Indoornavigation mittels Ortsinterpolation

Diplomarbeit im Fach Informatik
vorgelegt von
Adrian Bachmann
von Emmen LU
02-704-245

Angefertigt am
Institut für Informatik
der Universität Zürich
Prof. Dr. A. Bernstein

Betreuer: Peter Vorburger
Abgabe der Arbeit: 15. März 2006

Der Weg ist das Ziel

- Konfuzius, Chinesischer Philosoph

Zusammenfassung

Die Satellitennavigation ist im täglichen Leben fast nicht mehr wegzudenken. Leider ist die Satellitennavigation jedoch in geschlossenen Räumen nicht verwendbar, da die entsprechenden Signale nicht empfangen werden können. Es existieren daher eine Vielzahl von Arbeiten, welche sich mit Alternativen zur Satellitennavigation, insbesondere im Bereich Indoornavigation, beschäftigen. In dieser Diplomarbeit wird ein neuer Ansatz erarbeitet, welcher sich auf Sensorinformationen von Magnet- und Beschleunigungssensoren stützt. Bei bisherigen Arbeiten mit Beschleunigungssensoren war stets die Eichung, aufgrund von ungenauen Kalibrierungen und sich verändernden Kalibrierungswerten, eines der grössten Probleme. Neu soll deshalb mittels einer fortwährenden Ortsinterpolation die Kalibrierung Online bestimmt werden, wodurch dank einer aktuelleren, exakteren und der Situation angegassten Kalibrierung eine viel genauere Wegbestimmung sowie anschliessende Extrapolation möglich sein sollte.

Abstract

Satellite navigation is ubiquitous in our daily life. Unfortunately, satellite navigation signals can not be received somtimes. Due to this effect, a new stream of reseach has been emerged focussing on alternatives, especially in the area of indoor-navigation. In this diploma thesis a new approach is developed and presented. The new approach induces in a new way navigation information from accelerometer and magnetometer sensor data. The new approach overcomes the shortcoming of insufficient calibration - one of the major issues in current research. The main contribution of this work is an online calibration framework, which allows to adapt to changing border conditions. Thus, the result is a much more robust, precise, and up-to-date base for the path extrapolation.

Danksagung

An dieser Stelle möchte ich mich bei all denen herzlich bedanken, welche zur Verwirklichung dieser Diplomarbeit beigetragen haben.
Aufrichtiger Dank gilt an erster Stelle Prof. Dr. Abraham Bernstein und Peter Vorburger, welche mir die interessante Auseinandersetzung mit diesem Thema ermöglichten. Speziell danken möchte ich Peter Vorburger auch für die exzellente Betreuung. Er stand mir stets tatkräftig mit nützlichen Hinweisen sowie aufschlussreichen Meetings zur Seite und begleitete mich motivierend während meiner gesamten Arbeit.
Danken möchte ich auch meiner Freundin Maria, welche mein grosses Engagement für diese Arbeit stets unterstütze, diese Arbeit unzählige Male durchgelesen sowie Korrekturvorschläge angebracht hat und mir den nötigen privaten Halt gab. Vielen Dank auch für die unzähligen Stunden, welche wir rätselnd über den Sensordaten verbracht haben.

Ein grosser Dank geht auch an sweety LaTeX, du hast meine Nerven geschont und mir das Verfassen dieser Diplomarbeit sehr erleichtert. Nicht vergessen möchte ich auch MATLAB: ich danke dir vielmals für die vielen Stunden voller high-Performance! ;-)

Zu guter Letzt möchte ich noch allen Personen danken, welche mir mein Studium an der Universität Zürich ermöglichten und damit einen wesentlichen Beitrag zu meinem persönlichen Werdegang geleistet haben. Danken möchte ich dabei vor allem meinen Eltern Trudy und Joe, meiner Freundin Maria sowie der gesamten restlichen Familie. Die zahlreichen Gespräche und das Wissen eurer stets vorhandenen Unterstützung haben mich auf meinem Weg bestärkt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Die Kunst des sich Zurechtfinden in einem geografischen Raum und die Wegfindung zu einem bestimmten Ort wird allgemein als Navigation bezeichnet.
Die Navigation spielt dabei bereits seit über 6000 Jahren eine grosse Rolle im täglichen Leben. Anfänglich war die Navigation besonders für die Schifffahrt von grosser Bedeutung, wobei man sich primär anhand von Sternen oder Landmarken orientierte. Ab dem ersten Jahrtausend v. Chr. wurde diese Kunst zunehmend auch an Land, vor allem auf Expeditionen, eingesetzt. Die Navigation wurde dabei über die Jahre zunehmend perfektioniert. Vor allem die Entdeckung des Kompasses und schliesslich die Einführung der Satellitennavigation lieferten dazu einen wesentlichen Beitrag.
Mit der Inbetriebnahme des Global Positioning System (GPS) im Jahr 1995 wurde die Erschliessung neuer Anwendungsgebiete, wie beispielsweise die Autonavigation oder die Vermessung per Satellitennavigation, möglich. Heutzutage sind Satellitennavigationssysteme nicht mehr nur in der See- und Luftfahrt sondern auch im Auto, Outdoor-Bereich, Vermessungswesen sowie für militärische Zwecke im Einsatz. Die neusten Trends lassen zudem darauf schliessen, dass die Satellitennavigation weitere Kreise des täglichen Lebens erreichen und zu einem integralen Bestandteil vieler neuen Anwendungen wird. Dieser Trend lässt sich beispielsweise bei Mobiltelefonen der neusten Generation beobachten, welche bereits mit einem integrierten GPS-Empfänger ausgestattet sind. Durch die allgegenwärtigen Navigationsmöglichkeiten ergeben sich auch neue ortsabhängige Dienste wie beispielsweise eine Stadtführung über das Smartphone.
Navigation beinhaltet dabei spätestens seit dem Einzug der Satellitennavigation nicht länger nur noch die Berechnung einer Position sondern auch Informationen über die Geschwindigkeit, Beschleunigung, Wegfindung, Zeit und Kompassausrichtung.

1.1. Motivation

Neuste Trends lassen darauf schliessen, dass die Navigation immer mehr ein integraler Bestandteil des täglichen Lebens und der genutzten Dienste wird. Die Navigation per Satellit gestaltet sich dabei als äusserst hilfreich solange eine ausreichende Empfangsqualität der Signale gegeben ist. Befindet man sich jedoch in Gebäuden oder anderen Örtlichkeiten in welchen der Empfang gestört ist, fallen die notwendigen Informationen zur Navigation weg.

Bei Fahrzeugen, welche beispielsweise durch Tunnels fahren und dabei das GPS-Signal verlieren, wird diesem Problem durch Radsensoren, anhand welcher sich der zurückgelegte Weg errechnen lässt, entgegnet.
In der Outdoor-Navigation, welche losgelöst von fest installierten Sensoren ist, gestaltet sich die Überbrückung der GPS-Signal freien Zeit jedoch als äusserst kompliziert. Durch die geringe Sendeleistung der GPS-Satelliten ist es leider auch nur schwer möglich, mit dem GPS-System eine Indoor-Navigation vorzunehmen.
Ein System, welches in der Lage ist, unabhängig von festen Installationen und der Ausnutzung von physischen Randbedingungen, eine Navigation relativ genau fortzuführen, sobald das GPS-Signal weg fällt, wäre deshalb von grossem Nutzen. Die Verwendung von Kleinstbauteilen für ein solches System würde zudem die Integration in mobile Geräte wie beispielsweise Smartphones ermöglichen.

1.2. Ziel dieser Arbeit

Ziel dieser Diplomarbeit ist es, eine Orts- und Geschwindigkeitsabschätzung mittels Bewegungsdaten als Inertiales Navigationssystem (INS) zu erstellen. Im Gegensatz zu früheren Arbeiten, wird ein neuer Ansatz verfolgt, indem mittels andauernder Orts-Interpolation die Sensoren vorzu geeicht werden und diese Eichung dann eine verlässliche Ortsbestimmung zulässt, sobald das GPS-Signal ausfällt. Zudem ist es häufig von grossem Interesse, nicht nur den Start- und Endpunkt einer zurückgelegten Strecke zu kennen, sondern auch die Strecke selbst.
Aus der Arbeit soll daher ein Framework resultieren, welches mittels Ortsinterpolation eine möglichst genaue Navigation bei Ausfall bzw. nicht vorhanden sein des GSP-Signals zulässt.
Die Genauigkeit des Frameworks soll bei anschliessenden Experimenten verifiziert und dessen Grenzen bzw. Genauigkeit aufgezeigt werden.

1.3. Anwendungsgebiete

Eine ortsgenaue Navigation ohne vorhandenes GPS-Signal lässt die Erschliessung neuer Anwendungsgebiete und die Optimierung vorhandener Applikationen zu. Gerade die Indoor-Navigation, welche sich heutzutage nur sehr schwer bewerkstelligen lässt, könnte von einem solchen System profitieren. Die Navigation im Outdoor-Bereich könnte verbessert und die GPS-freie Zeit überbrückt werden, wie dies beispielsweise in der Autonavigation durch Radsensoren bereits geschieht.
Auch die Forschung könnte von solch einem System profitieren. Zum aktuellen Zeitpunkt ist es beispielsweise in der Walforschung nur möglich, Wegpunkte der zurückgelegten

Strecke eines Wals aufzuzeichnen, da der angehängte GPS-Empfänger nur an der Wasseroberfläche Navigationsinformationen empfangen kann. Der zurückgelegte Weg kann daher nur anhand von Wegpunkten rekonstruiert werden, wobei viele Informationen verborgen bleiben. Dank zusätzlichen Sensoren wäre auch für dieses Anwendungsgebiet eine Interpolation der zurückgelegten Strecke denkbar.

1.4. Aufbau dieser Diplomarbeit

Nach einer Übersicht über die verwandten Arbeiten folgen im nächsten Kapitel die theoretischen Grundlagen einer Interpolation. In einem nächsten Kapitel wird die verwendete Hard- und Software und dessen Eigenheiten vorgestellt. Danach wird aufgezeigt, wie die Hardware kalibriert wurde und die Interpolation vorgenommen wird. In Experimenten wird anschliessend veranschaulicht, wie gut die Interpolationsmethode funktioniert und welche Genauigkeit man bei diesem Verfahren erwarten kann. Der Abschluss dieser Arbeit bildet schliesslich das Fazit und Schlusswort, wobei auch die noch offenen Probleme aufgezeigt werden.

1.4.1. Anmerkungen zur Notation

Durch die grosse Vielfalt von möglichen Notationen, ist es für den Leser nicht immer einfach zu verstehen, was ein Autor genau meint. Um Missverständnisse zu vermeiden, wird deshalb in diesem Unterkapitel kurz aufgezeigt, welche Notation in dieser Arbeit verwendet wird.

Mathematische Ausdrücke

Mathematische Gleichungen und Formeln sind in dieser Arbeit nummeriert und als solche erkenntlich. Zudem wird folgende mathematische Notation angewandt:

- Vektoren werden mit einem Pfeil gekennzeichnet: $\vec{a}$
- Matrizen werden mit Grossbuchstaben gekennzeichnet: $\mathcal{A}$
- Winkel werden mit griechischen Buchstaben gekennzeichnet: α

Weibliche und männliche Form

Aus Gründen der Lesbarkeit wird in dieser Diplomarbeit darauf verzichtet, jeweils die weibliche und männliche Form aufzuführen. Bei Verwendung der männlichen Form ist selbstverständlich auch immer die weibliche Form gemeint.

1.5. Verwandte Arbeiten

In den vergangenen Jahren wurde viel Forschung in der GPS-unabhängigen Navigation, vor allem im Bereich so genannter inertialen Navigationssysteme (INS), betrieben. Ein Grossteil der Forschung beschäftigte sich dabei mit der Erhöhung der Genauigkeit von Navigationssystemen für Fussgänger, auch bekannt unter der Bezeichnung Pedestrian Navigation System (PNS). Inertiale Navigationssysteme sind dabei keine grundsätzliche Neuerfindung, sondern werden bereits seit Jahren erfolgreich in der Luft-, Raum- und Schifffahrt sowie in Bergwerken und der Autonavigation eingesetzt.
Das grösste Problem von INS ist der Integrationsfehler, welcher sich vor allem bei der Berechnung der Position über eine doppelte Integration der Geschwindigkeit bemerkbar macht. Das Aufintegrieren von Daten mit kleinen Fehlern lässt dabei immer grössere Ungenauigkeiten entstehen. Viele Ansätze versuchen daher genau diesem Problem zu entgegnen.
In diesem Unterkapitel sollen deshalb kurz einige verwandten Arbeiten und dessen Ansätze vorgestellt werden.

1.5.1. Kartenabstraktion

Gleich mehrere Arbeiten beschäftigen sich mit dem Thema der Kartenabstraktion. Von [Büchel 2004] wurde beispielsweise ein System entwickelt, welches aus folgenden drei Teilen besteht:

1. Ein persönliches Navigationsmodul (PNM: Personal Navigation Module), welches die Daten über die geografische Position eines Benutzers eruieren kann

2. Eine Datenbank, welche Kartenabstraktionen, bestehend aus Knoten und Verbindungen, des gewünschten geografischen Raumes beinhaltet

3. Algorithmen welche in der Lage sind, die Position aus dem PNM mit den Karten zu verbinden

Anhand des PNM wird dabei die Position der Person und deren zurückgelegter Weg eruiert. Dabei unterliegen die Angaben des PNM auch Fehler (wie sie bei allen INS vorkommen), wodurch die Positionsangaben des PNS bald ungenau sind. [Büchel 2004] versucht nun die Fehlabweichungen des PNM mit regelmässigen Korrekturen zu beseitigen. Dabei wird die durch das PNM eruierte Position des Users, mit in der Datenbank abgelegten, potentiellen Pfaden und des next-point Algorithmus abgeglichen.
Ein grosser Nachteil dieses Systems liegt sicherlich darin, dass für jedes Einsatzgebiet abstrahiertes Kartenmaterial vorliegen und in einer Datenbank entsprechend abgelegt werden muss.
Auch [Fang u. a. 2005] bestreitet einen ähnlichen Ansatz, wobei hier das Kartenmaterial zusätzlich verwendet wird, um die Präzision des Systems zu erhöhen. Dabei spielt die Kartenabstraktion eine weniger grosse Rolle als bei [Büchel 2004].

1.5.2. GPS Integration

Ein guter Weg um mit ungenauen Navigationsangaben von INS zurecht zu kommen, ist die Kombination von INS und GPS. Dieser Ansatz wird unter anderem in [Ladetto u. a. 2002], [Ladetto und Merminod 2002b], [Ladetto u. a. 2001], [Ford u. a. 2001] und diversen anderen Arbeiten verfolgt.
Das Verfahren ist dabei relativ einfach: Die INS-Positionsangaben werden mit Kalman-gefilterten GPS-Ortsinformationen aktualisiert. Dabei wird das INS mit GPS-Ortsinformationen versorgt und kann somit regelmässig nachkalibriert werden. Dieses System geht jedoch nur von kurzen GPS-Ausfällen aus, denn bei längeren Ausfällen wird das nicht mehr länger geeichte INS durch den Integrationsdrift sehr ungenau. Ein gutes Beispiel hierfür findet sich in [Ford u. a. 2001].

1.5.3. Schritterkennung

Die meisten INS welche als PNS eingesetzt werden, beinhalten Beschleunigungssensoren. Diese Sensoren messen dabei die Beschleunigung, meist in alle drei Dimensionen, welche auf die Hardware bzw. Sensoren wirkt. Die gemessene Beschleunigung beinhaltet dabei immer auch die Erdbeschleunigung. Selbst wenn die Hardware regungslos auf einem Tisch liegt, wirkt also die Erdbeschleunigung auf die Sensoren. Eine direkte Integration der Sensordaten zur Berechnung der Geschwindigkeit bzw. eine doppelte Integration zur Berechnung der Position ist also nicht möglich, da die Erdbeschleunigung immer auch in den Daten vorhanden ist. Im optimalsten Fall könnte man die Erdbeschleunigung einfach abziehen, was jedoch aufgrund der unbekannten Ausrichtung der Hardware zur Erdoberfläche nicht immer ganz einfach ist. Zudem besteht bei einer Integration immer auch das Problem mit dem Integrationsdrift.
Viele Forschungsgruppen haben deshalb nach Lösungen gesucht, die Geschwindigkeit

sowie Position anhand von Beschleunigungssensoren auf eine andere Art bestimmen zu können als mit einer Integration.
Ein interessanter Ansatz für dieses Problem taucht in vielen Publikationen wie beispielsweise [Brännström 2002], [Gabaglio und Merminod 1999] und [Ladetto und Merminod 2002a] auf. Ausgehend von den Daten der Beschleunigungssensoren sollen Schritte erkannt werden. Interessanterweise können dabei nicht nur die Schritte selber, sondern auch dessen Länge und Art (Seitenschritte, Schritte zurück, etc.) erkannt werden.
So interessant dieses System auch ist, hat es doch einige Nachteile zu verzeichnen. So muss es beispielsweise für jede Person und deren Schrittart neu kalibriert werden. Zusätzlich tragen Faktoren wie unterschiedliche Boden- und Terrainbeschaffenheit (z.B. Sand, Schnee, Felsen, etc.) sowie Fitness des Benutzers etc. zu Ungenauigkeiten bei. Zudem versagt das System komplett, sobald nicht mehr von normalen Schritten, wie beispielsweise auf einer Bergtour, ausgegangen werden kann oder die Person nicht mehr zu Fuss sondern beispielsweise auf einem Fahrrad oder in einem Kanu unterwegs ist.

1.5.4. Extrapolation der Geschwindigkeit und Position

Einführung

Basierend auf dem Prototypen der in dieser Diplomarbeit verwendeten Hardware, befasste sich Stephan Tschanz mit einer Extrapolationsmethode [Tschanz 2005] zur Berechnung von Navigationsinformationen mit einem INS. Folgende Methoden wurden dabei verwendet um exakte Navigationsinformationen einer sich bewegenden Person zu eruieren:

1. Erkennung der räumlichen Orientierung des Sensor-Boards- das Sensor-Board ist nicht an eine bestimmte Position am Körper gebunden

2. Integration der Beschleunigungsdaten zur Berechnung der Reisegeschwindigkeit

3. Doppelte Integration der Beschleunigungsdaten zur Berechnung des relativ zurückgelegten Weges

4. Extraktion von Magnetsensordaten zur Berechnung des Azimut der Bewegung

5. Extraktion von Winkelbeschleunigungsdaten zur Berechnung des relativen Azimut der Bewegung. Kombiniert mit der Extraktion des Azimut aus den Magnetsensordaten kann eine Kompensation von Fehlmessungen des Magnetsensors in magnetisch gestörten Bereichen erreicht werden.

Tschanz kalibrierte dabei zu Beginn die Sensoren und errechnete anschliessend mittels Extrapolation die gewünschten Navigationsinformationen wie Geschwindigkeit und Position. Dabei hat Tschanz der Erdbeschleunigung keine besondere Beachtung geschenkt, sondern diese in der Kalibrierung mit berücksichtigt.

Verwendete Hard- und Software

Tschanz verwendete für seine Arbeit zwei verschiedene Hardwarekomponenten: Smart-IT und ein Sensor-Board Prototyp von Peter Vorburger[1].
Smart-IT ist ein Sensor-Board mit zwei ADXL311 Beschleunigungssensoren von Analog Devices[2] und einem Multiplexer. Die Kommunikation mit dem Smart-IT Sensor-Board[3] erfolgte über eine RS232-Schnittstelle.
Das in [Tschanz 2005] verwendete Sensor-Board von Peter Vorburger ist ein experi-

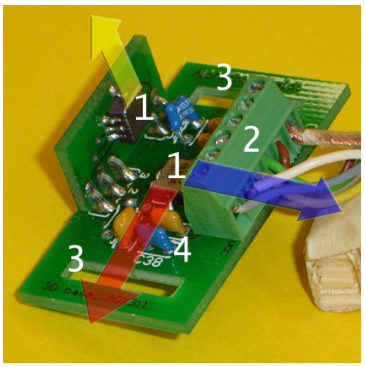

Abbildung 1.1.: Beschleunigungssensor-Board von Smart-IT

menteller Prototyp des in dieser Diplomarbeit verwendeten Sensor-Boards. Das Board beinhaltet ebenso zwei Beschleunigungssensoren vom Typ ADXL320 (vgl. hierzu [AnalogDevices 2004]) sowie eine RS232-Schnittstelle. Zudem sind zwei Winkelbeschleunigungssensoren vom Typ CG-L43B und CG-L53B (vgl. hierzu [NEC/TOKIN 2001] und [NEC/TOKIN 2005a]), beide von NEC/TOKIN, und ein Magnetsensor aus dem Hause Honeywell vom Typ HMC1053 (vgl. hierzu [Honeywell 2003]) auf dem Board untergebracht, wodurch ein Digitaler Magnetischer Kompass (DMC) erstellt werden kann.
Als Software bzw. Recorder wurde ein handelsübliches Notebook mit RS323-Schnittstelle auf der Softwarebasis von Windows XP und dessen mitgeliefertes Hyperterminal verwendet. Die Sensordaten wurden dabei von beiden Sensor-Boards über die RS232-Schnittstelle via Hyperterminal in Textfiles aufgezeichnet.

[1] Weitere Informationen zu Peter Vorburger können abgerufen werden unter http://www.ifi.unizh.ch/ddis/people/vorburger/
[2] Für weitere Informationen zu den hier verwendeten Beschleunigungssensoren beachte man das Datenblatt auf http://www.analog.com/UploadedFiles/Data_Sheets/243920868ADXL311_B.pdf
[3] Mehr Informationen zu Smart-IT sind abrufbar unter http://www.smart-its.org

Kalibrierung

Zur Extrapolation der Navigationsdaten hat [Tschanz 2005] vorgängig die Sensoren kalibriert.
Für die Kalibrierung der Beschleunigungssensoren hat er sich die Erdbeschleunigung g, welche in Zürich bekannt ist, zu Hilfe genommen und entsprechend den Umrechnungsfaktor der Sensoroutputs zur physikalischen Beschleunigung berechnet. Eine direkte Ableitung der Kalibrierung aus den technischen Daten hat Tschanz ausgeschlossen, da die Referenzdaten nur für eine bestimmte Versorgungsspannung gelten und daher für jeden Sensoreinsatz neu bestimmt werden müssten. Die Erdbeschleunigung hat Tschanz dabei in die Kalibrierung eingeschlossen und nicht separat betrachtet. Das genaue Kalibrierungsverfahren ist in Abbildung 1.2 ersichtlich.
Für die Kalibrierung der Magnetsensoren hat Tschanz die Sensor-Höchts- und Tiefst-

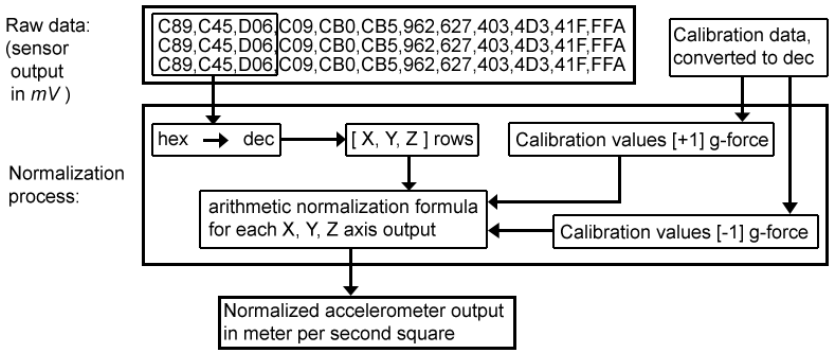

Abbildung 1.2.: Kalibrierung und Normalisierung der Beschleunigungssensoren [Tschanz 2005]

werte an einem magnetisch ungestörten Ort eruiert. Anschliessend berechnete er die Normalisierungsgrössen. Aus den normalisierten Magnetsensordaten lässt sich anschliessend die Nordrichtung bestimmen.
Das genaue Kalibrierungsverfahren ist in Abbildung 1.3 ersichtlich.
Die Kalibrierung der Winkelbeschleunigungssensoren hat Tschanz mit Hilfe eines sich um 360-Grad drehenden Körpers (im konkreten Fall handelte es sich aufgrund von fehlender Ausrüstung um einen handelsüblichen Bürostuhl bzw. um einen Plattenspieler) vorgenommen. Das genaue Kalibrierungsverfahren ist in Abbildung 1.4 ersichtlich.

Extrapolation

Die Ortsextrapolation basiert auf dem Prozess, bei welchem verschiedene Informationen von den Sensoren mittels Integration zu einem einzigen Informationsblock, bestehend

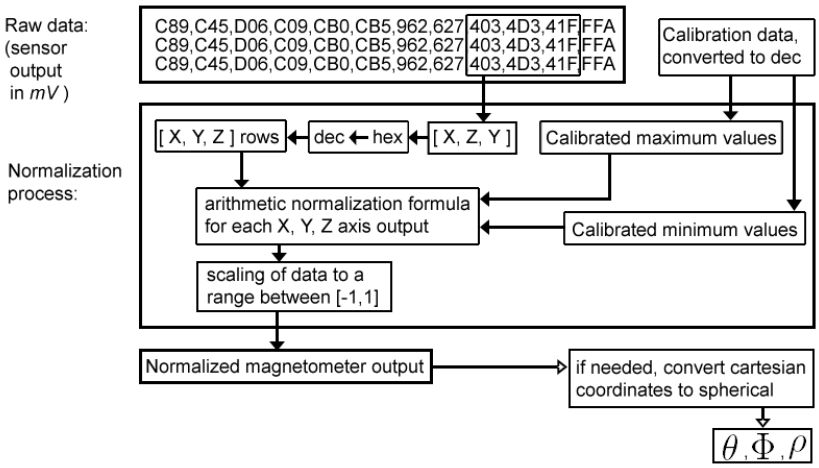

Abbildung 1.3.: Kalibrierung und Normalisierung der Magnetsensoren [Tschanz 2005]

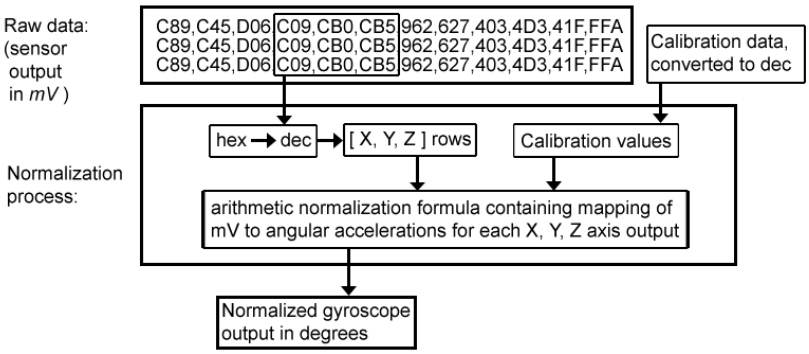

Abbildung 1.4.: Kalibrierung und Normalisierung der Winkelbeschleunigungssensoren [Tschanz 2005]

aus geografischer Position und Richtung, zusammengefügt werden. Um einen Ort extrapolieren zu können, muss der Startpunkt gegeben sein. Die Bewegung wird anschliessend, anhand von Sensorinformationen über die Zeit, ausgehend von einem gegebenen Startpunkt berechnet.
Die Geschwindigkeit und Position kann dabei mittels einfacher bzw. doppelter Integration der normalisierten Beschleunigungssensordaten errechnet werden.
Die geographische Ausrichtung relativ zum magnetischen Norden wird mit dem DMC errechnet. Magnetische Störfelder können dabei durch die zusätzlichen Informationen der Winkelbeschleunigungssensoren abgefangen werden.

Resultate und Fazit

Die Ortsextrapolationsmethode hängt von vielen Faktoren, wie beispielsweise die wechselnde Sensor-Board Ausrichtung, der Integrationsdrift und unterschiedliche Schrittarten, ab, welche eine genaue Ortsextrapolation erschweren. In den Versuchen von Tschanz mit den Beschleunigungssensoren kamen daher auch nur sehr ungenaue Resultate zum Vorschein. Abbildung 1.5 zeigt eine 100m-lange Strecke welche unterschiedlich schnell

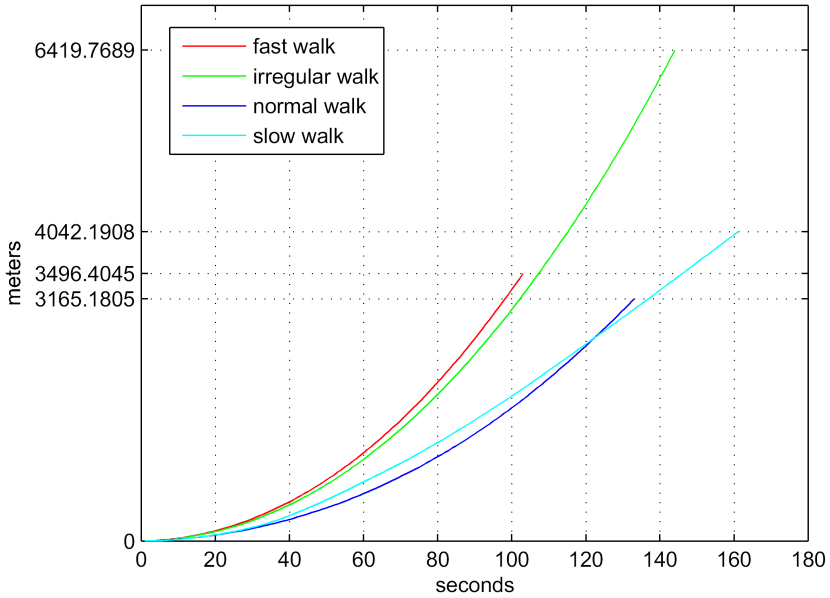

Abbildung 1.5.: Ortsextrapolation anhand der Beschleunigungssensoren [Tschanz 2005]

zurückgelegt wurde und die aus den Sensordaten errechnete zurückgelegte Strecke.
Die Probleme für die ungenauen Resultate waren dabei die folgenden:

- Die statische und zu ungenaue Kalibrierung
- Der Integrationsdrift zusammen mit kleinen Kalibrierungsabweichungen
- Das Vernachlässigen der Sensor-Board-Bewegungen
- Das dadurch fehlerhafte Berücksichtigen der Erdbeschleunigung (die Erdbeschleunigung wirkte nicht immer in die gleiche Richtung)

Trotzdem konnte [Tschanz 2005] aufzeigen, dass die Ortsinformationen der Magnet- und Winkelbeschleunigungssensoren sehr viel versprechend sind und in jedem Fall für ein künftiges PNS verwendet werden können.

