

# **CBS**

## **INTERNATIONAL BUSINESS SCHOOL**

### **Indoor-Navigation mit Inertial Measurement Units mobiler Endgeräte**

Portfolio als Hausarbeit für "Fallstudie SEuSI"  
Sommersemester 2024  
Dozent: Prof. Dr. Alexander Rachmann

Lucas Freier, Maximilian Fröhlich, Daniel Gilbers, Eric Lepp  
BSC WI 22W D  
Immatrikulationsnummern: 2201416, 2201387, 2201318, 2201349

## Inhaltsverzeichnis

Abbildungsverzeichnis .....	IV
Tabellenverzeichnis.....	V
Abkürzungsverzeichnis .....	VI
1 Einleitung .....	1
2 Inertial Measurement Unit in Smartphones .....	3
2.1 Beschleunigungsmesser .....	4
2.2 Gyroskop .....	6
2.3 Magnetometer .....	8
3 Methoden zur Fehlerkompensation und -korrektur.....	10
3.1 Sensorfusion .....	10
3.2 Rauschunterdrückung in Beschleunigungsmessern.....	10
3.3 Kalibrierungsmethoden in Beschleunigungssensoren .....	11
3.4 Drift und Bias-Kompensation.....	12
3.5 Vermeidung von Störungen in Magnetometern .....	12
4 Algorithmen zur Sensordatenverarbeitung.....	14
4.1 Kalman-Filter.....	14
4.2 Pedestrian Dead Reckoning.....	16
4.3 Madgwick-Algorithmus.....	17
5 Anforderungen an eine Indoor-Navigations-App .....	19
5.1 Qualitätsmerkmale nach internationaler Norm.....	19
5.2 Grundlegende Anforderungen an Navigationssysteme .....	20
5.3 Benutzererwartungen durch etablierte Standards des Marktes.....	22
5.4 Zielsetzung des Prototyps .....	24
6 Konzept .....	27
6.1 Modellierung .....	27
6.1.1 Ablaufdiagramm.....	27

6.1.2	Klassendiagramm .....	31
6.2	Design .....	34
7	Umsetzung .....	37
7.1	Softwarearchitektur .....	37
7.2	Speicherung der Daten .....	38
7.3	Auslesen und Umrechnen der Sensordaten .....	39
7.3.1	Kalman-Filter Implementierung .....	40
7.3.2	PDR-Implementierung .....	41
8	Fazit .....	43
	Literaturverzeichnis .....	44
	Anhang .....	49
	KI-Tools & KI-Nutzung .....	51

## Abbildungsverzeichnis

Abbildung 1: Aufbau eines MEMS-Beschleunigungssensors. ....	4
Abbildung 2: Prinzip eines MEMS-Gyroskops. ....	7
Abbildung 3: Illustration of a ferromagnetic MEMS magnetometer design. ....	9
Abbildung 4: Darstellung des Kalman-Prozesses. ....	15
Abbildung 5: Product quality model nach ISO/IEC 25010.....	20
Abbildung 6: Ranking der Top 12 beliebtesten Smartphone-Apps in den USA nach Reichweite im Oktober 2023.....	23
Abbildung 7: Ablaufdiagramm der Indoor-Navigation-App. ....	28
Abbildung 8: Klassendiagramm – L.Control. ....	31
Abbildung 9: Klassendiagramm – Node & Edge.....	32
Abbildung 10: Klassendiagramm – Produkt. ....	33
Abbildung 11: Klassendiagramm – Astar. ....	34
Abbildung 12: Designkonzept der Anwendung. ....	35
Abbildung 13: Paketdiagramm der Anwendung. ....	38
Abbildung 14: Beispiel für JSON-Objekte des Graphen und der Produkte.....	38
Abbildung 15: Code der Funktion requestSensors(). ....	39
Abbildung 16: Code der Funktion handleMotion(). ....	39
Abbildung 17: Code der Funktion handleOrientation(). ....	40
Abbildung 18: Code der Funktion kFilter(). ....	40
Abbildung 19: Code der Funktion detectPeak(). ....	41
Abbildung 20: Code der Funktion updatePosition(). ....	42
Abbildung 21: Code der Funktion calculatePosition(). ....	42

**Tabellenverzeichnis**

Tabelle 1: Priorität grundlegender Leistungsparameter von Navigationssystemen.....	21
Tabelle 2: Anforderungsspezifikation der Indoor-Navigations-App. ....	24
Tabelle 3: Lösungsspezifikation der Indoor-Navigations-App. ....	25

**Abkürzungsverzeichnis**

IMU	Inertial Measurement Unit
MEMS	Mikro-elektro-mechanisches System
PDR	Pedestrian Dead Reckoning
TTFF	Time To First Fix
UI	User Interface

# 1 Einleitung

Diese Arbeit entstand im Rahmen der Entwicklung einer Indoor-Navigations-App für die toom Baumarkt GmbH. Die Idee zu dieser Fallstudie entstand aus der Notwendigkeit, eine genaue Positionsbestimmung innerhalb von Gebäuden zu ermöglichen, in denen GPS-Signale nicht zuverlässig empfangen werden können. Eine solche Indoor-Navigationslösung ist insbesondere in großen Räumen wie Baumärkten notwendig, um den Nutzern eine effiziente Orientierung zu bieten. Der Einsatz von Inertial Measurement Units (IMUs) in Smartphones spielt dabei eine zentrale Rolle, da diese Bewegungsdaten und die Orientierung des Gerätes erfassen können.

Die zentrale Forschungsfrage dieser Fallstudie lautet daher: Wie kann Indoor-Navigation mit Inertial Measurement Units von mobilen Endgeräten realisiert werden? Ziel der Arbeit ist es, die technischen Grundlagen der IMU-Sensorik darzustellen und die wesentlichen Methoden zur Fehlerkompensation sowie die verwendeten Algorithmen zur Sensordatenverarbeitung zu untersuchen. Der Fokus liegt dabei auf zentralen Algorithmen wie dem Kalman-Filter und dem Pedestrian Dead Reckoning (PDR).

Die IMU besteht aus drei Hauptkomponenten: einem Beschleunigungssensor, der die lineare Beschleunigung in drei Achsen misst, einem Gyroskop, das die Winkelgeschwindigkeit misst, und einem Magnetometer, das die Richtung im Raum bestimmt. Diese Sensoren arbeiten zusammen, um die Bewegungen und die Orientierung des Gerätes kontinuierlich zu überwachen. Aufgrund ihrer geringen Größe und der niedrigen Herstellungskosten sind diese Sensoren in vielen mobilen Geräten wie Smartphones verbaut.

Ein Nachteil dieser Sensoren ist jedoch, dass sie anfällig für Messfehler wie Rauschen, Drift und Bias sind, die die Genauigkeit der Positionsbestimmung beeinträchtigen können. Diese Fehlerquellen zu minimieren ist ein wesentlicher Teil dieser Arbeit. Zur Verbesserung der Datenqualität werden Algorithmen wie der Kalman-Filter und PDR eingesetzt, um die Sensordaten zu verarbeiten und Fehler zu kompensieren. Das Ziel ist es, ein Konzept für ein mobiles System zu entwickeln, das ausschließlich auf den Daten der IMUs basiert und dem Nutzer die Position und Orientierung innerhalb eines Gebäudes liefert.

Die Arbeit ist wie folgt aufgebaut: Kapitel 2 behandelt die theoretischen Grundlagen, insbesondere die Funktionsweise von IMUs. In Kapitel 3 werden Kalibrierungsansätze zur Minimierung von Messfehlern vorgestellt. Kapitel 4 gibt einen Überblick über die

Algorithmen zur Sensordatenverarbeitung. Die Anforderungen an eine Indoor-Navigations-App werden in Kapitel 5 erläutert und das entsprechende Konzept wird in Kapitel 6 beschrieben. Kapitel 7 ist der Umsetzung des Konzeptes gewidmet. In Kapitel 8 wird ein Fazit gezogen und ein Ausblick auf weiteren Forschungsbedarf gegeben.



## 2 Inertial Measurement Unit in Smartphones

Eine Inertial Measurement Unit (IMU) bezeichnet ein elektronisches Gerät, welches die spezifische Kraft, Winkelgeschwindigkeit sowie das Magnetfeld eines Objekts misst, in dem es sich befindet. Die genannten Messungen werden in der Regel durch drei Hauptkomponenten durchgeführt:

- **Beschleunigungsmesser:** Misst die lineare Beschleunigung des Geräts in drei Achsen. Diese Sensoren nutzen die Trägheitskraft zur Messung und sind für die Erkennung von Bewegungen und Lageänderungen verantwortlich.
- **Gyroskop:** Misst die Winkelgeschwindigkeit oder die Drehgeschwindigkeit des Geräts. Es liefert Informationen über die Drehbewegungen in drei Achsen und hilft bei der Stabilisierung und Orientierung des Geräts.
- **Magnetometer:** Misst das Magnetfeld und kann als digitaler Kompass verwendet werden. Es hilft, die absolute Richtung im Raum zu bestimmen, indem es das Erdmagnetfeld misst.

Die Trägheitsnavigation basiert auf der Integration der gemessenen Beschleunigungen und Winkelgeschwindigkeiten, um die Position, Geschwindigkeit und Orientierung des Geräts zu bestimmen.<sup>1</sup>

In modernen Smartphones sind IMUs von entscheidender Bedeutung für eine Vielzahl von Funktionen, die das Nutzererlebnis erheblich verbessern. IMUs sind auch für die Bereiche Navigation und Ortung von entscheidender Bedeutung. Sie dienen in erster Linie als Ergänzung zu GPS-Systemen, indem sie Bewegungsdaten auch in Situationen liefern, in denen GPS-Signale schwach sind oder nicht empfangen werden können. Dies erweist sich insbesondere in urbanen Gebieten mit hoher Bebauungsdichte sowie in geschlossenen Räumen als vorteilhaft, da dort GPS-Signale häufig nicht zuverlässig empfangen werden können.<sup>2</sup>

Die kontinuierliche Bereitstellung von Bewegungsdaten ermöglicht eine genauere Positionierung sowie eine flüssigere Navigation. Auch für die Bewegungs- und Gestenerkennung in Smartphones sind IMUs unverzichtbar. Sie ermöglichen auch Anwendungen wie Fitness-Tracker, die Aktivitäten wie das Zählen von Schritten, die Messung des Kalorienverbrauchs und die Überwachung anderer körperlicher Aktivitäten

---

<sup>1</sup> Vgl. Michaelsen 2018, S. 3 ff.

<sup>2</sup> Vgl. Möllmann 2014, S. 3 f.

ermöglichen. Ein weiteres bedeutendes Anwendungsgebiet von IMUs liegt in der Stabilisierung von Bild- und Videoaufnahmen. Die von IMUs erfassten Daten werden in Smartphones verwendet, um unerwünschte Handbewegungen und Vibrationen zu detektieren und zu kompensieren. Dies führt zu einer Verbesserung der Bildqualität und -stabilität, insbesondere bei der Aufnahme von Videos oder Fotos in Bewegung.

IMUs bestehen aus mehreren Sensoren, die gemeinsam Informationen über die Bewegung und räumliche Lage eines Geräts bereitstellen. Typischerweise umfassen sie Beschleunigungssensoren, Gyroskope und häufig auch Magnetometer, welche die lineare Beschleunigung, die Drehgeschwindigkeit sowie das Magnetfeld eines Objekts messen.<sup>3</sup>

Hierbei spielen Mikro-elektro-mechanische Systeme (MEMS) eine zentrale Rolle. MEMS-Technologie ermöglicht die Miniaturisierung und Integration dieser Sensoren auf einem winzigen Chip, wodurch IMUs kompakt und kostengünstig in Geräten wie Smartphones, Wearables und Fahrzeugen eingesetzt werden können.<sup>4</sup>

Im Folgenden wird die Funktionsweise der MEMS-Beschleunigungssensoren, Gyroskope und Magnetometer detailliert dargestellt, wobei auch die damit verbundenen technischen Herausforderungen erörtert werden.

## 2.1 Beschleunigungsmesser

Ein MEMS-Beschleunigungsmesser misst die lineare Beschleunigung eines Objekts in drei Achsen. Diese Sensoren basieren auf mikroelektromechanischen Systemen, die kleine bewegliche Strukturen enthalten, die auf Kräfte reagieren und diese in elektrische Signale umwandeln.<sup>5</sup> Der grundlegende Aufbau eines solchen Sensors, wie in der Abbildung dargestellt, besteht aus mehreren wesentlichen Komponenten: Abbildung 1: Aufbau eines MEMS-Beschleunigungssensors.

