Innenraumfußgängerverfolgung mit Inertialsensoren und Gebäudeplänen

Masterarbeit

Andreas Stolpmann

Matrikelnummer: 2385872

5. Juni 2016

Fachbereich 3: Mathematik und Informatik

Betreuer: Felix Wenk
1. Gutachter: Prof. Dr. Udo Frese
2. Gutachter: Dr. Karsten Hölscher

Andreas Stolpmann

Innenraumfußgängerverfolgung mit Inertialsensoren und Gebäudeplänen

Indoor Pedestrian Tracking using Inertial Sensors and Floor Plans

Masterarbeit, Fachbereich 3: Mathematik und Informatik

Universität Bremen, 5. Juni 2016

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Bremen, den 5. Juni 2016

.....

(Andreas Stolpmann)

Zusammenfassung

Diese Arbeit behandelt die Entwicklung und Evaluation eines Systems zur Bestimmung und Verfolgung der Position einer Person innerhalb eines bekannten Gebäudes. Dabei wird ausschließlich auf die Daten einer am Schuh der Person angebrachten inertialen Messeinheit (IMU), sowie einen Gebäudeplan zurück gegriffen. Das heißt, dass keinerlei Infrastruktur innerhalb des Gebäudes verfügbar sein muss, um das System zu verwenden.

Die Funktionsweise des Systems basiert insbesondere auf der Verwendung so genannter *Pseudomessungen*. Diese dienen dazu Informationen über den absoluten Zustand der IMU in einen Unscented-Kalman-Filter (UKF) einfließen zu lassen, obwohl diese Informationen nicht direkt durch die verwendeten Sensoren gemessen werden können.

Um diese *Pseudomessungen* anwenden zu können enthält das System einen Algorithmus zur Detektion von Standphasen während des Laufens. Während dieser Standphasen ist bekannt, dass sich die IMU nicht bewegt. In diesem Fall kann zum Beispiel ein „*zero-velocity-update*“ durchgeführt werden. Dabei wird dem UKF eine Nullmessung für die absolute Geschwindkeit übergeben. Daraufhin wird die Schätzung des UKF an diese Messung angeglichen, wodurch eventuellen Fehlern in der Schätzung entgegen gewirkt wird.

Um die Position der Person in einem Gebäude zu bestimmen, wird ein Partikel-Filter eingesetzt dessen Partikel zu Beginn über das gesamte Gebäude verteilt sind. Nach jedem erkannten Schritt wird die Änderung des Zustands des UKF zur Position der Partikel des Partikel-Filters hinzugefügt. Verläuft die Bewegung eines Partikels dadurch durch eine Wand, ist klar, dass dieser Partikel nicht der tatsächlichen Position entsprechen kann. Er wird daraufhin aussortiert. Mit der Zeit sind nur noch Partikel vorhanden, die sich in der Nähe der tatsächlichen Position des Benutzers befinden. Die Position ist damit gefunden und kann weiter verfolgt werden.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Arbeit befasst sich mit der Entwicklung und Evaluation eines Systems zur Verfolgung der Bewegungen einer Person innerhalb eines Gebäudes. Es könnte unter anderem eingesetzt werden, um die Positionen von Rettungskräften zu bestimmen und so im Notfall schneller Hilfe leisten zu können. Des weiteren könnte das System auch dabei helfen, sich innerhalb von Flughäfen, Supermärkten oder ähnlichem zu orientieren. Daher soll das System so entwickelt werden, dass es in beliebigen Gebäuden einsetzbar ist. Die Verwendung von äußeren, fest installierten oder umgebungsabhängigen Hilfsmitteln ist somit nicht möglich. Systeme wie das Globale Positionsbestimmungssystem (GPS) können ebenfalls nicht eingesetzt werden, da diese innerhalb von Gebäuden meist keine Messungen liefern können.

Stattdessen wird auf eine inertiale Messeinheit (siehe Kapitel 2) und Gebäudepläne zurückgegriffen. Erstere kann platzsparend in oder an einem Schuh untergebracht werden und behindert somit die tragende Person nicht. Gebäudepläne können im Voraus in das System eingegeben werden und verursachen keine weiteren Kosten oder Installationsaufwand.

1.1 Ziel der Arbeit

Ziel der Arbeit ist es also ein System zu entwickeln welches nur mit Hilfe der Drehraten- und Beschleunigungsmessungen einer am Fuß angebrachten inertialen Messeinheit die Bewegungen einer Person verfolgen kann. Dazu wird ein Kalman-Filter verwendet um die eingehenden Sensordaten zu verarbeiten. Ein Kalman-Filter benötigt neben den relativen Messungen der inertialen Messeinheit weitere, absolute, Messungen. Da diese nicht zur Verfügung stehen soll zudem eine Schrittdetektion implementiert werden um darüber Annahmen über das System machen zu können und sogenannte *Pseudomessungen* zu generieren welche dann vom Kalman-Filter verwendet werden können. Um die Genauigkeit des Systems zu verbessern sollte zuletzt ein Partikel-Filter implementiert werden, welcher mit Hilfe der Ausgaben des Kalman-Filters, sowie einer Karte des Gebäudes in dem sich eine Person befindet, die genaue Position dieser Person innerhalb dieses Gebäudes bestimmt.

1.2 Aufbau der Arbeit

Im Folgenden werden zunächst einige Arbeiten beschrieben, welche den Stand der Technik im Themenbereich dieser Arbeit wiederspiegeln.

In Kapitel 2 wird ein Überblick über die verwendete inertiale Messeinheit und ihrer Sensoren gegeben. Außerdem wird auf die Probleme eingegangen, die damit einhergehen.

Kapitel 3 gibt einen Überblick über das für diese Arbeit implementierte Gesamtsystem. Die Hauptbestandteile werden in den Kapiteln 4 und 5 detailiert beschrieben.

Um das vorgestellte System testen und evaluieren zu können wurde zudem eine Grafische Benutzeroberfläche implementiert. Eine Beschreibung der Funktionen findet sich in Kapitel 6.

Eine Evaluation des vorgestellten Systems sowie ein Fazit zu diesem finden sich in Kapitel 7 und Kapitel 8.

1.3 Verwandte Arbeiten

Die Idee, die Bewegungen von Personen mit Hilfe von Inertialsensoren zu bestimmen, existiert bereits seit mehreren Jahren. Einen guten Überblick über verschiedene Techniken, welche auch andere als die in dieser Arbeit verwendeten Sensoren und Hilfsmittel gebrauchen, findet sich in [8], einem Überblickpaper von Harley aus dem Jahr 2013. Im Folgenden werden einige Veröffentlichungen betrachtet, welche verschiedene Techniken zur Lösung einiger Probleme bieten und zur Entstehung dieser Arbeit beigetragen haben. Allen gemein ist, dass sie die Inertialsensoren am Fuß der Person angebracht haben.

Foxlin beschrieb 2005 in [6] ein System, welches die Bewegungen von Personen mit Hilfe von Inertialsensoren, Magnetometer und GPS verfolgt. Besonders interessant in diesem Paper ist die Verwendung von „zero-velocity-updates (ZUPTs)“. Diese werden, bei dem verwendeten Extended-Kalman-Filter als zusätzliche Messungen verwendet, wenn bekannt ist, dass sich der Fuß auf dem Boden befindet und nicht bewegt. Dadurch kann dem während der Integration der (relativen) Messungen akkumulierten Fehler entgegen gewirkt werden, obwohl es keinen Sensor gibt, welcher eine absolute Messung der Geschwindigkeit liefern kann. Um herauszufinden wann sich der Fuß auf dem Boden befindet, werden die Werte der Accelerometer und Gyrometer beobachtet. Liegen diese lange genug unterhalb eines Schwellwerts, wird davon ausgegangen, dass sich der Fuß auf dem Boden befindet.

Widyawan und Beauregard beschrieben in [21] (2008) und [2] (2009) wie Gebäudekarten verwendet werden können, um die Genauigkeit der Bewegungsverfolgung weiter zu verbessern. Dazu verwenden sie einen Partikel-Filter, dessen Partikel mögliche Positionen darstellen. Die durch Inertialdaten errechneten Bewegungen (bzw. Schritte) werden an den Partikel-Filter übergeben, welcher dann die Partikel weiter bewegt. Kommt es dabei vor, dass ein

Partikel eine illegale Position erreicht oder sich durch eine Wand hindurch bewegt, wird die Bewertung dieses Partikels auf 0 gesetzt und der Partikel somit aussortiert. Dadurch werden nur noch Positionen in Betracht gezogen, welche durch eine Person auf sinnvollem Wege erreicht worden seien können. Insbesondere wird durch den Partikel-Filter auch Fehlern in den Messungen der Inertialsensoren entgegen gewirkt. Dies geschieht, da allen Bewegungen der Partikel eine zufällige Komponente hinzugefügt wird. Gegeben genügend Partikeln wird so sehr wahrscheinlich auch die tatsächliche Bewegung der Person durch mindestens einen Partikel abgedeckt.

Im Jahr 2006 beschrieb Beauregard in [1] wie neuronale Netze dazu verwendet werden können, um das Laufmuster einer Person zu erlernen und dadurch genauere Vorhersagen über Länge und Dauer von Schritten machen zu können. Das hilft insbesondere bei der Erkennung von Standphasen während des Laufens in denen „zero-velocity-updates“ ausgeführt werden sollen. Außerdem kann anhand der voraussichtlichen Länge der Schritte eine genauere Schätzung der neuen Position nach einem Schritt erfolgen. Ein großes Problem an dieser Lösung ist jedoch, dass sie nicht ohne weiteres von einer beliebigen Person verwendet werden kann, sondern für jeden Anwender ein neues Laufmuster gelernt werden muss.

Krach und Robertson entwickelten 2008 ein System zur Bewegungsverfolgung, bei welchem an beiden Füßen einer Person Inertialsensoren angebracht wurden ([13]). Dadurch können die Ergebnisse weiter verbessert werden, da es natürlich mehr Messdaten gibt auf denen die Schätzung aufgebaut werden kann. Außerdem wird die Schätzung nicht nur an einem Fuß festgemacht, sondern an einem Punkt genau zwischen beiden Füßen. Bewegt sich dann ein Fuß wird die Position und Rotation der Schätzung jeweils nur um die Hälfte dieser Bewegung verschoben. Dies führt dazu, dass die Varianz einer Schrittschätzung halbiert wird.

1.4 Beitrag der Arbeit

Diese Arbeit soll zeigen, dass die Findung und Verfolgung der Position von Personen in einem Gebäude auch mit sehr günstigen Hilfsmitteln bewerkstelligt werden kann und auch ohne den Einsatz eines Magnetometers auskommt. Zudem wurde untersucht welche Arten von Pseudomessungen neben den „zero-velocity-updates“ noch sinnvoll einzusetzen sind. Diese zusätzlichen Pseudomessungen führen, wie in der Evaluation gezeigt, nicht nur dazu, dass die Schätzung der Position verbessert wird sondern auch dazu, dass mit dieser Art von System auch die Verwendung von Aufzügen möglich wird.

Kapitel 2

Plattform

Dieses Kapitel beschreibt die Funktionsweise einer inertialen Messeinheit (englisch *inertial measurement unit*, *IMU*) sowie die Probleme die mit der Verwendung einer solchen einhergehen. Außerdem wird ein kurzer Überblick über die verwendete *IMU* gegeben.

2.1 Was ist eine *IMU*?

Eine *IMU* ist ein Verbund mehrerer *Accelerometer* und *Gyrometer*. Diese werden auch Inertial- oder Trägheitssensoren genannt und dienen der Bestimmung von Bewegungen.

Gyrometer messen um wieviel Grad sich der Sensor pro Sekunde um seine Achse dreht. In einer *IMU* sind drei solcher *Gyrometer* verbaut. Diese sind so angebracht, dass sie in Richtung der $x-$,$y-$ und $z-$Achsen der *IMU* zeigen (siehe Abb. 2.1) und so zusammen alle Drehungen der *IMU* messen können. Durch das Integrieren der Messungen können die Drehungen der *IMU* um beliebige Achsen nachvollzogen werden.

Accelerometer messen die Änderung der Geschwindigkeit, also die Beschleunigung, die in einer bestimmten Richtung auf den Sensor wirken. Die Beschleunigung wird in der SI-Einheit m/s^2 (Meter pro Sekunde zum Quadrat) angegeben. Sie sagt also, aus um wie viele Meter pro Sekunde sich die Geschwindigkeit eines Objekts pro Sekunde verändert. Wie bei den *Gyrometer*n enthält eine *IMU* drei dieser Sensoren. Somit kann die Beschleunigungen der *IMU* entlang der $x-$,$y-$ und $z-$Achsen gemessen werden (siehe Abb. 2.1). Durch die einzelnen Messungen kann somit der Beschleunigungsvektor der *IMU* $(\ddot{x}, \ddot{y}, \ddot{z})^T$ bestimmt werden. Dieser gibt die Beschleunigung in eine beliebige Richtung an.

Durch das einfache Integrieren des Beschleunigungsvektors kann dann der Geschwindkeitsvektor $(\dot{x}, \dot{y}, \dot{z})^T$ der *IMU* berechnet werden. Wird wiederum der Geschwindkeitsvektor integriert, erhält man die Position $(x, y, z)^T$ der *IMU*. Die Beschleunigungsvektoren müssen außerdem um die geschätzte Rotation der *IMU* gedreht werden, da die Beschleunigungen immer relativ zur *IMU* selbst gemessen werden, die Geschwindkeit und Position aber in „Weltkoordinaten“ gewünscht ist. Es ist außerdem darauf zu achten, dass die Beschleunigungsvektoren

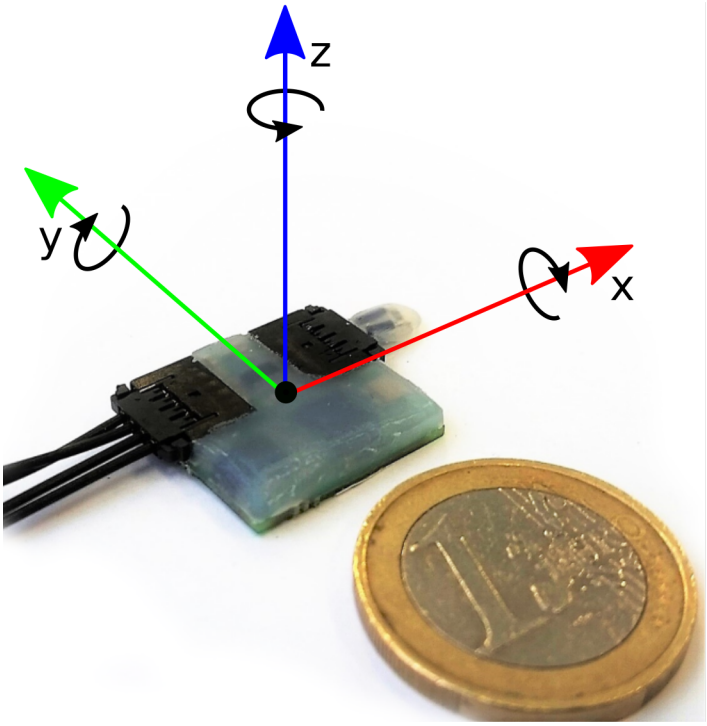


Abbildung 2.1 Die verwendete *IMU* mit ihren Achsen.

die Erdbeschleunigung von etwa $(0, 0, -9.81)^T m/s^2$ beinhalten. Diese muss also von dem, in „Weltkoordinaten“ gedrehten, Beschleunigungsvektor abgezogen werden bevor dieser integriert wird. Dies wird im Detail in Kapitel 4.4.1 beschrieben.

2.2 Probleme bei der Verwendung

Um aus den Messungen der *IMU* ihre Position zu bestimmen müssen diese Messungen, wie oben beschrieben, integriert werden. Allerdings sind, auf Grund von technischen Limitierungen sind die Messungen von Inertialsensoren immer fehlerbehaftet. Diese Fehler beeinflussen also die Berechnung der Position.

Dabei ist es so, dass selbst eine kleine Ungenauigkeit in der Messung der Drehgeschwindigkeit führt dazu, dass sich die geschätzte Rotation der *IMU* immer weiter verfälscht. Das bedeutet, dass die Beschleunigungsvektoren nicht mehr korrekt in „Weltkoordinaten“ gedreht werden können und somit die Erdbeschleunigung nicht korrekt abgezogen werden kann. Wird nun aus diesen Beschleunigungsvektoren die Position berechnet in dem sie doppelt integriert werden, wird auch der Fehler doppelt integriert. Dies führt dazu, dass der Fehler in der Position proportional zu t^3 steigt, wobei t die vergangene Zeit angibt. Dadurch können schon nach wenigen Sekunden Fehler von mehr als einem Meter enstehen (vgl. [6]).

Im Folgenden werden die wichtigsten Quellen für Fehler in den Messungen von Inertialsensoren aufgeführt (vgl. [23]).

Rauschen Die Messungen der *IMU* werden durch elektromagnetische Strahlungen gestört. Diese führen dazu, dass alle Messungen kleine, gaußverteilte Abweichungen von den tatsächlichen Werten aufweisen. Integriert man die Messungen, hebt sich das Rauschen auf, so das der Durchschnitt des Rauschens null ergibt. Allerdings steigt die Standardabweichung des Ergebnisses proportional zu $\sqrt{t}$.

Systematische Messabweichung Systematische Messabweichung bezeichnet eine konstante Abweichung der Messungen von den tatsächlichen Werten. Dies wäre zum Beispiel der durchschnittliche Messwert eines Gyrometer bei völliger Ruhe. Dieser Fehler in der Messung führt, nach einfacher Integration, zu einem Fehler in der Rotation, welcher mit der Zeit linear steigt. Ebenso kann es sein, dass ein Accelerometer immer ein wenig zu viel oder zu wenig misst.

Um dieser Art von Fehlern entgegen zu wirken, kann die Messung in Ruhelage über längere Zeit gemittelt werden. Der durchschnittliche Fehler wird dann von weiteren Messungen abgezogen, um den tatsächlichen Wert zu bestimmen. Allerdings führen die weiteren Fehlerquellen dazu, dass der Fehler nie exakt bestimmt werden kann.

Produktionsfehler Um die Messungen der einzelnen Sensoren miteinander zu vereinen, muss bekannt sein in welcher Ausrichtung sich diese zueinander befinden. Durch Produktionsfehler beziehungsweise Ungenauigkeiten kann es dazu kommen, dass die Achsen der Sensoren nicht genau senkrecht aufeinander stehen und so die Ergebnisse verfälscht werden.

Diese Fehler können gemessen und korrigiert werden. Ein Ansatz dazu findet sich in [16] von Skog et.al.. Allerdings gilt auch hier, dass andere Fehlerquellen dazu führen, dass das Ergebnis der Kalibrierung nie exakt sein kann.

Erdbeschleunigung Die Stärke der Erdbeschleunigung schwankt je nach Position auf der Erde. So liegt diese in Deutschland etwa bei $9.81 m/s^2$ und in Griechenland bei $9.80 m/s^2$. Eine solche Abweichung von nur $0.01 m/s^2$ verursacht nach nur 10 Sekunden einen Fehler von einem halben Meter.

Außerdem ist es daher selbst bei einem Sensor in absoluter Ruhe unmöglich genau zu sagen welcher Anteil an der Beschleunigungsmessung tatsächlich durch die Erdbeschleunigung verursacht wird und welcher Teil durch einen Fehler.

Temperatur Auch Temperaturunterschiede bewirken eine kleine Änderung der Messwerte. Diese entstehen zum einen durch die Umgebungstemperatur aber auch dadurch, dass sich der Sensor selbst aufheizt. Verfügt die *IMU* über einen Temperatursensor kann diesem Problem entgegen gewirkt werden. Dazu müsste allerdings zunächst ein Modell der Änderungen in den Messungen in Abhängigkeit zur Temperatur gefunden werden.

2.3 Für diese Arbeit verwendete *IMU*

Bei der für diese Arbeit verwendete IMU handelt sich es um das preisgünstige Model *BMX055*[1] der Firma *BOSCH*. Diese enthält neben den oben beschriebenen Gyrometern und Accelerometern auch ein Magnetometer welches allerdings aus den in Kapitel 2.4 genannten Gründen nicht verwendet wird.

2.4 Verwendung von Magnetometern

Magnetometer nutzen das Magnetfeld der Erde, um zu messen in welche Himmelsrichtung sie gedreht sind. Auch für das hier vorgestellte System wäre diese Information von großem Nutzen, da die tatsächliche Bewegungsrichtung natürlich sehr relevant ist, wenn man sich durch ein Gebäude bewegen möchte.