1.6. Neuer Ansatz: Interpolation der Ortsinformationen

Im Gegensatz zu [Tschanz 2005] soll in dieser Diplomarbeit ein neuer Weg beschritten werden. Anstelle einer einmaligen festen Kalibrierung, soll mittels Interpolation eine Online-Eichung des INS vorgenommen werden. Dabei sind Start- und Endpunkt bekannt, worauf eine Kalibrierung vorgenommen und anschliessend der zurückgelegte Weg zwischen Start und Ende interpoliert werden kann. Zusätzlich soll es durch die fortwährende Eichung auch möglich sein, eine viel genauere Extrapolation vorzunehmen, sobald das Referenzsignal (beispielsweise GPS) nicht mehr zur Verfügung steht.
Dabei wird auf einem neuen Sensor-Board aufgebaut, welches von Peter Vorburger nun im dritten Prototyp-Stadium vorliegt.
Für die Berechnung der benötigten Navigationsinformationen sollen dabei, analog zu [Tschanz 2005], Beschleunigungs, Magnet- sowie Winkelbeschleunigungssensoren eingesetzt werden. Zusätzlich wird ein Temperatursensor zur Korrektur von Temperatureinflüssen verwendet.
Zur Berechnung der Ortsinformationen sollen dabei zwei Verfahren geprüft werden: Einerseits die doppelte Integration der Beschleunigung über die Zeit zur direkten Berechnung des Ortes und andererseits die einfache Integration der Beschleunigung über die Zeit zur Berechnung der Geschwindigkeit, welche zusammen mit den Richtungsangaben der Magnetsensoren sowie der verstrichenen Zeit ebenfalls den Ort ergeben.

2. Theoretische Grundlagen

2.1. Einführung

In diesem Kapitel sollen Methoden vorgestellt werden, mit welchen sich eine Positionsangabe mit den vorhandenen Sensoren errechnen lässt. In den folgenden Kapiteln werden dann die Hardware, deren Kalibrierung und schliesslich Experimente mit den hier vorgestellten Methoden beschrieben.
Die Berechnung der Position ist mit den zur Verfügung stehenden Sensoren grundsätzlich auf verschiedene Arten möglich. Zwei davon erweisen sich als viel versprechend:

1. **Doppelte Integration der Beschleunigung**
   Bei der doppelten Integration werden die Beschleunigungsdaten zweifach über die Zeit integriert, wodurch sich direkt die zurückgelegte Distanz ergibt. Das grösste Problem bei der doppelten Integration ist der Integrationsdrift: kleine Fehler wirken sich gleich mehrfach aus, wodurch grosse Ungenauigkeiten bei der Berechnung entstehen können.

2. **Einfache Integration der Beschleunigung mit Richtungsinformationen**
   Bei der einfachen Integration werden die Beschleunigungsdaten einfach über die Zeit integriert, wodurch sich die Geschwindigkeit ergibt. Von der errechneten Geschwindigkeit ist jedoch nur die Geschwindigkeit selbst und nicht deren Richtung von Interesse. Die Richtungsangaben werden durch weitere Sensoren, im diesem Fall Magnet- bzw. Winkelbeschleunigungssensoren, eruiert. Da in diesem Fall nur eine einfache Integration der Geschwindigkeit nötig ist, setzen sich Fehler auch nicht in einem solch grossen Ausmass fort, wie bei einer doppelten Integration.

Ein grosses Problem bei beiden Berechnungsmethoden ist die Erdbeschleunigung. In jedem Fall messen die Beschleunigungssensoren auch die Erdbeschleunigung von rund $9.81\frac{m}{s^2}$, besser bekannt als 1g. Solange die relative Ausrichtung des Sensor-Boards zur Erdoberfläche gleich bleibend ist, kann die Erdbeschleunigung einfach abgezogen werden. Etwas schwieriger wird der Fall, wenn das Sensor-Board in ständiger Bewegung ist und seine relative Ausrichtung zur Erdoberfläche ständig verändert. Dabei wird sich eine zentrale Frage aufdrängen: In welche Richtung wirkt die Erdbeschleunigung relativ zum Sensor-Board und wie kann diese Information in Erfahrung gebracht werden?

Im Folgenden muss die Ausrichtung des Sensor-Boards etwas näher betrachtet werden. Es wird dabei von zwei Fällen ausgegangen und folgende Notation verwendet:

- **Lokales Bezugssystem**
  Das lokale Bezugssystem ist das Koordinatensystem, welches vom Sensor-Board und dessen Sensoren gegeben ist (vgl. Abbildung 2.1).

- **Globales Bezugssystem**
  Das globale Bezugssystem stellt die Erdoberfläche, im vorliegenden Fall in Zürich, und das darauf positionierte kartesische Koordinatensystem dar.

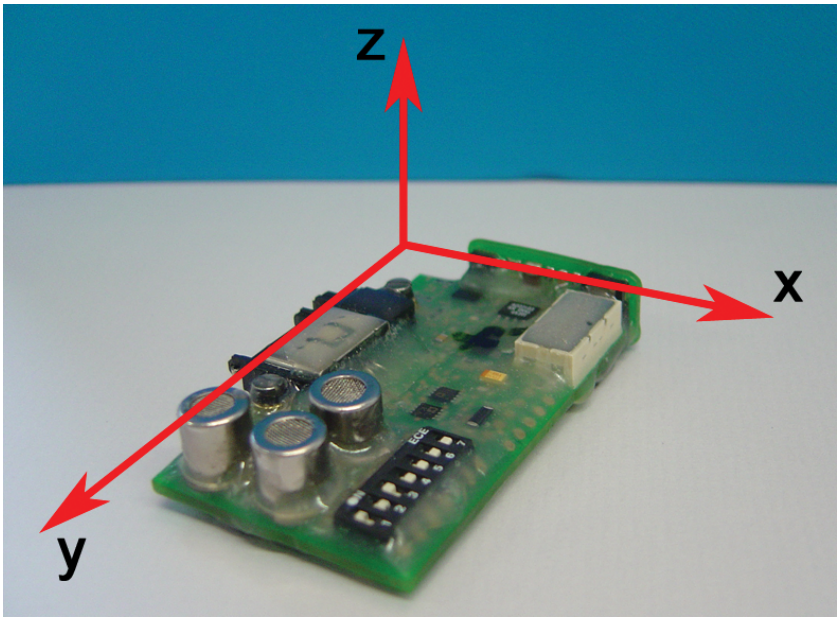

Abbildung 2.1.: Sensor-Board als lokales Bezugssystem

In den folgenden Unterkapiteln wird zwischen zwei Szenarien unterschieden. Im ersten Fall wird von einem parallelen lokalen Bezugssystem ausgegangen, wobei das Sensor-Board seine relative Ausrichtung zur Erdoberfläche nicht verändert. Dies ist beispielsweise der Fall, wenn das Sensor-Board zu jedem Zeitpunkt waagerecht zur Erdoberfläche ausgerichtet ist.
Im zweiten Fall, wird von einer Variante ausgegangen, bei welcher das Sensor-Board seine relative Ausrichtung zur Erdoberfläche ständig verändern kann (frei bewegliches lokales Bezugssystem). Dabei werden Methoden und Verfahren präsentiert, um die Erdbeschleunigung auch im Einsatz bei sich verändernder Board-Ausrichtung berücksichtigen zu können. Die Beantwortung der Frage, in welche Richtung die Erdbeschleunigung wirkt, wird dabei von zentraler Bedeutung sein.
Gerade für den letzten Fall, werden Drehmatrizen eine entscheidende Rolle spielen. Im nächsten Unterkapitel wird deshalb kurz in das Thema Drehmatrizen eingeführt.

2.2. Drehmatrizen

Mittels Drehmatrizen ist es möglich, eine Drehung im Koordinatensystem (euklidischer Raum) zu beschreiben. In diesem Fall wird von einem globalen Bezugssystem (bzw. globalen Koordinatensystem) ausgegangen, welches zu Beginn der Interpolation festgelegt werden muss. Dabei wird als kartesisches Koordinatensystem auch gleich das Koordinatensystem des Sensor-Boards festgelegt.
Verändert sich nun die Lage des Sensor-Boards zur ursprünglichen Lage (relativ zur Erdoberfläche), so können diese Veränderungen auch als Drehung des Koordinatensystems betrachtet werden.
Mittels Drehmatrizen ist es nun möglich, diese Drehungen im Koordinatensystem zurück ins ursprüngliche Koordinatensystem zu korrigieren.
Würden keine Korrektur der Drehungen vorgenommen, so müssten für jeden Zeitschritt die Unbekannten neu berechnet werden, was aufgrund der gegebenen Informationen/-Annahmen leider nicht möglich ist.
Für jeden Zeitschritt wird daher eine Drehmatrix berechnet, welche die Korrektur von der aktuellen Drehung des Koordinatensystems (bzw. Sensor-Boards) zur ursprünglichen Lage (zu Beginn der Interpolation festgelegt) ermöglicht. Daher können die Unbekannten auf Basis der festgelegten Lage (globalen Koordinatensystem) interpoliert und über die gesamte Dauer als statisch betrachtet werden.

2.2.1. Beispiel für den zweidimensionalen Raum

Die Funktionsweise von Drehmatrizen lässt sich an einem Beispiel im zweidimensionalen Raum relativ einfach demonstrieren.
Gegeben sei ein Punkt $P = (2, 1)$ relativ zum Koordinatensystem (x, y). Eine Drehung des Koordinatensystems um $-45°$ ergibt ein neues (gedrehtes) Koordinatensystem (x', y') (vgl. Abbildung 2.2). Die Koordinaten des Punktes P im Koordinatensystem (x', y') lassen sich nun mittels Drehmatrix berechnen.
Die Drehmatrix (vgl. hierzu [DMK/DPK 1995]) für diesen Fall hat folgende Form:

$$\mathcal{R} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{2.1}$$

Der Drehwinkel α beträgt in diesem Beispiel $-45°$ bzw. $-\frac{\pi}{4}$. Dadurch lassen sich die Koordinaten von Punkt P bzw. P' im Koordinatensystem (x', y') wie folgt errechnen:

$$P' = \begin{pmatrix} x' \\ y' \end{pmatrix} = \mathcal{R} \cdot P = \begin{pmatrix} \cos -\frac{\pi}{4} & -\sin -\frac{\pi}{4} \\ \sin -\frac{\pi}{4} & \cos -\frac{\pi}{4} \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2.121 \\ -0.707 \end{pmatrix} \tag{2.2}$$

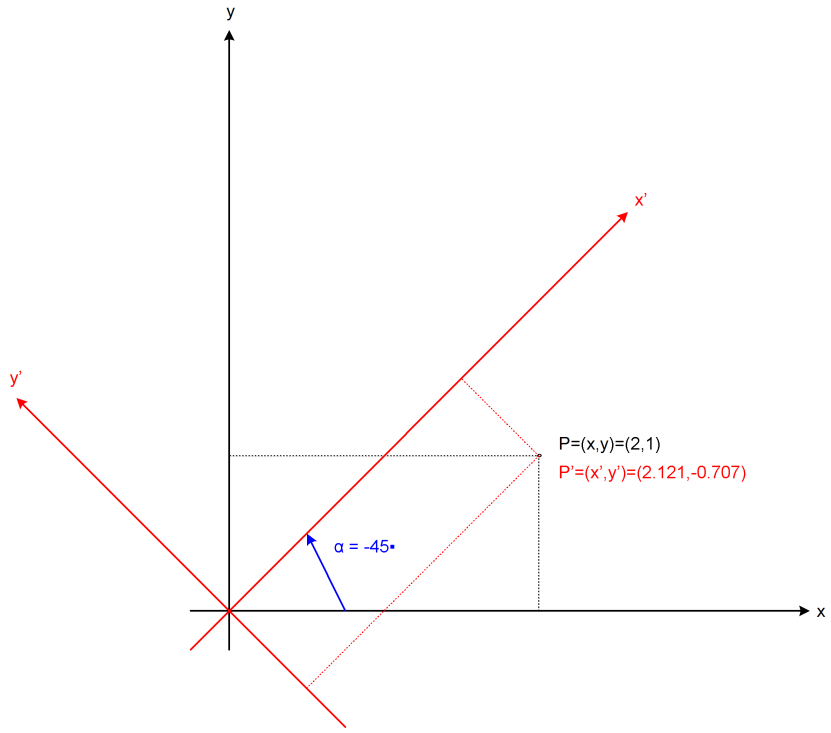

Abbildung 2.2.: Beispiel für die Berechnungen mit Drehmatrizen

Der umgekehrte Fall kann selbstverständlich auch mit Drehmatrizen berechnet werden. Ausgehend von einem gegebenen, globalen Koordinatensystem (x, y), dem Drehwinkel α sowie den Punktkoordinaten $P' = (x', y')$ aus dem (gedrehten, aktuell gültigen) Koordinatensystem (x', y'), lassen sich nun mittels Drehmatrize die Punktkoordinaten für P im Koordinatensystem (x, y) (dem zu Beginn festgelegten Bezugssystem) berechnen.

2.2.2. Drehmatrix für den dreidimensionalen Raum

Der Einsatz von Drehmatrizen ist selbstverständlich nicht auf den zweidimensionalen Raum beschränkt, sie existieren auch für den dreidimensionalen Raum. Eine Drehung kann dabei auf drei Arten geschehen:

1. Drehung um die x-Achse mit Winkel α
2. Drehung um die y-Achse mit Winkel β
3. Drehung um die z-Achse mit Winkel γ

Selbstverständlich ist dabei auch eine Kombination aller drei Fälle zulässig.
Für die Drehung um den Ursprung mit den Winkel α, β, oder γ gelten dabei folgende Drehmatrizen (vgl. hierzu [Eberhard 2005]):

- Drehung mit der x-Achse als Drehachse
  $\mathcal{X} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$
- Drehung mit der y-Achse als Drehachse
  $\mathcal{Y} = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}$
- Drehung mit der z-Achse als Drehachse
  $\mathcal{Z} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Die Drehmatrize $\mathcal{T}$ für eine beliebige Drehachse kann dabei durch Multiplikation der Drehmatrizen $\mathcal{X}$, $\mathcal{Y}$ und $\mathcal{Z}$ errechnet werden:

$$\mathcal{T} = \mathcal{X} \cdot \mathcal{Y} \cdot \mathcal{Z} \tag{2.3}$$

$$\mathcal{T} = \begin{pmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\beta \\ \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma & \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\beta \end{pmatrix} \tag{2.4}$$

2.3. Allgemeine Berechnung

In diesem Unterkapitel soll zuerst die allgemeine Berechnung hergeleitet werden. In weiteren Unterkapiteln werden die hergeleiteten Gleichungen dann unter bestimmten Annahmen angepasst und vereinfacht.
Die Geschwindigkeit lässt sich mittels Integration der Beschleunigung über die Zeit errechnen. Eine weitere Integration über die Zeit liefert schliesslich den zurückgelegten Weg.
Für die Herleitung müssen im vorliegenden Fall auch die Eigenschaften der verwendeten Hardware berücksichtigt werden. Als Ausgabe liefert das Sensor-Board Messwerte zwischen 0 und 4096. Eine Beschleunigung von $0\frac{m}{s^2}$ entspricht dabei nicht 0 Messpunkten.

Diese Verschiebung durch den ADC (für Details siehe Kapitel 3) muss mit der Subtraktion eines Offset $\vec{a}_{offset_{ADC}}$ korrigiert werden.
Auch die Erdbeschleunigung welche mit $9.81\frac{m}{s^2}$ bzw. einem g auf die Beschleunigungssensoren wirkt, muss in Form eines Offset $\vec{a}_{offset_g}$ subtrahiert werden.
Zudem können die um die Verschiebung korrigierten Messwerte nicht direkt als physikalische Grössen angesehen werden. 140 Messpunkte entsprechen beispielsweise ungefähr $9.81\frac{m}{s^2}$ bzw. einem g. Das Verhältnis zwischen Messpunkten und physikalischer Grösse $\frac{m}{s^2}$ wieder spiegelt die Matrix $\mathcal{A}$.
In den Berechnungen von [Tschanz 2005] sind zudem Nichtlinearitäten beim Sensoroutput aufgetaucht, welche es zu korrigieren gilt. Für die Korrektur der Nichtlinearitäten wird deshalb der Tensor $\mathcal{G}$ eingeführt.

2.3.1. Doppelte Integration der Beschleunigung

Durch eine Integration der Beschleunigung über die Zeit lässt sich die Geschwindigkeit berechnen. Dabei werden von den einzelnen Beschleunigungswerten $\vec{a_i}$ die beiden Offset $\vec{a}_{offset_{ADC_i}}$ und $\vec{a}_{offset_{g_i}}$ abgezogen. Zudem muss der bereinigte Messwert mit der Streckmatrize $\mathcal{A}_i$ und dem Tensor $\mathcal{G}$ multipliziert werden.
Der Tensor $\mathcal{G}$ wird folgendermassen definiert:

- $\mathcal{G} = \begin{pmatrix} g_x & 0 & 0 \\ 0 & g_y & 0 \\ 0 & 0 & g_z \end{pmatrix}$ wobei
  g_x = Funktion, welche einen nichtlinearen Output von x in einen linearen Output transformiert
  $g_x = g$(nichtlinearer Messwert x-Achse)

Zusammengefasst ergibt sich dabei folgende Gleichung:

$$\vec{v_j} = \sum_{i=1}^{j} \{\mathcal{A}_i \cdot [\mathcal{G}(\vec{a_i} - \vec{a}_{offset_{ADC_i}}) - \vec{a}_{offset_{g_i}}] \cdot \Delta t_i\} + \vec{v_0} \tag{2.5}$$

Bei Δt handelt es sich, gegeben durch die Hardware, um einen konstanten Faktor für alle i. Zudem wird die Annahme getroffen, dass die Streckmatrix für jeden Zeitpunkt i identisch ist, da es sich um eine Sensor-Eigenschaft handelt welche nicht von der Zeit abhängig ist.

- $\mathcal{A}_i = \mathcal{A}_{i+1}$ für alle i, daraus folgt $\mathcal{A}_i = \mathcal{A}$

- $\mathcal{A} = \begin{pmatrix} \sigma_x & 0 & 0 \\ 0 & \sigma_y & 0 \\ 0 & 0 & \sigma_z \end{pmatrix} = \vec{\vec{\sigma}}$

Im Folgenden wird der Einfachheit halber $\vec{\vec{\sigma}}$ jeweils als Streckfaktor bezeichnet. Die getroffenen Annahmen lassen die Gleichung etwas vereinfachen.

$$\vec{v_j} = \Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} [\mathcal{G}(\vec{a_i} - \vec{a}_{offset_{ADC_i}}) - \vec{a}_{offset_{g_i}}] + \vec{v_0} \tag{2.6}$$

Die Gleichung lässt sich zum besseren Verständnis auch in Vektorschreibweise darstellen.

$$\begin{pmatrix} v_{j_x} \\ v_{j_y} \\ v_{j_z} \end{pmatrix} = \Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} \Bigg\{ \mathcal{G}\Bigg(\begin{pmatrix} a_{i_x} \\ a_{i_y} \\ a_{i_z} \end{pmatrix} - \begin{pmatrix} a_{offset_{ADC_{i_x}}} \\ a_{offset_{ADC_{i_y}}} \\ a_{offset_{ADC_{i_z}}} \end{pmatrix} \Bigg) - \begin{pmatrix} a_{offset_{g_{i_x}}} \\ a_{offset_{g_{i_y}}} \\ a_{offset_{g_{i_z}}} \end{pmatrix} \Bigg\} + \begin{pmatrix} v_{0_x} \\ v_{0_y} \\ v_{0_z} \end{pmatrix} \tag{2.7}$$

Durch eine weitere Integration der Geschwindigkeit $\vec{v_j}$ über die Zeit, lässt sich die zurückgelegte Strecke bzw. die neue Position berechnen.

$$\vec{s_k} = \sum_{j=1}^{k} (\vec{v_j} \cdot \Delta t_j) + \vec{s_0} \tag{2.8}$$

Auch hier ist Δt wiederum, gegeben durch die Hardware, ein konstanter Faktor. Zudem kann $\vec{v_j}$ aus Gleichung 2.6 einsetzen.

$$\vec{s_k} = \Delta t \cdot \sum_{j=1}^{k} \{\Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} [\mathcal{G}(\vec{a_i} - \vec{a}_{offset_{ADC_i}}) - \vec{a}_{offset_{g_i}}] + \vec{v_0}\} + \vec{s_0} \tag{2.9}$$

Bei Δt sowie bei $\vec{\vec{\sigma}}$ und $\vec{v_0}$ handelt es sich bekannterweise um konstante Faktoren, wodurch sich diese Faktoren aus der Summe ziehen lassen und sich die Gleichung etwas vereinfachen lässt.

$$\vec{s_k} = \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \sum_{j=1}^{k} \sum_{i=1}^{j} \{\mathcal{G}(\vec{a_i} - \vec{a}_{offset_{ADC_i}}) - \vec{a}_{offset_{g_i}}\} + \vec{v_0} \cdot \Delta t \cdot k + \vec{s_0} \tag{2.10}$$

Die doppelte Summe lässt sich auch noch kürzen.

$$\vec{s_k} = \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{k} \{[k-i+1] \cdot [\mathcal{G}(\vec{a_i} - \vec{a}_{offset_{ADC_i}}) - \vec{a}_{offset_{g_i}}]\} + \vec{v_0} \cdot \Delta t \cdot k + \vec{s_0} \quad (2.11)$$

Auch diese Gleichung lässt sich zum besseren Verständnis auch in Vektorschreibweise darstellen.

$$\begin{pmatrix} s_{k_x} \\ s_{k_y} \\ s_{k_z} \end{pmatrix} = \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{k} \left\{ \left[k-i+1 \right] \cdot \left[\mathcal{G}\left(\begin{pmatrix} a_{i_x} \\ a_{i_y} \\ a_{i_z} \end{pmatrix} - \begin{pmatrix} a_{offset_{ADC_{i_x}}} \\ a_{offset_{ADC_{i_y}}} \\ a_{offset_{ADC_{i_z}}} \end{pmatrix} \right) - \begin{pmatrix} a_{offset_{g_{i_x}}} \\ a_{offset_{g_{i_y}}} \\ a_{offset_{g_{i_z}}} \end{pmatrix} \right] \right\} + \begin{pmatrix} v_{0_x} \\ v_{0_y} \\ v_{0_z} \end{pmatrix} \cdot \Delta t \cdot k + \begin{pmatrix} s_{0_x} \\ s_{0_y} \\ s_{0_z} \end{pmatrix} \quad (2.12)$$

2.3.2. Einfache Integration der Beschleunigung mit Richtungsinformationen

Für die einfache Integration der Beschleunigung wird von Gleichung 2.15 auf Seite 21 ausgegangen. Zur Berechnung der Position interessiert bei dieser Berechnungsmöglichkeit jedoch nur die Geschwindigkeit selbst, ohne Richtungsinformationen (euklidscher Betrag des Geschwindigkeitsvektors).
Die benötigten Richtungsinformationen können dem Magnetsensor, welcher für jeden Zeitpunkt i die relative Winkelveränderung pro Achse gegenüber den Zeitpunkt $i-1$ liefert, entnommen werden.
Zur Berechnung der Richtungen werden dabei die Drehwinkel α, β und γ und die daraus resultierende Drehmatrix $\mathcal{T}$, wie sie auf Seite 16 vorgestellt wurde, verwendet. Die Winkel α, β und γ verändern sich jedoch laufend, so dass sie für jeden Zeitpunkt i neu berechnet werden müssen. Das genaue Verfahren zur Berechnung dieser Winkel wird zu einem späteren Zeitpunkt vorgestellt.
Die Position lässt sich anschliessend aus dem euklidschen Betrag der Geschwindigkeit $|\vec{v_j}|$, der Zeit sowie der Richtungsinformationen in Form einer Drehmatrix $\mathcal{T}$ berechnen. Aus Gründen der Übersichtlichkeit wird darauf verzichtet, die Drehmatrix $\mathcal{T}$ von Seite 16 nochmals komplett aufzuführen.

$$\vec{s_k} = \sum_{j=1}^{k} (\mathcal{T}_j \cdot |\vec{v_j}| \cdot \Delta t) + \vec{s_0}, \text{ wobei } \mathcal{T}_j = \mathcal{T}(\alpha_j, \beta_j, \gamma_j) \quad (2.13)$$

Zum besseren Verständnis lässt sich diese Gleichung auch in Vektorschreibweise darstellen.

$$\begin{pmatrix} s_{k_x} \\ s_{k_y} \\ s_{k_z} \end{pmatrix} = \sum_{j=1}^{k} \left[\mathcal{T}_j \cdot \left| \begin{pmatrix} v_{j_x} \\ v_{j_y} \\ v_{j_z} \end{pmatrix} \right| \cdot \Delta t \right] + \begin{pmatrix} s_{0_x} \\ s_{0_y} \\ s_{0_z} \end{pmatrix}, \text{ wobei } \mathcal{T}_j = \mathcal{T}(\alpha_j, \beta_j, \gamma_j) \tag{2.14}$$

2.3.3. Nichtlinearitäten und Tensor $\mathcal{G}$

[Tschanz 2005] hat in seiner Arbeit, wie bereits in Kapitel 2.3 angedeutet, Nichtlinearitäten bei den Sensoren, insbesondere den Beschleunigungssensoren, ausgemacht. Diese wurden in den vorherigen Gleichungen jeweils mit dem Tensor $\mathcal{G}$ korrigiert.

Die Nichtlinearitäten der Beschleunigungssensoren sind in Messungen von [Tschanz 2005] (siehe Abbildung 2.3) sehr gut erkennbar.

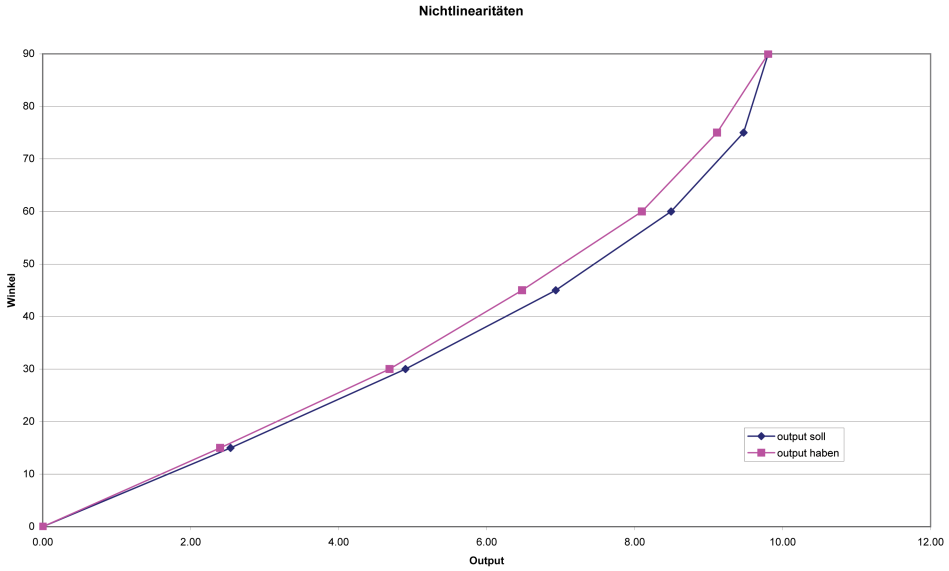

Abbildung 2.3.: Nichtlinearitäten der Beschleunigungssensoren

Die Spezifikationen der Beschleunigungssensoren (vgl. hierzu [AnalogDevices 2004] bzw. Tabelle 3.2), bezeichnen den Fehler durch Nichtlinearitäten jedoch mit $\pm 0.2\%$ des vollen Messbereichs. Bei der Messung von $1g$ entspricht dies $\frac{1}{500}g$. Der ADC (siehe Kapitel 3.3.2 bzw. 2.3) liefert jedoch nur eine Auflösung von ca. 140 Messpunkten pro g bzw. $\frac{1}{140}$.
Die gemessenen Nichtlinearitäten der Beschleunigungssensoren sind also vernachlässigbar klein im Vergleich zur Messauflösung des Sensor-Boards.
Nachdem die Messanlage von [Tschanz 2005] die Winkelbestimmung nicht exakt zugelassen hat, ist anzunehmen, dass die Abweichungen von der Ungenauigkeit der Sensorenausrichtung hergerührt hat.

