit steigender Höhe wird die korrigierte Höhe immer ungenauer. Dies ist hier allerdings kein großes Problem, da dies erst bei großen Höhe einen signifikanten Einfluss hat. Für die spätere Fehlerfilterung wird die Varianz des Fehlers der korrigierten Höhe konstant gesetzt. Korrekterweise müsste man die mit der Zeit ansteigende Unsicherheit in a und b berücksichtigen, wenn keine Korrektur der barometrischen Höhe stattfindet. Tests mit konstanten Wert lieferten allerdings keine schlechteren Ergebnisse.

7.3 Referenzhöhe

Zur Nutzung des oben beschriebenen Verfahrens sind Referenzhöhen nötig. Diese Referenzhöhen sollten mit einer möglichst hohen Genauigkeit zur Verfügung stehen. Andererseits genügt es allerdings, wenn diese Höhen nur mit einer geringen Datenrate zur Verfügung stehen. So reicht es z. B. aus, dass die Fehlerbestimmung alle 100 Sekunden durchgeführt wird. Die Nutzung der GPS-Höhe als Referenzhöhe ist aufgrund seiner hohen Ungenauigkeit, wie in Abschnitt 4.2.1 gezeigt, nicht möglich. Als Alternative zur Nutzung der GPS-Höhe soll hier nun ein sogenanntes “Map Matching” oder “Kartenabgleichsverfahren” genutzt werden. Bei dem “Map Matching” wird die aktuelle

Position in der Ebene, bestehend aus Länge λ und Breite φ , mit der Position von Referenzpunkten in einer topographischen Karte verglichen. Befindet man sich nun "in der Nähe" eines Referenzpunktes, so kann dessen Höhe als aktuelle Referenzhöhe verwendet werden und eine Fehlerbestimmung durchgeführt werden. Als Referenzpunkte können z. B. Höhenfestpunkte, wie sie von den Landesvermessungsämtern zur Verfügung gestellt werden, verwendet werden. Diese Punkte stehen je nach Gebiet in Abständen von wenigen 100 Metern bis einigen Kilometern zur Verfügung. Ihre Position und Höhe ist im Normalfall bis auf wenige Zentimeter genau bestimmt. Für diese Arbeit wurden entsprechende Referenzpunkte aus der Deutschen Grundkarte für den Bereich verwendet in dem eine Datenaufzeichnung durchgeführt wurde [TIMOL]. Die Abbildung 7.1 zeigt Beispiele von Karten die Höheninformationen beinhalten. Prinzipiell würde aber auch eine Datenbank mit Höhenfestpunkten und deren Koordinaten ausreichen.

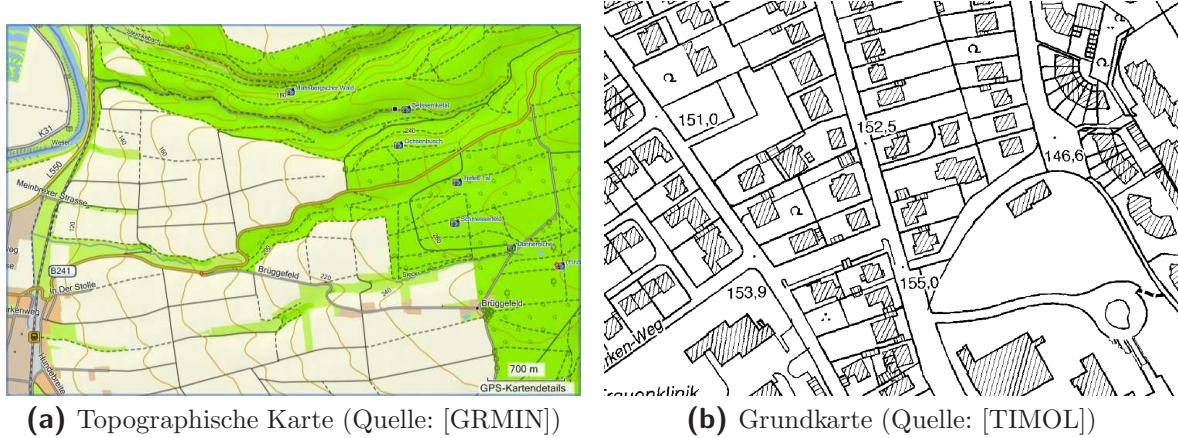


Abbildung 7.1: Unterschiedliche Karten mit Referenzhöhen

Um nun zu bestimmen ob man sich in der Nähe eines Referenzpunktes befindet, wird folgender Algorithmus verwendet:

1. Die Position und Höhe der Referenzpunkte wird in einer Tabelle gespeichert.
2. Als erstes wird der minimale Abstand zwischen der aktuellen Position und den Positionen der Referenzpunkte in der Tabelle bestimmt. Zur Abstandsbestimmung wird der euklidische Abstand, in Meter, genutzt.
3. Liegt der minimale Abstand unterhalb eines festgelegten Schwellwertes, so wird die zugehörige Höhe aus der Tabelle als Referenzhöhe genutzt. Ein Schwellwert kann z. B. 5 m sein.

Die Abstandsbestimmung wird analog zu den Gleichungen (6.2) und (6.3) durchgeführt:

$$\Delta x_{n,i} = R \cdot (\hat{\varphi}_p - \varphi_{r,i}) \quad (7.37)$$

$$\Delta x_{e,i} = R \cdot \cos(\hat{\varphi}_p) \cdot (\lambda_p - \lambda_{r,i}) \quad (7.38)$$

Es wird mit $R = 6371000\text{ m}$ ein mittlerer Erdradius angenommen. $\hat{\varphi}_p$ und $\hat{\lambda}_p$ ist die aktuelle Positionsschätzung in Form von Längen- und Breitengrad. $\varphi_{r,i}$ und $\lambda_{r,i}$ sind

die jeweiligen Positionen der Referenzpunkte. Anschließend wird der minimale Abstand mit

$$d_{min} = \min_i \left(\sqrt{\Delta x_{n,i}^2 + \Delta x_{e,i}^2} \right) \quad (7.39)$$

bestimmt. Ist nun

$$d_{min} < d_{schwell} \quad (7.40)$$

wird $h_t = h_{ref}(i)$ als Referenzhöhe verwendet. Andernfalls wird keine Referenzhöhe ausgegeben. h_{ref} enthält dabei die jeweiligen Referenzhöhen zu den Referenzpunkten aus der Tabelle. Für die spätere Fehlerfilterung wird angenommen, dass die Referenzhöhe h_t genau bekannt ist und damit $\sigma_t = 0$ gesetzt. Diese Annahme ist gerechtfertigt, da die Referenzhöhen selber hoch genau bestimmt wurden. Außerdem wird der Unterschied zwischen der Referenzhöhe und der wirklichen Höhe bei einem Abstand von maximal 5 m im Allgemeinen zu vernachlässigen sein.

7.4 Implementierung

In diesem Kapitel soll die Implementierung der Filterung allgemein und im speziellen in Matlab beschrieben werden. Die Filterung wird dabei so aufgebaut, dass die eigentliche barometrische Höhenberechnung und Fehlerfilterung in einem Funktionsblock stattfinden. Dies hat den Vorteil, dass durch einen Funktionsaufruf direkt aus den Messwerten des Barometers eine korrigierte Höhe bestimmt werden kann, wenn eine Referenzhöhe gegeben ist. Lediglich die Referenzhöhe wird außerhalb der Funktion bestimmt und beim Funktionsaufruf übergeben. Aber auch dies hat den Vorteil, dass die Bestimmung der Referenzhöhe leicht angepasst werden kann. Das Vorgehen zur Bestimmung einer korrigierten Höhe soll hier kurz skizziert werden:

1. Im ersten Schritt sind die initialen Werte für die barometrische Höhenberechnung zu setzen. Hierzu gehören die Referenztemperatur T_0 , der Referenzdruck P_0 und die zugehörige Referenzhöhe h_0 . Bereits hier kann eine erste Messwertaufnahme stattfinden, um die Werte zu bestimmen.
2. Anschließend sind die Messwerte des Barometers und GPS-Empfängers aufzunehmen.
3. Im nächsten Schritt sind die Rauschparameter für das Systemmodell sowie für das Messmodell zu setzen. Diese Werte können sowohl statisch gesetzt als auch dynamisch bestimmt werden. Auch die Initialisierung der Kovarianzen des Kalman-Filters ist möglich, wenn dies nötig ist.
4. Ist die Initialisierung abgeschlossen, kann mit der Höhenberechnung begonnen werden. Zuerst wird dafür geprüft, ob sich in der Nähe ein Referenzpunkt befindet. Ist dies der Fall, so wird dieser verwendet.
5. Anschließend kann der Funktionsblock zur Höhenbestimmung aufgerufen werden. In diesem Block wird zuerst die barometrische Höhe mit den gegebenen Referenzwerten berechnet. Ist nun eine Referenzhöhe gegeben, wird das Kalman-Filter zur Fehlerbestimmung aufgerufen und die Fehlerparameter abgespeichert. Ist keine

Referenzhöhe gegeben, wird nur die Kovarianzmatrix des Zustandes propagiert. Unabhängig davon, ob eine Referenzhöhe gegeben war, wird mit den zuletzt bekannten Fehlerparametern eine korrigierte Höhe berechnet. Diese korrigierte Höhe wird schließlich ausgegeben.

6. Im nächsten Schritt können nun die nächsten Messewerte aufgenommen und verarbeitet werden. Es wird also wieder bei Schritt 3 begonnen.

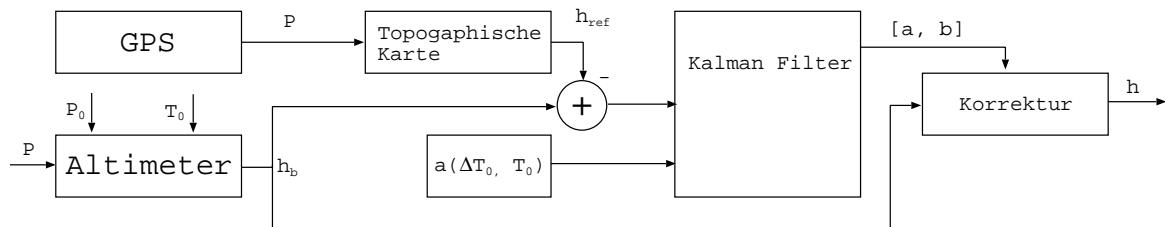


Abbildung 7.2: Filter zur Berechnung einer korrigierten Höhe

Die Abbildung 7.2 zeigt ein Blockschaltbild des Systems.

Die Demonstration der Höhenfilterung für real aufgenommene Daten ist im Matlab-Skript “data_demo.m” implementiert. Dabei wurden GPS-Positionen und die Messewerte des Barometers aufgezeichnet. Die Referenzpunkte wurden von Hand aus den oben genannten Datenquellen gewonnen.

Listing 7.1: data_demo.m

```

1 % This script will demonstrate:
2 %   - the usage of recorded data consisting of pressure, temperature, 2D
3 %     positions and reference data consisting of 2D positions and heights
4 %   - height calculation based on pressure readings and filtering based on
5 %     reference heights using "baro_height_filter"
6
7 %% load preprocessed data (llh transformed from degmm.mmm to deg.ssss)
8 data_path = '../../../../../data/Oak-Pressure/Testfahrt-13.08.2009-Rad/';
9 load([data_path,'stamp-pressure-temp-pos.m'], '-mat') % measurements
10 load([data_path, 'refpos.m'], '-mat') % reference positions/heights
11 P = data(:,2); % get pressure readings
12 % get temperature readings (reduced by 5.5K because of sensor offset during measurement)
13 T = data(:,3)-5.5;
14 llh = data(:,4:5); % get latitude/Longitude
15 npts = size(P,1); % number of measurements
16
17 % initial values
18 h_init = 170; % h1 = 170m
19 t_init = 16.5+273.15; % T1 = 16.5°C at h1
20 p_init = P(1); % initial pressure at h1
21
22 %% Filtering
23 dt = 0.1; % time step
24 std_P = 10; % standard deviation of noise in pressure readings (Pa/Sample)
25 std_T = 0.02; % standard deviation of noise in temperature readings (K/Sample)
26 % standard deviation in atmospheric temperature change (of reference temperature)
27 std_T0 = 0.07;
28 % standard deviation in atmospheric pressure change (of reference pressure)
29 std_P0 = 4;
30
  
```

```

31 % estimate for scale factor standard deviation using T0 = 270K
32 std_scale = std_T0/270*sqrt(dt);
33 % estimate for bias standard deviation using 0.1m per Pa
34 std_bias = std_P0*0.1*sqrt(dt);
35 % estimate for height measurement standard deviation using 0.1m per Pa
36 std_h = std_P0*0.1;
37 % estimate for scale factor measurement standard deviation using T0 = 270K
38 std_scale_t = std_T/270;
39
40 covariance = [std_scale^2 0; 0 std_bias^2]; % initial covariance
41 R = blkdiag(std_h^2, std_scale_t^2); % measurement noise
42 Q = blkdiag(std_scale^2, std_bias^2); % process noise
43
44 state = zeros(2,1); % initialize state
45 a = zeros(npts+1,1); % scale factor
46 b = zeros(npts+1,1); % bias
47 for i = 1:npts
48     h_ref(i) = get_h_ref(ref_pos, llh(i,:), 11); % find reference heights at 11m radius
49     [h_baro(i), h_baro_c(i), a(i + 1), b(i + 1), state, covariance] = ...
50         baro_height_filter(P(i), T(i), h_ref(i), h_init, t_init, ...
51                             p_init, a(i), b(i), state, covariance, R, Q);
52 end

```

Zur Bestimmung, ob ein Referenzpunkt in der Nähe liegt oder nicht, wird die Funktion "get_h_ref.m" genutzt. Dieser Funktion wird eine Tabelle mit Referenzpunkten, der aktuellen Position und die maximale Distanz übergeben.

Listing 7.2: get_h_ref.m

```

1 function [h_ref, dist] = get_h_ref(ref_pos, pos, dist_max)
2 %GET_H_REF function which returns a reference height within a radius of dist_max
3 %
4 % function [h_ref, dist] = get_h_ref(ref_pos, pos, dist_max)
5 %
6 % INPUTS
7 % ref_pos = table of reference positions and reference heights in
8 %           [lat lon height] (degree, degree, meter)
9 % pos = reference position in [lat lon] (degree, degree)
10 % dist_max = maximum distance in meter to reference point
11 %
12 % OUTPUTS
13 % h_ref = reference height if within radius of dist_max else NaN
14 % dist = distance to reference point in meter
15
16 % find nearest reference height and keep distance in meter and position in table
17 [dist idx] = min(dist_llh_enu(ref_pos(:,1:2), pos(1:2)));
18
19 % only return reference height if distance within radius of dist_max else return NaN
20 if dist < dist_max
21     h_ref = ref_pos(idx,3);
22 else
23     h_ref = NaN;
24 end

```

Zur Bestimmung der Distanz zwischen den Referenzpunkten und der aktuellen Position wird die Funktion "dist_llh_enu.m" genutzt. Dieser Funktion wird wiederum eine Tabelle mit den Positionen der Referenzpunkte und die aktuelle Position übergeben.

Listing 7.3: dist_llh_enu.m

```

1 function [dist] = dist_llh_enu (llh, ref_llh)
2 %DIST_LLH_ENU calculate distance between reference position and list of positions
3 %
4 % function [dist] = dist_llh_enu (llh, ref_llh)

```

```

5 %
6 %
7 %     llh = list of positions [lat lon] in (degree, degree)
8 %     ref_llh = reference position [lat_ref lon_ref] in (degree, degree) to calculate
9 %             distances to
10 %
11 %
12 %     OUTPUTS
13 %         dist = vector of distances to each position
14
15 dim = min(size(llh,2), size(ref_llh,2));    % if dim == 2:
16 if dim == 2                                % only use latitude and longitude for distance calculation
17     npts = size(llh,1);                      % number of samples
18     llh(:,3) = zeros(npts,1);                 % set heights to zero so we can use llh2enu
19     ref_llh(3) = 0;
20
21 enu = llh2enu(llh, ref_llh);                % get enu distances to reference
22 dist = sqrt(sum(enu.^2,2));                  % calculate euclidean distance

```

Schließlich wird zur Umrechnung der LLH-Koordinaten in ENU-Koordinaten, ausgehend von der aktuellen Position, die Funktion "llh2enu.m" genutzt. Dieser Funktion wird wiederum die Tabelle mit den Positionen der Referenzpunkte sowie die aktuelle Position übergeben. Diese Funktion implementiert die Gleichungen (7.38) und (7.37) zur Umrechnung. Damit ist das Ergebnis nur ein angenäherter Wert, welcher für kleine Distanzen aber ein ausreichende Näherung darstellt.