Innerhalb von Gebäuden liefert diese Art von Sensor allerdings eher unbrauchbare Ergebnisse. Dies liegt vor allem daran, dass elektrische Kabel, wie sie in jedem Gebäude verbaut sind, und andere elektrische Geräte ebenfalls elektromagnetische Strahlung aussenden, durch welche die Messungen der Sensoren stark verfälscht werden. Untersuchungen zu diesem Thema finden sich in [22], [20] und [6].

Aus diesem Grund wurde in dieser Arbeit auf die Verwendung des Magnetometers verzichtet.

[1] `https://www.bosch-sensortec.com/bst/products/all_products/bmx055` (Abruf: 29.05.2016)

Kapitel 3

Systemüberblick

In diesem Kapitel wird ein Überblick über die verschiedenen Komponenten des Systems gegeben und deren Zusammenspiel erläutert. Eine grafische Repräsentation des Gesamtsystems findet sich in Abb. 3.1 auf Seite 10.

3.1 Steuerung des Systems

Zur Steuerung des Systems durch den Benutzer und zur Anzeige verschiedener Zeichnungen wurde eine grafische Benutzeroberfläche (*GUI*) implementiert (siehe Kapitel 6). Da die Benutzeroberfläche, die Verarbeitung der Sensordaten und das Logging (siehe Kapitel 3.3) in verschiedenen Threads ausgeführt werden, wird eine Schnittstelle zwischen diesen Komponenten benötigt. Diese stellt der *SystemController* bereit.

Alle Befehle des Benutzers werden daher zunächst an den *SystemController* weitergegeben. Dieser verwaltet den gewünschten Zustand des Systems, so dass andere Teile des Systems diesen dort abfragen können. Außerdem wird über den *SystemController* das Aufnehmen und Abspielen von *Logfiles* gesteuert.

Die möglichen Systemzustände sind RESET, INIT und TRACKING. Durch den Zustand RESET wird das System in den Ausgangszustand versetzt. Dies bedeutet, dass die Rotation und Position des *Kalman-Filter* und der Partikel des *Partikel-Filter* auf den parametrisierten Startzustand zurückgesetzt werden. Im Zustand INIT wird, wie in Kapitel 4 beschrieben, die Startrotation der *IMU* berechnet. Die tatsächliche Bewegungsverfolgung geschieht im Zustand TRACKING.

Um aus dem System heraus Zeichnungen in der Benutzeroberfläche anzuzeigen werden diese ebenfalls an den *SystemController* gesendet. Diese Zeichnungen werden wiederum von der *GUI* abgefragt und angezeigt. Um diese Zeichnungen anzulegen und zu aktualisieren verfügt der *SystemController* über eine Reihe von Funktionen welche aus dem gesamten System erreichbar sind.

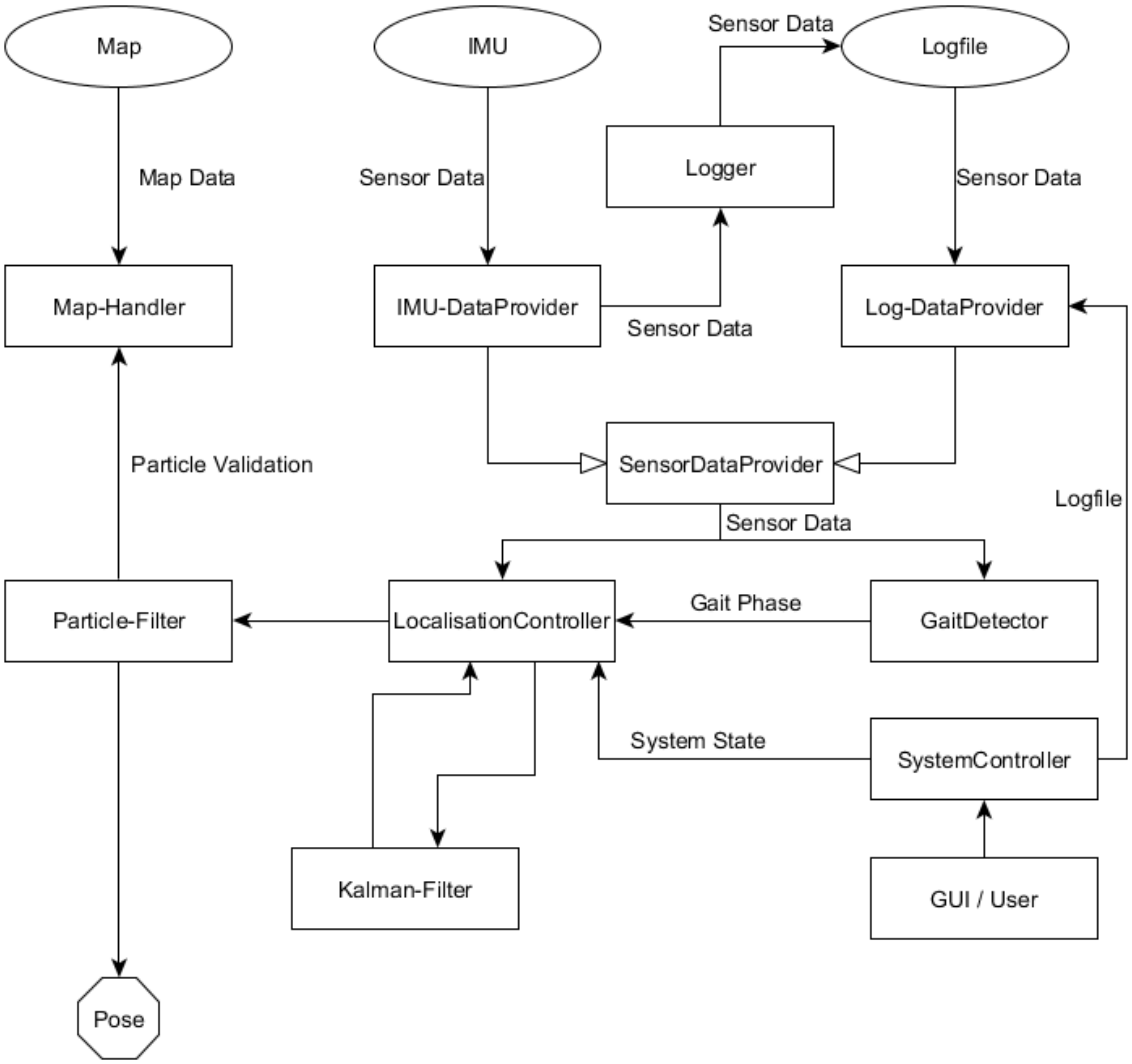


Abbildung 3.1 Systemüberblick

3.2 Verarbeitung der Sensordaten

Um die Sensordaten der *IMU* verarbeiten zu können, müssen diese zunächst durch einen *SensorDataProvider* bereit gestellt werden. Im Normalfall ruft der *IMU-DataProvider* diese Daten von direkt von der *IMU* ab. Sobald neue Sensordaten verfügbar sind (etwa alle 20 Millisekunden, vgl. Kapitel 2.3) werden diese zum einen an den *Logger* und zum anderen an den *GaitDetector* sowie den *LocalisationController* übergeben. Der *GaitDetector* dient zur Erkennung von Laufphasen (Stehen/Bewegen) und wird in Kapitel 4.5 genauer erläutert.

Der *LocalisationController* steuert die Verwendung des *Kalman-Filter*s und des *Partikel-Filter*s. Der *Kalman-Filter* wird je nachdem in welchen Zustand sich das System befindet und welche Laufphase durch den *GaitDetector* erkannt wurde mit verschiedenen Informationen versorgt und errechnet daraufhin eine erste Schätzung der Lage und Position der *IMU*. Dies wird in Kapitel 4 genauer erläutert. Der *Partikel-Filter* wird jeweils am Ende einer Standphase aufgerufen und erhält die aktuelle Schätzung des Kalman-Filters. Die Partikel des *Partikel-Filter* werden darauf hin aktualisiert und durch die Kartendaten des *Map-Handler*s validiert. Genauere Informationen dazu finden sich in Kapitel 5.

Der *Map-Handler* enthält den Gebäudeplan des Gebäudes in dem das System derzeit eingesetzt wird. Dieser beinhaltet sowohl die Positionen der Wände auf den einzelnen Stockwerken des Gebäudes, sowie die Positionen von Treppen und Aufzügen über die das Stockwerk gewechselt werden kann.

3.3 Logging

Um verschiedene Parametersätze effektiv miteinander vergleichen zu können, wurde ein Logging-Mechanismus implementiert, über welchen sämtliche Sensordaten zusammen mit ihrem Zeitstempel aufgezeichnet werden können. Dazu werden dem *Logger* wie oben beschrieben in jedem Zeitschritt die aktuellen Daten der *IMU* übergeben. Diese Daten werden in einem Ringpuffer gespeichert und von einem separaten Thread in eine Datei, das *LogFile*, geschrieben. Neben den Sensordaten wird außerdem der aktuelle Systemzustand gespeichert, so dass dieser bei der Wiedergabe der Aufnahme berücksichtigt werden kann.

Um die aufgenommenen Daten abzuspielen, muss der Benutzer zunächst die gewünschte Datei auswählen (siehe Kapitel 6.1.2). Der *SystemController* sorgt dann dafür, dass die Sensordaten nicht länger durch den *IMU-DataProvider* sondern durch den *Log-DataProvider* bereit gestellt werden. Das restliche System verwendet die wiedergegebenen Daten nun genauso, als kämen diese direkt von der *IMU*.

Kapitel 4

Navigation mit Inertialsensoren

Um die Bewegungen des Benutzers nachvollziehen zu können, müssen die Messungen der in Kapitel 2 beschriebenen *IMU* interpretiert und fusioniert werden. Dieses Kapitel beschreibt wie dies durch den Einsatz eines Unscented-Kalman-Filters (Kapitel 4.1.1) mit geeigneten Modellen (Kapitel 4.4) sowie einem Algorithmus zur Erkennung von Laufzyklen (Kapitel 4.5) gelöst wird.

Wie in Abschnitt 2.2 beschrieben, ist es nicht sinnvoll die Position des Benutzers allein durch das Integrieren die Messungen der *IMU* zu bestimmen. Da aber bekannt ist, dass die *IMU* am Fuß der Person angebracht ist, können einige Annahmen gemacht werden durch welche die Genauigkeit der Schätzung enorm steigt. Die wichtigste dieser Annahmen ist, dass sich der Fuß einer gehenden Person abwechselnd in einer Stand- und einer Bewegungsphase befindet. Wann welche Phase erreicht ist, wird mit Hilfe des in Kapitel 4.5 beschriebenen Schrittdetektors bestimmt. Durch die Aufteilung der Schrittphasen kann ausgenutzt werden, dass sich der Fuß während der Standphase auf dem Boden befinden und sich der Sensor nur minimal bewegt. Etwaigen, aufakkumulierten Fehlern kann dann, durch so genannte *Pseudomessungen*, entgegengewirkt werden.

4.1 Grundlagen

Dieser Abschnitt enthält einige Grundlagen für die Verwendung der *IMU* sowie des Kalman-Filters. Diese werden in den folgenden Abschnitten verwendet.

4.1.1 Kalman-Filter

Der Kalman-Filter dient dazu den Zustand eines Systems, trotz fehlerbehafteter Messungen, möglichst genau zu schätzen ([17]). Er wurde 1960 von Rudolf Kálmán in [12] vorgestellt und bietet eine Möglichkeit der Umsetzung eines Bayes-Filters (vgl. [5]) für lineare Systeme bei normal-verteiltem Rauschen.

Der Filter stellt die Schätzung des Zustands ($\mathcal{X}_t \in \mathbb{R}^n$) zu einem Zeitpunkt t als Normal-Verteilung $\mathcal{N}(\mu_t, \Sigma_t)$ mit Mittelwert $\mu_t \in \mathbb{R}^n$ und Kovarianz $\Sigma_t \in \mathbb{R}^{n \times n}$ dar (vgl. [19]). Der Initialzustand des Filters ist damit

$$\mu_0 = E(\mathcal{X}_0), \Sigma_0 = Cov(\mathcal{X}_0) \tag{4.1}$$

$\mathcal{X}_0$ wird dabei meist durch eine initiale Messung gewonnen, kann aber auch geraten werden.

Ein neuer Zustand wird berechnet, indem der alte Zustand mit neuen Messungen verrechnet wird. Da der Zustand niemals zurückgesetzt, sondern nur durch neue Messungen weitergetragen wird, wird dabei über alle bisherigen Zustände gewichtet „gemittelt“. Wie stark die jeweils neueste Messung in den Zustand eingeht, hängt dabei davon ab, wie groß die bekannte Ungenauigkeit in der Messung ist und wie sicher sich der Filter ist, dass seine derzeitige Schätzung richtig ist ([17]).

Die eingehenden Messungen sind unterteilt in sogenannte Übergangsmessungen $u_t \in \mathbb{R}^p$ und Zustandsmessungen $z_t \in \mathbb{R}^m$. Übergangsmessungen sind relative Messungen. Sie enthalten also nur Informationen darüber wie sich der Zustand im letzten Zeitschritt verändert hat. Zu diesen gehören zum Beispiel die Messungen einer *IMU*. Bei den Zustandsmessungen handelt es sich um absolute Messungen. Diese machen also direkte Aussagen über den Zustand oder einen Teil des Zustands. Dies wäre zum Beispiel eine GPS-Messung welche die Position des Empfängers direkt messen kann oder die erwähnten *Pseudomessungen.*

Der Kalman-Filter besteht daher aus zwei Teilen, dem *Prädiktionsschritt* welcher die Übergangsmessungen verarbeitet und dem *Korrekturschritt* welcher die Zustandsmessungen verarbeitet.

4.1.1.1 Prädiktionsschritt

Der *Prädiktionsschritt* dient hauptsächlich dazu eine Vorhersage darüber zu treffen wie sich der Zustand des Systems seit dem letzten Zeitschritt verändert hat. Dazu sind nicht in jedem Fall Übergangsmessungen nötig. So kann zum Beispiel die Position eines sich bewegenden Objekts auch ohne Übergangsmessungen geschätzt werden, wenn dessen Geschwindigkeit ebenfalls Teil des Zustands ist.

Um diese Vorhersage zu treffen wird ein Dynamikmodell ($\mathcal{G} : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$) benötigt:

$$\mathcal{G}(\mathcal{X}_{t-1}, u_t) = A_t \mathcal{X}_{t-1} + B_t u_t + g_t, A_t \in \mathbb{R}^{n \times n}, B_t \in \mathbb{R}^{n \times p}, g_t \in \mathbb{R}^n \tag{4.2}$$

Der neue Zustand sowie dessen Kovarianz wird wie folgt berechnet:

$$\mu_t = \mathcal{G}(\mu_{t-1}, u_t) \tag{4.3}$$

$$\Sigma_t = A_t \Sigma_{t-1} {A_t}^T + B_t \Sigma_{\gamma t} {B_t}^T + \Sigma_{\epsilon t} \tag{4.4}$$

$\Sigma_{\gamma t} \in \mathbb{R}^{p \times p}$ entspricht dabei der Kovarianz des Rauschens der Messung und $\Sigma_{\epsilon t} \in \mathbb{R}^{n \times n}$ die Kovarianz eines möglichen Rauschens des Zustands selbst. Letzteres wird verwendet um die Unsicherheit durch unmodellierte Effekte einzurechnen.

4.1.1.2 Korrekturschritt

Der *Korrekturschritt* dient dazu die Vorhersagen des *Prädiktionsschritt*s mit Hilde der Zustandsmessungen zu korrigieren. Diese müssen allerdings nicht zwangsläufig in jedem Zeitschritt zur Verfügung stehen. In diesem Fall würde der *Korrekturschritt* ausgelassen werden und der Zustand nur duch den *Prädiktionsschritt* weiter geführt.

Um die Zustandsmessungen in den Zustand einfließen zu lassen, wird zunächst ein Messmodell ($\mathcal{H} : \mathbb{R}^n \to \mathbb{R}^m$) benötigt. Dieses gibt an, wie die Messung $z_t \in \mathbb{R}^m$ gegeben dem aktuellen Zustand aussehen müsste.

$$\mathcal{H}(\mathcal{X}_t) = C_t \mathcal{X}_t + h_t, C_t \in \mathbb{R}^{m \times n}, h_t \in \mathbb{R}^m \tag{4.5}$$

Mit Hilfe des Messmodells lässt sich die *Innovation* w, die Abweichung der tatsächlichen von der erwarteten Messung, berechnen.

$$w = z_t - \mathcal{H}(\mu_t) \tag{4.6}$$

Im nächsten Schritt wird die Kalman-Matrix (K_t) berechnet. Diese gibt an, wie stark die *Innovation* den aktuellen Zustand beeinflussen soll.

$$K_t = \Sigma_{t-1} {C_t}^T (C_t \Sigma_{t-1} {C_t}^T + \Sigma_{\delta t})^{-1} \tag{4.7}$$

Wobei $\Sigma_{\delta t} \in \mathbb{R}^{m \times m}$ der Kovarianz des Rauschens der Messung entspricht.

Nun kann der neue Zustand berechnet werden:

$$\mu_t = \mu_{t-1} + K_t w \tag{4.8}$$

$$\Sigma_t = \Sigma_{t-1} - K_t C_t \Sigma_{t-1} \tag{4.9}$$

4.1.2 Unscented Kalman-Filter

Da der *Kalman-Filter* nur für lineare Systeme korrekte Ergebnisse liefern kann, wurde der *Unscented Kalman-Filter, UKF* ([11]) entwickelt. Die Idee dieser Erweiterung ist es, nicht nur den Mittelwert der Schätzung zu betrachten und die Modelle auf diesen anzuwenden um den neuen Zustand zu errechnen, sondern alle Modelle auch für Punkte auszuwerten die in der Nähe dieses Mittelwerts liegen. Die dadurch erhaltenen Zustände werden dann wiederum, anhand ihrer Kovarianz, gewichtet gemittelt um eine genauere Schätzung des tatsächlichen Zustands zu bekommen. Der UKF kann auch für lineare Systeme verwendet werden. Die

Ergebnisse entsprechen dann denen des oben beschriebenen „normalen“ Kalman-Filters.

Um die erwähnten Punkte zu erhalten wird die Normal-Verteilung des Zustands betrachtet. Die gewünschten Punkte liegen jeweils eine Standardabweichung (ein Sigma) vom Mittelwert entfernt und werden daher *Sigmapunkte* genannt. Für jede Dimension des Zustandraums wird jeweils ein Punkt vor und ein Punkt hinter dem Mittelwert verwendet. Daher erhält man bei einem n-Dimensionalen Zustandsraum, zusammen mit dem Mittelwert, $2n+1$ *Sigmapunkte*. ([19]).

Eine mögliche Art die Standardabweichung in jeder Dimension des Zustands zu erhalten um damit die *Sigmapunkte* zu errechnen ist laut Frese ([7]) das Anwenden der Choleskyzerlegung ($LL^T = \Sigma$) auf die Kovarianzmatrix des Zustands (Σ). Die Standardabweichung sind dann in den Spaltenvektoren $L_{|i}, i = 0 \dots n-1$ zu finden ([19]). Die *Sigmapunkte* ($\mathcal{S}$) sind damit:

$$\mathcal{S} = \begin{pmatrix} \mu & \mu + L_{|i}|_{i=0}^{n-1} & \mu - L_{|i}|_{i=0}^{n-1} \end{pmatrix} \tag{4.10}$$

Dies lässt sich laut [7] auch vereinfacht darstellen als:

$$\mathcal{S} = \begin{pmatrix} \mu & \mu + \sqrt{\Sigma} & \mu - \sqrt{\Sigma} \end{pmatrix} \tag{4.11}$$

Die für den UKF angepassten Algorithmen für den *Prädiktions-* und *Korrekturschritt* lauten damit wie folgt.

