Bachelorarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Intelligente Systeme und Robotik

Erweiterter Kalman-Filter zur Orientierungsabschätzung von humanoiden Robotern

Simon Gene Gottlieb
Matrikelnummer: 4379660
gene@staubsaal.de

Betreuer:
Prof. Dr. Raúl Rojas
Dr. Hamid Reza Mobalegh

Berlin, 30. September 2013

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

30. September 2013

Simon Gene Gottlieb

Simon Gene Gottlieb

Zusammenfassung

Das Einsatzgebiet von Robotern wird stetig größer. Es existieren bereits sehr spezialisierte Roboter in der Industrie, welche unter kontrollierten Bedingungen arbeiten. Die Entwicklung bringt einfache Roboter auch in den Alltag des Haushaltes. So gibt es Roboter, die im Haus staubsaugen oder im Garten den Rasen mähen. Damit Roboter unterschiedliche und komplexere Aufgaben erledigen können, wird an humanoiden Robotern geforscht.

Das Abschätzen der eigenen Orientierung ist für humanoide Roboter von wichtiger Bedeutung. Es ist eine wesentliche Grundlage, um einen stabilen Gang, eine räumliche Zuordnung von Objekten oder auf unerwartete Impulse von Außen zu reagieren. In dieser Arbeit wird ein erweiterter Kalman-Filter, der aus Daten des Gyroskops und Beschleunigungssensors die Orientierung bestimmen soll, implementiert. Das Ziel ist es, die Vorteile von Gyroskop und Beschleunigungssensors zu vereinen, um in Echtzeit präzise Daten zu erhalten.

Um das gewünschte Ergebnis mittels erweiterten Kalman-Filters zu erhalten, ist ein vorsichtiges Wählen von Parameter notwendig. In dieser Arbeit werden verschiedene Parametersätze und Ihre Auswirkungen auf die Vereinigung der Vorteile von Gyroskop und Beschleunigungssensor evaluiert.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

In der Robtik gibt es eine schnelle fortschreitende Entwicklung. Roboter werden größer, schneller und können immer kompliziertere Aufgaben bewältigen. Eine Strömung der Robotik fokusiert auf Roboter, die sich in einer Umgebung, die für den Menschen geschaffen wurde, zurecht finden. Diese Umgebung ist nicht konstant, sondern unterliegt ständigen Veränderungen. Daher ist ein humanoider Roboter, also ein Roboter der eine menschenähnliche Gestalt aufweist, besonders interessant. Eine große Herausforderung ist das Laufen auf zwei Beinen. Hierfür ist es hilfreich, wenn die Orientierung des Roboters bekannt ist. Dadurch können die Beinbewegungen besser gesteuert werden, wodurch ein stabiler und aufrechter Gang erreicht werden kann.
Im FUmanoids Team wird an humanoiden Robotern geforscht. Bei der Entwicklung und Weiterentwicklung von Robotern ist das Ziel, die Eigenschaften eines Roboters besser zu verstehen, um dann leistungsfähigere und robustere Roboter zu bauen. Es wird außerdem auf eine kostengünstige Entwicklung und Reparatur geachtet.
In der alten 2012-Roboter-Generation der FUmanoids kam eine Inertial Measurement Unit (IMU) zum Einsatz. Diese IMU besitzt einen integrierten Prozessor inklusive Software, welche die Berechnung für die Orientierungsabschätzung übernimmt. Die elektrischen Eigenschaften der IMU war den rauen Gegebenheiten im RoboCup und im Labor nicht gewachesen und ist deshalb häufig kaputt gegangen. Um diese IMU auszutauschen wird der integrierte Prozessor notwendigerweise mit ausgetauscht. Dies verursacht unnötige Kosten.
In der neuen 2013-Roboter-Generation wurde deshalb diese IMU gegen eine andere IMU ausgetauscht. Die neue IMU besitzt keinen eigenen Prozessor, ist dafür aber deutlich günstiger. Bei einem Defekt der IMU ist diese austauschbar, ohne dass ein Prozessor mit ausgetauscht wird.
Zur Auswertung der IMU-Daten, wird diese an einen Prozessor angeschlossen. Dieser übernimmt die nötigen Berechnungen um die Orientierung abzuschätzen. Außerdem kann dieser auch andere Teile des Roboters ansteuern, erfüllt somit mehrere Zwecke.
Bisher ist die Einsicht in die Funktionsweise der Orientierungsabschätzung der 2012-Roboter-Generation sehr begrenzt, da vorallem auf fertige Komponenten gesetzt wurde. Für die 2013-Generation muss neue Software geschrieben werden. Diese muss aus den Daten, die von der IMU geliefert werden, das Rauschen entfernen. Desweiteren müssen die verschiedenen Daten zusammengeführt werden, um eine zuverlässige Aussage über die aktuelle Orientierung des Roboters zu machen.
Die Motivation ist es, auftretene Defekte leichter und kostengünstiger zu beheben. Außerdem soll der Prozessor nicht nur IMU spezifische Berechnungen

übernehmen, sonder auch für weitere Aufgaben genutzt werden. Hierdurch werden auch Kosten gespart. Darüber hinaus ist eine detaillierte Auseinandersetzung mit den Parameter für die Abschätzung der Orientierung notwendig. Dies soll auch zu verbesserten Abschätzungen führen.

1.2 Ziel der Arbeit

Der Gyroskop- und Beschleunigungssensor der IMU haben verschiedene Vor- und Nachteile, die für die Abschätzung der Orientierung wichtig sind. Während z.B. die Daten des Gyroskops einen Drift aufweisen, sind die Beschleunigungsdaten stark verrauscht.
In dieser Arbeit soll ein erweiterter Kalman-Filter erarbeitet werden, mit dem Ziel die Vorteile der beiden Sensoren zu vereinen, um damit die Orientierung in Echtzeit abzuschätzen. Dieser Filter soll die Daten filtern und in einem Zustandsmodell vereinen. Im Zustandsmodell wird die Orientierung mit Hilfe von Quaternion abgebildet. Am Ende wird evaluiert welchen Einfluss der erweiterte Kalman-Filter und die verschiedenen Parameter dieses Filters auf die Eigenschaften "Verzögerungen", "Drift" und "Rauschen" hat. Um zu zeigen wie der erweiterter Kalman-Filter die Vorteile der Sensoren vereint, werden diese Daten direkt mit den Daten aus dem Gyroskop und dem Beschleunigungssensor verglichen.

1.3 Aufbau der Arbeit

Nachdem im vorherigem Abschnitt die Motivation und Ziele erläutert wurden, gibt es an dieser Stelle eine genauere Erklärung der Struktur und Aufbau dieser Arbeit.

Kapitel 2 - Andere Arbeiten Zuerst werden Arbeiten, die aus der gleichen Arbeitsgruppe stammen und sich mit einer ähnlichen Plattform beschäftigen, vorgestellt. Danach werden die wichtigsten Arbeiten über Orientierungsabschätzung und erweiterten Kalman-Filtern vorgestellt.

Kapitel 3 - Plattform Die Plattform der FUmanoids wird vorgestellt. Es gibt eine allgemeine Beschreibung der Roboter, so wie eine genaue Erläuterung der für diese Arbeit wichtigen Module.

Kapitel 4 - Grundlagen An dieser Stelle werden die mathematischen Grundlagen erläutert. Es wird der Kalman-Filter und der erweiterte Kalman-Filter erklärt. Danach werden Quaternionen vorgestellt und ihre Vorteile gegenüber Rotationsmatrizen sowie Eulerwinkel dargelegt.

Kapitel 5 - Modellierung Zuerst wird das Problem der Orientierungsabschätung formuliert und theoretisch gelöst. Dazu wird ein Modell

gefunden, dass den Zustand des Roboters beschreibt. Der erweiterte Kalman-Filter und Quaternionen werden in einen Zusammenhang gebracht.

Kapitel 6 - Implementierung Die konkrete Implementierung und ihre Abhängigkeiten mit verschiedenen Hardware- und Softwarekomponenten wird dargelegt und Lösungsvorschläge vorgestellt. Es wird erklärt welche Parameter wie kalibriert werden können.

Kapitel 7 - Evaluierung Die vorgestellte Implementierung wird getestet und in verschiedenen Versuchen evaluiert. Desweiteren werden die Schwierigkeiten der Daten des Magnetometers erörtert.

Kapitel 8 - Zusammenfassung und Ausblick In diesem Kaptel wird der Inhalt dieser Arbeit zusammengefasst und ein Ausblick über weitere Entwicklungsmöglichkeiten gegeben.

2 Andere Arbeiten

Zur Bestimmung der Orientierungen gibt es bereits viele Arbeiten. Diese unterscheiden sich in vielen Dinge, z.B.: Ob es nur eine Simulation oder ein konkretes System ist, welche Sensoren eingesetzt werden oder wie die Orientierung modelliert wird. Die richtige Wahl hängt meist vom System ab, das beschrieben wird und den Einschränkung, die das System hat.

Borrmann [4] hat für das Berlin United - Racing Team eine inertiale Messeinheit für ein autonomes Modellfahrzeug implementiert. Dieses Modellfahrzeug baut auf einer ähnlichen Hardwarearchitektur auf, wie die FUmanoid Roboter. So kommt der gleichen IMU-Sensoren zum Einsatz. Außerdem wird ein Teil der Software in beiden Teams eingesetzt. In der Arbeit von Borrmann, wird die Annahme gemacht, dass das Fahrzeug immer auf einer ebenen Fläche fährt und nicht umkippt. Es findet also nur eine Rotation in in einer Ebenen Fläche statt. Der Ansatz reicht nicht, um die Orientierung für einen humanoiden Roboter abzubilden, da dieser sich nicht immer in einer aufrechten Position befindet. Wir benötigen für unseren Roboter eine Rotation im 3D-Raum. Die Sensoren, die im Modellfahrzeug und unseren Robotern verwendet werden, hat Borrmann detailliert beschrieben.

Bachman et al. [2] präsentiert eine Filtermethode, die Daten aus MARG Sensoren (Magnetic, Angular Rate und Gravity) nutzt. Zum Filtern verwenden sie einen Komplementär-Filter, welcher auf Quaternionen aufbaut. Im Wesentlichen ist das Ziel ihrer Arbeit den auftretenden Drift der Orientierung, der bei der Bestimmung mit einem Gyroskop anfällt, über weitere Sensoren wie dem Magnetometer und Beschleunigungssensor zu kompensieren. Zur Modellierung der Orientierung nutzten sie Quaternionen anstatt einer Eulerwinkel-Repräsentation. Hiermit vermeiden sie Singularitätsprobleme (Gimbal Lock). Den Komplementär-Filter haben sie auf einem konkreten Test-System implementiert und getestet.

Auf die Arbeit von Bachman et al. [13] bauen Marins et al. auf. Sie nutzten auch MARG-Sensoren und Quaternionen zur Modellierung. Anstelle eines Komplementär-Filters nutzten sie einen erweiterten Kalman-Filter. Durch diesen erreichen sie eine geringere Ordnung des Ausgabevektors, sowie einen geringeren Rechenaufwand.

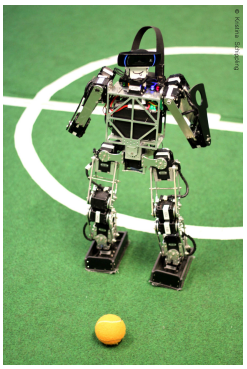
Es gibt verschiedene Kalman-Filter mit unterschiedlichen stärken. LaViola Jr. [11] vergleicht den erweiterten Kalman-Filter (EKF) mit einem unscented Kalman-Filter (UKF). Er stellt fest das in vielen Anwendungen der UKF zu besseren Ergebnissen führt und verweist auf die Artikel [7] [19] [18] und [12]. Als größten Vorteil von UKF führt er an, dass im Gegensatz zum EKF keine händische Berechnung der Jacobi-Matrix notwendig ist, schränkt aber

auch ein, dass die Bestimmung in seinem Anwendungsfall nicht besonders schwierig ist.
Im Vergleich von UKF und EKF zur Orientierungsbestimmung kann er nicht feststellen, dass der UKF bessere Arbeit verrichtet als der EKF. Darüber hinaus stellt er heraus, dass der UKF mehr zusätzliche Berechnungen benötigt und die Dynamik von Quaternionen eine quasi lineare Natur besitzt, weshalb er EKF als die besseren Wahl für dieses Problem hält.