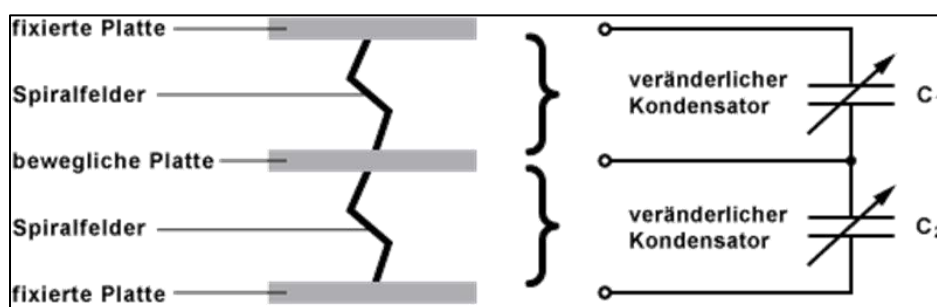


Abbildung 1: Aufbau eines MEMS-Beschleunigungssensors.

<sup>3</sup> Vgl. Möllmann 2014, S. 3.

<sup>4</sup> Vgl. Köster 2018, S. 10 ff.

<sup>5</sup> Vgl. Möllmann 2014, S. 24 f.

- **Bewegliche Platte (Massedämpfer):** Diese Platte bewegt sich in Reaktion auf die an den Sensor angelegte Beschleunigung. Die Bewegung der Platte verändert den Abstand zwischen den feststehenden und beweglichen Platten.
- **Feststehende Platten:** Diese Platten sind an den Enden des Systems befestigt und bleiben während der Beschleunigung stationär.
- **Spiralfedern:** Die bewegliche Platte ist durch Spiralfedern mit den feststehenden Platten verbunden. Diese Federn stellen sicher, dass die Platte in ihre Ausgangsposition zurückkehrt, sobald die Beschleunigung aufhört.
- **Kapazitive Änderung:** Die Bewegung der beweglichen Platte führt zu einer Änderung der Kapazität der Kondensatoren C1 und C2. Diese kapazitive Änderung wird genutzt, um die Größe und Richtung der Beschleunigung zu bestimmen. Eine Bewegung der Platte nach oben oder unten verändert die Kapazitäten C1 und C2, wodurch die Beschleunigung entlang der entsprechenden Achse ermittelt werden kann.

Diese kapazitive Methode der Messung ist besonders effektiv, da sie eine hohe Empfindlichkeit und eine schnelle Reaktionszeit ermöglicht. Das elektrische Signal, das durch die Kapazitätsänderung entsteht, wird dann in ein digitales Signal umgewandelt, das zur weiteren Verarbeitung verwendet wird.<sup>6</sup>

Rauschen und Kalibrierungsfehler stellen bedeutende Herausforderungen bei der Nutzung von MEMS-Beschleunigungsmessern dar. Diese Fehlerquellen können die Genauigkeit und Zuverlässigkeit der gemessenen Daten erheblich beeinträchtigen. Rauschen bezieht sich auf zufällige Schwankungen im Ausgangssignal des Beschleunigungsmessers, die nicht mit der tatsächlichen Bewegung korrelieren. Diese Schwankungen können durch verschiedene interne und externe Faktoren verursacht werden:<sup>7</sup>

- **Thermisches Rauschen:** Wird durch die thermische Bewegung der Atome im Sensor verursacht.
- **Elektronisches Rauschen:** Entsteht durch elektrische Interferenzen innerhalb der Schaltung des Sensors.
- **Mechanisches Rauschen:** Kann durch Vibrationen und andere mechanische Einflüsse auf den Sensor hervorgerufen werden.

---

<sup>6</sup> Vgl. Schnabel 2017.

<sup>7</sup> Vgl. Külls 2016, S. 66 ff.

Das Rauschen kann die Messgenauigkeit erheblich beeinträchtigen, insbesondere bei sehr kleinen oder langsamen Bewegungen. Es kann zu einer Verschlechterung des Signal-Rausch-Verhältnisses führen, was die präzise Erfassung der Bewegung erschwert.<sup>8</sup>

Kalibrierungsfehler entstehen durch Ungenauigkeiten bei der initialen Einstellung des Sensors und können systematische Abweichungen im Ausgangssignal verursachen. Zu den häufigsten Ursachen gehören:

- **Fertigungstoleranzen:** Abweichungen in der Produktion können dazu führen, dass die Sensoren nicht exakt kalibriert sind.
- **Alterung:** Mit der Zeit können sich die Eigenschaften des Sensors ändern, was zu Drift und Bias führt.
- **Umgebungseinflüsse:** Veränderungen in der Temperatur, Feuchtigkeit und anderen Umgebungsbedingungen können die Kalibrierung beeinflussen.

Kalibrierungsfehler führen zu systematischen Fehlern im Ausgangssignal, was bedeutet, dass die gemessenen Werte konstant um einen bestimmten Betrag vom tatsächlichen Wert abweichen. Diese Abweichungen können in vielen Anwendungen problematisch sein, insbesondere in präzisen Messsystemen wie in der Navigation oder der Bewegungsverfolgung.

## 2.2 Gyroskop

Ein MEMS-Gyroskop misst die Winkelgeschwindigkeit eines Objekts, indem es die Coriolis-Kraft nutzt, die auf eine schwingende Masse wirkt, wenn das Gerät rotiert. In der Abbildung bewegt sich die Resonanzmasse entlang der Bewegungsrichtung der Masse, während sie durch Federn in Position gehalten wird. Wenn das Gerät dreht, erzeugt die Coriolis-Kraft eine Verschiebung der Resonanzmasse, die von den Coriolis-Erfassungssensoren erfasst wird. Diese Verschiebung wird proportional zur Drehgeschwindigkeit in ein elektrisches Signal umgewandelt, das für die genaue Bewegungs- und Drehungsdatenerfassung verwendet wird.<sup>9</sup> Abbildung 2: Prinzip eines MEMS-Gyroscopes.

---

<sup>8</sup> Vgl. Külls 2016, S. 66 ff.

<sup>9</sup> Vgl. Watson 2016.

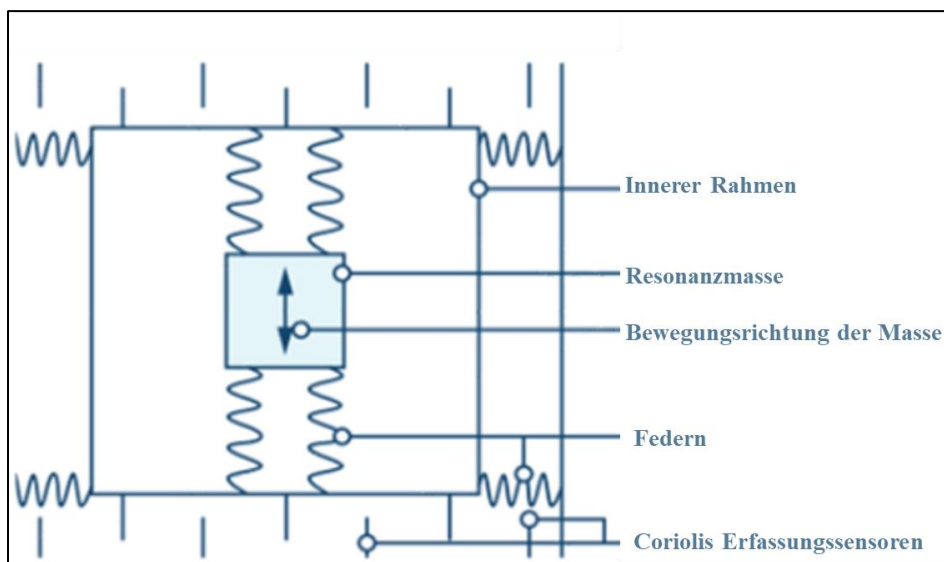


Abbildung 2: Prinzip eines MEMS-Gyroskops.

Drift und Bias sind zwei der wichtigsten Fehlerarten, die bei MEMS-Gyroskopen auftreten und die Genauigkeit der Messungen erheblich beeinträchtigen können.

- **Drift:** Drift bezieht sich auf die allmähliche Abweichung des Gyroskopsignals vom tatsächlichen Wert über die Zeit. Diese Abweichung entsteht aufgrund interner Effekte, wie mechanischen Spannungen und Änderungen der Umgebungstemperatur. Der Drift führt dazu, dass die gemessene Winkelgeschwindigkeit im Laufe der Zeit immer ungenauer wird, was besonders problematisch bei Anwendungen ist, die eine hohe Präzision erfordern, wie z.B. Indoor-Navigation.<sup>10</sup>
- **Bias:** Bias ist ein konstanter, systematischer Fehler im Ausgangssignal des Gyroskops. Dieser Fehler kann durch Fertigungstoleranzen, mechanische Spannungen oder elektrische Einflüsse entstehen. Der Bias führt dazu, dass das Gyroskop eine konstante Winkelgeschwindigkeit anzeigt, selbst wenn keine tatsächliche Drehbewegung stattfindet. Das bedeutet, dass das Gyroskop einen scheinbaren Dreheffekt registriert, auch wenn das Gerät in Ruhe ist. Diese falsche Messung resultiert in einer kontinuierlichen Fehlausgabe, die die tatsächliche Bewegung des Geräts verzerrt.<sup>11</sup>

<sup>10</sup> Vgl. Michaelsen 2018, S. 30 ff.

<sup>11</sup> Vgl. Michaelsen 2018, S. 30 ff.

## 2.3 Magnetometer

Ein MEMS-Magnetometer misst die Stärke und Richtung des Magnetfelds, in dem es sich befindet. Diese Sensoren werden häufig in Smartphones und anderen mobilen Geräten verwendet, um die Ausrichtung des Geräts in Bezug auf das Erdmagnetfeld zu bestimmen. Das Magnetometer funktioniert wie ein digitaler Kompass und hilft bei der Navigation und Orientierung. MEMS-Magnetometer basieren auf verschiedenen physikalischen Prinzipien wie dem Hall-Effekt, dem magnetoresistiven Effekt oder dem Induktionsprinzip. Beim Hall-Effekt erzeugt ein Magnetfeld in einem leitenden Material eine Spannung, die proportional zur Feldstärke ist. Magnetoresistive Effekte beschreiben die Änderung des elektrischen Widerstands eines Materials, wenn es einem Magnetfeld ausgesetzt wird. Das Induktionsprinzip beruht darauf, dass in einer Spule eine Spannung induziert wird, wenn sich das Magnetfeld in ihrer Nähe ändert. Diese verschiedenen Effekte ermöglichen die präzise Messung von Magnetfeldern in kompakten und mobilen Geräten.<sup>12</sup>

Magnetometer sind besonders nützlich in Kombination mit Beschleunigungsmessern und Gyroskopen, da sie die absolute Richtung im Raum liefern können, während Beschleunigungsmesser und Gyroskope relative Bewegungen messen.

Abbildung 3: Illustration of a ferromagnetic MEMS magnetometer design. zeigt die Funktionsweise eines Magnetometers, dessen Messprinzip auf der magnetischen Torsion beruht. Ein Magnet, der sich in einem äußeren Magnetfeld befindet, erfährt ein magnetisches Drehmoment, das ihn zu drehen versucht. Der Magnet ist an einem Torsionsstab befestigt, der auf dieses Drehmoment mit einem mechanischen Drehmoment reagiert. Unter Torsion versteht man die Verdrehung eines Objekts, in diesem Fall des Balkens, durch die Einwirkung von Kräften. Das System erreicht einen Gleichgewichtszustand, wenn das mechanische Drehmoment das magnetische Drehmoment ausgleicht. Die resultierende Torsion des Balkens wird gemessen und zur Berechnung der Stärke und Richtung des äußeren Magnetfeldes verwendet.<sup>13</sup>

---

<sup>12</sup> Vgl. Michaelsen 2018, S. 16 f. und Möllmann 2014, S. 26 ff.

<sup>13</sup> Vgl. Vasquez 2004, S. 28 ff.