Prädiktionsschritt:

$$\mathcal{S}_{t-1} = \begin{pmatrix} \mu_{t-1} & \mu_{t-1} + \sqrt{\Sigma_{t-1}} & \mu_{t-1} - \sqrt{\Sigma_{t-1}} \end{pmatrix} \tag{4.12}$$

$$\bar{\mathcal{S}}_t^* = \mathcal{G}(u_t, \mathcal{S}_{t-1}) \tag{4.13}$$

$$\bar{\mu}_t = \frac{1}{2n+1} \sum_{i=0}^{2n} \bar{\mathcal{S}}_t^{*[i]} \tag{4.14}$$

$$\bar{\Sigma}_t = \frac{1}{2} \sum_{i=0}^{2n} (\bar{\mathcal{S}}_t^{*[i]} - \bar{\mu}_t)(\bar{\mathcal{S}}_t^{*[i]} - \bar{\mu}_t)^\top + \Sigma_{\varepsilon t} \tag{4.15}$$

Korrekturschritt:

$$\mathcal{S}_t = \begin{pmatrix} \bar{\mu}_t & \bar{\mu}_t + \sqrt{\bar{\Sigma}_t} & \bar{\mu}_t - \sqrt{\bar{\Sigma}_t} \end{pmatrix} \tag{4.16}$$

$$\mathcal{Z}_t = \mathcal{H}(\mathcal{S}_t) \tag{4.17}$$

$$\hat{z}_t = \frac{1}{2n+1} \sum_{i=0}^{2n} \mathcal{Z}_t^{[i]} \tag{4.18}$$

$$\Sigma_t^z = \frac{1}{2} \sum_{i=0}^{2n} (\mathcal{Z}_t^{[i]} - \hat{z}_t)(\mathcal{Z}_t^{[i]} - \hat{z}_t)^\top + \Sigma_{\delta t} \tag{4.19}$$

$$\Sigma_t^{x,z} = \frac{1}{2} \sum_{i=0}^{2n} (\mathcal{S}_t^{[i]} - \bar{\mu}_t)(\mathcal{Z}_t^{[i]} - \hat{z}_t)^\top \tag{4.20}$$

$$K_t = \Sigma_t^{x,z} (\Sigma_t^z)^{-1} \tag{4.21}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t) \tag{4.22}$$

$$\Sigma_t = \bar{\Sigma}_t - K_t \Sigma_t^z K_t^\top \tag{4.23}$$

4.1.3 Matrix-Exponential-Darstellung

Eine Möglichkeit Rotationen darzustellen ist die *Matrix-Exponential-Darstellung* (vgl. [7]). Diese gibt eine Rotation als Drehung um einen Vektor v mit Winkel $\|v\|$ an:

$$Rot(v) = Rot(v/\|v\|, \|v\|),\ Rot(0) = I \tag{4.24}$$

$$\tag{4.25}$$

Im dreidimensionalen Fall kann eine solche Rotation mit Hilfe der *Rodriguez-Formel* (vgl. [7]) wieder als Rotationsmatrix dargestellt werden.

$$Rot\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}, \alpha\right) = \begin{pmatrix} (1-c)x^2 + c & (1-c)xy - sz & (1-c)xz + sy \\ (1-c)xy + sz & (1-c)y^2 + c & (1-c)yz - sx \\ (1-c)xz - sy & (1-c)yz + sx & (1-c)z^2 + c \end{pmatrix}, \tag{4.26}$$

$$mit\ c = cos(\alpha), s = sin(\alpha) \tag{4.27}$$

Um eine Rotationsmatrix ($Q \in SO(3)$) als Drehung um einen Vektor darstellen zu können, wird die Umkehrfunktion *aRot* von *Rot* benötigt:

$$aRot(Q) = \frac{\alpha}{2\sin(\alpha)} \begin{pmatrix} Q_{2,1} - Q_{1,2} \\ Q_{0,2} - Q_{2,0} \\ Q_{1,0} - Q_{0,1} \end{pmatrix},\ \alpha = cos^{-1}\left(\frac{Q_{0,0} + Q_{1,1} + Q_{2,2} - 1}{2}\right) \tag{4.28}$$

$$aRot(I) = (0,0,0)^T \tag{4.29}$$

Für den Fall, dass die Rotationsmatrix keine Drehung beschreibt, also der Indentitätsmatrix (I) entspricht, ist das Ergebnis der Nullvektor.

4.1.4 Schätzen von Rotationen mit dem Kalman-Filter

Rotationen werden meistens durch Rotationsmatrizen dargestellt. Diese sind im dreidimensionalen Raum definiert als

$$SO(3) = \{Q \in \mathbb{R}^{3\times 3} | Q^T Q = I, \|Q\| = 1\}. \tag{4.30}$$

Diese haben einige Nebenbedingungen, die bei der Verwendung im Zustand eines Kalman-Filters ein Problem darstellen. Der Filter weiß nichts von diesen Bedingungen und behandelt die Elemente der Matrix unabhängig von einander. Auch die Darstellung als Quaternion leidet unter dem gleichen Problem. Repräsentation welche nur drei Werte enthalten, wie Euler-Winkel oder der skalierte Vektor (Kapitel 4.1.3) enthalten sogenannte Singularitäten. An diesen können einige Rotationen nicht mehr oder nur noch durch sehr große Änderungen

hinzugefügt werden.

Eine Lösung für dieses Problem, die auch in dieser Arbeit verwendet wird, stellen Hertzberg et.al. in [9] vor. Diese besteht im Wesentlichen darin den Zustand als so genannte $\boxplus$-*Manigfaltigkeit* ($\mathcal{S} \subset \mathbb{R}^s$) anzusehen. Diese Verfügen über zwei Operatoren, den $\boxplus$-Operator („box-plus“) sowie den $\boxminus$-Operator („box-minus“) mit

$$\boxplus : \mathcal{S} \times \mathcal{V} \to \mathcal{S}, \tag{4.31}$$

$$\boxminus : \mathcal{S} \times \mathcal{S} \to \mathcal{V}. \tag{4.32}$$

mit $V \subset \mathbb{R}^v$, wobei für alle $x \in \mathcal{S}$ gilt

$$x \boxplus 0 = x, \tag{4.33}$$

$$\forall y \in \mathcal{S} : x \boxplus (y \boxminus x) = y, \tag{4.34}$$

$$\forall \delta \in \mathcal{V} : (x \boxplus \delta) \boxminus x = \delta. \tag{4.35}$$

Durch den $\boxminus$-Operator erhält man also den Unterschied (δ) zwischen zwei Zuständen. Der $\boxplus$-Operator addiert eine Änderung auf einen Zustand.

Besteht der Zustand des Kalman-Filters nun aus einer Rotationsmatrix ist $\mathcal{S} = SO(3)$ und $\mathcal{V} = \mathbb{R}^3$. Die Operatoren könne dann definiert werden als:

$$y = x \boxplus \delta = x \cdot Rot(\delta) \tag{4.36}$$

$$\delta = y \boxminus x = aRot(x^{-1} \cdot y) \tag{4.37}$$

$$mit\ x, y \in SO(3), \delta \in \mathbb{R}^3 \tag{4.38}$$

Auf diese Art können sich die Vorteile verschiedener Darstellungen zu Nutze gemacht werden. Zum einen enthält der Zustand eine Repräsentation für die Rotation welche frei von Singularitäten ist. Zum anderen können Änderungen an der Rotation mit drei Werten ohne Nebenbedingungen dargestellt werden. Da diese Änderungen im vorliegenden Anwendungsfall jeweils nur sehr klein sind, treten hier keine Probleme mit Singularitäten auf.

Der Kalman-Filter kann nun auf diese Operatoren zurückgreifen um die Änderung des Zustands zu berechnen und diese zum Zustand hinzuzufügen. Der Algorithmus für den UKF auf Manigfaltigkeiten lautet nach [7]:

Prädiktionsschritt:

$$\mathcal{S}_{t-1} = \begin{pmatrix} \mu_{t-1} & \mu_{t-1} \boxplus \sqrt{\Sigma_{t-1}} & \mu_{t-1} \boxplus -\sqrt{\Sigma_{t-1}} \end{pmatrix} \tag{4.39}$$

$$\bar{\mathcal{S}}_t^* = \mathcal{G}(u_t, \mathcal{S}_{t-1}) \tag{4.40}$$

$$\bar{\mu}_t = \texttt{MeanOfSigmaPoints}(\mathcal{S}_t^*) \tag{4.41}$$

$$\bar{\Sigma}_t = \frac{1}{2} \sum_{i=0}^{2n} (\bar{\mathcal{S}}_t^{*[i]} \boxminus \bar{\mu}_t)(\bar{\mathcal{S}}_t^{*[i]} \boxminus \bar{\mu}_t)^\top + \Sigma_{\varepsilon t} \tag{4.42}$$

Korrekturschritt:

$$\mathcal{S}_t = \begin{pmatrix} \bar{\mu}_t & \bar{\mu}_t \boxplus \sqrt{\bar{\Sigma}_t} & \bar{\mu}_t \boxplus -\sqrt{\bar{\Sigma}_t} \end{pmatrix} \tag{4.43}$$

$$\mathcal{Z}_t = \mathcal{H}(\mathcal{S}_t) \tag{4.44}$$

$$\hat{z}_t = \frac{1}{2n+1} \sum_{i=0}^{2n} \mathcal{Z}_t^{[i]} \tag{4.45}$$

$$\Sigma_t^z = \frac{1}{2} \sum_{i=0}^{2n} (\mathcal{Z}_t^{[i]} - \hat{z}_t)(\mathcal{Z}_t^{[i]} - \hat{z}_t)^\top + \Sigma_{\delta t} \tag{4.46}$$

$$\Sigma_t^{x,z} = \frac{1}{2} \sum_{i=0}^{2n} (\mathcal{S}_t^{[i]} \boxminus \bar{\mu}_t)(\mathcal{Z}_t^{[i]} - \hat{z}_t)^\top \tag{4.47}$$

$$K_t = \Sigma_t^{x,z} (\Sigma_t^z)^{-1} \tag{4.48}$$

$$\mu_t = \bar{\mu}_t \boxplus K_t (z_t - \hat{z}_t) \tag{4.49}$$

$$\Sigma_t = \bar{\Sigma}_t - K_t \Sigma_t^z K_t^\top \tag{4.50}$$

`MeanOfSigmaPoints` :

Eingabe: Sigmapunkte $\mathcal{S}^{[i]}, i = 0, \ldots, 2n$

Ausgabe: Mittelwert der Sigmapunkte μ'

$$\mu'_0 = \mathcal{S}^{[0]} \tag{4.51}$$

$$\mu'_{k+1} = \mu'_k \boxplus \frac{1}{2n+1} \sum_{i=0}^{2n} (\mathcal{S}^{[i]} \boxminus \mu'_k) \tag{4.52}$$

$$\mu' = \lim_{k \to \infty} \mu'_k \tag{4.53}$$

In der Praxis kann die Berechnung des Mittelwerts in der Sigmapunkte in `MeanOfSigmaPoints` abgebrochen werden sobald sich das Ergebnis nicht mehr ändert oder eine maximale Anzahl von Iterationen erreicht ist.

4.2 Messungen der IMU

Die Messungen der *IMU* erfolgen immer relativ zu dessen Ursprung also im lokalen Koordinatensystem (englisch *local reference frame*, l, Abb. 4.1). Dies sind zum einen die Gyrome-

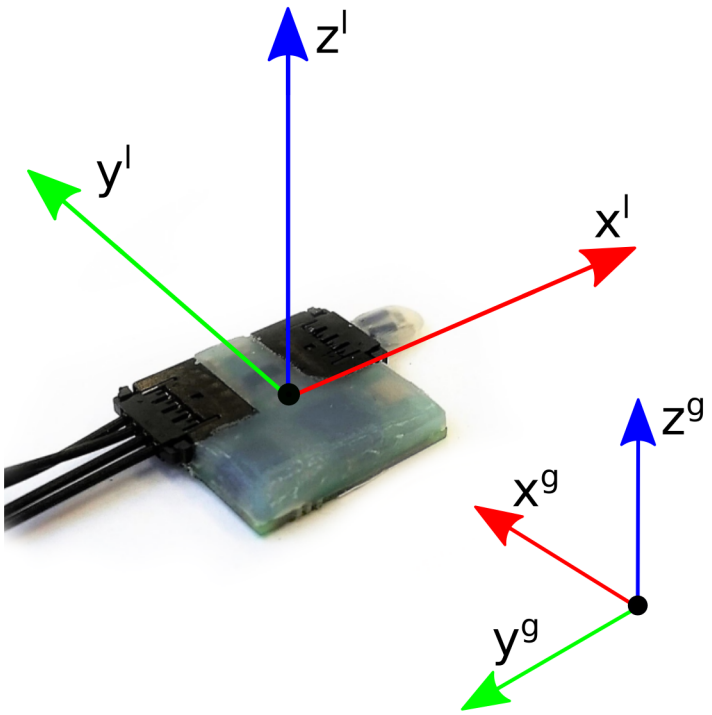


Abbildung 4.1 Visualisierung des lokalen und des globalen Koordinatensystems.

termessungen

$$\omega_t^l = (\omega_t^{l_x}, \omega_t^{l_y}, \omega_t^{l_z})^T \tag{4.54}$$

und zum anderen die Accelerometersmessungen

$$a_t^l = (a_t^{l_x}, a_t^{l_y}, a_t^{l_z})^T \tag{4.55}$$

l_x, l_y und l_z geben dabei die Messungen entlang der entsprechenden Achsen der *IMU* an. t ist der Zeitpunkt zu dem die Messung erfolgt ist.

4.3 Zustand

Der Zustand $\mathcal{X}_t = (Q_t, p_t^g, v_t^g)$ des in dieser Arbeit verwendeten Kalman-Filters setzt sich aus den folgenden Komponenten zusammen

- **Rotation:** $Q_t \in SO(3)$
- **Position:** $p_t^g = (p_t^{g_x}, p_t^{g_y}, p_t^{g_z})^T$
- **Geschwindigkeit:** $v_t^g = (v_t^{g_x}, v_t^{g_y}, v_t^{g_z})^T$

Die Werte sind jeweils relativ zur Startposition im globalen Koordinatensystem.

Q_t ist die Rotationsmatrix welche die Rotation des lokalen Koordinatensystem zum Zeitpunkt t im global Koordinatensystem angibt. So das ein Vektor v^l im lokalen Koordinatensystem durch

$$v^g = Qv^l. \tag{4.56}$$

in das globale Koordinatensystem (englisch *global reference frame*, g, Abb. 4.1) gedreht werden kann.

Initialisiert wird der Zustand mit

$$Q_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.57}$$

$$p_0^g = v_0^g = (0,0,0)^T \tag{4.58}$$

Die Kovarianzmatrix des Zustands (Σ_t) wird initialisiert mit

$$\Sigma_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.59}$$

Die Operatoren $\boxplus$ und $\boxminus$ sind für diesen Zustand definiert als:

$$\mathcal{X}_2 = \mathcal{X}_1 \boxplus \delta = \begin{pmatrix} Q_1 \cdot Rot(\delta_r) \\ p_1 + \delta_p \\ v_1 + \delta_v \end{pmatrix}, \tag{4.60}$$

$$\delta = \mathcal{X}_2 \boxminus \mathcal{X}_1 = \begin{pmatrix} aRot(Q_1^{-1} \cdot Q_2) \\ p_2 - p_1 \\ v_2 - v_1 \end{pmatrix}. \tag{4.61}$$

4.4 In dieser Arbeit verwendete Modelle

Um den Kalman-Filter optimal nutzen zu können müssen die Abläufe in der echten Welt durch Modelle beschrieben werden. Diese Modelle geben an, wie sich die erhaltenen Messungen auf die Position, Rotation und Geschwindigkeit der *IMU* in der Welt auswirken.

Je nachdem in welcher Phase des Laufens sich das System befindet, werden unterschiedliche Modelle für den Korrekturschritt verwendet. Allen gemein ist das Modell für den Prädiktionsschritt.

4.4.1 Prädiktionsschritt

Der Prädiktionsschritt führt den Zustand des Systems gegeben neuer Messungen und dem alten Zustand in jedem Zeitschritt fort. Auf diese Weise werden alle Messungen der *IMU* integriert. Dies geschieht in vier Schritten.

1. Hinzufügen der gemessenen Drehgeschwindigkeiten zur geschätzten Rotation:

$$Q_t = Q_{t-1} Rot(\omega_t^l \cdot \delta t) \tag{4.62}$$

2. Rotieren der gemessenen Beschleunigungen in das globale Koordinatensystem:

$$a_t^g = Q_t a_t^l \tag{4.63}$$

3. Berechnung der neuen, geschätzten Position:

$$p_t^g = p_{t-1}^g + v_{t-1}^g \cdot \delta t + (a_t^g + \boldsymbol{g}^g) \cdot \frac{\delta t^2}{2} \tag{4.64}$$

4. Berechnung der neuen, geschätzten Geschwindigkeit:

$$v_t^g = v_{t-1}^g + (a_t^g + \boldsymbol{g}^g) \cdot \delta t \tag{4.65}$$

Dabei entspricht $\boldsymbol{g}^g$) der Beschleunigung durch die Gravitation und δt der Zeit die seit der letzten Messung vergangen ist.

Im Prädiktionsschritt des UKFs wird, wie oben beschrieben der Kovarianz des Zustands ein geschätzter Fehler hinzugefügt. In dieser Arbeit wird dafür folgende Matrix verwendet

$$\Sigma_{\epsilon t} = \begin{bmatrix} \Sigma_Q & 0 & 0 \\ 0 & \Sigma_p & 0 \\ 0 & 0 & \Sigma_v \end{bmatrix} \delta t \tag{4.66}$$

Wobei Σ_Q die Ungenauigkeit der Rotation, Σ_p die Ungenauigkeit der Position und Σ_v die Ungenauigkeit der Geschwindigkeit angeben

$$\Sigma_Q = \begin{bmatrix} 0.0001 & 0 & 0 \\ 0 & 0.0001 & 0 \\ 0 & 0 & 0.0001 \end{bmatrix} \tag{4.67}$$

$$\Sigma_p = \Sigma_v = \begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 0.001 & 0 \\ 0 & 0 & 0.001 \end{bmatrix} \tag{4.68}$$

Der Hauptteil des Fehlers entsteht allerdings durch die Messungen. Um diesen Fehler analog zum linearen Prädiktionsschritt (Gleichungen 4.3 und 4.4) zu berücksichtigen, werden die

Sigma-Punkte für Zustand $\mathcal{X}$ und Übergangsmessungen ω und a gemeinsam generiert. Dazu wird der Zustand des UKF um diese Messungen erweitert, so dass auch dessen Kovarianz um die Ungenauigkeit der Messung selbst erweitert werden kann. Diese Ungenauigkeit wird dann vom UKF bei der Berechnung des neuen Zustands und dessen Kovarianz berücksichtigt.

Der verwendete Zustand ist:

$$\hat{\mathcal{X}}_t = (Q_t, p_t^g, v_t^g, \omega_t^l, a_t^l) \tag{4.69}$$

Die Kovarianzmatrix wird um die Varianzen der Sensoren erweitert. Die verwendeten Werte wurden dem Datenblatt der *IMU* entnommen.

$$\hat{\Sigma}_t = \begin{bmatrix} \Sigma_t & 0 & 0 \\ 0 & \Sigma_\omega & 0 \\ 0 & 0 & \Sigma_a \end{bmatrix} \tag{4.70}$$

mit

$$\Sigma_\omega = \begin{bmatrix} 0.0000238820 & 0 & 0 \\ 0 & 0.0000238820 & 0 \\ 0 & 0 & 0.0000238820 \end{bmatrix} \tag{4.71}$$

$$\Sigma_a = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix} \tag{4.72}$$

Am Ende des Prädiktionsschritts werden $\mathcal{X}_{t+1}$ und $\Sigma_t t+1$ wieder aus $\hat{\mathcal{X}}_t t+1$ bzw. $\hat{\Sigma}_t t+1$ extrahiert.

4.4.2 Korrekturschritt

Im Korrekturschritt des Kalman-Filters werden absolute Messungen des gesamten Zustands oder Teilen davon verwendet um die Vorhersagen zu korrigieren. Im vorliegenden Anwendungsfall sind allerdings keine Sensoren verfügbar, welche die Position, Geschwindigkeit oder Rotation der *IMU* direkt messen. Daher wird auf Pseudomessungen zurück gegriffen, um die Fehler der Messungen der *IMU* auszugleichen.

Pseudomessungen sind keine echten, von Sensoren aufgenommenen, Messungen. Stattdessen werden Annahmen über den Zustand des Systems gemacht. Durch diese Annahmen kann dann eine Messung generiert werden, welche durch den Kalman-Filter so behandelt wird, als sei es eine tatsächliche Messung.

Im Folgenden wird vorgestellt welche Pseudomessungen und Messmodelle in den drei Phasen des Systems verwendet werden.

4.4.2.1 Initialisierung

Um die Messungen der *IMU* sinnvoll nutzen zu können, muss zuerst der Zustand initialisiert werden. Insbesondere ist es dabei wichtig zu wissen, wo sich der Boden, relativ zur *IMU*, befindet. Andernfalls gäbe es keine Möglichkeit zwischen der Beschleunigungen durch die Gravitation und denen durch Bewegungen zu unterscheiden. Aus diesem Grund muss der Fuß währen der Initialisierung fest auf dem Boden stehen.