3 Plattform

3.1 Übersicht

Das FUmanoid-Team nimmt regelmäßig in der Kid Size League des RoboCup-Turniers erfolgreich teil[1]. Die Roboter werden mit dem Fokus, den Regeln[5] der Kid Size League zu entsprechen, entwickelt, weshalb die Hardware der FUmanoids regelmäßig überarbeitet wird. Die aktuellen Änderungen betreffen die komplette Mechanik so wie die gesamte Elektronik. In Abbildung 1a ist ein FUmanoid der aktuellen 2013 Generation zu sehen. Dieser Roboter hat folgende Eigenschaften:



(a) 2013 FUmanoid Roboter beim Laufen. (Foto: Kristina Schippling)

Abbildung 1

- 20 Motoren
- eine Logitech c920 Webcam (640x480)
- Odroid-X2 mit 4x 1.7GHz
- 2x ARM Cortex-M4 32b MCU+FPU
- ca. 4,5kg schwer (inkl. Akku)
- 100Hz Motion-Updates
- 30Hz Cognition-Updates
- eine Inertial Measurement Unit (IMU)
- LAN, Wifi
- Federngestützte Hüften
- Linaro
- Software entwickelt in C, C++, Java, Python, Perl und Xabsl

3.2 Software-Module

Für die FUmanoid Roboter sind unterschiedliche Software-Module im Einsatz, die in Abbildung 2 zu sehen sind. An erster Stelle steht die FUmanoid Software, die auf dem Hauptprozessor des Roboters läuft. Dies ist die Kernsoftware, welche die meiste Arbeit verrichtet. Es werden von hier Funktionen wie Laufen, Kamera, Verhalten und vieles mehr ausgewertet und gesteuert.

Der Hauptprozessor, welcher vier Kerne besitzt, sitzt in einer Odroid-X2 Platine. An diesem ist ein Motorboard angeschlossen, das mit zwei ARM Cortex-M4 Prozessoren bestückt ist. Die Hauptaufgabe des ersten ARM-Prozessors ist die Kommunikation mit den Motoren, sowie die Kommunikation zwischen Odroid und dem zweiten ARM-Prozessor. An dem zweiten Pro-

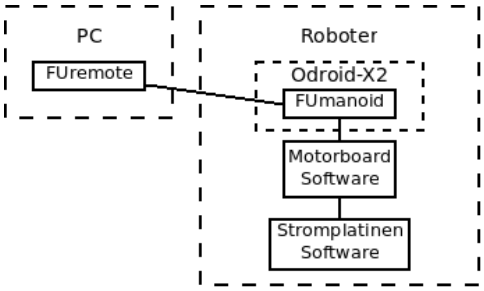
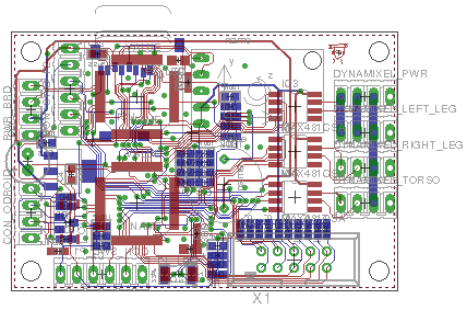


Abbildung 2: Wichtige Software-Module eines FUmanoid Roboters

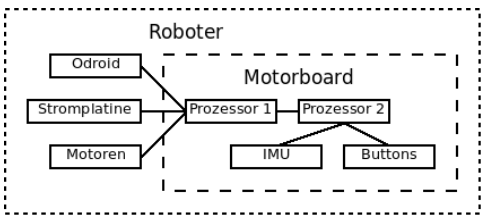
zessor ist eine IMU angeschlossen, welche alle wichtigen Daten für die Orientierungsbestimmung liefert. Ein wesentlicher Teil dieser Arbeit beschäftigt sich mit der IMU und der Software auf diesem Prozessor. In Abbildung 3b ist eine schematische Darstellung des Motorboards zu sehen.
Für die Stromversorgung ist eine eigene Platine verbaut. Diese überwacht kontinuierlich die Spannungsversorgung und ist in der Lage, im Fehlerfall den Strom für Teile oder des gesamten Roboters abzuschalten.
Zur Überwachung des Roboters wird die Software FUremote eingesetzt. FUremote ermöglicht Sensorwerte, Modellierungen und Entscheidungen einzusehen. Es können außerdem verschiedene Parameter des Roboters über die Benutzteroberfläche ausgelesen und konfiguriert werden. Der Roboter ist nicht abhängig von dieser Software und ein Betrieb ist auch ohne FUremote möglich.

3.3 Motorboard

In Abbildung 3a ist der Schaltplan des Motorboards zusehen. In Abbildung 3b ist die Beziehung der Elektronik noch einmal schemenhaft dargestellt. Bei der IMU handelt es sich um die Bausteine L3GD20 und LSM303DLHC von STMicroelectronics. Im L3GD20 ist ein 3-Achsen Gyroskop sowie ein



(a) Schaltplan vom Motorboard



(b) Schematischer Aufbau des Motorboard

Abbildung 3

Temperatursensor verbaut, während im LSM303DLHC Baustein ein 3-Achsen Beschleunigungssensor, ein 3-Achsen Magnetometer und ein Temperatursensor verbaut ist. Beide Bausteine sind über den I^2C-Bus des zweiten ARM-Prozessors angeschlossen.

3.3.1 Beschleunigungssensor

Der verbaute Beschleunigungssensor misst entlang dreier Achsen, die auf den Sensor wirkende Beschleunigung. Tatsächlich wird dies über die Messung der Trägheit auf eine Masse, die dem Sensor bekannt ist, realisiert. Interessanterweise wird hiermit auch die kontinuierliche, auf den Sensor wirkende, Gravitation der Erde gemessen. So misst ein im Ruhezustand befindender Sensor immer eine Beschleunigung von etwa $9.81\frac{m}{s^2}$.
Da die Beschleunigung auf drei unabhängigen Achsen gemessen wird, kann hieraus die Richtung, in die beschleunigt wird, ermittelt werden.
In unserem Fall können wir diese beiden Eigenschaften nutzen, um Information zwischen dem Koordinatensystem des Sensors und dem Inertialsystem zu ermitteln. Mehr zum Inertialsystem in Kapitel 4.1.
Ein großer Vorteil dieser Daten ist, dass wir eine einzige Messung zu einem beliebigen Zeitpunkt nehmen können und daraus die aktuelle Orientierung, im Bezug auf das Inertialsystem, bestimmen können.

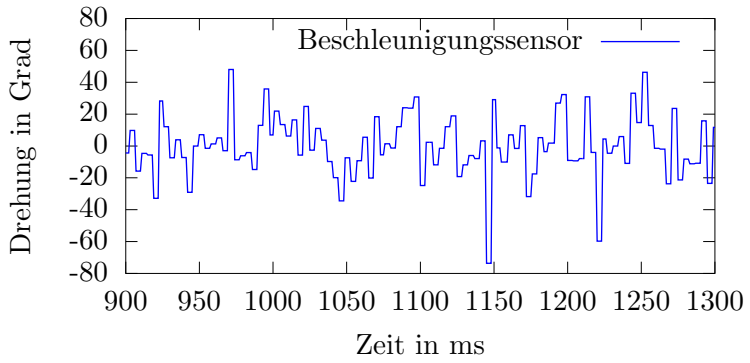


Abbildung 4: Es wird die Rotation um die y-Achse, während der Roboter läuft, gezeigt. Es ist starkes Rauschen zu sehen. Der Wert schwankt zwischen 50 und -70 Grad.

Ein Nachteil ist, dass diese Daten, wie in Abbildung 4 zu sehen, stark rauschen. Es muss eine Mittelung oder Glättung über mehrere Datensätzen geben, damit die Daten verwertbar werden. Eine einfache Mittelung führt zwar zu verbesserten Daten, aber diese haben eine starke zeitliche Verzögerung. Es sind viele Datensätze notwendig und es ist immer noch Rauschen vorhanden. Ein weiteres Problem ist, dass diese Daten keine Aussage über die Drehung um den Richtungsvektor selbst machen können. Für diese Infor-

mation müssen weitere Sensoren einbezogen werden.
Obwohl der Sensor Beschleunigungen misst, sind Beschleunigungen, die durch die Translationsbeschleunigung (Vorwärtsbewegung und Seitwärtsbewegung des Roboters) entstehen so gering, dass der Fehler praktisch nicht messbar ist. Dies kann daher für unsere weitere Betrachtung vernachlässigt werden.

3.3.2 Gyroskop

Ein Gyroskop, häufig auch Drehratensensor genannt, ist ein Sensor mit dem die Rotationsgeschwindigkeit des Sensor gemessen werden kann.
Die Messungen werden an drei unabhängigen Achsen getätigt. Die gelieferten Daten sind in der Einheit Radien pro Sekunde (r/s), welche erst durch eine Integration über die Zeit relative Rotations-Daten ergeben. In Abbildung 5

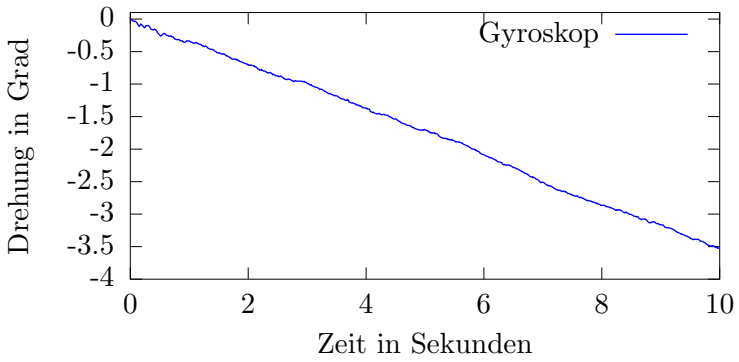


Abbildung 5: Es wird die relative Rotation um die y-Achse (Daten sind über die Zeit integriert), während der Roboter steht, gezeigt. Es ist ein Drift aber kaum Rauschen zu sehen.

ist ein Gyroskop zu sehen, das nicht bewegt wird. Trotzdem werden kleine Drehgeschwindigkeiten gemessen. Da diese über die Zeit integriert werden, erhält man immer eine Drehung, die verschieden von 0 ist. Dieser Effekt wird Drift genannt. Ohne Drift wäre eine konstante Linie zu sehen.
Ein weiterer Nachteile ist, dass das Gyroskop nur eine relative Drehung im Raum beschreiben kann. Es gibt keinen Zusammenhang zwischen dem Gyroskop und dem Inertialsystem. Auch wenn zu einem beliebigen Zeitpunkt eine Beziehung zum Inertialsystem besteht, geht diese Information aufgrund des Drifts mit der Zeit verloren. Vorteilhaft ist, dass die Daten sehr wenig Rauschen aufweisen und sich daher über einen kurzen Zeitrum präzise und vergleichweise verzögerungsfreie Daten generieren lassen.

3.3.3 Magnetometer

Das Magnetometer misst Magnetfelder und die Richtung in der sie wirken. Meist wird mit dem Magnetometer das Erdmagnetfeld entlang, drei un-

abhängiger Achsen, gemessen. Diese drei Werte können als Richtungsvektor aufgefasst werden, welcher genutzt werden kann, um festzustellen wo Norden und Süden sind. Diese Information lässt sich zum Messen der Drehung um die z-Achse nutzen, die bei der Erhebung der Beschleunigungssensor-Daten und Gyroskop-Daten fehlt.
Ähnlich wie bei dem Beschleunigungssensor benötigt man nur einen Messpunkt, um eine absolute Drehung festzustellen.
Leider befindet sich der Magnetometer im Torso des Roboters. Dies ist eine Anforderung des Regelwerkes für die Humanoid League[vgl. 15, S. 7]. Aufgrund der Motoren des Roboters wird das gemessene Erdmagnetfeld so stark von den Magnetfeldern der Motoren überlagert, dass diese Daten nicht eingesetzt werden konnten.