Listing 7.4: llh2enu.m

```

1 function [enu] = llh2enu(llh, ref_llh)
2 %LLH2ENU transforms coordinates from llh [lat lon height] to enu [east north up]
3 %
4 %     function [enu] = llh2enu(llh)
5 %
6 %     INPUTS
7 %         llh = matrix of llh coordinates [lat lon height] in (degree, degree, meter)
8 %         ref_llh = reference position [lat_ref lon_ref height] in (degree, degree, meter)
9 %             if ref_llh is left empty the mean of all position
10 %                 measurements will be used as reference/center
11 %
12 %     OUTPUTS
13 %         enu = matrix of enu coordinates [east north up] in (meter, meter, meter)
14
15 npts = size(llh,1);
16
17 R0 = 6371000;                                % Earth radius
18
19 if nargin == 1
20     ref_llh = mean(llh);                      % mean position as reference/center
21 end
22
23 lat2n = R0*(pi/180);                         % distance between two latitudes
24 % distance between two longitudes depending on the latitude
25 lon2e = R0*(pi/180)*cos(ref_llh(1)*(pi/180));
26
27 enu = (llh - repmat(ref_llh,npts,1)) * [0 lat2n 0;lon2e 0 0; 0 0 1];

```

Die Funktion "baro_height_filter.m" wird genutzt um aus Messwerten des Barometers, der Referenzhöhe, den initialen Werten zur barometrischen Höhenberechnung und dem letzten Filterzustand sowie den Kovarianzen für die Kalman-Filterung eine neue Höhe zu berechnen. Dazu wird die barometrische Höhe berechnet, das Kalman-Filter aufgerufen und die korrigierte Höhe aus der barometrischen Höhe berechnet.

Listing 7.5: baro_height_filter.m

```

1 function [h_baro, h_baro_c, a2, b2, state, covariance] = baro_height_filter (P, T, ...
2                                     h_ref, h_init, t_init, p_init, a1, b1, state, covariance, R, Q)
3 %BARO_HEIGHT_FILTER filter to calculate height readings using pressure readings and
4 %                               reference heights
5 %
6 %   function [h_baro, h_baro_c, a2, b2, state, covariance] = baro_height_filter (P, ...
7 %                                     T, h_ref, h_init, t_init, p_init, a1, b1, state, covariance, R, Q)
8 %
9 %   INPUTS
10 %     P = actual pressure readings in Pa
11 %     T = actual temperature readings in K
12 %     h_ref = reference height at acutal position
13 %     h_init = initial height for barometric calculation
14 %     t_init = initial temperature at initial height
15 %     p_init = initial pressure at initial height h
16 %     a1 = previous scale factor
17 %     b1 = previous bias value
18 %     state = prevoius state: [scale bias]'
19 %     covariance = last covariance
20 %     R = measurement noise matrix: diag([std_h^2, std_scale_t^2])
21 %     Q = process noise matrix: diag([std_scale^2, std_bias^2])
22 %
23 %   OUTPUTS
24 %     h_baro = simple barometric height
25 %     h_baro_c = corrected barometric height using height_filter
26 %     a2 = next scale factor
27 %     b2 = next bias value
28 %     state = next state: [scale bias]'
29 %     covariance = next covariance matrix
30
31 npts = size(P,1);
32 a(1) = a1;                                % inital scale factor
33 b(1) = b1;                                % initial bias
34 t0_filter = t_init + 0.0065*h_init; % T0
35
36 for i = 1:npts
37     h_baro(i) = barheight(P(i), h_init, t_init, p_init); % calculate heighth
38     if ~isnan(h_ref(i))                         % only do correction if reference height available
39         [state, covariance] = height_filter(state, covariance, h_baro(i), ...
40                                         h_ref(i), T(i), t0_filter, 1, R, Q);
41         a(i+1) = state(1);                        % scale factor
42         b(i+1) = state(2);                        % bias
43     else
44         [state, covariance] = height_filter(state, covariance, NaN, ...
45                                         NaN, NaN, NaN, 0, NaN, Q);
46         a(i+1) = a(i);                          % keep scale factor
47         b(i+1) = b(i);                          % keep bias
48     end
49     h_baro_c(i) = (h_baro(i) - b(i+1)) / (1 + a(i+1)); % do correction
50 end
51
52 a2 = a(2:end);
53 b2 = b(2:end);

```

Das eigentliche Kalman-Filter zur Fehlerbestimmung ist in der Funktion "height_filter.m" implementiert. Das Kalman-Filter bestimmt aus der barometrischen Höhe, der Referenzhöhe sowie der Temperaturmessung die aktuellen Fehler.

Listing 7.6: height_filter.m

```

1 function [state2, covariance2] = height_filter(state1, covariance1, h_baro, h_ref, ...
2                                               T, T0, update, R, Q)
3 %HEIGHT_FILTER Error Kalman filter to estimate scale and bias error in height
4 %                               measurements given a reference height
5 %

```

```

6 %     function [state2, covariance2] = height_filter(state1, covariance1, h_baro, ...
7 %                                         h_ref, T, T0, update, R, Q)
8 %
9 % INPUTS
10 %     state1 = last state: [a b]' (a := scale factor, b := bias)
11 %     covariance1 = last covariance matrix
12 %     h_baro = estimated barometric height
13 %     h_ref = reference height
14 %     T = temperature at actual height
15 %     T0 = reference temperature
16 %     update = do state update/estimation
17 %             0 = do covariance propagation only
18 %             1 = do state update/estimation
19 %     R = measurement noise matrix: diag([std_h^2 std_at^2])
20 %     Q = process noise matrix: diag([std_a.^2, std_b.^2])
21 %
22 % OUTPUTS
23 %     state2 = new state: [a b]'
24 %     covariance2 = new covariance matrix
25
26 %% Initialization
27 % some constants for the moment
28 KF_update = 2;           % KF update type
29
30 %% do filtering
31 % prediction
32 [state2, covariance2] = predict_EKF(state1, covariance1, Q);
33
34 % update if specified
35 if update==1
36     [state2, covariance2] = update_EKF(state2, covariance2, R, KF_update, ...
37                                         h_baro, h_ref, T, T0);
38 end
39 end
40
41 %% Prediction
42 function [x, P] = predict_EKF(x, P, Q)
43     % System matrix relating [a b]' between states
44     F = [1 0
45          0 1];
46
47     % Control input matrix relating noise [n_a n_b]' to next state
48     G = [1 0
49          0 1];
50
51     % covariance propagation/modification and state prediction
52     Q2 = G * Q * G';
53     x = F * x;
54     P = F * P * F' + Q2;
55 end
56
57 %% Update
58 function [x, P] = update_EKF(x, P, R, type, h_baro, h_ref, T, T0)
59
60     % Measurement matrix
61     H = [h_ref 1
62          1      0];
63
64     % calculate difference between expected measurements given by barometer and
65     % height algorithm and measurements by reference height generator
66     z = input_measurement(h_baro, h_ref, T, T0);
67
68     % do update of state and covariance matrix
69     zpred = H * x;
70     v = z - zpred;
71
72     switch type
73     case 1
74         [x, P] = KF_update_simple(x, P, v, R, H);
75     case 2

```

```

76      [x, P] = KF_update_joseph(x, P, v, R, H);
77  case 3
78      [x, P] = KF_update_cholesky(x, P, v, R, H);
79 otherwise
80     error('Invalid choice of KF update')
81 end
82 end
83
84 %% calculate difference between estimates and measurements
85 function d_y = input_measurement(h_baro, h_ref, T, T0)
86     gamma = -0.0065;
87
88     d_h = h_baro - h_ref;
89     d_a = (T - gamma*h_ref - T0)/T0;
90
91     d_y = [d_h; d_a];
92 end
93 end

```

7.5 Tests

In diesem Kapitel sollen kurze Tests des Höhenfilters anhand von simulierten und real aufgenommenen Daten durchgeführt werden. Diese Tests sollen die Funktionsfähigkeit des Höhenfilters demonstrieren.

Simulation

Als Basis für eine Simulation wird ein Temperatur- und Druckprofil verwendet wie es in Abschnitt 3.4 erzeugt wird. Als Referenzhöhen werden Höhen aus dem bekannten Höhenprofil verwendet. Dazu wird für die Simulation in Schritten von 100 Sekunden jeweils eine bekannte Höhe als Referenzhöhe verwendet. Das Map-Matching wird hier noch nicht verwendet. Auch für diese Simulation, wie schon im Abschnitt 3.6, werden die Standardabweichungen des treibenden Rauschens der Random-Walk-Prozesse um den Faktor 10 höher eingestellt um die Änderungen deutlicher zeigen zu können. Die Funktion des Filters ist aber auch bei “normalen” Werten in gleicher Weise gewährleistet, wie weitere Tests gezeigt haben. Die Abbildung 7.3 zeigt die Ergebnisse

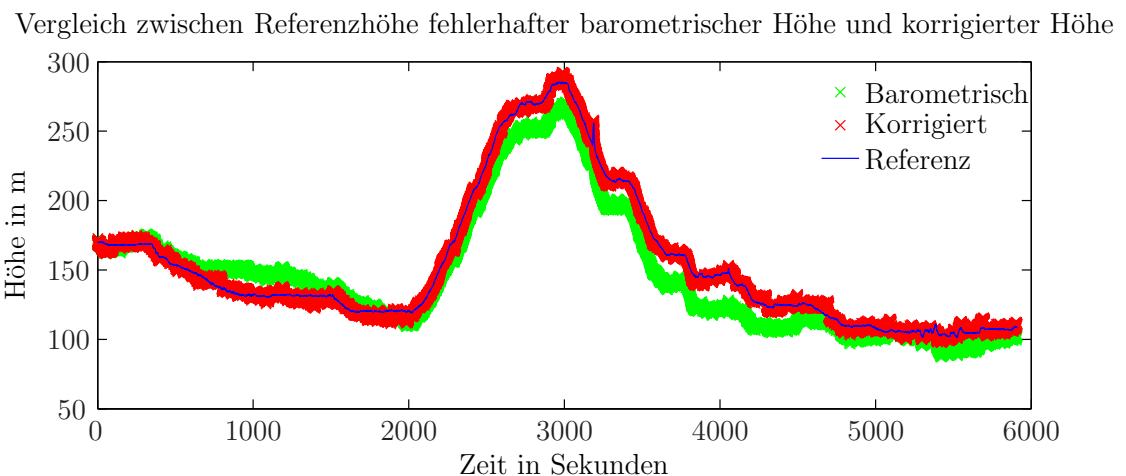


Abbildung 7.3: Test des Höhenfilters

der Höhenfilterung. Aufgezeichnet ist das ideale Höhenprofil, das fehlerbehaftete barometrische Höhenprofil und das durch das Filter korrigierte Höhenprofil. Es ist zu erkennen, dass die barometrische Höhe von der idealen Höhe an den meisten Stellen abweicht. Die korrigierte Höhe hingegen zeigt nur geringe Abweichungen zur idealen Höhe. Wie erwartet ist der Höhenfilter in der Lage die Fehler, die durch die veränderlichen Referenzwerte verursacht werden, zu korrigieren. Die Fehler welche durch das Messrauschen verursacht werden, werden nicht korrigiert. Dies ist hier allerdings auch nicht beabsichtigt.

Als nächstes sollen die geschätzten Werte für den Skalierungsfaktor und den Bias mit den theoretischen Werten verglichen werden. Dazu werden die beiden Gleichungen für den Skalierungsfaktor a und den bias b aus (7.8) ausgewertet und die Ergebnisse zusammen mit den jeweiligen geschätzten Werten in eine Grafik eingetragen. Die erste Abbildung in 7.4 zeigt das Ergebnis für den Skalierungsfaktor und die zweite Abbildung in 7.4 das Ergebnis für den Bias. Es ist deutlich zu erkennen dass der Filter in der

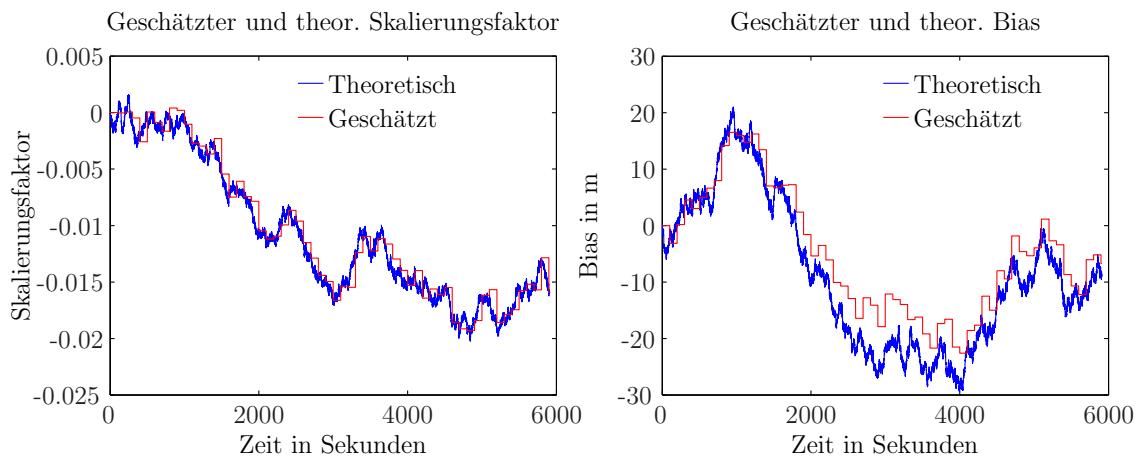


Abbildung 7.4: Skalierungsfaktor und Bias

Lage ist die Werte für den Skalierungsfaktor zu schätzen. Auch die Werte für den Bias werden geschätzt, wobei sich hier größere Abweichungen zeigen. Die größeren Abweichungen in der Schätzung des Bias liegen darin begründet, dass die theoretischen Werte nur anhand des linearisierten Modells berechnet wurden. Gut zu erkennen ist auch die grobe Auflösung der Schätzung. Dies liegt darin begründet, dass eine Schätzung nur stattfinden kann, wenn eine Referenzhöhe vorliegt. In der Zwischenzeit werden die Werte konstant gehalten.

Abschließend soll noch der eigentliche Höhenfehler dargestellt werden. Zum Vergleich werden hier jeweils der Unterschied zwischen der barometrischen Höhe und der idealen Höhe sowie der Unterschied zwischen der korrigierten Höhe und der idealen Höhe in Abbildung 7.5 dargestellt. Zur besseren Veranschaulichung der Punkte an denen eine Korrektur durchgeführt wird, werden hier nur die ersten 1000 Sekunden des Datensatzes gezeigt. Es ist deutlich zu erkennen dass der Fehler in der barometrischen Höhe mit der Zeit eine zunehmende Tendenz zeigt. Bei der korrigierten Höhe hingegen ist zu

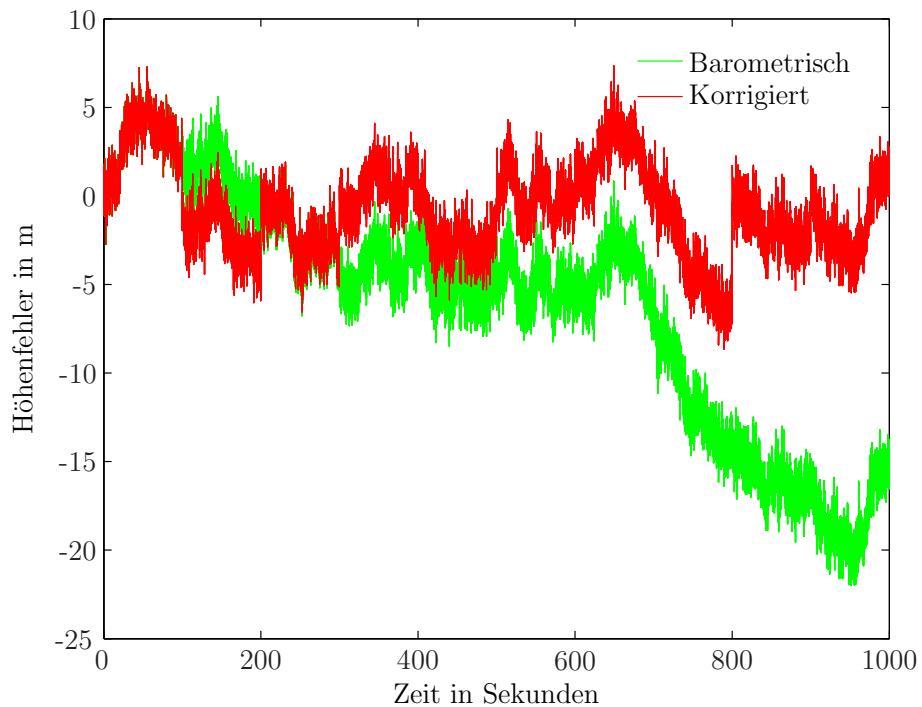


Abbildung 7.5: Höhenfehler vor/nach Korrektur

erkennen, dass sich der Fehler in Grenzen hält, in diesem Fall etwa kleiner 5 Meter. Außerdem sind die Stellen an denen eine Korrektur durchgeführt wird zu erkennen. Eine Korrektur wird alle 100 Sekunden durchgeführt und ist z. B. besonders bei 100 Sekunden, 200 Sekunden und 800 Sekunden zu erkennen.