Kalibrierung der Erdbeschleunigung Da, wie in Kapitel 2.2 beschrieben, nicht klar ist wie hoch die Beschleunigung durch die Gravitation am Standort des Benutzers wirklich ist. Da dieser Wert aber enorm wichtig für die Genauigkeit des Systems ist wird er als erstes kalibriert. In einer perfekten Welt entspricht die Erdbeschleunigung genau $\|a\|$, also der Norm des durch die Accelerometer gemessenen Beschleunigungsvektors. Da die Messungen allerdings ein Rauschen enthalten werden diese über 100 Messungen gemittelt so dass

$$\mathbf{g} = (0, 0, -\sum_{n=0}^{99} \|a_n\|/100)^T (\approx (0, 0, -9.8)^T) \tag{4.73}$$

Dieser Wert beinhaltet auch etwaige systematische Fehler der Messungen. Das ist aber okay, beziehungsweise sogar erwünscht, denn für die weiteren Berechnungen ist nur relevant welcher Wert von den Messungen abgezogen werden muss um nur die durch die Bewegung der Person entstehenden Beschleunigungen zu erhalten. Ein Problem würde dies nur darstellen, wenn der absolute gemessene Wert zum Beispiel immer 10% zu niedrig läge und die *IMU* auf den Kopf gedreht werden würde. In diesem Fall würde der Fehler verstärkt werden. Da sich die *IMU* aber fest am Fuß des Benutzers befindet und sich ihre Rotation dort nur relativ wenig ändert kommt dieses Problem nicht zum tragen.

Berechnung der Startrotation Wenn sichergestellt ist, dass die *IMU* nicht bewegt wird, ist es, dank der Gravitation, relativ leicht herauszufinden wo sich der Boden befindet. Dieser liegt genau entgegen der Richtung der gemessenen Beschleunigung. Daher kann folgendes Modell für den ersten von 3 Updateschritten im Kalman-Filter verwendet werden

$$\mathcal{H}^1(\mathcal{X}_t) = Q_t^{-1} \cdot -\boldsymbol{g}^g \tag{4.74}$$

Die Idee des Modells ist, dass wenn die geschätzte Rotation der *IMU* mit der tatsächlichen Rotation übereinstimmt, als Beschleunigung genau $-\boldsymbol{g}^g$ gemessen werden müsste.

Bei der Wahl der Kovarianz der Messung muss darauf geachtet werden diese nicht zu klein zu wählen, da der genaue Wert von $\boldsymbol{g}^g$ nicht bekannt ist. Die verwendete Kovarianzmatrix

lautet

$$\Sigma_{H_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.75}$$

Eine falsche Schätzung der Lage der *IMU* lässt sich außerdem auch durch die im Prädiktionsschritt angepasste Position und Geschwindigkeit des Zustands erkennen. Da vorausgesetzt wird, dass sich die *IMU* nicht bewegt, sind alle Werte, die hier von Null abweichen, als Fehler anzusehen. Es können hier also Pseudomessungen verwendet werden, um die Genauigkeit der Schätzung weiter zu verbessern. Die dazu verwendeten Modelle lauten

$$\mathcal{H}^2(\mathcal{X}_t) = v_t^g \tag{4.76}$$

sowie

$$\mathcal{H}^3(\mathcal{X}_t) = p_t^g \tag{4.77}$$

In Worten ausgedrückt bedeuten die Modelle, dass genau die geschätzte Geschwindigkeit beziehungsweise Position gemessen werden müsste. Die verwendeten Messungen entsprechen dabei allerdings immer jeweils dem Nullvektor, so dass jeder andere geschätzte Wert als Fehler angesehen wird. $\mathcal{H}^2$ wird auch als *Zero-Velocity-Update* bezeichnet ([8]).

Durch diese Schritte werden nicht nur die Position und Geschwindigkeit bei Null gehalten sondern auch die Schätzung der Rotation verbessert.

Da die Pseudomessung nahezu perfekt ist, können die Kovarianzen hier sehr klein gewählt werden. Es muss dann nur sichergestellt werden, dass die *IMU* während der Initialisierung tatsächlich nicht bewegt wird.

$$\Sigma_{\mathcal{H}^2} = \Sigma_{\mathcal{H}^3} = \begin{bmatrix} 0.0001 & 0 & 0 \\ 0 & 0.0001 & 0 \\ 0 & 0 & 0.0001 \end{bmatrix} \tag{4.78}$$

Die Rotation um die z-Achse in Weltkoordinaten kann durch die Initialisierung nicht bestimmt werden. Dies ist generell nicht möglich, solange nur Gyrometer und Accelerometer verwendet werden da diese keinen Anhaltspunkt für diese Rotation liefern. Ein Magnetometers wiederum würde die benötigten Daten liefern, wird aber aus den in Kapitel 2.4 genannten Gründen nicht verwendet. Allerdings ist diese Rotation für die Berechnung der relativen Bewegung zur Startposition auch nicht nötig. Wie die absolute Richtung der Bewegung innerhalb des Gebäudes bestimmt wird, wird in Kapitel 5 erläutert.

Das Ergebnis der Initialisierung ist in Abb. 4.2 zu erkennen. Die genaue Rotation der am Fuß angebrachten *IMU* ist dem System bekannt und kann im Folgenden verwendet werden.

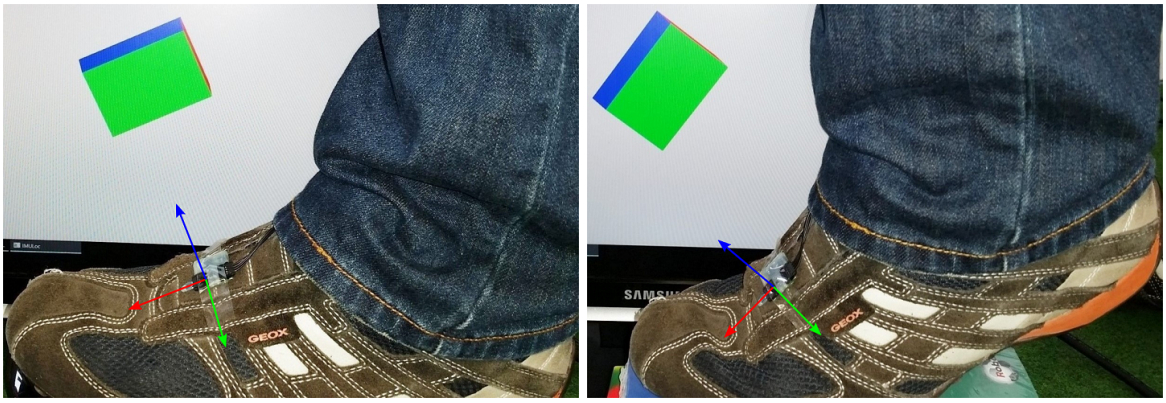


Abbildung 4.2 Ergebnis der Initialisierung. Im Vordergrund ist jeweils die am Fuß angebrachte *IMU* zu sehen. Im Hintergrund die geschätzte Rotation der *IMU* auf einem Bildschirm.

4.4.2.2 Standphase

Analog zur Initialisierung ist in der Standphase bekannt, dass sich der Fuß auf dem Boden befindet. Daher können auch hier die Modelle $\mathcal{H}^1$ (4.74) und $\mathcal{H}^2$ (4.76) verwendet werden. Es ist nur darauf zu achten die Kovarianzen hier etwas höher zu wählen da die Schrittdetektion auf Schwellwerten basiert und nicht absolut sicher ist, dass der Fuß völlig ruhig ist

$$\Sigma_{\mathcal{H}^2} = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \tag{4.79}$$

Die Position des Fußes hat sich hier natürlich geändert, so dass $\mathcal{H}^3$ (4.77) nicht mehr angewendet werden kann. Es ist allerdings bekannt, dass sich die z-Position des Fußes bei Null befindet. Es kann daher das Modell

$$\mathcal{H}^4(\mathcal{X}_t) = p_t^{g_z} \tag{4.80}$$

verwendet werden. Dadurch ist sichergestellt, dass die Position auf dem Boden bleibt. Außerdem werden etwaige Fehler zurückgerechnet und so auch hier die geschätzte Rotation sowie die Position in der x-y-Ebene weiter verbessert. Die verwendete Varianz ist $\Sigma_{H_4} = 0.1$ Es ist darauf zu achten, dass als Messung hier jeweils die Höhe der aktuellen Etage des Gebäudes relativ zum Startpunkt angegeben werden muss.

Da das hier entwickelte System Bewegungen über mehrere Ebenen eines Gebäudes verfolgen soll, müssen zwei Spezialfälle beachtet werden. Zum einen kann, wenn sich die Person innerhalb einer Treppenregion befindet, das Modell $\mathcal{H}^4$ nicht mehr verwendet werden da sich die z-Position hier natürlich ändern kann.

Für die Verwendung von Aufzügen gilt das Gleiche. Zudem muss in Aufzügen auch auf die Verwendung von $\mathcal{H}^2$ verzichtet werden da der Aufzug eine Geschwindigkeit haben kann.

Allerdings bewegt sich der Aufzug nur in z-Richtung wodurch es möglich ist ein *Zero-Velocity-Update* nur für die x-y-Ebene durchzuführen.

$$\mathcal{H}^5(\mathcal{X}_t) = (v_t^{g_x}, v_t^{g_y})^T \tag{4.81}$$

Die verwendete Pseudomessung ist dann $(0,0)^T$. Die Kovarianz entspricht der von $\mathcal{H}^2$.

Obwohl der Aufzug beschleunigt und diese Beschleunigung auch von der *IMU* gemessen wird kann $\mathcal{H}^1$ weiter verwendet werden. Dies liegt daran, dass die Beschleunigung des Aufzugs nur sehr gering ist (Etwa $0.5m/s^2$ im Fall des für die Evaluation verwendeten Aufzugs). Diese geht im Rauschen des Messmodells unter und hat somit nur einen sehr geringen Einfluss auf die Rotationsschätzung.

Aufgrund der Beschaffenheit des Kalman-Filters werden gefundene Fehler in der Geschwindigkeit auch dazu verwendet Fehler in der Position in der x-y-Ebene zu korrigieren. Anschaulich ergibt das auch Sinn, denn wenn die geschätzte Geschwindigkeit zu Beginn der Standphase nicht Null ist, heißt das, dass sie zu vorher zu hoch war und somit die Position zu viel verändert hat. Gleiches gilt für die Schätzung der Orientierung. Dies geschieht dadurch, dass der Kalman-Filter die Kovarianz zwischen Position und Geschwindigkeit mit schätzt. Ändert sich dann die Geschwindigkeit muss sich auch die Position entsprechend ändern.

Es gibt also keine Messungen, welche die Position in der x-y-Ebene und die Rotation um die z-Achse direkt beeinflussen. Dies ist für die Funktion des Gesamtsystems kein Problem, da diese Werte, wie in Kapitel 4 beschrieben, durch die Verwendung eines Partikel-Filters gewonnen werden. Allerdings führt dies dazu, dass die Kovarianzen im Zustand des Kalman-Filters an den entsprechenden Stellen immer weiter wachsen. Fließkommazahlen können in Computersystem allerdings nicht beliebig groß werden. Außerdem nimmt die Genauigkeit der Ergebnisse von Fließkommaoperationen ab wenn sehr große mit sehr kleinen Zahlen verrechnet werden. Wird das System also über einen sehr langen Zeitraum verwendet führt dies dazu, dass System irgendwann nicht mehr benutzbar ist und neu gestartet werden muss. Bei einer „normalen“ Nutzungsdauer von ein paar Stunden sollte dieses Problem nicht auftreten. Wenn das System länger verwendet werden soll müsste hier eine Lösung gefunden werden. Ein Ansatz könnte es sein die Ergebnisse des Partikel-Filters wieder in den Kalman-Filter einfließen zu lassen. Dies kann allerdings nicht ohne Weiteres als normale Messung erfolgen dies wiederum den Partikel-Filter beeinflussen würde.

4.4.2.3 Bewegungsphase

Während der Bewegungsphase können keine Annahmen über die Rotation, Position oder Geschwindigkeit gemacht werden. Dementsprechend wird in dieser Phase nur der Prädiktionsschritt des Kalman-Filters ausgeführt.

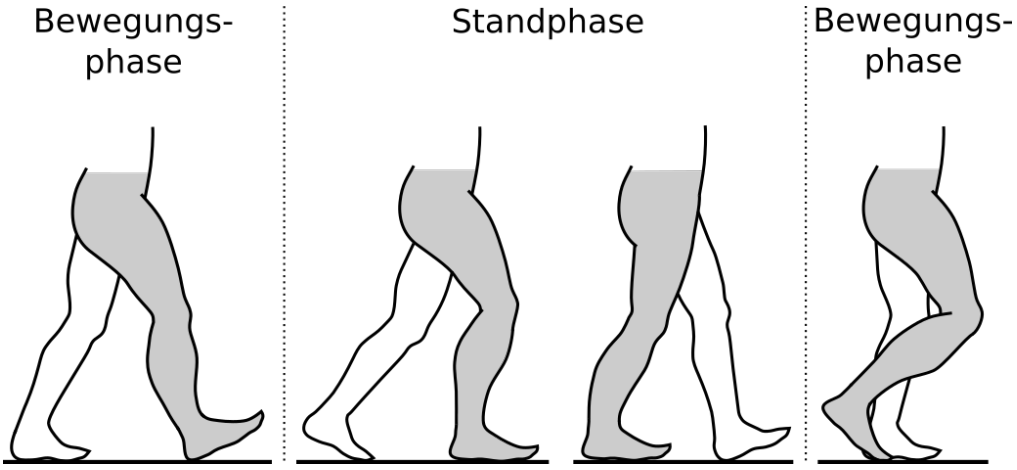


Abbildung 4.3 Wechsel zwischen Stand und Bewegungsphasen. Während der Standphase ruht der Fuß auf dem Boden während sich der andere Fuß bewegt.

4.5 Schrittdetektor

Während eine Person läuft wechselt der Fuß ständig zwischen einer Stand und einer Bewegungsphase (vgl. Abb. 4.3). Um zu bestimmen wann eine Standphase beginnt und endet wurde ein Schwellwert basierter Ansatz gewählt (vgl. [6]).

Eine Standphase soll dann erkannt werden, wenn sich der Fuß über einen gewissen Zeitraum („Ruhezeit“) nicht, oder nur sehr wenig, bewegt hat. Dazu wird zunächst, in jedem Zeitschritt, überprüft ob sich die Messwerte der Accelerometer und Gyrometer unterhalb von empirisch ermittelten Schwellwerten befinden. Ist das der Fall, bedeutet dies den Anfang einer potentiellen Standphase. In den folgenden Zeitschritten wird der Durchschnitt der Sensorwerte aufgenommen. Bleibt die Abweichung neuer Messungen von diesem Durchschnitt lange genug unter entsprechenden Schwellwerten wird eine Standphase erkannt.

Der verwendete Algorithmus folgt nun als Pseudocode. Die für die Evaluation verwendete

Belegung der Parameter findet sich in Tabelle 4.1

Eingabe : Gyrometermessungen $\omega_{\mathbf{t}}^{\mathbf{l}}$
Accelerometersmessungen $\mathbf{a}_{\mathbf{t}}^{\mathbf{l}}$
DeltaTime dt (~0.002 Sekunden)

Ausgabe : Standphase?

Initialisierung : $moving$ = true
$st = 0$
gyroBuffer = Ringpuffer mit 200 Einträgen
accBuffer = Ringpuffer mit 200 Einträgen

if *moving* **then**
  `/* Check if we stopped moving */`
  $moving \mid= \|\omega_{\mathbf{t}}^{\mathbf{l}}\| > maxGyroNormForStand$
  $moving \mid= |\|\mathbf{a}_{\mathbf{t}}^{\mathbf{l}}\| - \|\mathbf{g}\|| > maxAccNormForStand$
else
  `/* Check if we are still not moving */`
  $moving \mid= |\omega_{\mathbf{t}}^{\mathbf{l}}.x - gyroBuffer.Average.x| > maxGyroOffsetForWalk.x$
  $moving \mid= |\omega_{\mathbf{t}}^{\mathbf{l}}.y - gyroBuffer.Average.y| > maxGyroOffsetForWalk.y$
  $moving \mid= |\omega_{\mathbf{t}}^{\mathbf{l}}.z - gyroBuffer.Average.z| > maxGyroOffsetForWalk.z$
  $moving \mid= |\mathbf{a}_{\mathbf{t}}^{\mathbf{l}}.x - accBuffer.Average.x| > maxAcceleratorOffsetForWalk.x$
  $moving \mid= |\mathbf{a}_{\mathbf{t}}^{\mathbf{l}}.y - accBuffer.Average.y| > maxAcceleratorOffsetForWalk.y$
  $moving \mid= |\mathbf{a}_{\mathbf{t}}^{\mathbf{l}}.z - accBuffer.Average.z| > maxAcceleratorOffsetForWalk.z$
end
if *moving* **then**
  $st = 0$
  gyroBuffer.Clear
  accBuffer.Clear
else
  st += dt;
  gyroBuffer $\leftarrow \omega_{\mathbf{t}}^{\mathbf{l}}$
  accBuffer $\leftarrow \mathbf{a}_{\mathbf{t}}^{\mathbf{l}}$
end
return $st > minStillTimeForStand$

Der Graph in Abb. 4.4 zeigt den Verlauf der „Ruhezeit“(st) während der letzten 2000 Zeitschritte (~4 Sekunden). Erreicht dieser Wert die Marke von 0.15 Sekunden wird eine Standphase erkannt. Demnach sind hier drei Standphasen zu erkennen.

Paramter	Wert
minStillTimeForStand	$0.15 Sekunden$
maxGyroNormForStand	$20°$
maxAccNormForStand	$0.6 m/s^2$
maxGyroOffsetForWalk.x	$8°$
maxGyroOffsetForWalk.y	$8°$
maxGyroOffsetForWalk.z	$8°$
maxAcceleratorOffsetForWalk.x	$0.8 m/s^2$
maxAcceleratorOffsetForWalk.y	$0.8 m/s^2$
maxAcceleratorOffsetForWalk.z	$2 m/s^2$

Tabelle 4.1

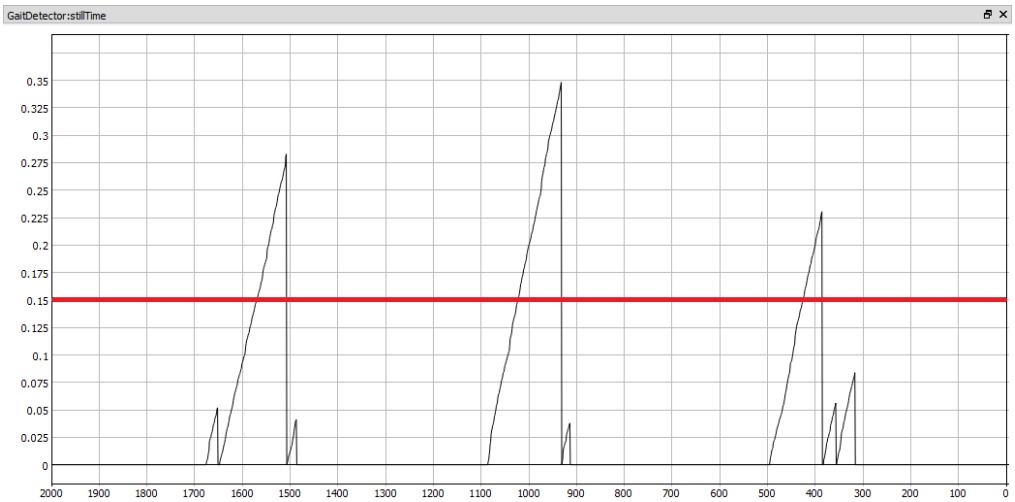


Abbildung 4.4 Die „Ruhezeit“(st) während der letzten 2000 Zeitschritte (~4 Sekunden). Eine Standphase wird erkannt sobald der Wert die Marke von 0.15 Sekunden überschreitet. Demnach sind hier drei Standphasen zu erkennen.

Kapitel 5

Integration von Kartendaten

Um bestimmen zu können, an welcher Position sich eine Person in einem Gebäude befindet, wird natürlich eine Karte dieses Gebäudes benötigt. Dieses Kapitel beschreibt, wie diese Karte zusätzlich verwendet werden kann, um die Genauigkeit der Bewegungsverfolgung aus Kapitel 4 zu verbessern.