3.3.4 Temperatursensor

Die Genauigkeiten von Beschleunigungssensor, Gyroskop und Magnetometer sind Temperaturabhängig. Sie neigen dazu, die Skala, auf der sie arbeiten, in Abhängigkeit der Temperatur zu verschieben und zu verzerren. Die Idee des Temperatursensors ist, diese Effekte, durch Berücksichtigung der aktuellen Temperatur, zu kompensieren.
Während diese Arbeit geschrieben wurde, gab es keine funktionierende Schnittstelle zum Temperatursensor. Daher kommt der Sensor in dieser Arbeit nicht zum Einsatz.

4 Grundlagen

4.1 Koordinatensystem

Um die Lage und Orientierung eines Objektes in einem Raum wiederzugeben, ist ein Koordinatensystem als Referenz notwendig. Unser Referenzsystem ist das Inertialsystem, welches die Erdoberfläche darstellt. Die Oberfläche wird durch die x-und y-Achse aufgespannt. Die z-Achse ist orthogonal zu dieser Fläche, wobei die positive Richtung nach unten zeigt. Es handelt sich um ein rechtshändiges System. Entsprechend sind auch die Drehrichtung, mit Blick auf die Achse, mit dem Uhrzeigesniss definiert.

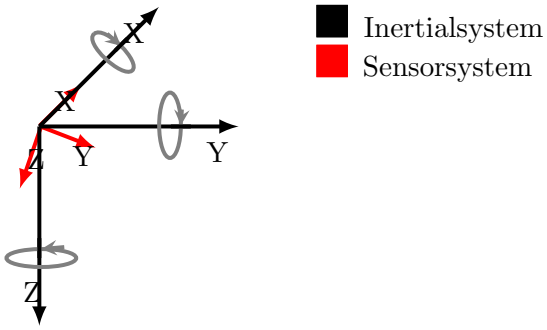


Abbildung 6: Das Sensorkoordinatensystem ist um die x-Achse gedreht.

Ein weiteres Koordinatensystem ist das, in dem sich der Sensor befindet. Der Sensor befindet sich immer im Ursprung des Koordinatensystem. Dieses System wird auch als sensorfestes Koordinatensystem bezeichnet. Es rotiert sich mit der Drehung des Sensors. Auch hier handelt es sich um ein rechtshändiges System. Die Drehrichtungen sind wie beim Inertialsystem definiert.
Die Orientierung beschreibt wie das sensorfeste Koordinatensystem zum Inertialsystem gedreht ist.

In dieser Arbeit wird keine Verschiebung des Koordinatensystem angewendet. Das führt dazu, dass die beiden Koordinatensysteme immer den gleichen Ursprung haben.

4.2 Verwendung von Quaternionen

Um die Orientierung eines Objektes wiederzugeben, gibt es verschiedene mathematische Konstrukte, die unterschiedliche Vor- und Nachteile haben. Mögliche Lösungen sind Eulerwinkel, Rotationsmatrizen oder Quaternionen. In dieser Arbeit werden dafür Quaternionen benutzt, deren Vorteile gegenüber Rotationsmatrizen und Eulerwinkeln in Kapitel 4.2.3 und 4.2.4 erklärt werden.

Der Zahlbereich der Quaternionen wird mit $\mathbb{H}$ bezeichnet. Quaternionen erweitern den Zahlbereich der reellen Zahlen mit drei weiteren Komponenten, die auch als Imaginärteil bezeichnet werden. Die Zahlen des Imaginärteil werden mit i, j und k gekennzeichnet. Die erste Komponente wird als Realteil bezeichnet.

Ein Quaternion $q \in \mathbb{H}$ lässt sich mit $q = q_0 + q_1 i + q_2 j + q_3 k$ beschreiben, wobei $q_{0-3} \in \mathbb{R}$
In dieser Arbeit wird ein Quaternion auch wie folgt bezeichnet

$$q = (q_0, q_1, q_2, q_3) \tag{1}$$
$$= (q_0, \vec{q}) \tag{2}$$

wobei

$$\vec{q} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} \tag{3}$$

Die Darstellung des Imaginärteils als Vektor ergibt später einige Vereinfachungen bei den Operationen.
Ist ein Skalar $a \in \mathbb{R}$ gegeben, so kann dieser auch als Quaternion $q = (a, 0, 0, 0)$ aufgefasst werden. Dieses Quaternion besteht nur aus dem Realteil.
Analog dazu wird ein Vektor $\vec{x} \in \mathbb{R}^3$ als ein Quaternion mit nur Imaginärteil mit $q = (0, x_1, x_2, x_3)$ aufgefasst.

4.2.1 Operationen

An dieser Stelle wird nur auf die Rechenoperationen eingegangen, die für diese Arbeit entscheidend sind.

4.2.1.1 Addition Die Addition zweier Quaternionen $p, q \in \mathbb{H}$ wird durch die Formel

$$p + q = +(p_0, p_1, p_2, p_3) + (q_0, q_1, q_2, q_3) \tag{4}$$
$$= (p_0 + q_0, p_1 + q_1, p_2 + q_2, p_3 + q_3) \tag{5}$$

definiert. [vgl. 3, S. 1]

4.2.1.2 Multiplikation Die Multiplikation zweier Quaternionen $p, q \in \mathbb{H}$ entspricht der Formel

$$\begin{aligned} p \cdot q =& (p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3) && (6)\\ &+(p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2)i && (7)\\ &+(p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1)j && (8)\\ &+(p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0)k && (9) \end{aligned}$$

Es wird jede Komponente paarweise multipliziert. [vgl. 3, S. 1] Deutlich einfacher lässt sich die Multiplikation ausdrücken, wenn der Imaginärteil der Quaternionen als Vektoren aufgefasst wird und das Kreuzprodukt($\otimes$) aus der Vektorrechnung zur Hilfe genommen wird.

$$\begin{aligned} p \cdot q =& (p_0, \vec{p}) \cdot (q_1, \vec{q}) && (10)\\ =& (p_0q_0 - \vec{p} \odot \vec{q}, p_0\vec{q} + q_0\vec{p} + \vec{p} \otimes \vec{q}) && (11) \end{aligned}$$

[vgl. 6, S. 35]

4.2.1.3 Konjugation Die Konjugation eines Quaternion q wird als $\bar{q}$ gekennzeichnet. Ist

$$q = q_0 + q_1i + q_2j + q_3k \tag{12}$$

gegeben, so ist $\bar{q}$ wie folgt definiert

$$\bar{q} := q_0 - q_1i - q_2j - q_3k \tag{13}$$

[vgl. 3, S. 1]. Da es sich nur um ein Vorzeichenwechsel der Imaginärkomponenten handelt ist die Konjugierung sehr effizient.

4.2.1.4 Norm und Betrag Die Norm von $q \in \mathbb{H}$ ist immer reel und positiv.

$$Norm(q) := q \cdot \bar{q} = q_0^2 + q_1^2 + q_2^2 + q_3^2 \tag{14}$$

Der Betrag von $q \in \mathbb{H}$ entspricht der euklidischen Länge des Vektors (q_0, q_1, q_2, q_3). Dies entspricht der Wurzel aus der Multiplikation von q und $\bar{q}$ welche die Norm des Quaternion ist.

$$|q| := \sqrt{Norm(q)} = \sqrt{q\bar{q}} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{15}$$

[vgl. 3, S. 1]. Aufgrund des Ziehens der Wurzel ist diese Operation sehr rechenaufwändig und sollte sparsam eingesetzt werden.

4.2.1.5 Inverse Es gilt für das Inverse q^{-1} von $q \in \mathbb{H}$, dass $q \cdot q^{-1} = 1$ sein muss. Wird q^{-1} wie folgt konstruiert, so erhält man immer eine gültige Inverse.

$$q^{-1} = \bar{q} \cdot \frac{1}{q\bar{q}} \tag{16}$$

[vgl. 6, S. 34]. Interessant hieran ist das $q\bar{q}$ im Fall einer Einheitsquaternion den Wert 1 hat. Es findet also eine Teilung durch 1 statt die wegfällt. In diesem Fall ist das Inverse identisch mit der Konjugation

$$|q| = 1 \tag{17}$$

$$\Rightarrow q^{-1} = \bar{q} \tag{18}$$

Die Berechnung der Inverse ist also sehr effizient.

4.2.1.6 Konvertierung zur Rotationsmatrix Da ein Quaternion eine Drehung im dreidimensionalem Raum darstellt, lässt sich ein Einheitsquaternion auch in eine Rotationsmatrix konvertieren. Ist keine Einheitsquaternion muss diese durch eine Normalisierung umgewandelt werden. Die Konvertierung zur Rotationsmatrix ist dann mit folgender Formel möglich:

$$q = (q_0, q_1, q_2, q_3), |q| = 1 \tag{19}$$

$$D_q = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & -2q_0q_3 + 2q_1q_2 & 2q_0q_2 + q_1q_3 \\ 2q_0q_3 + 2q_1q_2 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & -2q_0q_1 + 2q_2q_3 \\ -2q_0q_2 + 2q_1q_3 & 2q_0q_1 + 2q_2q_3 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{20}$$

4.2.2 Verwendung zur Orientierung

Quaternion lassen sich auch zur Orientierungsbeschreibung einsetzten.

4.2.2.1 Rotation eines Punktes Soll ein Punkt $\vec{x} \in \mathbb{R}^3$ rotiert werden, wobei die Rotation durch das Quaternion $q \in \mathbb{H}$ beschrieben wird, kann man entweder das Quaternion zu einer Rotationsmatrix konvertieren und mit dieser multiplizieren oder durch ein äquivalente Multiplikation die Rotation direkt mit der Quaternion bestimmen.

$$\vec{x}' = q \cdot \vec{x} \cdot q^{-1} \tag{21}$$

[vgl. 3, S. 3]

4.2.2.2 Drehung um eine Achse Ein Quaternion, das keine Rotation abbilden soll, lautet $q_0 = (1, 0, 0, 0)$ oder $q_1 = (-1, 0, 0, 0)$. Ist eine Rotationsachse $\vec{r} \in \mathbb{R}^3$ mit $|\vec{r}| = 1$ gegeben, so wie einem Winkel α, so kann man hieraus direkt ein Quaternion ableiten.

$$q = (\cos\frac{\alpha}{2}, \vec{r} \cdot \sin\frac{\alpha}{2}) \tag{22}$$

Es entsteht hierbei ein Einheitsquaternion. [vgl. 6, S. 38]

4.2.2.3 Mehrere Drehungen Angenommen es sind zwei Drehungen q_0 und q_1 gegeben. Soll ein Punkt zuerst um q_1 und dann um q_0 rotiert werden, so entspricht dies q_{01} wenn $q_{01} = q_0 \cdot q_1$.
Interessant an dieser Stelle ist, die Möglichkeit die Orientierung eines Objektes sowohl im Sensorfestenkoordinatensystem als auch im Inertialsystem anzugeben. Dazu muss die Drehung um eine Achse wie Abschnitt 4.2.2.2 beschrieben werden. Angenommen die aktuelle Drehung wird mit q_0 bezeichnet und q_1 ist die Drehung, die angewendet werden soll. So entspricht $q_0 \cdot q_1$ der Anwendung der Drehung $q1$ im Sensorfestenkoordinatensystem, während $q_1 \cdot q_0$ der Anwendung der Drehung $q1$ im Inertialsystem.