Test mit realen Daten

Um den Höhenfilter mit Daten in einer realen Umgebung zu testen, wurde während einer Testfahrt mit dem Fahrrad ein Datensatz aufgezeichnet. Der Datensatz enthält dabei Messungen für den Luftdruck und die Temperatur sowie 2D-Positionsinformationen in der Ebene. Die Höhenschätzungen des GPS stehen aus technischen Gründen nicht zur Verfügung. Zur Datenaufzeichnung wurde eine frühe Version des NT-DataLogger verwendet. Der NT DataLogger ist ein Aufzeichnungsprogramm welches im Fachgebiet Nachrichtentechnik der Universität Paderborn entwickelt wurde um Daten unterschiedlicher Sensoren synchron und in einem einheitlichen Format aufzuzeichnen. Die Referenzhöhen wurden aus der Deutschen Grundkarte von [TIMOL] entnommen und zusammen mit ihren 2D-Koordinaten in einer Liste hinterlegt. Die Liste wurde wie in Abschnitt 7.3 beschrieben, zur Höhenfilterung verwendet. Eine Referenzhöhe wurde dann verwendet, wenn der aktuelle Abstand zur Referenzhöhe kleiner als 11 Meter war. Der Wert von 11 Meter wurde hier gewählt, da Positionsinformationen vom GPS nur mit einer Rate von 1 Hz zur Verfügung standen. Die Varianzen der Random-Walk-Prozesse wurden auf die in Abschnitt 3.3 bestimmten Werte eingestellt. Die Abbildung 7.6 zeigt die Ergebnisse der Höhenfilterung. Eingezeichnet sind jeweils die barometri-

sche Höhe, die korrigierte Höhe und, wenn vorhanden, die verwendete Referenzhöhe. Es ist zu erkennen, dass die barometrische Höhe von der korrigierten Höhe abweicht.

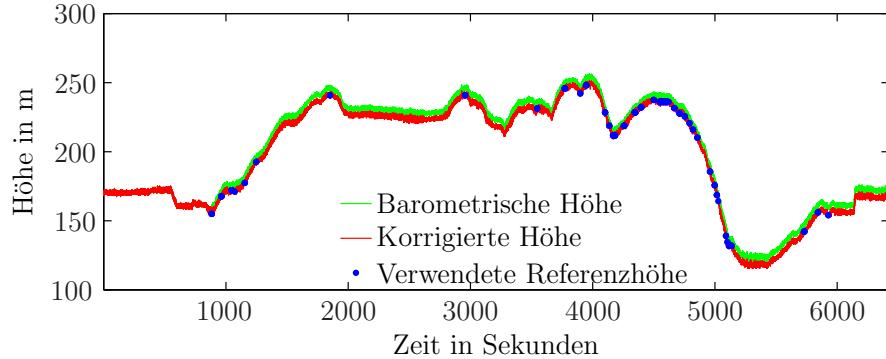


Abbildung 7.6: Barometrische und korrigierte Höhe

Dies dürfte an geringfügig falsch initialisierten Referenzwerten für Temperatur, Druck und initialer Höhe liegen. Allerdings ist auch zu erkennen, dass der Höhenfilter in der Lage ist die Fehler zu korrigieren. Die Abbildungen in 7.7 zeigen die geschätzten Werte für Skalierungsfaktor und Bias. Zusätzlich ist in Abbildung 7.7 ein “theoretischer” Wert

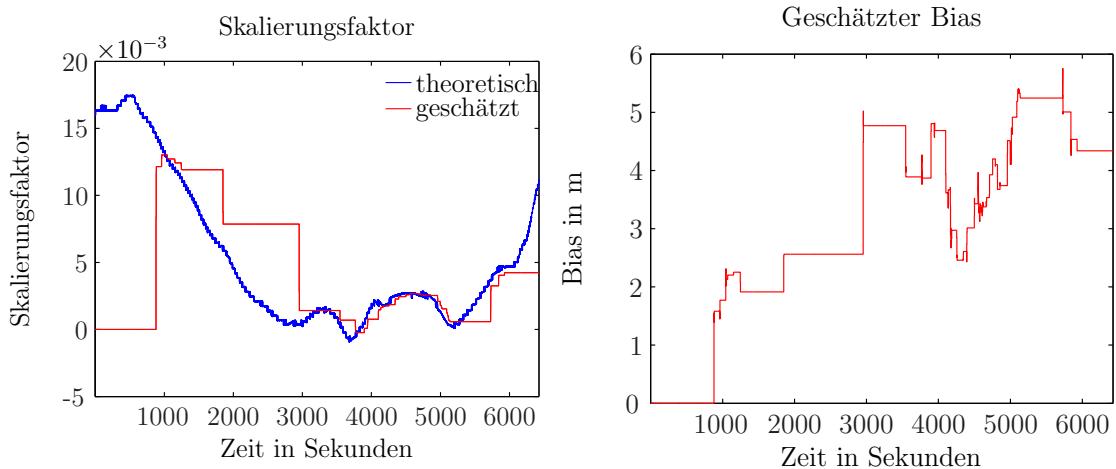


Abbildung 7.7: Skalierungsfaktor und geschätzter Bias

für den Skalierungsfaktor eingezeichnet. Dieser Wert wurde anhand der Gleichung für den Skalierungsfaktor aus (7.25) berechnet. Für die benötigte Höhe wurde die korrigierte Höhe eingesetzt. Diese Vorgehen ist formal gesehen zwar nicht ganz korrekt, liefert aber eine gute Abschätzung für den Skalierungsfaktor und soll hier nur zur graphischen Verdeutlichung und Kontrolle dienen. Es ist gut zu erkennen, dass der Höhenfilter den Skalierungsfaktor bestimmen kann. Der Abfall der Temperatur am Anfang der Messung sowie der Anstieg am Ende und damit auch die Änderung des Skalierungsfaktors sind dadurch zu erklären, dass die Messungen in einem Gebäude begonnen wurden und

auch dort endeten.

Die folgende Abbildung 7.8 zeigt die Strecke die für die Datenaufnahme abgefahren wurde. Die Strecke ist dabei als blaue Linie eingezeichnet. Die Referenzpunkte sind als rote und schwarze Punkte eingezeichnet. Ein schwarzer Referenzpunkt wurde für die Höhenfilterung verwendet. Ein roter Punkt wurde aufgrund seiner Entfernung von mehr als 11 Metern zur nächstgelegenen GPS-Messung nicht verwendet.

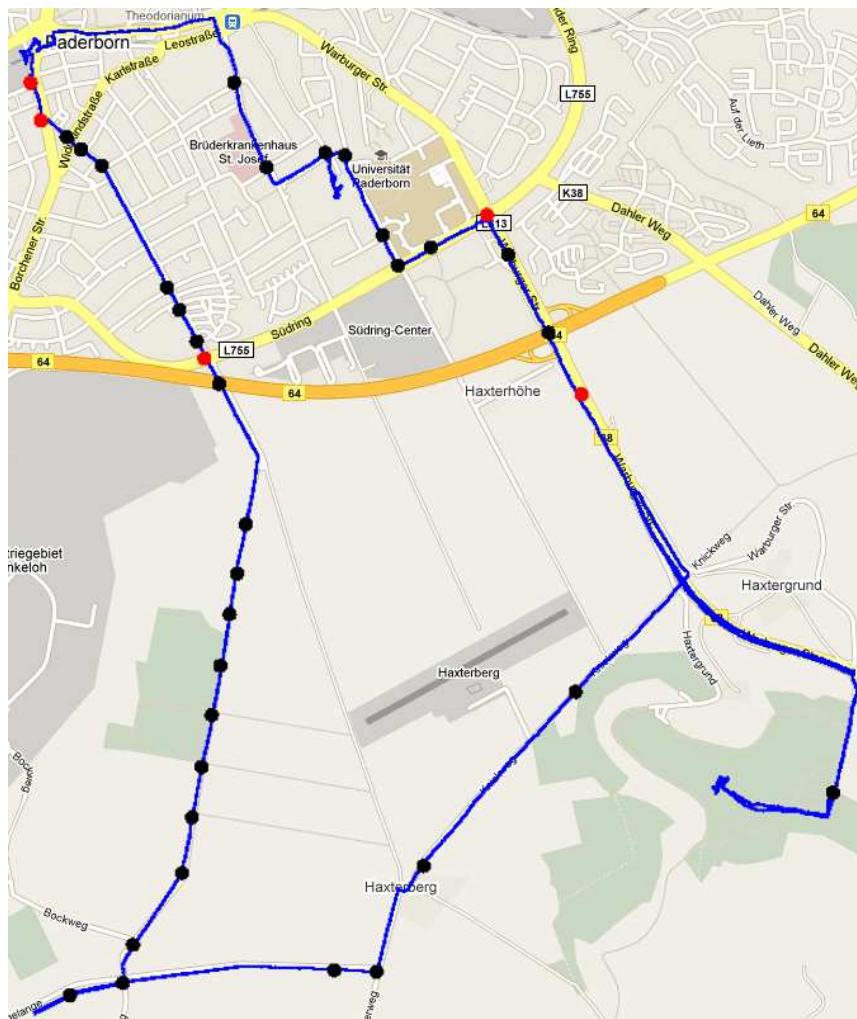


Abbildung 7.8: Teststrecke mit Referenzpunkten

Zusammenfassung

Sowohl in der Simulation als auch bei der Verwendung von realen Daten konnte gezeigt werden, dass der Höhenfilter sein Funktion erfüllt. Bei den real aufgezeichneten Daten ist außerdem gut zu erkennen, dass die barometrische Höhenbestimmung auch über längere Zeitabschnitte genaue Werte für die Höhe liefert. Insbesondere in der zweiten Hälfte, also oberhalb von 3000 Sekunden, zeigen sich Änderungen von nur etwa +/- 1 Meter im Bias. Verglichen mit einer Genauigkeit der GPS-Höhe von +/- 5 Meter, unter guten Bedingungen, ist dies ein erfolgversprechendes Ergebnis.

8 Kombination GPS/INS und Barometer

Ein Problem bei der GPS/INS-Integration ist die schlechte Qualität der vom GPS-Empfänger gelieferten Höhe. Eine Alternative zur Höhenbestimmung per GPS ist die barometrische Höhenberechnung, wie sie in Kapitel 3 vorgestellt wurde. Die barometrische Höhenberechnung liefert meist genauere Höheninformationen. Daher bietet sich eine Kombination von GPS, INS und Barometer an. Diese Kombination bietet zahlreiche Vorteile:

- Bessere Höhenschätzung und damit eine bessere Fehlerbestimmung für das INS und die Inertialsensoren.
- Höhere Datenrate der barometrischen Höhenschätzung im Vergleich zu GPS.
- Andauernde Verfügbarkeit der Höheninformationen, auch wenn kein GPS zur Verfügung steht.

All diese Vorteile führen zu einer besseren und zuverlässigeren Fehlerbestimmung für das INS im Vergleich zur Verwendung der GPS-Höhe. Aufgrund der geringeren Fehler in den Höheninformationen des Barometers, ist die Fehlerbestimmung für das INS genauer. Durch die höhere Datenrate der Höheninformationen ist es außerdem möglich, die Fehler des INS mit einer höheren Datenrate zu schätzen und damit die gesamte Schätzung zu verbessern. Außerdem bietet die Verfügbarkeit der Höheninformationen bei einem GPS-Ausfall die Möglichkeit, auch dann die Fehlerfilterung fortzuführen, wenn keine GPS-Informationen zur Verfügung stehen. Aufgrund von möglichen Korrelationen zwischen den einzelnen Fehlern kann auch alleine die Verwendung der Höheninformation die gesamte Fehlerschätzung verbessern und somit zu einer besseren Stützung des INS führen.

8.1 Kombination

Die Kombination von GPS/INS und Barometer lässt sich auf sehr einfache Weise durchführen. Anstelle den GPS-Messwert für die Höhe zu verwenden, wird bei der Kalman-Filterung die barometrische Höhe verwendet. Des weiteren müssen die Rauschwerte für die Höhenmessung entsprechend angepasst werden. Zu der in Abschnitt 6.3 beschriebenen Implementierung ergeben sich dabei folgende Änderungen:

- Zusätzlich zu der Initialisierung der Werte für das INS müssen die Werte für die barometrische Höhenberechnung initialisiert werden.

- Des weiteren ist zusätzlich eine Messwertaufnahme des Barometers nötig.
- Außerdem sind die Rauschwerte für das Systemmodell sowie das Messmodell des Höhenfilters zu bestimmen und/oder zu setzen. Auch eine Abschätzung für die zu erwartenden Rauschwerte der später korrigierten barometrischen Höhe ist möglich.
- Neben der Berechnung der Navigationslösung ist nun zusätzlich die Berechnung der korrigierten barometrischen Höhe nötig. Hierzu wird zuerst anhand der vom INS gelieferten Positionsinformationen geprüft, ob eine Referenzhöhe bekannt ist. Anschließend wird das Höhenfilter zur Berechnung und Korrektur der Höhe aufgerufen. Anstelle der GPS-Höhe kann nun die korrigierte barometrische Höhe verwendet werden. Zusätzlich kann eine Anpassung der Rauschwerte für die Höhenmessung anhand der vom Höhenfilter gelieferten Schätzfehlerkovarianzen stattfinden.
- Schließlich kann die Fehlerfilterung und Korrektur für das INS wie bereits beschrieben, fortgesetzt werden.

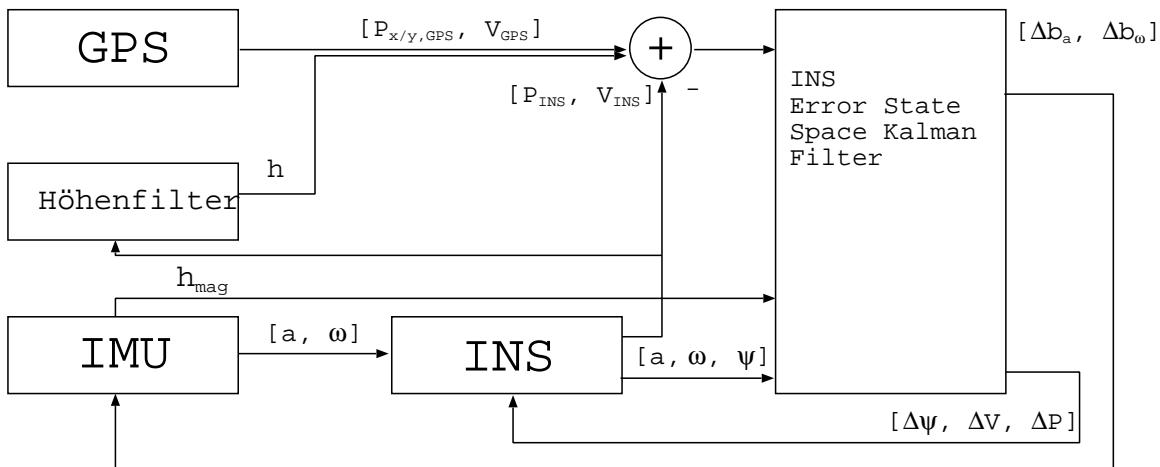


Abbildung 8.1: Kombination GPS, INS, Höhenfilter

Die Abbildung 8.1 zeigt das Blockschaltbild des gesamten Systems. Dabei ist der Block "Höhenfilter" wie im vorherigen Kapitel beschrieben, aufgebaut. Das GPS im GPS/INS-Filter wurde dabei durch GPS + Barometer ersetzt, außerdem wurde das GPS im Höhenfilter durch die INS-Navigationslösung, bevor eine Korrektur der INS-Lösung durchgeführt wurde, ersetzt.

Zum Einbau des Höhenfilters und zur Verwendung der korrigierten barometrischen Höhe anstelle der GPS-Höhe wurde zuerst das Matlab-Skript "gps_ins_demo.m" angepasst. Hinzugefügt wurde eine Tabelle mit Referenzpunkten. Zu Simulationszwecken wurde hier z. B. alle 100 Sekunden ein Referenzpunkt abgespeichert. Außerdem wurde die Initialisierung der Kovarianzen für das Systemmodell und Messmodell des Höhenfilters hinzugefügt und die Kovarianz für das Messmodell des GPS/INS-Filters entsprechend angepasst um die Varianz der Höhenmessung zu berücksichtigen.