Durch die beschriebene Bewegungsverfolgung können nur relative Bewegungen zur Startposition berechnet werden. Da aber der genaue Startpunkt und insbesondere die genaue Ausrichtung der Startposition nicht bekannt sind, kann allein dadurch keine Aussage über die tatsächliche Position der Person in dem Gebäude gemacht werden. Zudem wird die Bewegungsverfolgung, wie bereits beschrieben, über längere Zeit immer ungenauer, so, dass dafür eine Lösung gefunden werden muss.

Die grundsätzliche Idee zur Verbesserung besteht in der Verwendung mehrerer Hypothesen über die Position sowie Rotation der Person. Diese Hypothesen erhalten jeweils die gleiche Eingabe aus der Bewegungsverfolgung, welche jedoch jeweils eine zusätzliche zufällige Komponente bekommen. Außerdem unterscheiden sich die Startpositionen und Rotationen der Hypothesen. Gegeben einer genügend großen Menge an Hypothesen führt dies dazu, dass es immer mindestens eine Hypothese gibt, welche der tatsächlichen Position der Person entspricht oder dieser zumindest sehr nahe kommt.

Zusätzlich wird die Annahme gemacht, dass sich Personen nicht durch Wände bewegen können. Bewegt sich nun also eine der Hypothesen durch eine Wand, ist bekannt, dass diese falsch sein muss.

Die Umsetzung dieser Idee geschieht mit Hilfe eines Partikel-Filters. Im Folgenden werden daher zunächst die Grundlagen von Partikel-Filtern im Allgemeinen beleuchtet (Kapitel 5.1). Anschließend wird erläutert, wie dies für diese Arbeit umgesetzt wurde (Kapitel 5.3, 5.4, 5.5). Außerdem wird die Frage beantwortet, wie aus der Menge unterschiedlicher Hypothesen wieder eine Position berechnet wird, an der sich die Person tatsächlich befindet (Kapitel 5.6).

5.1 Grundlagen des Partikel-Filters

Partikel-Filter sind eine Approximation eines Bayes-Filters (vgl. [5]). Sie gehören in die Familie der der sequentiellen Monte-Carlo-Methoden (vgl. [4], [10]). Partikel-Filter approximieren den Zustand eines Systems (zum Beispiel eine Position) als eine Menge von gewichteten Hypothesen oder *Partikeln* ([23])

$$\mathcal{S} = \{(x_t^i, w_t^i) \mid i = 1, \ldots, N\} \tag{5.1}$$

wobei x_t^i dem Zustand des i-ten Partikels und w_t^i seiner Gewichtung zum Zeitpunkt t entspricht. Die Gewichtung gibt an, wie wahrscheinlich es ist, dass dieser Partikel den tatsächlichen Zustand widerspiegelt. Initialisiert wird dieser Wert mit $w = 1/N$ wobei N die Anzahl der Partikel darstellt.

Der Vorteil des Partikel-Filters gegenüber dem Kalman-Filter, welcher ebenfalls ein Bayes-Filter ist, ist, dass die Schätzung des Zustands hier keine Normal-Verteilung ist. Das heißt, dass beliebige Zustände im Zustandsraum beliebig wahrscheinlich sein können. Es ist also möglich auszudrücken, dass sich die Person zum Beispiel entweder in Raum A oder in Raum B befindet. In diesem Fall würden sich einige Partikel in einem Raum und einige Partikel in dem anderen Raum befinden.

Der Algorithmus des Partikel-Filters besteht im Wesentlichen aus drei Schritten, welche in jedem Zeitschritt ausgeführt werden. Im ersten Schritt wird, ähnlich zum Kalman-Filter, der Zustand jedes Partikels, mit Hilfe eines Modells, weitergetragen. Man erhält also eine Vorhersage darüber, wie der Zustand des Partikels im neuen Zeitschritt aussieht.

Im zweiten Schritt wird dieser neue Zustand mit aktuellen Messungen ($\mathbf{z}_t$) verglichen und seine Gewichtung angepasst. Je besser die Vorhersage zur Messung passt, desto höher wird der Partikel gewichtet.

$$w_t^i = w_{t-1}^i * p(\mathbf{z}_t | \mathbf{x}_t^i) \tag{5.2}$$

Im dritten Schritt, dem so genannten *Resampling*, wird eine neue Menge von Partikeln gebildet. Dazu werden Partikel aus der Menge der bisherigen Partikel entnommen und in die neue Menge eingefügt. Die Wahrscheinlichkeit mit der ein Partikel in die neue Menge übernommen wird ist proportional zu dessen Gewichtung. Dabei kann es, je nach Art des Partikel-Filters, vorkommen, dass einzelne Partikel mehrfach der neuen Menge hinzugefügt werden.

Auf diese Weise werden nach und nach Partikel aus der Menge entfernt bei denen es unwahrscheinlich ist, dass sie dem tatsächlichen Zustand nahe kommen.

5.2 Zustand

Der Zustand eines Partikels im vorliegenden System besteht aus einer 3D-Position, einer 2D-Rotation sowie einer Referenz auf das Stockwerk in dem sich dieser befindet.

$$\mathbf{x}_t = (x_t, y_t, z_t, \theta_t, floor_t) \tag{5.3}$$

$floor_t$ wird benötigt, da die Position beziehungsweise die Bewegung eines Partikels jeweils von den Parametern des aktuellen Stockwerks abhängig sind. Dazu gehören sowohl die Positionen der Wände des Stockwerks sowie dessen Höhe.

5.3 Initialisierung

Für die Initialisierung des Partikel-Filters gibt es im wesentlichen zwei Varianten, welche je nach Vorwissen über die Position des Benutzers bessere Resultate zeigen.

Bekannte Startposition

Ist die Startposition des Benutzers bekannt, kann die Position aller Partikel an dieser Stelle initialisiert werden. Ebenso kann die Rotation aller Partikel mit dem gleichen Wert initialisiert werden, wenn diese bekannt ist. Ist die Rotation der Startposition nicht bekannt, werden die Partikel so angeordnet, dass möglichst viele Richtungen durch Partikel abgedeckt werden. Gegeben einer bekannten Startposition $(x_0, y_0, z_0, floor_0)$ werden die Partikel wie folgt initialisiert

$$\mathbf{x}_0^i = (x_0, y_0, z_0, (\frac{\pi}{2}/N) \cdot i - \pi, floor_0) \tag{5.4}$$

$$\tag{5.5}$$

Unbekannte Startposition

Bei unbekannter Startposition werden die Partikel im Zustandsraum gleich verteilt. Es ist darauf zu achten, dass bei diesem Ansatz mehr Partikel und Zeit benötigt werden, um die Position des Benutzers zu bestimmen. Dies liegt vor allem daran, dass an jeder möglichen Startposition auch alle möglichen Startrotationen möglich sind. Während der Evaluation hat sich herausgestellt, dass eine systematische Verteilung der Partikel im Allgemeinen bessere Resultate liefert als eine zufällige Verteilung. Dies lässt sich vor allem dadurch erklären, dass es um den gesamten Zustandsraum abzudecken, unendlich viele Partikel geben müsste. Da dies natürlich nicht der Fall ist, kann es bei einer zufälligen Verteilung vorkommen, dass kein Partikel mit der annähernd richtigen Rotation in der Nähe der tatsächlichen Position initialisiert wird. Ein Fehler in der Rotation wirkt sich allerdings über längere Strecken viel mehr auf die Position aus als ein kleiner initialer Fehler in der Position selbst. Bei einer

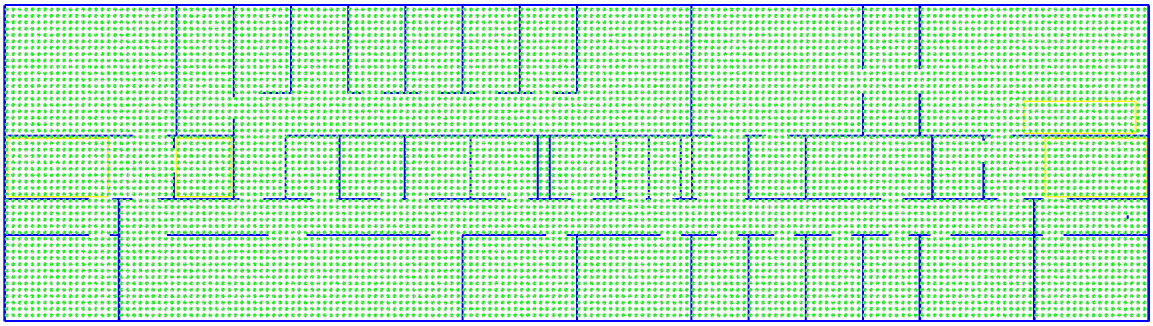


Abbildung 5.1 Systematisch initialisierte Partikel. Es wurden hier 141.120 Partikel pro Etage verwendet.

systematischen Verteilung der Partikel (siehe Abb. 5.1) gibt es zwar größere Lücken zwischen den Positionen aber es ist sichergestellt, dass es einige Partikel mit einer annähernd richtigen Rotation gibt. Es werden daher jeweils 16 Partikel mit gleich verteilten Rotationen an einer Position initialisiert. Die Positionen liegen jeweils 30 Zentimeter von einander entfernt. Dies entspricht etwa 175 Partikeln pro Quadratmeter. Im Fall des Cartesiums sind dies 141.120 Partikel pro Etage.

In der Praxis währe es auch denkbar Partikel an allen möglichen Eingängen des Gebäudes zu initialisieren. Somit währen wesentlich weniger Partikel nötig und die Bewegungen könnten trotzdem zuverlässig verfolgt werden.

5.4 Update Schritt

Der Algorithmus des Partikel-Filters wird immer am Ende einer Standphase ausgeführt. Dies hat zwei Gründe. Ersten sorgen die zusätzlichen Messungen während der Standphase (Kapitel 4.4.2.2) dafür, dass Fehlern in der Schätzung während der Bewegungsphase entgegen gewirkt wird. Die Genauigkeit der Schätzung der Position und der Rotation ist also direkt nach der Standphase am besten. Daher ergibt es Sinn nur diese, „fehlerbereinigten“ Werte für den Partikel-Filter zu verwenden. Zweitens kann auf diese Art ein vollständiger Schritt verarbeitet werden. Mit dem Wissen über die Länge und Rotation des Schritts kann leichter überprüft werden ob der Partikel weiterhin valide ist. Zudem kann das Rauschen, was auf die Position und Rotation jedes Partikels addiert wird, in Abhängigkeit der Größe der Veränderung der Position und Rotation gewählt werden. Dies ergibt Sinn, da bei einem größeren Schritt auch ein größerer Fehler zu erwarten ist.

5.4.1 Update des Zustands

Die Bewegung eines Partikels wird aus der Änderung des Zustands des Kalman-Filters ($\mathcal{X}_t = (\mathcal{C}_t, p_t, v_t)$, vgl. Kapitel 4.3) berechnet.

$$\Delta_x = p_t^x - p_{t-1}^x \tag{5.6}$$
$$\Delta_y = p_t^y - p_{t-1}^y \tag{5.7}$$
$$\Delta_z = p_t^z - p_{t-1}^z \tag{5.8}$$
$$\Delta_\theta = atan2(\mathcal{C}_t^{1,0}, \mathcal{C}_t^{0,0}) - atan2(\mathcal{C}_{t-1}^{1,0}, \mathcal{C}_{t-1}^{0,0}) \tag{5.9}$$

Der neue Zustand eines Partikels ist damit

$$x_t = x_{t-1} + \Delta_x + \epsilon_x \tag{5.10}$$
$$y_t = y_{t-1} + \Delta_y + \epsilon_y \tag{5.11}$$
$$z_t = z_{t-1} + \Delta_z + \epsilon_z \tag{5.12}$$
$$\theta_t = \theta_{t-1} + \Delta_\theta + \epsilon_\theta \tag{5.13}$$

Wobei $\epsilon_x, \epsilon_y, \epsilon_z$ und ϵ_θ normal verteilte Zufallsvariablen sind mit

$$\epsilon_x \sim \mathcal{N}(0, (1 - e^{-0.5|\Delta_x|}) \cdot \Phi_x + \Psi_x) \tag{5.14}$$
$$\epsilon_y \sim \mathcal{N}(0, (1 - e^{-0.5|\Delta_y|}) \cdot \Phi_y + \Psi_y) \tag{5.15}$$
$$\epsilon_z \sim \mathcal{N}(0, (1 - e^{-0.5|\Delta_z|}) \cdot \Phi_z + \Psi_z) \tag{5.16}$$
$$\epsilon_\theta \sim \mathcal{N}(0, (1 - e^{-0.5|\Delta_\theta|}) \cdot \Phi_\theta + \Psi_\theta) \tag{5.17}$$

Die Parameter Φ und Ψ geben dabei an wie groß der vermutete Fehler der Zustandsänderung ist. Dabei wird Φ mit der Größe der Änderung skaliert, so dass große Änderungen mehr Rauschen hervorrufen. Ψ gibt einen festen Fehler an. Dies ist nötig, da sonst kein Rauschen addiert werden würde, wenn keine Bewegung gemessen wird. Allerdings könnte es sich dabei ebenfalls um einen Fehler handeln. Bei der Wahl dieser Parameter sollte darauf geachtet werden, den Fehler nicht zu unterschätzen. Da sich die Partikel dann nicht genügend verteilen können und somit der Zustandsraum nicht ausreichend abgedeckt wird um sicherzustellen, dass mindestens ein Partikel der tatsächlichen Position nahe kommt. Ein Überschätzen des Fehlers hingegen führt nur dazu, dass sich die Partikel weiter von einander entfernen. Dies wird allerdings dadurch ausgeglichen, dass Partikel, die sich durch Wände hindurch bewegen, aussortiert werden. Trotzdem sollten diese Parameter natürlich nicht zu hoch eingestellt werden, da sich die Partikel sonst in größeren Räumen sehr weit verteilen und nicht mehr klar ist wo sich die Person innerhalb des Raums befindet.

Zusätzlich zu der Limitierung der Partikelposition in der x-y-Ebene durch die Wände des Gebäudes ist deren z-Position durch die Höhe des Stockwerks, auf dem sich der Partikel befindet, beschränkt. Diese Beschränkung wird nur aufgehoben, wenn sich der Partikel innerhalb einer

Aufzugs- oder Treppen-Region befindet. In allen anderen Fällen wird dessen z-Position in jedem Updateschritt auf die Höhe des Stockwerks zurück gesetzt.

Ist die Beschränkung aufgehoben wird überprüft, ob der Partikel eine bestimmte Höhe über oder unter dem aktuellen Stockwerk erreicht hat. Ist das der Fall, wird angenommen, dass sich dieser nun auf einem neuen Stockwerk befindet und das Stockwerk des Partikels ($floor$) wird entsprechend angepasst.

$$floor_t = \begin{cases} floor_{t-1} + 1, & \text{wenn } z_t > MapHeight(floor_{t-1}) + 1,8m \\ floor_{t-1} - 1, & \text{wenn } z_t < MapHeight(floor_{t-1}) - 1,8m \\ floor_t, & \text{sonst} \end{cases} \tag{5.18}$$

5.4.2 Update der Gewichtung

Wie bereits erwähnt, ist die Gewichtung eines Partikels insbesondere davon abhängig ob dessen letzte Bewegung durch eine Wand verlaufen ist oder nicht.

$$MapIntersectsWall(x_{t-1}, y_{t-1}, x_t, y_t, floor_t) = [true, false] \tag{5.19}$$

In diesem Fall ist die Gewichtung des Partikels

$$w_t = 0 \tag{5.20}$$

und somit invalide.

Andernfalls werden die Partikel gewichtet in dem die Änderung der Höhe des Partikels im Vergleich zu der des Kalman-Filter-Zustands betrachtet wird (vgl. [23]). Dazu benötigen wir zunächst die Änderung der z-Position des Partikels.

$$\delta_z = (z_t - z_{t-1}) \tag{5.21}$$

Für Partikel innerhalb einer Treppenregion ist δ relativ klein und nur von dem addierten Rauschen abhängig. Befindet sich der Partikel jedoch nicht in einer Treppenregion, wird dessen z-Position auf die Höhe seines aktuellen Stockwerks begrenzt, wodurch $\delta = \Delta_z + \epsilon_z$.

Die neue Gewichtung berechnet sich nun aus der Wahrscheinlichkeit der Änderung der z-Position des Kalman-Filter-Zustands (Δ_z) gegeben der Änderung der z-Position des Partikels (δ_z)

$$P(\Delta_z|\delta_z) \propto \frac{1}{\Phi_z\sqrt{2\pi}} e^{-0.5\frac{(\Delta_z - \delta_z)^2}{{\Phi_z}^2}} \tag{5.22}$$

Vereinfacht kann die Berechnung des neuen Gewichts nun geschrieben werden als

$$w_t = w_{t-1} * e^{-0.5\frac{(\Delta_z - \delta_z)^2}{{\Phi_z}^2}}. \tag{5.23}$$

5.5 Resampling

Partikel, denen eine Gewichtung von Null zugewiesen wurde, werden bei der Berechnung der endgültigen Position des Benutzers (Kapitel 5.6) nicht verwendet, da diese offensichtlich nicht der tatsächlichen Position entsprechen können. Ähnliches gilt für Partikel mit einer sehr geringen Gewichtung, diese haben entsprechend nur einen sehr geringen Einfluss auf die endgültigen Position. Mit der Zeit würden also immer weniger Partikel zur Verfügung stehen mit denen eine sinnvolle Position errechnet werden kann. Im Resampling-Schritt wird dem entgegen gewirkt, in dem diese Partikel durch Partikel mit höherer Gewichtung ersetzt werden. Dazu wird eine neue Menge von Partikeln gebildet welche zufällig aus der bisherigen Menge von Partikeln gezogen werden. Dabei werden Partikel mit höherer Gewichtung vorgezogen. Insbesondere werden Partikel mit einer Gewichtung von Null aussortiert.

In der Literatur findet sich dazu eine große Anzahl von Algorithmen ([18], [3]). In dieser Arbeit wird der relativ einfache, aber nicht weniger effektive, Ansatz des *Low Variance Resamplings* (vgl. [18, Seite 86]) gewählt. Dieser funktioniert wie folgt.

Für jeden Partikel ($\hat{\mathbf{x}}^i$) in der neuen Partikelmenge soll ein Partikel ($\mathbf{x}^j$) aus der alten Menge gesucht um durch diesen ersetzt zu werden. Dabei ist es möglich, dass sich die Anzahl der Partikel in der neuen Menge von der in der alten Menge unterscheidet. Dies ist der Fall, wenn, um einen großen Teil des Zustandsraums abzudecken, eine sehr große initiale Anzahl Partikel gewählt wird. In diesem Fall soll diese Zahl, bis zu einer Mindestanzahl (N^0), verringert werden. Die Zahl der Partikel in der neuen Menge entspricht der Anzahl der Partikel mit einem Gewicht ungleich Null

$$\hat{N} = max(\sum_{n=0}^{N-1} 1\{w^n \neq 0\}, N^0) \tag{5.24}$$

Nun werden die einzelnen Gewichtungen der Partikel aufakkumuliert, so dass

$$acc^j = \sum_{n=0}^{j} w^n. \tag{5.25}$$

Diese Liste wird nun durchlaufen. Der Partikel $\mathbf{x}^j$ wird als $\hat{\mathbf{x}}^i$ in die neue Partikelmenge

übernommen, wenn $acc^j > (r + i \cdot \delta_w)$ mit

$$\delta_w = \sum_{n=0}^{N-1} w^n / \hat{N}, \tag{5.26}$$

$$r \sim \mathcal{U}(0, \delta_w] \tag{5.27}$$

Dabei ist $\mathcal{U}(0, \delta_w]$ die uniforme Verteilung zwischen 0 (inklusive) und δ_w (exklusive).

Als Algorithmus sieht das dann so aus:

```
1  r ~ U(0, δ_w]
2  j = 0
3  for i = 0 to N̂ − 1 do
4  |   while acc^j <= r do
5  |   |   j = j + 1
6  |   end
7  |   x̂^i = x^j
8  |   ŵ^i = 1/N̂
9  |   r = r + δ_w
10 end
```

Dabei ist es unmöglich, dass ein Partikel mit einer Gewichtung von Null in die neue Menge übernommen wird. Das heißt, dass alle Partikel, deren Bewegung durch eine Wand verlaufen ist, aussortiert werden. Partikel mit einer geringen Gewichtung haben ebenfalls eine geringere Chance in die neue Menge übernommen zu werden. Wenn der Benutzer also eine Treppe oder einen Aufzug verwendet, werden Partikel, welche sich nicht in einer Treppen oder Aufzugs-Region befinden, ebenfalls nach kurzer Zeit aussortiert.