4.2.3 Vergleich zu Rotationsmatrizen

Rotationsmatrizen bestehen aus 3x3 Elementen aus $\mathbb{R}$. Mit diesen lassen sich, wie mit Quaternionen, Drehungen im dreidimensionalem Raum darstellen.
Im Gegensatz zu Quaternionen benötigt eine 3x3 Matrix aber 9 Elemente und somit deutlich mehr Speicherplatz, was in speicherarmen Systemen entscheidend sein kann. Eine Multiplikation von Quaternionen benötigt 16 Multiplikation und 12 Additionen, während es bei Matrizen 27 Multiplikationen und 18 Additionen sind.[16, S. 5]. Die Multiplikation von Quaternion ist somit weniger rechenintensive als die von Matrizen.
Rotationsmatrizen stellen Drehung an orthogonalen Achsen dar. Werden viele Rotationsmatrizen multipliziert gibt es aufgrund von numerischen Ungenauigkeiten akkumulierte Fehler, die die Orthogonalität aufheben. Dies muss durch regelmäßige Normalisierung der Matrizen der Zwischenergebnisse entgegen gewirkt werden. Die Normalisierung kosten extra Rechenleistung und verkompliziert die fehlerfreie Verwendung von Rotationsmatrizen.[16, S. 6]

4.2.4 Vergleich zu Eulerwinkeln

Eine weitere Möglichkeit Drehungen in einem dreidimensionalem Raum abzubilden, ist mit der Hilfe von Eulerwinkelen. Dazu werden drei Winkelangaben benutzt, die jeweils eine Drehung um eine Achse beschreiben. Grundsätzlich lässt sich jede Drehung mit Eulerwinkeln beschreiben. Es ist nicht immer möglich aus zwei beliebigen Drehungen im Raum A und B eine Überführung aus A nach B zu beschreiben. Dies ist der Fall, wenn sich die Drehungen A oder B in einem Gimballock befinden.
Um mit Eulerwinkeln arbeiten zu können, müssen diese regelmäßig in Rotationsmatrizen umgerechnet werden, somit kommen noch Probleme aus dem Bereich der Matrizen dazu.

4.3 Jacobi-Matrix

Eine Jacobi-Matrix ist eine Matrix, die alle partiellen Ableitungen einer Funktion enthält [10, S. 129].

Angenommen es ist eine Funktionen $f := \begin{pmatrix} f^{(1)} \\ f^{(2)} \\ \vdots \\ f^{(m)} \end{pmatrix}$ gegeben. Dann ist die Jaobi-Matrix J_f wie folgt definiert:

$$J_f(a) := \begin{pmatrix} \frac{\partial f^{(1)}}{\partial x_1}(a) & \frac{\partial f^{(1)}}{\partial x_2}(a) & \dots & \frac{\partial f^{(1)}}{\partial x_n}(a) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f^{(m)}}{\partial x_1}(a) & \frac{\partial f^{(m)}}{\partial x_2}(a) & \dots & \frac{\partial f^{(m)}}{\partial x_n}(a) \end{pmatrix} \tag{23}$$

$$\tag{24}$$

Die Matrix $J_f(a)$ ist eine lineare Approximation der Funktion f im Punkt a. Die lineare Funktion kann potenziell schneller berechnet werden. Außerdem ist es in manche Algorithmen möglich, die Funktion auf nicht lineare Funktionen zu erweitert, in dem eine Jacobi-Matrix verwendet wird.

4.4 Kalman-Filter

Der Kalman-Filter wurde erstmals im Jahr 1960 in einem Artikel veröffentlicht [9]. Um die Formeln aus dem ursprunglichen Artikel zu verstehen, sind weitere Arbeiten [11], [20], [14], [8] sowie [17], für die folgende Beschreibung des Kalman-Filters und des erweiterten Kalman-Filters, berücksichtigt wurden. Ein Kalman-Filter wird genutzt, um Rauschen, das bei der Erhebung von Daten anfällt, aus den Daten zu reduzieren. Hierzu ist es notwendig, dass das System, das gemessen wird, sowie seine Störquellen, bekannt sind und mathematisch abgebildet werden können.

Es handelt sich beim Kalman-Filter um einen Bayes'schen Minimun-Varianz-Schätzer, der auf lineare Systeme angewendet werden kann. Ein Minimum-Varianz-Schätzer versucht anhand der gemessenen Daten, die Parameter des zugrundeliegende Modell so zu wählen, dass das Model die Daten möglichst gut erklärt. Die Besonderheit des Kalman-Filters ist seine mathematische Struktur. Es ist bei dieser nicht notwendig, eine explizite Historie der Daten zu speichern und ist: daher sehr speicherschonend und nicht rechenintensiv.

In einem Kalman-Filter finden im Wesentlichen zwei Schritte statt. Zum einem wird im Predict-Step eine Vorhersage der Zustandsänderung gemacht. Hierfür muss bekannt sein in welchem Zustand das System ist, welches durch ein geeignetes Model abgebildet wird. Anhand dieses Modells wird dann eine

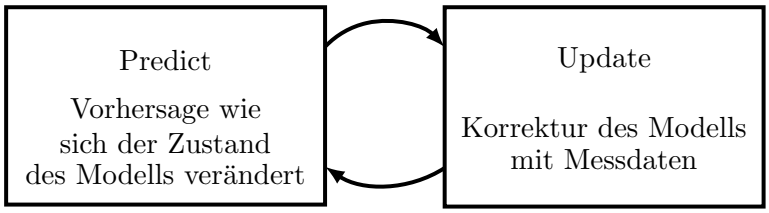


Abbildung 7: Update- und Korrekturschritt wechseln sich ab

Vorhersage über einen bestimmten Zeitraum gemacht. Im zweiten Schritt, dem Update-Step, wird das Modell mit Messdaten abgeglichen und korrigiert. Hierfür muss bekannt sein, wie der Zusammenhang zwischen Modell und Messdaten ist. Außerdem muss in beiden Schritten bekannt sein, wie groß die zu erwartende Abweichung aufgrund von Rauschen in der Vorhersage, bzw. in den Messdaten ist.
Es ist nicht notwendig, dass sich die Schritte abwechseln. Es können mehrere Predict-Steps oder mehrere Update-Steps hintereinander ausgeführt werden.

Zustandsmodell Das Zustandsmodell wird im Kalman-Filter durch zwei Größen modelliert. Zum einen durch den Zustand unseres System zu einem Zeitpunkt t, den wir mit $\vec{x}_t$ bezeichnen. In diesem sind Werte wie z.B. die aktuelle abgeschätzte Position, Geschwindigkeit oder Orientierung abgebildet. Dazu gibt es eine Fehler-Kovarianzmatrix, die den Variablennamen P_t trägt. P_t gibt Aufschluss über die Varianz der Werte aus $\vec{x}_t$. Aus dieser Matrix lässt sich ablesen, wie groß das Rauschen, welche die Güte des der Werte in $\vec{x}_t$ beschreibt. Werden viele Werte gemessen, die nicht zu den vorhergesagten Werten in $\vec{x}_t$ passen, werden die Werte in der Matrix P_t größer. Sind die Vorhersagen gut, werden die Werte in P_t kleiner.

Predict-Step Um eine Vorhersage zu machen, wird folgende Formel benutzt:

$$\vec{x}_t = F_t\vec{x}_{t-1} + B_t\vec{u}_t \tag{25}$$

$$P_t = F_tP_{t-1}tF_t^T + Q_t \tag{26}$$

Die Formel (25) soll aus einem alten Zustand $\vec{x}_{t-1}$ einen neuen Zustand $\vec{x}_t$ vorhersagen. Die Matrix F_t ist eine lineare Funktion, welche die Zustandsänderung, in Abhängigkeit der Zeit, von $\vec{x}_{t-1}$ beschreibt. Die Funktion F_t darf in jedem Vorhersage-Schritt eine andere Funktion sein bzw. mit anderen Werten besetzt sein.
Bei $B_t\vec{u}_t$ handelt es sich um einen Term, der es ermöglicht Einflüsse, die von außerhalb des eigenen Systems einwirken, mitzubeachten. Für diese Arbeit ist dies aber ohne Bedeutung.

Die Formel (26) beschreibt, wie die Kovarianzmatrix P_{t-1} vorhergesagt wird. Dies ist ähnlichd wie die Vorhersage vom Zustand $\vec{x}_t$. Darüber hinaus wird Q_t addiert. Die Matrix Q_t repräsentiert die Ungenauigkeit, die Aufgrund der Vorhersage zum Modell hinzukommt. Zwar kann Q_t für jede Vorhersage andere Werte besitzen, jedoch wird häufig ein konstantes Q verwendet. Dieses wird mit einer Zeitspanne Δt, das die Zeitspanne seit dem letzten Perdict-Step angibt, multipliziert, weil dadurch das Rauschen bei großen Intervallen zwischen Predicts vergrößert wird.

$$Q_t = Q * \Delta t \tag{27}$$

Update-Step Für den Korrekturschritt werden folgende Formeln benutzt:

$$\vec{y} := \vec{z}_t - H_t \cdot \vec{x}_{t-1} \tag{28}$$
$$K := P_{t-1} H_t^T (H_t P_{t-1} H_t^T + R_t)^{-1} \tag{29}$$
$$\vec{x}_t = \vec{x}_{t-1} + K \cdot \vec{y} \tag{30}$$
$$P_t = (I - K H_t) P_{t-1} \tag{31}$$

Der Vektor $\vec{z}_t$ repräsentiert die gemessenen Daten. Wir wollen unser Modell so anpassen, dass es die gemessenen Daten besser beschreibt.
Dafür benutzten wir eine lineare Funktion H_t. Diese berechnet Messdaten, die wir mit unserem aktuellen Modell erwarten.
Die Variable $\vec{y}$ repräsentiert die Differenz zwischen gemessenen Daten und den Daten die wir erwartet haben. Je größer dieser Wert, desto ungenauer war unser Modell.

Um diesen Fehler zu korrigieren, berechnen wir eine Innovations-Matrix K (29). Diese Matrix bewertet wie stark den gemessen Daten, beim aktuellen Zustandsmodell, vertraut werden kann. Die Innovationsmatrix ist von verschiedenen Faktoren abhängig: Auf der einen Seite ist wichtig wie genau bisherige Vorhersagen waren. Dies wird durch die Matrix P_{t-1} geregelt. Auf der anderen Seite ist das aktuelle Rauschverhalten (R_t) des Sensors wichtig. Je geringer das Rauschen des Sensors ist, desto mehr können wir den Messdaten vertrauen, was wiederum zu größeren Werte in der Innovations-Matrix führt.
Das Rauschen des Sensors wird durch R_t repräsentiert. Ist bekannt, dass sich das Rauschverhalten der gemessenen Daten nicht verändert, kann ein konstantes R verwendet werden. (30) berechnet den neuen Zustand des Models mit Berücksichtigung der berechneten Innovation. Im letzten Schritt muss auch die neue Kovarianzmatrix in Abhängigkeit der Innovation berechnet werden.

4.5 Erweiterter Kalman-Filter

Das Problem des Kalman-Filters gegenüber des erweiterten Kalman-Filters ist, dass bei der Vorhersage und Korrektur die Funktionen F_t und H_t benutzt werden. Aufgrund ihrer repräsentation als Matrix, können sie nur lineare Funktionen abbilden.
Um nicht lineare Funktionen f_t und h_t einzusetzen, ist es notwendig, den Kalman-Filter zu erweitern.
Angepasster Predict-Step:

$$\vec{x}_t = f_t(\vec{x}_{t-1}, \vec{u}_t) \tag{32}$$
$$P_t = F_t P_{t-1} t F_t^T + Q_t \tag{33}$$

Angepasster Update-Step:

$$\vec{y} := \vec{z}_t - h_t(\vec{x}_{t-1}) \tag{34}$$
$$K := P_{t-1} H_t^T (H_t P_{t-1} H_t^T + R_t)^{-1} \tag{35}$$
$$\vec{x}_t = \vec{x}_{t-1} + K \cdot \vec{y} \tag{36}$$
$$P_t = (I - K H_t) P_{t-1} \tag{37}$$

In den Formeln bleiben weiterhin F_t und H_t bestehen. Es können nicht alle F_t und H_t mit den Funktionen f_t und h_t ersetzt werden. Anstelle dessen linearisieren wir diese Funktionen im Punkt x_{t-1}. Wir benutzen die lineare Eigenschaft der Funktion in einem einzelnen Punkt und nehmen an, dass diese ausreicht, um eine Vorhersage oder Korrektur zu machen.