Listing 8.1: Modifikation von gps_ins_demo.m

```

1 % Reference height generation for filtering
2 % only use every "step"th height value as reference for filtering
3 % (in this case one reference per 100 seconds)
4 step_baro = round(100/dt);
5 ref_pos = [lat_prof(step_baro:step_baro:end)*rad2deg; ...
6 lon_prof(step_baro:step_baro:end)*rad2deg; ...
7 height_prof(step_baro:step_baro:end)']; % copy every "step"th value
8
9 % extract initial values from ideal profile
10 h_init_baro = height_prof(1); % save initial height
11 t_init_baro = T_baro(1); % save initial temperature at initial height in Kelvin
12 p_init_baro = P_baro(1); % save initial pressure at initial height
13
14 % Parameters for Barometer
15 Rbaro = Rins;
16 % estimate for scale factor standard deviation using T0 = 270K
17 std_scale = std_T0_baro/270*sqrt(dt);
18 % estimate for bias standard deviation using 0.1m per Pa
19 std_bias = std_P0_baro*0.1*sqrt(dt);
20 % estimate for height measurement standard deviation using 0.1m per Pa
21 std_h = std_P_baro*0.1;
22 % estimate for scale factor measurement standard deviation using T0 = 270K
23 std_scale_t = std_T_baro/270;
24 R_baro = blkdiag(std_h^2, std_scale_t^2); % measurement noise
25 Q_baro = blkdiag(std_scale^2, std_bias^2); % process noise
26 % modify measurement noise because we will use barometric measurements for height
27 Rbaro(3,3) = (std_h)^2 + (std_bias*sqrt(step_baro))^2;
28 Rbaro(3,6) = 0;
29 Rbaro(6,3) = 0;
30
31 [q_s, v_s, x_s, llh_s, omega_ib_b_s, a_ib_b_s, st, cv] = ...
32 gps_ins(q0_b_n, x0, v0, llh0, dt, omega_ib_b, a_ib_b, h_b, llh_gps_f, vel_gps_f, ...
33 Rbaro, Q, ref_pos, est_P_baro, est_T_baro, h_init_baro, t_init_baro, ...
34 p_init_baro, R_baro, Q_baro, 1);

```

Das Matlab-Skript “gps_ins.m” wurde ebenfalls angepasst, um die barometrischen Messungen zu verarbeiten. Dazu wurden die Übergabeparameter erweitert um die Werte für die barometrische Höhenberechnung zu übergeben. Außerdem wurde ein Parameter eingeführt, der es ermöglicht, zwischen der Verarbeitung der Daten vom Barometer und der Verarbeitung von GPS Daten alleine, zu wählen. Zusätzlich wird nun der Höhenfilter aufgerufen, um eine korrigierte Höhe auf Basis der barometrischen Daten zu berechnen. Außerdem wird dem GPS/INS-Filter nun wahlweise die barometrische Höhe, anstelle der GPS-Höhe übergeben. In der hier vorgestellten Implementierung findet die Verarbeitung der barometrischen Höhe nur zusammen mit den GPS-Positionsmessungen statt. Dies kann aber leicht erweitert werden, so dass die barometrischen Daten mit der Datenrate des barometrischen Sensors in die Navigationslösung einfließen. In dem folgenden Listing werden nur die angepassten Codefragmente aufgeführt.

```

1 function [q_s, v_s, x_s, llh_s, omega_ib_b, a_ib_b, st, cv] = ...
2 gps_ins(q0_b_n, x0, v0, llh0, dt, omega_ib_b, a_ib_b, h_b, llh_gps, vel_gps, R, Q, ...
3 ref_pos, P_baro, T_baro, h_init, t_init, p_init, R_baro, Q_baro, varargin)
4 %GPS_INS do combined strapdown calculation and INS/GPS Kalman filtering
5 %
6 % function [q_s, v_s, x_s, llh_s, omega_ib_b, a_ib_b, st, cv] = ...
7 % gps_ins(q0_b_n, x0, v0, llh0, dt, omega_ib_b, a_ib_b, h_b, llh_gps, vel_gps, R, ...
8 % Q, ref_pos, P_baro, T_baro, h_init, t_init, p_init, R_baro, Q_baro, varargin)
9 %
10 % INPUTS
11 %
12 % ref_pos = reference heights for barometer "calibration"
13 % P_baro = barometric pressure

```

```
14 %      T_baro = temperature in kelvin
15 %      h_init = initial height
16 %      t_init = initial tempererature at initial height
17 %      p_init = initial pressure at initial height
18 %      R_baro = measurement noise for height filter
19 %      Q_baro = process noise for height filter
20 %      varargin = set to "1" if barometer should be used
21
22 % check if barometer should be used
23 if size(varargin) == 0
24     use_baro = 0;
25 elseif size(varargin) > 0
26     use_baro = varargin{1};
27 end
28
29 a_baro = 0;                      % initialize scale factor
30 b_baro = 0;                      % initialize bias
31 state_baro = zeros(2,1);          % initialize state for heigt filter
32 covariance_baro = zeros(2);       % initialize covariance
33
34 % added after strapdown calculation:
35 if use_baro == 1
36     %% Height calculation
37     % find reference heights at 5m radius
38     h_ref(i) = get_h_ref(ref_pos, llh_s(i,:)*[180/pi 0 0; 0 180/pi 0; 0 0 1], 5);
39     % calculate barometric height
40     [h_baro(i), h_baro_c(i), a_baro, b_baro, state_baro, covariance_baro] = ...
41         baro_height_filter(P_baro(i), T_baro(i), h_ref(i), h_init, t_init, p_init, ...
42         a_baro, b_baro, state_baro, covariance_baro, R_baro, Q_baro);
43     h_meas = h_baro_c(i);           % set height measurement to barometric height
44     %% (end) Height calculation
45 else
46     h_meas = llh_gps(i,3);        % else use GPS height
47 end
48
49 % measured values (for gps + barometer)
50 measurements = [[llh_gps(i,1:2) h_meas]' vel_gps(i,:)'] h_b(i,:')];
```

9 Auswertungen zum kombinierten Filter

In diesem Kapitel sollen Auswertungen zur erläuterten Filterstruktur vorgestellt werden. Gezeigt werden Ergebnisse einer Simulation des kompletten GPS/INS-Filters in Kombination mit dem barometrischen Höhenfilter. Außerdem soll die Auswertung eines während einer Testfahrt unter realistischen Bedingungen aufgezeichneten Datensatzes gezeigt werden.

9.1 Simulierte Daten

Als erstes soll die Funktion des GPS/INS-Filter anhand von Simulationen überprüft werden. Dazu wurden Testdaten anhand des in Kapitel 5 beschriebenen Vorgehens erzeugt. Die Daten wurden mit einer Datenrate von 50 Hz erzeugt. Die Rauschwerte für die zu erzeugenden Daten wurden, wie in den Abschnitten 3.3 und 4.2 bestimmt, eingestellt. Die verwendeten GPS-Daten wurden dabei jeweils mit und ohne Vorfilterung als Eingangsgröße für das GPS/INS-Filter übergeben und stehen jeweils mit einer Datenrate von 1 Hz zur Verfügung. Außerdem wurde das GPS/INS Filter jeweils mit und ohne barometrische Höhe getestet. Auch die barometrische Höhe wird, wie die GPS-Höhe, mit einer Datenrate von 1 Hz verwendet. Die für die Simulation eingestellten Rauschwerte sind in Tabelle 9.1 noch einmal zusammengefasst. Die Abbildungen 9.1 bis 9.7 zeigen die Ergebnisse der Simulation unter Verwendung der generierten Daten und bei Verwendung von gefilterten GPS-Daten. Die Tabelle 9.2 fasst die Ergebnisse der Simulationen noch einmal zusammen. Die Abbildungen der Simulation ohne Vorfilterung der GPS-Daten unterscheiden sich nicht wesentlich von den gezeigten. Darauf folgt die Kombination aus GPS und INS ohne Barometer. Es ist zu erkennen, dass das GPS/INS-Filter bereits in der Lage ist, eine bessere Höhenschätzung als das GPS alleine zu liefern. Zusätzlich stehen hier die Höheninformationen nun mit

Betrachtung der Höhenfehler

Abbildung 9.1 zeigt den Vergleich unterschiedlicher Höhen. Die ideale Höhe ist dabei durch die gelbe Linie als Referenz gegeben. Deutlich ist die unterschiedliche Qualität der verschiedenen Höhen zu erkennen. Die GPS-Höhe hat dabei die höchste Varianz. Zu beachten ist hier, dass die GPS-Höhe nur mit einer Datenrate von 1 Hz zur Verfügung steht. Darauf folgt die Kombination aus GPS und INS ohne Barometer. Es ist zu erkennen, dass das GPS/INS-Filter bereits in der Lage ist, eine bessere Höhenschätzung als das GPS alleine zu liefern. Zusätzlich stehen hier die Höheninformationen nun mit

Sensor - Größe	Wert
INS - Abtastintervall	$\Delta t = 0.02 \text{ s}$
Beschleunigung - inhärentes Rauschen (Leistungsdichte)	$\sigma_a = 0.5 \cdot 10^{-3} \frac{\text{m}}{\sqrt{\text{s}}}$
Beschleunigung - Biasrauschen (Leistungsdichte)	$\sigma_{ba} = 0.3 \cdot 10^{-3} \frac{\text{m}^2}{\sqrt{\text{s}}}$
Drehrate - inhärentes Rauschen (Leistungsdichte)	$\sigma_\omega = 0.014 \frac{\text{°}}{\sqrt{\text{s}}}$
Drehrate - Biasrauschen (Leistungsdichte)	$\sigma_{bw} = 0.017 \frac{\text{°}}{\sqrt{\text{s}}}$
Magnetometer - inhärentes Rauschen (std. Abweichung)	$\sigma_{mag} = 110 \mu\text{T}$
Barometer - Abtastintervall	$\Delta t_b = 1 \text{ s}$
Druck - inhärentes Rauschen (std. Abweichung)	$\sigma_P = 10 \text{ Pa}$
Temperatur - inhärentes Rauschen (std. Abweichung)	$\sigma_T = 0.1 \text{ K}$
Referenztemperaturänderung (Leistungsdichte)	$\sigma_{T_0} = 0.004 \frac{\text{K}}{\sqrt{\text{s}}}$
Referenzdruckänderung (Leistungsdichte)	$\sigma_{P_0} = 0.4 \frac{\text{Pa}}{\sqrt{\text{s}}}$
Korrigierte barometrische Höhe (Varianz)	$\sigma_{p,d} = 1.5 \text{ m}$
GPS - Abtastintervall	$\Delta t_{GPS} = 1 \text{ s}$
GPS - Rauschen (Position ungefiltert)	$\sigma_{p,n/e} = 10 \text{ m}$
GPS - Rauschen (Höhe ungefiltert)	$\sigma_{p,d} = 20 \text{ m}$
GPS - Rauschen (Horizontalgeschwindigkeit ungefiltert)	$\sigma_{v,n/e} = 0.5 \frac{\text{m}}{\text{s}}$
GPS - Rauschen (Vertikalgeschwindigkeit ungefiltert)	$\sigma_{v,d} = 15 \frac{\text{m}}{\text{s}}$
GPS - Rauschen (Position gefiltert)	$\sigma_{pf,n/e} = 3 \text{ m}$
GPS - Rauschen (Höhe gefiltert)	$\sigma_{pf,d} = 10 \text{ m}$
GPS - Rauschen (Horizontalgeschwindigkeit gefiltert)	$\sigma_{vf,n/e} = 0.5 \frac{\text{m}}{\text{s}}$
GPS - Rauschen (Vertikalgeschwindigkeit gefiltert)	$\sigma_{vf,d} = 2 \frac{\text{m}}{\text{s}}$

Tabelle 9.1: Parameter für die Datenerzeugung

einer Datenrate von 50 Hz zur Verfügung. Dies stellt also bereits eine deutliche Verbesserung dar. Bei der unkorrigierten barometrischen Höhe ist zu erkennen, dass die Abweichung zur Referenz immer größer wird. Diese Effekt wird durch die Änderung der Referenzwerte für die barometrische Höhenberechnung verursacht und konnte bereits in Abschnitt 7.5 beobachtet werden. Wie ebenfalls in Abschnitt 7.5 gezeigt werden konnte, zeigt sich auch hier, dass bei der korrigierten Höhe die Fehler, welche durch eine Änderung der Referenzwerte für die barometrische Höhenberechnung verursacht werden, korrigiert werden. Dies geschieht durch die Verwendung des Höhenfilters. Abschließend ist zu erkennen, dass die Kombination aus GPS/INS und korrigierter barometrischer Höhe das beste Ergebnis liefert. Die so berechnete Höhe wird durch die schwarze Linie dargestellt. Aufgrund ihrer geringen Abweichungen zur Referenz wird sie fast vollständig von dieser bedeckt.

Die Abbildungen 9.2 und 9.3 zeigen jeweils die absolute Abweichung zur Referenz sowie die Verteilungsfunktion der Abweichungen. Auch hier ist in absteigender Richtung für den Höhenfehler die Reihenfolge GPS, GPS/INS, korrigierte Höhe und GPS/INS mit korrigierter Höhe zu erkennen. Der Fehler der unkorrigierten barometrische Höhe liegt hier zwar zwischen GPS/INS und korrigierter Höhe dies lässt aber sich nicht verallgemeinern, da die Abweichung auch größer werden kann.

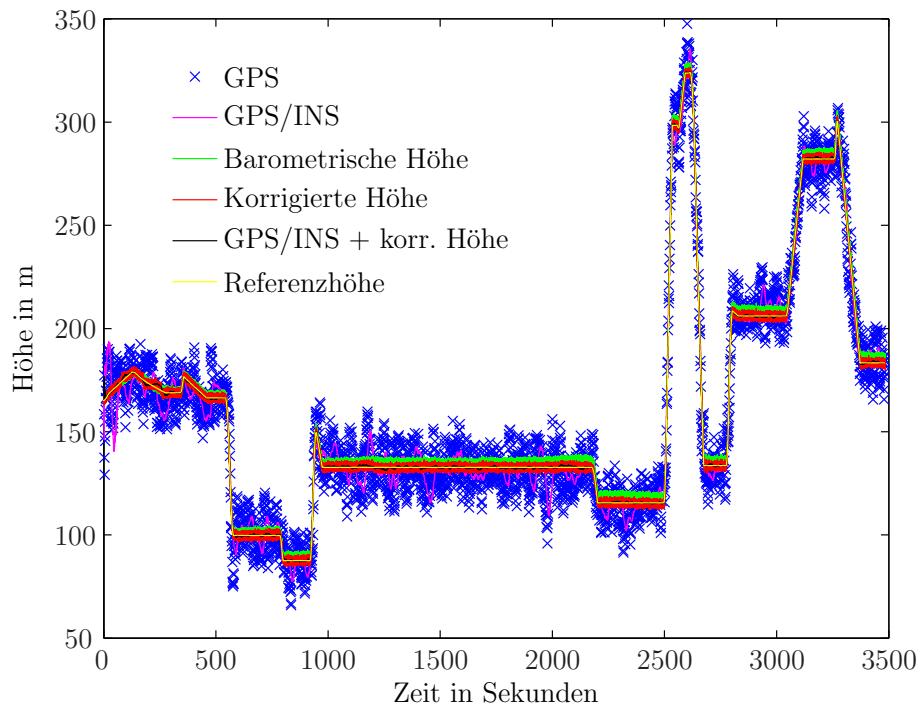


Abbildung 9.1: Vergleich unterschiedlicher Höhen

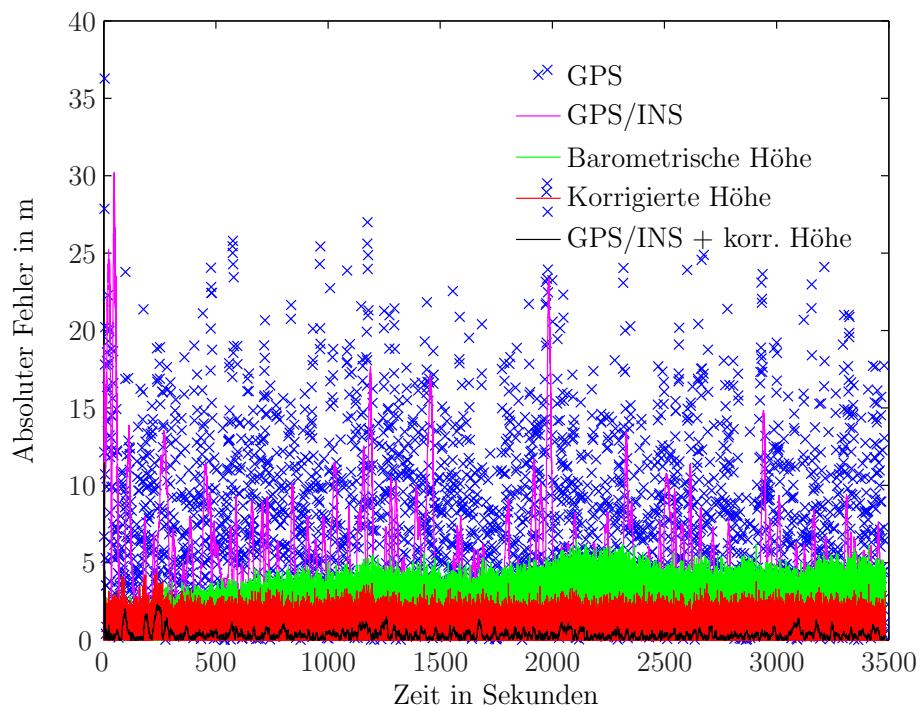


Abbildung 9.2: Absolute Fehler verschiedener Höhen

In der Abbildung 9.3 ist die kumulative Verteilungsfunktion des Höhenfehlers abgebildet. Es ist zu erkennen, dass der Fehler in 90 % aller Fälle bei dem kombinierten Filter unter ca. 1 m liegt, während er im Vergleich bei GPS bei etwa 15 m liegt.