5.6 Entgültige Positionsbestimmung

Um die tatsächliche Position $(X_t, Y_t, Z_t, \Theta_t)$ des Benutzers zu erhalten muss diese aus der der Menge der Partikel extrahiert werden. Dazu gibt es eine Reihe von Algorithmen von denen einige von Laue und Röfer [14] beschrieben werden. Die einfachste Variante wäre den Durchschnitt aller Partikel Positionen und Rotationen zu berechnen.

$$X_t = \sum_{i=0}^{N-1} x_t^i / N \tag{5.28}$$

$$Y_t = \sum_{i=0}^{N-1} y_t^i / N \tag{5.29}$$

$$Z_t = \sum_{i=0}^{N-1} z_t^i / N \tag{5.30}$$

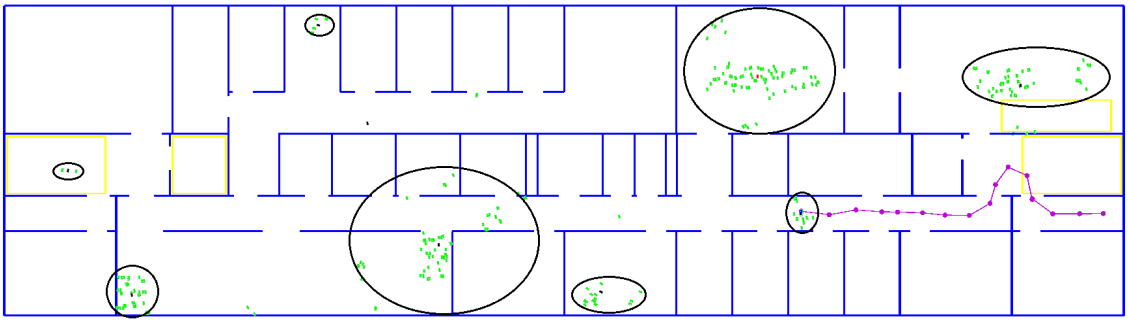


Abbildung 5.2 Bereits nach wenigen Schritten bilden sich einzelne Cluster von Partikeln. Sobald nur noch ein Cluster vorhanden ist, ist die Positionssuche abgeschlossen.

Dabei muss darauf geachtet werden, dass der Durchschnitt der Rotation, auf Grund der Singularität bei $+/-\pi$, nicht durch einfaches addieren und dividieren berechnet werden kann. So sollte zum Beispiel der Durchschnitt von $(\pi - \epsilon)$ und $(-\pi + \epsilon)$, für kleine ϵ, π bzw. $-\pi$ sein. Stattdessen erhält man 0. Eine Lösung für dieses Problem (vgl. [23]) ist es, stattdessen zunächst den Richtungsvektor (h_x, h_y) zu berechnen und den Winkel (Θ) über diesen zu bestimmen.

$$h_x = \sum_{i=0}^{N-1} cos\ \theta_t^i \tag{5.31}$$

$$h_y = \sum_{i=0}^{N-1} sin\ \theta_t^i \tag{5.32}$$

$$\Theta_t = atan2(h_y, h_x) \tag{5.33}$$

Im Fall, dass die Startposition bekannt ist und somit alle Partikel relativ nah beieinander liegen, sind die Ergebnisse dieses Ansatzes durchaus akzeptabel.

Ist die Startposition jedoch nicht bekannt, sind zu Beginn der Positionssuche alle Partikel über das gesamte Gebäude verstreut. In diesem Fall lässt sich zunächst keine Aussage darüber treffen, wo die tatsächliche Position liegt. Nach einigen Schritten bilden sich allerdings bereits einige Häufungen (Cluster) von Partikeln (vgl. Abb. 5.2). Der einfache Ansatz würde hier weiterhin kein sinnvolles Ergebnis liefern. Stattdessen wird hier ein Algorithmus verwendet welcher diese Cluster erkennt und zumindest einige ungefähre Positionen ausgeben kann (vgl. [23]). Je mehr sich die Person bewegt, desto weniger Cluster gibt es. Sobald nur noch ein Cluster vorhanden ist, ist die Position gefunden.

Um einen Cluster zu finden wird zunächst ein zufälliges Partikel als Mittelpunkt des neuen Clusters ausgewählt. Alle Partikel, die sich innerhalb einer bestimmten Distanz zum Mittelpunkt des Clusters befinden, werden diesem zugeordnet. Danach wird aus diesen Partikeln ein neuer Mittelpunkt berechnet, in dem die durchschnittliche Position wie oben bestimmt wird. Dies wird solange wiederholt, bis dem Cluster keine neuen Partikel hinzugefügt werden. Um weiter Cluster zu finden, wird der Algorithmus erneut mit den übrigen Partikeln ausgeführt. Solange bis alle Partikel einem Cluster zugeordnet sind.

Der Abstand den ein Partikel zum Mittelpunkt eines Clusters haben muss, um zu diesem hinzugefügt zu werden, wurde empirisch ermittelt. Er liegt in der x-y-Ebene bei 5 Metern und in der z-Ebene bei einem Meter.

Da dieser Algorithmus einen quadratischen Aufwand bezüglich der Anzahl der Partikel besitzt, wird er erst ausgeführt, sobald die Anzahl der Partikel unterhalb einer Maximalanzahl liegt. Dies ist jedoch keine wirkliche Einschränkung, da bei einer sehr hohen Anzahl von Partikeln diese über das gesamte Gebäude verteilt sind, so dass noch keine eine sinnvolle Aussage über die Position getroffen werden kann.

Kapitel 6

Grafische Benutzeroberfläche

Um das vorgestellte System zu steuern und um die Ausgaben visualisieren zu können wurde eine grafische Benutzeroberfläche (Abb. 6.1) implementiert. Sie basiert auf Qt5 [1] und OpenGL [2] und wurde in Aufbau und Funktionsweise an die Simulationsoftware *SimRobot* [15] angelehnt. Die Benutzeroberfläche (GUI) besteht aus mehreren Teilen und Funktionen, welche im Folgenden erklärt werden.

6.1 Gesamtansicht

In Abb. 6.1 wird eine mögliche Konfiguration der Benutzeroberfläche dargestellt. Die einzelnen Ansichten (Views) können dabei frei eingegliedert und verschoben werden. Welche Daten zu einem Zeitpunkt angezeigt werden, kann also vom Benutzer ebenso frei gewählt werden, wie die Größe und die Aufteilung der einzelnen Views.

6.1.1 ViewSelector

Das zentrale Element zur Verwaltung der Views stellt der sogenannte *ViewSelector* (vgl. Abb. 6.2) dar. Durch diesen können ähnlich der TreeView in *SimRobot* einzelne Views per Mausklick geöffnet werden. Der *ViewSelector* enthält die *MapView*, die *RotationView*, sowie eine Reihe von *PlotViews*.

6.1.2 Toolbar

Die *Toolbar* Abb. 6.3 dient zur Steuerung des Gesamtsystems. Es sind die folgenden Funktionen verfügbar:

- *RESET*: Versetzt das System in den Ausgangszustand. Der Kalman-Filter und der

[1] http://www.qt.io
[2] http://www.opengl.org

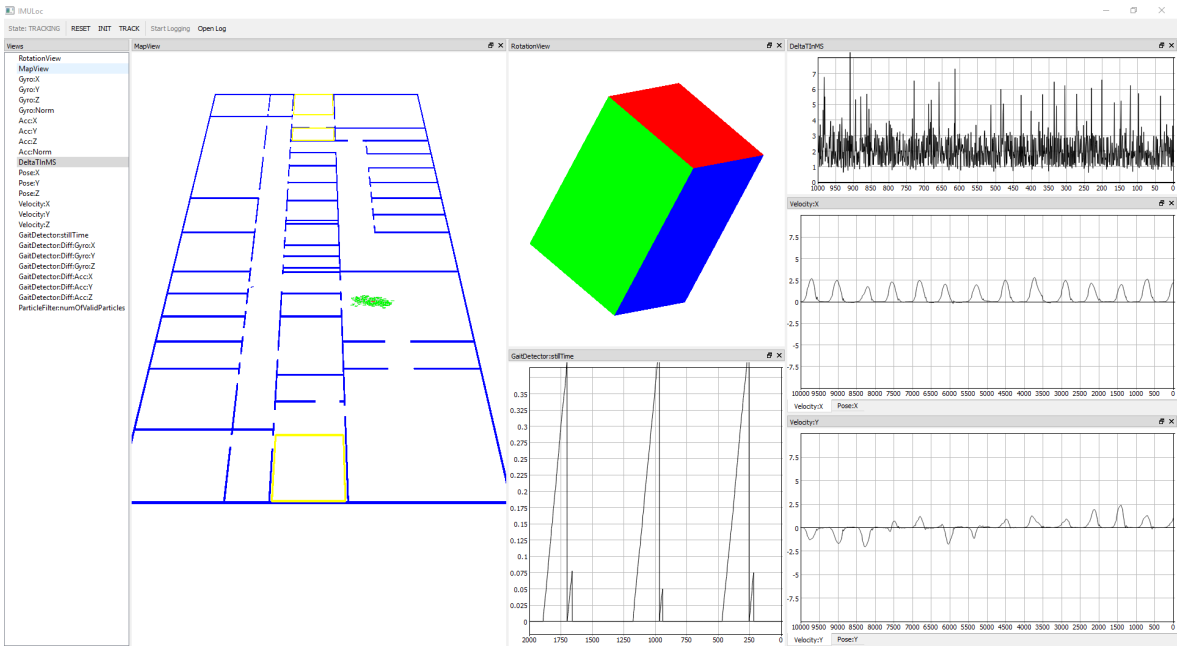


Abbildung 6.1 Die Benutzeroberfläche in einer möglichen Konfiguration.

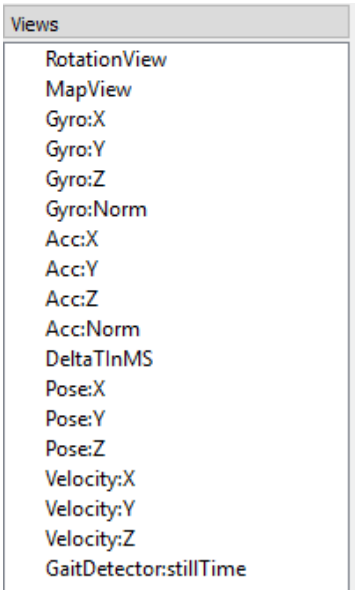


Abbildung 6.2 Im *ViewSelector* können verschiedene Views zum öffnen ausgewählt werden. Diese müssen vorher im Quellcode festgelegt werden.

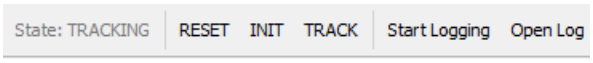


Abbildung 6.3 Über die *Toolbar* kann das System gesteuert werden.

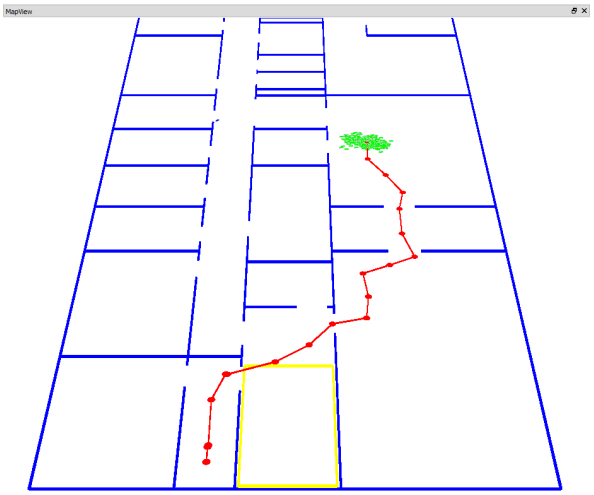


Abbildung 6.4 Die *MapView* zeigt die Gebäudekarte in 3D sowie weitere Zeichnungen wie die derzeitige Positionsschätzung, den zurück gelegten Pfad und die Position der einzelnen Partikel des Partikel-Filters.

Partikel-Filter werden initialisiert.

- *INIT*: Versetzt das System in den Initialisierungszustand. Dies ist der Zustand in dem die Startrotation der *IMU* ermittelt wird.
- *TRACK*: Versetzt das System in den Verfolgungszustand. In diesem Zustand werden die Bewegungen des Benutzers verfolgt und in der *MapView* angezeigt.
- *Start Logging*: Startet die Aufnahme eines Logs. Alle Sensordaten sowie der Systemzustand werden in eine Datei geschrieben und können später ausgelesen und wiedergegeben werden. Um die Aufnahme anzuhalten kann der Knopf erneut betätigt werden, er ist nun mit *Stop Logging* beschriftet.
- *Open Log*: Öffnet einen Dateiauswahldialog über den eine Logdatei zur Wiedergabe ausgewählt werden kann.

6.2 Views

In den Views werden konkrete Daten des System visualisiert. Dazu gibt es drei verschiedene Typen:

6.2.1 MapView

Der wichtigste Bestandteil der GUI ist die *MapView*. Sie zeigt neben der verwendeten Gebäudekarte und der aktuellen Positionsschätzung beliebige weitere Zeichnungen. So zeigt (vgl. Abb. 6.4) auch die Position aller Partikel des Partikel-Filters in grün, sowie den Pfad der

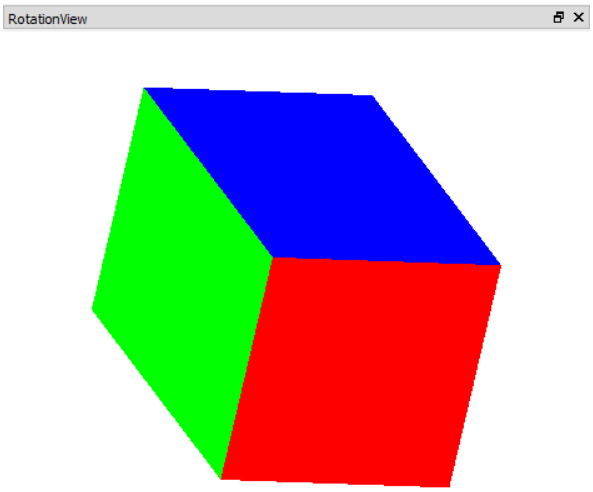


Abbildung 6.5 Die *RotationView* zeigt die aktuelle Schätzung der Lage der *IMU*.

bisher zurück gelegt wurde in rot. Weitere Zeichnungen können leicht hinzugefügt werden in dem die entsprechenden Makros innerhalb des Quellcodes verwendet werden.

Um die eingebauten Zeichnungen zu aktivieren beziehungsweise zu deaktivieren können folgende Hotkeys verwendet werden:

- *A*: Die ersten Ebene der Gebäudekarte
- *S*: Die zweiten Ebene der Gebäudekarte
- *P*: Die Partikel des Partikel-Filters (grün)
- *Y*: Die Schätzung des bisher zurück gelegten Pfads (rot)
- *X*: Die Schätzung des bisher zurück gelegten Pfads ohne Verwendung des Partikel-Filters (violet)

Die Karte kann mit Hilfe der linken Maustaste bewegt werden. Die rechte Maustaste dient zusammen mit den Tasten *Strg* und *Shift* zum rotieren der Karte. Durch die Verwendung des Mausrads kann herran und herraus gezoomt werden.

6.2.2 RotationView

Die *RotationView* (vgl. Abb. 6.5) zeigt die aktuelle Schätzung der Lage der *IMU*. Die rote Seite des Würfels zeigt in Richtung der x-Achse der Imu, die grüne Seite in Richtung der y-Achse und die blaue Seite in Richtung der z-Achse.

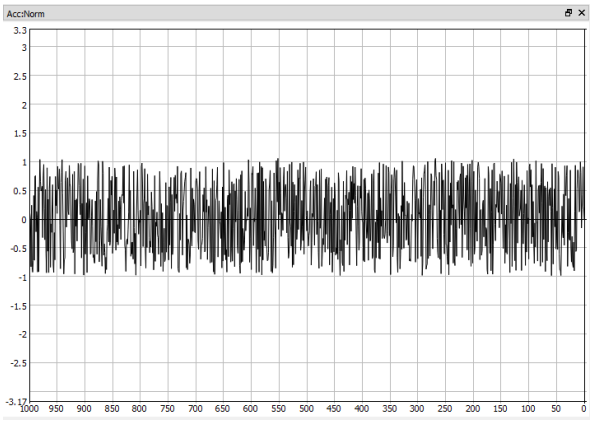


Abbildung 6.6 Die *PlotView* zeigt den Verlauf von Werten über die Zeit. In diesem Beispiel die Norm des gemessenen Beschleunigungsvektors.

6.2.3 Plots

Die *PlotView* (vgl. Abb. 6.6) zeichnet den Verlauf von bestimmten Werten über die Zeit. Innerhalb des Quellcodes können beliebig viele *PlotView* erstellt und mit Werten versorgt werden. Es ist dabei auch möglich mehrere Graphen in der selben *PlotView* darzustellen. Die verfügbaren *PlotView*s sind im *ViewSelector* zu finden und können von dort geöffnet werden. Mit Hilfe der linken Maustaste sowie dem Mausrad kann der Wertebereich des Graphen angepasst werden. Die Anzahl der angezeigten Werte kann mit Hilfe von *Strg* + Mausrad geändert werden.

Kapitel 7

Evaluation

In diesem Kapitel wird die Genauigkeit der einzelnen Systemkomponenten sowie des Gesamtsystems evaluiert. Es folgt daher zunächst die Evaluation des Schrittdetektors (Kapitel 7.1) sowie des Gesamtsystems (Kapitel 7.2) und der Verfolgung der Position nur mit Hilfe der IMU-Daten (Kapitel 7.3), also ohne Verwendung des Partikel-Filters. Anschließend werden die Ergebnisse einiger Tests vorgestellt, welche die Funktionalität des System in verschiedenen Situationen zeigen. Dies sind die Verwendung von Aufzügen (Kapitel 7.4), das Verhalten bei langen Laufzeiten und großen Räumen (Kapitel 7.5), sowie die Positionsfindung bei unbekannter Startposition (Kapitel 7.6 und 7.7).

7.1 Schrittdetektor

Um die Erkennungsrate des Schrittdetektors zu testen wurde drei Testpersonen aufgetragen jeweils 200 Schritte zu laufen. Dabei wurden jeweils 100 Schritte in einem, für die jeweilige Person, normalem Tempo und 100 Schritte in einem erhöhten Tempo gegangen. Die Ergebnisse finden sich in Tab. 7.1.

	Erkennungrate		False-Positives
	Normales Tempo	Erhöhtes Tempo	
Person 1	93%	69%	0
Person 2	83%	68%	0
Person 3	96%	74%	0
Gesamt	91%	70%	0
	80%		

Tabelle 7.1 Ergebnisse der Evaluation des Schrittdetektors

Die verwendeten Parameter (Tab. 4.1) wurden anhand der Bewegungen der ersten Person kalibriert. Für die Verwendung mit den anderen Testpersonen wurden keine Änderungen vorgenommen. Es zeigt sich also, dass die Schrittdetektion nicht zur Gänze unabhängig von

der Gangart des Verwenders ist. Wie in den nächsten Abschnitten gezeigt wird, reichen allerdings auch diese, etwas niedrigeren, Erkennungsraten aus um Bewegungen der Testperson zu verfolgen.

Insbesondere zeigte sich wie in Tab. 7.1 zu sehen, dass keinerlei False-Positives auftraten. Dies ist für die Funktion des Gesamtsystems besonders wichtig, da diese die Schätzung stark verschlechtern würden. Dagegen führen fehlende Erkennungen lediglich dazu, dass die Schätzung in diesem Schritt nicht verbessert werden kann.

Während der Entwicklung des Schrittdetektors zeigte sich bereits, dass die Erkennungsrate mit steigenden Laufgeschwindigkeit abnimmt. Bei einer rennenden Person ist davon auszugehen, dass keine Schritte erkannt werden. Es wurde daher hier auf eine genaue Evaluation verzichtet.