Linearisieren können wir, indem F_t und H_t über die Jacobi-Matrizen der Funktionen f_t und h_t definiert wird.

$$F_t(\vec{x}, \vec{u}) = \begin{pmatrix} \frac{\partial f_t^{(1)}(\vec{x},\vec{u})}{\partial x_1} & \cdots & \frac{\partial f_t^{(1)}(\vec{x},\vec{u})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_t^{(m)}(\vec{x},\vec{u})}{\partial x_1} & \cdots & \frac{\partial f_t^{(m)}(\vec{x},\vec{u})}{\partial x_n} \end{pmatrix} \tag{38}$$

$$H_t(\vec{x}) = \begin{pmatrix} \frac{\partial h_t^{(1)}(\vec{x})}{\partial x_1} & \cdots & \frac{\partial h_t^{(1)}(\vec{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_t^{(m)}(\vec{x})}{\partial x_1} & \cdots & \frac{\partial h_t^{(m)}(\vec{x})}{\partial x_n} \end{pmatrix} \tag{39}$$

Mit Hilfe dieser Matrizen können wir nun auch einen Kalman-Filter auf Modelle anwenden, die nicht linear sind. Die einzige Einschränkung ist, dass der erweiterte Kalman-Filter nicht optimal ist.

5 Modellierung

5.1 Modellierung des Kalman-Filters

Um den erweiterten Kalman-Filter erfolgreich einsetzen zu können, gibt es zwei wesentliche Funktionen, die modelliert werden müssen. Zum einen im Predict-Step die Funktion $f_t(x)$, für diese müssen wir auch die Jacobi-Matrix F_t bestimmen. Zum anderen müssen wir die Funktion $h(x)$ bzw. die Jacobi-Matrix H_t festlegen. Darüber hinaus müssen die Kovarianzmatrizen Q_t und R_t festgelegt werden.

5.1.1 Vorhandene Daten

Zuerst wird eine Bestandsaufnahme der Daten, die zur Verfügung stehen, gemacht. In regelmäßigen Abständen werden die Werte des Gyroskops gemessen. Diese Daten werden als $\vec{g}_t = (g_0, g_1, g_2)^T$ bezeichnet. Wobei die Variablen g_{0-2} in Radianten pro Sekunde(r/s) angegeben sind. Wir gehen aufgrund von Kapitel 3.3.2 davon aus, dass diese Daten weitgehend rauschfrei, aber einem Drift ausgesetzt sind.

Desweiteren haben wir die Daten vom Beschleunigungssensor $\vec{a}_t = (a_0, a_1, a_2)^T$. Die Einheit von a_{0-2} sind Meter pro Quadratsekunde (m/s^2). Der Wert von $\vec{a}_t$ werden wir als Richtungsvektor interpretieren. Da die Gravitation der Erde entlang der z-Achse, des Inertialsystems wirkt, gibt uns das Informationen über die Orientierung des Sensors zum Inertialsystems. Wie wir aus Kaptitel 3.3.1 wissen, sind diese Daten zwar driftfrei dafür starkem Rauschen unterlegen.

Mit diesen Daten haben wir genug Informationen, um Daten zu generieren, die den Drift um die x- und y-Achse kompensieren. Das Modell, welches im folgendem Abschnitt vorgestellt wird, wird daher den Drift um die z-Achse nicht kompensieren können. Für diese Kompensation wären weitere Daten notwendig.

5.1.2 Zustandsmodell

Den Zustand $\vec{x}$ zum Zeitpunkt t bezeichnen wir mit $\vec{x}_t$. In unserem Zustandsmodell möchten wir unsere aktuelle Orientierung modellieren. Wir wählen daher $\vec{x}$ als vierdimensionalen Vektor, wobei jedes Element ein Element aus einem Quaternion repräsentiert. Unseren Zustand können wir wie folgt ausdrücken:

$$\vec{x}_t = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{40}$$

Es handelt sich bei $\vec{x}_t$ um einen Vektor nicht um ein Quaternion. Um $\vec{x}_t$ als Quaternion zu intepretieren, wird die Funktion q verwendet:

$$q(\vec{x}) := (x_0, (x_1, x_2, x_3)^T) = (x_0, x_1, x_2, x_3) \tag{41}$$

5.1.3 Predict-Step

Für den den Predict-Step müssen wir die Funktion $f_t(x)$ festlegen. Diese Funktion beschreibt die Veränderung des Zustands seit dem letzten Predict-Step. Gegeben haben wir den Wert Δt, der uns die Zeitstpanne seit dem letztem Predict angibt. Außerdem bekommen wir die gemessenen Winkelgeschwindigkeiten $\vec{g}_t$. Um aus unserer Geschwindigkeit die zurückgelegten Winkel in Radien zu errechnen, müssen wir diese mit der vergangenen Zeit Δt multiplizieren.

$$\vec{z} := \begin{pmatrix} z_0 \\ z_1 \\ z_2 \end{pmatrix} = \vec{g}_t * \Delta t \tag{42}$$

Der Vektor $\vec{z}$ repräsentiert den zurückgelegten Winkel seit der letzten Messung. Wir müssen nun unseren aktuellen Zustand um diese Rotation weiter drehen. Dazu verwenden wir die Formel aus Abschnitt 4.2.2.2 und wenden die Drehung entsprechend Abschnitt 4.2.2.3 an.

$$f_t(\vec{x}_t) = q(\vec{x}_t) \cdot \overbrace{\left(\cos\left(\frac{z_0}{2}\right), \sin\left(\frac{z_0}{2}\right) \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right)}^{\text{Rotation um x-Achse des Sensors}} \tag{43}$$

$$\cdot \left(\cos\left(\frac{z_1}{2}\right), \sin\left(\frac{z_1}{2}\right) \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right) \tag{44}$$

$$\cdot \left(\cos\left(\frac{z_2}{2}\right), \sin\left(\frac{z_2}{2}\right) \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right) \tag{45}$$

Nicht ganz korrekt ist, dass wir nacheinander um die drei Achsen rotieren. Korrekterweise müssten die Rotationen gleichzeitig erfolgen. Dies wird an dieser Stelle vernachlässigt, da aufgrund der kleinen Zeitspanne zwischen zwei Vorhersage es zu nur sehr kleinen Fehler kommt. Der Fehler kann durch kleinere Zeitspannen weiter minimiert werden.
Als letzter Schritt muss noch die Jacobi-Matrix F_t, wie in Abschnitt 4.3 beschrieben, von f_t bestimmt werden.

5.1.4 Update-Step

In dem Update-Step aktualisieren wir unseren Zustand mit der gemessenen Orientierung des Beschleunigungssensors. Dazu müssen wir die Funktion h_t bestimmen. Diese Funktion sagt zu unserem aktuellen Zustandsmodell, was wir voraussichtlich messen müssten.

Der Beschleunigungssensor misst in unserem Fall die Erdbeschleunigung. Es entsteht dadurch eine Einschränkung der gemessenen Orientierung. Wir können keine Aussage über die Rotation entlang der z-Achse im Inertialsystem machen. Dies bedeutet, dass wir keine Korrektur dieser Rotationsachse machen können. Nur mit Beschleunigungssensor und Gyroskop gibt es auch keine Möglichkeit diesen Fehler zu korrigieren. Unser Modell kann daher nur entlang der y- und x- Achsen den Drift kompensieren.

Die Funktion h_t muss eine Vorhersage über die Messdaten vom Beschleunigungssensor machen. Ohne Rotierung des Sensorsystems würden wir $(0, 0, 0)^T$ messen. Nachdem sich die Orientierung unseres Sensors verändert hat, müssen wir unseren erwarteten Messwert ins Inertialsystem umrechnen. Da $\vec{x}_t$ die Drehung aus dem Inertialsystem ins Sensorsystem darstellt, müssen wir es mit der Inversen des Quaternion $q(\vec{x}_t)^{-1}$ rotieren. Die Funktion q sagt, dass der Vektor als Quaternion zu interpretieren ist (siehe Formel (41)).

$$h_t(\vec{x}_t) = q(\vec{x}_t)^{-1} \cdot \left(0, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}\right) \cdot q(\vec{x}_t) \tag{46}$$

Als letzter Schritt muss noch die Jacobi-Matrix H_t, wie in Abschnitt 4.3 beschrieben, von h_t bestimmt werden.

5.1.5 Rausch-Kovarianzmatrizen

Es gibt im Kalman-Filter die beiden Kovarianzmatrizen Q_t und R_t, die das Rauschverhalten beschreiben. Der Parameter Q_t beschreibt das Rauschen, dass bei der Vorhersage des Systems entsteht. Dies liegt daran, dass die Beschreibung des Systems durch die Funktion f_t nicht alle Einflüsse abbilden kann. Zum einen hat das Gyroskop einen Drift, der die Genauigkeit einschränkt. Zum anderen wird die Rotation nicht exakt gemessen. So geht sie von seriellen Drehungen um die drei Achsen des Sensors aus, wobei es sich eigentlich um eine einzige Drehung um alle drei Achsen handelt. Diese Dinge werden unter anderem von Q_t berücksichtigt. Die Kovarianzmatrix R_t beschreibt das Rauschen der Daten, die vom Sensor stammen. Häufig handelt es sich bei R_t um einen konstanten Wert, der oft ohne Index nur als R bezeichnet wird, da dieser Konstant über die Zeit sind. Ist Q_t oder R_t

zu klein gewählt, wird ein Teil des Rauschens nicht heraus gefiltert. Werden auf der anderen Seite Q_t oder R_t zu groß gewählt, werden Veränderungen im System erst mit einer Zeitverzögerung modelliert. Daher ist es wichtig, eine gute Wahl von Q_t und R_t zu treffen. Die Wahl von Kovarianzmatrizen ist außerdem abhängig vom Einsatzzweck. Der Einsatzzweck entscheidet wie viel Rauschen toleriert werden kann und wie groß die Verzögerungszeit sein darf. Diese Parameter werden im Abschnitt 7 ausführlich erprobt.

6 Implementierung

6.1 Vorbereitungen

6.1.1 Verwendete Mathebibliothek

Um den Kalman-Filter möglichst allgemein zu halten und Modifikationen an den Berechnungen möglichst leicht zu ermöglichen, sollen die Formeln nicht in primitiven Datentypen von C implementiert werden, sondern über abstrakte Strukturen, die Quaternionen, Matrizen und Vektoren abbilden, implementiert werden. Da keine passende Bibliothek gefunden werden konnte, ist im Rahmen dieser Arbeit eine eigene entwickelt worden.

Es ist eine Matrixstruktur implementiert. Diese bietet folgende Operationen: Addition zweier Matrizen, Bestimmung der adjunkten Matrix, Bestimmung der Determinante, Vergleich zweier Matrizen, Berechnung der Inverse, LR-Zerlegung einer Matrix, Multiplikation zweier Matrizen, normal- und gaußverteilte Matrix Initialisierung, Subtraktion zweier Matrizen und Transposition von Matrizen.

Auf der Matrixstruktur aufbauend ist die Vektorstruktur implementiert. Ein Vektor ist in dieser Bibliothek immer ein Spaltenvektor. Jeder Vektor kann zu jedem Zeitpunkt als Matrix mit einer Spalte intepretiert werden. Somit sind alle Matrix-Operationen auch mit einem Vektor möglich. Darüber hinaus bieten Vektoren weitere Funktionen an: Kreuzprodukt, Punktprodukt, Normalisierung, Längenbestimmung.

Quaternionen sind wiederum auf der Vektorstruktur aufgebaut. Jedes Quaternion lässt sich zu jedem Zeitpunkt als 4x1 Vektor oder nur der Imaginärteil als 3x1 Vektor ansprechen. Somit sind mit einem Quaternion alle Operationen von Vektoren und Matrizen möglich. Darüber hinaus werden noch spezielle Funktionen für Quaternionen angeboten wie: Konjugation von Einheitsquaternionen, SLERP-Interpolation zwischen zwei Quaternionen, Multiplikation von Quaternionen, Rotation eines 3d-Vektors um ein Quaternion, Skalierung eines Quaternion, Umwandlung eines Quaternion zu Eulerwinkeln oder Rotationsmatrizen.

6.1.2 Kalibrierung des Gyroskops

Der Drift des Gyros hat einen geringen Einfluss auf die Bestimmung der Drehungen um die x- und y-Achse, da dieser kontinuierlich über den Kalman-Filter mit den Daten vom Beschleunigungssensor korrigiert wird. Auf die Bestimmung der Drehung um die z-Achse hat dies aber eine große Auswirkung. An dieser Stelle können die Daten vom Beschleunigungssensor nicht benutzt werden, um den Drift zu kompensieren.