Auf Grund der obigen Argumentation kann die Annahme getroffen werden, dass der Sensoroutput als linear betrachtet werden kann und demzufolge der Tensor $\mathcal{G}$ sich zur Einheitsmatrix reduziert. Insofern kann darauf verzichtet werden, Tensor $\mathcal{G}$ weiterhin in den Berechnungen zu berücksichtigen.
Gleichung 2.6 lässt sich daher vereinfachen, indem Tensor $\mathcal{G}$ weggelassen wird.

$$\vec{v_j} = \Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} (\vec{a_i} - \vec{a}_{offset_{ADC_i}} - \vec{a}_{offset_{g_i}}) + \vec{v_0} \tag{2.15}$$

Gleiches kann auch für Gleichung 2.11 gemacht werden.

$$\vec{s_k} = \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{k} [(k - i + 1) \cdot (\vec{a_i} - \vec{a}_{offset_{ADC_i}} - \vec{a}_{offset_{g_i}})] + \vec{v_0} \cdot \Delta t \cdot k + \vec{s_0} \tag{2.16}$$

2.4. Paralleles lokales Bezugssystem

Solange sich die Ausrichtung des Sensor-Boards relativ zur Erdoberfläche nicht ändert, also statisch ist, kann von einer Trennung der beiden Offset abgesehen werden. Der Gesamtoffset $\vec{a}_{offset_i}$ kann daher einfach durch eine Interpolation berechnet werden und beinhaltet dabei den ADC-Offset $\vec{a}_{offset_{ADC_i}}$ sowie den g-Offset $\vec{a}_{offset_{g_i}}$ (Erdbeschleunigung).

2.4.1. Doppelte Integration der Beschleunigung

Ausgehend von der allgemeinen Gleichung 2.15 kann durch folgende Annahmen die Gleichung noch etwas vereinfacht werden:

- Der Gesamtoffset besteht aus dem ADC- und g-Offset
  $\vec{a}_{offset_i} := \vec{a}_{offset_{ADC_i}} + \vec{a}_{offset_{g_i}}$
- Der Gesamtoffset ist zu jedem Zeitpunkt identisch
  $\vec{a}_{offset_i} = \vec{a}_{offset_{i+1}}$ für alle i, daraus folgt $\vec{a}_{offset_i} = \vec{a}_{offset}$

$$\vec{v_j} = \Delta t \cdot \vec{\vec{\sigma}} \cdot (\sum_{i=1}^{j} \vec{a_i} - \vec{a}_{offset}) + \vec{v_0} \tag{2.17}$$

Durch die getroffene Annahme, dass der Offset über alle i konstant ist, kann dieser aus der Summe genommen werden.

$$\vec{v_j} = \Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} \vec{a_i} - \Delta t \cdot \vec{\vec{\sigma}} \cdot j \cdot \vec{a}_{offset} + \vec{v_0} \tag{2.18}$$

Die Gleichung lässt sich dabei zum besseren Verständnis auch in Vektorschreibweise darstellen.

$$\begin{pmatrix} v_{j_x} \\ v_{j_y} \\ v_{j_z} \end{pmatrix} = \Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} \begin{pmatrix} a_{i_x} \\ a_{i_y} \\ a_{i_z} \end{pmatrix} - \Delta t \cdot \vec{\vec{\sigma}} \cdot j \cdot \begin{pmatrix} a_{offset_x} \\ a_{offset_y} \\ a_{offset_z} \end{pmatrix} + \begin{pmatrix} v_{0_x} \\ v_{0_y} \\ v_{0_z} \end{pmatrix} \tag{2.19}$$

Eine weitere Integration, wie auf Seite 18, liefert wiederum direkt die Position.

$$\vec{s_k} = \sum_{j=1}^{k} (\Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} \vec{a_i} - \Delta t \cdot \vec{\vec{\sigma}} \cdot j \cdot \vec{a}_{offset} + \vec{v_0}) \cdot \Delta t + \vec{s_0} \tag{2.20}$$

Auch in diesem Fall handelt es sich bei Δt um einen konstanten Faktor. Zudem können die Summen noch umgeformt werden, was weitere Vereinfachungen zulässt.

$$\vec{s_k} = \Delta t \cdot \left[\sum_{j=1}^{k} (\Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} \vec{a_i}) - \sum_{j=1}^{k} (\Delta t \cdot j \cdot \vec{\vec{\sigma}} \cdot \vec{a}_{offset}) + k \cdot \vec{v_0} \right] + \vec{s_0} \tag{2.21}$$

Der Offset $\vec{a}_{offset}$, Δt sowie $\vec{\vec{\sigma}}$ sind Konstanten, wodurch sich die Summe kürzen lässt. Zudem kann der ganzer Ausdruck noch ausmultipliziert werden.

$$\vec{s_k} = \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \sum_{j=1}^{k} \sum_{i=1}^{j} \vec{a_i} - \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \frac{(k+1) \cdot k}{2} \cdot \vec{a}_{offset} + k \cdot \vec{v_0} \cdot \Delta t + \vec{s_0} \tag{2.22}$$

Schliesslich lässt sich die doppelte Summe noch vereinfachen.

$$\vec{s_k} = \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{k} [(k-i+1) \cdot \vec{a_i}] - \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \frac{(k+1) \cdot k}{2} \cdot \vec{a}_{offset} + k \cdot \vec{v_0} \cdot \Delta t + \vec{s_0} \tag{2.23}$$

Auch diese Gleichung lässt sich zum besseren Verständnis in Vektorschreibweise darstellen.

$$\begin{aligned} \begin{pmatrix} s_{k_x} \\ s_{k_y} \\ s_{k_z} \end{pmatrix} &= \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{k} \left[(k-i+1) \cdot \begin{pmatrix} a_{i_x} \\ a_{i_y} \\ a_{i_z} \end{pmatrix} \right] \\ &\quad - \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \frac{(k+1) \cdot k}{2} \cdot \begin{pmatrix} a_{offset_x} \\ a_{offset_y} \\ a_{offset_z} \end{pmatrix} \\ &\quad + k \cdot \begin{pmatrix} v_{0_x} \\ v_{0_y} \\ v_{0_z} \end{pmatrix} \cdot \Delta t + \begin{pmatrix} s_{0_x} \\ s_{0_y} \\ s_{0_z} \end{pmatrix} \end{aligned} \tag{2.24}$$

2.4.2. Einfache Integration der Beschleunigung mit Richtungsinformationen

Für die Methode der einfachen Integration kann von der allgemeinen Gleichung 2.13 auf Seite 19 ausgegangen werden.
Zu beachten ist jedoch, dass auch hier für die Berechnung der Geschwindigkeit $\vec{v_j}$ der Offset nicht getrennt werden muss, sondern als ein einziger Offset a_{offset} betrachtet werden kann (vgl. Gleichung 2.17).
Dabei kann wieder von der Drehmatrix $\mathcal{T}$, welche auf Seite 16 zu finden ist, ausgegangen werden.

$$\vec{s_k} = \sum_{j=1}^{k} (\mathcal{T}_j \cdot |\vec{v_j}| \cdot \Delta t) + \vec{s_0}, \text{ wobei } \mathcal{T}_j = \mathcal{T}(\alpha_j, \beta_j, \gamma_j)$$

Beziehungsweise in Vektorschreibweise

$$\begin{pmatrix} s_{k_x} \\ s_{k_y} \\ s_{k_z} \end{pmatrix} = \sum_{j=1}^{k} \left[\mathcal{T}_j \cdot \left| \begin{pmatrix} v_{j_x} \\ v_{j_y} \\ v_{j_z} \end{pmatrix} \right| \cdot \Delta t \right] + \begin{pmatrix} s_{0_x} \\ s_{0_y} \\ s_{0_z} \end{pmatrix}, \text{ wobei } \mathcal{T}_j = \mathcal{T}(\alpha_j, \beta_j, \gamma_j)$$

Zum besseren Verständnis kann Gleichung 2.18 noch eingesetzt werden.

$$\begin{aligned} \vec{s_k} &= \sum_{j=1}^{k} (\mathcal{T}_j \cdot |\Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} \vec{a_i} - \Delta t \cdot \vec{\vec{\sigma}} \cdot j \cdot \vec{a}_{offset} + \vec{v_0}| \cdot \Delta t) + \vec{s_0} \\ &\quad \text{wobei } \mathcal{T}_j = \mathcal{T}(\alpha_j, \beta_j, \gamma_j) \end{aligned} \tag{2.25}$$

2.5. Frei bewegliches lokales Bezugssystem

Im Gegensatz zum vorherigen Unterkapitel, wird nun auch davon ausgegangen, dass sich die Ausrichtung des Sensor-Boards relativ zur Erdoberfläche verändern kann. Somit kann der Offset nicht mehr konstant betrachtet werden, da die Erdbeschleunigung nicht mehr zu jedem Zeitpunkt i mit den gleichen Beschleunigungsanteilen auf die Sensoren wirkt.
Dank den sehr genauen Magnetsensoren, welche auf dem Board zur Verfügung stehen, besteht die Möglichkeit, Veränderungen in der Ausrichtung des Boards (Drehungen), ausgehend von der Anfangslage, relativ zur Erdoberfläche in Form von Winkel zu bestimmen.
Es besteht daher die Möglichkeit, sämtliche nicht drehvarianten Variabeln mit einer Drehmatrix (vgl. hierzu Kapitel 2.2) auf die ursprüngliche Lage zu korrigieren. Die Erdbeschleunigung, welche auf die Beschleunigungssensoren wirkt, wird daher, in Bezug auf das globale Bezugssystem, als Vektor $\vec{g}$ definiert. Der bisherige Offset_g wird neu als $\text{Offset}_g = \mathcal{T} \cdot \vec{g}$ definiert.
Somit lässt sich Vektor $\vec{g}$ über alle i als konstant bestimmen, wobei die Multiplikation mit der Drehmatrix $\mathcal{T}$ den effektiven Offset_g ergibt.
Dadurch kann jedoch Offset_g nicht mehr über alle i als konstant betrachtet werden, wodurch dieser nicht mehr zusammen mit dem Offset_{ADC} als Gesamtoffset betrachtet werden kann.

2.5.1. Doppelte Integration der Beschleunigung

Mittels Drehmatrix $\mathcal{T}$ lässt sich der globale Vektor $\vec{g}$ in das lokale Bezugssystem führen. Zusätzlich werden folgende Annahmen getroffen:

- Als Drehmatrix gelte $\mathcal{T}_i$ (vgl. hierzu auch Kapitel 2.2)
- $\vec{a}_{offset_{g_i}} = \mathcal{T}_i \cdot \vec{g}$, wobei $\vec{g} = \text{const.}$ über alle i
- $\vec{a}_{offset_{ADC_i}} = \vec{a}_{offset_{ADC_{i+1}}}$ für alle i, daraus folgt $\vec{a}_{offset_{ADC_i}} = \vec{a}_{offset_{ADC}}$
- Winkel α bezeichne die Drehung um die x-Achse gegenüber der Ursprungslage
- Winkel β bezeichne die Drehung um die y-Achse gegenüber der Ursprungslage
- Winkel γ bezeichne die Drehung um die z-Achse gegenüber der Ursprungslage

Ausgehend von der allgemeinen Gleichung 2.15 ergibt sich folgende Gleichung unter den getroffenen Annahmen.

$$\vec{v_j} = \Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} (\vec{a_i} - \vec{a}_{offset_{ADC}} - \mathcal{T}_i \cdot \vec{g}) + \vec{v_0}, \text{ wobei } \mathcal{T}_i = \mathcal{T}_i(\alpha_i, \beta_i, \gamma_i) \tag{2.26}$$

Eine weitere Integration, ausgehend von Gleichung 2.16 auf Seite 21, liefert schliesslich direkt die Position.

$$\begin{aligned} \vec{s_k} &= \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{k} [(k - i + 1) \cdot (\vec{a_i} - \vec{a}_{offset_{ADC}} - \mathcal{T}_i \cdot \vec{g})] + \vec{v_0} \cdot \Delta t \cdot k + \vec{s_0} \\ &\quad \text{wobei } \mathcal{T}_i = \mathcal{T}_i(\alpha_i, \beta_i, \gamma_i) \end{aligned} \tag{2.27}$$

Zum einfacheren Verständnis lässt sich die Gleichung auch in Vektorschreibweise darstellen.

$$\begin{aligned} \begin{pmatrix} s_{k_x} \\ s_{k_y} \\ s_{k_z} \end{pmatrix} &= \Delta t^2 \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{k} \left\{ \left[k - i + 1 \right] \cdot \left[\begin{pmatrix} a_{i_x} \\ a_{i_y} \\ a_{i_z} \end{pmatrix} - \begin{pmatrix} a_{offset_{ADC_x}} \\ a_{offset_{ADC_y}} \\ a_{offset_{ADC_z}} \end{pmatrix} - \right. \right. \\ &\quad \left. \left. \mathcal{T}_v \cdot \begin{pmatrix} g_x \\ g_y \\ g_z \end{pmatrix} \right] \right\} + \begin{pmatrix} v_{0_x} \\ v_{0_y} \\ v_{0_z} \end{pmatrix} \cdot \Delta t \cdot k + \begin{pmatrix} s_{0_x} \\ s_{0_y} \\ s_{0_z} \end{pmatrix} \end{aligned} \tag{2.28}$$

2.5.2. Einfache Integration der Beschleunigung mit Richtungsinformationen

Die einfache Integration gestaltet sich wiederum relativ einfach. Es kann dabei für die Geschwindigkeit von Gleichung 2.26 ausgegangen werden. Die Position lässt sich anschliessend ausgehend von Gleichung 2.13 auf Seite 19 wie folgt errechnen.

$$\vec{s_k} = \sum_{j=1}^{k} (\mathcal{T}_j \cdot |\vec{v_j}| \cdot \Delta t) + \vec{s_0}, \text{ wobei } \mathcal{T}_j = \mathcal{T}_j(\alpha_j, \beta_j, \gamma_j)$$

Zum besseren Verständnis kann Gleichung 2.26 noch eingesetzt werden.

$$\begin{aligned} \vec{s_k} &= \sum_{j=1}^{k} [\mathcal{T}_j \cdot |\Delta t \cdot \vec{\vec{\sigma}} \cdot \sum_{i=1}^{j} (\vec{a_i} - \vec{a}_{offset} - \mathcal{T}_i \cdot \vec{g}) + \vec{v_0}| \cdot \Delta t] + \vec{s_0} \\ &\quad \text{wobei } \mathcal{T}_j = \mathcal{T}_j(\alpha_j, \beta_j, \gamma_j) \text{ und } \mathcal{T}_i = \mathcal{T}_i(\alpha_i, \beta_i, \gamma_i) \end{aligned} \tag{2.29}$$

3. Verwendete Hard- und Software

3.1. Übersicht

Im vorherigen Kapitel wurde eingehend auf die theoretischen Berechnungsmöglichkeiten einer Positionsangabe eingegangen. In diesem Kapitel soll nun die in den folgenden Kapitel verwendete Hard- und Software vorgestellt werden.
Die verwendete Hardware besteht dabei aus zwei Primärkomponenten: Einem handelsüblichen Palm Treo sowie einem von Peter Vorburger entwickelten Sensor-Board.
Zu beachten gilt es dabei sicherlich, dass es sich bei dieser Diplomarbeit um keine Real-Time Implementierung der vorgestellten Algorithmen handelt. Sämtliche Daten werden offline in MATLAB berechnet, nach einer Datenaquisition mit der vorher genannten Hardware.

3.2. Palm Treo 650

Zur Datenaufnahme wurde ein handelsübliches Smartphone vom Typ Palm Treo 650 verwendet. Dank der relativ einfachen Palm-Schnittstelle (auf Basis RS-232) ist eine direkte Kommunikation zwischen dem Smartphone und dem Sensor-Board möglich.
Besonders hervorzuheben ist dabei die handliche Grösse des Geräts, wodurch realistische Testszenarien und Experimente, mit Weitblick auf einen reellen Einsatz, möglich sind.
Auf dem Palm wurde dabei die Terminalsoftware CS Online 2.1.0 von Conklin Systems[4] verwendet. Diese Terminalsoftware erlaubt es via RS-232-Schnittstelle, Befehle an das Sensor-Board zu senden sowie die vom Board ausgegebenen Daten in Dateien abzuspeichern. Als Datenträger wurden dabei handelsübliche SD-Karten mit einer Grösse von 2GB verwendet, welche vom Palm Treo als Datenspeicher unterstützt werden. Bei den SD-Karten musste jedoch auf deren maximal mögliche Schreibgeschwindigkeit geachtet werden, damit die Daten auch fehlerfrei in der gegebenen Zeit gespeichert werden können.

[4] Weitere Informationen zu CS Online sind abrufbar unter
http://www.conklinsystems.com/palm/online.php

Abbildung 3.1.: Palm Treo 650

3.3. Sensor-Board

In dieser Diplomarbeit wurde der dritte Prototyp des Sensor-Board von Peter Vorburger verwendet. Vorherige Prototypen wurden bereits von anderen Diplomanden wie [Tschanz 2005] verwendet, wobei die in früheren Arbeiten aufgetauchten Mängel auf die dritte Version hin behoben wurden. Dadurch steht nun eine leistungsfähige und zuverlässige Hardware zur Verfügung.
Das Sensor-Board der dritten Generation besteht aus folgenden Primärkomponenten:

- Ein Prozessor mit integriertem ADC
- Ein Temperatursensor vom Typ Analog Devices AD2103
- Zwei Beschleunigungssensoren vom Typ Analog Devices ADXL320
- Ein Magnetsensor vom Typ Honeywell HMC1053
- Zwei Winkelbeschleunigungssensoren vom Typ NEC TOKIN CG-L53
- Ein Winkelbeschleunigungssensor vom Typ NEC TOKIN CG-L43
- Drei Gassensoren
- Eine Palm- bzw. RS232-Schnittstelle

In den folgenden Unterkapiteln werden die für diese Arbeit relevanten Komponenten noch etwas näher vorgestellt.

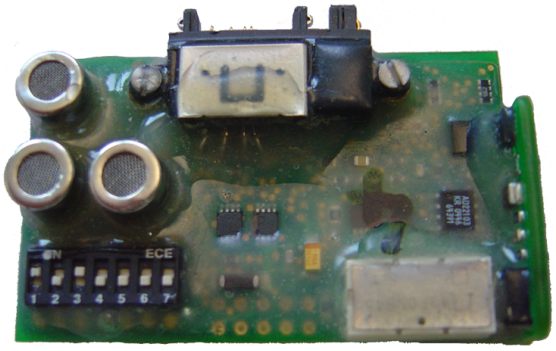

Abbildung 3.2.: Sensor-Board (Ansicht von oben)

3.3.1. Prozessor

Der Prozessor ist das eigentliche Herzstück des Sensor-Boards. Er übernimmt die gesamte Steuerung und ermöglicht die Umwandlung der analogen Daten (Spannungen) von den Sensoren zu digitalen Daten (Messpunkte) welche über eine RS-232-Schnittstelle übertragen werden.
Für die vorliegende Arbeit ist jedoch nur der so genannte Analog Digital Wandler, kurz ADC, von weiterem Interesse, weshalb im folgenden Unterkapitel explizit auf den ADC eingegangen wird.

3.3.2. Analog Digital Wandler (ADC)

Sämtliche Sensoren liefern ihre Messresultate als Spannung in Volt (analoges Signal). So entspricht eine durch den Temperatursensor gemessene Raumtemperatur von 20°C beispielsweise einer Spannung von 1.25V.
Die Ausgangsspannung der Sensoren, welche einem analogen Signal entspricht, muss für die weitere Verarbeitung in ein digitales Signal umgewandelt werden. Diese Aufgabe wird im vorliegenden Fall durch den ADC übernommen.
Der auf dem Sensor-Board eingesetzte ADC arbeitet dabei mit 12Bit wodurch sich $2^{12} = 4096$ mögliche Messpunkte (digitales Signal) ergeben. Der Eingangsspannungsbereich (analoges Signal) beträgt 0 bis 2.5V, wodurch sich die Spannung pro Messpunkt errechnen lässt.

$$\frac{2.5V}{4096 \text{ Messpunkte}} = \frac{0.6103515625mV}{\text{Messpunkt}} \tag{3.1}$$

Das Sensor-Board liefert somit über die Palm- bzw. RS232-Schnittstelle Messwerte zwischen 0 und 4096 für jeden Sensor.

Die Daten müssen selbstverständlich noch interpretiert und verarbeitet werden. Aus einem Messwert von 1044 Messpunkten, welche beispielsweise der Temperatursensor via RS-232-Schnittstelle liefert, kann nicht direkt die gewünschte Information, in diesem Fall die Temperatur in °C, abgelesen werden.
Zusätzlich ist darauf hinzuweisen, dass der ADC keine Minuswerte kennt. So entspricht beispielsweise eine Beschleunigung von $0\frac{m}{s^2}$ nicht 0 Messpunkten. Diese Eigenschaft wurde bereits im Theorieteil angedeutet, als darauf hingewiesen wurde, dass jeweils ein Offset von den Beschleunigungssensordaten abgezogen werden muss.
Auf die Interpretation bzw. Kalibrierung der Sensoren wird im nächsten Kapitel noch genauer eingegangen.

3.3.3. Temperatursensor

Auf dem Sensor-Board ist auch ein Temperatursensor vom Typ AD22103 untergebracht, welcher Temperaturen von 0 bis 100°C zuverlässig messen kann. Obwohl die Temperatur

Parameter	**Min**	**Durchschnitt**	**Max**
Messbereich	0°C		+100°C
Möglicher Messfehler der Temperatur		±0.75°C	±2.5°C

Tabelle 3.1.: Technische Daten des Temperatursensors AD22103 [AnalogDevices 1995]

auf den ersten Blick relativ wenig mit Navigationsinformationen zu tun hat, so hat sie doch einen Einfluss auf die Genauigkeit der verwendeten Hardware, was wiederum von Interesse ist.

Einfluss der Temperatur auf die Sensordaten

Im vorliegenden Fall ist der Temperatursensor besonders für die Korrektur der temperaturbedingten Ungenauigkeiten bei den (Winkel-)Beschleunigungssensoren interessant, da die Umgebungs- bzw. Betriebstemperatur immer auch einen Einfluss auf die Genauigkeit der Sensoren hat.
Könnte man von einer konstanten Umgebungstemperatur ausgehen, so würde der Temperatureinfluss in der Eichung bzw. Offset mit berücksichtigt sein. Sind die Sensoren jedoch wechselnden Umgebungs- und Betriebstemperaturen ausgesetzt, so muss dies berücksichtigt werden, um Ungenauigkeiten bzw. Fehlabweichungen zu vermeiden und zu kompensieren. Dank dem auf dem Sensor-Board untergebrachten Temperatursensor lässt sich die Umgebungstemperatur jederzeit bestimmen und mit einem entsprechenden Algorithmus die temperaturbedingten Fehler korrigieren.
Das genaue Verfahren zur Korrektur von temperaturabhängigen Fehler wird in Kapitel 4 näher vorgestellt.

3.3.4. Beschleunigungssensoren

Die auf dem Sensor-Board eingesetzten Beschleunigungssensoren vom Typ Analog Devices ADXL320 erlauben die Messung von Beschleunigungen bis zu ±5g auf allen drei Achsen (x, y und z). Dabei muss beachtet werden, dass die Beschleunigungssensoren auch die Erdbeschleunigung wahrnehmen und in die Messresultate einbeziehen. Eine entsprechende Korrektur muss somit vorgenommen werden, wie sie bereits im Theorieteil mit einem Offset_g vorgesehen wurde.
Zusätzlich ist auch zu beachten, dass externe Einflüsse wie Temperaturschwankungen

Parameter	**Wert**
Messbereich	±5g
Sensitivität (durchschnittlich)	174mV/g
Sensitivitätswechsel durch die Temperatur	0.01%/°C
Nichtlinearität über den gesamten Messbereich	0.2%

Tabelle 3.2.: Technische Daten der Beschleunigungssensoren ADXL320 ($V_s = 3V$) [AnalogDevices 2004]

das Messergebnis der Sensoren beeinflussen kann (vgl. hierzu auch [AnalogDevices 2004] bzw. Kapitel 3.3.3). Auch hierfür sollte eine entsprechende Korrektur der Messdaten vorgenommen werden. Auf die Korrektur des Temperatureinflusses wird in Kapitel 4 nochmals detailliert eingegangen.

3.3.5. Magnetsensor

Das Sensor-Board beherbergt auch einen 3-Achsen Magnetsensor vom Typ Honeywell HMC1053. Mittels Magnetsensor lässt sich eine Änderung in der Orientierung des Sensor-

Parameter	**Min**	**Durchschnitt**	**Max**
Betriebstemperatur	-40°C		+125°C
Feuchtigkeit			85% bei +85°C
Genauigkeit	0.8 mV/V/gauss	1.0 mV/V/gauss	1.2 mV/V/gauss
Auflösung		120 μgauss	

Tabelle 3.3.: Technische Daten des Magnetsensors HMC1053 [Honeywell 2003]

Boards feststellen. In diesem Fall ist dabei nicht primär die Ausrichtung des Sensor-Board gegen Norden von Interesse (bzw. in welcher Kompassorientierung das Board liegt) sondern welche Winkelveränderungen zwischen zwei Zeitschritten aufgetreten sind (Differenz der Winkel).

Mit den Winkelveränderungsinformationen lassen sich dabei die Winkel α, β und γ herleiten, welche für die im Theorieteil eingeführte Drehmatrix notwendig sind.
Besonders erwähnenswert ist dabei, dass keine Integration der Daten vorgenommen werden muss zur Bestimmung der Winkel. Dies macht den Sensor zu einer äusserst genauen Informationsquelle.
Unglücklicherweise hat jedoch auch diese Medaille eine Kehrseite. So existieren im täglichen Leben diverse Magnetfelder, welche die Magnetsensoren stören können. Ein Störfilter für den Magnetsensor, welche diese Störfelder erkennen könnte, wäre daher äusserst hilfreich.

3.3.6. Winkelbeschleunigungssensoren

Die auf dem Sensor-Board untergebrachten Winkelbeschleunigungssensoren, auch bekannt als Gyroskope, sind in der Lage, Winkelbeschleunigungen zu messen. Aus den Daten lassen sich dabei, analog zum Magnetsensor, Winkelveränderungen berechnen. Im Gegensatz zum Magnetsensor unterliegen die Winkelbeschleunigungssensoren jedoch keinen Störfelder. Trotzdem haben sie einen entscheidenden Nachteil gegenüber dem Magnetsensor: zur Berechnung der Winkelveränderungen ist eine doppelte Integration nötig.
Wie bereits im Theorieteil angesprochen, unterliegen mehrfache Integrationen immer

Parameter	**Wert**
Genauigkeit	0.66 mV/Grad/s
Ausgangsspannung bei keiner Winkelbeschleunigung	-5 bis +75 mV
Sensitivitätswechsel durch die Temperatur	$\pm$15%

Tabelle 3.4.: Technische Daten der Winkelbeschleunigungssensoren CG-L53/CG-L43 [NEC/TOKIN 2001] [NEC/TOKIN 2005a]

auch einem Integrationsdrift, was zu Ungenauigkeiten führen kann.
Auch bei den Winkelbeschleunigungssensoren muss jedoch auf temperaturbedingte Fehler hingewiesen werden, welche bei Temperaturschwankungen auftreten können. Die Winkelbeschleunigungsdaten müssen daher ebenso wie die Daten der Beschleunigungssensoren entsprechend korrigiert werden.
Gerade aber durch die Resistenz gegenüber magnetischen Störfeldern könnten die Winkelbeschleunigungssensoren sehr gut als Störfilter für den Magnetsensor eingesetzt werden, wie es auch bereits [Tschanz 2005] vorschlägt.

3.4. MATLAB

MATLAB ist eine kommerzielle mathematische Software der Firma The MathWorks zur Lösung diverser mathematischer Probleme und zur grafischen Darstellung der Ergebnisse. Dabei rechnet MATLAB hauptsächlich mit Matrizen was auch den Namen begründet[5].
Wie bereits in der Einführung angemerkt, wurden in dieser Arbeit keine online Berechnungen auf der Hardware vorgenommen. Sämtliche Berechnungen dieser Arbeit wurden Offline in MATLAB vorgenommen. Die Hardware wurde dabei lediglich dazu verwendet, die Sensordaten in einem realistischen Testszenario aufzuzeichnen.
Für die Berechnungen wurden dabei entsprechende Algorithmen in MATLAB abgebildet (vgl. hierzu Kapitel 5).
MATLAB ist für die entsprechenden Berechnungen insofern optimal, da für die Berechnungen grösstenteils Matrizenoperationen notwendig sind, worauf MATLAB bekanntermaßen spezialisiert ist.
Weitere Informationen zu MATLAB finden sich zudem auf der Homepage[6] von The MathWorks.