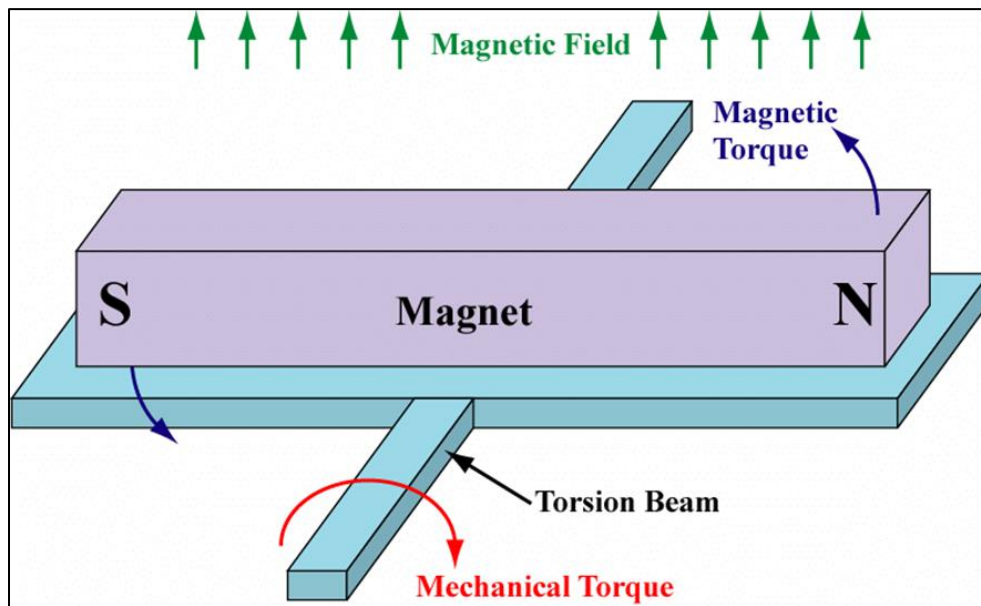


Abbildung 3: Illustration of a ferromagnetic MEMS magnetometer design.

Magnetometer sind anfällig für verschiedene Störungen und Verzerrungen, die ihre Genauigkeit beeinträchtigen können. Diese Störungen können durch externe magnetische Felder, elektrische Ströme oder ferromagnetische Materialien in der Nähe des Sensors verursacht werden. Zu den häufigsten Problemen gehören:<sup>14</sup>

- **Weichmagnetische Verzerrungen:** Diese entstehen durch nahegelegene Objekte aus weichem Eisen oder anderen magnetischen Materialien, die das Magnetfeld verzerren.
- **Hartmagnetische Verzerrungen:** Diese werden durch permanent magnetisierte Materialien in der Nähe des Sensors verursacht und führen zu einer konstanten Verschiebung im gemessenen Magnetfeld.
- **Elektromagnetische Interferenzen:** Diese werden durch elektrische Ströme und elektronische Geräte verursacht, die ein störendes magnetisches Feld erzeugen.<sup>15</sup>

<sup>14</sup> Vgl. Möllmann 2014, S. 26 ff.

<sup>15</sup> Vgl. Vasquez 2004, S. 28 ff.

### **3 Methoden zur Fehlerkompensation und -korrektur**

In diesem Abschnitt werden spezifische Methoden und Techniken zur Kompensation und Korrektur, der in den vorherigen Abschnitten genannten Herausforderungen behandelt. Diese Methoden zielen darauf ab, die Genauigkeit und Zuverlässigkeit der Sensordaten zu erhöhen, indem sie Rauschen, Kalibrierungsfehler, Drift, Bias sowie Störungen und Verzerrungen minimieren. Es ist jedoch wichtig zu beachten, dass die nachfolgend beschriebenen Lösungen als theoretische Ansätze und Möglichkeiten zu verstehen sind.

#### **3.1 Sensorfusion**

Sensorfusion bezieht sich auf die Integration und Kombination von Daten aus mehreren Sensoren, um eine genauere und robustere Schätzung der Messgrößen zu erhalten. In Smartphones wird Sensorfusion verwendet, um die Daten von Beschleunigungsmessern, Gyroskopen und Magnetometern zu kombinieren. Diese Sensoren liefern jeweils unterschiedliche, aber komplementäre Informationen über die Bewegungen und die Orientierung des Geräts. Die Fusion dieser Daten ermöglicht es, die Stärken jedes Sensors zu nutzen und zeitgleich die individuellen Schwächen zu kompensieren. Die Bedeutung der Sensorfusion in Smartphones liegt in ihrer Fähigkeit, präzise und zuverlässige Bewegungs- und Positionsdaten zu liefern. Diese Daten sind entscheidend für eine Vielzahl von Anwendungen, darunter Navigation, Spiele, Augmented Reality und die Stabilisierung von Bildern und Videos. Durch die Kombination der Daten mehrerer Sensoren können Smartphones genauere Informationen über ihre Bewegung und Ausrichtung bereitstellen, was zu einer verbesserten Benutzererfahrung führt.<sup>16</sup>

#### **3.2 Rauschunterdrückung in Beschleunigungsmessern**

Rauschen stellt eine der signifikantesten Herausforderungen bei der Nutzung von MEMS-Beschleunigungsmessern dar, da es die Genauigkeit der erfassten Daten in erheblichem Maße beeinträchtigen kann. Um das Rauschen zu minimieren, werden diverse Signalverarbeitungstechniken eingesetzt, welche darauf abzielen, unerwünschte Störungen im Ausgangssignal zu unterdrücken. Eine weit verbreitete Methode zur Rauschunterdrückung ist der Einsatz von Tiefpassfiltern. Diese Filter lassen lediglich Signale mit niedrigen Frequenzen durch und blockieren höhere Frequenzen, die typischerweise als Rauschen auftreten. Der Einsatz von Tiefpassfiltern erlaubt eine

---

<sup>16</sup> Vgl. Michalsen 2018, S. 51 ff.



effektive Reduktion störender Hochfrequenzanteile, welche, wie in Kapitel 2.1 bereits erwähnt, durch thermisches, elektronisches oder mechanisches Rauschen verursacht werden. Das Resultat ist ein klareres und verlässliches Ausgangssignal, welches die tatsächliche Bewegung des Objekts präziser widerspiegelt. Des Weiteren kann neben der Filterung auch die Signalmittelwertbildung zur Rauschunterdrückung zum Einsatz kommen. Diese Technik basiert auf der Berechnung des Mittelwerts über eine Serie von Messungen. Da Rauschen in der Regel zufällig und unregelmäßig auftritt, führt die Mittelwertbildung zu einer Glättung der Schwankungen und somit zu einer Reduktion des Rauschens. Dies erweist sich insbesondere bei geringen oder langsamen Bewegungen als vorteilhaft, da hier das Signal-Rausch-Verhältnis von entscheidender Bedeutung ist. Die Kombination der genannten Techniken erlaubt eine signifikante Verringerung des Rauschens in MEMS-Beschleunigungsmessern, was zu einer Verbesserung der Messgenauigkeit und Zuverlässigkeit führt.<sup>17</sup>

### **3.3 Kalibrierungsmethoden in Beschleunigungssensoren**

Ein häufiges Problem bei der Verwendung von MEMS-Beschleunigungssensoren sind Kalibrierfehler, die durch verschiedene Faktoren wie Fertigungstoleranzen, Alterung der Sensoren oder Änderungen der Umgebungsbedingungen verursacht werden und systematische Abweichungen im Ausgangssignal verursachen. Diese Ungenauigkeiten können mit der Zeit die Zuverlässigkeit der Messungen beeinträchtigen.<sup>18</sup>

Um diese Fehler zu minimieren, werden verschiedene Kalibrierverfahren eingesetzt. Eine Möglichkeit ist die Einpunktkalibrierung, bei der der Sensor auf einen einzigen Referenzpunkt eingestellt wird, um dort eine optimale Messgenauigkeit zu erreichen. Eine genauere Methode ist die Mehrpunktkalibrierung, bei der der Sensor an mehreren Referenzpunkten kalibriert wird, um die Messgenauigkeit über den gesamten Betriebsbereich zu verbessern. Diese Methode ermöglicht die Anpassung des Sensors an unterschiedliche Umgebungsbedingungen und liefert somit konsistentere Ergebnisse.<sup>19</sup>

Darüber hinaus verfügen moderne MEMS-Sensoren häufig über Selbstkalibrierungsmechanismen, die während des Betriebs kontinuierlich Anpassungen vornehmen. Diese Mechanismen ermöglichen es den Sensoren, sich automatisch an Veränderungen wie Temperaturschwankungen oder den natürlichen Alterungsprozess

---

<sup>17</sup> Köster 2018, S. 64 ff.

<sup>18</sup> Michaelsen 2018, S. 35 ff.

<sup>19</sup> Michaelsen 2018, S. 35 ff.

anzupassen. Dadurch kann die Messgenauigkeit langfristig aufrechterhalten und Kalibrierfehler reduziert werden.

### 3.4 Drift und Bias-Kompensation

Drift und Bias sind häufige Fehler in MEMS-Sensoren, die die Genauigkeit der Messungen im Laufe der Zeit beeinträchtigen können. Um diese Fehler zu minimieren und die Messgenauigkeit aufrechtzuerhalten, werden verschiedene Algorithmen eingesetzt, darunter:

- **Temperaturkompensation:** Diese Methode minimiert die Auswirkungen von Temperaturschwankungen auf MEMS-Sensoren. Da Temperaturänderungen Drift und Bias verstärken können, überwachen Temperaturkompensationsalgorithmen kontinuierlich die Umgebungstemperatur und passen die Messungen entsprechend an. Ein Temperaturmodell, das auf Kalibrierdaten basiert, wird verwendet, um die Temperaturabhängigkeit der Sensoren zu korrigieren. Dadurch bleiben die Messungen auch bei schwankenden Temperaturen präzise.<sup>20</sup>
- **Automatische Rekalibrierung:** Diese Technik korrigiert Drift und Bias während des laufenden Betriebs des Geräts. Der Prozess sorgt dafür, dass die Sensoren auch über längere Zeiträume hinweg genaue Messungen liefern. Algorithmen überwachen kontinuierlich die Sensordaten und passen die Kalibrierungsparameter an, wenn systematische Abweichungen erkannt werden. Dies gewährleistet eine anhaltende Messgenauigkeit ohne manuelle Eingriffe.<sup>21</sup>

### 3.5 Vermeidung von Störungen in Magnetometern

Magnetometer sind besonders anfällig für Störungen durch elektromagnetische Interferenzen (EMI) und magnetische Verzerrungen, die die Messgenauigkeit erheblich beeinträchtigen können. Spezielle Techniken werden eingesetzt, um diese Störungen zu reduzieren und die Zuverlässigkeit der Messdaten zu erhöhen.

Eine häufig verwendete Maßnahme zur Reduzierung von EMI ist der Einsatz von Tiefpassfiltern. Diese Filter lassen nur niederfrequente Signale durch und blockieren hochfrequente Störsignale, die typischerweise von elektrischen Geräten oder drahtloser

---

<sup>20</sup> Vgl. Michaelsen 2018, S. 19.

<sup>21</sup> Vgl. Michaelsen 2018, S. 12.

Kommunikation in der Umgebung erzeugt werden. Tiefpassfilter helfen, externe Störquellen zu minimieren und das Signal-Rausch-Verhältnis zu verbessern.<sup>22</sup>

Neben der Filterung werden auch physikalische Abschirmungen eingesetzt. Dabei werden Materialien wie Mu-Metall oder Weicheisen verwendet, um das Magnetometer vor externen magnetischen Störungen zu schützen. Diese Materialien haben die Fähigkeit, Magnetfelder abzuschirmen oder umzulenken und verhindern so, dass unerwünschte Magnetfelder die Messungen des Magnetometers verfälschen. Zusätzlich zur Filterung und Abschirmung ist eine regelmäßige Kalibrierung des Magnetometers erforderlich, um die verbleibenden Störungen und Verzerrungen zu minimieren. Moderne Magnetometer verfügen häufig über automatische Kalibrieralgorithmen, die das Umgebungsfeld kontinuierlich überwachen und Anpassungen vornehmen, um magnetische Verzerrungen zu kompensieren. Diese Algorithmen können sowohl weichmagnetische als auch hartmagnetische Verzerrungen erkennen und kompensieren, was die Messgenauigkeit erheblich verbessert.<sup>23</sup>

---

<sup>22</sup> Vgl. Jahns 2013, S. 103 ff.

<sup>23</sup> Vgl. Jahns 2013, S. 103 ff. und S.117.

## 4 Algorithmen zur Sensordatenverarbeitung

Die Sensordatenverarbeitung ist zentraler Aspekt für die Positions- und Orientierungsschätzung in Indoor-Navigationssystemen, insbesondere in Umgebungen, in denen GPS-Signale unzuverlässig oder nicht verfügbar sind. Diese Systeme basieren auf den IMU-Daten. Die Herausforderung besteht darin, das Rauschen dieser Sensordaten zu filtern und gleichzeitig eine zuverlässige Schätzung über längere Zeiträume zu gewährleisten.

Ein wesentlicher Aspekt der Sensordatenverarbeitung ist die Sensorfusion, bei der Daten aus verschiedenen Quellen kombiniert werden, um eine genauere Schätzung zu erzielen. Die Algorithmen müssen Ungenauigkeiten in den einzelnen Datenquellen berücksichtigen und in nahezu Echtzeit eine Schätzung der Position liefern.