Da der Drift nahezu konstant ist, können wir diesen einmalig bestimmen und damit unsere Werte, die vom Gyroskop kommen, verrechnen, so dass das Gyroskop in Ruhe sehr geringe Drehgeschwindigkeiten liefert. Aufgrund der Temperaturschwankung und dem damit veränderten Driftverhalten des Sensors, ist es nicht möglich, den Drift komplett zu kompensieren. Dafür konnte der Drift im gegebenen Rahmen minimirt werden.

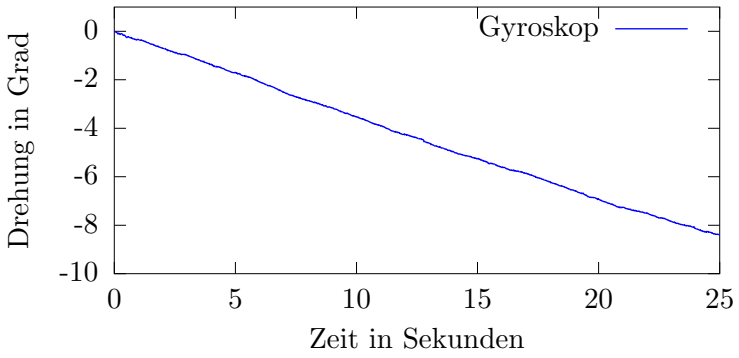


Abbildung 8: Gyroskop: Drift um die z-Achse vor der Kalibrierung

Um den Drift zu bestimmen, sammeln wir die Gyrodaten über eine Zeitspanne von zehn Sekunden. Es ist wichtig, dass das Gyroskop hierbei nicht bewegt wird. Der Durchschnittswert der gesammelten Daten wird nun von jedem weiteren Messwert abgezogen, um einen möglichst driftfreien Wert zu erhalten.

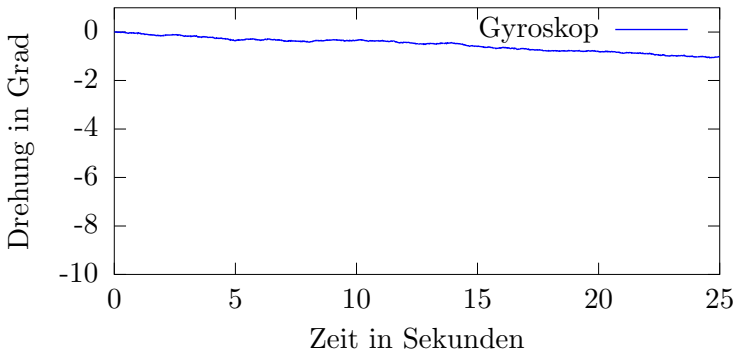


Abbildung 9: Gyroskop: Drift um die z-Achse nach der Kalibrierung

6.1.3 Kalibrierung der Bezugsorientierung

Bis jetzt haben wir die Annahme getroffen, dass die IMU bereits perfekt im Torso des Roboters befestigt ist und keine Drehung innerhalb des Roboters hat. Da dies leider nicht der Fall ist, müssen wir die Drehung zwischen Torso und IMU bestimmen und alle Werte, die wir aus dem Kalman-Filter erhalten, in die entgegengesetzte Richtung rotieren. So erhalten wir die gewünschte Orientierung, die in anderen Teilen der Software verwendet werden kann.
Eine Kalibrierung des Roboters im Stehen ist nicht möglich. Das Problem ist, dass der Torso zum Boden nicht senkrecht zum Boden ausgerichtet ist. Dies liegt daran, dass die Motoren in Füßen, Beinen, Bauch und Rücken, wegen der Winkelgenauigkeiten der Servos, nicht perfekt kalibrierbar sind. Um diese Fehler zu vermeiden, legen wir den Roboter auf den Bauch. Die Vorderseite des Torsos muss mit dem Boden ohne Lücken abschließen.

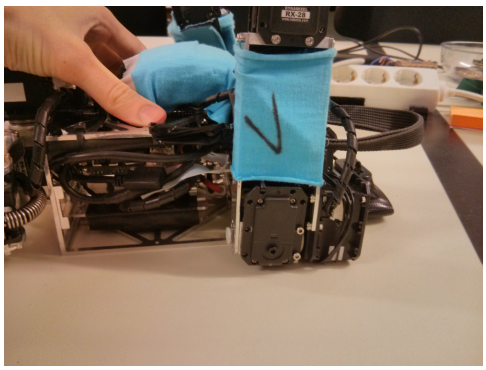


Abbildung 10: Der Roboter wird zum Kalibrieren des Offsets in eine wohl definierte Position gebracht. Die Vorderseite des Torsos muss auf einer Ebene komplett aufliegen.

Die jetzt gemessene Orientierung entspricht unserem Offset um 90° um die y-Achse gedreht. Diese multiplizieren wir mit einem Quaternion, das wie in Abschnitt 4.2.2.2 eine Drehung um -90° um die y-Achse beschreibt. Dadurch erhalten wir den tatsächlichen Offset der IMU.
Bevor die Werte aus $\vec{x}$ in andere Teile des Softwaresystems weitergeben, multiplizieren wir diese zuerst mit der Inverse unseres Offsets.

7 Evaluation

Für die Bewertung der Ergebnisse des erweiterten Kalman-Filters, brauchen wir eine oder mehrere Vergleichsgrößen. Als erstes schauen wir uns die Parametrisierung des Kalman-Filters an und welche Auswirkung diese hat. Danach vergleichen wir eine konkrete Parametrisierung mit ungefilterten Gyroskop- und Beschleunigungsdaten. Als Vergleichswert nehmen wir den Winkel um die y-Achse des Roboters.

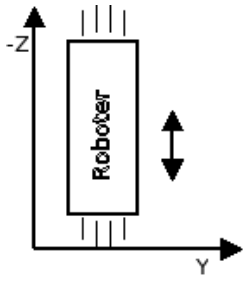
Über die Gyroskopdaten wissen wir, dass sie verzögerungs- und sehr rauscharm sind. Durch Vergleiche der Daten des Kalman-Filters lässt sich die Verzögerungszeit errechnen. Daneben können wir außerdem eine Aussage über das Rauschverhalten des Kalman-Filters gegenüber des Beschleunigungssensors treffen.

Vergleichen wir die Kalman-Filter-Daten zu den Beschleunigungsdaten, können wir einen potentiellen Drift ermitteln und messen, wie lange es dauert, diesen auszugleichen. Ähnlich wie bei den Gyroskopdaten können wir eine Aussage über das Rauschverhalten treffen.

Im Folgendem werden wir mit zwei Versuchsaufbauten experimentieren: Im ersten Aufbau steht der Roboter aufrecht und wird Freihand um ca. 60° nach vorne und wieder zurück die aufrechte Position gebeugt.
Im zweiten Szenario macht der Roboter Laufbewegungen auf der Stelle. Beim Laufen macht der Roboter viele Auf- und Abbewegungen des Oberkörpers, die zusätzliches Rauschen verursachen.
In einem dritten Versuch untersuchen wir das verbleibende Driftverhalten um die z-Achse.
Zum Abschluss werfen wir noch einen Blick auf die Magnetometerdaten und schauen uns die Probleme dieser Daten an.

7.1 Versuch 1 - Rausch- und Driftverhalten

In diesem Versuch läuft der Roboter für kurze Zeit auf der Stelle. Der gleiche Versuch wird für verschiedene Parameter des Kalman-Filters ausprobiert. Vor allem soll der Einfluss der Parameter Q und R gezeigt werden. Der Ausdruck $Q = 0.1$ steht für eine 3x3-Matrix die auf ihrer Diagonale die Werte 0.1 stehen hat und ansonsten 0 ist.
Für die Auswertung werden in den Diagrammen die Drehungen um die y-Achse gezeigt.



(a) Während des Laufen bewegt sich der Roboter auf und ab. Dies verursacht sehr ungenaue Daten vom Beschleunigungssensor.

Abbildung 11

Wir werden uns die drei Parameterpaare ($Q =$

$0.0001, R = 10$), ($Q = 0.01, R = 0.1$) und ($Q = 0.001, R = 1$) anschauen. Diese Parameter wurden experimentel bestimmt und haben sich als interessant herausgestellt.

Q=0.0001, R=10 Im ersten Fall schauen wir uns an, was passiert wenn man ein relative großes Q und kleines R wählt. Ein großes Q bedeutet, dass dem Vorhersageschritt und damit dem Gyroskop stark vertraut wird. Ein kleines R bedeutet das dem Korrekturschritt und dem damit verbundenen Beschleunigungssensor, nicht so stark vertraut wird.

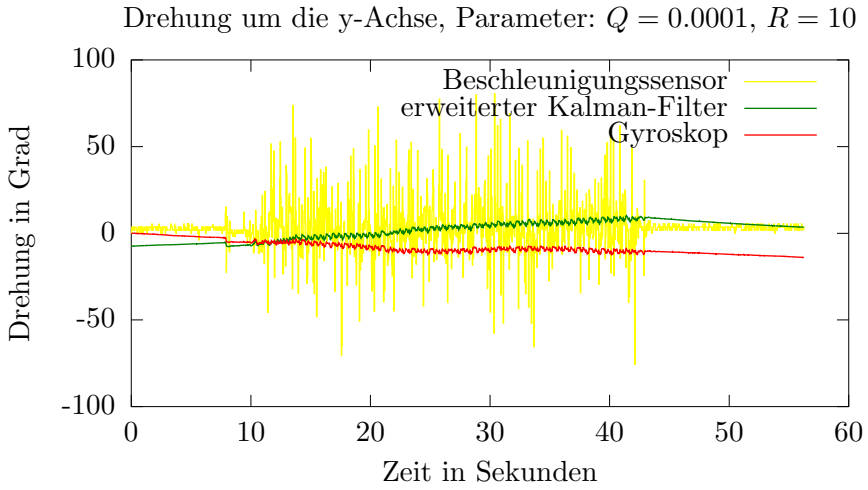


Abbildung 12: Die Abweichung der Kalman-Filter-Daten vom Beschleunigungssensor sind groß, es dauert mehrere Sekunden bis die Daten des Kalman-Filters sich diesem wieder annähern.

In Abbildung 12 sind die gemessenen Daten zu sehen. In den ersten 1000ms stand der Roboter still. Im Bereich 1000ms bis ca 4200ms ist der Roboter auf der Stelle gelaufen. Ab 4200ms stand der Roboter wieder still. Wobei die roten Werte die Gyroskopdaten, die blauen Werte die Beschleunigungsdaten und die grünen die Daten aus dem Kalman-Filter visualisieren.
In den ersten 1000ms kann man beobachten, wie groß das Rauschen der Sensoren im Ruhezustand ist. Der Drift des Gyroskops ist gut zu beobachten. Dieser Drift wird nur wenig durch die Bewegungen des Roboters im Bereich 1000ms-4200ms beeinflusst.
Das Grundrauschen des Beschleunigungssensors ist auch in Ruhe deutlich größer als des Gyroskops. Der Vorteil ist, dass der Beschleunigungsensor keinem Drift unterliegt und sehr konstant um den Wert 0° bleibt. Sobald der Roboter sich in Bewegung setzt, ist das Rauschen des Sensors nochmal deutlich größer und nicht geeignet, um daraus, ohne weitere Daten, eine Vorhersage der Orientierung des Roboters zu machen.

Die grüne Linie repräsentiert die Daten des Kalman-Filters. Das erste ersichtliche Problem ist die Initialisierung. Diese fällt häufig auf ungeeignete Werte. Wir sehen im Beireich 0ms-1000ms und 4200ms-5500ms, dass die Kalman-Filter-Daten sich langsam an den korrekten Wert von 0° annähernd, aber dies nicht stark genug, sodass es im Bereich 1000ms bis 4200ms es zu einem Drift kommt, der nicht korrigiert wird.

Q=0.01, R=0.1 Im zweiten Fall wollen wir die relative Größe von Q und R tauschen, also ein kleines Q wählen und großes R. Dadurch müsste der Kalman-Filter dem Beschleunigungssensor stärker folgen und weniger den Daten des Gyroskops.