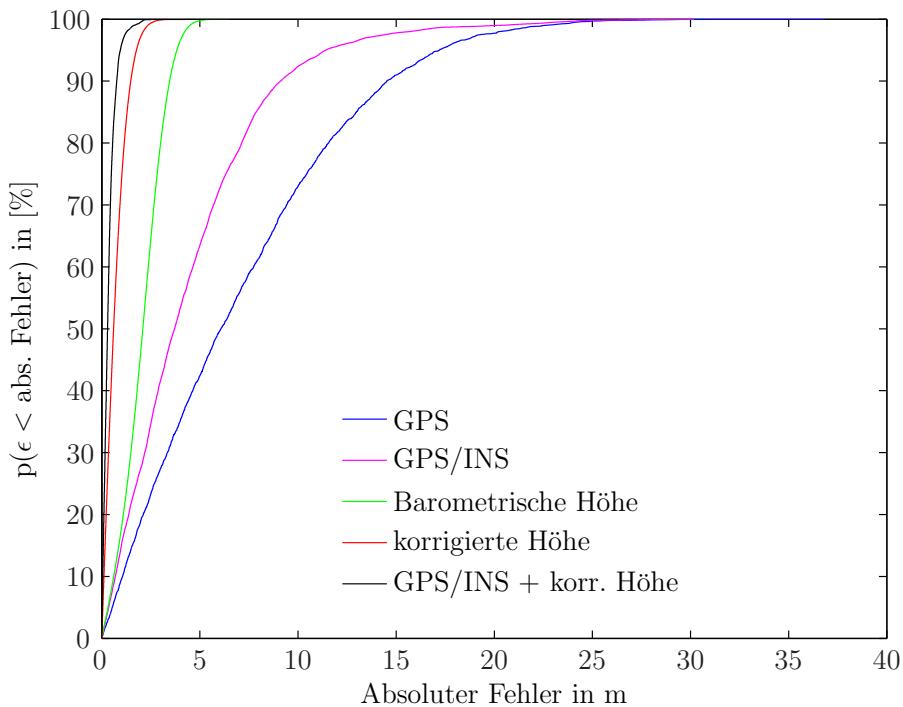


Abbildung 9.3: Verteilung der Höhenfehler

Betrachtung der Positionsfehler

Die Abbildungen 9.4 und 9.5 zeigen jeweils den Betrag der Abweichung zur jeweiligen idealen Position im dreidimensionalen Raum sowie die Verteilungsfunktion der Abweichungen. Auch hier ist in absteigender Richtung für den Positionsfehler die Reihenfolge GPS, GPS/INS und GPS/INS mit korrigierter Höhe zu erkennen. Wie sich in Tabelle 9.2 zeigt, ist diese Verbesserung lediglich auf die Verbesserung der Höheninformationen zurückzuführen. Der Grund hierfür dürfte in der Anwendung von zu optimistischen Annahmen für die Vorfilterung der GPS-Daten liegen. In diesem Fall ist das INS/GPS-Filter also anscheinend nur in der Lage die Höheninformationen zu verbessern sowie Positionsdaten mit 50 Hz, bei gleicher Qualität wie GPS, zu liefern. Auch wenn dieses Ergebnis noch nicht für die Verwendung des INS spricht, bleibt zu bedenken, dass hier bereits eine Genauigkeit angenommen wird, wie sie GPS normalerweise nur unter optimalen Bedingungen erreicht. Die Simulation des GPS/INS ohne vorherige Filterung der GPS-Daten zeigt außerdem auch, dass das INS durchaus in der Lage ist die Positionsschätzung zu verbessern, wie Tabelle 9.2 zeigt. Mit etwa 3 Meter schient lediglich in diesem Fall die untere Grenze der Genauigkeit des INS, bei Stützung durch GPS mit 1 Hz, erreicht zu sein. Außerdem wird sich bei der Untersuchung der realen Daten zeigen, dass das INS, in Kombination mit dem Barometer, in der Lage ist einen

GPS-Ausfall von mehreren Sekunden, bei hoher Dynamik, zu überbrücken.

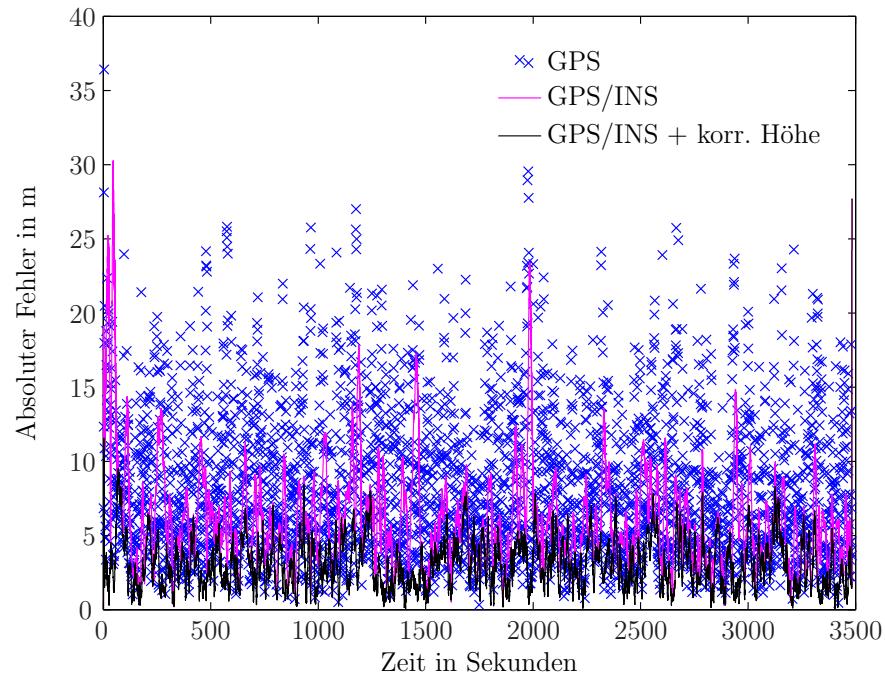


Abbildung 9.4: Absolute Positionsfehler

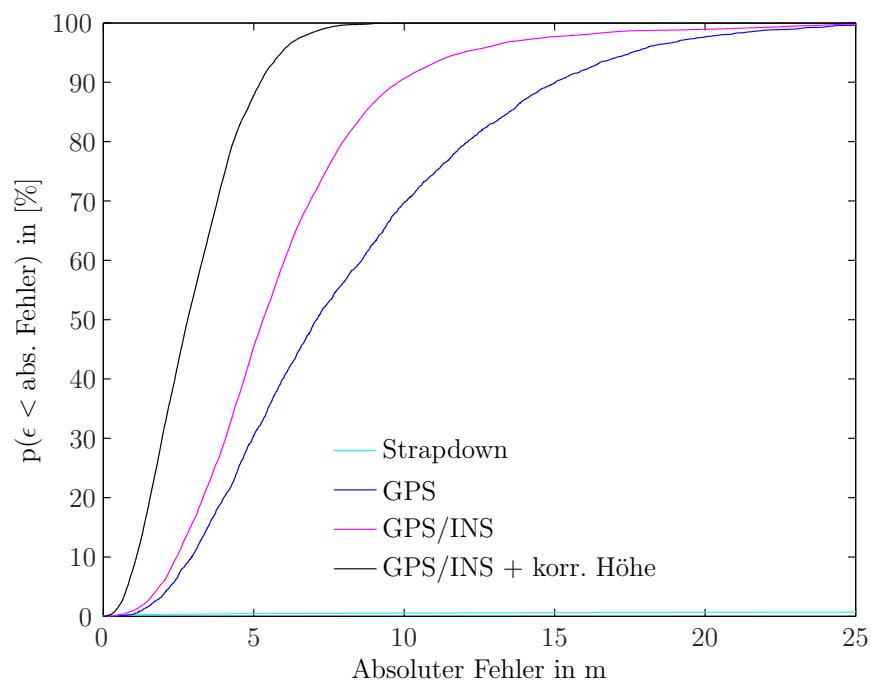


Abbildung 9.5: Verteilung der Positionsfehler

In der Abbildung 9.5 ist die kumulative Verteilungsfunktion des Positionsfehlers abgebildet. Es ist zu erkennen, dass der Fehler in 90 % aller Fälle bei dem kombinierten Filter bei ca. 5 m liegt, während er im Vergleich bei GPS bei etwa 15 m liegt.

Betrachtung der Geschwindigkeitsfehler

Die Abbildungen 9.6 und 9.7 zeigen den Betrag der Abweichung zur jeweiligen idealen Geschwindigkeit im dreidimensionalen Raum sowie die Verteilungsfunktion der Abweichungen. Wie schon bereits bei den Höheninformationen und Positionsinformationen, zeigt sich auch hier für die Geschwindigkeitsfehler in Absteigender Richtung die Reihenfolge GPS, GPS/INS sowie GPS/INS mit Barometer. Im Gegensatz zu den Positionsinformationen führt hier das GPS/INS allerdings zu einer Verbesserung der Geschwindigkeit in allen drei Komponenten, wie Tabelle 9.2 zeigt. Die Verwendung der korrigierten barometrischen Höhe führt ebenfalls zu einer Verbesserung der Geschwindigkeit gegenüber dem GPS/INS alleine. Dies ist insofern nicht ungewöhnlich, da die IMU Beschleunigungen liefert aus denen die Geschwindigkeiten geschätzt werden können. Eine Verbesserung der Geschwindigkeitsinformationen ist somit durch einfache Integration direkter möglich, im Gegensatz zur zweifachen Integration bei den Positionsinformationen. Analog gilt dies für die Verbesserung der Höhe durch das Barometer. Das Barometer liefert direkt Informationen über die Höhe. Zur Verbesserung der Geschwindigkeit kann diese Information zusätzlich genutzt werden, da ein geringere Schwankung in der Höhe auch einen geringeren Fehler in der Geschwindigkeit zur Folge hat.

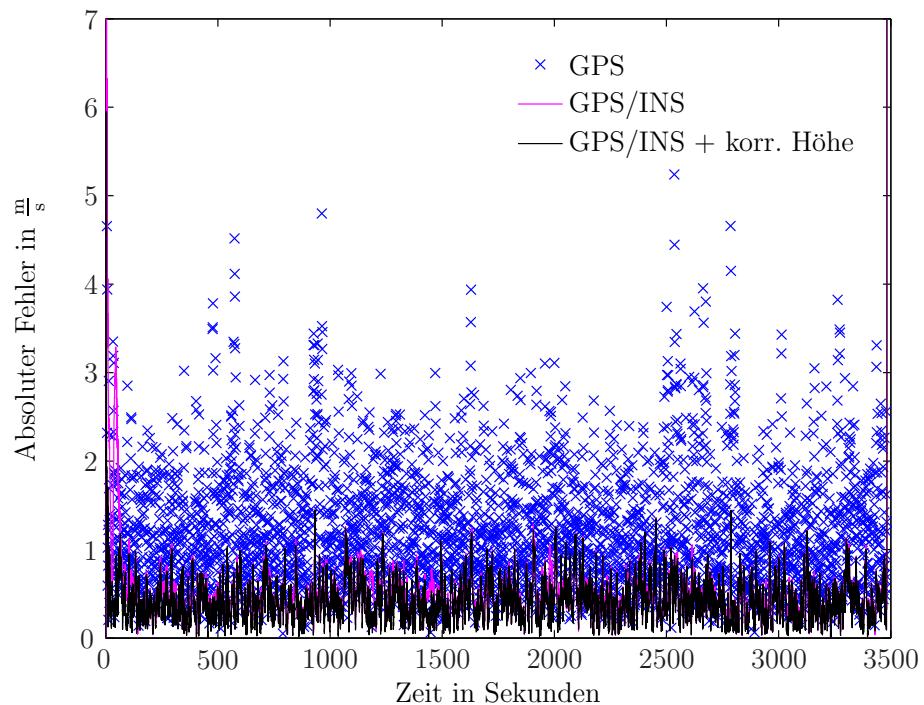


Abbildung 9.6: Absolute Geschwindigkeitsfehler

In der Abbildung 9.7 ist die kumulative Verteilungsfunktion des Geschwindigkeitsfehlers abgebildet. Es ist zu erkennen, dass der Fehler in 90 % aller Fälle bei dem kombinierten Filter bei ca. $0.8 \frac{\text{m}}{\text{s}}$ liegt, während er im Vergleich bei GPS bei etwa $2 \frac{\text{m}}{\text{s}}$ liegt.

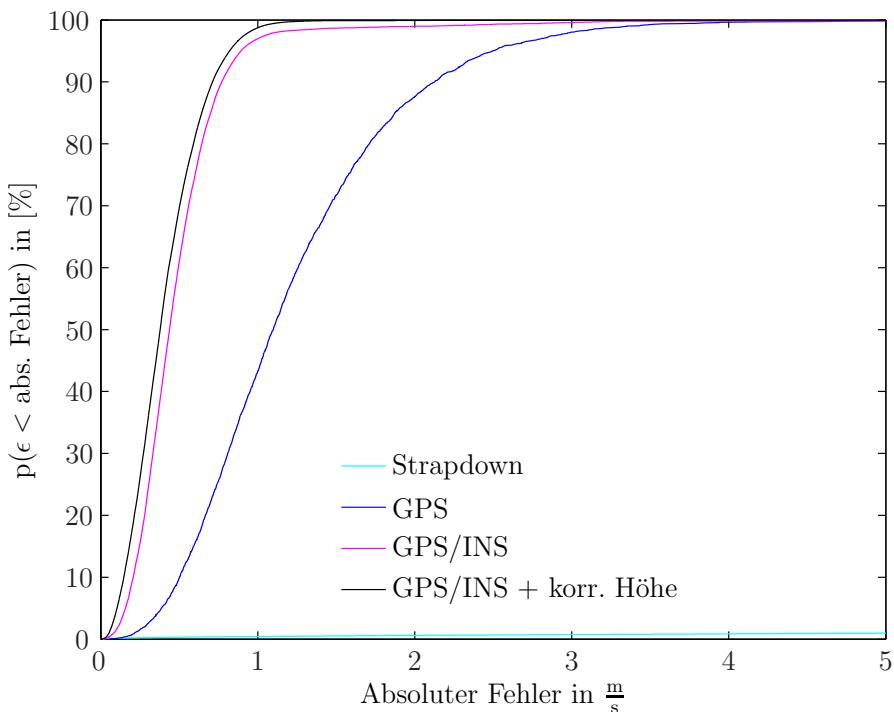


Abbildung 9.7: Verteilung der Geschwindigkeitsfehler

Zusammenfassung

Die Simulation hat gezeigt, dass die Kombination aus GPS, INS und Barometer zu einem verbesserten Ergebnis für die Höhenschätzung führt. Außerdem konnte gezeigt werden, dass das GPS/INS-Filter in der Lage ist die Höhenschätzung zusätzlich zum Höhenfilter zu verbessern. Des weiteren zeigt sich, dass das GPS/INS-Filter in der Lage ist, die Geschwindigkeitsschätzung in allen drei Raumrichtungen zu verbessern. Die Feststellung, dass das GPS/INS-Filter nicht in der Lage ist in der Ebene eine bessere Positionsschätzung zu liefern als das hier verwendete GPS-Filter, sollte weiter untersucht werden. Der Grund hierfür dürfte in einer zu optimistischen und evtl. unrealistischen Modellierung und Filterung der GPS-Daten liegen. Untersuchungen mit ungefilterten GPS-Daten haben gezeigt, dass das GPS/INS-Filter in der Lage ist, die Positions- schätzung in allen drei Koordinatenrichtungen zu verbessern. Hier empfiehlt sich für weitere Untersuchungen der Einsatz entsprechender Toolboxen, wie z. B. [GPSAT] und [GPFLT]), zur Modellierung realistischerer GPS-Daten.