Die Genauigkeit der Schätzung hängt von der Bewegungsdynamik und der Qualität der Sensordaten ab. Die Algorithmen müssen in der Lage sein, Fehler wie Drift und Bias zu erkennen und zu korrigieren, um Abweichungen in der Positionsschätzung zu minimieren, wie im vorherigen Kapitel erläutert. Hierbei kommen häufig Fehlerzustandsmodelle zum Einsatz, die die Dynamik dieser Fehler explizit modellieren und in den Schätzprozess integrieren.

Darüber hinaus ist die Resistenz der Algorithmen gegenüber Änderungen in der Umgebung oder im Verhalten des Benutzers von Bedeutung. Algorithmen zur Sensordatenverarbeitung müssen in der Lage sein, auf plötzliche Änderungen, wie beispielsweise Variationen in der Gangart oder Magnetfeldstörungen, adäquat zu reagieren, ohne die Genauigkeit der Schätzungen zu beeinträchtigen.

In den folgenden Unterkapiteln werden drei Algorithmen zur Sensordatenverarbeitung vorgestellt, die zur Fusion der Daten dienen und darauf abzielen, auftretende Fehler zu minimieren.

### 4.1 Kalman-Filter

Das Grundprinzip des Kalman-Filters besteht aus zwei wesentlichen Schritten. Im Vorhersageschritt wird der zukünftige Zustand eines Systems auf Basis des aktuellen Zustands und des zugrunde liegenden Modells prognostiziert. Im darauffolgenden Aktualisierungsschritt wird diese Vorhersage durch die tatsächlichen Messungen überprüft und entsprechend korrigiert.

Der Kalman-Filter verwendet eine Reihe von mathematischen Operationen, um den Zustand eines dynamischen Systems kontinuierlich zu schätzen und zu aktualisieren.

Diese Operationen lassen sich in fünf Hauptschritte unterteilen: Zustandsvorhersage, Kovarianzvorhersage, Kalman-Verstärkung, Zustandsaktualisierung und Kovarianzaktualisierung.<sup>24</sup> (Vgl. Abbildung 4)

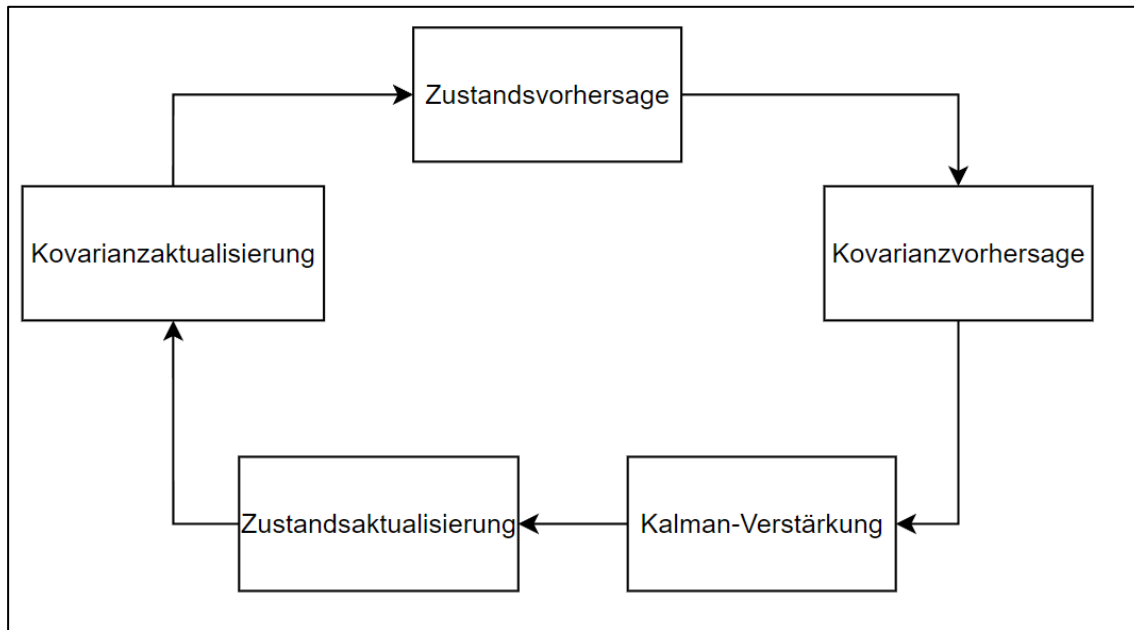


Abbildung 4: Darstellung des Kalman-Prozesses.<sup>25</sup>

Im Zustandsvorhersageschritt wird der zukünftige Zustand des Systems basierend auf dem aktuellen Zustand und einem Modell des Systems geschätzt. Dies geschieht, indem der aktuelle Zustand des Systems durch eine Gleichung transformiert wird, die das dynamische Verhalten des Systems beschreibt. Das Modell berücksichtigt dabei auch eventuelle externe Einflüsse oder Steuerungsvariablen.<sup>26</sup>

Parallel erfolgt die Kovarianzvorhersage, bei der die Schätzfehler der Zustandsvorhersage eingegrenzt wird. Schätzfehler geben an, wie sehr der geschätzte Zustand vom tatsächlichen Zustand abweichen könnte. Die resultierende Ungenauigkeit wird als Kovarianzmatrix dargestellt, welche Informationen über die Stärke der Abhängigkeit der verschiedenen Zustandsgrößen sowie die Genauigkeit der Vorhersage beinhaltet. In diesem Schritt findet zudem eine Berücksichtigung des Rauschens statt, welches durch die Schätzfehler des Modells bedingt ist.<sup>27</sup>

<sup>24</sup> Vgl. Welch / Bishop 2002, S. 5 ff.

<sup>25</sup> Vgl. Welch / Bishop 2002, S. 4 ff.

<sup>26</sup> Vgl. Welch / Bishop 2002, S. 4 ff.

<sup>27</sup> Vgl. hierzu und zu Folgenden Welch / Bishop 2002.

Der nächste Schritt ist die Berechnung der Kalman-Verstärkung. Diese Verstärkung ist ein Schlüsselfaktor, der bestimmt, wie stark die Schätzung des zukünftigen Zustands basierend auf neuen Messungen korrigiert werden soll. Sie stellt das optimale Gewicht dar, das den neuen Messungen im Vergleich zur Vorhersage zugeordnet wird. Dabei wird berücksichtigt, wie genau die Messungen im Vergleich zu der Vorhersage mit den Ungenauigkeiten sind.

Im Zustandsaktualisierungsschritt wird die Schätzung des Zustands durch die Berücksichtigung der neuen Messungen korrigiert. Die vorhergesagte Schätzung wird anhand der Abweichung zwischen den tatsächlich gemessenen Werten und den vorhergesagten Werten angepasst, wobei die Kalman-Verstärkung verwendet wird, um die optimale Korrektur zu berechnen.

Schließlich erfolgt die Kovarianzaktualisierung, bei der die Unsicherheit der neuen Schätzung angepasst wird. Die aktualisierte Kovarianzmatrix spiegelt nun die verringerte Ungenauigkeit wider, die sich aus der verbesserten Schätzung ergibt. Durch diesen Schritt wird der Kalman-Filter auf zukünftige Vorhersagen vorbereitet, indem die Schätzfehler neu kalibriert werden.

## 4.2 Pedestrian Dead Reckoning

Pedestrian Dead Reckoning (PDR) ist eine Technik zur Schätzung der Position eines sich bewegenden Objekts, in diesem Fall einer Person, basierend auf der Schrittzählung und der Erkennung der Bewegungsrichtung. Diese Methode erweist sich insbesondere in Innenräumen als nützlich, da GPS-Signale unzuverlässig sind oder nicht zur Verfügung stehen. Das Grundprinzip von PDR beruht auf der Nutzung von IMU-Daten, um Schritte zu erkennen und die Bewegungsrichtung zu bestimmen. Durch die Integration der Schrittlängen und Bewegungsrichtungen lässt sich die aktuelle Position der Person abschätzen.<sup>28</sup>

Im ersten Schritt des Prozesses erfolgt die Schritterkennung, bei der das System die Bewegungsmuster des Nutzers analysiert, um zu bestimmen, wann ein Schritt ausgeführt wurde. Hierbei werden typische Signale wie Beschleunigungsmuster genutzt, um Schritte eindeutig zu identifizieren. Diese Signale weisen charakteristische Veränderungen auf, wenn ein Schritt gemacht wird, die vom System erkannt und gezählt werden.<sup>29</sup>

---

<sup>28</sup> Vgl. Kuang et al. 2018.

<sup>29</sup> Vgl. Jiang et al. 2022.

Nach der Erkennung eines Schrittes wird die Schrittlängenschätzung vorgenommen. Die Schrittlänge kann von mehreren Faktoren abhängen, darunter die Geschwindigkeit des Nutzers, der Neigungswinkel des Geländes und persönliche Eigenschaften wie die Beinlänge. Basierend auf den Beschleunigungsdaten und möglicherweise weiteren Kontextinformationen schätzt das System, wie weit sich der Nutzer mit jedem Schritt bewegt hat. Parallel dazu wird die Richtungserkennung durchgeführt, bei der die Bewegungsrichtung des Nutzers bestimmt wird. Diese Information kann zum Beispiel aus den Daten der Gyroskope und Magnetometer der IMUs abgeleitet, die Änderungen in der Orientierung und Ausrichtung des Nutzers erfassen. So kann das System feststellen, in welche Richtung sich die Person während der Schrittbewegung bewegt.<sup>30</sup>

Zuletzt erfolgt die Aktualisierung der Position. Dabei wird die geschätzte Schrittlänge in der erkannten Richtung zur aktuellen Position des Benutzers addiert, um die neue Position zu bestimmen. Dieser Prozess wird kontinuierlich wiederholt, so dass das System die Position des Benutzers in Echtzeit verfolgen kann, indem es die Anzahl der Schritte, die Schrittlänge und die Bewegungsrichtung integriert.<sup>31</sup>

### 4.3 Madgwick-Algorithmus

Der Madgwick-Algorithmus, entwickelt von Sebastian Madgwick, ist ein Algorithmus zur Sensordatenfusion, der speziell für die Berechnung der Orientierung konzipiert wurde. Er kombiniert die Daten von Beschleunigungsmessern, Gyroskopen und Magnetometern und nutzt einen Quaternion-basierten Ansatz. Quaternionen sind eine mathematische Repräsentation zur Beschreibung von Rotationen im dreidimensionalen Raum. Sie bestehen aus vier Komponenten und ermöglichen Berechnung von Rotationen, die in verschiedenen technischen und physikalischen Anwendungen verwendet wird. Das Grundprinzip des Madgwick-Algorithmus besteht darin, den Fehler zwischen der geschätzten und der tatsächlichen Orientierung mithilfe eines Optimierungsalgorithmus zu minimieren. Durch die Verwendung von Quaternionen zur Darstellung der Orientierung wird eine konsistentere und kontinuierlichere Schätzung ermöglicht.<sup>32</sup>

Ein zentraler Bestandteil des Madgwick-Algorithmus ist die Fehlerberechnung. Hierbei wird der Unterschied zwischen der geschätzten und der tatsächlichen Orientierung bestimmt. Dieser Fehler ergibt sich aus der Differenz zwischen den vom Algorithmus

---

<sup>30</sup> Vgl. Kuang et al. 2018 und Jiang et al. 2022.

<sup>31</sup> Vgl. Kuang et al. 2018.

<sup>32</sup> Vgl. Madgwick 2021 und Valenti et al. 2015.

geschätzten Sensorwerten und den tatsächlich gemessenen Werten. Das Ziel ist es, diesen Fehler so klein wie möglich zu halten, um eine möglichst genaue Schätzung der Orientierung zu erzielen. Um den Fehler zu minimieren, setzt der Algorithmus auf einen Gradient-Descent-Optimierungsansatz. Dabei handelt es sich um eine Methode, die iterativ den besten Satz von Parametern sucht, um den Fehler zu reduzieren. Der Algorithmus passt die Quaternionen schrittweise an, indem er in die Richtung des steilsten Abstiegs auf der Fehlerkurve geht. Durch diese iterative Anpassung wird die Schätzung der Orientierung ständig verfeinert, sodass der Fehler immer weiter abnimmt und die Berechnung der Orientierung immer genauer wird.<sup>33</sup>

Ein weiterer Schritt ist die Normalisierung der Quaternionen. Da Quaternionen eine besondere mathematische Struktur haben, müssen sie normiert, also auf eine bestimmte Länge skaliert werden, um gültig zu bleiben. Dies ist essenziell, um sicherzustellen, dass die Orientierungsschätzung konsistent bleibt. Durch die Normalisierung wird verhindert, dass die Berechnungen durch numerische Instabilitäten beeinträchtigt werden, was die Genauigkeit und Zuverlässigkeit des Algorithmus weiter erhöht.<sup>34</sup>

---

<sup>33</sup> Vgl. Madgwick 2021.