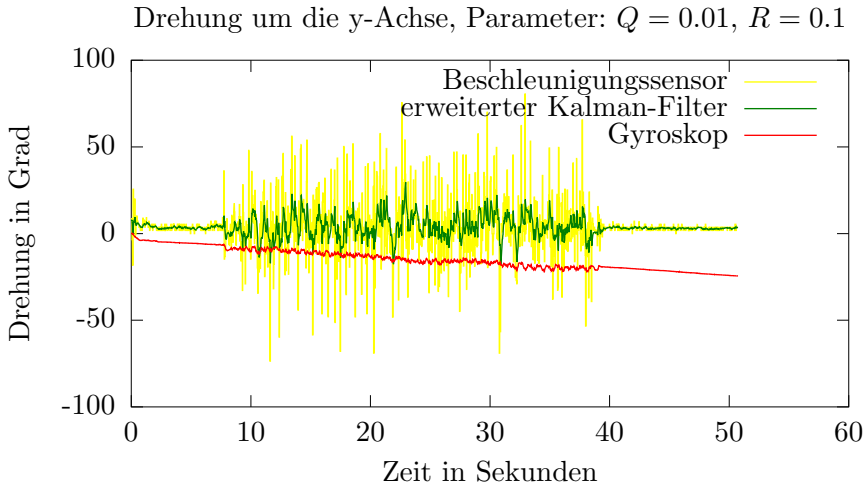


Abbildung 13: Der Kalman-Filter folgt dem Beschleunigungssensor zu stark und hat ähnlich viel Rauschen.

Wie zu erwarten, sehen wir in Abbildung 13, dass sich der Kalman-Filter deutlich besser dem Beschleunigungssensor anpasst. Das Problem bei diesem Parametersatz ist, dass die Daten vom Kalman-Filter ähnlich großes Rauschen wie der Beschleunigungssensor aufweisen. Somit haben wir nur einen geringen Vorteil des Kalman-Filters gegenüber dem Beschleunigungssensor erzielt.

Q=0.001, R=1 So kommen wir zu unserem dritten Parametersatz: Wir wählen Q=0.001 und R=1.

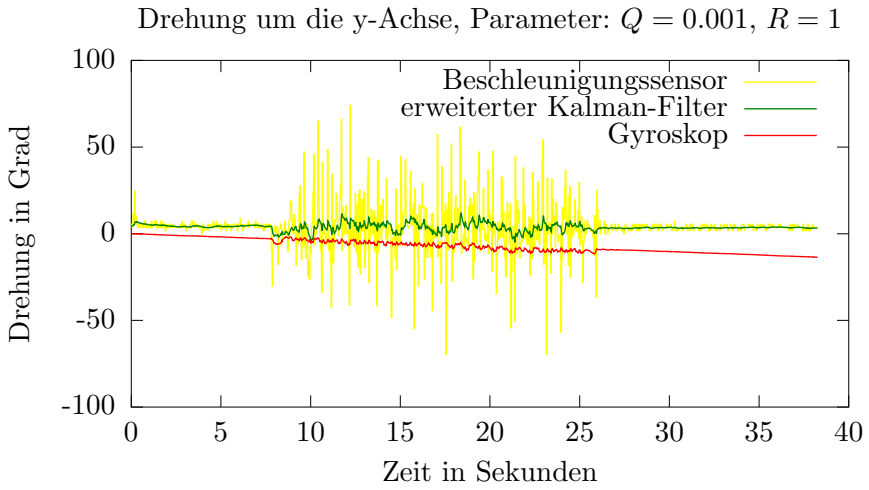


Abbildung 14: Kompromiss, sichtbares zusätzliches Rauschen, aber dafür kaum Drift.

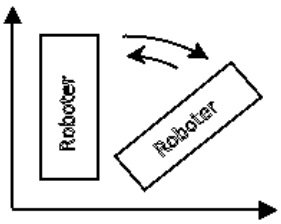
In Abbildung 14 sehen wir einen Verlauf, der deutlich mehr unserem Ziel, die Vorteile von Beschleunigungssensor und Gyroskope zu vereinen, näher kommt. Wir sehen das kurzzeitige Veränderungen der Daten des Gyroskops stark in die Ergebnisse des Filters einfließen. Trotzdem wird der Drift vom Gyroskop kompensiert. Es ist zwar immer noch eine deutliche Zunahme des Rauschverhaltens der Daten vom Kalman-Filter zu beobachten, diese sind im Vergleich zum vorherigen Parameterdatensatz deutlich kleiner.

Mit dem ersten Parametersatz haben wir Parameter gewählt, die sich zu stark den Gyroskop folgen und deshalb den Drift nicht kompensieren können. Im zweiten Fall ist der Kalman-Filter zu stark dem Beschleunigungssensor gefolgt und hat daher keinen Mehrwert geliefert. Mit dem dritten Datensatz haben wir ein Ergebnis, das zufriedenstellend ist. Es ist nicht möglich, generelle optimale Parameter zu finden. Die Auswahl der optimalen Parameter, hängt stark von der konreten Anwendung ab.

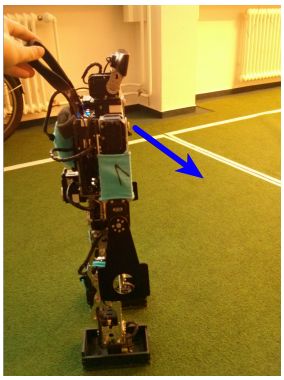
7.2 Versuch 2 - Verzögerung

Mit dem vorherigen Versuch konnten wir den Einfluss der Parameter auf das Rauschverhalten beobachten. Durch die stetige Auf- und Abbewegung des Roboters und des dabei entstehenden Rauschens, ist es schwer festzustellen, ob es eine zeitliche Verzögerung zwischen den Sensordaten und den Daten des Kalman-Filter gibt. Daher benötigen wir einen anderen Versuchsaufbau, um die Verzögerung zu untersuchen.

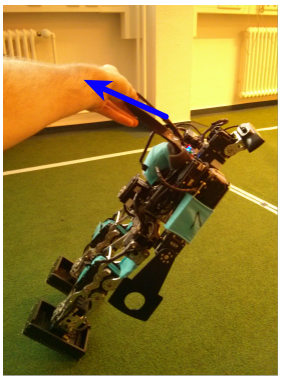
In diesem Versuch startet der Roboter in der Standposition. Er wird dann



(a) Schematischer Ablauf des Versuches. Der Roboter wird zügig auf ca. 60° gedreht und wieder zurück.



(b) Der Roboter wird, um seine y-Achse, nach vorne gedreht.



(c) Wenn etwa 60° erreicht sind, wird der Roboter wieder in die Standposition gebracht.

Abbildung 15

manuell und zügig auf 60° gekippt und dann sofort wieder zurück in die Ausgangsposition gebracht. Da nur Daten innerhalb eines Versuches verglichen werden, genügt das ungefähre Drehen des Roboters. Hier wurde der Parametersatz des vorherigen Versuches genutzt.

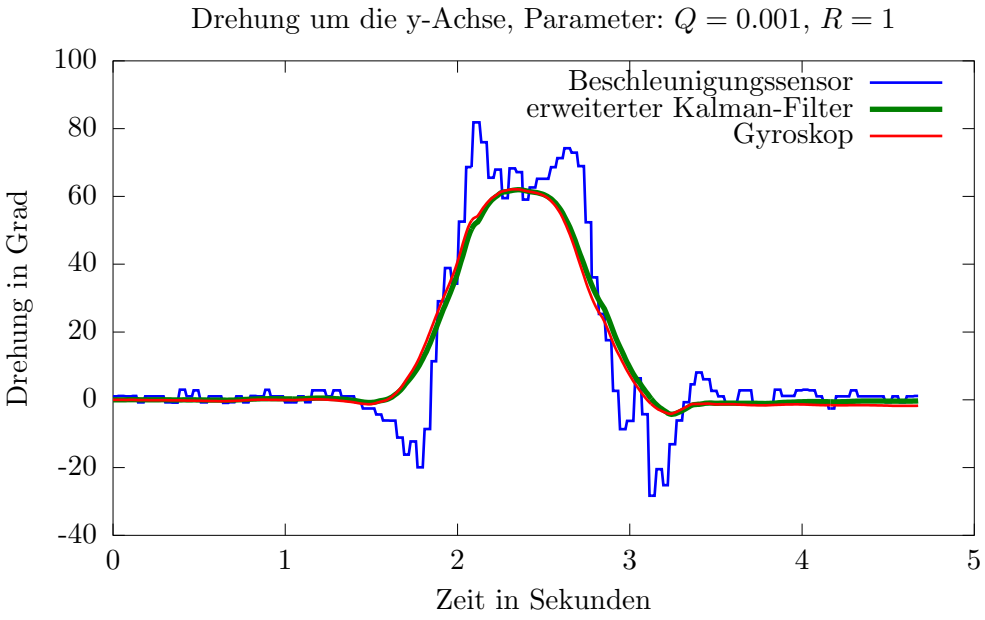


Abbildung 16: Es ist deutlich eine sehr kleine Verzögerung zwischen Gyroskop und Kalman-Filter zu erkennen.

Ein kleiner Auszug der Werte im Bereich der Spitze, die größten Werte sind markiert:

Zeitpunkt	Gyroskop	Kalman-Filter
2320 ms	62.102°	61.804°
2330 ms	62.109°	61.884°
2340 ms	**62.114°**	61.930°
2350 ms	62.107°	61.968°
2360 ms	62.048°	**61.979°**
2370 ms	61.890°	61.914°
2380 ms	61.831°	61.850°

Tabelle 1

Das Extremum des Kalman-Filters wird 20ms später erreicht als vom Gyroskop. Außerdem erreicht der Kalman-Filter im Maximum nicht den gleichen Wert. Der Grund der Verzögerung steht vermutlich im Zusammhang mit dem ungenauem Verhalten des Beschleunigungssensors. So verursacht dieser, dass der Kalman-Filter im ersten Moment nicht dem Gyroskop folgt. Es handelt sich allerdings nur um wenige Grad Abweichung und einer kurzen Verzögerungszeit. Dies dürfte für die meisten Anwendungen des Roboters ausreichend sein. Der Roboter arbeitet auf Motion-Ebene mit 100Hz. Bei 20ms Verzögerungen, wären das ca. 2 bis 3 Motion-Durchläufe zu spät. Auf der Cognition-Ebene wird mit nur 30Hz gearbeitet. Hier ist mit nur einem Cognition-Durchlauf Verspätung zu rechnen.

7.3 Versuch 3 - Drift um z-Achse

Der Kalman-Filter kann den Drift des Gyroskops über den Beschleunigungssensor kompensieren. Leider ist dies nur um der x- und y-Achse möglich. Der Beschleunigungssensor kann keine Aussage über Drehung um die z-Achse machen. Dies führt dazu, dass an dieser Stelle der Drift durch den Kalman-Filter nicht kompensierbar ist. Da dieser außerdem über die Zeit nicht konstant ist, kann der Drift durch einen konstanten Wert, wie er in Abschnitt 6.1.2 beschrieben wird, nur über kurze Intervalle minimiert werden.
In diesem Versuch verwenden wir die Driftkompensation aus Abschnit 6.1.2. Wir kalibrieren den Roboter und messen den Drift. Danch lassen wir den Roboter für 10 Minuten laufen. Anschließend messen wir den Drift erneut.

Das Ergebniss des Versuches sieht wie folgt aus: Direkt nach der Kalibrierung, aber noch vor dem Laufen haben, wurde ein Drift von $2.0255°/min$

gemessen. Nach der 10 minütigen Laufenphase ist dieser auf $18.691°/min$ angestiegen. Weitere Versuche haben ähnliche Werte ergeben.

Diese Werte sind genau genug, um für kurze Zeit eine Abschätzung der Drehung um die z-Achse zu machen. Eine kurze Zeit bedeutet wenige Minuten, dies ist aber von der Anwendung abhängig.
Um den Drift über längere Zeit zu kompensieren, müssen weitere Daten erhoben werden, die mit in das Model des Kalman-Filters einfließen.