Die Tabelle 9.2 fasst die Ergebnisse der Simulationen jeweils für die Durchgänge mit vorgefilterten GPS-Daten und mit ungefilterten GPS-Daten noch einmal zusammen. Aufgelistet sind jeweils die Standardabweichungen der Fehler der berechneten Positions-

und Geschwindigkeitswerte nach der Korrektur durch das GPS/INS-Filter. Der Index p bezeichnet die Positionsverste und der Index v die Geschwindigkeitswerte. Die Erweiterung des Index um n , e oder d bezieht sich auf die einzelnen Komponenten in Nord, Ost und Unten-Richtung.

	Größe	GPS	GPS/INS	GPS/INS + Barom.	korr. Barom.
GPS gefiltert	σ_h [m]	8.914334	5.590429	0.413503	0.876455
	σ_p [m]	9.557862	6.542884	3.419117	
	$\sigma_{p,n}$ [m]	2.479327	2.423121	2.416840	
	$\sigma_{p,e}$ [m]	2.395896	2.384307	2.382910	
	$\sigma_{p,d}$ [m]	8.914334	5.590429	0.413503	
	σ_v [$\frac{\text{m}}{\text{s}}$]	1.818164	0.514322	0.470757	
	$\sigma_{v,n}$ [$\frac{\text{m}}{\text{s}}$]	0.486763	0.331830	0.330502	
	$\sigma_{v,e}$ [$\frac{\text{m}}{\text{s}}$]	0.758115	0.332936	0.332887	
	$\sigma_{v,d}$ [$\frac{\text{m}}{\text{s}}$]	1.579254	0.208734	0.039582	
GPS ungefiltert	σ_h [m]	20.170197	4.401432	0.606866	0.884195
	σ_p [m]	24.647010	6.226572	4.435353	
	$\sigma_{p,n}$ [m]	9.952425	3.052974	3.041692	
	$\sigma_{p,e}$ [m]	10.079063	3.174421	3.170518	
	$\sigma_{p,d}$ [m]	20.170197	4.401432	0.606866	
	σ_v [$\frac{\text{m}}{\text{s}}$]	15.280874	0.518507	0.502306	
	$\sigma_{v,n}$ [$\frac{\text{m}}{\text{s}}$]	0.521060	0.346610	0.345432	
	$\sigma_{v,e}$ [$\frac{\text{m}}{\text{s}}$]	0.832108	0.356959	0.357216	
	$\sigma_{v,d}$ [$\frac{\text{m}}{\text{s}}$]	15.249302	0.145918	0.073383	

Tabelle 9.2: Ergebnisse der Simulationen

9.2 Reale Daten

Nachdem die Funktion des GPS/INS-Filters in Kombination mit dem Höhenfilter durch die Simulationen gezeigt werden konnte, wurde ein Test mit realen Daten durchgeführt. Dazu wurde während einer Testfahrt mit dem Auto, wie bei dem Test des Höhenfilters in Abschnitt 7.5, ein Datensatz mit dem NT-Datalogger angezeichnet. Der Datensatz besteht dabei aus Positions- und Geschwindigkeitsinformationen vom GPS-Empfänger, Beschleunigungswerten und Drehraten von der IMU sowie Druck- und Temperaturmessungen vom Barometer. Die Daten des Magnetometers wurden nicht genutzt, da ihre Qualität nicht sichergestellt werden konnte. Die Aufzeichnung der IMU-Daten wurde mit 50 Hz durchgeführt, die Daten des Barometers stehen mit einer Datenrate von 10 Hz zur Verfügung und die Daten des u-blox GPS-Empfängers mit 2 Hz. Die jeweiligen Positionen und Höhen der Referenzpunkte wurden aus der Deutschen Grundkarte [TIMOL] entnommen. Die Rauschwerte wurden dabei analog zu Tabelle 9.1, wie in der Simulation eingestellt. Ein Unterschied zu der Simulation in 9.1 ist hier, dass die barometrischen Daten mit 10 Hz zur Verfügung stehen und während der Fehlerfiltrierung immer verwendet werden, auch wenn keine GPS-Werte zur Verfügung stehen. Es

wird sich zeigen, dass dieses Vorgehen die Positionsschätzung durch das INS während eines GPS-Ausfalls deutlich verbessert. GPS-Ausfälle wurden während der Auswertung durch Nicht-Verwendung der GPS-Werte zu den entsprechenden Zeitpunkten, simuliert.

Teststrecke

Um möglichst viele Aspekte der Filterung zu testen, wurde bei der Auswahl der Teststrecke darauf geachtet, verschiedenste Fahrsituationen abzudecken. Hierzu gehören insbesondere Kurvenfahrten und große Höhenunterschiede. Als Teststrecke wurde dabei eine Fahrt Richtung Altenbeklen mit einem Abstecher auf das Eggegebirge ausgewählt. Die Abbildung 9.8 zeigt die Teststrecke. Die Positionen der Referenzpunkte sind durch blaue Punkte gekennzeichnet.

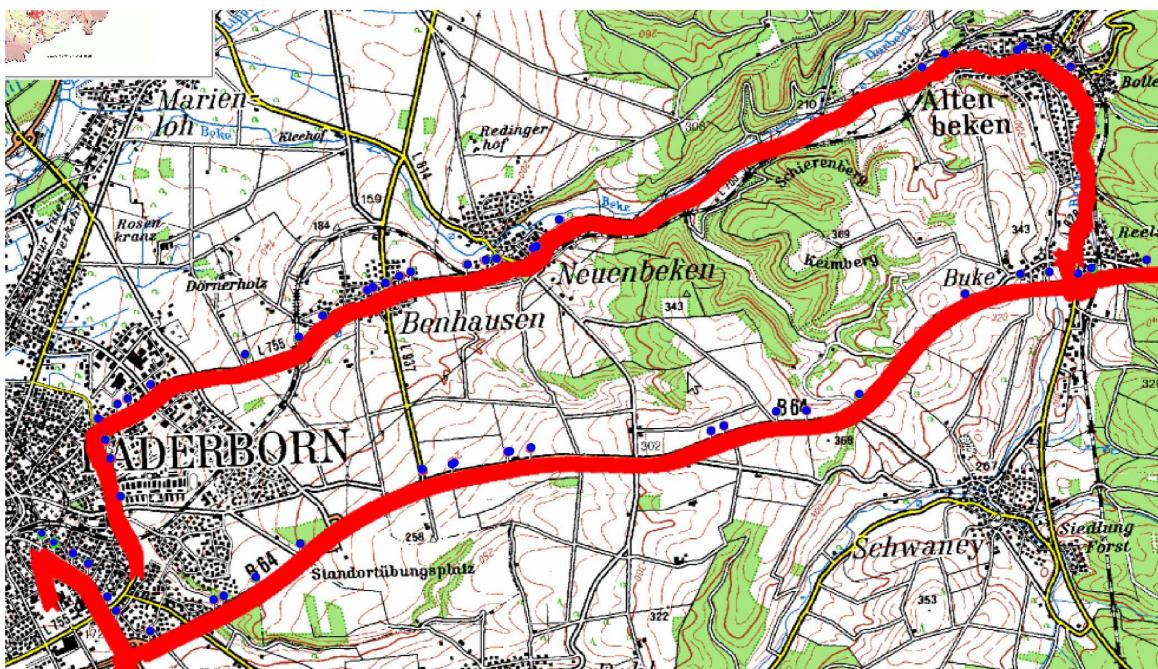


Abbildung 9.8: Teststrecke für die Datenaufzeichnung

Höhenprofil

Die Abbildung 9.9 zeigt das Höhenprofil der Teststrecke. Abgebildet sind dabei die verschiedenen aufgenommenen Höhen. Es ist darauf zu achten, dass hier die Höhe über dem Ellipsoid abgebildet ist. Der Unterschied zur Meereshöhe beträgt in diesem Fall konstant +45.5 Meter. Es ist zu erkennen, dass die barometrische Höhe von der korrigierten barometrischen Höhe abweicht. Dies ist durch die veränderten Referenzparameter zu erklären. Der Höhenfilter ist allerdings in der Lage, die dadurch verursachten Fehler zu korrigieren. Die GPS-Höhe wird in der Abbildung von den anderen Höhen überdeckt. Die Abbildung 9.10 zeigt daher einen Ausschnitt, in dem deutlich starke Schwankungen in der GPS-Höhe, im Vergleich zur barometrischen Höhe, zu erkennen sind. Die Unterschiede zwischen GPS und Barometer betragen dabei etwa 5 bis 8 Meter. Auch hier zeigt sich, dass der Höhenfilter in der Lage ist, bessere Höheninformationen zu liefern.

Zusätzlich ist zu erkennen, dass die Kombination aus GPS/INS und Höhenfilter eine weitere Verbesserung bringt.

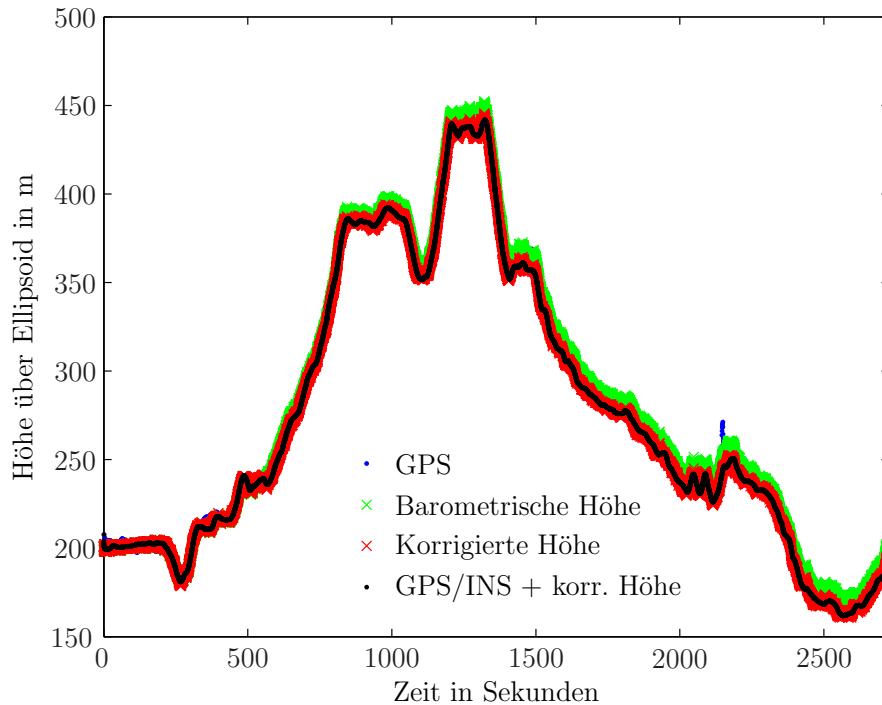


Abbildung 9.9: Höhen über dem Ellipsoid

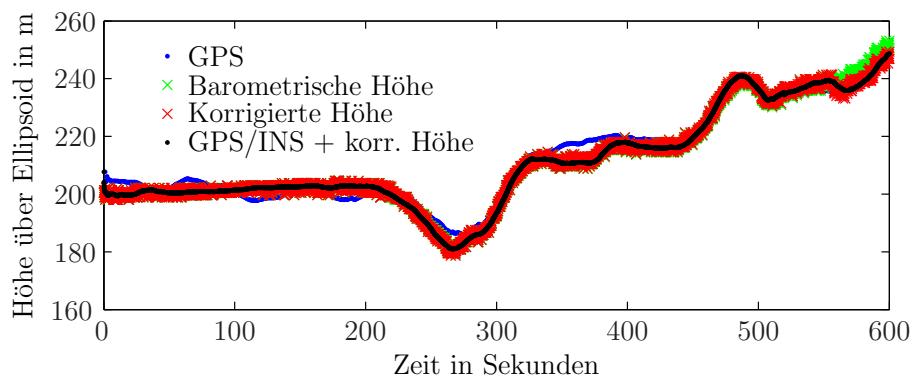


Abbildung 9.10: Ausschnitt aus dem Höhenprofil

Als nächstes sollen die vom Höhenfilter geschätzten Parameter betrachtet werden. In Abbildung 9.11 sind der geschätzte Bias, der geschätzte Skalierungsfaktor und zum Vergleich ein theoretischer Skalierungsfaktor abgebildet. Wie schon in Abschnitt 7.5 gezeigt werden konnte, ist das Höhenfilter auch hier in der Lage, die Fehler zu schätzen. Die Variation des Bias von etwa +/- 1 Meter zeigt auch hier, dass die korrigierte barometrische Höhe einen guten Schätzwert liefert, auch wenn sich gegen Ende der

Testfahrt etwas stärkere Variationen von +/- 2 Meter zeigen. Dies ist allerdings immer noch kleiner als die Variationen von etwa +/- 5 Meter in der GPS-Höhe. Der Grund für die Schwankungen liegt in einer schlechten Synchronisation der aufgezeichneten Daten, so dass ein Referenzpunkt zu früh oder zu spät gefunden und verwendet wird.

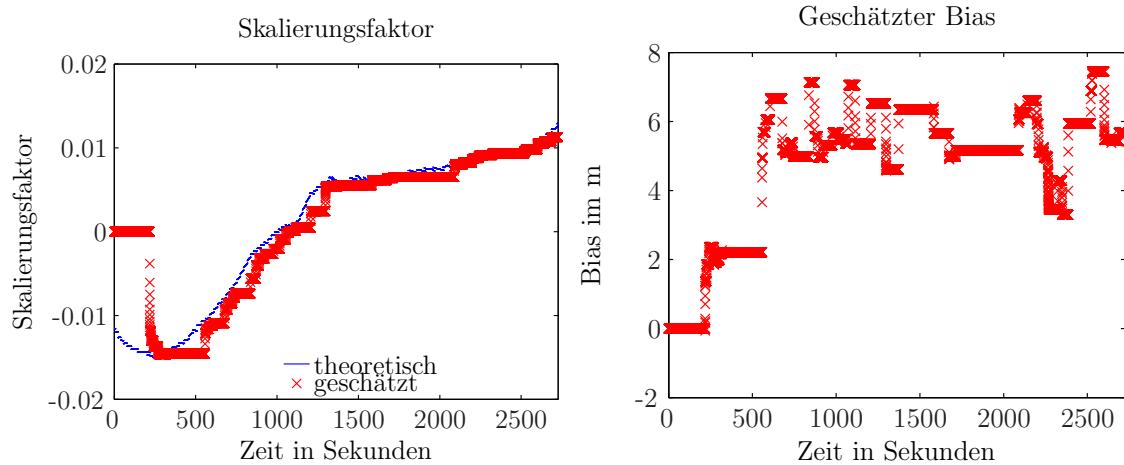


Abbildung 9.11: Skalierungsfaktor und Bias

Schätzung der Fehler der IMU

Neben der Fehlerschätzung für das Barometer ist auch die Schätzung der Fehler der inertialen Messeinheit ein wichtiger Aspekt des GPS/INS-Filters. Das GPS/INS-Filter sollte in der Lage sein, die Biaswerte der Beschleunigungssensoren und Drehratensensoren zu schätzen, damit diese korrigiert werden können. Am Beispiel der Biaswerte der Drehratensensoren soll dies hier gezeigt werden. Die Abbildung 9.12 zeigt die geschätzten Biaswerte für alle drei Achsen der IMU.