<sup>34</sup> Vgl. Valenti et al. 2015.



## 5 Anforderungen an eine Indoor-Navigations-App

In diesem Kapitel werden die Anforderungen an eine Indoor-Navigations-App der toom Baumarkt GmbH definiert. Dazu werden in Kapitel 5.1 die Qualitätsmerkmale für Software nach internationalem Standard vorgestellt. Daraus ergeben sich erste nicht-funktionale Anforderungen. In Kapitel 5.2 werden grundlegende technische Anforderungen betrachtet, die für jede Kategorie von Navigationssystemen gelten und ebenfalls in den Bereich der nicht-funktionalen Qualitätsanforderungen fallen. Dies umfasst neben dem speziellen Fall der Indoor-Navigationssysteme auch Systeme für den Außeneinsatz sowie Sicherheits-, Vermessungs- und Telekommunikationssysteme. Kapitel 5.3 befasst sich mit den Erwartungen der Nutzer an die Funktionen und die User Experience einer Navigations-App. Als Grundlage dient dabei der Marktführer im Bereich Navigations-Apps, Google Maps. Kapitel 5.4 widmet sich schließlich der Definition der funktionalen Anforderungen für einen ersten Prototypen. Dabei werden die Anforderungen aus Kunden- und Entwicklersicht zusammengeführt. Grundlage für die Entwicklung des Prototyps ist die Lösungsspezifikation in diesem Unterkapitel.

### 5.1 Qualitätsmerkmale nach internationaler Norm

Die internationale Norm für System- und Softwarequalität (ISO/IEC 25010:2023) definiert neun Qualitätsmerkmale für Softwareprodukte. Neben dem funktionalen Merkmal Functional suitability, das die Kernaufgaben der Anwendung abdeckt, beschreiben die Qualitätsmerkmale die nicht-funktionalen Anforderungen an die Softwarequalität. Jedes dieser Merkmale besteht aus mehreren Untermerkmalen.<sup>35</sup>

Die nicht-funktionalen Merkmale mit der höchsten Priorität für eine Indoor-Navigations-App sind die Performance efficiency, Interaction capability und Reliability (vgl. Abbildung 5). Die Performance efficiency bezieht sich auf den Ressourcenverbrauch. Bei mobilen Endgeräten sind beispielsweise Akkuladung und Netzwerkbandbreite Ressourcen, mit denen sparsam umgegangen werden muss. Die Interaction capability beschreibt die Fähigkeit zur Interaktion zwischen Benutzer und Anwendung. Sie ist damit eine Voraussetzung für die generelle Gebrauchstauglichkeit der Anwendung. Das dritte Merkmal, Reliability, umfasst die Fähigkeit der Anwendung,

---

<sup>35</sup> Vgl. ISO/IEC 2023, S. 10.

fehlerfrei und damit zuverlässig zu funktionieren. Ohne ein gewisses Maß an Zuverlässigkeit ist die Anwendung nutzlos.<sup>36</sup>

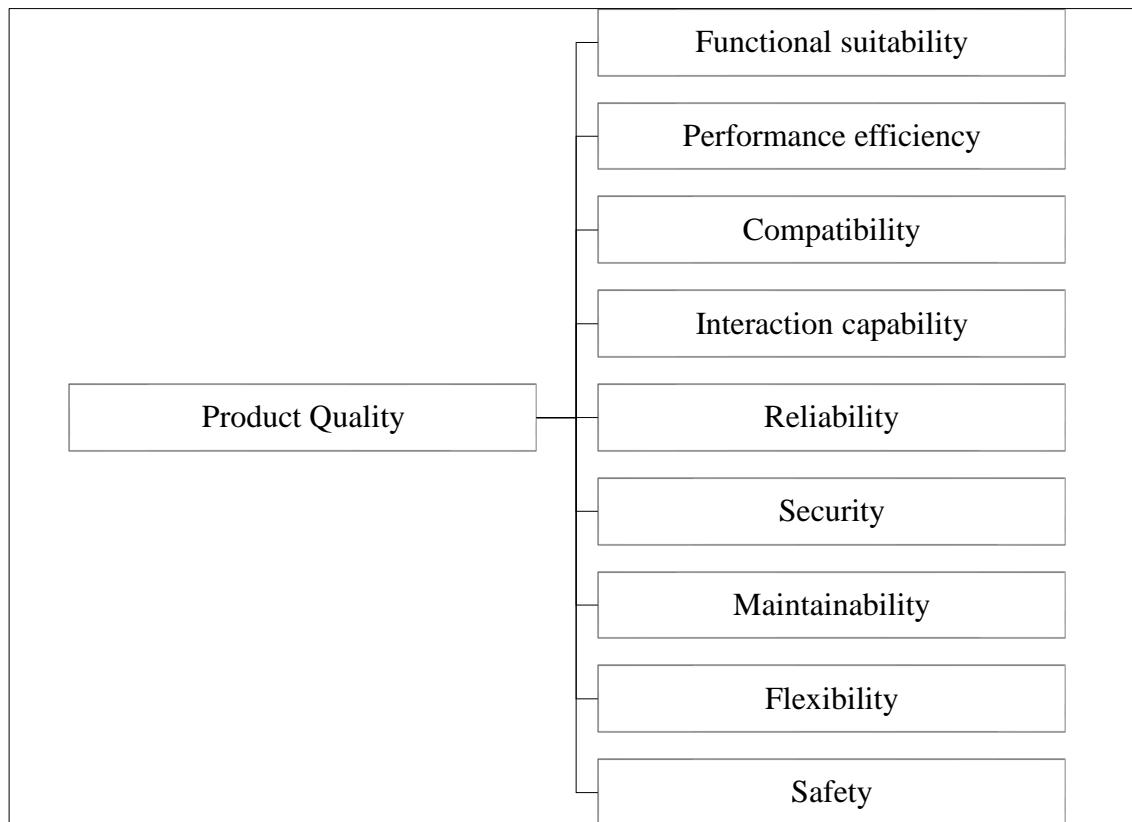


Abbildung 5: Product quality model nach ISO/IEC 25010.<sup>37</sup>

Die restlichen Qualitätsmerkmale spielen für den Fall einer Indoor-Navigations-App keine große Rolle. Compatibility beschreibt die Fähigkeit Daten mit anderen Produkten auszutauschen und Security dies nur bei entsprechenden Berechtigungen zu tun. Maintainability bezieht sich auf die Wartbarkeit und Erweiterbarkeit und Flexibilität auf die Anpassungsfähigkeit an unterschiedliche Umgebungen (z. B. Hardware oder Klima). Das letzte Merkmal, Safety, umfasst die Fähigkeit, den Nutzer vor Schaden zu bewahren.

## 5.2 Grundlegende Anforderungen an Navigationssysteme

Die European GNSS Agency definierte 2020 in ihrem User technology report grundlegende Anforderungen und Leistungsparameter an Navigationssysteme. Diese beziehen sich auf Qualitätsanforderungen der verwendeten Technik und sind je nach Kategorie des Navigationssystems unterschiedlich priorisiert. Mobile Endgeräte zählen z. B. zu den High Volume Devices, Such- und Rettungslösungen zu den Safety Devices,

<sup>36</sup> Vgl. hierzu und zu Folgendem ISO/IEC 2023, S. 2 ff.

<sup>37</sup> In Anlehnung an: ISO/IEC 2023, S. 10.

Vermessungssysteme zu den Accuracy Devices und Telekommunikationssysteme benötigen Synchronisationslösungen aus der Kategorie Timing Devices. Der in dieser Fallstudie betrachtete Anwendungsfall einer Indoor-Navigations-App fällt in die Kategorie der High-Volume Devices, im Gegensatz z. B. zu Flugsicherheitssystemen, die in die Kategorie der Safety Devices fallen. Der Vollständigkeit halber sind alle vier Kategorien in Tabelle 1 aufgeführt. Die dort aufgeführten Leistungsparameter sind größtenteils als nicht-funktionale Anforderungen an den Prototyp definiert und zur leichteren Referenzierung nummeriert und mit dem Präfix N versehen.<sup>38</sup>

<b>Priorität:</b>	<b>Hoch</b>	<b>Mittel</b>	<b>Niedrig</b>
N1: Verfügbarkeit: > 95%	<b>H</b> S A T		
N2: Time To First Fix (TTFF): < 3s	<b>H</b> S A T	T	
N3: Stromverbrauch	<b>H</b>	S	A T
N4: Indoor-Penetration	<b>H</b>	A	S T
N5: Genauigkeit: 10cm - 100cm	S A T	<b>H</b>	
N6: Robustheit	S A T	<b>H</b>	
N7: Kontinuität	S A T	<b>H</b>	
N8: Latenz	S	A	<b>H</b> T
N9: Integrität	S A T		<b>H</b>
<b>H: High Volume, S: Safety, A: Accuracy, T: Timing</b>			

Tabelle 1: Priorität grundlegender Leistungsparameter von Navigationssystemen.<sup>39</sup>

Die ersten vier Punkte haben die höchste Priorität für High Volume Devices. Verfügbarkeit (N1) ist die Zeit, in der Position und Navigation berechnet werden können. Sie sollte über 95% liegen. Die Time To First Fix (N2) beschreibt die Zeit bis zur ersten Informationsausgabe nach der Aktivierung. Dies kann der Start der Anwendung, der Start der Navigation oder die Neuberechnung der Route sein und sollte im Bereich unter 3 Sekunden liegen. Der Stromverbrauch (N3) sollte möglichst gering sein. Die Indoor-

<sup>38</sup> Vgl. European GNSS Agency 2020, S. 4.

<sup>39</sup> In Anlehnung an: European GNSS Agency 2020, S. 31, 47, 63 und 76.

Penetration (N4) ist vor allem für GPS-Systeme von Interesse und kann daher im Rahmen dieser Fallstudie vernachlässigt werden.<sup>40</sup>

Die nächsten drei Punkte haben eine mittlere Priorität für High Volume Devices. Die Genauigkeit (N5) beschreibt die Differenz zwischen der tatsächlichen und der berechneten Position und sollte, für die mittlere Prioritätsstufe, in 95% der Zeit im Dezimeterbereich zwischen 10 und 100 cm liegen. Für die anderen Kategorien von Navigationssystemen, die Genauigkeit höher priorisieren, ist ein Zentimeterbereich kleiner als 10 cm erforderlich. Robustheit (N6) bezieht sich auf den Umgang mit Fehlern, z. B. durch Map-Matching-Algorithmen<sup>41</sup>, und Kontinuität (N7) auf den unterbrechungsfreien Betrieb.

Die beiden Leistungsparameter mit der geringsten Priorität für High Volume Devices sind Latenz (N8) und Integrität (N9). Latenz beschreibt die Zeit zwischen Input und Output, z. B. zwischen dem Drehen des Mobiltelefons und dem Drehen der Karte. Richtwerte für die Latenz werden im User technology report 2020 nicht näher definiert. Integrität ist die Fähigkeit, Hinweise auf mögliche Gefahren oder Qualitätsverluste zu geben. In einer Indoor-Navigations-App sollten z. B. Hinweise zur Genauigkeit angezeigt werden, ähnlich wie bei Google Maps durch einen halbtransparenten Kreis, der sich in der Größe verändert.

Bezieht man diese Anforderungen auf die Norm ISO/IEC 25010:2023, so fallen die meisten in die Bereiche Reliability (Availability: N1, N4, N7; Fault tolerance: N6) und Performance efficiency (Time behaviour: N2, N8; Ressource utilization: N3). Die Integrität (N9) fällt in das Untermerkmal Hazard warning des Qualitätsmerkmals Safety. Eine Besonderheit ist die Anforderung der Genauigkeit (N5), die in die Kategorie der Functional correctness fällt und somit einen Bezug zu den funktionalen Anforderungen hat.<sup>42</sup>

### **5.3 Benutzererwartungen durch etablierte Standards des Marktes**

Die internationale Norm für Menschzentrierte Gestaltung interaktiver Systeme (ISO 9241-210:2019) definiert User Experience als die „Wahrnehmungen und Reaktionen einer Person, die aus der tatsächlichen und/oder der erwarteten Benutzung eines Systems

---

<sup>40</sup> Vgl. hierzu und zu Folgenden European GNSS Agency 2020, S. 99.