7.2 Genauigkeit des Systems

Um die Genauigkeit der Positionsverfolgung zu evaluieren wurde ein Pfad, wie in Abbildung 7.1 zu sehen, markiert. Dieser wurde von den Testpersonen abgeschritten. Der Partikel-Filter wurde dazu, wie in Kapitel 5.3 beschrieben, mit einer bekannten Startposition aber unbekannten Startrotation initialisiert. Abbildung 7.2 zeigt die einzelnen Wegpunkte in grün sowie den geschätzten Pfad von einem der Testläufe in rot. Die roten Punkte geben dabei jeweils an, an welcher Position eine Standphase festgestellt wurde. Tabelle 7.2 beinhaltet die durchschnittliche Abweichung der erkannten Standpositionen von den Wegpunkten. Die durchschnittliche Abweichung aller Testläufe betrug 279 mm. Die hohe maximale Abweichung von 985 mm lässt sich durch nicht erkannte Standphasen erklären.

	Durschnittliche Abweichung		Maximale Abweichung	
	Vorwärts	Rückwärts	Vorwärts	Rückwärts
Person 1	265 mm	232 mm	638 mm	799 mm
Person 2	374 mm	197 mm	985 mm	510 mm
Person 3	202 mm	406 mm	475 mm	715 mm
Gesamt	280 mm	278 mm	985 mm	799 mm
	279 mm		985 mm	

Tabelle 7.2 Ergebnisse der Evaluation zur Genauigkeit des Systems. Die verwendete Strecke wird in Abb. 7.2 gezeigt.

Abbildung 7.3 zeigt einen Rundgang über die ersten beiden Etagen des Cartesiums. Es ist insbesondere zu sehen, dass auch die Verwendung von Treppen kein Problem dar stellt und das System auch über lange gerade Strecken keinen großen Fehler aufbaut.

Abbildung 7.1 Ein Teil der Wegpunkte für die Evaluation der Genauigkeit der Positionsverfolgung.

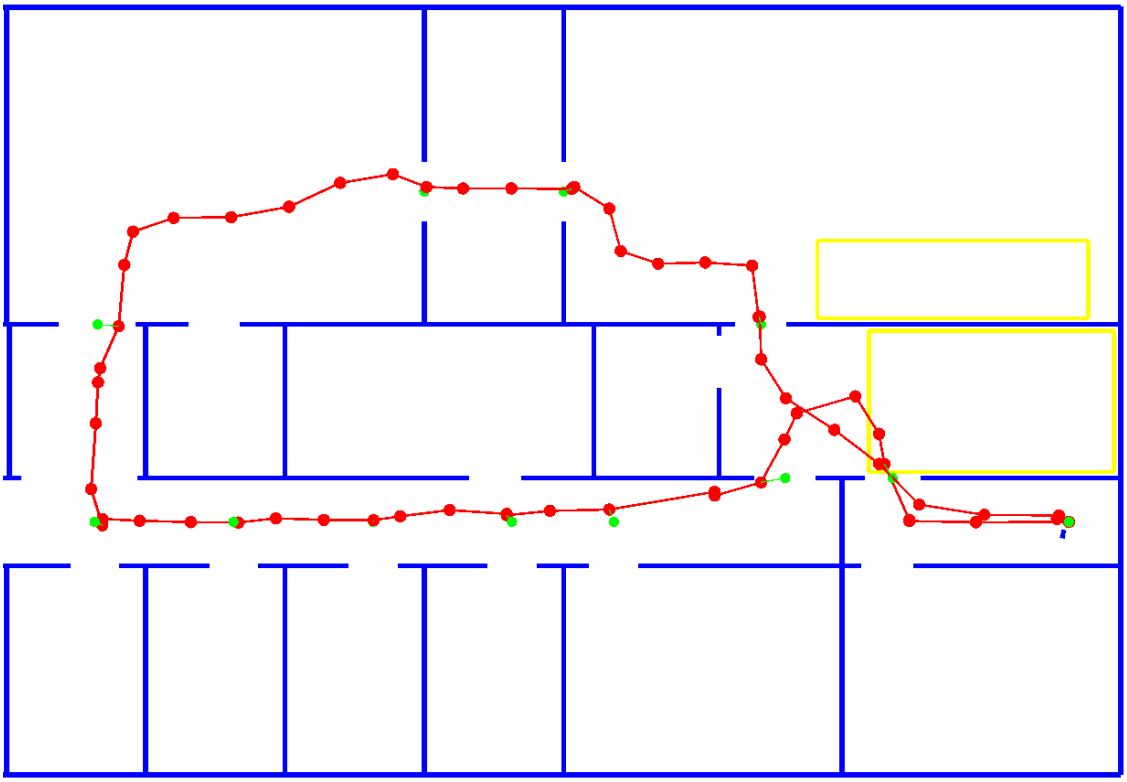


Abbildung 7.2 Der Pfad, der zur Evaluation der Genauigkeit der Positionsverfolgung verwendet wurde. Die einzelnen Wegpunkte sind grün markiert. Die Abbildung zeigt außerdem den geschätzten Pfad sowie die geschätzten Standpositionen im zweiten Testlauf von Person 1 in rot.

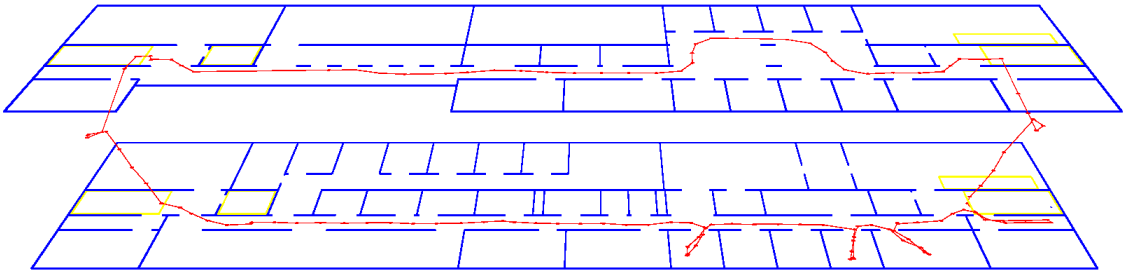


Abbildung 7.3 Ein Testlauf über mehrere Ebenen des Cartesiums.

7.3 Genauigkeit nur IMU

Um die Genauigkeit des Systems ohne die Verwendung des Partikel-Filters zu testen wurde die Startposition und Rotation der Schätzung des Kalman-Filters mit dem tatsächlichen Wert initialisiert. Daraufhin, kann wie in Kapitel 7.2 beschrieben, jeweils der Abstand der geschätzten Position zu den markierten Wegpunkten berechnet werden. Tabelle 7.3 zeigt die Ergebnisse dieses Tests. Es ist zu sehen, dass die Abweichungen, bei kurzen Strecken, mit 266 mm tatsächlich leicht niedriger liegen als mit der Verwendung des Partikel-Filters. Dies liegt unter anderem daran, dass sich die Partikel des Partikel-Filters einen zufälligen Anteil haben, sich weiter ausbreiten und somit der Mittelpunkt nicht immer genau an der tatsächlichen Position liegt. Über längere Strecken steigt die Abweichung aus den in Kapitel 2.2 angegeben Gründen mit der Zeit allerdings immer mehr wenn kein Partikel-Filter verwendet wird. Da für diese Auswertung die selben Messwerte verwendet wurden wie in Abschnitt 7.2 wurden durch den Schrittdetektor auch die selben Standphasen erfasst. Daher lässt sich auch hier eine hohe maximale Abweichung von 950 mm feststellen.

	Durschnittliche Abweichung		Maximale Abweichung	
	Vorwärts	Rückwärts	Vorwärts	Rückwärts
Person 1	282 mm	260 mm	744 mm	601 mm
Person 2	375 mm	191 mm	950 mm	416 mm
Person 3	190 mm	297 mm	408 mm	536 mm
Gesamt	282 mm	249 mm	950 mm	601 mm
	266 mm		950 mm	

Tabelle 7.3 Ergebnisse der Evaluation zur Genauigkeit der Positionsverfolgung nur mit Hilfe der *IMU*. Es wurden die selben Messwerte wie für die Evaluation in Tab. 7.2 verwendet.

In Abbildung 7.4 ist die geschätzte Position ohne Verwendung des Partikel-Filters bei einem längeren Testlauf abgebildet. Es zeigt sich, dass auch ohne die Verwendung des Partikel-Filters ein, je nach Anwendungsfall, brauchbares Ergebnis geliefert werden kann. Dazu muss jedoch die Startposition und insbesondere die Startrotation genau bekannt sein da zum Beispiel eine Abweichung von nur 5 Grad in einer Entfernung von 50 Metern bereits einen Fehler von etwas mehr als 4 Metern verursacht.

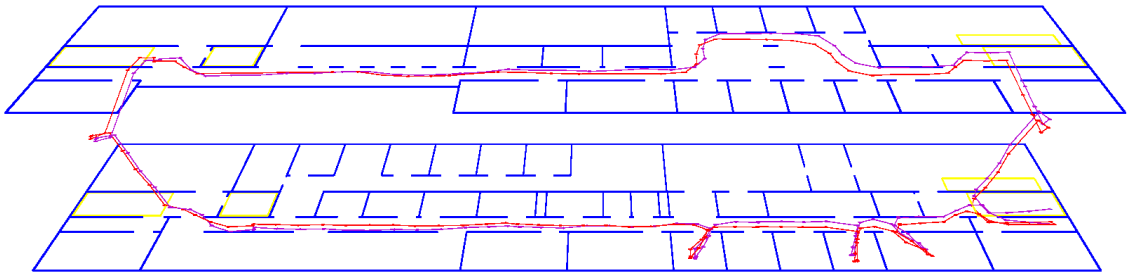


Abbildung 7.4 Ein Testlauf über mehrere Ebenen des Cartesiums. Das Bild zeigt den geschätzten Pfad einmal ohne Verwendung des Partikel-Filters (violett) und einmal mit (rot).

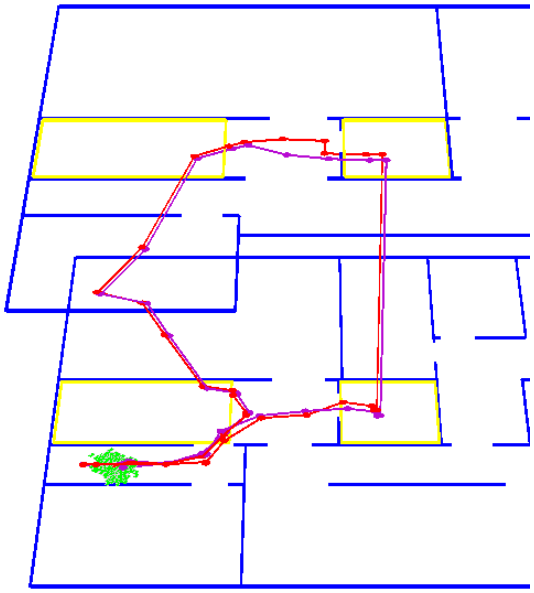
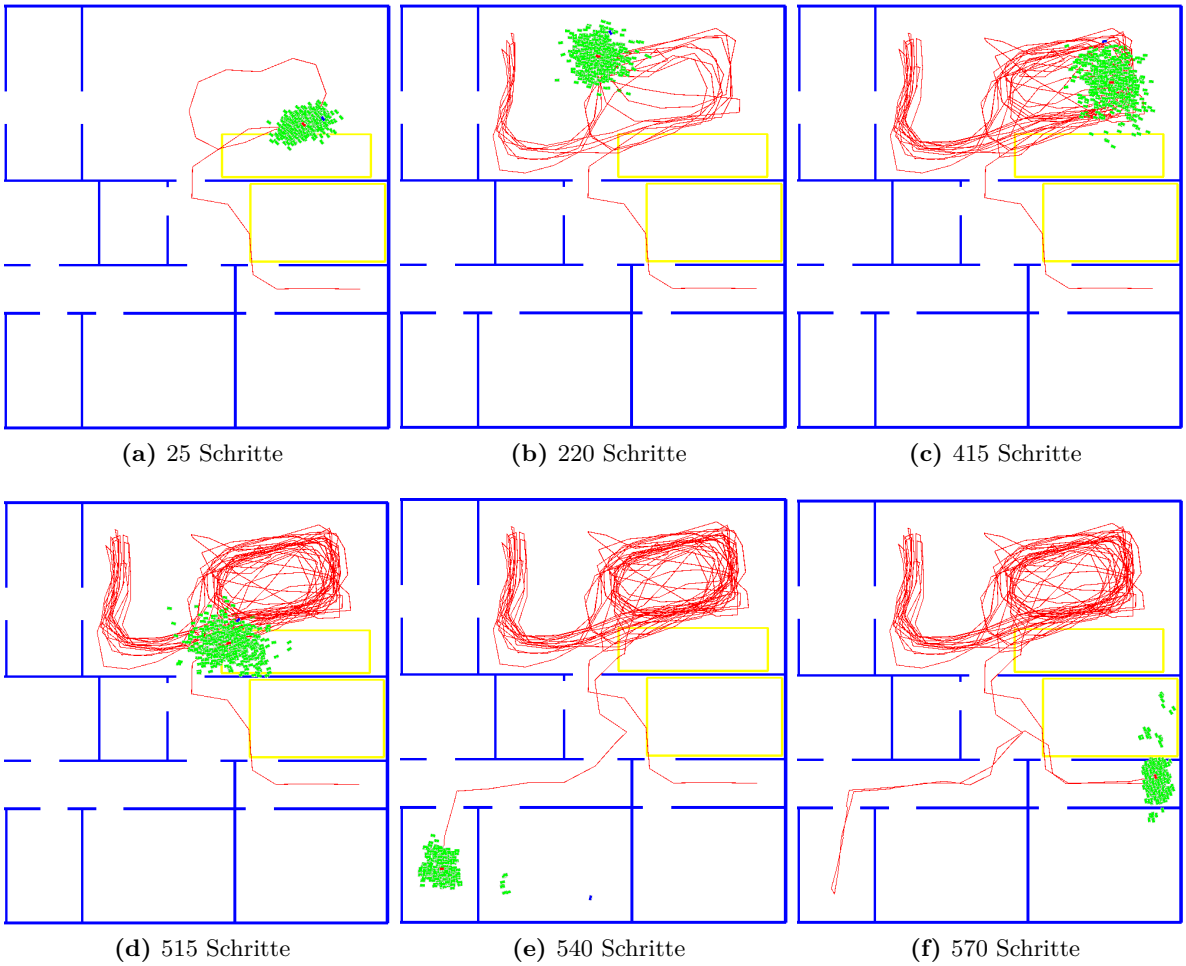


Abbildung 7.5 Ein Test der Funktion bei Verwendung von Aufzügen. Der rote Pfad entspricht der Schätzung des Partikelfilters, der violette der des Kalman-Filters.

7.4 Verwendung von Aufzügen

Wie in Abbildung 7.5 gezeigt wird, ist auch die Verwendung von Aufzügen mit dem entwickelten System möglich. Allerdings muss, wie in Kapitel 4 erläutert, dafür darauf verzichtet werden bei einer Standphase ein *Zero-Velocity-Update* für die z-Richtung auszuführen, sobald sich Partikel innerhalb einer Aufzugregion befinden. Dies führt zu einer leichten Verschlechterung der Genauigkeit der Schätzungen wenn sich die Person nicht tatsächlich wartend innerhalb des Aufzugs befindet sondern sich bewegt. Dies ist insbesondere dann der Fall wenn die Startposition unbekannt ist, da dann auch innerhalb der Aufzugregion Partikel eingestreut werden.



(a) 25 Schritte **(b)** 220 Schritte **(c)** 415 Schritte

(d) 515 Schritte **(e)** 540 Schritte **(f)** 570 Schritte

Abbildung 7.6 Ein Testlauf über 10 Minuten im größten Raum des Cartesiums. Der geschätzte Pfad ist in rot eingezeichet, die Position der einzelnen Partikel in grün.

7.5 Verwendung in großen Räumen

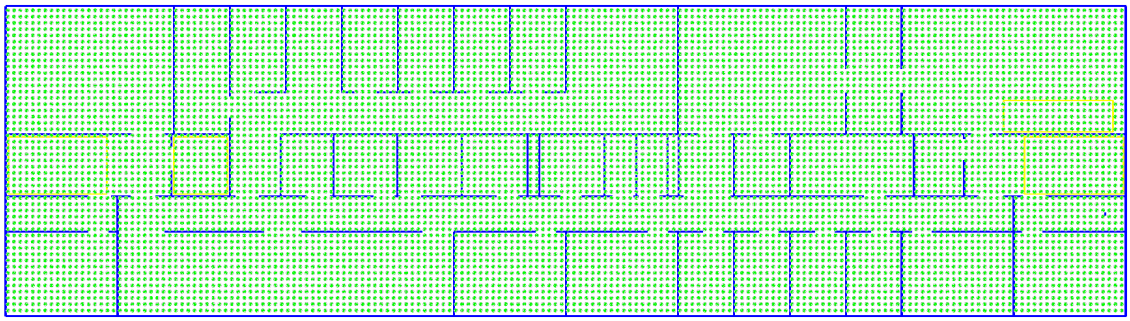
Abbildung 7.6 zeigt einen Testlauf über 10 Minuten im größten Raum des Cartesiums. In den Abbildungen 7.6a bis 7.6c lässt sich gut erkennen, dass sich die Partikel mit der Zeit immer weiter verteilen. Das System ist sich also mit der Zeit immer weniger sicher wo sich die Person genau befindet. Da die Verteilung allerdings durch die Wände des Raums begrenzt wird, bleiben die Partikel zu jedem Zeitpunkt dicht genug beieinander um eine Positionsbestimmung möglich zu machen. Dass die geschätzte Position tatsächlich weiterhin der realen Position entspricht, lässt sich in den Abbildungen 7.6d bis 7.6f erkennen. Nach dem Ende des Testlaufs in Abbildung 7.6d, verlässt die Person den Raum und begibt sich zunächst in ein kleineres Büro (Abb. 7.6e) und dann zurück an die Startposition (Abb. 7.6f).

7.6 Positionsfindung bei unbekannter Startposition

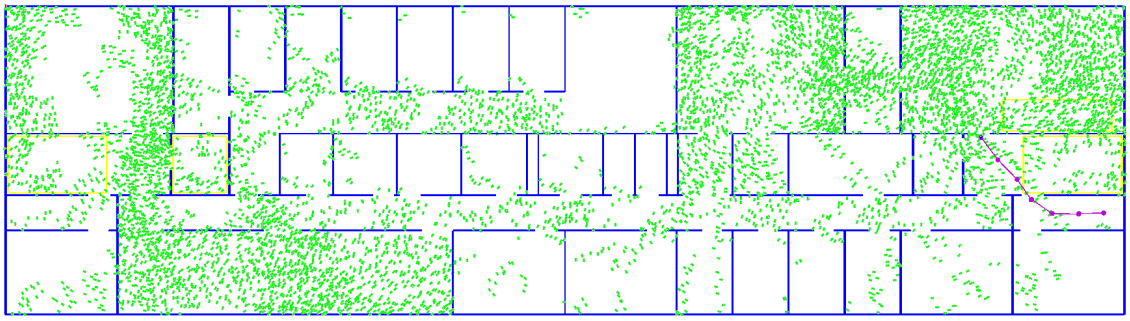
Wie bereits in Kapitel 5.3 angesprochen wurde, kann es vorkommen, dass die tatsächliche Startposition des Nutzers nicht bekannt ist. In diesem Fall können die Partikel nicht alle an der gleichen Position initialisiert werden. Stattdessen werden diese, wie in Abbildung 7.7a gezeigt, systematisch verteilt. Um sicher zu gehen, dass die tatsächliche Position des Nutzers von mindestens einem Partikel abgedeckt wird muss anfänglich eine relativ große Menge von Partikeln verwendet werden. Während die Wahrscheinlichkeit einer erfolgreichen Positionsfindung natürlich mit der Anzahl der Partikel steigt muss darauf geachtet werden, dass das System weiterhin echtzeitfähig bleibt und daher keine beliebig große Anzahl gewählt werden kann. Für das Cartesium haben sich 141.120 Partikel pro Etage als ausreichend herausgestellt.

Abbildung 7.7 zeigt einen Testlauf bei unbekannter Startposition. Der anhand des Kalman-Filters bestimmte Pfad ist in violett eingezeichnet und entspricht in etwa dem tatsächlichen Pfad. Der Kalman-Filter wurde dazu mit der tatsächlichen Startposition initialisiert. Abbildung 7.7a zeigt die neu initialisierten Partikel. Bereits nach fünf Schritten haben sich, wie in Abbildung 7.7b zu sehen, einige Häufungen gebildet. Andere Regionen beinhalten keine Partikel mehr. Des weiteren ist zu erkennen, dass hier die Anzahl der verwendeten Partikel bereits stark abgenommen hat. Zum Zeitpunkt von Abbildung 7.7c haben sich die meisten Partikel in einem Cluster zusammen gefunden. Damit ist die Positionsfindung abgeschlossen. In Abbildung 7.7d ist der Pfad des Nutzers ab dem Zeitpunkt der Positionsfindung in rot eingezeichnet.