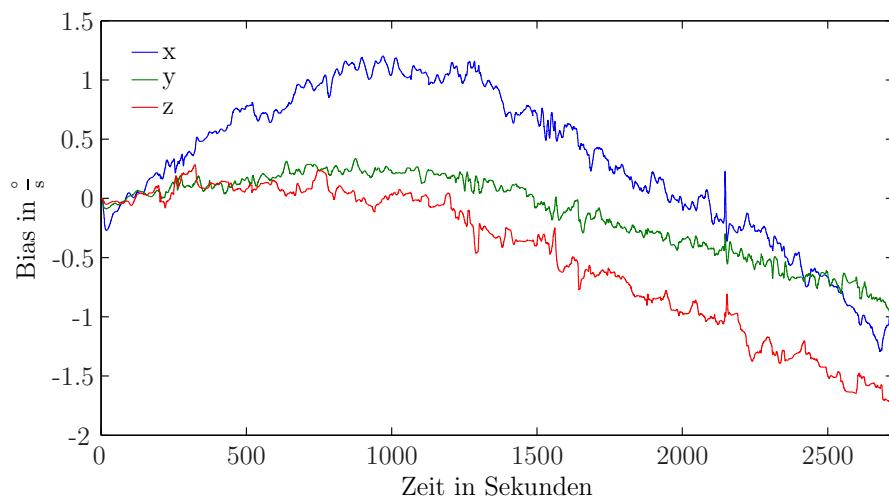


Abbildung 9.12: Geschätzter Bias in den Drehraten

ten Biaswerte für alle drei Achsen der IMU. Es ist zu erkennen, dass der GPS/INS-Filter

in der Lage ist, Biaswerte zu schätzen. Zur Überprüfung, ob die Schätzung korrekt ist, werden die gemessenen Drehraten einmal vor und einmal nach der Korrektur des Bias betrachtet. Dies ist in Abbildung 9.13 dargestellt. Es zeigt sich deutlich, dass die

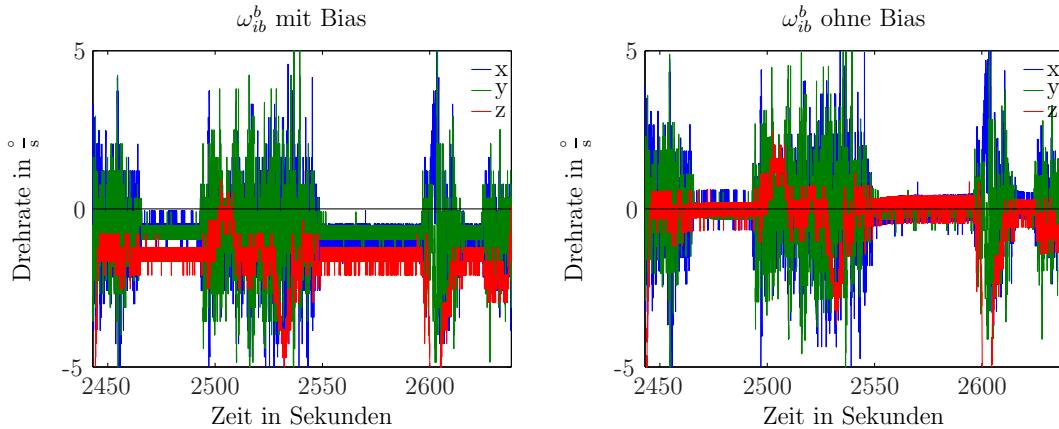


Abbildung 9.13: Drehraten mit und ohne Bias

Drehraten vor der Korrektur einen von Null abweichenden Wert haben, obwohl dieser Null sein sollte. Nach der Korrektur ist diese Abweichung beseitigt. Zur besseren Darstellung wurde hier absichtlich ein Abschnitt gewählt, in dem der Sensor in Ruhe war. Natürlich ist das GPS/INS-Filter auch in der Lage, den Bias zu schätzen wenn der Sensor nicht in Ruhe ist. Dasselbe Verhalten zeigt sich auch bei der Schätzung der Biaswerte für die Beschleunigungssensoren, wie in Abbildung 9.14 dargestellt, wobei bei den Biasen der Beschleunigungswerte eine größere Variabilität erkennbar ist. Dies lässt

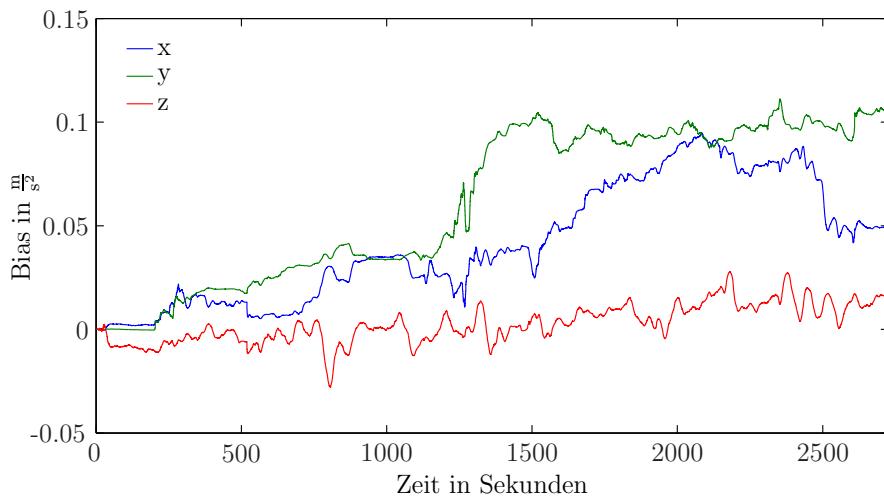


Abbildung 9.14: Geschätzter Bias in den Beschleunigungen

sich damit begründen, dass eine hohe Synchronität zwischen den gemessenen Beschleunigungen und daraus berechneten Geschwindigkeiten sowie den vom GPS-Empfänger

gemessenen Geschwindigkeiten nötig ist. Dies ist aufgrund von Verzögerungen im GPS-Empfänger nicht immer gegeben und kann daher zu Schwankungen im geschätzten Bias führen, auch wenn versucht wurde, die Verzögerungen bei der Auswertung möglichst zu berücksichtigen.

Eine interessante Anmerkung sei zu den Biasverläufen noch gemacht: vergleicht man einmal die Abbildungen der Biasverläufe 9.14 und 9.12 mit dem Verlauf der gemessenen Temperatur bzw. dem Skalierungsfaktor in Abbildung 9.11, so scheinen die Verläufe recht ähnlich. Dies kann hier Zufall sein aber auch auf einen temperaturabhängigen Bias schließen lassen, wie er in Abschnitt 4.2 zwar eingeführt aber weiter nicht untersucht wurde. Für die Aufnahmen dieser Testfahrt wurde der MotionNode verwendet. Da dem Fachgebiet Nachrichtentechnik mittlerweile eine neuere IMU von Xsens [XSMTI] zur Verfügung steht, welche unter anderem auch eine temperaturabhängige Kompen-sation des Bias durchführt, sollte diese für weitere Untersuchungen verwendet werden. Alternativ könnte auch versucht werden, eine bessere Kalibrierung des MotionNode durchzuführen. Aufgrund des hohen Aufwandes wäre dies aber relativ aufwendig.

Verhalten bei GPS-Ausfall mit Verfügbarkeit von barometrischen Daten

Ein Hauptziel des Einsatzes von Inertialsensoren ist die Möglichkeit, einen GPS-Ausfall zu überbrücken oder ganz ohne GPS eine Navigationslösung zu berechnen. Anhand der, während der Testfahrt aufgezeichneten Daten, soll getestet werden, wie sich die Kombination aus GPS/INS und Höhenfilter während eines GPS-Ausfalls verhält. Ein wichtiger Punkt ist hier, dass die vom Höhenfilter berechnete Höhe auch während des GPS-Ausfalls weiter zur Verfügung steht und während der ganzen Auswertung mit voller Datenrate von 10 Hz genutzt wird. Hierzu wurde das GPS/INS-Filter angepasst, um

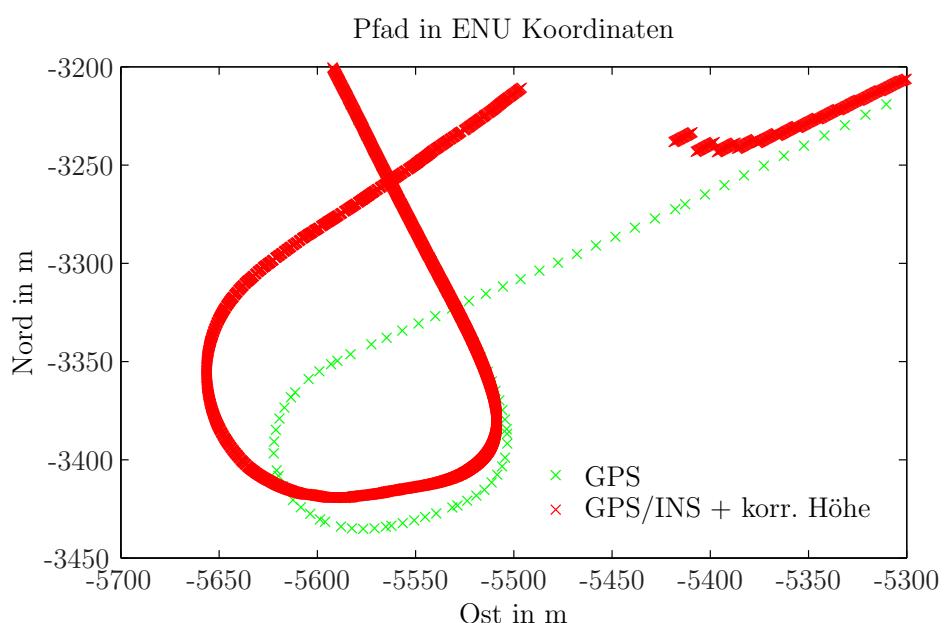


Abbildung 9.15: GPS-Ausfall 1 (mit Barometer)

auch nur anhand eines Höhenwertes eine Fehlerbestimmung durchzuführen. Um dies zu erreichen, ist es lediglich nötig, aus Gleichung (6.65), dem Messmodell des GPS/INS-Filters, die Zeile für den Höhenfehler separat auszuwerten. Ein GPS-Ausfall wurde simuliert indem die entsprechenden GPS-Werte vor der Auswertung entfernt wurden. Die Abbildungen 9.15 bis 9.17 zeigen die Ergebnisse für die 2D-Positionskoordinaten in der Ebene. Die grünen Punkte stellen die vom GPS-Empfänger aufgezeichneten Ko-

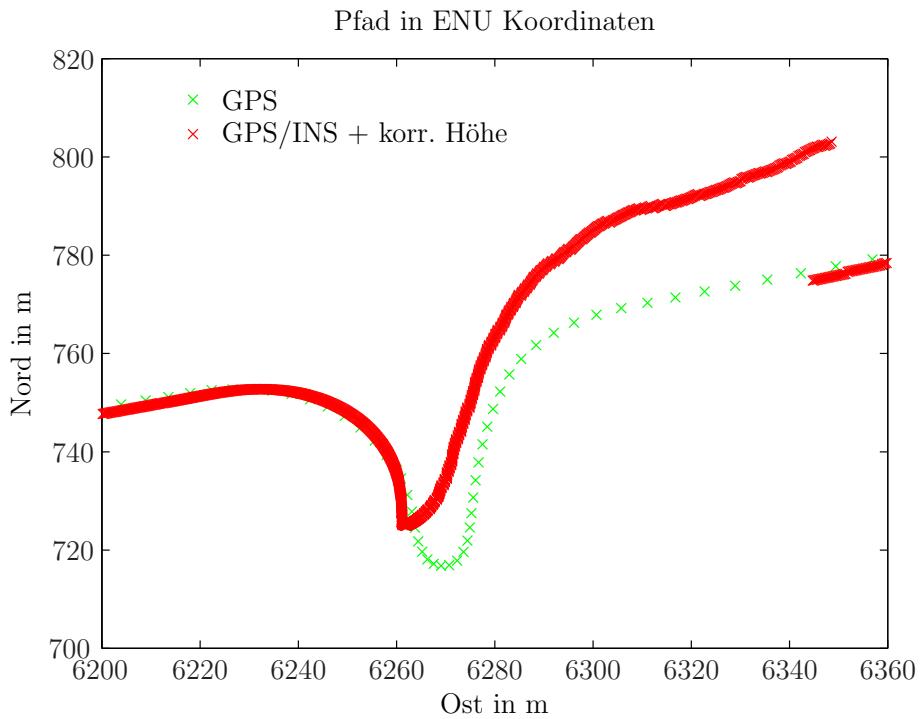


Abbildung 9.16: GPS-Ausfall 2 (mit Barometer)

ordinaten dar und die roten Punkte die vom INS, nach der Korrektur, berechneten Koordinaten. Die Dauer der GPS-Ausfälle betragen 35 Sekunden (Abb. 9.15), 23 Sekunden (Abb. 9.16) und 82.5 Sekunden (Abb. 9.17). Die Ausfälle beginnen jeweils etwas vor den Übergängen vom geraden Stück in die Kurve. Es ist deutlich zu erkennen, dass das INS in der Lage ist, die gefahrene Trajektorie zu schätzen. Allerdings ist auch zu erkennen, dass die Fehler in der Schätzung mit der Zeit anwachsen. Der Grund für die anwachsenden Fehler dürfte in der Qualität der Datenaufzeichnung und der Qualität der eingesetzten IMU liegen. Gibt es Abweichungen in der Synchronität der Daten, verschlechtert sich auch die Fehlerkennung und Korrektur durch das GPS/INS-Filter. Weist die IMU stark zeitvariante Fehler auf, was hier der Fall ist, verschlechtert sich ohne Stützung auch die Navigationslösung die auf Basis der Daten der IMU berechnet wird. Auch wenn die Abweichungen mit der Zeit relativ groß werden, ist das Ergebnis trotzdem positiv zu Werten. Das INS war in der Lage der Richtung der Trajektorie zu folgen. Auch das Stoppen vor einer Ampel zum Ende des GPS-Ausfalls in Abbildung 9.17 ist in der Navigationslösung des INS näherungsweise zu erkennen. Verbesserungen dürften durch eine bessere Datenaufzeichnung und stabilere Sensoren möglich sein. Bei der Betrachtung der Geschwindigkeiten ergeben sich ähnliche Bilder. Daher werden

diese hier nicht gesondert aufgeführt.

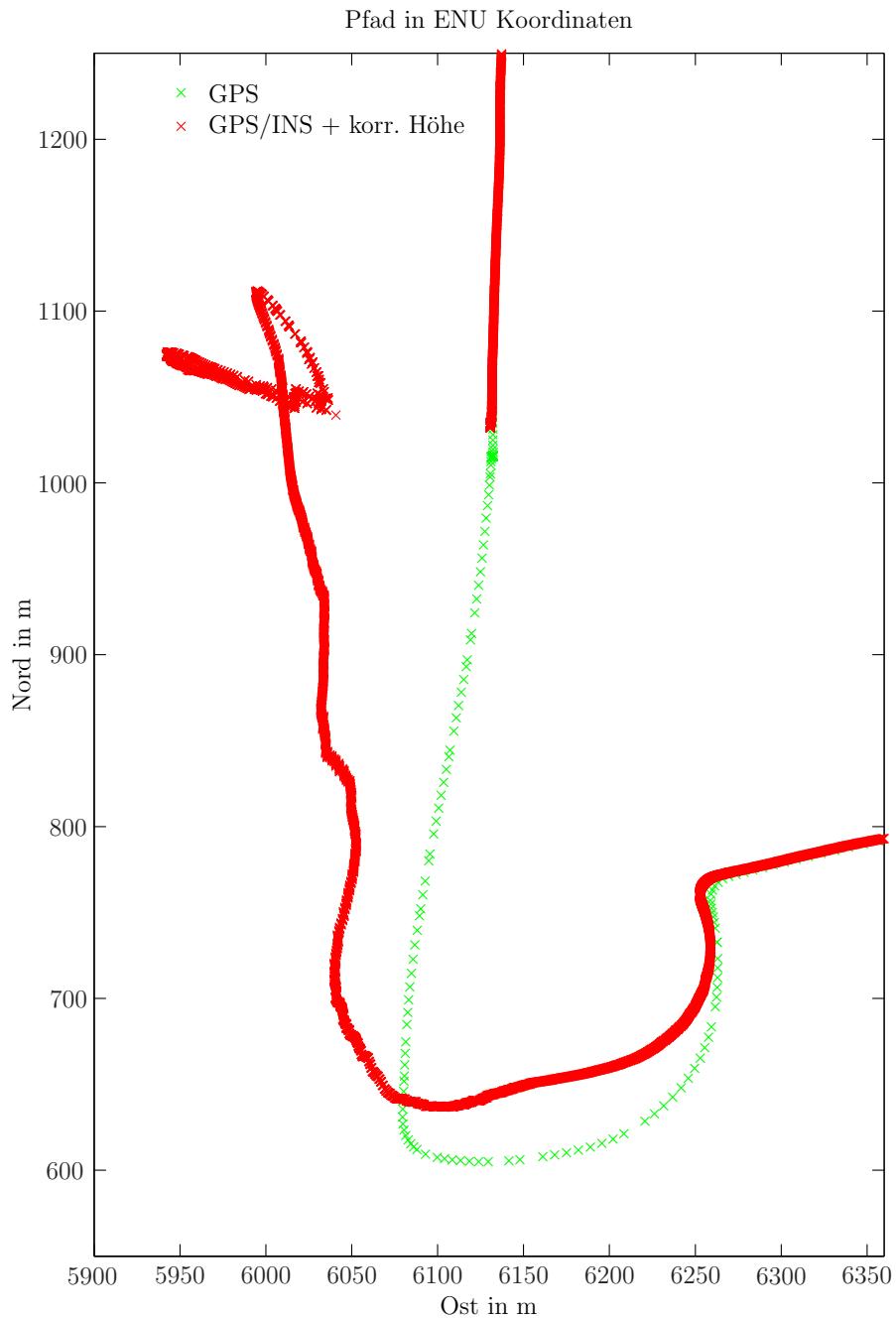


Abbildung 9.17: GPS-Ausfall 3 (mit Barometer)

Die Funktion des Höhenfilters wurde bereits vorher gezeigt. Da hier die korrigierte barometrische Höhe die ganze Zeit zur Verfügung steht, liefert auch das INS während des GPS-Ausfalls korrekte Höhenwerte.