[5] MATLAB ist die Abkürzung von MATrix LABoratory
[6] http://www.mathworks.com/products/matlab/

4. Kalibrierung der Sensoren

4.1. Übersicht

In den vorherigen Kapitel wurde die Kalibrierung bereits mehrfach angesprochen. In diesem Kapitel soll deshalb aufgezeigt werden, wie die Sensoren für den Einsatz kalibriert werden. Dabei stellt sich vor allem eine Frage in den Vordergrund: Wie kann aus Messwerten (0 bis 4096 Messpunkte) die benötigte physikalische Grösse berechnet werden?
Die Kalibrierung ist dabei selbstverständlich sensorspezifisch und variiert zwischen den Sensoren erheblich, wodurch sämtlichen relevanten Sensoren ein eigenes Unterkapitel gewidmet ist.

4.2. Temperatursensor

Der Temperatursensor benötigt keine eigentliche Kalibrierung. Für die Berechnung der Temperatur, basierend auf den Messwerten, kann die Transferfunktion aus [AnalogDevices 1995] verwendet werden.
Für die Berechnungen mit der Transferfunktion ist jedoch die Kenntnis der Spannungswerte, insbesondere die Versorgungsspannung (Vs) sowie das Bezugspotential (GND), nötig.
Die Spannungswerte wurden deshalb mit einem Voltmeter gemessen, wobei die in Tabelle 4.1 ersichtlichen Spannungswerte resultierten. Dabei ist für die folgenden Berechnungen

Grösse	**Wert**
Einspeisung (Supply)	3.21V
Bezugspotential (GND)	0.613V
Versorgungsspanung (Vs)	3.21V - 0.613V = 2.597V

Tabelle 4.1.: Gemessene Spannungsversorgung Temperatursensor AD22103

zu beachten, dass keine Versorgungsspannung von 3.3V vorliegt.

Anhand der Transferfunktion aus [AnalogDevices 1995] lässt sich aus der Ausgangsspannung des Sensors die Temperatur errechnen. Dabei ist auch zu berücksichtigen, dass sämtliche Spannungswerte in Volt vorliegen müssen.

$$V_{out} = \frac{V_s}{3.3V} \cdot (0.25V + \frac{0.028V}{°C} \cdot T_A) \tag{4.1}$$

Nach der Temperatur aufgelöst, ergibt dies folgende Gleichung:

$$T_A \cdot °C = \frac{V_{out} - \frac{V_s}{3.3V} \cdot 0.25V}{\frac{V_s}{3.3V} \cdot 0.028V} \tag{4.2}$$

Nun lassen sich noch die gemessenen Werte aus Tabelle 4.1 einsetzen und die Gleichung dadurch etwas vereinfachen.

$$T_A \cdot °C = \frac{V_{out} - 0.196742}{0.02203515} \tag{4.3}$$

Nun kann anhand des Sensoroutputs in Messpunkten die aktuelle Temperatur berechnet werden. Dabei müssen die gemessenen Messpunkte anhand von Gleichung 3.1 zurück in Volt umgerechnet werden. Unbedingt zu beachten ist dabei auch, dass von der errechneten Spannung noch das Bezugspotential aus Tabelle 4.1 abgezogen werden muss, bevor die Umrechnung mit Gleichung 4.3 vorgenommen werden kann.

4.2.1. Beispiel zur Berechnung der Temperatur

Die Berechnung lässt sich an einem Beispiel verdeutlichen. Bei einer Messung am 23. Dezember 2005 lieferte der Temperatursensor ein durchschnittliches Messresultat von 2282.93 Messpunkten.
Als Erstes wird die ausgegebene Spannung ($V_{out_{brutto}}$) des Temperatursensors, ausgehend von Gleichung 3.1, berechnet.

$$2282.93 \text{ Messpunkte} \cdot \frac{0.6103515625mV}{\text{Messpunkt}} = 1.3933V \tag{4.4}$$

Nun muss zur Berechnung von V_{out} noch das Bezugspotential gemäss Tabelle 4.1 abgezogen werden.

$$V_{out} = 1.3933V - 0.613V = 0.78V \tag{4.5}$$

Schliesslich kann nun mit Gleichung 4.3 die gemessene Temperatur errechnet werden.

$$\frac{0.78 - 0.196742}{0.02203515} = 26.48°C \tag{4.6}$$

4.3. Beschleunigungssensoren

Die Kalibrierung der Beschleunigungssensoren stellt ein Spezialfall dar. Im Gegensatz zu [Tschanz 2005] wird in dieser Arbeit komplett darauf verzichtet, die Beschleunigungssensoren vorgängig zu kalibrieren. Die Kalibrierung soll durch die Interpolation online erfolgen wobei zwei Kalibrierungswerte bestimmt werden:

- **Offset**
  Mit dem Offset wird die Anzahl Messpunkte bestimmt, um welche der Nullwert auf der Skala von 0 bis 4096 verschoben ist. Je nach Szenario (vgl. Kapitel 5) wird dabei auch die Erdbeschleunigung mit berücksichtigt.

- **Streckfaktor**
  Der Streckfaktor repräsentiert den Umrechnungsfaktor zwischen Messpunkten und physikalischer Grösse $\frac{m}{s^2}$.

In dieser Arbeit wird bewusst auf eine Online-Kalibrierung Wert gelegt. Durch die Online-Kalibrierung bei der Interpolation sollen Kalibrierungsfehler vermieden werden, wodurch der Integrationsdrift bei der Integration sich nicht in solch grossem Ausmass bemerkbar machen sollte, wie bei [Tschanz 2005] (vgl. hierzu auch Abbildung 1.5 in Kapitel 1).

4.3.1. Korrektur des Temperatureinflusses

Bereits in Kapitel 3.3.3 wurde der Einfluss der Temperatur auf die Sensorinformationen angedeutet. Besonders die Beschleunigungssensoren sprechen auf eine sich verändernde Umgebungstemperatur mit Abweichungen an. Der Hersteller beziffert diese Abweichungen mit 0.01%/°C (vgl. Tabelle 3.2 auf Seite 31).
Dieser Einfluss kann sehr deutlich in Abbildung 4.1 beobachtet werden: Mit steigender Temperatur sinkt die mit den Beschleunigungssensoren gemessene Erdbeschleunigung. Auf der Abbildung ist eine steigende Temperatur von rund 20.2°C auf rund 25.2°C zu sehen. Gleichzeitig sinkt der Sensoroutput des Beschleunigungssensors, bei gleich bleibender effektiver Beschleunigung, von 2129 auf 2121 Messpunkte. Zur besseren Visualisierung wurde für diese Abbildung ein Smooth (über 100 Messpunkte) über die gesamten

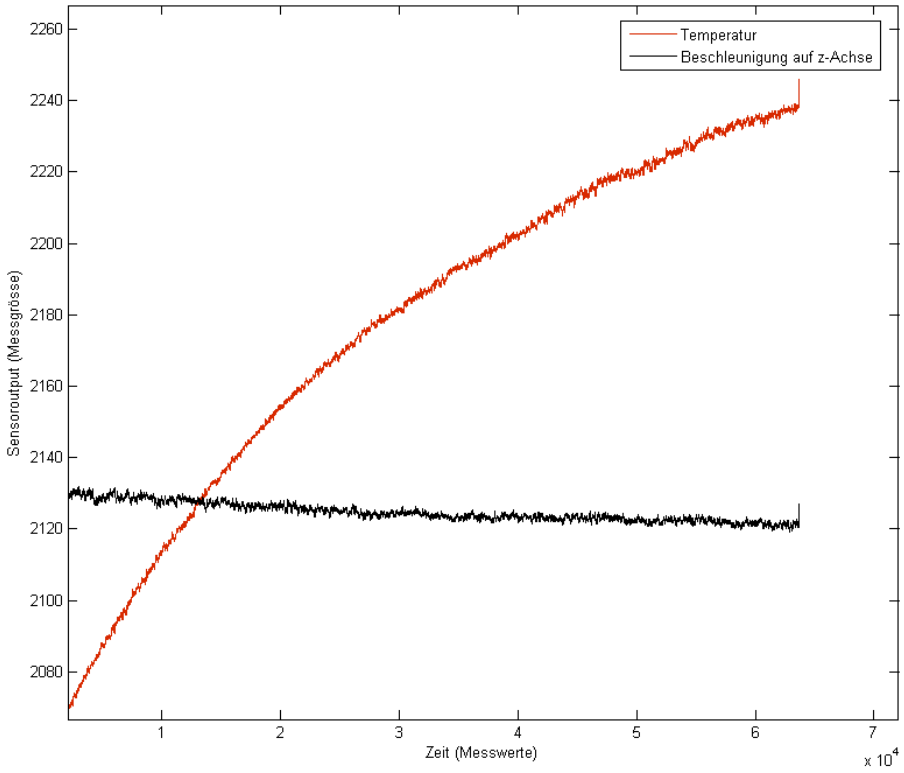

Abbildung 4.1.: Einfluss der Temperatur auf die Beschleunigungssensoren

Daten angewendet.
Zur Vermeidung dieser temperaturbedingten Abweichungen muss deshalb eine Bereinigung der Beschleunigungssensordaten vorgenommen werden.
Für die Korrektur der temperaturbedingten Abweichungen wird von einem linearen Zusammenhang zwischen Temperaturveränderung und Veränderung des Sensoroutputs ausgegangen. Zudem wird die Annahme getroffen, dass die Veränderungen des Sensoroutputs gleichzeitig mit einer Temperaturveränderung auftritt. Diese Annahme bestätigt sich auch bei entsprechenden Messungen wie sie beispielsweise in Abbildung 4.2 erkenntlich ist.
Die Temperaturveränderung trat in diesem Fall bei rund 4000 Messwerten auf, indem die Hardware in einen Kühlschrank gelegt wurde. In Abbildung 4.3 wurde der Beschleunigungssensoroutput zur besseren Darstellung hoch skaliert, damit die unmittelbare Reaktion noch besser zum Vorschein kommt.

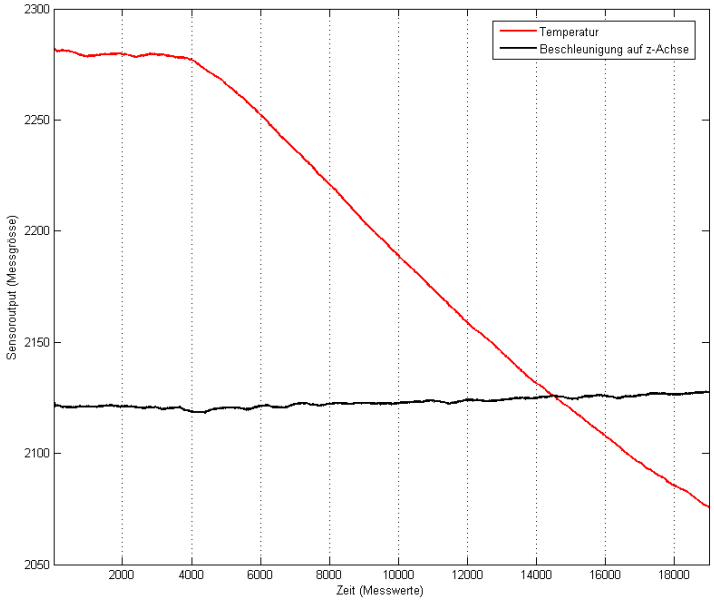

Abbildung 4.2.: Unverzögerte Reaktion des Beschleunigungssensoroutputs bei einer Temperaturveränderung

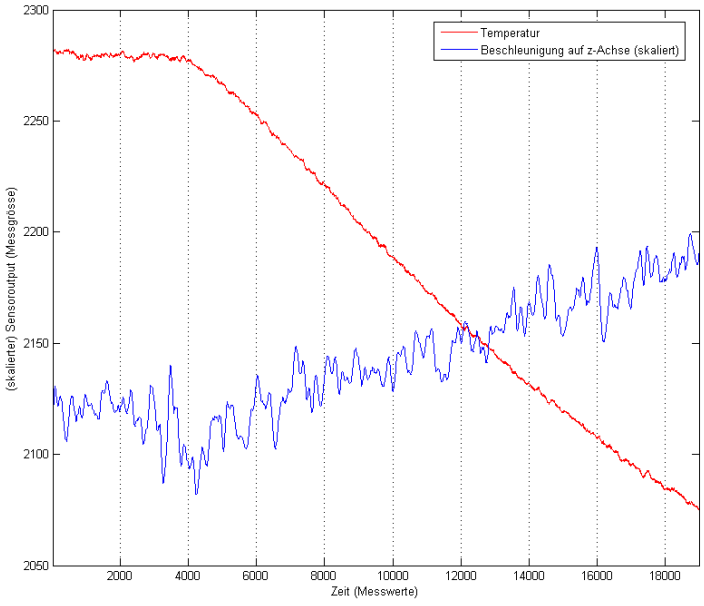

Abbildung 4.3.: Unverzögerte Reaktion des Beschleunigungssensoroutputs bei einer Temperaturveränderung (Beschleunigungsdaten skaliert)

Mit den vorher getroffenen Annahmen kann eine Korrekturgleichung aufgestellt werden, welche einen Korrekturfaktor beinhaltet (vgl. Gleichung 4.7).

$$a_{corr} = a_T \cdot (T_A - T_{Eichung}) \cdot a + a \tag{4.7}$$

Für die Berechnung des Korrekturfaktors a_T muss die Eichungstemperatur $T_{Eichung}$ festgelegt werden, welche vom Temperatursensor bei der Eichung gemessen wird. Anschliessend ist für eine Korrektur nur noch die Temperaturdifferenz zur Eichungstemperatur von Interesse. Die Eichung der Beschleunigungssensoren kann dabei relativ zur Eichungstemperatur vorgenommen werden. Sämtliche temperaturbedingten Abweichungen können anschliessend anhand der Korrekturgleichung beseitigt werden, indem die Beschleunigungssensordaten stets auf die Eichungstemperatur zurück korrigiert werden. Die einzelnen Faktoren der Korrekturgleichung (vgl. Gleichung 4.7) lassen sich dabei relativ einfach bestimmen:

- Der Korrekturfaktor a_T wird einmalig festgelegt und bleibt konstant
- Die aktuelle Temperatur T_A liefert der Temperatursensor direkt als Messwert
- Die Eichungstemperatur $T_{Eichung}$ wird bei der Eichung zusammen mit dem Korrekturfaktor festgelegt und bleibt anschliessend konstant
- Die Beschleunigungssensorwerte a werden direkt von den Beschleunigungssensoren geliefert

Für die Berechnung wird nur die Temperaturdifferenz zwischen Eichungstemperatur und aktueller Temperatur benötigt. Es kann daher darauf verzichtet werden, den Sensoroutput des Temperatursensors gemäss Gleichung 4.3 in °C umzurechnen.
Zur Bestimmung des Korrekturfaktors a_T kann die Erdbeschleunigung zur Hilfe genommen werden, welche konstant ist, solange das Sensor-Board keinen relativen Bewegungen zur Erdoberfläche ausgesetzt ist.
Zur Bestimmung des Korrekturfaktors wurden zwei Messungen, bei unterschiedlicher Umgebungstemperatur, aufgenommen. Die Temperatur bei der ersten Messung wurde anschliessend als Eichungstemperatur $T_{Eichung}$ angenommen. Mit Hilfe der zweiten Messung konnte Δa und ΔT bestimmt werden und somit den Korrekturfaktor a_T, da die effektive Beschleunigung in beiden Messungen die selbe ist.
Folgende Gleichung verdeutlicht das angewandte Verfahren:

$$a_T \cdot (T_1 - T_{Eichung}) \cdot a_1 + a_1 = a_T \cdot (T_2 - T_{Eichung}) \cdot a_2 + a_2 \tag{4.8}$$

Dabei gilt gemäss vorheriger Erläuterung $T_{i=1} = T_{Eichung}$ wodurch sich die Gleichung

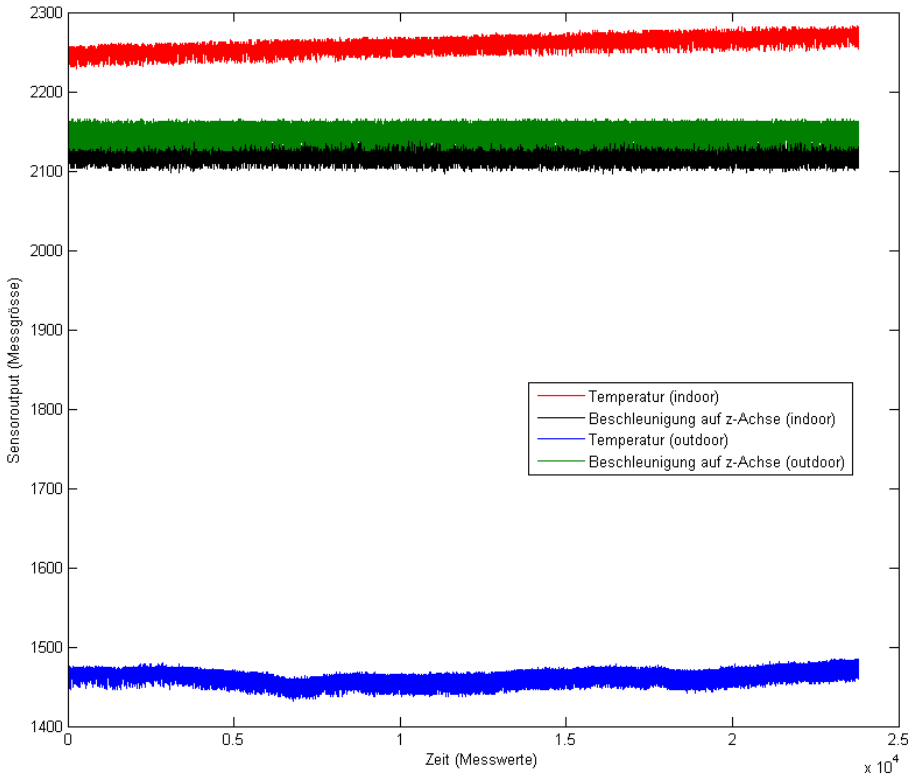

Abbildung 4.4.: Messungen Indoor und Outdoor zur Bestimmung des Temperaturkorrekturfaktors für die Beschleunigungssensoren

vereinfachen lässt.

$$a_1 = a_T \cdot (T_2 - T_{Eichung}) \cdot a_2 + a_2 \tag{4.9}$$

Für die Eichung wurden zwei gleichlange Messungen vorgenommen:

- **indoor**
  Das Sensor-Board wurde ohne Bewegungseinflüsse in einem Raum auf einen Tisch bei rund 25.8°C gelegt

- **outdoor**
  Das Sensor-Board wurde ebenfalls ohne Bewegungseinflüsse auf eine glatte Oberfläche draussen bei rund 3.8°C gelegt

In beiden Fällen wirkt wie bereits erwähnt die Erdbeschleunigung mit rund $9.81 \frac{m}{s^2}$ auf das Sensor-Board. In Abbildung 4.4 sind die Messresultate visualisiert, wobei der Einfluss der

Temperatur bei diesen Messungen noch deutlicher zu Vorschein kommt als in Abbildung 4.1.
In Tabelle 4.2 sind zudem die für die Berechnung benötigten Messgrössen aufgeführt.

	Indoor	**Outdoor**
Temperatur (Durchschnitt)	2261.0136	1463.0238
Beschleunigung (z-Achse, Durchschnitt)	2120.0754	2147.5679

Tabelle 4.2.: Benötigte Messgrössen zur Bestimmung des Temperaturkorrekturfaktors für die Beschleunigungssensoren

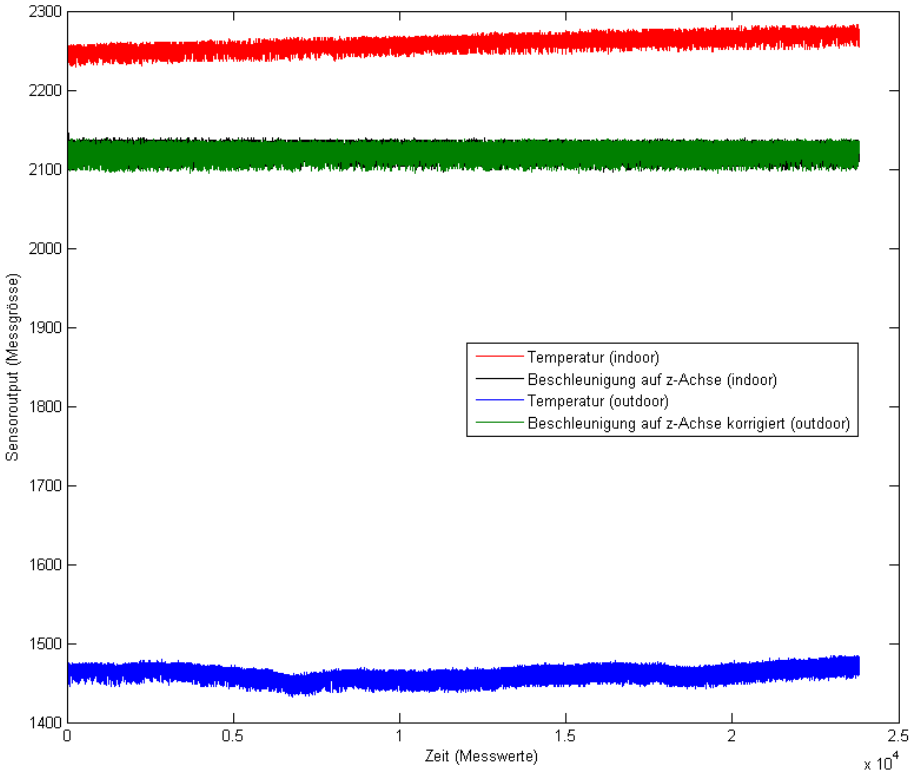

Abbildung 4.5.: Temperaturkorrigierte Messwerte der Beschleunigungssensoren

Basierend auf Gleichung 4.9 kann nun der benötigte Korrekturfaktor berechnet werden. Dabei wird, basierend auf den Messwerten aus Tabelle 4.2, die Eichungstemperatur wie folgt festgelegt: $T_{Eichung} = 2261.0136$. Umgerechnet mit Gleichung 4.3 entspricht dies einer Temperatur von 25.88°C.

$$2120.0754 = a_T \cdot (1463.0238 - 2261.0136) \cdot 2147.5679 + 2147.5679 \tag{4.10}$$

Aufgelöst nach dem Korrekturfaktor a_T kann die benötigte Grösse berechnet werden.

$$a_T = 0.000016042417, \text{ bei } T_{Eichung} = 2261.0136 \tag{4.11}$$

Zur Korrektur der Beschleunigungsdaten kann, basierend auf Gleichung 4.7, künftig folgende Gleichung verwendet werden:

$$a_{corr} = 0.000016042417 \cdot (T_A - 2261.0136) \cdot a + a \tag{4.12}$$

Zur Illustration des Verfahrens wird wurde die Korrekturgleichung auf die Messwerte aus Abbildung 4.4 angewandt. Die Korrektur ist dabei sehr deutlich in Abbildung 4.5 zu erkennen.
Abschliessend muss noch darauf hingewiesen werden, dass diese Temperaturkorrektur, insbesondere der Korrekturfaktor, Sensor-Board spezifisch sein könnte. Beim Einsatz mit anderen Sensor-Boards ist daher zu prüfen, ob die Kalibrierung bzw. Bestimmung des Korrekturfaktors für jedes Sensor-Board neu vorgenommen werden muss.

4.4. Magnetsensor

Der Magnetsensor ist gleich mehrfach von Interesse. Einerseits kann er eingesetzt werden, um Richtungsangaben (Winkel) zu berechnen, welche für die Methode der einfachen Integration nötig sind (vgl. hierzu Kapitel 2), und andererseits werden die Drehwinkel für die Berechnung der jeweiligen Drehmatrizen benötigt, wie sie in Kapitel 2.2.2 eingeführt wurden.
Trotz anschliessender Kalibrierung muss jedoch darauf hingewiesen werden, dass der verwendete Magnetsensor nicht nur das Erdmagnetfeld misst, sondern auch alle anderen Magnetfelder. Es muss daher immer auch mit Störmagnetfeldern gerechnet werden, welche entsprechend berücksichtigt werden müssen. Besonders die Winkelbeschleunigungssensoren könnten dabei von grossem Nutzen sein.

4.4.1. Kalibrierung

Damit sich die entsprechenden Winkel aus den Magnetsensordaten berechnen lassen, ist jedoch eine einmalige Kalibrierung erforderlich. Im Gegensatz zu [Tschanz 2005] werden die Sensordaten jedoch nicht normalisiert, sondern nur um einen festgelegten (kalibrierten) Offset korrigiert.

Der Verzicht auf die Normalisierung, wie sie von [Tschanz 2005] vorgenommen wurde, ermöglicht eine einmalige Kalibrierung, welche zudem weltweit seine Gültigkeit hat. Durch die unterschiedliche Stärke des Erdmagnetfeldes je nach Höhe über Meer sowie Bereitengrad[7], müsste bei einer Normalisierung der Daten nämlich, je nach Standortveränderung, eine Neukalibrierung vorgenommen werden.
Der Verzicht auf eine Normalisierung kann auch damit begründet werden, dass die Magnetsensordaten von einem einzigen Magnetsensor stammen, welcher bereits unter den Achsen normalisiert ist.
Zur Bestimmung des Offsets je Achse wurden mit dem Sensor-Board mehrere Aufnahmen, in einem magnetisch ungestörten Raum, aufgenommen. Anschliessend wurden die Offsets nach folgendem Verfahren berechnet:

$$\text{Offset} = \frac{\text{Sensorhöchswert} - \text{Sensortiefstwert}}{2} + \text{Sensortiefstwert} \tag{4.13}$$

Die resultierten Offsetwerte sind in Tabelle 4.3 ersichtlich.
In Abbildung 4.6 ist der Sensoroutput (offsetkorrigiert) bei einer 360-Grad Drehung

	Offset
Sensor 1 (x)	1530.25
Sensor 2 (z)	1424
Sensor 3 (y)	1493.125

Tabelle 4.3.: Offsets des Magnetsensors

ersichtlich. Nach der Kalibrierung der Sensoren mittels Offset, besteht die Möglichkeit, anhand des Sensoroutputs die entsprechenden Drehwinkel zu berechnen. Im anschliessenden Unterkapitel wird das entsprechende Vorgehen noch genauer erläutert.
Abbildung 4.7 zeigt zudem für jeden Winkel den entsprechenden zu erwartenden Sensoroutput.

[7] In Zürich ist das Magnetfeld beispielsweise rund 0.47-0.48 Gauss stark. Am Magnetpol herrscht jedoch eine Stärke von rund 0.6 Gauss und am Äquator von rund 0.3 Gauss.