7.4 Versuch 4 - Magnetometer

Eine weitere Idee ist es, den Drift um die z-Achse über ein Magnetometer zu kompensieren. Leider ist das Magnetometer starken Einflüssen von den Motoren des Roboters ausgesetzt.
Das Magnetometer misst entlang dreier Achsen das magnetische Feld der Erde. Diese drei Werte können als Vektor intepretiert werden, die in die Richtung des magnetischen Norden zeigen. Dies ist sehr ähnlich zum Beschleunigungssensor der Richtung Erde zeigt.

Das Erdmagnetfeld ist sehr schwach, weshalb Gegenstände aus Metall dieses leicht ablenken und abschwächen können. Sind die Gegenstände nicht beweglich, kann diese statische Veränderung kompensiert werden. Schwieriger wird es, wenn die Gegenstände ihre Magnetischeeigenschaft ändern. Dies ist der Fall bei den Elektromotoren. Das Magnetfeld wird ständig durch den Einfluss der Elektromotoren verändert und verschlechter potentiell das gemessene Erdmagnetfeld. Als Experiment, um festzustellen wie geeignet die Daten von dem Magnetometer sind, drehen wir den Roboter um 360° um seine z-Achse. Dies geschieht einmal im Stand, ohne Einfluss der Motoren und einmal während der Roboter Laufbewegungen ausführt und somit das Magnetfeld ständigen Veränderungen ausgesetzt ist.

Magnetfeld im Stand Das erste Experiment zeigt folgende Daten:

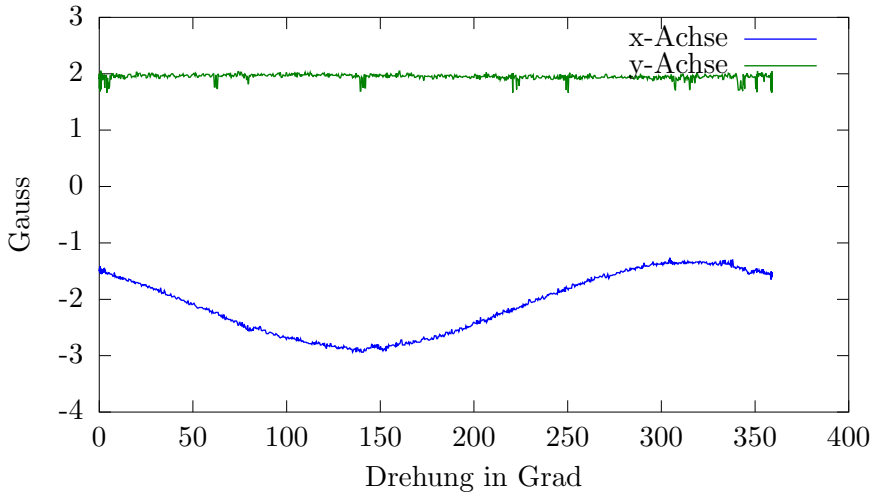


Abbildung 17: Der stehende Roboter wird um ca. 360° um die z-Achse gedreht. Zu sehen ist der Einfluss des Magnetfeldes auf das Magnetometer. Die Drehung ist entlang der x-Achse gut zu erkennen. Auf der y-Achse ist der Einfluss der Störer zu groß.

Die blaue Linie steht für die x-Achse. Es ist eine deutliche sinusähnliche Kurve zu sehen. Die grüne Linie, die für die y-Achse steht, bietet leider keine Informationen. Auf die Darstellung der z-Achse wurde verzichtet. In diese würden konstant $-8.1 Gauß$ gemessen. Dies enstpricht dem kleinsten Wert, der vom Sensor gemessen werden kann.
Aus der x-Achse lassen sich präsize Werte herauslesen. Soll zu einem gegebenen Wert des Magnetometers, die entsprechende Drehung bestimmt werden, ist dies auf wenige Grad genau möglich. Leider ist die Sinusfunktion surjektiv. Daher gibt es keine eindeutige Zuordnung. Zu einem Magnetometerwert sind meist zwei verschiedene Drehungen möglich.

Magnetfeld beim Laufen Das zweite Experiment zeigt folgende Daten:

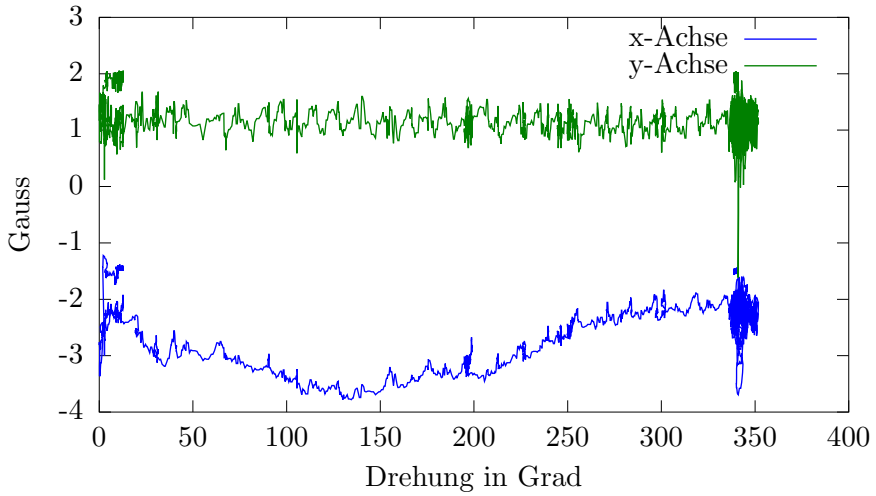


Abbildung 18: Der Roboter wird beim Laufen um ca. 360° gedreht. Zu sehen ist der Einfluss des Magnetfeldes auf das Magnetometer. Die Drehung ist entlang der x-Achse gut zu erkennen. Auf der y-Achse ist der Einfluss der Störer zu groß.

Zwar ist in Abbildung 18, entlang der x-Achse, eine sinusähnliche Kurve zu erkennen, doch ist diese deutlich verrauschter, als die aus Abbildung 17. Auf der y-Achse sind keine Informationen entnehmbar. Desweiteren ist auffällig, dass der Bereich, in dem sich die Kurven befinden, um etwa -1 *Gauß* verschoben ist. Dies macht es schwer, einen direkten Zusammenhang zwischen den Magnetometerdaten und der tatsächliche Drehung, unabhängig von der aktuellen Bewegung des Roboters, aufzuzeigen. Ein erweiterter Kalman-Filter führt hier nicht zum Ziel. Es sind weitere Informationen notwendig, die etwas über die Verschiebung der Kurven aussagen und somit erst eine Rückrechnung in Winkel zulassen.

8 Zusammenfassung und Ausblick

In dieser Arbeit wird gezeigt, wie man die Orientierung eines humanoiden Roboters in Echtzeit abschätzen kann. Dazu wurden die Grundlagen für einen erweiterten Kalman-Filter vorgestellt. Es wurde gezeigt, wie man die Orientierung über Quaternionen formulieren kann. Danach wurde ein Modell für den Kalman-Filter entwickelt. Der Kalman-Filter arbeitet mit Daten aus dem Beschleunigungssensor und Gyroskop, mit dem Ziel die Vorteile beider Sensoren zu vereinen. Der Vorteil des Gyroskops ist sein geringes Rauschen, während der Beschleunigungssensor keinen Drift aufweist. Durch Nutzung von verschiedenen Parametersätzen wurde gezeigt, wie man beide Vorteile im Kalman-Filter vereinen kann. Außerdem wurde der Drift um die z-Achse bestimmt und evaluiert. Es wurde festgestellt, dass der Drift für kurze Zeiträume von wenigen Minuten ausreichend Genauigkeit bietet. Es wurden die Daten des Magenometers betrachtet. Diese würden sich in der Theorie hervorragend für die Driftkompensation um die z-Achse eignen. Es wurde gezeigt, dass diese starken Störungen ausgestetzt sind und somit keine einfache Verwendung möglich ist.

Der Drift um die z-Achse ist momentan nur für kurze Zeitabschnitte tolerierbar. Um diesem weiter zu entgegnen wäre, eine Integration des Temperatursensor interessant. Mit diesem könnten bekannte auftretene Drifts in Abhängigkeit der Temperatur besser kompensiert werden.
Interessant wäre, herrauszufinden, ob es Algorithmen, die aus den Magnetometer-Daten brauchbare Informationen extrahieren.
Sollte dies nicht möglich sein, könnte man versuchen, die Magnetometer-Daten durch anderen Daten zu ersetzten. So wäre es, denkbar Daten aus der Bildverarbeitung zu nutzten. So könnte über statische Objekte, die gesehen werden, ein Drift festgestellt werden. Dies dürfte vor allem erfolgsversprechend sein, wenn das Objekt weit entfernt ist.
Es wäre auch denkbar, eine automatische Kalibrierung einzubauen. Der Roboter könnte in einer beliebigen Pose für wenige Sekunden stehenbleiben. In dieser Zeit kann der Drift neu bestimmt werden. Dies erfordert aber ein explizites Verhalten des Roboters, das auf der Kognitivenebene der Software berücksichtig werden muss.

Literatur

[1] *Awards and Scores*. 2013. URL: `http://www.fumanoids.de/awards-and-scores`.

[2] ER Bachmann u. a. "Orientation tracking for humans and robots using inertial sensors". In: *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on*. IEEE. 1999, S. 187–194.

[3] Wilhelm Blaschke. *Kinematik und Quaternionen*. Dt. Verl. der Wiss., 1960.

[4] Daniel Borrmann. "Integration einer inertialen Messeinheit in die Mikrocontroller Plattform eines Autonomen Modellfahrzeugs". Bachelorarbeit. Freie Universität Berlin, 2012.

[5] *Downloads Humanoid Rules*. 2013. URL: `http://www.tzi.de/humanoid/bin/view/Website/Downloads`.

[6] Patrick Du Val. *Homographies, quaternions and rotations*. Clarendon Press, 1964.

[7] Simon J. Julier und H. F. Durrant-Whyte. "Navigation and Parameter Estimation of High Speed Road Vehicles". In: *Robotics and Automation Conference*. unknown. 1995, S. 101–105.

[8] Simon J Julier und Jeffrey K Uhlmann. "New extension of the Kalman filter to nonlinear systems". In: *AeroSense'97*. International Society for Optics und Photonics. 1997, S. 182–193.

[9] Rudolph Emil Kalman u. a. "A new approach to linear filtering and prediction problems". In: *Journal of basic Engineering* 82.1 (1960), S. 35–45.

[10] Wilfred Kaplan. *Advanced calculus*. Bd. 2. Addison-Wesly, 1973.

[11] Joseph J LaViola Jr. "A comparison of unscented and extended Kalman filtering for estimating quaternion motion". In: *American Control Conference, 2003. Proceedings of the 2003*. Bd. 3. IEEE. 2003, S. 2435–2440.

[12] Peihua Li, Tianwen Zhang und Bo Ma. "Unscented Kalman filter for visual curve tracking". In: *Image and Vision Computing* 22.2 (2004), S. 157–164.

[13] Joăo Luís Marins u. a. "An extended Kalman filter for quaternion-based orientation estimation using MARG sensors". In: *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*. Bd. 4. IEEE. 2001, S. 2003–2011.

[14] Stefan Otte. "Where am I? What's going on? – World Modelling using Multi-Hypothesis Kalman Filters for Humanoid Soccer Robots". Master Thesis. Freie Universität Berlin, 2012.

[15] *RoboCup Soccer Humanoid League Rules and Setup*. 2013. URL: `http://www.tzi.de/humanoid/pub/Website/Downloads/HumanoidLeagueRules2013-05-28.pdf`.

[16] Eugene Salamin. *Application of quaternions to computation with rotations*. Techn. Ber. Working Paper, 1979.

[17] *Sensor Fusion using the Kalman Filter*. 2013. URL: `http://campar.in.tum.de/Chair/KalmanFilter`.

[18] Rudolph Van Der Merwe und Eric A Wan. "Efficient derivative-free Kalman filters for online learning." In: *ESANN*. Citeseer. 2001, S. 205–210.

[19] Eric A Wan und Rudolph Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. IEEE. 2000, S. 153–158.

[20] Greg Welch und Gary Bishop. *An introduction to the Kalman filter*. 1995.