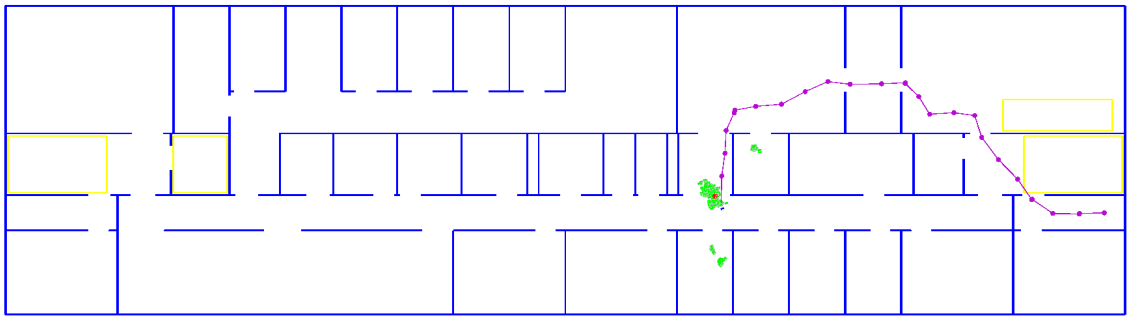
Wie bereits in Kapitel 5.5 beschrieben werden invalide Partikel, also solche, die sich durch eine Wand bewegt haben, im Resampling-Schritt des Partikel-Filters nicht neu eingestreut wenn die Gesamtzahl der Partikel über einem Maximalwert (hier 501) liegt. Der Grund dafür ist, dass zwar viele Partikel benötigt werden um eine Position zu finden, wenn diese aber gefunden ist, es nur wenige braucht um diese weiter zu verfolgen. Abbildung 7.8 zeigt daher die Anzahl der verwendeten Partikel innerhalb der ersten 20 Sekunden des Testlaufs. Es ist gut zu erkennen, dass insbesondere die ersten Schritte dazu führen, dass viele Partikel aussortiert werden. Nach 2 Schritten sind dies bereits 50 Prozent, nach 5 Schritten 90 Prozent.



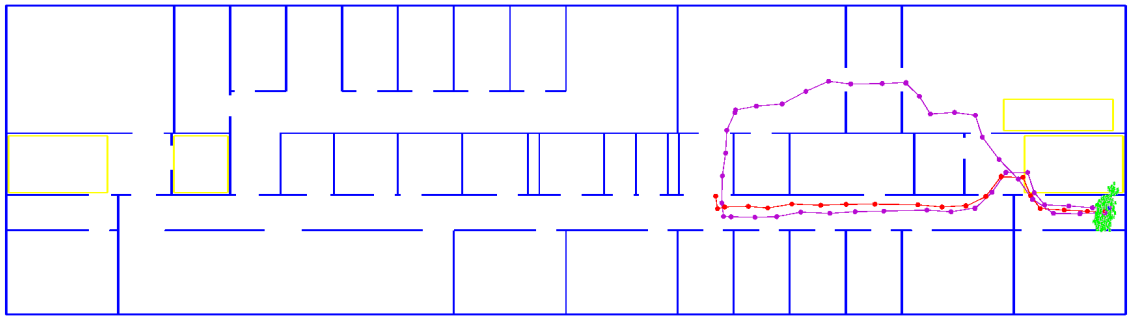
(a)



(b)



(c)



(d)

Abbildung 7.7 Ein Testlauf mit unbekannter Startposition. Die Anhand der *IMU* bestimmte Position ist in violett eingezeichnet und entspricht in etwa der tatsächlichen Position. Abbildung (a) zeigt die neu initialisierten Partikel. In Abbildung (b) haben sich bereits einige Häufungen von Partikeln gebildet. Zum Zeitpunkt von Abbildung (c) wurde die tatsächliche Position bestimmt welche dann weiter verfolgt wird. Der Weg der ab diesem Zeitpunkt zurück gelegt wurde ist in Abbildung (d) in rot markiert.

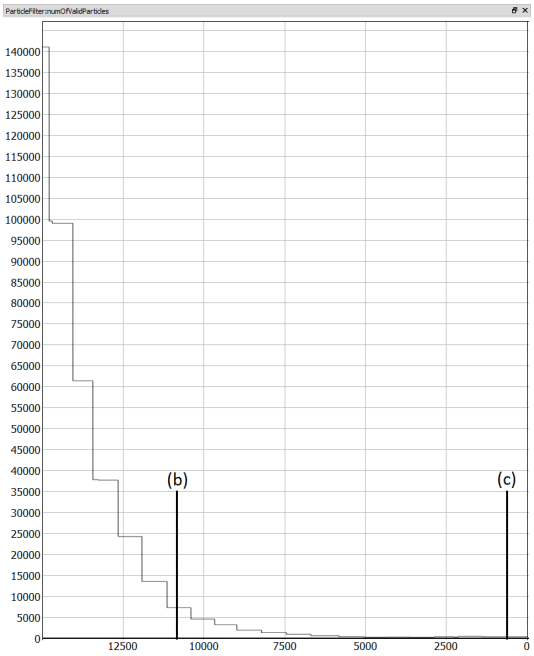


Abbildung 7.8 Die Anzahl der verwendeten Partikel während der ersten 20 Sekunden des Testlaufs. Es ist zu sehen, dass diese mit jedem Schritt abnimmt da immer mehr Partikel als nicht valide gekennzeichnet werden. Zudem sind die Zeitpunkte der Abbildungen 7.7b und 7.7c eingezeichnet.

7.7 Positionsfindung unter Verwendung einer Treppe

Die Gewichtung der Partikel wird, wie in Kapitel 5.4.2 beschrieben, auch in Abhängigkeit der Änderung der Höhe der Partikel relativ zu der der Schätzung des Kalman-Filters berechnet. Wird also eine zum Beispiel eine Treppe benutzt, werden im Resampling-Schritt des Partikel-Filters (Kapitel 5.5) mit der Zeit auch die Partikel aussortiert, die sich nicht in der Nähe einer Treppen- oder Aufzugregion befinden. Abbildung 7.9 zeigt das Verhalten des Systems bei unbekannter Startposition und der Verwendung einer Treppe.

Der anhand des Kalman-Filters bestimmte Pfad ist in violett eingezeichnet und entspricht wieder in etwa dem tatsächlichen Pfad. Der Kalman-Filter wurde dazu wieder mit der tatsächlichen Startposition initialisiert. Abbildung 7.9a zeigt die neu initialisierten Partikel. Die Abbildungen 7.9b und 7.9c zeigen die Verteilung der Partikel nach zwei beziehungsweise drei Treppenstufen. Es ist gut zu erkennen, dass die Anzahl der Partikel außerhalb von Treppen- und Aufzugregionen sehr schnell stark abnimmt.

In Abbildung 7.9d wurde die zweite Etage erreicht. In der unteren Etage sind nun kaum noch Partikel vorhanden. Da hier nur zwei Etagen modelliert wurden, wurden alle Partikel, die sich in die dritte Etage begeben haben wurden so behandelt, als wenn sie sich durch eine Wand hindurch bewegt hätten und wurden aussortiert. Eine weitere wichtige Beobachtung ist, dass durch die Symmetrie des Gebäudes nicht klar ist, ob sich die Person an der Treppe auf der linken oder auf der rechten Seite der Abbildung befindet. Dies ist auch in Abbildung 7.9e gut zu erkennen. Hier befindet sich die Person wieder im Erdgeschoss des Gebäudes.

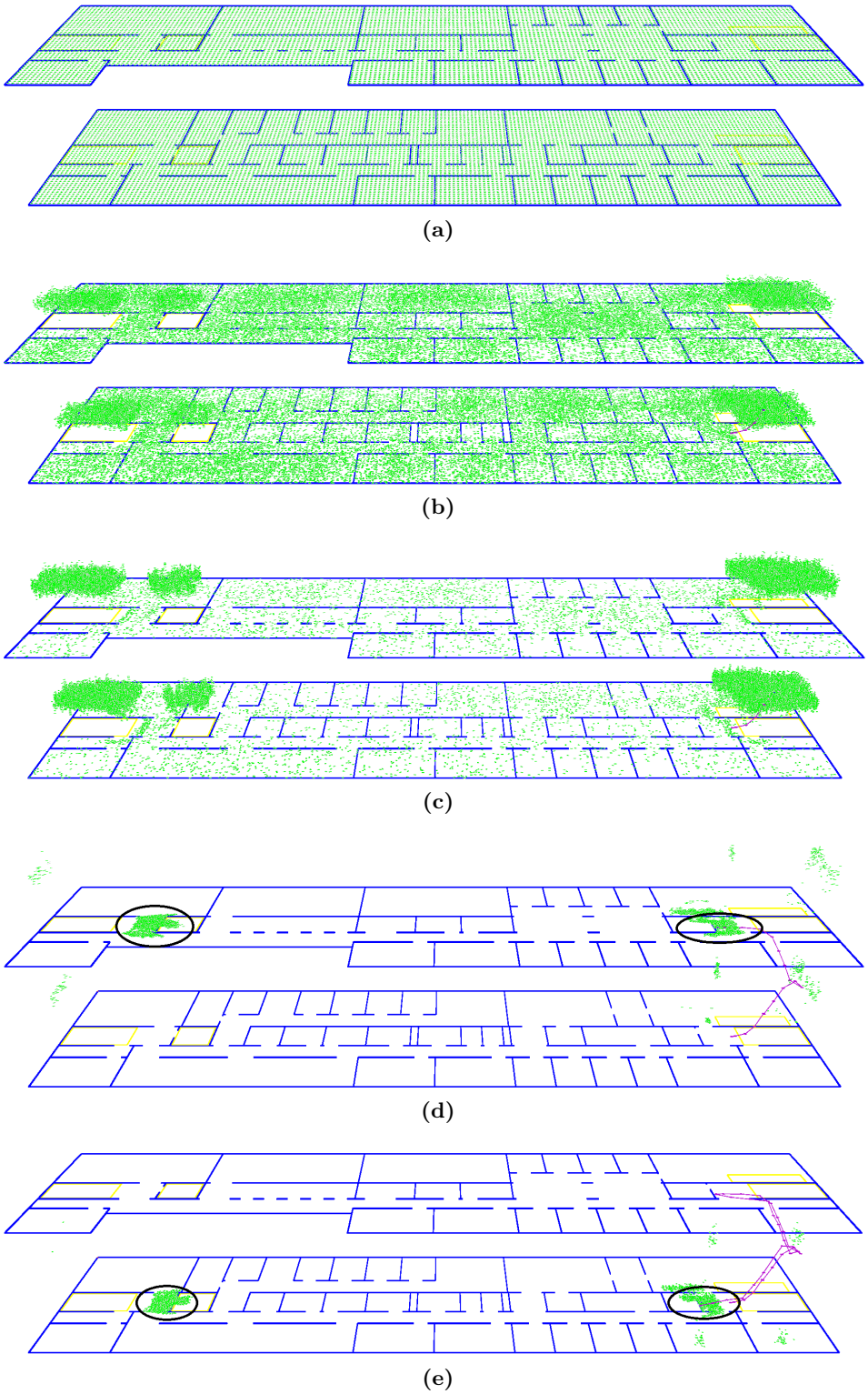


Abbildung 7.9 Das Verhalten des Systems bei unbekannter Startposition und der Verwendung einer Treppe. (a) zeigt die neu initialisierten Partikel. In (b) und (c) befindet sich die Person auf der zweiten beziehungsweise dritten Treppenstufe. In (d) wurde das obere Ende der Treppe erreicht, es sind beinahe keine Partikel mehr in der untereren Etage vorhanden. In (e) befindet sich die Person wieder im Erdgeschoss, nun ist die obere Etage frei von Partikeln. Die Position konnte auf zwei Orte eingegrenzt werden.

Obwohl die Position also nicht auf einen Ort eingegrenzt werden konnte ist so zumindest sicher auf welcher Etage sich die Person befindet. Würde sich die Person nun weiter durch das Gebäude bewegen würden sich früher oder später die Partikel einer Seite durch eine Wand bewegen und aussortiert werden.

Kapitel 8

Fazit und Ausblick

Ziel der vorliegenden Arbeit war es, die Position einer Personen innerhalb eines bekannten Gebäudes zu lokalisieren, deren Bewegung zu verfolgen und diese zu visualisieren. Dazu wurde ein Gesamtsystem entwickelt welches die Sensordaten einer am Fuß der Person angebrachten inertialen Messeinheit verarbeitet um die Bewegungen der Person zu verfolgen. Außerdem wird eine Gebäudekarte verwendet um die Position der Person festzustellen und die Ergebnisse der Bewegungsverfolgung zu verbessern. Zudem wurde ein Schritterkenner implementiert. Dieser erlaubt es Annahmen über den Zustand der Messeinheit zu machen, welche von dieser selbst nicht gemessen werden können. Diese Annahmen helfen dabei den starken Ungenauigkeiten der Messungen entgegenzuwirken und machen somit die Funktion des Gesamtsystems erst möglich.

Wie in der Evaluation in Kapitel 7 gezeigt, liegt die Abweichung der vom System geschätzten Position von der tatsächlichen Position in den meisten Fällen deutlich unter einem Meter. Zudem ist auch die Verwendung von Treppen und Aufzügen ohne Weiteres möglich. Es lässt sich also sagen, dass die Ziele der Arbeit erreicht wurden. Trotzdem lässt sich die Genauigkeit des System weiter verbessern. So sind inertiale Messeinheiten erhältlich, die wesentlich genauere Messungen liefern. Durch diese würde sich insbesondere die Schätzung bei Verwendung von Treppen und Aufzügen verbessern da hier weniger Annahmen über den Zustand der Messeinheit getroffen werden können und den Messungen somit mehr Gewicht zufällt.

Eine weitere Möglichkeit zur Verbesserung des Systems liegt darin, einen besseren Ansatz für das Erkennen von Schritten beziehungsweise von Standphasen zu finden. Der für diese Arbeit implementierte Schritterkenner funktioniert nur dann verlässlich, wenn sich die Person in einem normalen Tempo bewegt. Je schneller sich die Person bewegt, desto weniger gut funktioniert die Erkennung. Dies ist insbesondere dann ein Problem wenn die Person, zum Beispiel in einem Notfall, rennt. In diesem Fall ist davon auszugehen, dass keine Schritte erkannt werden. Hier müsste also ein robusterer Ansatz entwickelt werden. Dies könnte zum Beispiel im Rahmen einer darauf ausgerichteten Bachelor- oder Masterarbeit geschehen.

Um das System in der Praxis verwendbar zu machen, müsste außerdem dafür gesorgt werden,

dass die gemessenen Daten über Funk, W-Lan oder ähnlichem an einen externen Rechnen gesendet werden. Derzeit muss die Messeinheit direkt über ein Kabel an ein Notebook angeschlossen sein, auf dem die entwickelte Software läuft. Außerdem währe eine automatische Generierung der Gebäudekartendaten aus einem Bauplan von Vorteil. Obwohl dies nur einmalig pro Gebäude geschehen muss ist dieser Prozess derzeit sehr zeitaufwändig, insbesondere wenn alle oder viele Gebäude einer großen Stadt eingegeben werden sollen.

Literaturverzeichnis

[1] Beauregard, Stéphane: A helmet-mounted pedestrian dead reckoning system. In: *Applied Wearable Computing (IFAWC), 2006 3rd International Forum on* VDE, 2006, S. 1–11

[2] Beauregard, Stéphane: *Infrastructureless pedestrian positioning*, Universität Bremen, Diss., 2009

[3] Douc, R. ; Cappe, O.: Comparison of resampling schemes for particle filtering. In: *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, 2005. – ISSN 1845–5921, S. 64–69

[4] Doucet, Arnaud ; Godsill, Simon ; Andrieu, Christophe: On sequential Monte Carlo sampling methods for Bayesian filtering. In: *Statistics and Computing* 10 (2000), Nr. 3, 197–208. http://dx.doi.org/10.1023/A:1008935410038. – DOI 10.1023/A:1008935410038. – ISSN 1573–1375

[5] Fox, Dieter ; Hightower, Jeffrey ; Kauz, Henry ; Liao, Lin ; Patterson, Don: Bayesian techniques for location estimation. In: *Proceedings of the 2003 workshop on location-aware computing* Citeseer, 2003, S. 16–18

[6] Foxlin, E.: Pedestrian tracking with shoe-mounted inertial sensors. In: *Computer Graphics and Applications, IEEE* 25 (2005), Nov, Nr. 6, S. 38–46. http://dx.doi.org/10.1109/MCG.2005.140. – DOI 10.1109/MCG.2005.140. – ISSN 0272–1716

[7] Frese, Udo ; Schöder, Lutz: Skript zur Vorlesung Theorie der Sensorfusion / Universität Bremen. Version: Januar 2015. https://svn-agbkb.informatik.uni-bremen.de/ufrese/teaching/tds/skript/pdf/tds14skript.pdf. 2015. – Forschungsbericht

[8] Harle, Robert: A survey of indoor inertial positioning systems for pedestrians. In: *Communications Surveys & Tutorials, IEEE* 15 (2013), Nr. 3, S. 1281–1293

[9] Hertzberg, Christoph ; Wagner, René ; Frese, Udo ; Schröder, Lutz: Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. In: *Information Fusion* 14 (2013), Nr. 1, S. 57–77

[10] Hightower, Jeffrey ; Borriello, Gaetano: Particle filters for location estimation in ubiquitous computing: A case study. In: *UbiComp 2004: Ubiquitous Computing*. Springer, 2004, S. 88–106

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

[12] Kalman, Rudolph E.: A new approach to linear filtering and prediction problems. In: *Journal of Fluids Engineering* 82 (1960), Nr. 1, S. 35–45

[13] Krach, Bernhard ; Roberston, Patrick: Cascaded estimation architecture for integration of foot-mounted inertial sensors. In: *Position, Location and Navigation Symposium, 2008 IEEE/ION* IEEE, 2008, S. 112–119

[14] Laue, Tim ; Röfer, Thomas: Pose Extraction from Sample Sets in Robot Self-Localization-A Comparison and a Novel Approach. In: *ECMR*, 2009, S. 283–288

[15] Laue, Tim ; Spiess, Kai ; Röfer, Thomas: SimRobot – A General Physical Robot Simulator and Its Application in RoboCup. Version: 2006. `http://dx.doi.org/10.1007/11780519_16`. In: Bredenfeld, Ansgar (Hrsg.) ; Jacoff, Adam (Hrsg.) ; Noda, Itsuki (Hrsg.) ; Takahashi, Yasutake (Hrsg.): *RoboCup 2005: Robot Soccer World Cup IX*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006. – ISBN 978–3–540–35438–3, 173–183

[16] Skog, Isaac ; Händel, Peter: Calibration of a MEMS inertial measurement unit. In: *XVII IMEKO World Congress*, 2006, S. 1–6

[17] Stolpmann, Andreas: *Ein autonomer Trainer für Fußball spielende Roboter*, Universität Bremen, Bachelorarbeit, 2014

[18] Thrun, Sebastian ; Burgard, Wolfram ; Fox, Dieter: *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. The MIT Press, 2005 (Intelligent robotics and autonomous agents). – ISBN 0262201623

[19] Tsogias, Alexis: *Laufen für den NAO-Roboter mittels Zero-Moment Point Preview Control*, Universität Bremen, Masterarbeit, 2016

[20] Wenk, Felix ; Frese, Udo: Posture from motion. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on* IEEE, 2015, S. 280–285

[21] Widyawan ; Klepal, Beauregard S. Martin: A Backtracking Particle Filter for Fusing Building Plans with PDR Displacement Estimate. In: *Proceedings of the 5th Workshop on Positioning, Navigation and Communication (WPNC 2008)*, 2008, S. 207–212

[22] Wong, Charence ; Zhang, Zhi-Qiang ; Lo, Benny ; Yang, Guang-Zhong: Wearable Sensing for Solid Biomechanics: A Review. In: *Sensors Journal, IEEE* 15 (2015), Nr. 5, S. 2747–2760

[23] Woodman, Oliver: *Pedestrian Localisation for Indoor Environments*, University of Cambridge, Computer Laboratory, Diss., September 2010. `https://www.cl.cam.ac.uk/research/dtg/www/files/publications/public/abr28/ojw28_thesis.pdf`