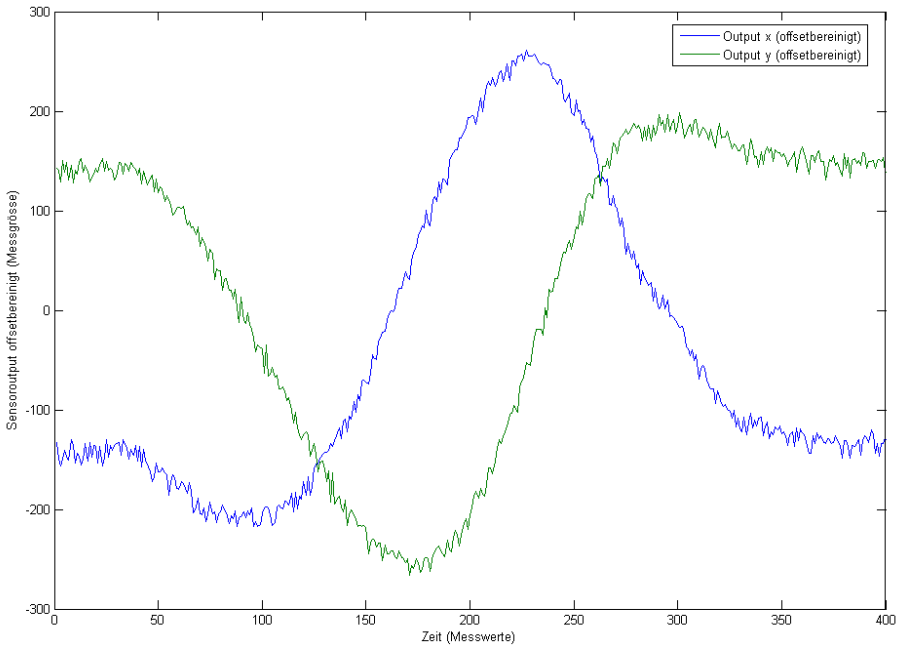

Abbildung 4.6.: Offsetkorrigierter Sensoroutput bei einer 360-Grad Drehung

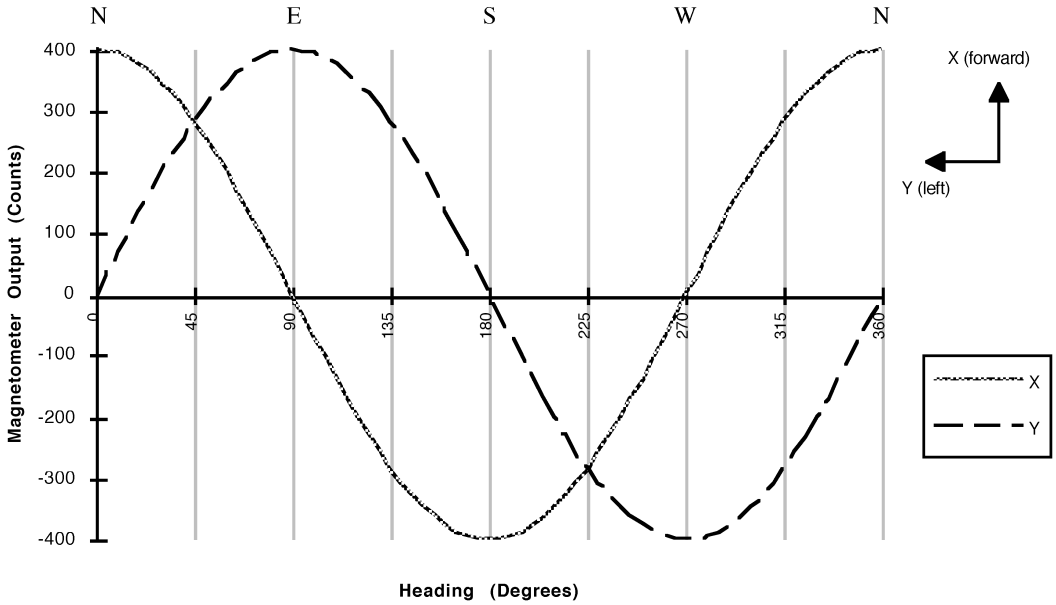

Abbildung 4.7.: Sensoroutput für x und y für verschiedene Kompassrichtungen (Winkel) [Honeywell 1995]

4.4.2. Winkelberechnung

Nach der Bereinigung der Daten durch Abzug des entsprechenden Offsets, lässt sich der Winkel zu jedem Zeitpunkt berechnen (vgl. hierzu auch Gleichung 4.14 bzw. [Honeywell 1995]).

$$\begin{aligned} Winkel\ (y > 0) &= 90 - \arctan\left(\frac{x}{y}\right) \cdot \frac{180}{\pi} \\ Winkel\ (y < 0) &= 270 - \arctan\left(\frac{x}{y}\right) \cdot \frac{180}{\pi} \\ Winkel\ (y = 0, x < 0) &= 180.0 \\ Winkel\ (y = 0, x > 0) &= 0.0 \end{aligned} \tag{4.14}$$

x und y bezeichnen dabei den korrigierten Sensoroutput der entsprechenden Achse.
Bei einer Berechnung der Winkel, für die in Abbildung 4.6 ersichtlichen korrigierten Sensordaten, ist die vorgenommene 360-Grad Drehung sehr schön zu erkennen (vgl. Abbildung 4.8).
Für den vorliegenden Fall ist jedoch die effektive Gradausrichtung (nach Norden) des

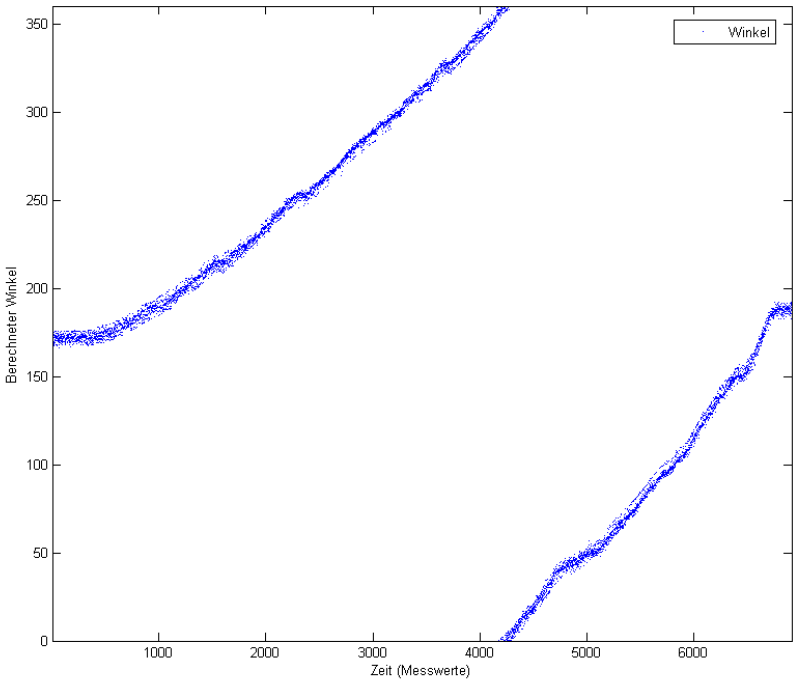

Abbildung 4.8.: Berechnete Winkel bei einer 360-Grad Drehung

Sensor-Boards nicht von Interesse. Viel mehr sind die relativen Lageveränderungen gegenüber der Anfangsposition von Interesse. Daher kann auf die Berücksichtigung des ortsabhängigen Deklinationswinkel verzichtet werden.

4.4.3. Berechnung der Winkel für die Drehmatrizen

Zur Bestimmung der in Kapitel 2.2.2 eingeführten Drehmatrizen sind die entsprechenden Drehwinkel α, β und γ erforderlich. An dieser Stelle sei nochmals auf die Bedeutung der einzelnen Winkel hingewiesen:

1. α bezeichnet den Winkel einer Drehung um die x-Achse
2. β bezeichnet den Winkel einer Drehung um die y-Achse
3. γ bezeichnet den Winkel einer Drehung um die z-Achse

Ausgehend von der Berechnungsmethode 4.14 lassen sich durch die Magnetsensordaten der x, y und z-Achse alle drei Winkel wie folgt berechnen:

- **Winkel** α

$$\begin{aligned} Winkel\ \alpha\ (z > 0) &= 90 - \arctan\left(\frac{x}{z}\right) \cdot \frac{180}{\pi} \\ Winkel\ \alpha\ (z < 0) &= 270 - \arctan\left(\frac{x}{z}\right) \cdot \frac{180}{\pi} \\ Winkel\ \alpha\ (z = 0, x < 0) &= 180.0 \\ Winkel\ \alpha\ (z = 0, x > 0) &= 0.0 \end{aligned} \tag{4.15}$$

- **Winkel** β

$$\begin{aligned} Winkel\ \beta\ (z > 0) &= 90 - \arctan\left(\frac{y}{z}\right) \cdot \frac{180}{\pi} \\ Winkel\ \beta\ (z < 0) &= 270 - \arctan\left(\frac{y}{z}\right) \cdot \frac{180}{\pi} \\ Winkel\ \beta\ (z = 0, y < 0) &= 180.0 \\ Winkel\ \beta\ (z = 0, y > 0) &= 0.0 \end{aligned} \tag{4.16}$$

- **Winkel γ**

$$\begin{aligned} Winkel\ \gamma\ (y > 0) &= 90 - \arctan\left(\frac{x}{y}\right) \cdot \frac{180}{\pi} \\ Winkel\ \gamma\ (y < 0) &= 270 - \arctan\left(\frac{x}{y}\right) \cdot \frac{180}{\pi} \\ Winkel\ \gamma\ (y = 0, x < 0) &= 180.0 \\ Winkel\ \gamma\ (y = 0, x > 0) &= 0.0 \end{aligned} \tag{4.17}$$

4.5. Winkelbeschleunigungssensoren

Die Winkelbeschleunigungssensoren, auch bekannt unter dem Namen Gyroskop bzw. Gyro, sind vor allem in Bezug auf Winkelveränderungen von Interesse. Die Winkelbeschleunigungen lassen sich in Winkelveränderungen umrechnen, was redundante Daten zu den Magnetsensoren liefern könnte. Dadurch wäre es beispielsweise möglich, die Winkelinformationen fehlertolerant zu berechnen, da die Magnetsensoren leider diversen Störfeldern, welche das Messresultat beeinflussen, unterliegen.
Die Kalibrierung der Winkelbeschleunigungssensoren gestaltet sich dabei als äusserst umständlich wobei Probleme mit der Ungenauigkeit bei der Kalibrierung schliesslich die Verwendung dieser Sensoren mit dem vorhandenen Kalibrierungsequipment verunmöglichte. Trotzdem wird in diesem Unterkapitel vorgestellt, wie die Winkelbeschleunigungssensoren kalibriert wurden und wo die Schwierigkeit für dessen Verwendung liegt.
Die Kalibrierung wird in mehreren Schritten vorgenommen. Als Erstes müssen die Sensorwerte der Winkelbeschleunigungssensoren ebenso wie die Beschleunigungssensoren in Bezug auf Temperatureinflüsse korrigiert werden. Dabei können die Erkenntnissen aus Kapitel 4.3.1 verwendet werden. Anschliessend muss der Offset je Sensor berechnet bzw. kalibriert werden und schliesslich müssen die Winkelbeschleunigungsdaten in Winkel umgerechnet (integriert) werden.

4.5.1. Korrektur des Temperatureinflusses

Wie bereits bei den Beschleunigungssensoren muss auch bei den Winkelbeschleunigungssensoren mit Ungenauigkeiten infolge Temperaturschwankungen gerechnet werden. Der Hersteller gibt diese Ungenauigkeiten mit $\pm 15\%$ an (vgl. hierzu Tabelle 3.4 bzw. [NEC/TOKIN 2001] und [NEC/TOKIN 2005a]). Leider sind keine weiteren Informationen über diese Ungenauigkeiten bzw. dessen Bezugsgrösse bekannt.
Die temperaturbedingten Abweichungen sind jedoch in Abbildung 4.9 sehr gut erkenn-

bar. Der Einfluss ist dabei sogar noch einiges schwerwiegender als bei den Beschleunigungssensoren (siehe Abbildung 4.4).
Analog zu Kapitel 4.3.1 kann jedoch relativ einfach ein Temperaturkorrekturfaktor be-

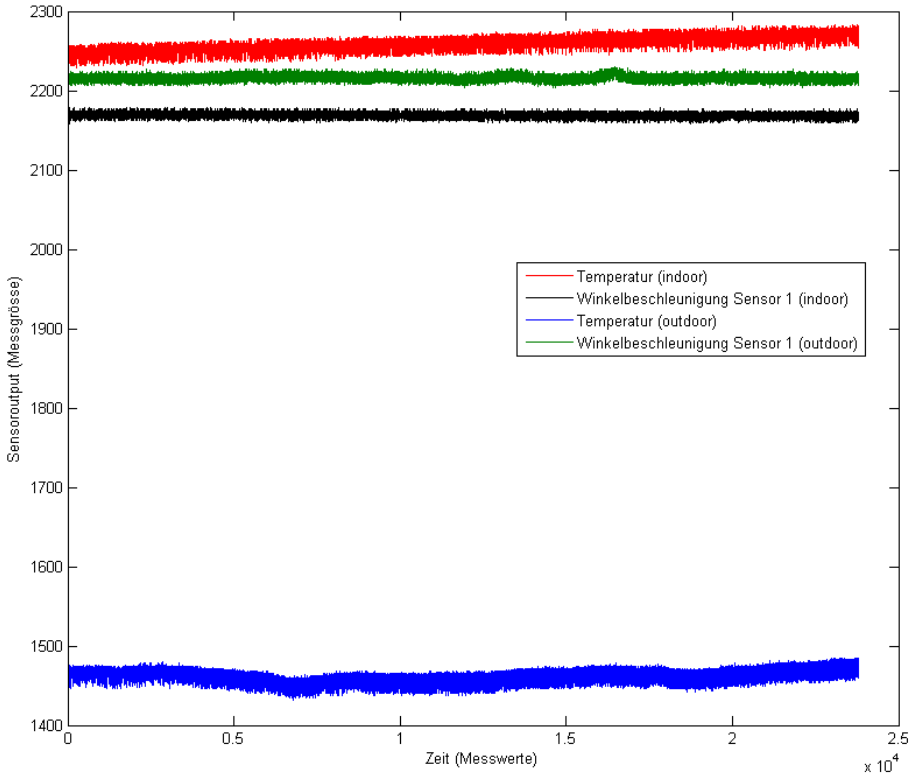

Abbildung 4.9.: Messungen Indoor und Outdoor zur Bestimmung des Temperaturkorrekturfaktors für die Winkelbeschleunigungssensoren

stimmt werden. Auch in diesem Fall kann wiederum von einer linearen Abhängigkeit sowie unverzögerter Reaktion ausgegangen werden. Die Korrekturgleichung entspricht dabei der bekannten Gleichung 4.7 aus Kapitel 4.3.1.

$$g_{corr} = g_T \cdot (T_A - T_{Eichung}) \cdot g + g \tag{4.18}$$

Für die Berechnung des Korrekturfaktors g_T muss wiederum die Eichungstemperatur $T_{Eichung}$ festgelegt werden, welche vom Temperatursensor bei der Eichung gemessen wird. Die einzelnen Faktoren der Korrekturgleichung lassen sich dabei relativ einfach bestimmen:

- Der Korrekturfaktor g_T wird einmalig festgelegt und bleibt konstant

- Die aktuelle Temperatur T_A liefert der Temperatursensor direkt als Messwert
- Die Eichungstemperatur $T_{Eichung}$ wird bei der Eichung zusammen mit dem Korrekturfaktor festgelegt und bleibt anschliessend konstant
- Die Winkelbeschleunigungssensorwerte g werden direkt von den Winkelbeschleunigungssensoren geliefert

Zur Bestimmung des Korrekturfaktors g_T wurden wiederum zwei Messungen vorgenommen. In diesem Fall handelt es sich dabei um die gleiche Messung wie in Kapitel 4.3.1 wobei die Sensorwerte in Abbildung 4.9 ersichtlich sind. In Tabelle 4.4 sind zudem die für die Berechnung benötigten Messgrössen aufgeführt.

	Indoor	**Outdoor**
Temperatur (Durchschnitt)	2261.0136	1463.0238
Winkelbeschleunigung (Sensor 1, Durchschnitt)	2169.1776	2216.3144

Tabelle 4.4.: Benötigte Messgrössen zur Bestimmung des Temperaturkorrekturfaktors für die Winkelbeschleunigungssensoren

Basierend auf Gleichung 4.9 kann nun der benötigten Korrekturfaktor berechnet werden. Dabei wird, basierend auf den Messwerten aus Tabelle 4.4, die Eichungstemperatur als $T_{Eichung} = 2261.0136$ festgelegt, wobei dies umgerechnet mit Gleichung 4.3 wiederum einer Temperatur von rund 25.88°C entspricht.

$$2169.1776 = g_T \cdot (1463.0238 - 2261.0136) \cdot 2216.3144 + 2216.3144 \tag{4.19}$$

Aufgelöst nach dem Korrekturfaktor a_T kann die benötigte Grösse berechnet werden:

$$g_T = 0.000026652104, \text{ bei } T_{Eichung} = 2261.0136 \tag{4.20}$$

Zur Korrektur der Winkelbeschleunigungsdaten kann, basierend auf Gleichung 4.18, künftig folgende Gleichung verwendet werden:

$$g_{corr} = 0.000026052104 \cdot (T_A - 2261.0136) \cdot g + g \tag{4.21}$$

In Abbildung 4.10 sind die korrigierten Messwerte aus Abbildung 4.9 dargestellt, wobei nun der Sensoroutput des Winkelbeschleunigungssensors bei beiden Messungen übereinander liegt.

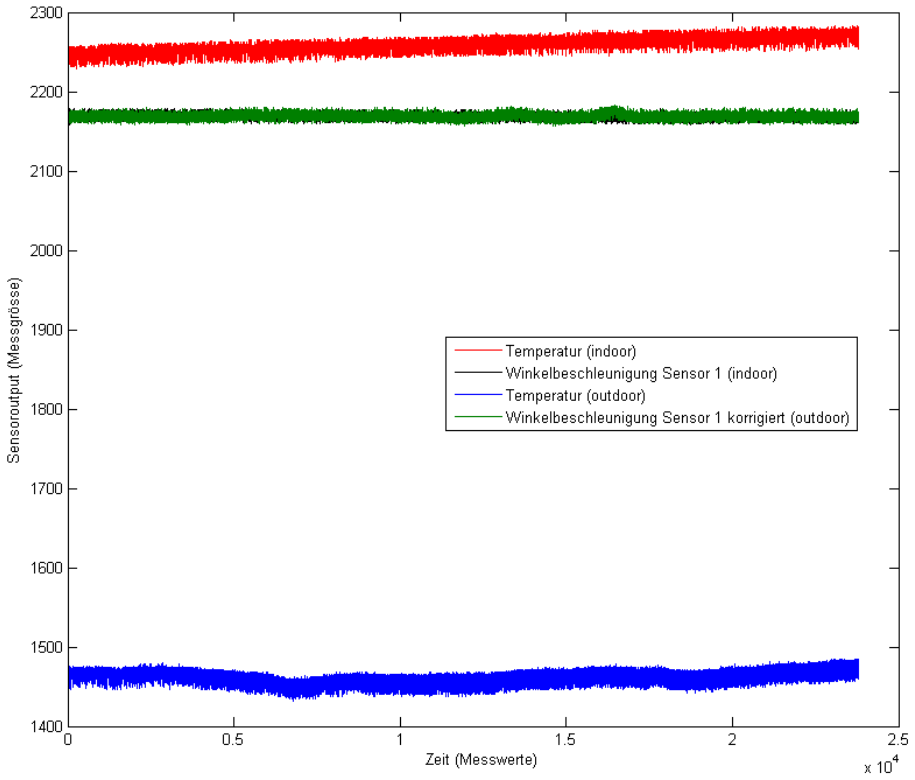

Abbildung 4.10.: Temperaturkorrigierte Messwerte der Winkelbeschleunigungssensoren

4.5.2. Kalibrierung: Offsetberechnung

Nach der Bereinigung der Daten von Temperatureinflüssen, kann die weitere Kalibrierung vorgenommen werden. Als Erstes interessiert dabei sicherlich der Offset pro Sensor, welcher durch die Umwandlung in digitale Messdaten durch den ADC entstanden ist. Zudem muss allenfalls ein Streckfaktor berechnet werden, welcher die Messdaten in physikalische Grössen überführt.
Zur Berechnung der Offsets wurden drei Messungen vorgenommen. Die Messungen wurden bei unterschiedlicher Orientierung des Sensor-Boards zur Erdoberfläche, sowie bei wechselnder Temperatur aufgenommen. Die unterschiedlichen Lagen bei den Aufnahmen sind in Abbildung 4.11 ersichtlich.

Abbildung 4.11.: Unterschiedliche Lagen des Sensor-Board zur Kalibrierung der Winkelbeschleunigungssensoren

1. Das Sensor-Board ist parallel zur Erdoberfläche orientiert (x-y-Ebene parallel zur Erdoberfläche) (vgl. (1) in Abbildung 4.11)

2. Das Sensor-Board steht senkrecht zur Erdoberfläche (x-z-Ebene parallel zur Erdoberfläche) (vgl. (2) in Abbildung 4.11)

3. Das Sensor-Board steht senkrecht zur Erdoberfläche (y-z-Ebene parallel zur Erdoberfläche) (vgl. (3) in Abbildung 4.11)

Für jede Lage wurde eine Kalibrierungs-Aufnahme vorgenommen, wobei sich das Sensor-Board bewegungsfrei auf einem Tisch befand. Die resultierten temperaturbereinigten Daten sind in Abbildung 4.12 ersichtlich. Anhand des durchschnittlichen, sowie temperaturbereinigten, Sensoroutputs (siehe Tabelle 4.5) konnten anschliessend die Offsets berechnet werden, welche in Tabelle 4.6 ersichtlich sind.
In Abbildung 4.13 ist zudem der korrigierte Sensoroutput (Temperatureinfluss sowie

	Messung 1	**Messung 2**	**Messung 3**
Gyro1 (y-Achse, Durchschnitt)	2177.8654	2179.1757	2178.3225
Gyro2 (x-Achse, Durchschnitt)	2039.0647	2039.2433	2038.8643
Gyro3 (z-Achse, Durchschnitt)	2047.8297	2048.0541	2048.9727

Tabelle 4.5.: Durchschnittlicher Sensoroutput zur Berechnung der Offsets (Winkelbeschleunigungssensoren)

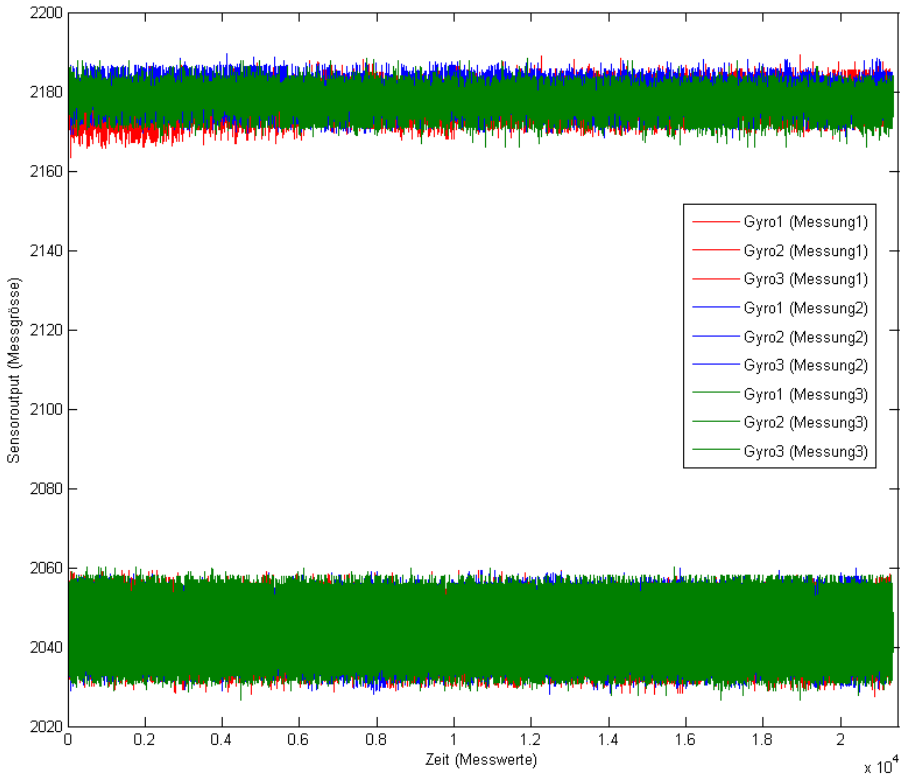

Abbildung 4.12.: Offset der Winkelbeschleunigungssensoren bei drei verschiedenen Messungen

Offsets) der drei Winkelbeschleunigungssensoren für alle drei Aufnahmen ersichtlich. Dabei ist gut zu erkennen, wie der Sensoroutput aller drei Sensoren für alle drei Aufnahmen sich im Bereich von ±10 überlappt.
Ein allfälliger Streckfaktor kann zu diesem Zeitpunkt leider noch nicht berechnet werden

Sensor	**Offset**
Gyro1 (y-Achse)	2178.4546
Gyro2 (x-Achse)	2039.0574
Gyro3 (z-Achse)	2048.2855

Tabelle 4.6.: Offset der Winkelbeschleunigungssensoren

sondern muss anhand von Referenzdaten bestimmt werden, welche beispielsweise vom Magnetsensor geliefert werden. Zum jetzigen Zeitpunkt wird von einem Streckfaktor = 1 ausgegangen.

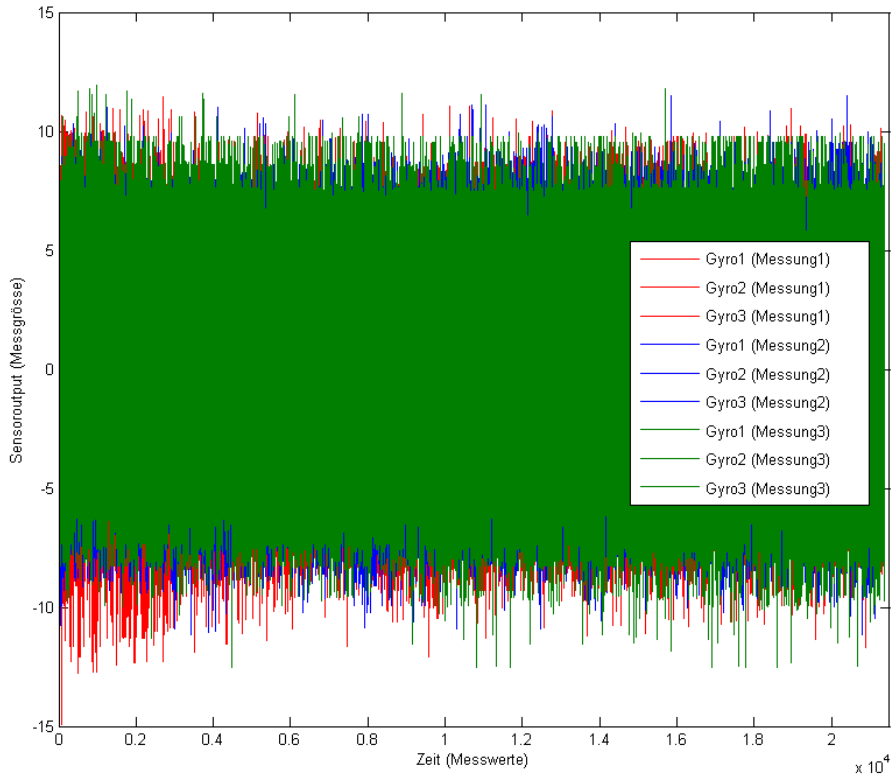

Abbildung 4.13.: Sensoroutput der Winkelbeschleunigungssensoren nach der Korrektur von temperaturbedingten Abweichungen sowie Abzug der Offsets

4.5.3. Winkelberechnung

Die Winkelbeschleunigungssensoren eignen sich insofern optimal zur Drehwinkelberechnung, da die Winkelbeschleunigungssensoren die Winkelbeschleunigung um eine Achse messen. Insofern liefern die Winkelbeschleunigungssensoren für jeden Drehwinkel α, β und γ eine Beschleunigung in $\frac{RAD}{s^2}$, welche nur noch in einen Winkel umgerechnet werden muss.
Die Berechnung des Winkels aus einer Beschleunigung ist zu vergleichen mit der Berechnung des Ortes aus den Beschleunigungssensoren (vgl. Kapitel 2), denn auch bei den Winkelbeschleunigungssensoren muss eine doppelte Integration vorgenommen werden, um die gewünschte Information (der entsprechende Drehwinkel) zu erhalten.

Der Drehwinkel lässt sich gemäss [DMK/DPK 1995] und [Tschanz 2005] wie folgt berechnen:

$$\begin{aligned} \alpha &= \frac{\Delta\omega}{\Delta t}, \text{ wobei } \alpha = \text{Winkelbeschleunigung} \\ \omega &= \frac{\Delta\varphi}{\Delta t}, \text{ wobei } \omega = \text{Winkelgeschwindigkeit} \\ \varphi &= \Delta t^2 \cdot \alpha, \text{ wobei } \varphi = \Delta\text{Drehwinkel in RAD} \end{aligned} \tag{4.22}$$

Angewendet auf die Sensordaten errechnet sich der Drehwinkel φ wie folgt:

$$\varphi = \Delta t^2 \cdot \sum_{i=1}^{k} [(k - i + 1) \cdot g_i] - \Delta t^2 \cdot \frac{(k+1) \cdot k}{2} \cdot g_{offset} \tag{4.23}$$

Da im vorliegenden Fall jedoch der Offset direkt von den Sensordaten abgezogen werden kann, lässt sich die Gleichung noch etwas vereinfachen:

$$\varphi = \Delta t^2 \cdot \sum_{i=1}^{k} [(k - i + 1) \cdot g_i] \tag{4.24}$$

Bei φ handelt es sich jedoch um einen Winkel im Bogenmass. Für den vorliegenden Fall werden die Winkel jedoch in Grad benötigt, weshalb eine entsprechende Umrechnung vorgenommen werden muss. Die entsprechende Umrechnung lässt sich dabei folgendermassen vornehmen:

$$\varphi_{DEG} = \varphi_{RAD} \cdot \frac{180}{\pi} \tag{4.25}$$

Ausgehend von den Sensor-Rohdaten, ergibt sich folgendes Schema zur Berechnung der Drehwinkel:

1. Bereinigung der Daten von Temperatureinflüssen (vgl. Gleichung 4.21)
2. Abzug der Offsets von den Daten (vgl. Tabelle 4.6)
3. Zweifache Integration (vgl. Gleichung 4.24)
4. Umrechnung der Winkel von RAD in DEG (vgl. Gleichung 4.25)
5. Multiplikation der Winkel mit dem Streckfaktor

Der Streckfaktor müsse, wie bereits angesprochen, über Referenzdaten berechnet werden. Als Referenzdaten können beispielsweise die Daten des Magnetsensors verwendet werden. In diesem Fall war jedoch die Berechnung des Streckfaktors nicht möglich, aufgrund von Kalibrierungsproblemen bzw. des Integrationsdrift. Auf die genauen Ursachen der Probleme bei der Kalibrierung wird im nächsten Unterkapitel noch detailliert eingegangen.