<sup>41</sup> Vgl. Chao et al. 2020, S. 121.

<sup>42</sup> Vgl. ISO/IEC 2023, S. 2, 5 und 8 f.

... resultieren“<sup>43</sup>. Sie umfasst unter anderem die Vorstellungen und Vorlieben des Nutzers nicht nur während, sondern auch vor und nach der Nutzung des Systems.<sup>44</sup>

Um zu verstehen, welche Erwartungen ein Nutzer an eine Navigations-App hat, folgt eine Analyse der derzeit beliebtesten Navigations-App. Wie Abbildung 6 zeigt, lag Google Maps im Oktober 2023 mit 59% Reichweite auf Platz 4 der beliebtesten Smartphone-Apps in den USA und ist damit die beliebteste Navigations-App. Im Vergleich dazu lag der Konkurrent Apple Maps mit 37% Reichweite auf Platz 12.

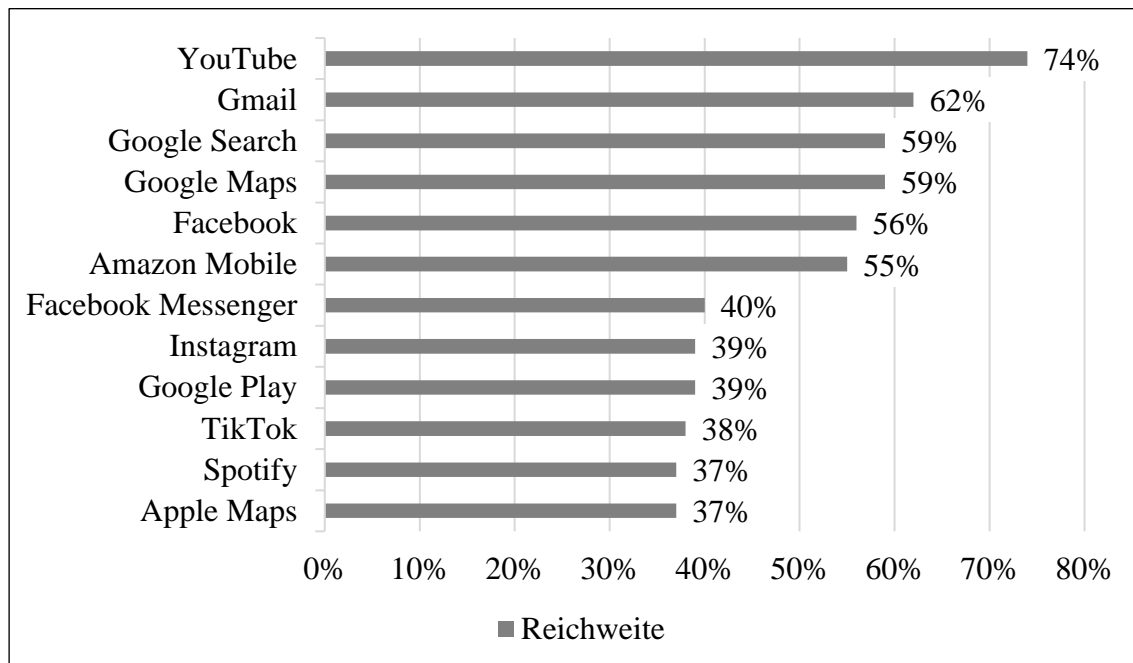


Abbildung 6: Ranking der Top 12 beliebtesten Smartphone-Apps in den USA nach Reichweite im Oktober 2023.<sup>45</sup>

Als Marktführer hat Google Maps Standards für die User Experience einer Navigations-App gesetzt, die heute von den Nutzern einer solchen Anwendung erwartet werden. Zu diesen Erwartungen gehört beispielsweise eine Karte der Umgebung, die die aktuelle Position anzeigt und auf diese zentriert ist. Die Drehung der Karte in Blickrichtung entsprechend der Ausrichtung des Endgerätes sowie das Zoomen und Verschieben der Karte mittels Gestensteuerung gehören ebenfalls zum Standard.<sup>46</sup>

Für die Zielauswahl muss ein Eingabefeld zur Verfügung stehen, das während der Eingabe Vorschläge anzeigt. Nach Zielauswahl sollte die schnellste Route auf der Karte

<sup>43</sup> DIN 2020, S. 11.

<sup>44</sup> Vgl. DIN 2020, S. 11.

<sup>45</sup> Vgl. Statista 2023 (leicht modifiziert).

<sup>46</sup> Vgl. Google LLC 2024.

angezeigt werden und die Navigation sollte über eine Schaltfläche gestartet und gestoppt werden können. Während der Navigation müssen zusätzliche Richtungspfeile und Beschreibungen eingeblendet werden.

## 5.4 Zielsetzung des Prototyps

Die Anforderungsspezifikation, der in dieser Fallstudie entwickelten Indoor-Navigations-App ergibt sich aus den zuvor definierten Erwartungen der Nutzer. Für den Prototyp sind dies die von Google Maps etablierten und in Tabelle 2 dargestellten Standards.

Prototyp	Optional
Umgebungskarte, zentriert auf Position	Informationen über Produkte in der Nähe
Rotation in Blickrichtung	Routenberechnung für mehrere Produkte
Gestennavigation	Unternehmens User Interface (UI)
Zielauswahl mit Vorschlägen	Standortbasierte Hinweise
Routenanzeige	Bewegungsdatenspeicherung
Start- und Stoppbare Navigation	Heat Maps
Richtungspfeile und -beschreibung	Integration in Softwareinfrastruktur

Tabelle 2: Anforderungsspezifikation der Indoor-Navigations-App.

Darüber hinaus gibt es optionale Anforderungen, die für eine Weiterentwicklung über einen Prototyp hinaus relevant sind. Dazu gehören z. B. die Anzeige von Informationen über Produkte in der Nähe und eine Routenberechnung für mehrere Produkte, z. B. für den gesamten Warenkorb. Hinzu kommen die Anforderungen an ein UI für Unternehmen sowie die Funktionalität für standortbezogene Hinweise wie Werbung und Kundenfeedback.

Zu den Stakeholdern gehört neben den Kunden im Markt auch das Unternehmen, das Anforderungen an eine Indoor-Navigations-App stellt. Diese wurden aus einem Meeting zwischen toom und der Firma Oriient New Media Ltd. abgeleitet, die toom eine Softwarelösung zur Indoor-Navigation mittels Magnetometrie anbietet.<sup>47</sup> Sie beziehen sich auf das Unternehmens UI und sind daher für den Prototyp dieser Fallstudie optional. Dieses UI dient der Darstellung verschiedener wertvoller Nutzerdaten, die aus der

<sup>47</sup> Vgl. Oriient New Media Ltd. 2023.

Navigationsanwendung als anonymisierte Bewegungsdaten gespeichert werden sollen. Aus diesen Bewegungsdaten können dann Heat Maps generiert werden, die zeigen, wie sich die Kunden in bestimmten Zeiträumen im Markt bewegen und wo sie sich länger aufhalten. Darüber hinaus ist eine Integration in die bestehende Softwarelandschaft des Unternehmens wünschenswert. So können z. B. durch einen Abgleich mit Bon-Daten zusätzliche Heatmaps erstellt werden, die die Ziele der Kunden im Markt darstellen. Eine weitere Anforderung ist die Auswertung von Überwachungskameradaten, um die Bewegungsprofile von Kunden zu ergänzen, die die App nicht nutzen.

<b>Prototyp</b>
F1: Speicherung von Kartendaten (Bild und Graphdaten)
F2: Anzeige einer Umgebungskarte
F3: Speicherung von Produktpositionen
F4: Zugang zu Entwicklermenü
F5: Initialpositionierung per Kamera und QR-Code
F6: Relative Positionierung per Inertialsensoren (N5: Genauigkeit)
F7: Zentrierung auf Benutzerposition (automatisch & durch Button)
F8: Rotation der Karte in Blickrichtung
F9: Zoomen und bewegen der Karte durch Gestensteuerung
F10: Suchfeld zur Zielauswahl
F11: Zielvorschläge anhand der Texteingabe
F12: Routenberechnung von Benutzerposition zum Ziel
F13: Anzeige der Route auf Karte
F14: Button zum Starten und Stoppen der Navigation
F15: Anzeige von Richtungspfeilen und -beschreibungen während der Navigation
NF1: Reliability (N1: Verfügbarkeit)
NF2: Performance efficiency (N2: TTFF; N3: Stromverbrauch)
NF3: Interaction capability (Operability)

Tabelle 3: Lösungsspezifikation der Indoor-Navigations-App.

Um die Anforderungsspezifikation umsetzen zu können, ergibt sich eine weitere Lösungsspezifikation, die in Tabelle 3 aufgeführt ist.<sup>48</sup> Die dort aufgeführten funktionalen und nicht-funktionalen Anforderungen sind zur leichteren Referenzierung nummeriert und mit dem Präfix F, bzw. NF versehen.

Zunächst müssen die Kartendaten in einem geeigneten Format gespeichert werden (F1). Dies umfasst sowohl die visuelle Ebene für die spätere Anzeigefunktion (F2) als auch den für die Routenberechnung notwendigen Graphen. Zur einfachen Bearbeitung des Graphen und der Produktpositionen (F3) im Prototyp muss ein Entwicklermenü zur Verfügung stehen (F4), das ein- und ausgeschaltet werden kann. Für die Initialpositionierung soll die Kamera verwendet werden (F5), z. B. durch Scannen eines QR-Codes. Nach der Initialpositionierung müssen die Daten der Bewegungssensoren verwendet werden, um die relative Positionsänderung zu berechnen (F6). Hierbei ist besonders auf die Genauigkeit zu achten (NF5). Mit diesen Positionsdaten kann dann die Karte zentriert (F7) und in Blickrichtung rotiert (F8) werden. Dies sollte beim Start der Anwendung automatisch geschehen. Wurde die Karte manuell verschoben (F9), muss eine Schaltfläche die Funktionalität zur Verfügung stellen. Für die Zielauswahl muss ein Suchfeld angezeigt werden (F10), das anhand der Texteingabe mehrere Vorschläge zur Auswahl anzeigt (F11). Schließlich muss die Anwendung eine Routenberechnungsfunktion von der Position des Nutzers zum ausgewählten Ziel anbieten (F12). Die berechnete Route muss durch eine Linie auf der Karte dargestellt werden (F13). Ab diesem Zeitpunkt muss eine Schaltfläche zum Starten der Navigation angezeigt werden (F14). Sobald die Navigation läuft, muss eine Schaltfläche zum Stoppen der Navigation vorhanden sein. Die letzte funktionale Anforderung (F15) bezieht sich auf die Anzeige von entsprechenden Richtungspfeilen und Beschreibungen während der Navigation.

Zu den nicht-funktionalen Anforderungen an den Prototyp gehört erstens die Reliability (NF1), die durch eine Verfügbarkeit von über 95% gewährleistet wird. Zweitens muss eine angemessene Performance efficiency (NF2) erreicht werden. Diese setzt sich zusammen aus einer TTFF von weniger als drei Sekunden und einem Stromverbrauch, der mit normalem Surfen im Internet vergleichbar ist. Schließlich wird eine Interaction capability (NF3) gefordert, die durch die Operabilität, d. h. die einfache Bedienbarkeit durch den Nutzer, nachgewiesen werden kann.<sup>49</sup>

---

<sup>48</sup> Vgl. Ebert 2022, S. 102.

<sup>49</sup> Vgl. ISO/IEC 2023, S. 3.



## 6 Konzept

In diesem Kapitel werden die zentralen Elemente der Indoor-Navigations-App für die toom Baumarkt GmbH beschrieben. Dazu werden im ersten Unterkapitel das Ablaufdiagramm vorgestellt, das die logischen Prozesse der App von der Initialisierung bis zur Zielerreichung visualisiert. Das Diagramm zeigt die Interaktionen zwischen den verschiedenen Komponenten der App und gibt einen Überblick über die Benutzerführung und die internen Abläufe. Im zweiten Unterkapitel wird das Klassendiagramm erläutert. Es definiert die wichtigsten Klassen und deren Beziehungen, einschließlich der L.Control-Klassen, die die Benutzeroberfläche steuern, sowie der Node- und Edge-Klassen, die den zugrunde liegenden Navigationsgraphen modellieren. Außerdem wird auf die Implementierung des Astar-Algorithmus eingegangen, der für die Routenplanung verwendet wird. Abschließend wird im dritten Unterkapitel das Design der App beschrieben. Im Fokus steht dabei das minimalistische UI, das dem Benutzer eine intuitive Navigation ermöglicht. Die visuelle Gestaltung, einschließlich der Kartenansicht und der interaktiven Elemente wie Suchfeld, Kompass und Richtungspfeile wird hierbei genauer beschrieben.