Verhalten bei GPS-Ausfall ohne Verfügbarkeit von barometrischen Daten

Die Abbildung 9.18 zeigt das Ergebnis der INS-Navigationslösung ohne Verwendung der barometrischen Höhe während des GPS-Ausfalls. Dargestellt sind dieselben Ausfälle wie in Abbildung 9.16 und 9.17. Es ist deutlich zu erkennen dass der Fehler bei dem dritten GPS-Ausfall ohne Verwendung der barometrischen Höhe schneller anwächst, wie bei dem Ausfall mit Verwendung der barometrischen Höhe. Dies führt schnell zu einer unbrauchbaren Lösung. Auch bei dem zweiten GPS-Ausfall ohne barometrische Höhe wird der Fehler etwas größer wie einem Ausfall mit barometrischer Höhe. Lediglich bei dem ersten GPS-Ausfall ergibt sich kein sichtbarer Unterschied. Für die vom INS berechneten Höhen ergeben sich in allen drei Fällen sehr schnell große Fehler. Zusätzlich

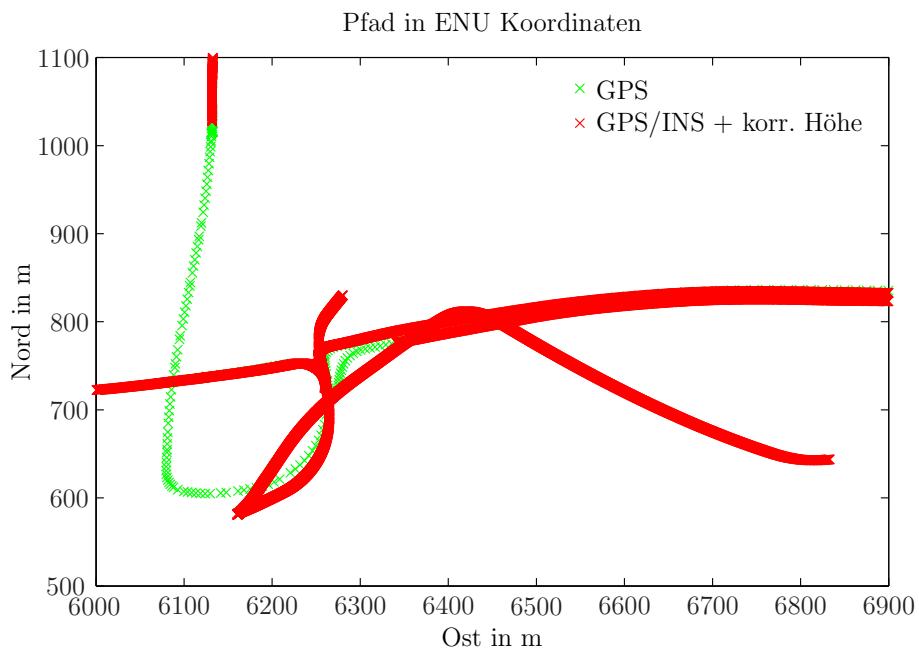


Abbildung 9.18: GPS-Ausfall 2 und 3 (ohne Barometer)

zu den vorherigen Untersuchungen mit simulierten Daten konnte hier gezeigt werden, dass die Verwendung der barometrischen Höhe auch einen deutlichen Vorteil bei der Positionsberechnung während eines GPS-Ausfalls bringt. Auch die Funktionsfähigkeit des INS zur Positionsbestimmung ohne andauernde Stützung durch GPS konnte gezeigt werden.

10 Zusammenfassung und Ausblick

Dieses Kapitel soll eine kurze Zusammenfassung der erzielten Ergebnisse sein und einen Ausblick für die Zukunft geben.

Zusammenfassung

Zielsetzung dieser Arbeit war es ein Verfahren zur Orientierungs-, Geschwindigkeits und Positionsbestimmung vorzustellen und zu Testen. In Kapitel 2 wurde ein Verfahren zur inertialen Navigation vorgestellt. Das Verfahren konnte erfolgreich auf seine Funktion getestet werden. Auch die in Kapitel 3 vorgestellte barometrische Höhenberechnung wurde erfolgreich getestet und konnte sein Potential zur Lieferung von zuverlässigen Höheninformationen zeigen. Die benötigten Sensoren wurden in Kapitel 4 vorgestellt. Soweit möglich, wurden auch eventuelle Fehler der Sensoren bestimmt und Korrekturmöglichkeiten angegeben sowie angewendet. Hierdurch konnte die Qualität der für die inertiale Navigation nötigen Sensoren verbessert werden. Das in Kapitel 5 vorgestellte Verfahren zur künstlichen Datenerzeugung konnte ebenfalls erfolgreich eingesetzt werden, um die in dieser Arbeit eingeführten Verfahren zu testen. Mit den in Kapitel 6 und Kapitel 7 vorgestellten Filterverfahren konnten sowohl die barometrische Höhenberechnung als auch die inertiale Navigation verbessert werden. Die in Kapitel 8 vorgestellte Kombination aus GPS und Barometer zur Unterstützung des inertialen Navigationsystems konnte zeigen, dass sie eine Verbesserung gegenüber einer klassischen Stützung per GPS bietet. Schließlich konnten die Tests in Kapitel 9 die wesentlichen Ziele dieser Arbeit bestätigen:

- Ein Verfahren zur inertialen Navigation wurde implementiert und erfolgreich getestet.
- Die Kombination aus inertialer Navigation und GPS ermöglicht eine Verbesserung der inertialen Navigation.
- Die Nutzung eines Barometers zur Stabilisierung der Höhenmessung zeigt weitere Verbesserungen für die inertiale Navigation gegenüber klassischen Verfahren.

Kurze Selbstkritik

Aufgrund des Umfangs dieser Arbeit wurde die Darstellung einiger Aspekte kürzer gehalten. Hiermit sollte zumindest ein grober Überblick über die vorgestellten Verfahren und ihre Möglichkeiten gegeben sein. Neben den hier vorgestellten Ergebnissen wurden

während der Untersuchung und Implementierung der Verfahren kleinere Test durchgeführt, die hier nicht weiter eingeflossen sind. Zum Großteil bilden die hier vorgestellten Ergebnisse die Ergebnisse dieser Tests allerdings ab.

Ausblick

Auch wenn diese Arbeit bereits das Potential der inertialen Navigation zeigen konnte, bleiben auch für die Zukunft weitere Punkt zu bearbeiten. Um die Fehlerfilterung weiter zu verbessern und in unterschiedlichsten Bedingungen optimal durchführen zu können, ist es z. B. nötig die Eigenschaften einiger Zufallsprozesse, wie dem Sensorrauschen oder dem Biasrauschen dynamisch zu schätzen. Hierzu wird bereits eine entsprechende Diplomarbeit durchgeführt.

Die Implementation in dieser Arbeit basieren noch auf dem Postprocessing der Daten. Für praktische Einsatz ist es nötig, die Implementation zu erweitern um eine Echtzeit-Verarbeitung zu ermöglichen.

Neben dem hier vorgestellten Einsatzgebiet der Fahrzeugnavigation im Außenbereich, kann das Verfahren der Inertialnavigation auch zur Personenlokalisierung im Innenbereich eingesetzt werden. Hierzu sind entsprechende Erweiterungen und Anpassungen nötig, um eine Stützung mit anderen Daten als GPS zu ermöglichen.

Abschließend sei noch erwähnt, dass aus dieser Arbeit ein Paper hervorgegangen ist welches zum Review für den “7th Workshop on Positioning, Navigation and Communication 2010 [WPNC10]” eingereicht wurde.

Literaturverzeichnis

- [ALE09] ALEKSJ CHINAEV, “*Rauschparameter der Inertialsensoren und ihre Simulation*”, Aleksj Chinaev, 2009
- [BORTZ] J.E. BORTZ, “*A new mathematical formulation for strapdown inertial navigation*”, IEEE Transactions on Aerospace and Electronic Systems, 1971 7(1):61 - 66
- [TIT04] DAVID H. TITERTON, JOHN L. WESTON, “*Strapdown Intertial Navigation Technology - 2nd Edition*”, The Institution of Electrical Engineers, 2004
- [WEN07] JAN WENDEL, “*Integrierte Navigationssysteme*”, Oldenburg, 2007
- [6DOF] WIKIPEDIA - 6 degree of freedom
<http://en.wikipedia.org/wiki/6DOF>
- [BAHST] WIKIPEDIA - Die Höhenstufen
http://de.wikipedia.org/wiki/Barometrische_Höhenformel#Die_H.C3.B6henstufen
- [BARER] ICK-HO WHANG, WON-SANG RA - Barometer error identification filter design using sigma point hypotheses
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4406559&isnumber=4406493>
- [CALMG] NOAA - NOAA’s Geophysical Data Center - Geomagnetic Data
<http://www.ngdc.noaa.gov/geomagmodels/struts/calcPointIGRF>
- [CAUCY] WIKIPEDIA - Cauchy-Verteilung
<http://de.wikipedia.org/wiki/Cauchy-Verteilung>
- [CSV] WIKIPEDIA - CSV (Dateiformat)
[http://de.wikipedia.org/wiki/CSV_\(Dateiformat\)](http://de.wikipedia.org/wiki/CSV_(Dateiformat))
- [DOP] WIKIPEDIA - Dilution of precision (GPS)
[http://en.wikipedia.org/wiki/Dilution_of_precision_\(GPS\)#Meaning_of_DOP_Values](http://en.wikipedia.org/wiki/Dilution_of_precision_(GPS)#Meaning_of_DOP_Values)
- [DGPS] WIKIPEDIA - Differential Global Positioning System
http://de.wikipedia.org/wiki/Differential_Global_Positioning_System

- [EGM96] NASA, NIMA - EGM96: The NASA GSFC and NIMA Joint Geopotential Model
<http://cddis.gsfc.nasa.gov/926/egm96/egm96.html>
- [EGNOS] WIKIPEDIA - European Geostationary Navigation Overlay Service
http://de.wikipedia.org/wiki/European_Geostationary_Navigation_Overlay_Service
- [GRMIN] GARMIN - Topo Deutschland V3 - Gesamt
<https://buy.garmin.com/shop/shop.do?pID=33441&pvID=34754>
- [GIS] PHYSIKALISCH-TECHNISCHE BUNDESANSTALT (PTB) - Gravity Information System PTB
<http://www.ptb.de/cartoweb3/SISproject.php>
- [GLI] GLI INTERACTIVE LLC - GLI Interactive LLC
<http://www.motionnode.com/>
- [GPFLT] GPSOFT LLC - GPSoft LLC: Navigation System Integration and Kalman Filter Toolbox 2.0 for MATLAB
<http://www.gpsoftnav.com/nsikf.html>
- [GPINS] GPSOFT LLC - GPSoft LLC: INS Toolbox 3.0 for MATLAB
<http://www.gpsoftnav.com/inertial.html>
- [GPSAT] GPSOFT LLC - GPSoft LLC: SatNav ToolBox 3.0 for MATLAB
<http://www.gpsoftnav.com/satnav.html>
- [HID] WIKIPEDIA - Human Interface Device
http://de.wikipedia.org/wiki/Human_Interface_Device
- [INSMA] INTERSEMA - INTERSEMA - MS5540C Altimeter/Barometer Module
<http://www.intersema.ch/products/guide/calibrated/ms5540c/>
- [KMFLT] KEVIN MURPHY - Kalman filter toolbox for Matlab
<http://people.cs.ubc.ca/~murphyk/Software/Kalman/kalman.html>
- [LN3NS] LN-3 INERTIAL NAVIGATION SYSTEM
<http://www.radelow.ch/anderes/avionik/inert/ln3/index.htm>
- [MNDOC] GLI INTERACTIVE LLC - MotionNode - Documentation
<http://www.motionnode.com/documentation.html>
- [MNDRV] GLI INTERACTIVE LLC - MotionNode - Software Updates
<http://www.motionnode.com/update/>
- [MNGTS] GLI INTERACTIVE LLC - Getting Started Guide
<http://www.motionnode.com/manual/Getting%20Started.pdf>
- [MNMGAG] GLI INTERACTIVE LLC - Magnetic Calibration Guide
<http://www.motionnode.com/manual/Magnetic%20Calibration.pdf>

- [MNSDK] GLI INTERACTIVE LLC - MotionNode - Software Development Kit (SDK)
<http://www.motionnode.com/sdk.html>
- [MNTUT] GLI INTERACTIVE LLC - MotionNode Tutorial
<http://www.motionnode.com/manual/MotionNode%20Tutorial.pdf>
- [MTNDE] GLI INTERACTIVE LLC - MotionNode Specification Sheet
http://www.motionnode.com/motionnode_spec_r00.pdf
- [NL] NAVILOCK - NAVILOCK - Produkte
http://www.navilock.de/produkte/gruppen/3/Kabel_Empfaenger
- [NMEA] DALE DEPRIEST - NMEA data
<http://www.gpsinformation.org/dale/nmea.htm>
- [QUAT] WIKIPEDIA - Quaternion
<http://de.wikipedia.org/wiki/Quaternion>
- [RTMAT] WIKIPEDIA - Euler angles
http://en.wikipedia.org/wiki/Euler_angles#Table_of_matrices
- [SAPOS] BEZIRKSREGIERUNG KÖLN - SAPOS - Bezirksregierung Köln - Geobasis NRW
<http://www.lverma.nrw.de/produkte/raumbezug/sapos/SAPoS.htm>
- [SBAS] WIKIPEDIA - Satellite Based Augmentation System
http://de.wikipedia.org/wiki/Satellite_Based_Augmentation_System
- [SCHEB] WIKIPEDIA - Schiefe Ebene
http://de.wikipedia.org/wiki/Schiefe_Ebene
- [SFDEM] SiRF - SiRFDemo
http://www.navilock.de/support/gruppen/13/Boards_und_Module/60409_NL-500ERS_Sirf3_RS232_Modul.html?type=5
- [SIRF] U-BLOX - u-blox
<http://www.u-blox.com/>
- [TIMOL] TOPOGRAPHISCHES INFORMATIOMSMANAGEMENT NORDRHEIN-WESTFALEN - TIM-Online NRW
<http://www.tim-online.nrw.de/tim-online/index.jsp>
- [TORDX] TORADEX - Toradex
<http://www.toradex.com/De/>
- [TORUB] TORADEX - Oak USB Sensoren
http://www.toradex.com/De/Products/Oak_USB_Sensors

- [TUM] TECHNISCHE UNIVERSITÄT MÜNCHEN - Geodesy and Geoinformation
<http://www.gug.bv.tum.de/seiten/technik/physik.html>
- [UBLOX] SiRF - SiRF
<http://www.sirf.com/>
- [UBPRT] U-BLOX - u-blox5 Protocol Specifications
http://www.u-blox.com/images/downloads/Product_Docs/u-blox5_Protocol_Specifications%28GPS.G5-X-07036%29.pdf
- [UCNTR] U-BLOX - u-center
<http://www.u-blox.com/de/download-center.html>
- [USA76] NASA, NOAA, USAF - U.S. Standard Atmosphere, 1976
http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770009539_1977009539.pdf, 1976
- [WETTR] UNI PADERBORN - Wetterstation “Universität Paderborn”
<http://wetter.upb.de/wetter3w.html>
- [WPNC10] 7th Workshop on Positioning, Navigation and Communication 2010 (WP-NC'10)
<http://www.wpnc.net/wpnc10.html>
- [XSMTI] XSENS - Miniature Attitude and Heading Sensor MTi - Xsens
<http://www.xsens.com/en/general/mti>

A Inhaltsübersicht der Begleit-DVD

Alle Zusatzinformationen sind in digitaler Form auf der beiliegenden DVD enthalten.
Im Folgenden wird die Verzeichnisstruktur der DVD erläutert:

- **Baro_Filter** – Enthält den barometrischen Höhenfilter.
- **Calibration** – Enthält Daten zur Kalibrierung der Sensoren.
- **GPS_INS** – Enthält den GPS/INS-Filter.
- **data** – Enthält aufgezeichnete Testdaten.

Der Inhalt der aufgeführten Verzeichnisse ist zur besseren Übersicht ggf. in Unterverzeichnisse aufgeteilt. Der CD ist außerdem ein weiteres Inhaltsverzeichnis beigelegt.