4.5.4. Probleme mit der Kalibrierung

Im Gegensatz zu den Beschleunigungssensoren ist bei den Winkelbeschleunigungssensoren eine vorgängige Kalibrierung, sprich Berechnung des Offset, nötig. Eine Online-Kalibrierung zur Vermeidung des Integrationsdrift wie sie bei den Beschleunigungssensoren vorgenommen wird, ist in diesem Fall leider nur sehr schwer möglich.
Der Offset für jede Achse muss somit, wie in den vorherigen Kapiteln beschrieben, vorab berechnet werden. Leider hat sich bei den Berechnungen gezeigt, dass sich der Integrationsdrift aufgrund der zu ungenauen Offsetberechnungen dermassen gravierend auswirkt, dass die berechneten Winkel von den Gyrodaten einer viel zu grossen Ungenauigkeit unterliegen, damit sie für die Navigation verwendet werden könnten.
Die Zahlenwerte aus Tabelle 4.7 verdeutlichen die grosse Ungenauigkeit bei der Offsetbestimmung. Der Offset verändert sich unglücklicherweise von Aufnahme zu Aufnahme immer um einige Messpunkte. Die erste Spalte beinhaltet nochmals den kalibrierten Offset (siehe Kapitel 4.5.2) und die zweite Spalte einen erneut berechneten Offset bei einer Aufnahme über eine Stunde.
Die grossen Abweichungen beim Offset lassen die Ungenauigkeit bei der Winkelberech-

Sensor	**Offset kalibriert (vgl. Tabelle 4.6)**	**Offset berechnet (Aufnahme von 1h)**	**Abweichung (in Prozent)**
Gyro1 (y-Achse)	2178.4546	2187.9500	+0.44%
Gyro2 (x-Achse)	2039.0574	2035.3417	−0.18%
Gyro3 (z-Achse)	2048.2855	2046.5979	−0.09%

Tabelle 4.7.: Integrationsdrift bei den Winkelbeschleunigungssensoren aufgrund der ungenauen Offsetbestimmung

nung mittels einer doppelten Integration bereits erahnen.
Ausgehend von der zweiten (potentiell genaueren) Offsetbestimmung wurden erneut Messungen durchgeführt. Das Sensor-Board wurde dabei jeweils 90-Grad um die z-Achse gedreht und der zurückgelegte Winkel berechnet. Die Dauer der Aufnahme wurde bewusst variabel gewählt, dass der Integrationsdrift besser ersichtlich wird. In Tabelle 4.8 sind die resultierten Werte aus den Aufnahmen ersichtlich, wobei der Bedarf einer extrem exakten Offsetbestimmung verdeutlicht wird.
Dabei wird auch deutlich, dass eine extrem genaue Kalibrierung notwendig wäre. Die

	Messung 1	**Messung 2**
Dauer der Aufnahme	20s	40s
Berechneter Winkel in Grad (Offset = 2046.5979)	413.09	600.48
Benötigter Offset, damit Winkel = 90 Grad	2044.6751	2045.5673
Differenz zwischen benötigtem und berechnetem Offset	1.9228	1.0342

Tabelle 4.8.: Integrationsdrift der Winkelbeschleunigungsdaten bei einer 90-Grad Drehung

Messgenauigkeit der Winkelbeschleunigungssensoren ist jedoch nur gerade auf einen Messpunkt genau. Die benötigten Kalibrierungsgrössen erfordern jedoch eine viel grössere Genauigkeit zur zuverlässigen sowie genauen Winkelbestimmung.
Aufgrund der grossen Ungenauigkeiten ist die Berechnung der Drehwinkel nach einer offline-Kalibrierung, wie sie in diesem Kapitel vorgenommen wurde, für Navigationszwecke leider nicht geeignet. Es muss daher eine andere Lösung für die Berechnung der Drehwinkel bzw. der Offsets gefunden werden. Leider ist eine Alternative nicht einfach zu finden. Als erste Möglichkeit ist eine Online-Kalibrierung denkbar, welche jedoch nur unter bestimmten Voraussetzungen funktionieren könnte. Eine weitere Möglichkeit ist auf die Winkelinformationen zu verzichten und die Winkelbeschleunigungssensoren nur als Fehlererkennungsinstrument für die Magnetsensoren zu verwenden.
Beide Ansätze werden deshalb in den folgenden Unterkapiteln kurz erläutert.

4.5.5. Online-Kalibrierung

Im vorherigen Unterkapitel wurde deutlich, dass die berechneten Drehwinkel bei einer offline-Kalibrierung leider nur ungenügend genau sind. Gleichzeitig wurde aber auch auf einen alternativen Ansatz hingewiesen: In Tabelle 4.8 sind die berechneten Offsets, welche zu einem berechneten Winkel von 90-Grad führen würden, ersichtlich.
Dieses Verfahren könne selbstverständlich auch im Sinne einer Interpolation (der Offsets) angewendet werden, sofern man davon ausgehen könnte, dass Start- und Endwinkel bzw. dessen Winkelveränderung ($\Delta\alpha$, $\Delta\beta$ und $\Delta\gamma$) bekannt sind. Dies ist jedoch eine recht starke Annahme, welche nicht in jedem Fall getroffen werden kann, da unter anderem auch bekannt sein müsste, wie oft das Sensor-Board beispielsweise gedreht wurde.
In dieser Diplomarbeit sprengt die Interpolation der benötigten Offsets leider den Rahmen, da die Start- und Endwinkel in den getroffenen Annahmen nicht bekannt sind. Für eine künftige Interpolation mit bekannten Winkelveränderungen könnte man jedoch relativ einfach von Gleichung 4.23 ausgehen.

4.5.6. Fehlererkennung bei den Magnetsensordaten

Wie bereits mehrfach angesprochen, unterliegt der Magnetsensor Störfelder, welche das Messresultat verfälschen können. Eine Überprüfung der Magnetsensordaten mit den Winkelbeschleunigungsdaten wäre deshalb eine mögliche Variante, Störfelder zu erkennen und die Daten entsprechend zu filtern.
Für ein solches Verfahren müssten die Winkelbeschleunigungsdaten auch nicht doppelt integriert werden, sondern könnten als Rohdaten verwendet werden, wodurch auch kein Integrationsdrift auftreten würde. Sobald vom Winkelbeschleunigungssensor eine auffällig grosse Winkelveränderung gemessen würde, könnte eine Überprüfung der Winkelbeschleunigungsdaten Aufschluss darüber geben, ob es sich um ein Störfeld oder eine normale Winkelveränderung handelt.

5. Interpolation

5.1. Übersicht

In diesem Kapitel werden die Methoden und Verfahren aus den vorherigen Kapitel zusammengefasst und dazu verwendet, mittels Ortsinterpolation eine möglichst genaue Ortsbestimmung vornehmen zu können. Hierfür wird ein Algorithmus schrittweise entwickelt.
In einem ersten Schritt werden die Voraussetzungen aufgezeigt, welche für das Funktionieren des Ortsinterpolationsalgorithmus erfüllt sein müssen. In einem weiteren Schritt wird der der Algorithmus Schritt für Schritt aufgebaut und beschrieben. Zum Abschluss wird der Algorithmus in MATLAB abgebildet und dessen Grenzen aufgezeigt.
Im nächsten Kapitel wird dann der Ortsinterpolationsalgorithmus bei Experimenten angewandt.

5.2. Voraussetzungen

Für die in dieser Diplomarbeit angestrebten Lösung, muss davon ausgegangen werden, dass Start- sowie Endpunkt bekannt sind (Interpolation). Ebenso ist, je nach Szenario, die Kenntnis eines Referenzpunktes, aufgrund der Anzahl zu interpolierenden Unbekannten, zwischen Start und Ende notwendig. Durch den gegebenen Start- und Endpunkt, sowie gegebenenfalls einem Referenzpunkt, lassen sich die Beschleunigungssensoren sehr präzis kalibrieren, wodurch ein Integrationsdrift aufgrund ungenauer Kalibrierung weitgehend vermeiden werden kann.

5.3. Ortsinterpolationsalgorithmus

Der Ortsinterpolationsalgorithmus ist relativ einfach zu verstehen. Er lässt sich grob in fünf Schritte gliedern:

1. Import der Daten

2. Bereinigung der Daten in Bezug auf den Temperatureinfluss

3. Berechnung der Drehwinkelveränderungen gegenüber den Ausgangswinkeln

4. Berechnung der Interpolationsgrössen: Offsets sowie Streckfaktoren

5. Weginterpolation

In den folgenden Unterkapiteln werden diese fünf Schritte noch etwas genauer erläutert.

5.3.1. Import der Daten

Das Sensor-Board speichert die gemessenen Daten via RS-232 Schnittstelle und der Terminalsoftware CS Online auf einer handelsüblichen SecureDigital Speicherkarte ab (vgl. hierzu auch Kapitel 3). Die Daten werden dabei in normale Textdateien abgespeichert, welche jeweils 12 kommagetrennte Hexadezimalwerte pro Zeiteinheit beinhalten. Die einzelnen Zeiteinheiten sind durch einen Zeilenumbruch voneinander getrennt. Listing 5.1 zeigt einen Ausschnitt einer Textdatei, in welche die Hardware die entsprechenden Sensordaten abgespeichert hat.

Listing 5.1: Sensordatenbeispiel

```
933,7F5,7F9,840,886,7FD,7FB,224,269,58E,7CC,5D9
92A,7F0,7FB,856,886,7FE,7FA,22E,269,590,7D1,5D2
930,7F9,7F8,84C,884,7F3,7FE,231,26A,59A,7D8,5CD
930,7ED,7F2,848,886,7E8,7F5,22F,269,580,7D6,5D9
934,7E6,7FC,846,889,7EA,7FA,226,259,593,7D5,5D4
935,7F6,7F2,83B,885,7F1,7F7,22E,253,589,7D3,5D6
92D,7F1,7EF,847,883,7F6,800,22D,259,595,7C4,5D7
92B,7ED,7E7,852,886,7F6,7FE,224,259,59A,7DD,5D3
```

Die 12 kommagetrennten Hexadezimalwerte entsprechen dem Sensoroutput der 12 Sensoren in folgender Reihenfolge:

1. Temperatursensor

2. Beschleunigungssensor y-Achse

3. Beschleunigungssensor x-Achse

4. Beschleunigungssensor z-Achse

5. Winkelbeschleunigungssensor y (Drehachse)
6. Winkelbeschleunigungssensor x (Drehachse)
7. Winkelbeschleunigungssensor z (Drehachse)
8. Gassensor 1
9. Gassensor 2/3
10. Magnetsensor x
11. Magnetsensor z
12. Magnetsensor y

In einem ersten Schritt muss überprüft werden, ob bei der Aufnahme Zeilen mit Fehlern aufgetreten sind, welche keine vollständigen Daten enthalten. Diese müssen entsprechend aus den Daten gelöscht werden. Da dies jedoch, dank den schnellen Speicherkarten, ein äusserst seltenes Phänomen ist, wird durch den Algorithmus nur geprüft, ob es Fehler gibt. Die allfällige Korrektur der Fehler geschieht manuell durch Datenbearbeitung in einem herkömmlichen Texteditor.
Für die weitere Verarbeitung müssen die Werte anschliessend aus den aufgenommenen Textdateien in eine Matrix von Anz. Messwerte * 12 Sensoren importiert werden. Zudem muss eine Umrechnung aus dem Hexadezimalsystem in das Dezimalsystem vorgenommen werden.

5.3.2. Bereinigung der Daten in Bezug auf Temperatureinflüsse

Im nächsten Schritt müssen die Daten in Bezug auf Temperatureinflüsse bereinigt werden. Dies betrifft im vorliegenden Fall einzig die Beschleunigungssensoren. Das Verfahren zur Korrektur des Temperatureinflusses auf die Beschleunigungssensoren wurde dabei bereits eingehend in den Kapitel 4.3.1 beschrieben, weshalb an dieser Stelle darauf verzichtet wird, das Verfahren nochmals zu erläutern.

5.3.3. Berechnung der Drehwinkelveränderungen

Die Drehwinkelveränderungen sind für den vorliegenden Algorithmus je nach Szenario gleich zweifach von Interesse. Einerseits werden sie zur Richtungsbestimmung und andererseits zur Offsetberechnung eingesetzt. In beiden Fällen bedarf es der Berechnung der

Drehwinkel α, β und γ. Dabei sind jedoch nicht die absoluten Winkel, sondern vielmehr die Winkelveränderung gegenüber dem Ausgangswinkel von Interesse. Es muss daher ein Anfangswinkel berechnen werden, welcher jeweils von den anschliessend berechneten Winkeln abgezogen werden muss.
Die Drehwinkel lassen sich relativ einfach anhand der Magnetsensoren bestimmen, wobei das genaue Berechnungsverfahren bereits in Kapitel 4.4.2 ausführlich beschrieben wurde.
Auf die Berechnung der Drehwinkel aus den Winkelbeschleunigungssensoren wird aufgrund der Kalibrierungsprobleme verzichtet (vgl. hierzu Kapitel 4.5.4).

5.3.4. Berechnung der Interpolationsgrössen: Offsets sowie Streckfaktoren

Durch die Interpolation der Kalibrierungsdaten sollte eine erhebliche Steigerung bei der Genauigkeit möglich sein, da sich der Integrationsdrift, aufgrund ungenauer Kalibrierungsdaten, nicht mehr so schwerwiegend auswirken sollte und somit eine genauere Ortsbestimmung zulässt.
Für die Berechnung der benötigten Kalibrierungsgrössen müssen dabei die genauen Positionen von Start- und Endpunkt bekannt sein. Zudem ist ein Referenzpunkt irgendwo zwischen Start und Ende notwendig.
Erst durch diese drei gegebenen Ortsinformationen lassen sich die Interpolationsglei-

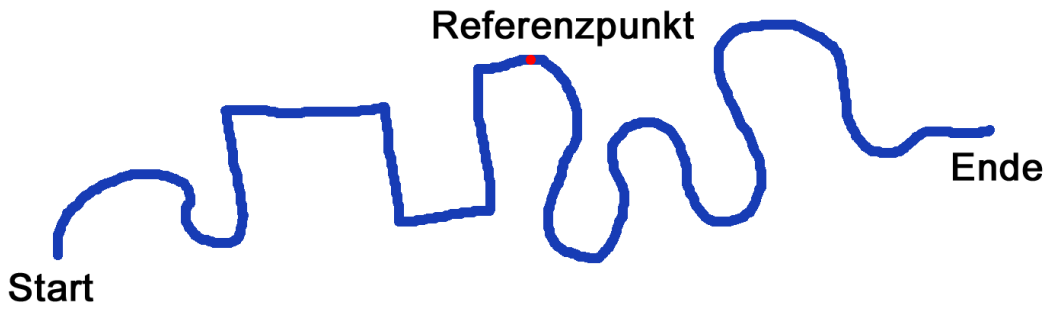

Abbildung 5.1.: Benötigte Standortinformationen zur Berechnung der Interpolationsgrössen

chungen aus Kapitel 2 lösen, da bis zu drei unbekannte Grössen ($\vec{a}_{offset}$ bzw. $\vec{a}_{offset_{ADC}}$, $\vec{\vec{\sigma}}$ und je nach Szenario $\vec{g}$) in den Berechnungen vorkommen. Die drei Strecken Start-Ende, Start-Referenzpunkt und Referenzpunkt-Ende liefern dabei drei Gleichungen für maximal drei Unbekannte, wodurch sich alle Unbekannten berechnen lassen.
Für die Interpolation muss dabei analog zum Kapitel 2 zwischen zwei Fällen unterschieden werden, da sich die Berechnung etwas unterscheidet.

Berechnung der Interpolationsgrössen: Paralleles lokales Bezugssystem

Basierend auf der Theorie aus Kapitel 2 lässt sich der zurückgelegte Weg anhand der Sensordaten berechnen. Hierfür ist jedoch die Kenntnis mehrerer Kalibrierungsgrössen notwendig, welche im Hinblick auf eine erhöhte Genauigkeit interpoliert werden sollen. Ausgehend von einem parallelen lokalen Bezugssystem (vgl. hierzu Kapitel 2.4), müssen lediglich der Streckfaktor $\vec{\sigma}$ sowie der Offset $\vec{a}_{offset}$ interpoliert werden.
Die in dieser Arbeit eingesetzten Beschleunigungssensoren messen, wie bereits mehrfach erwähnt, immer auch die Erdbeschleunigung von rund $9.81\frac{m}{s^2}$, welche auf die entsprechenden Sensoren wirkt. Damit die Erdbeschleunigung von den Messdaten abgezogen werden könnte, müsste diese immer in die gleiche Richtung relativ zu lokalen Bezugssystem wirken. Dies entspricht genau der Annahme in diesem Szenario. Es muss daher nur ein Offset berechnet werden, welcher dabei auch gleich die Erdbeschleunigung beinhaltet. Der berechnete Gesamtoffset kann anschliessend von den Daten abgezogen werden und beinhaltet auch gleich die Erdbeschleunigung. Die Interpolation der Unbekannten stellt, zumindest Theoretisch, kein grosses Problem dar, da nur sechs Unbekannte bei neun möglichen Gleichungen berechnen werden müssen.

Berechnung der Interpolationsgrössen: Frei bewegliches lokales Bezugssystem

Etwas komplizierter wird der Fall, wenn das Sensor-Board in ständiger Bewegung sein kann, was eine durchaus realistische Annahme ist (vgl. hierzu Kapitel 2.5). Damit sich der Offset auch in diesem Fall bestimmen lässt, muss er in zwei Offsets, $\vec{a}_{offset_{ADC}}$ und g, aufgeteilt werden. Der Offset $\vec{a}_{offset_{ADC}}$ beinhaltet dabei nur den ADC-Offset. Die Erdbeschleunigung wird als separater Offset $\vec{g}$ betrachtet. Um jedoch den Offset $\vec{g}$ interpolieren zu können, müsste die Erdbeschleunigung immer in die gleiche Richtung wirken, da sonst der Offset $\vec{g}$ zu jedem Zeitpunkt einen anderen Wert hätte.
Damit die Berechnung des Offset $\vec{g}$ dennoch möglich ist, werden Drehmatrizen zur Hilfe genommen, indem der Offset $\vec{g}$ jeweils mit der Drehmatrix multiplizieren wird und so die Erdbeschleunigung in das lokale Bezugssystem gedreht wird. Damit bleibt der Offset $\vec{g}$ konstant für jeden Zeitpunkt und lässt sich ebenfalls relativ einfach berechnen. Die Interpolation der Unbekannten stellt dabei wiederum, zumindest theoretisch, kein grosses Problem dar, da neun Unbekannte bei neun Gleichungen berechnet werden müssen.

5.3.5. Weginterpolation

Nachdem die benötigten Unbekannten im vorherigen Schritt interpoliert wurden, kann nun die Position zu jedem Zeitpunkt berechnet werden, was mit anderen Worten dem zurückgelegten Weg entspricht.

Je nach Berechnungsmethode muss der zurückgelegte Weg auf eine andere Art interpoliert werden.
Bei der Methode der doppelten Integration (siehe Gleichung 2.23 und 2.27 in Kapitel 2) kann mit Kenntnis von Streckfaktor $\vec{\vec{\sigma}}$, Offset $\vec{a}_{offset_{ADC}}$ und Offset $\vec{g}$ bzw. Offset $\vec{a}_{offset}$ die Position zu jedem Zeitpunkt direkt berechnet werden.
Bei der alternativen Berechnungsmethode wird darauf verzichtet, die Richtungsinformationen aus den Beschleunigungssensordaten zu berechnen (siehe Gleichung 2.25 und 2.29 in Kapitel 2). In dieser Methode werden die Richtungsangaben alternativ anhand von Magnetsensordaten berechnet.
Der einfachste Weg um die Winkelinformationen für die Richtungsangaben zu verwenden, ist dabei der Einsatz von Drehmatrizen (siehe Gleichung 2.4 in Kapitel 2). Die berechneten Drehwinkelveränderungen α, β und γ können dabei für jeden Zeitpunkt in eine Drehmatrix eingerechnet werden, wobei die resultierenden Matrizen mit den entsprechenden Längenangaben (ungerichtete Norm des Richtungsvektors) multipliziert werden.

5.4. Implementierung des Algorithmus in MATLAB

Nachdem der Interpolationsalgorithmus im vorherigen Unterkapitel logisch beschrieben wurde, muss er in MATLAB abgebildet werden. Dies geschieht mit so genannten Funktionen welche für MATLAB programmiert werden können.
In diesem Unterkapitel soll daher für jeden Algorithmusschritt das entsprechende Vorgehen in MATLAB aufgezeigt werden, ohne dabei den gesamten Sourcecode Zeile für Zeile zu erläutern.
Im Gegensatz zu den vorherigen Kapiteln, wird in diesem Unterkapitel keine Fallunterscheidung mehr zwischen den Szenarien vorgenommen. Die ersten drei Schritte des Algorithmus sind jedoch für alle Szenarien identisch und somit allgemeingültig. Die Interpolation unterscheidet sich jedoch je nach Szenario erheblich. Trotzdem wird darauf verzichtet, sämtliche Szenarien vorzustellen. Stellvertretend wird nur das Szenario "Paralleles lokales Bezugssystem - doppelte Integration" besprochen. Selbstverständlich findet sich aber der Sourcecode von allen Szenarien auf der beiliegenden CD-ROM (siehe Anhang C.

5.4.1. Parameter der Funktionen

Die Weginterpolationsalgorithmen können direkt aus MATLAB aufgerufen werden, wobei diverse Parameter mitgegeben werden müssen und der interpolierte Weg als Wegmatrix zurückgegeben wird.

Die folgende Liste erläutert die benötigten Parameter etwas genauer wobei auch der benötigte Datentyp angegeben wird:

1. Pfad und Dateiname des Textdatei mit den Sensordaten; String

2. Dauer der Gesamtaufnahme in Sekunden; Integer

3. Geschwindigkeit, welche auf das Sensor-Board für jede Achse (x, y und z) beim Startpunkt wirkt(e); 1x3-Double-Matrix

4. Startpunkt in Meter (x, y und z); 1x3-Double-Matrix

5. Endpunkt in Meter (x, y und z); 1x3-Double-Matrix

6. Referenzpunkt in Meter (x, y und z); 1x3-Double-Matrix

7. Zeitpunkt in Sekunden nach dem Start, bei welchem der Referenzpunkt erreicht wurde; Integer

8. Smoothgrösse in Datenpunkten; Integer

Ein gültiger Aufruf der Funktion ist in Listing 5.2 ersichtlich.

Listing 5.2: Beispiel eines Funktionsaufruf

```
weg = interpolation_parallel_a('c:\aufnahme-gerade', 38, [0,0,0], [0,0,0], [0,0,0],
    [10,1,1], 16, 10);
```

5.4.2. Import der Daten

Im ersten Schritt müssen die Daten, welche in Textdateien abgespeichert sind, in eine Matrix importiert werden. Dabei wird überprüft ob die Datei fehlerhafte Zeilen beinhaltet und bei Bedarf mit einer Fehlermeldung abgebrochen. Anschliessend werden die Daten aus dem Hexadezimal ins Dezimalsystem umgerechnet. Als letzter Punkt findet ein Smooth über die Daten statt. Dabei kann je nach mitgeliefertem Parameter die Datenmenge verringert und dadurch die Performance verbessert werden. Bei einem Smooth werden die Daten über die gegebene Smoothgrösse gemittelt und anschliessend verkleinert auf den entsprechenden Mittelwert.
Die folgenden Abbildungen (vgl. Abbildungen 5.2 und 5.3) zeigen eine Weginterpolation (doppelte Integration bei parallelem lokalen Bezugssystem) bei unterschiedlichen Smoothgrössen. Der Datensatz beinhaltet in diesem Fall ursprünglich 2829 Messwerte und zeigt wie von einem Startpunkt aus, eine gerade Strecke zum Referenzpunkt und wieder

zurück zum Startpunkt zurückgelegt wurde (für weitere Informationen zur Bedeutung der Aufnahme vgl. Kapitel 6). Dabei ist gut erkennbar, dass eine Smoothgrösse bis 100 Messpunkte keinen signifikanten Einfluss auf den berechneten Weg hat.

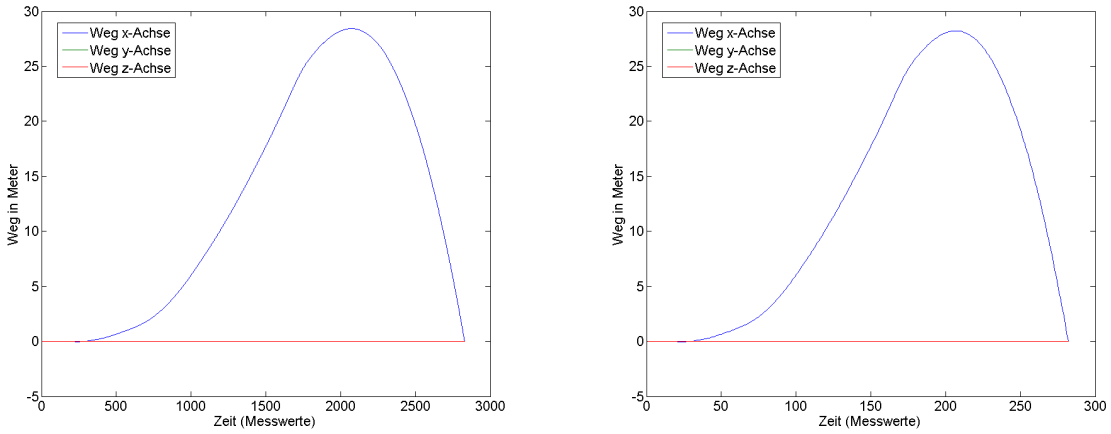

Abbildung 5.2.: Weginterpolation bei einer Smoothgrösse von 1 (links) und 10 (rechts)

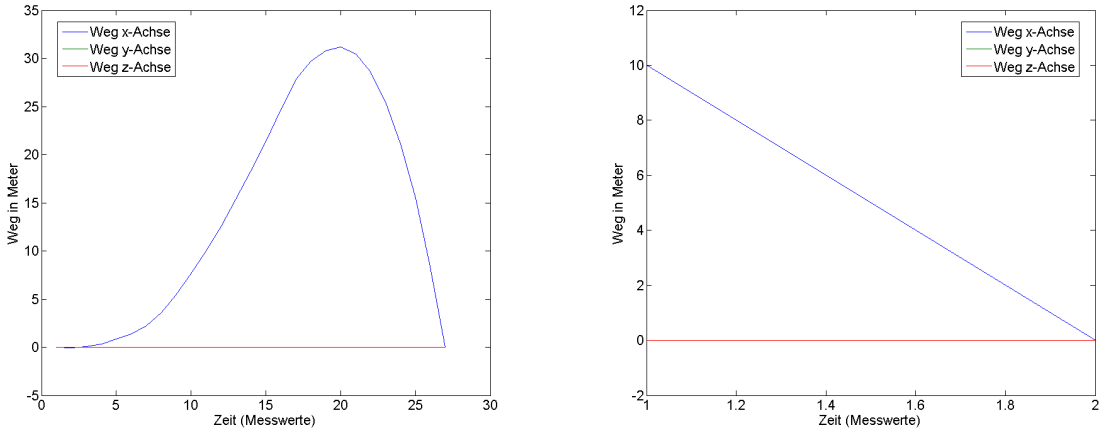

Abbildung 5.3.: Weginterpolation bei einer Smoothgrösse von 100 (links) und 1000 (rechts)

Der Sourcecode für den Datenimport ist in Listing B.1 (siehe Anhang B) ersichtlich.

5.4.3. Bereinigung der Daten in Bezug auf Temperatureinflüsse

In einem weiteren Schritt werden die Beschleunigungsdaten von allfälligen Temperatureinflüssen bereinigt. Dabei wird Gleichung 4.12 aus Kapitel 4.3.1 angewandt. Der entsprechende Sourcecode ist in Listing B.2, wiederum in Anhang B, ersichtlich.

5.4.4. Berechnung der Drehwinkelveränderungen

Die benötigten Drehwinkel α, β und γ werden im vorliegenden Fall anhand der Magnetsensordaten und den entsprechenden Berechnungsgleichungen aus Kapitel 4.4.3 hergeleitet.
Dabei gilt es auch zu beachten, dass die Drehwinkel für die weiteren Berechnungen im Bogenmass vorliegen müssen, weshalb eine entsprechende Umrechnung vorgenommen werden muss. Der entsprechende Sourcecode ist wiederum in Anhang B, Listing B.3 ersichtlich.

5.4.5. Berechnung der Interpolationsgrössen: Offsets sowie Streckfaktoren

Der wohl aufwändigste Teil des gesamten Algorithmus ist die Interpolation der Unbekannten. Dabei werden anhand der Sensordaten für jede Achse bis zu drei Gleichungen, basierend auf Kapitel 2, erstellt, wobei Start-, End- und Referenzpunkt bekannt sind. Die Interpolationsgrössen werden in den entsprechenden Gleichungen jeweils als Unbekannte aufgeführt und sollen von MATLAB mittels SOLVE[8] berechnet werden. Dabei werden die Interpolationsgrössen derart berechnet, dass der anschliessend berechnete Weg bei Start, Ende und Referenzpunkt vorbeiführen muss, wodurch eine grössere Genauigkeit der Weginterpolation erreicht werden sollte.
Besonders aufwändig ist die Erstellung der zu lösenden Gleichungen, welche je nach Anzahl Messwerten, sehr gross werden können und in MATLAB zu erheblichen Performanceproblemen führen. Auch die anschliessende Lösung des Gleichungssystems stellt für MATLAB eine grosse Hürde dar. Auf die Probleme von MATLAB wird jedoch im folgenden Kapitel noch detailliert eingegangen. Der entsprechende Sourcecode für das Szenario "Paralleles lokales Bezugssystem - doppelte Integration" ist in Listing B.4 ersichtlich.