### 6.1 Modellierung

Die Modellierung der Indoor-Navigations-App dient dazu, die grundlegenden Prozesse und Strukturen der Anwendung visuell darzustellen und zu definieren. Sie bildet die Grundlage für die technische Umsetzung und sorgt dafür, dass die einzelnen Komponenten effizient miteinander interagieren. In den folgenden Abschnitten wird die Modellierung der App anhand zwei Diagramme beschrieben, die sowohl die Abläufe als auch die technische Architektur verdeutlichen.

#### 6.1.1 Ablaufdiagramm

Im folgenden Abschnitt wird das Ablaufdiagramm der Indoor-Navigations-App ausführlich beschrieben. Dabei werden die einzelnen Schritte des Navigationsprozesses von der Initialisierung bis zur Zielerreichung dargestellt. Das Diagramm veranschaulicht die logischen Abläufe innerhalb der App sowie die Interaktionen zwischen den verschiedenen Komponenten. Zudem wird beschrieben, wie der Nutzer durch die App geführt wird und wie die einzelnen Module zusammenarbeiten, um eine reibungslose Navigation zu gewährleisten. Ziel ist es, einen umfassenden Überblick über die technischen Abläufe und die Benutzerinteraktionen zu geben.

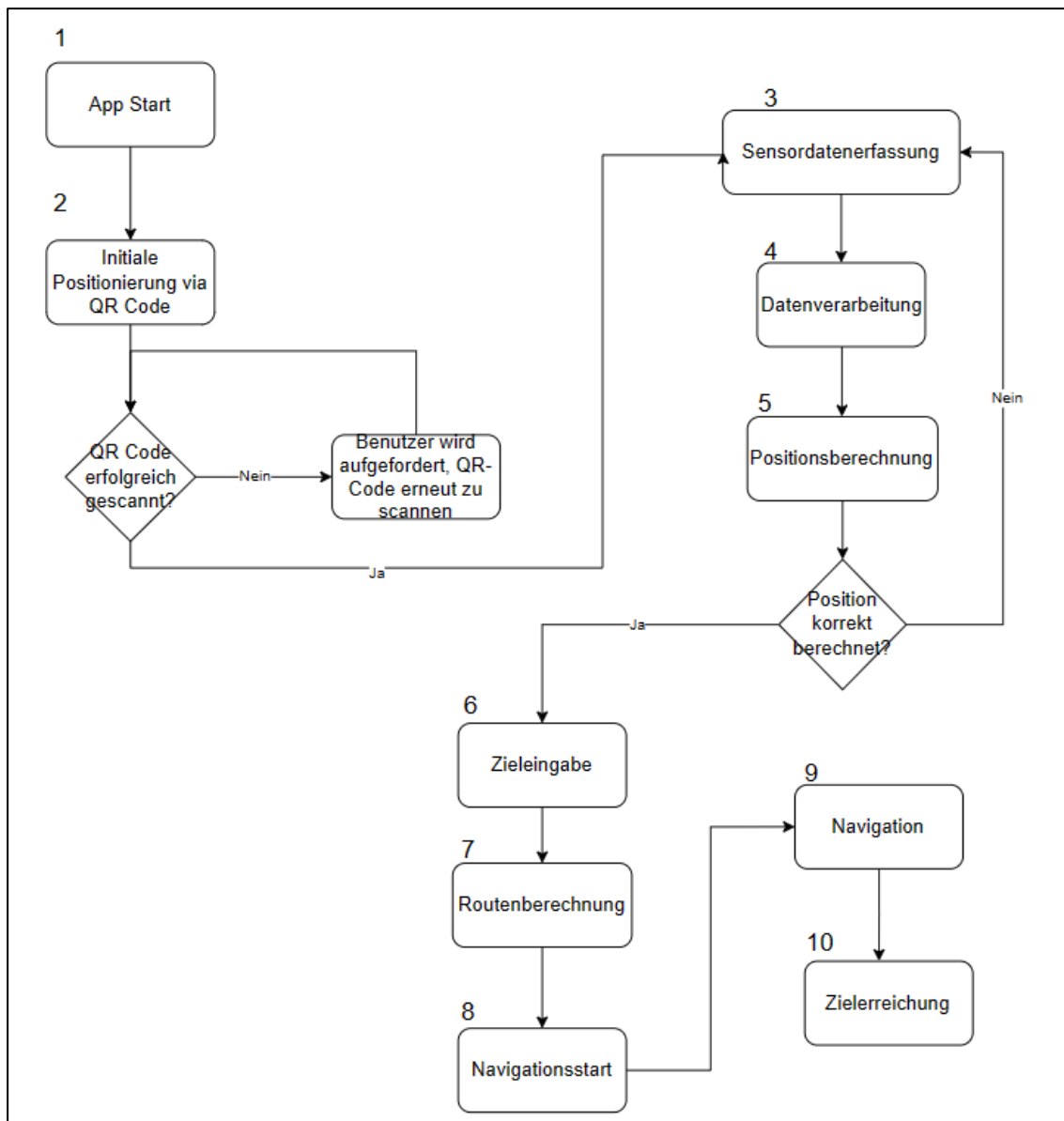


Abbildung 7: Ablaufdiagramm der Indoor-Navigation-App.

### 1. App Start

Dies ist der Beginn der Nutzung der Indoor-Navigations-App. Der Benutzer öffnet die App auf seinem mobilen Gerät. Die App ist darauf ausgelegt, auf den Bewegungssensoren des Geräts zu basieren, um die Navigation innerhalb eines Gebäudes zu ermöglichen. Dies stellt den Ausgangspunkt dar, an dem die App ihre Dienste für den Benutzer verfügbar macht.

Beim Öffnen der App werden mehrere Initialisierungsprozesse gestartet. Dazu gehört das Laden der Benutzeroberfläche, die Aktivierung von Sensoren wie Beschleunigungsmessern und Gyroskopen, die für die Erfassung von Bewegungen erforderlich sind. Darüber hinaus wird die App auf die Sammlung von Sensordaten vorbereitet.

## 2. Initiale Positionierung via QR-Code

Vor dem Start der Navigation muss die aktuelle Position des Nutzers im Gebäude ermittelt werden. Die App ermöglicht dies durch das Scannen eines QR-Codes, der an einem festen Punkt innerhalb des Gebäudes angebracht ist. Dies bietet eine Ausgangsposition für die anschließende Navigation.

Die Kamera des Smartphones wird aktiviert, um den QR-Code zu scannen. Dieser Code enthält Positionsdaten, die den Startpunkt der Navigation festlegen. Dies ist notwendig, da IMU-Sensoren keine Positionsdaten liefern können, sondern nur Bewegungsinformationen. Durch den QR-Code wird die exakte Anfangsposition gesetzt. Alternativ könnten auch andere Technologien wie Bluetooth verwendet werden, aber der QR-Code bietet eine kostengünstige und datenschutzfreundliche Lösung.

## 3. Sensordatenerfassung

Nachdem die Startposition erfasst wurde, beginnt die App, kontinuierlich die Sensordaten des Geräts zu sammeln. Die Daten stammen von der IMUs, die Beschleunigung und Drehbewegung erfasst. Diese Daten sind essenziell für die Navigation, da sie die Bewegung des Geräts im Raum überwachen.

Diese Rohdaten werden in nahezu Echtzeit verarbeitet und in physikalische Einheiten umgerechnet, die dann zur Berechnung der Position und Bewegung genutzt werden.

## 4. Datenverarbeitung

Die erfassten Sensordaten werden mit Algorithmen verarbeitet, um die Position des Nutzers im Raum zu bestimmen. Zwei Hauptalgorithmen kommen hier zum Einsatz: Der Kalman-Filter und der PDR. Diese Algorithmen arbeiten zusammen, um generell eine bessere Datenqualität gewährleisten zu können.

## 5. Positionsberechnung

Auf der Grundlage der verarbeiteten Sensordaten wird die aktuelle Position des Nutzers berechnet. Diese Berechnung wird in beinahe Echtzeit durchgeführt und ständig aktualisiert, während der Nutzer sich durch das Gebäude bewegt. Die Positionsdaten werden auf der Karte der App angezeigt, sodass der Nutzer immer seine genaue Position im Raum sehen kann.

Die Schritte des Nutzers werden durch den PDR erkannt und analysiert. Dabei wird die Schrittlänge geschätzt, basierend auf Faktoren wie der Schrittfrequenz und dem Bewegungsmuster des Nutzers.

Die Bewegungsrichtung wird durch die Orientierung des Smartphones ermittelt, wobei die Daten des Gyroskops verwendet werden, um Drehbewegungen zu erkennen. Durch die Kombination dieser Daten wird die Position des Nutzers kontinuierlich berechnet und auf der Karte in regelmäßigen Intervallen aktualisiert.

## 6. Zieleingabe

Der Nutzer gibt ein Ziel ein, zu dem er navigieren möchte. Die App bietet eine Suchfunktion, bei der der Nutzer ein gesuchtes Produkt innerhalb des Gebäudes eingibt oder aus vorgeschlagenen Zielen auswählt. Dies bildet den Startpunkt der Routenberechnung. Die App greift auf Daten mit den Positionen im Gebäude zurück, um Ziele vorzuschlagen, die für den Nutzer relevant sein könnten.

## 7. Routenberechnung

Nachdem der Nutzer ein Ziel ausgewählt hat, berechnet die App die optimale Route von der aktuellen Position zum Ziel.

Die App verwendet eine interne Karte des Gebäudes, die als Graph modelliert ist. Auf dieser Grundlage wird die kürzeste Route zum Ziel berechnet. Außerdem wird die berechnete Route auf der Karte visuell dargestellt, indem eine Linie den Weg von der aktuellen Position des Nutzers zum Ziel markiert.

## 8. Navigationsstart

Der Benutzer startet die Navigation und die App beginnt, ihn in nahezu Echtzeit auf der berechneten Route zu führen. Während der Navigation werden die aktuelle Position und Richtungspfeile auf der Karte angezeigt. Während der Navigation aktualisiert die App kontinuierlich die Position des Nutzers. Dies geschieht durch fortlaufende Berechnung der Bewegungsdaten in Kombination mit den Informationen aus der Route.

Falls der Nutzer von der geplanten Route abweicht, berechnet die App automatisch eine neue Route zum Ziel. Die Navigationsanweisungen werden durch Richtungspfeile auf der Karte visualisiert und durch Textanweisungen ergänzt. Der Nutzer wird benachrichtigt, wann er abbiegen oder geradeaus gehen muss.

## 9. Navigation

Während der Navigation zeigt die App in nahezu Echtzeit die Position und die Bewegungsrichtung des Nutzers an. Sie gibt kontinuierlich Richtungsanweisungen, um den Nutzer sicher zum Ziel zu führen.

Sobald der Nutzer eine Kreuzung oder Abzweigung erreicht, zeigt die App klare Richtungspfeile an, um den weiteren Weg zu weisen.

## 10. Zielerreichung

Der Nutzer erreicht sein Ziel, und die Navigation wird beendet. Die App informiert den Nutzer darüber, dass er sein Ziel erreicht hat. Anschließend kann der Nutzer die Navigation beenden oder eine neue Route zu einem anderen Ziel starten.

### 6.1.2 Klassendiagramm

In diesem Unterkapitel wird das Klassendiagramm der Indoor-Navigations-App beschrieben. Dabei wird auch auf die Rolle der wichtigsten Klassen wie L.Control, Node, Edge und Astar eingegangen, um die technischen Zusammenhänge und die Funktionsweise der App zu verdeutlichen.