[8] SOLVE ist ein Befehl der Symbolic Math Toolbox. Weitere Informationen zur Symbolic Math Toolbox sind abrufbar unter http://www.mathworks.com/products/symbolic/

5.4.6. Weginterpolation

Nachdem mittels SOLVE die benötigten Unbekannten berechnet wurden, lässt sich der Standort zu jedem Zeitpunkt berechnen, was mit anderen Worten dem potentiell zurückgelegten Weg entspricht.
Der entsprechende Sourcecode für das Szenario "Paralleles lokales Bezugssystem - doppelte Integration" ist in Listing B.5 ersichtlich.

5.5. Grenzen der vorliegenden Lösung

Die vorliegende Interpolationsmethode, und der dazugehörige Algorithmus, kennt prinzipiell nur eine Grenze: die verwendete Hard- und Software. Dabei spielt vor allem die Datenmenge, gleich in mehreren Bereichen, eine Rolle. Einerseits ist der Speicherplatz auf dem verwendeten Palm Treo 650 beschränkt, lässt jedoch eine Aufnahme von rund 37 Stunden zu. Andererseits ist MATLAB nur in der Lage rund 12'612'266 Messpunkte aufzunehmen[9], da keine grösseren Matrizen erstellt werden können.
Eine grössere Einschränkung bietet jedoch die Stromversorgung des Palms. Dieser ist mit dem Standardakku nur gerade in der Lage, das Sensor-Board während ca. 8.5 Stunden[10] ausreichend mit Strom zu versorgen.
Die wohl grösste Einschränkung bietet jedoch der SOLVER von MATLAB. Dieser ist nur beschränkt in der Lage, die Komplexität der Interpolationsgleichungen in Form eines Gleichungssystems zu lösen. Je mehr Datensätze es zu berücksichtigen gilt, je mehr Zeit benötigt der SOLVER um eindeutige Lösungen zu finden. Oftmals scheitert er sogar oder findet gar keine Lösung. Auf diese Probleme wird jedoch im nächsten Kapitel nochmals detailliert eingegangen.

[9] Dies entspricht ungefähr einer maximalen Aufnahmedauer von 10.5 Stunden
[10] Bei eingeschalteten Temperatur-, Beschleunigungs-, Winkelbeschleunigungs- und Magnetsensoren

6. Experimente und Resultate

In diesem Kapitel soll das Testszenario und die daraus resultierten Resultate vorgestellt werden. Gleich zu Beginn muss angemerkt werden, dass die Komplexität der Interpolationsgleichungen in MATLAB zu erheblichen Problemen geführt hat, so dass leider nur ein Szenario, und selbst dieses nur unvollständig, berechnet werden konnte. In diesem Kapitel wird deshalb auch auf die Probleme bei der Weginterpolation eingegangen.

6.1. Testszenario

Um die Genauigkeit der in Kapitel 2 vorgestellten Interpolationsmethoden zu überprüfen, wurde ein einfaches Szenario erarbeitet, auf dessen Basis anschliessend die verschiedenen Algorithmen getestet werden konnten.
Das Szenario basiert auf folgenden Eigenschaften, wobei die Stecke abgelaufen wurde und das Sensor-Board (x-y-Ebene) parallel zur Erdoberfläche in der Hand gehalten wurde:

- Start- und Endpunkt sind identisch (x=0, y=0, z=0)
- Die Anfangsgeschwindigkeit beträgt $0\frac{m}{s}$
- Die Bewegung findet nur in Richtung der x-Achse statt
- Der Referenzpunkt liegt bei x=10
- Die gesamte Aufnahmedauer beträgt 38s
- Der Zeitpunkt der Referenzaufnahme beträgt 16s

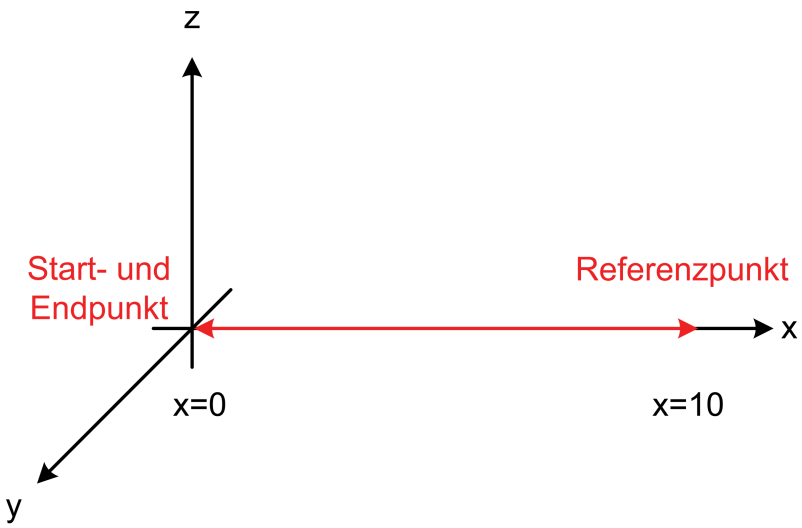

Abbildung 6.1.: Testszenario

6.2. Resultate

Die Sensordatenaufnahmen aus dem soeben vorgestellten Szenario wurde anschliessend durch die verschiedenen Algorithmen berechnet, wobei unterschiedliche Resultate entstanden sind. In den folgenden Unterkapiteln werden deshalb die Resultate aus der Weginterpolation für jede Methode separat präsentiert.

6.2.1. Paralleles lokales Bezugssystem: doppelte Integration

Die Weginterpolation für den einfachsten Berechnungsfall stellt für MATLAB kein all zu grosses Problem dar. Der berechnete Weg (vgl. Abbildung 6.2) ist dabei sehr schön zu erkennen, wobei auch erwartungsgemäss Start-, Referenz-, und Endpunkt auf dem interpolierten Weg liegen.
Erwartungsgemäss sollte diese Berechnungsmethode das schlechteste Resultat liefern, da eine zweifache Integration angewandt wird und Kalibrierungsfehler somit zweifach aufaddiert werden. Zudem wird bei dieser Berechnungsmethode angenommen, dass die Lage des Sensor-Board relativ zur Erdoberfläche nicht verändert wird. Daher wird auch darauf verzichtet, die Winkelinformationen in die Berechnung mit einzubeziehen. Das Sensor-Board wurde im vorliegenden Szenario jedoch bewegt, weshalb Fehler zu erwarten sind.
Der interpolierte Weg (vgl. Abbildung 6.2) führt dabei erwartungsgemäss durch den gegebenen Start-, End- sowie Referenzpunkt. Leider muss jedoch angemerkt werden, dass der interpolierte Weg bis auf x=30m führt, obwohl im Szenario der Referenzpunkt

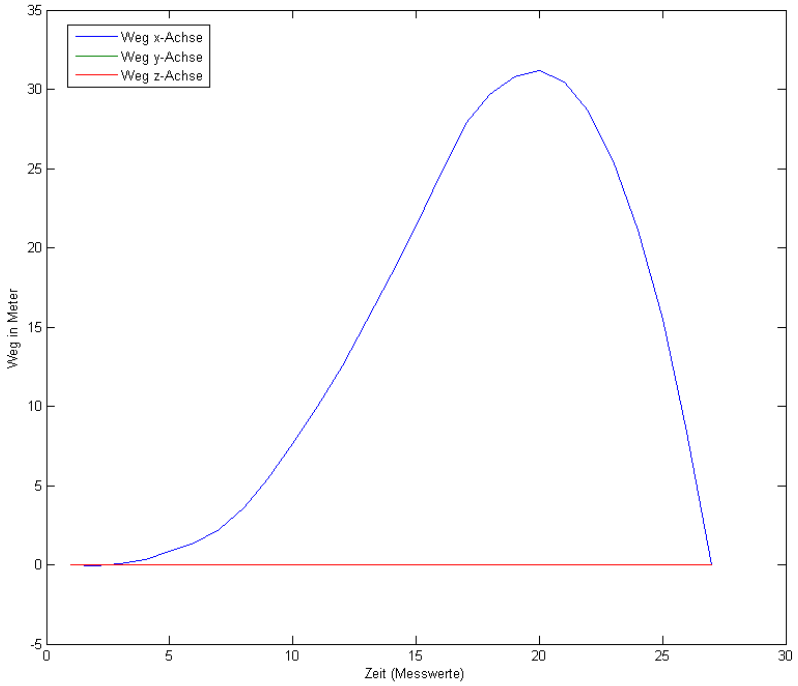

Abbildung 6.2.: Interpolation des zurückgelegten Wegs (Paralleles lokales Bezugssystem: doppelte Integration)

bei x=10m zu einem Zeitpunkt von ca. 11 Messwerten liegt. Ursache dieser Parabel, welche bis nach x=30 reicht, ist der Integrationsdrift. Trotzdem ist das Ergebnis der Interpolation ansehnlich, da es sich um die gleiche Grössenordnung wie beim tatsächlich zurückgelegten Weg handelt.
Für die anderen Berechnungsmethoden, insbesondere der einfachen Integration, wären massiv bessere Ergebnisse zu erwarten.

6.2.2. Paralleles lokales Bezugssystem: einfache Integration

Für den Fall der einfachen Integration sind bei der Interpolation mit MATLAB grosse Probleme aufgetaucht. der SOLVER war leider nicht in der Lage, selbst bei einem Smooth von 1000, die benötigten Interpolationsgrössen zu berechnen.
Die Komplexität des Gleichungssystems ist auch nicht zu unterschätzen, da unter anderem diverse Quadratwurzeln in die Berechnung mit einbezogen werden müssen. Zur Lösung des Problems wurden daher ein Versuch gestartet, die zu berechnenden Wurzeln mittels Heron-Verfahren[11] (auch bekannt als Babylonisches Wurzelziehen) anzunähern.

[11]Das Heron-Verfahren ist ein Spezialfall des Newton-Verfahrens

Die Iterationsvorschrift lautet dabei:

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2}, \text{ wobei es gilt } \sqrt{a} \text{ zu berechnen} \tag{6.1}$$

x_0 ist dabei als $x_0 = 1$ definiert.
Angewandt mit n=1 auf die Interpolationsgleichung 2.25 von Seite 24 wurde erneut versucht, eine Lösung für die Interpolationsgleichungen zu finden. Leider scheiterte MATLAB auch an der massiv vereinfachten Variante des Problems.
Das vereinfachte Gleichungssystem wurde daher in Mathematica[12] übertragen und von dessen Solver gelöst, wobei der in Abbildung 6.3 ersichtliche Weg interpoliert werden konnte. Eine weitere Iteration (n=2) bei der Wurzelannäherung konnte jedoch bereits nicht mehr erfolgreich gelöst werden.
Unglücklicherweise entspricht der interpolierte Weg keineswegs den Erwartungen, da

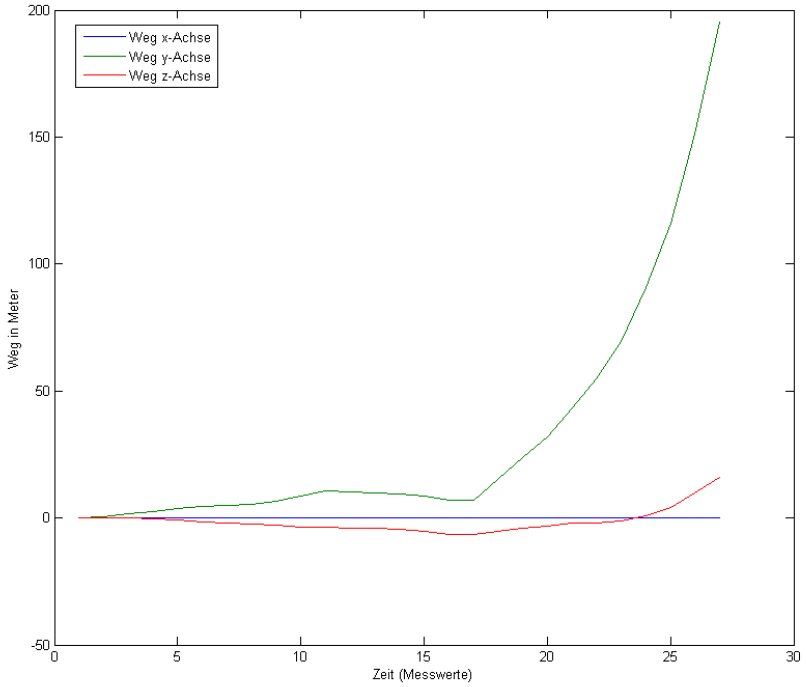

Abbildung 6.3.: Interpolation des zurückgelegten Wegs (Paralleles lokales Bezugssystem: einfache Integration)

sich keine der Achsen bei Referenz- und Endpunkt an die vorgegebenen Randbedingungen hält. Als Ursachen dieser Weginterpolation sind wohl der Integrationsdrift sowie

[12]Weitere Informationen zu Mathematica sind erhältlich auf
http://www.wolfram.com/products/mathematica/index.html

Floating-Point Fehler anzuführen, welche leider in diesem Szenario nicht beseitigt werden konnten.

6.2.3. Frei bewegliches lokales Bezugssystem: doppelte Integration

Die dritte Berechnungsmethode stellte bei der Interpolation der Unbekannten kein all zu grosses Problem dar. MATLAB wie auch Mathematica waren sehr schnell in der Lage, die gesuchten Interpolationsgrössen zu finden. Der interpolierte Weg ist dabei in Abbildung 6.4 ersichtlich.

Leider entspricht auch bei dieser Berechnungsmethode das Ergebnis keineswegs den

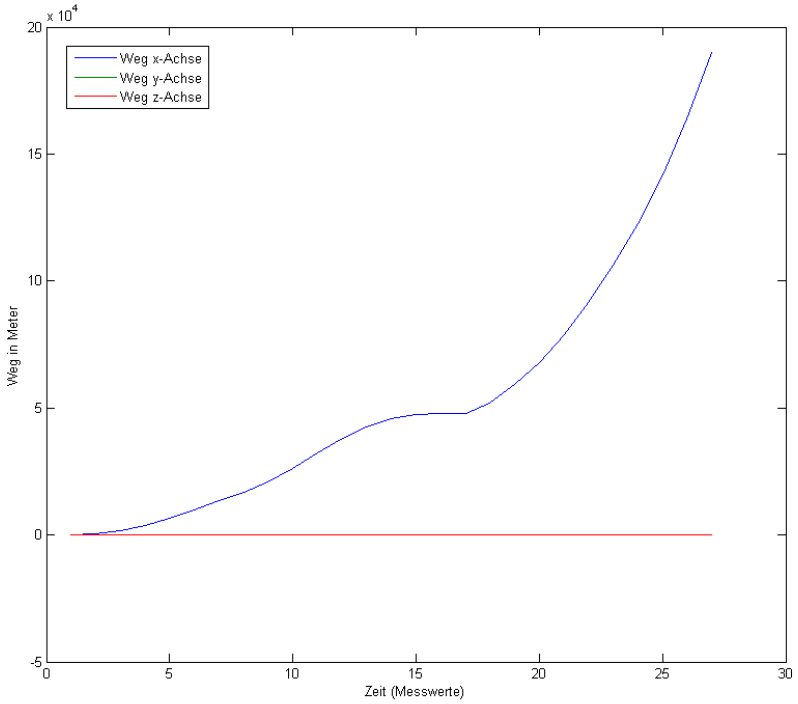

Abbildung 6.4.: Interpolation des zurückgelegten Wegs (Frei bewegliches lokales Bezugssystem: doppelte Integration)

Erwartungen. Zwar verhalten sich die y- und z-Achse entsprechend den Erwartungen, die x-Achse hingegen steigt Wellenförmig von einem x-Wert = 0 bis hin zu einem x-Wert von über 19'000m. Auch hier liegt die Vermutung nahe, das der Integrationsdrift zusammen mit Floating-Point Fehlern zu diesem Ergebnis beigetragen haben. Leider war es auch in diesem Fall nicht möglich, diese Fehler zu beseitigen.

6.2.4. Frei bewegliches lokales Bezugssystem: einfache Integration

Die letzte Interpolationsmethode bereitete schliesslich die grösste Mühe. Leider war es trotz zahlreichen Versuchen und dem Einsatz von hoher Rechenleistung, weder mit MATLAB noch mit Mathematica möglich, die gegebenen Integrationsgleichungen zu berechnen und die Unbekannten zu eruieren. Auch eine Wurzelannäherung mit dem Heron-Verfahren konnte die Performanceprobleme nicht lösen. Es können daher keine Aussagen über dieses Verfahren gemacht werden.

6.3. Diskussion

Aufgrund von zahlreichen Problemen, vor allem mit MATLAB, war es leider nicht möglich, sämtliche Verfahren komplett durch zu rechnen und über verschiedene Testszenarien zu verifizieren.
Die im Kapitel 2 entwickelten Interpolationsmethoden stellten sich als nicht ganz trivial bei der Berechnung heraus und konnten vielfach trotz Annäherungen nicht berechnet werden. Selbst eine Reduktion der Anzahl Messwerte durch einen Smooth brachte nicht in jedem Fall das gewünschte Resultat.
Es müsste daher in einem künftigen Testumfeld vor allem auch Wert auf die Reduktion der Komplexität sowie Performance steigernde Massnahmen[13] gelegt werden.
Trotzdem konnte in diesem Testszenario aufgezeigt werden, dass mit einer Interpolation die Genauigkeit gegenüber einer reinen Extrapolation mit vorheriger Kalibrierung markant gesteigert werden kann.

[13] Beispielsweise wurden die implementierten MATLAB-Funktionen ausgiebig mittels Profile, in Bezug auf die Performance, verbessert.

7. Fazit und Schlusswort

7.1. Fazit

Im Verlauf der vorliegenden Diplomarbeit konnte gezeigt werden, dass eine Weginterpolation theoretisch möglich ist und gelöst werden kann. Dabei wurden in der Theorie entsprechende wohldefinierte Gleichungssysteme präsentiert.
In Bezug auf die Hardware wurde deutlich, dass die Messauflösung der Sensoren vollkommen ausreichend für Navigationszwecke ist. Die Sensorgenauigkeit konnte zudem durch die Korrektur von temperaturbedingten Einflüssen erheblich gesteigert werden. Besonders hervorzuheben ist die Genauigkeit des Magnetsensors. Dadurch, dass keine Integration der Daten für die Berechnung der Winkel notwendig ist, handelt es sich bei diesem Sensor um ein äusserst präzises Navigationsinstrument.
Trotz vieler gelöster Probleme und der theoretischen Herleitung des Problems, wurde mit dieser Arbeit auch an Grenzen, besonders durch die verwendete Software, gestossen. Diese werden im folgenden Unterkapitel detailliert beschrieben.

7.2. Grenzen dieser Arbeit

Entgegen der ursprünglichen Planung sind im Verlauf dieser Arbeit zahlreiche Probleme aufgetaucht, welche leider auch nicht allesamt gelöst werden konnten. Im Folgenden werden deshalb nochmals die wichtigsten Probleme aufgezeigt.

Kalibrierung der Winkelbeschleunigungssensoren

Im Verlauf der Kalibrierung der Winkelbeschleunigungssensoren wurde leider klar, dass diese wider erwarten weniger präzis sind als dies von [Tschanz 2005] beschrieben wurde. Insbesondere die extrem genaue Kalibrierung, welche für deren Einsatz nötig wäre, ist äusserst schwer zu bewerkstelligen, da die Messgenauigkeit (1 Messpunkt) ein vielfaches kleiner ist, als dies für eine Winkelberechnung nötig wäre.
Dadurch war es leider auch nicht möglich, die Winkelbeschleunigungssensoren, nebst

den Magnetsensoren, für die Winkelbestimmung zu verwenden. Der alleinige Einsatz der Magnetsensoren stellt dabei auch keine befriedigende Situation dar, da diese magnetischen Störfeldern unterliegen und somit fehlerhafte Daten liefern können. Eine Lösung zur Fehlererkennung müsste daher dringendst angestrebt werden, wobei die Winkelbeschleunigungssensoren für die reine Fehlererkennung verwendet werden könnten (vgl. hierzu die weiterführenden Informationen auf Seite 77).

Resultate aus den Experimenten

Die Experimente haben gezeigt, dass der Integrationsdrift (vgl. Abbildung 6.2), vor allem bei einer zweifachen Integration, wie erwartet ein grosses Problem darstellt. Durch den Bedarf einer hohen Genauigkeit machten sich unglücklicherweise auch Floating-Point Fehler bemerkbar, welche dazu führten, dass die interpolierten Wegstrecken um ein Vielfaches zu ungenau waren (vgl. Abbildung 6.4).

Aussagekraft der Experimente

Selbstverständlich ist auch die Aussagekraft eines einzelnen Experiments stark an zu zweifeln. Um die Aussagekraft der Experimente zu erhöhen, müssten viel mehr Experimente mit verschiedenen Testszenarien durchgeführt werden. Leider war dies jedoch nicht möglich und auch nicht sinnvoll, aufgrund der bereits erläuterten Probleme bei den Berechnungen der Integrationsgrössen (vgl. hierzu Kapitel 6).

Performance bei der Berechnung

Ein grosses Problem ist auch die fehlende Performance von MATLAB. Trotz dem Einsatz von leistungsstarker Hardware, war MATLAB vielfach nicht in der Lage, die benötigten Integrationsgleichungen als Gleichungssystem zu lösen. Sämtliche Tuning-Versuche brachten leider nur einen geringen Erfolg. In bestimmten Fällen konnte Mathematica jedoch in einem Bruchteil der Zeit das entsprechende Problem lösen, was wenigstens eine Validierung der Resultate zugelassen hat.

7.3. Weiterführende Arbeiten

In diesem Unterkapitel werden weiterführende Arbeiten aufgezeigt. Diese konnten leider aus Gründen von fehlenden zeitlichen Ressourcen in dieser Arbeit nicht mehr durchgeführt werden.

Prüfung von Alternativen zu MATLAB

Leider verursachte MATLAB einen Grossteil der vorhandenen Probleme. Ein alternativer Weg mit Mathematica wurde zwar angestrebt und getestet, jedoch auch nur mit mässigem Erfolg. Eine Prüfung von alternativer Software wie beispielsweise Octave[14], SciLab[15] oder Rlab[16] könnte daher von Vorteil sein. Zudem ist zu Überprüfen, ob die Implementierung der entsprechenden Funktionen aus Performancegründen nicht besser in einer Programmiersprache wie C++ anstelle der MATLAB-eigenen Sprache vorgenommen werden sollte.

Erhöhung der Genauigkeit durch parallele Aufzeichnung

Für zukünftige Aufnahmen sollte Wert darauf gelegt werden, die Aufnahmen nicht nur mit einem Gerät vorzunehmen, sondern gleich mehrere Geräte zu bündeln, um so mehrere Sensordatenaufnahmen des gleichen Testszenario zu erhalten. Damit könnte die Messgenauigkeit durch die Bildung von Mittelwerten über alle Geräte entscheidend verbessert werden.
Eine denkbare Variante wäre dabei beispielsweise die Montage mehrerer Geräte an eine Dachlatte.

Weiterführende Experimente

Zur Validierung der Erkenntnisse aus dieser Arbeit sollten unbedingt noch weiterführende Experimente durchgeführt werden. Dabei sollen die Testszenarien auch zwischen den beiden Varianten "paralleles lokales Bezugssystem" und "frei bewegliches lokales Bezugssystem" unterschieden werden.

[14]Webseite: http://www.octave.org/
[15]Webseite: http://www.scilab.org/
[16]Webseite: http://rlab.sourceforge.net/

Denkbar wären beispielsweise folgende Bewegungsarten für die Testszenarien:

1. Paralleles lokales Bezugssystem
    - Sensor-Board auf Servierwagen
    - Laufen
    - Rennen
    - Autofahrt
2. Frei bewegliches lokales Bezugssystem
    - Laufen
    - Rennen
    - Laufen indoor über mehrere Etagen
    - Autofahrt

Für jede Bewegungsart sollten dabei verschiedene Bewegungsmuster überprüft werden wie beispielsweise:

- Gerade Strecke
- Kreis
- Rechteck (z.B. auf Sportplatz)
- Beliebiges Bewegungsmuster

Einbezug der Winkelbeschleunigungssensoren zur Fehlererkennung

Wie bereits mehrfach angedeutet, unterliegen die Magnetsensoren im täglichen Leben diversen Störfeldern. Um allfällige Störfelder zu erkennen und nicht fehlerhafte Daten in die Berechnungen einzubinden, wäre es deshalb denkbar, die Winkelbeschleunigungssensoren für die Fehlererkennung zu verwenden. Dabei wäre keine Eichung dieser Sensoren notwendig, wodurch auch keine Kalibrierungsfehler auftauchen sollten. Sobald der Magnetsensor eine aussergewöhnlich grosse Winkelveränderung messen würde, müsste anhand der Winkelbeschleunigungssensoren lediglich geprüft werden, ob diese auch einen

grössere Winkelbeschleunigung erfahren haben. Je nach Resultat könnte man anschliessend die von den Magnetsensoren gemessene Winkelveränderung als Störung oder tatsächliche (aussergewöhnlich grosse) Winkelveränderung klassifizieren.

Ongoing Calibration

Ein letzter Ansatz ist die fortwährende Eichung (bzw. Interpolation) der Sensoren. Dabei sollten die Kalibrierungsgrössen vorzu bestimmt werden, wobei die vergangenen Kalibrierungswerte nicht vergessen gehen sollten, sondern in die neue Kalibrierung mit einbezogen werden sollten. Dieser Ansatz erinnert dabei an ein Neuronales Netz, wie es in der künstlichen Intelligenz bestens bekannt ist. Denkbar wären dabei Varianten bei welchen gute Kalibrierungswerte belohnt und schlechte bestraft würden. Ebenso sind Varianten mit und ohne "Forgetting" denkbar.
Dank einem solchen Kalibrierungssystem könnte die Genauigkeit der Interpolation entscheidend verbessert werden. Ausserdem würde das System so vorzu geeicht und sollte nach einiger zeit so gut geeicht sein, dass sogar längere reine Extrapolationsphasen (ohne weitere Eichung) relativ genau vorgenommen werden könnten.

7.4. Schlusswort

Obwohl mit dieser Arbeit leider das ursprüngliche Ziel nicht ganz erreicht wurde, so konnten trotzdem viele Neuerkenntnisse bezüglich einer Ortsbestimmung mittels Interpolation erworben werden. Besonders bedauernswert am nicht Erreichen des eigentlichen Zieles ist dabei die Tatsache, dass die Ursache nicht bei fehlerhaften Methoden oder Annahmen zu suchen ist, sondern bei der Software, welche nicht in Stande war, die theoretisch erarbeiteten Grundlagen in Form einer Interpolation zu berechnen. Leider konnten dadurch auch die Vorteile der Interpolation nicht aufgezeigt werden.
Persönlich konnte ich trotzdem sehr viel von dieser Arbeit profitieren. Vor allem wagte ich mich in Bereiche vor, welche für mich bisher eher tabu waren. Die dabei gesammelten Erfahrungen sind für mich von unschätzbarem Wert.
Trotzt den massiven Problemen bei den Berechnungen, sehe ich positiv und motiviert auf die geleistete Arbeit zurück. Ich bin nach wie vor der festen Überzeugung, dass mit einer Interpolation und der vorhandenen Hardware eine zuverlässige Navigation möglich ist. Sollte es dennoch möglich sein, die entsprechenden Integrationsgleichungen zu lösen (etwa durch bessere Hard- und Software oder eine Vereinfachung der Gleichungen), so kann ich ein Fortführen der Arbeit nur empfehlen.

Abkürzungsverzeichnis

ADC	Analog Digital Converter
CD-ROM	Compact Disc - Read Only Memory
DMC	Digital Magnetic Compass
GND	Ground
GPS	Global Positioning System
Gyro	Gyroskop, Winkelbeschleunigungssensor
INS	Inertiales Navigationssystem
MüM	Meter über Meereshöhe
MATLAB	Matrix Laboratory
PDA	Personal Digital Assistant
PNM	Personal Navigation Module
PNS	Pedestrian Navigation System
RS-232	Radio Sector 232, Serielle Schnittstelle

Literaturverzeichnis

[AnalogDevices 1995] AnalogDevices: *AD22103 - 3.3 V Supply, Voltage Output Temperature Sensor with Signal Conditioning*. Analog Devices, 1995

[AnalogDevices 2004] AnalogDevices: *ADXL320 - Small an Thin ± 5 g Accelerometer*. Analog Devices, 2004

[Büchel 2004] Büchel, Daniela: *Developpement d'une solution de navigation robuste pour l'environnement construit*, Ecole Polytechnique Fédérale de Lausanne, Diplomarbeit, 2004

[Brännström 2002] Brännström, Fredrick: *Positioning techniques alternative to GPS*, Luleå University of Technology, Diplomarbeit, September 2002

[Caruso und Smith 1998] Caruso, Michael J. ; Smith, Dr. Carl H.: A New Perspective on Magnetic Field Sensing / Honeywell Sensor Products. 1998. – Application Note

[DMK/DPK 1995] DMK/DPK: *Mathematik - Physik: Formeln und Tafeln*. 6. durchges. Auflage. Orell Füssli, 1995

[Eberhard 2005] Eberhard, Prof. P.: *Merkblatt zur Vorlesung Maschinendynamik: Euler- und Kardan-Winkel*. 2005. – Institut für Technische und Numerische Mechanik, Universität Stuttgart

[Fang u. a. 2005] Fang, Lei ; Antsaklis, Panos J. ; Montestruque, Luis ; McMickell, M. B. ; Lemmon, Michael ; Sun, Yashan ; Fang, Hui ; Koutroulis, Ioannis ; Haenggi, Martin ; Xie, Min ; Xie, Xiaojuan: Design of a Wireless Assisted Pedestrian Dead Reckoning System - The NavMote Experience. In: *IEEE Transactions on Instrumentation and Measurement, 2005*, 2005

[Ford u. a. 2001] Ford, Tom ; Neumann, Janet ; Bobye, Mike: OEM4 Inertial: An Inertial/GPS Navigation System on the OEM4 Receiver. In: *14th International Technical Meeting of the Satellite Division of the Institute of Navigation, Salt Lake City, Utah, USA*, 2001

[Gabaglio und Merminod 1999] Gabaglio, Vincent ; Merminod, Bertrand: Real-time

calibration of length of steps with GPS and accelerometers. In: *Third International Symposium on Global Navigation Satellite Systems, Genova*, 1999

[Giaglis u. a. 2002] Giaglis, George M. ; Pateli, Ada ; Fouskas, Kostas ; Kourouthanassis, Panos ; Tsamakos, Argiris: On the Potential Use of Mobile Positioning Technologies in Indoor Environments. In: *15th International Conference on Electronic Commerce, Bled, Slovenia*, 2002

[Gilliéron und Merminod 2003] Gilliéron, Pierre-Yves ; Merminod, Bertrand: Personal Navigation System for Indoor Applications. In: *11th World Congress of the International Association of Institutes of Navigation, Berlin, Germany*, 2003

[Honeywell 1995] Honeywell: AN-203: Compass Heading Using Magnetometers / Honeywell Sensor Products. 1995. – Application Note

[Honeywell 1996] Honeywell: AN-201: Set/Reset Pulse Circuits For Magnetic Sensors / Honeywell Sensor Products. 1996. – Application Note

[Honeywell 2000] Honeywell: AN-209: Magnetic Current Sensing / Honeywell Sensor Products. 2000. – Application Note

[Honeywell 2002a] Honeywell: AN-211: Applications of Magnetic Position Sensors / Honeywell Sensor Products. 2002. – Application Note

[Honeywell 2002b] Honeywell: AN-213: Set/Reset Function for Magnetic Sensors / Honeywell Sensor Products. 2002. – Application Note

[Honeywell 2003] Honeywell: *HMC1051/HMC1052/HMC1053 - 1, 2 and 3-Axis Magnetic Sensors*. Honeywell Sensor Products, 2003

[Honeywell 2004] Honeywell: AN-216: Mounting Tips for LCC Magnetic Sensors / Honeywell Sensor Products. 2004. – Application Note

[Ladetto u. a. 2001] Ladetto, Quentin ; Gabaglio, Vincent ; Merminod, Bertrand: Two Different Approaches for Augmented GPS Pedestrian Navigation. In: *International Symposium on Location Based Services for Cellular Users, Locellus, Munich, Germany*, 2001

[Ladetto und Merminod 2002a] Ladetto, Quentin ; Merminod, Bertrand: An Alternative Approach to Vision Techniques - Pedestrian Navigation System based on Digital Magnetic Compass and Gyroscope Integration. In: *6th World Multiconference on Systemics, Cybernetics and Information, Orlando, USA*, 2002

[Ladetto und Merminod 2002b] Ladetto, Quentin ; Merminod, Bertrand: In step

with INS - Navigation for the Blind, Tracking Emergency Crews. In: *GPS World* (2002), Oktober, S. 30–38

[Ladetto u. a. 1999] Ladetto, Quentin ; Merminod, Bertrand ; Terrier, Philippe ; Schutz, Yves: On Foot Navigation: When GPS alone is not enough. In: *Third International Symposium on Global Navigation Satellite Systems*, 1999

[Ladetto u. a. 2002] Ladetto, Quentin ; Seeters, J. van ; Sokolowski, S. ; Sagan, Z. ; Merminod, Bertrand: Digital Magnetic Compass and Gyroscope for Dismounted Soldier Position & Navigation. In: *Military Capabilities enables by Advances in Navigation Sensors, Sensors & Electronics Technology Panel, NATO-RTO meetings, Istanbul, Turkey*, 2002

[NEC/TOKIN 2001] NEC/TOKIN: *CG-L43: Ceramic Gyro*. NEC/TOKIN, 2001

[NEC/TOKIN 2005a] NEC/TOKIN: *CG-L53: Ceramic Gyro*. NEC/TOKIN, 2005a

[NEC/TOKIN 2005b] NEC/TOKIN: *Piezoelectric Devices 2*. NEC/TOKIN, 2005b

[Tschanz 2005] Tschanz, Stephan: *Towards A Framework For An Intertial Navigation System: Position And Velocity Extrapolation Of Motion-Based Data*, Universität Zürich, Institut für Informatik, Diplomarbeit, September 2005

[Weinberg 2002] Weinberg, Harvey: AN-602: Using the ADXL202 in Pedometer and Personal Navigation Applications / Analog Devices. 2002. – Application Note

Anhang

A. Aufgabenstellung

Das Global Positioning System GPS erlaubt es Ort- und Geschwindigkeit für alltägliche Anwendungen genau genug zu bestimmen. Da dieses System leider nur funktioniert, wenn direkter Sichtkontakt zu Satelliten gegeben ist, kann es in Gebäuden nicht benutzt werden. Deshalb wäre es von grossem Nutzen, wenn man, sobald die GPS Informationen wegfallen, den Ort und die Geschwindigkeit anhand anderer Anhaltspunkte bestimmen könnte.
Das Ziel dieser Diplomarbeit ist, eine Orts- und Geschwindigkeits-Abschätzung mittels Bewegungsdaten (Translations- und Rotations-Beschleunigungen, sowie dem Erdmagnetfeld) zu erstellen. Dies beinhaltet einerseits die Erarbeitung theoretischer Grundlagen und andererseits die Durchführung mehrerer Experimente.
In dieser Diplomarbeit wird ein neuer Ansatz verfolgt, indem mittels andauernder Orts-Interpolation die Sensoren vorzu geeicht werden und diese Eichung dann eine verlässliche Ortsbestimmung zulässt, sollte das GPS-Signal ausfallen. Ausserdem ist es für andere Anwendungen äusserst interessant, nicht einen Weg zu extrapolieren, sondern den zurückgelegten Weg zu bestimmen (reine Interpolation).

Im Rahmen dieser Arbeit sollen folgende Tätigkeiten angegangen werden:

1. **Einarbeitung**
   Eine gute Einführung in das Gebiet bietet die Literaturübersicht in [Tschanz 2005]. Für die Interpretation der Messdaten können die Erkenntnisse in [Tschanz 2005] hilfreich sein. Ausserdem stehen alle Datenblätter zur verwendeten Hardware zur Verfügung.

2. **Erarbeitung der theoretischen Grundlagen zur Interpolationsmethode**
   Es soll abgeklärt werden, unter welchen Annahmen/Bedingungen die Interpolationsmethode theoretisch eindeutige Resultate liefern sollte und welche Faktoren zu Abweichungen beitragen können.
   Am Schluss soll ein Algorithmus für die Interpolation vorgestellt werden und wie anhand der darauf berechneten Daten eine weiterführende Ortsbestimmung erreicht werden kann.

3. **Durchführung der Experimente**
   Anhand der oben erarbeiteten Faktoren sollten Experimente zusammengestellt werden, welche die Abhängigkeiten von den jeweiligen Faktoren veranschaulichen

können. Anschliessend sollen die Experimente durchgeführt und analysiert werden. Dies beinhaltet auch Fehlerabschätzungen.

B. MATLAB Funktionen

Es wird darauf verzichtet, an dieser Stelle sämtliche MATLAB-Funktionen komplett abzudrucken. Stellvertretend werden jedoch einige Auszüge abgedruckt, auf welche in der Arbeit Bezug genommen wird. Die kompletten MATLAB-Funktionen befinden sich jedoch auf der beiliegenden CD-ROM (siehe Anhang C).

B.1. Datenimport

Listing B.1: Datenimport

```
% 1) Datenimport der Sensordaten
% ------------------------------
%
fid=fopen(dateiname); % Datei öffnen
temp=textscan(fid, '%s%s%s%s%s%s%s%s%s%s%s%s', 'delimiter', ',', 'endOfLine', '\r\n');
    % Datei nach Muster durchsuchen und Werte in temp ablegen

datasize=size(temp{1},1) - 2; % Anzahl Datensätze (Zeilen) eruieren. Die erste und
    letzte Zeile wird dabei abgeschnitten
sensordaten = zeros(datasize,12); % Leere Matrize für die Speicherung der Datensätze
    anlegen

for n=1:datasize % Für jede Zeile der temp-Matrix
    for i = 1:12 % Für jede Spalte der temp-Matrix
        sensordaten(n,i)=hex2dec(temp{i}(n+1)); % Wert aus der temp-Matrix wird in
            Dezimal umgerechnet und in die sensordaten-Matrix abgelegt
    end
end
ST = fclose(fid); % Datei schliessen
clear temp;
disp '1.1) Datenimport erfolgreich abgeschlossen';

% Datensmooth zur Performancesteigerung
%%%%
if(smoothsize<1) % Prüfen ob Smoothgrösse min = 1
    disp 'Die Smootgrösse muss mindestens = 1 sein!'; % Fehlerausgabe
    error 'execution terminated'; % Programmabbruch
end

temp = zeros(datasize,12); % Temporäre Matrix anlegen für den Smooth
for i=1:12 % Für alle Sensoren
    temp(1:end,i) = smooth(sensordaten(1:end,i),smoothsize); % Smooth durchführen
end

temp2 = zeros(round(datasize/smoothsize)-1,12); % Zweite temporäre Matrix anlegen für
    die Durchschnittswerte
for i=1:12 % Für alle Sensoren
```

```
33      for k=1:size(temp2,1) % Für jeden neuen Datenwert
34      temp2(k,i) = sum(temp(1+(k-1)*smoothsize:k*smoothsize,i))/smoothsize; %
            Durchschnitt berechnen
35      end
36  end
37
38  sensordaten = temp2; % Gesmoothte Werte der sensordaten-Matris zuweisen
39  clear temp; % temp-Matrix zur Performancesteigerung löschen
40  clear temp2; % temp2-Matrix zur Performancesteigerung löschen
41
42  datasize=(size(sensordaten,1)); % Neue Datensize berechnen
43  disp '1.2) Datensmooth erfolgreich abgeschlossen'; % Erfolgsmeldung ausgeben
```

B.2. Temperaturbereinigung

Listing B.2: Temperaturbereinigung

```
1  % 2) Bereinigung von temperaturbedingten Fehler
2  % ------------------------------------------------
3  %
4  for i=1:datasize % Für jede Zeile der sensordaten-Matrix
5      sensordaten(i,2) = 0.0000160424175978 * (sensordaten(i,1)-2261.013569) *
           sensordaten(i,2) + sensordaten(i,2); % y-Achse bereinigen
6      sensordaten(i,3) = 0.0000160424175978 * (sensordaten(i,1)-2261.013569) *
           sensordaten(i,3) + sensordaten(i,3); % x-Achse bereinigen
7      sensordaten(i,4) = 0.0000160424175978 * (sensordaten(i,1)-2261.013569) *
           sensordaten(i,4) + sensordaten(i,4); % z-Achse berienigen
8  end
9
10 disp '2) Temperaturbereinigung erfolgreich vorgenommen'; % Erfolgsmeldung ausgeben
```

B.3. Drehwinkelberechnung

Listing B.3: Drehwinkelberechnung

```
1  % 3) Berechnung der Drehwinkel aus den Magnetsensordaten
2  % ------------------------------------------------
3  %
4  drehwinkel = zeros(datasize,3); % Matrix zur Speicherung der berechneten Drehwinkel je
       Zeitpunkt
5  magoffsetx = 1530.25; % Offset des Magnetsensors der x-Achse statisch festlegen (gemäss
       Kalibrierung)
6  magoffsety = 1424; % Offset des Magnetsensors der y-Achse statisch festlegen (gemäss
       Kalibrierung)
7  magoffsetz = 1493.125; % Offset des Magnetsensors der z-Achse statisch festlegen
       (gemäss Kalibrierung)
8  winkel = sensordaten(1:end,10:12); % Die Winkel in eine eigene Matrix speichern
9  winkel(1:end,1) = winkel(1:end,1)-magoffsetx; % Abzug des Offsets von allen Daten
10 winkel(1:end,2) = winkel(1:end,2)-magoffsetz; % Abzug des Offsets von allen Daten
11 winkel(1:end,3) = winkel(1:end,3)-magoffsety; % Abzug des Offsets von allen Daten
12
13 % Berechnung des Ausgangswinkels ALPHA
14 if(winkel(1,2)>0) % Falls z grösser als NULL
15     alpha0 = 90 - atan(winkel(1,1)/winkel(1,2)) * 180/pi; % Winkel berechnen
16 elseif(winkel(1,2)<0) % Falls z kleiner als NULL
```

```
17      alpha0 = 270 - atan(winkel(1,1)/winkel(1,2)) * 180/pi; % Winkel berechnen
18  elseif(winkel(1,2)==0) % Falls z gleich NULL
19      if(winkel(1,1)<0) % Falls x kleiner als NULL
20          alpha0 = 180; % Winkel gleich 180
21      else % Sonst
22          alpha0 = 0; % Winkel gleich NULL
23      end
24  end
25
26  % Berechnung des Ausgangswinkels BETA
27  if(winkel(1,2)>0) % Falls z grösser als NULL
28      beta0 = 90 - atan(winkel(1,3)/winkel(1,2)) * 180/pi; % Winkel berechnen
29  elseif(winkel(1,2)<0) % Falls z kleiner als NULL
30      beta0 = 270 - atan(winkel(1,3)/winkel(1,2)) * 180/pi; % Winkel berechnen
31  elseif(winkel(1,2)==0) % Falls z gleich NULL
32      if(winkel(1,3)<0) % Falls y kleiner als NULL
33          beta0 = 180; % Winkel gleich 180
34      else % Sonst
35          beta0 = 0; % Winkel gleich NULL
36      end
37  end
38
39  % Berechnung des Ausgangswinkels GAMMA
40  if(winkel(1,3)>0) % Falls y grösser als NULL
41      gamma0 = 90 - atan(winkel(1,1)/winkel(1,3)) * 180/pi; % Winkel berechnen
42  elseif(winkel(1,3)<0) % Falls y kleiner als NULL
43      gamma0 = 270 - atan(winkel(1,1)/winkel(1,3)) * 180/pi; % Winkel berechnen
44  elseif(winkel(1,3)==0) % Falls y gleich NULL
45      if(winkel(1,1)<0) % Falls x kleiner als NULL
46          gamma0 = 180; % Winkel gleich 180
47      else % Sonst
48          gamma0 = 0; % Winkel gleich NULL
49      end
50  end
51
52  drehwinkel = zeros(datasize,3); % Anlegen einer Matrix zur Speicherung aller
        Drehwinkelveränderungen
53
54  % Berechnung der Winkelveränderungen zu jedem Zeitpunkt
55  for i=1:datasize
56
57      % Winkelveränderung ALPHA zum Zeitpunkt i
58      if(winkel(i,2)>0) % Falls z grösser als NULL
59          drehwinkel(i,1) = (90 - atan(winkel(i,1)/winkel(i,2)) * 180/pi) - alpha0; %
                Winkelveränderung berechnen
60      elseif(winkel(i,2)<0) % Falls z kleiner als NULL
61          drehwinkel(i,1) = (270-atan(winkel(i,1)/winkel(i,2))*180/pi) - alpha0; %
                Winkelveränderung berechnen
62      elseif(winkel(i,2)==0) % Falls z gleich NULL
63          if(winkel(i,1)<0) % Falls x kleiner als NULL
64              drehwinkel(i,1) = 180 - alpha0; % Winkelveränderung berechnen
65          else % Sonst
66              drehwinkel(i,1) = 0 - alpha0; % Winkelveränderung berechnen
67          end
68      end
69
70      % Winkelveränderung BETA zum Zeitpunkt i
71      if(winkel(i,2)>0) % Falls z grösser als NULL
72          drehwinkel(i,2) = (90 - atan(winkel(i,3)/winkel(i,2)) * 180/pi) - beta0; %
                Winkelveränderung berechnen
73      elseif(winkel(i,2)<0) % Falls z kleiner als NULL
74          drehwinkel(i,2) = (270 - atan(winkel(i,3)/winkel(i,2)) * 180/pi) - beta0; %
                Winkelveränderung berechnen
75      elseif(winkel(i,2)==0) % Falls z gleich NULL
76          if(winkel(i,3)<0) % Falls y kleiner als NULL
77              drehwinkel(i,2) = 180 - beta0; % Winkelveränderung berechnen
```

```
78          else % Sonst
79              drehwinkel(i,2) = 0 - beta0; % Winkelveränderung berechnen
80          end
81      end
82
83      % Winkelveränderung GAMMA zum Zeitpunkt i
84      if(winkel(i,3)>0) % Falls y grösser als NULL
85          drehwinkel(i,3) = (90 - atan(winkel(i,1)/winkel(i,3)) * 180/pi) - gamma0; %
                Winkelveränderung berechnen
86      elseif(winkel(i,3)<0) % Falls y kleiner als NULL
87          drehwinkel(i,3) = (270 - atan(winkel(i,1)/winkel(i,3)) * 180/pi) - gamma0; %
                Winkelveränderung berechnen
88      elseif(winkel(i,3)==0) % Falls y gleich NULL
89          if(winkel(i,1)<0) % Falls x kleiner als NULL
90              drehwinkel(i,3) = 180 - gamma0; % Winkelveränderung berechnen
91          else % Sonst
92              drehwinkel(i,3) = 0 - gamma0; % Winkelveränderung berechnen
93          end
94      end
95  end
96
97  clear winkel; % Matrix Winkel löschen zur Performancesteigerung
98
99  drehwinkel = drehwinkel*pi/180; % Alle Drehwinkel in Bogenmass umrechnen
100
101 disp '3) Drehwinkel erfolgreich berechnet'; % Erfolgsmeldung ausgeben
```

B.4. Berechnung der Interpolationsgrössen (Paralleles lokales Bezugssysstem - doppelte Integration)

Listing B.4: Berechnung der Interpolationsgrössen (Paralleles lokales Bezugssysstem - doppelte Integration)

```
1  % 4) Berechnung der Interpolationsgrössen: Streckfaktor und Offset (je Achse)
2  % ------------------------------------------------------------------------
3  %
4
5  tvalue = dauer/datasize; % Berechnung des Delta-t
6
7  k1value = datasize; % Anzahl Zeitwerte von Start bis Ende
8  k2value = round(treferenz/tvalue); % Anzahl Zeitwerte von Start bis Referenzpunkt
9
10 % Zur Performancesteigerung berechnen wir die Summen iim voraus, da diese
11 % in diesem Fall keine Unbekannten beinhalten
12 sum1x = 0; % Initialisierung mit Nullwert
13 sum1y = 0; % Initialisierung mit Nullwert
14 sum1z = 0; % Initialisierung mit Nullwert
15 sum2x = 0; % Initialisierung mit Nullwert
16 sum2y = 0; % Initialisierung mit Nullwert
17 sum2z = 0; % Initialisierung mit Nullwert
18
19 for i=1:k1value % Für alle Zeitwerte von Start bis Ende
20     sum1x = sum1x + ((k1value-i+1)*sensordaten(i,3)); % Berechnung der Summe für die
           x-Achse
21     sum1y = sum1y + ((k1value-i+1)*sensordaten(i,2)); % Berechnung der Summe für die
           y-Achse
22     sum1z = sum1z + ((k1value-i+1)*sensordaten(i,4)); % Berechnung der Summe für die
           z-Achse
23 end
```

```
24
25 for i=1:k2value % Für alle Zeitwerte von Start bis Referenzpunkt
26     sum2x = sum2x + ((k2value-i+1)*sensordaten(i,3)); % Berechnung der Summe für die
       x-Achse
27     sum2y = sum2y + ((k2value-i+1)*sensordaten(i,2)); % Berechnung der Summe für die
       y-Achse
28     sum2z = sum2z + ((k2value-i+1)*sensordaten(i,4)); % Berechnung der Summe für die
       z-Achse
29 end
30
31 offsetterm1 = (k1value+1)*k1value/2; % Berechnung des Offsetterms für Start bis Ende
32 offsetterm2 = (k2value+1)*k2value/2; % Berechnung des offsetterms für Start bis
       Referenzpunkt
33
34 % Schlussposition
35 sxende = sende(1,1); % Zuweisung der Eingabewerte (aus einer Matrix) in eine eigene
       Variable
36 syende = sende(1,2); % Zuweisung der Eingabewerte (aus einer Matrix) in eine eigene
       Variable
37 szende = sende(1,3); % Zuweisung der Eingabewerte (aus einer Matrix) in eine eigene
       Variable
38
39 % Referenzposition
40 sxreferenz = sreferenz(1,1); % Zuweisung der Eingabewerte (aus einer Matrix) in eine
       eigene Variable
41 syreferenz = sreferenz(1,2); % Zuweisung der Eingabewerte (aus einer Matrix) in eine
       eigene Variable
42 szreferenz = sreferenz(1,3); % Zuweisung der Eingabewerte (aus einer Matrix) in eine
       eigene Variable
43
44 % Startposition
45 sxstart = sstart(1,1); % Zuweisung der Eingabewerte (aus einer Matrix) in eine eigene
       Variable
46 systart = sstart(1,2); % Zuweisung der Eingabewerte (aus einer Matrix) in eine eigene
       Variable
47 szstart = sstart(1,3); % Zuweisung der Eingabewerte (aus einer Matrix) in eine eigene
       Variable
48
49 % Startgeschwindigkeit
50 vxstart = vstart(1,1); % Zuweisung der Eingabewerte (aus einer Matrix) in eine eigene
       Variable
51 vystart = vstart(1,2); % Zuweisung der Eingabewerte (aus einer Matrix) in eine eigene
       Variable
52 vzstart = vstart(1,3); % Zuweisung der Eingabewerte (aus einer Matrix) in eine eigene
       Variable
53
54 % Definition der einzelnen Gleichungen
55 equation1 = sprintf('%.10g = %.10g^2 * sx * %.10g - %.10g^2 * sx * %.10g * ox + %.10g *
       %.10g * %.10g + %.10g', sxende, tvalue, sum1x, tvalue, offsetterm1, k1value,
       vxstart, tvalue, sxstart);
56 equation2 = sprintf('%.10g = %.10g^2 * sy * %.10g - %.10g^2 * sy * %.10g * oy + %.10g *
       %.10g * %.10g + %.10g', syende, tvalue, sum1y, tvalue, offsetterm1, k1value,
       vystart, tvalue, systart);
57 equation3 = sprintf('%.10g = %.10g^2 * sz * %.10g - %.10g^2 * sz * %.10g * oz + %.10g *
       %.10g * %.10g + %.10g', szende, tvalue, sum1z, tvalue, offsetterm1, k1value,
       vzstart, tvalue, szstart);
58 equation4 = sprintf('%.10g = %.10g^2 * sx * %.10g - %.10g^2 * sx * %.10g * ox + %.10g *
       %.10g * %.10g + %.10g', sxreferenz, tvalue, sum2x, tvalue, offsetterm2, k2value,
       vxstart, tvalue, sxstart);
59 equation5 = sprintf('%.10g = %.10g^2 * sy * %.10g - %.10g^2 * sy * %.10g * oy + %.10g *
       %.10g * %.10g + %.10g', syreferenz, tvalue, sum2y, tvalue, offsetterm2, k2value,
       vystart, tvalue, systart);
60 equation6 = sprintf('%.10g = %.10g^2 * sz * %.10g - %.10g^2 * sz * %.10g * oz + %.10g *
       %.10g * %.10g + %.10g', szreferenz, tvalue, sum2z, tvalue, offsetterm2, k2value,
       vzstart, tvalue, szstart);
61
```

```
62 disp '4.1) Gleichungen erfolgreich erstellt';
63
64 solution = solve(equation1,equation2,equation3,equation4,equation5,equation6); % Suche
       nach den Unbekannten durch 6 Gleichungen und 6 Unbekannte
65
66 if (size(solution,1)==0) % Falls keine Lösung gefunden wurde, wird eine Fehlermeldung
       ausgegeben
67     disp(' ');
68     disp(' ');
69     disp('FEHLER BEIM SUCHEN VON EINDEUTIGEN LÖSUNGEN FÜR DIE OFFSETS UND DEN
           STRECKFAKTOR!');
70     error 'execution terminated';
71 end
72
73 disp '4.2) Berechnung der Interpolationsgrössen erfolgreich vorgenommen: Unbekannte
       berechnet'; % Erfolgsmeldung ausgeben
```

B.5. Weginterpolation (Paralleles lokales Bezugssysstem - doppelte Integration)

Listing B.5: Weginterpolation (Paralleles lokales Bezugssysstem - doppelte Integration)

```
1  % 5) Weginterpolation
2  % ---------------------
3  %
4
5  sx = double(solution.sx); % Zuweisung der gefundenen Lösung in eine Variable inkl.
       Umrechnung nach Double
6  sy = double(solution.sy); % Zuweisung der gefundenen Lösung in eine Variable inkl.
       Umrechnung nach Double
7  sz = double(solution.sz); % Zuweisung der gefundenen Lösung in eine Variable inkl.
       Umrechnung nach Double
8  ox = double(solution.ox); % Zuweisung der gefundenen Lösung in eine Variable inkl.
       Umrechnung nach Double
9  oy = double(solution.oy); % Zuweisung der gefundenen Lösung in eine Variable inkl.
       Umrechnung nach Double
10 oz = double(solution.oz); % Zuweisung der gefundenen Lösung in eine Variable inkl.
       Umrechnung nach Double
11
12 weg=zeros(datasize,3); % Initialisierung der Lösungsmatrix
13
14 for k=1:k1value % Für alle Zeitwerte von Start bis Ende
15     tempsumx = 0; % Temporäre Summe gleich NULL
16     tempsumy = 0; % Temporäre Summe gleich NULL
17     tempsumz = 0; % Temporäre Summe gleich NULL
18     for i=1:k % Verschachtelte Summe durchlaufen
19         tempsumx = tempsumx+((k-i+1)*sensordaten(i,3)); % Summe für die x-Achse
               berechnen
20         tempsumy = tempsumy+((k-i+1)*sensordaten(i,2)); % Summe für die y-Achse
               berechnen
21         tempsumz = tempsumz+((k-i+1)*sensordaten(i,4)); % Summe für die z-Achse
               berechnen
22     end
23     weg(k,1) = tvalue^2 * sx * tempsumx - tvalue^2 * sx * ((k+1)*k)/2 * ox + k *
           vxstart * tvalue + sxstart; % Position auf der x-Achse zum Zeitpunkt k
           berechnen und in die Lösungsmatrix speichern
24     weg(k,2) = tvalue^2 * sy * tempsumy - tvalue^2 * sy * ((k+1)*k)/2 * oy + k *
           vystart * tvalue + systart; % Position auf der y-Achse zum Zeitpunkt k
           berechnen und in die Lösungsmatrix speichern
```

```
25      weg(k,3) = tvalue^2 * sz * tempsumz - tvalue^2 * sz * ((k+1)*k)/2 * oz + k *
            vzstart * tvalue + szstart; % Position auf der z-Achse zum Zeitpunkt k
            berechnen und in die Lösungsmatrix speichern
26  end
27
28  disp '5) Weg erfolgreich interpoliert'; % Erfolgsmeldung ausgeben
```

C. Inhalt der CD-ROM

Auf der beiliegenden CD-ROM befinden sich nebst der kompletten Arbeit als PDF-Datei auch die MATLAB-Funktionen und vieles mehr. Für eine genaue Inhaltsangabe betrachte man Tabelle C.1, in welcher die genaue Verzeichnisstruktur inkl. Inhaltsangabe ersichtlich ist.

Verzeichnis	**Inhalt**
Diplomarbeit	Komplette Diplomarbeit als PDF-Datei
LaTeX	Komplette Diplomarbeit als LaTeX-Dateien inkl. allen Bildern im PNG- und EPS-Format
Zusammenfassung	Zusammenfassung der Diplomarbeit in Deutsch und Englisch je als PDF- und TXT-Datei (unformatierter Text)
MATLAB	Sämtliche implementierten MATLAB-Funktionen als m-Dateien
Sensordaten	Sämtliche Sensordaten welche verwendet wurden u. A. für die Kalibrierung
Literatur	Sämtliche verwendete und zitierte Literatur als PDF-Dateien

Tabelle C.1.: Verzeichnisstruktur sowie Inhalt der CD-ROM

D. CD-ROM