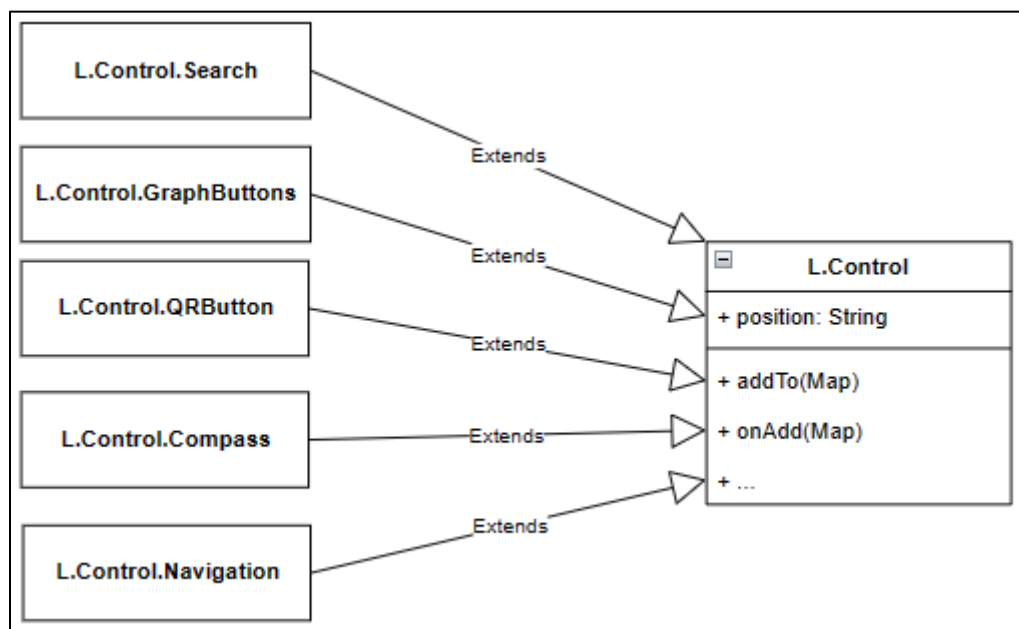


Abbildung 8: Klassendiagramm – L.Control.

Die L.Control-Klassen spielen eine zentrale Rolle bei der Darstellung der Benutzeroberfläche der Indoor-Navigations-App. Jede dieser Klassen implementiert unterschiedliche Steuerelemente, die es dem Benutzer ermöglichen, die App zu nutzen und mit ihr zu interagieren. Die L.Control.Search-Klasse beispielsweise stellt ein

Suchfeld zur Verfügung, mit dem der Benutzer nach Zielen oder Positionen auf der Karte suchen kann. Diese Klasse fügt sich in die Karte ein und bietet eine intuitive Möglichkeit, durch die Karte zu navigieren und relevante Orte zu finden.

Die L.Control.QRButton-Klasse ist ein Steuerelement, das speziell dafür geeignet ist, den QR-Code-Scanner der App zu öffnen. Dieser Button ermöglicht es dem Benutzer, durch einfaches Antippen den QR-Code zu scannen, um die aktuelle Position im Gebäude festzulegen oder zu aktualisieren. Die L.Control.Compass-Klasse ist dafür zuständig, die Karte basierend auf der Ausrichtung des Kompasses automatisch auszurichten. Sie aktiviert oder deaktiviert die Live-Ausrichtung der Karte, je nachdem, ob der Kompass aktiv ist. Wenn der Kompass aktiviert ist, passt sich die Karte automatisch der Ausrichtung des Nutzers an, und der Benutzer kann die Karte nicht manuell steuern. Ist der Kompass deaktiviert, hat der Benutzer die volle Kontrolle über die Kartennavigation. Diese Funktion erleichtert die Orientierung in Innenräumen, indem sie die Karte dynamisch an die Blickrichtung des Nutzers anpasst. Die L.Control.Navigation-Klasse ist ein zentrales Steuerelement, das speziell dafür entwickelt wurde, die Navigation in der App zu starten. Sie stellt einen Button bereit, mit dem der Benutzer den Navigationsprozess aktivieren kann. Sobald der Button betätigt wird, beginnt die App, die Route zu berechnen und die Navigation einzuleiten. Diese Klasse fokussiert sich ausschließlich auf das Starten der Navigation und leitet den Benutzer durch die zuvor berechnete Route.

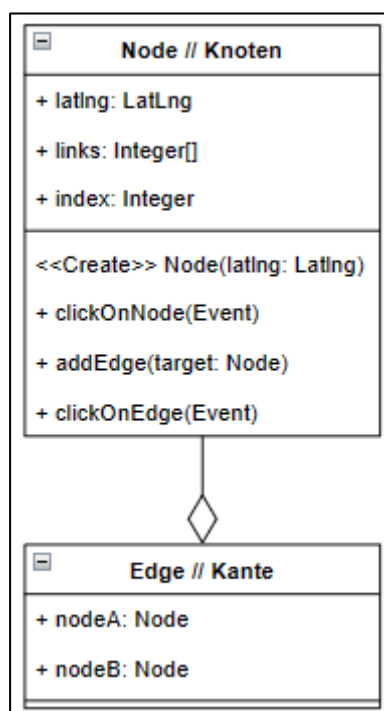


Abbildung 9: Klassendiagramm – Node & Edge.

Die Node, Edge und Product-Klassen bilden die Grundlage für den zugrunde liegenden Graphen, der für die Navigation innerhalb des Gebäudes verwendet wird. Node (Knoten) repräsentiert einzelne Positionen auf der Karte. Jeder Knoten enthält Koordinaten (Latitude und Longitude), Verbindungen (Edges) zu anderen Knoten und eine Identifikationsnummer (Index). Diese Knoten bilden zusammen mit den Edge-Klassen den Graphen, der zur Berechnung von Routen verwendet wird. Eine Edge repräsentiert die Verbindung zwischen zwei Knoten, die die Distanz oder die Verbindung zwischen den Orten beschreibt. Diese Struktur ermöglicht es der App, verschiedene Punkte miteinander zu verknüpfen und so die Grundlage für die Navigation zu schaffen.

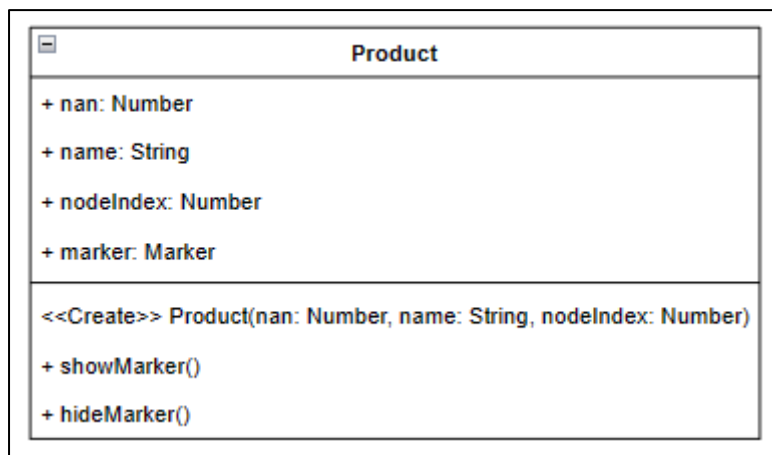


Abbildung 10: Klassendiagramm – Produkt.

Die Product-Klasse repräsentiert spezifische Produkte oder wichtige Orte innerhalb des Gebäudes, die auf der Karte durch Marker visualisiert werden. Jeder Produktmarker ist mit einem bestimmten Knoten (Node) im Navigationsgraphen verknüpft, wobei der Node Index auf die exakte Position des Produkts verweist. Der `nodeIndex` gibt also den Standort des Produkts auf der Karte an, indem er auf den entsprechenden Knoten zeigt, der die geografischen Koordinaten des Produkts repräsentiert.

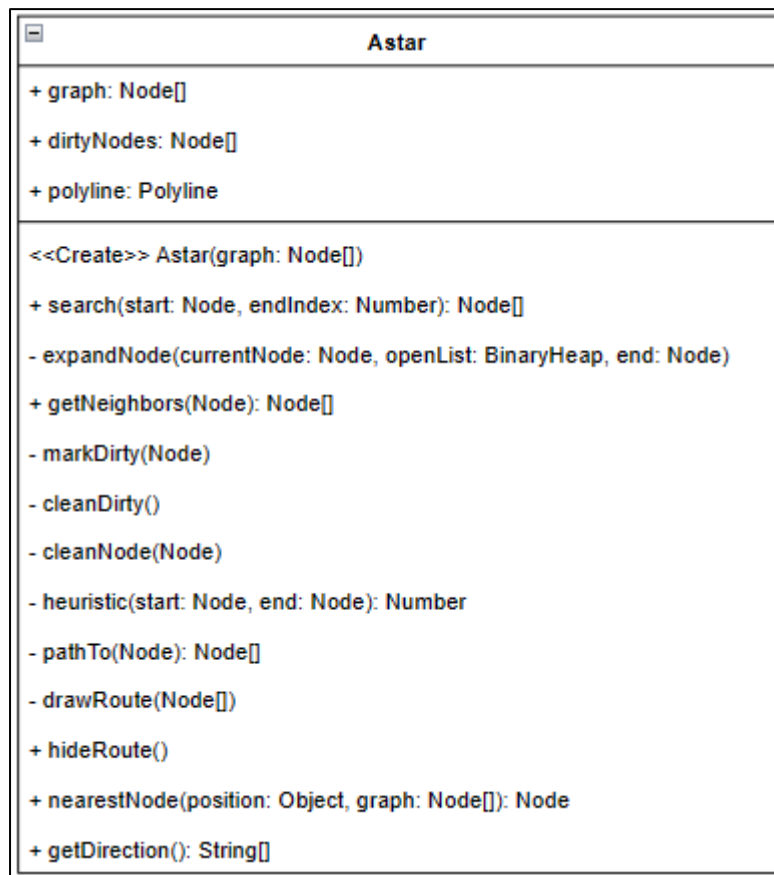


Abbildung 11: Klassendiagramm – Astar.

Der Astar-Algorithmus, der durch die Astar-Klasse implementiert wird, ist für die Berechnung der besten Route zwischen zwei Punkten im Graphen verantwortlich. Der Astar-Algorithmus ist eine gängige Methode zur Pfadfindung und die Search-Methode wird verwendet, um die kürzeste Route zwischen zwei Knoten zu ermitteln. Dabei wird eine Liste von Knoten durchlaufen und die optimale Verbindung ermittelt. Eine der wichtigsten Methoden dieser Klasse ist `getDirection()`, die dafür verantwortlich ist, die Richtungspfeile und Textanweisungen für den Benutzer zu berechnen. Zum Beispiel, dass der Nutzer links abbiegen soll. Dies ist entscheidend, da der Benutzer während der Navigation klare Anweisungen benötigt, um sein Ziel zu erreichen.

## 6.2 Design

Im folgenden Abschnitt wird das Design der Indoor-Navigations-App detailliert beschrieben. Dabei liegt der Fokus auf der Gestaltung der Benutzeroberfläche, den verwendeten UI-Elementen sowie der visuellen Struktur. Es wird erläutert, wie das minimalistische Design zur Benutzerfreundlichkeit beiträgt und welche Überlegungen bei der Anordnung der interaktiven Elemente, wie der Kartenansicht, der Suchleiste und der Navigationsanweisungen, eine Rolle gespielt haben.





Abbildung 12: Designkonzept der Anwendung.

Beim Öffnen der Indoor-Navigations-App wird die Benutzeroberfläche in einem klaren und minimalistischen Design dargestellt. Der größte Teil des Bildschirms wird von einer Kartenansicht eingenommen, die das Layout des Gebäudes zeigt. Gänge und Bereiche des Gebäudes sind durch blaue Rechtecke gekennzeichnet, während der Hintergrund in neutralem Grau gehalten ist. Die Suchleiste, prominent im oberen Bereich des Bildschirms platziert, erlaubt es dem Benutzer, nach Produkten oder Bereichen im Gebäude zu suchen. Sie ist schlicht und intuitiv gestaltet, sodass eine einfache Bedienung gewährleistet ist.

In der rechten unteren Ecke befinden sich zwei wesentliche Bedienelemente: ein Kompass-Button und ein QR-Code-Scanner-Button. Der Kompass-Button dient dazu, die Karte basierend auf der Ausrichtung des Nutzers anzupassen. Der QR-Code-Button ermöglicht es dem Nutzer, durch Scannen eines QR-Codes die Startposition innerhalb des Gebäudes zu erfassen. Die aktuelle Position des Nutzers wird durch einen blauen Punkt auf der Karte angezeigt, der in nahezu Echtzeit aktualisiert wird, um den Standort des Benutzers im Gebäude darzustellen.

Im zweiten Zustand der App wird die Suchfunktion aktiviert und ein Suchbegriff eingegeben. In diesem Fall wurde „holz“ eingegeben, woraufhin eine Dropdown-Liste mit passenden Vorschlägen wie „Holzleim“ erscheint. Diese Vorschläge erleichtern die Auswahl eines Produkts.

Im dritten Zustand wird ein Produkt ausgewählt, in diesem Fall „Holzleim“, aus der Vorschlagsliste aus. Daraufhin wird eine Route zum gewählten Produkt berechnet und als blaue Linie auf der Karte dargestellt, während das Ziel durch einen Marker auf der

Karte markiert ist. Zudem erscheint ein „Starten“-Button im linken unteren Bereich, der es dem Benutzer ermöglicht, die Navigation zu beginnen. Sobald dieser Button gedrückt wird, führt die App den Benutzer in nahezu Echtzeit zum ausgewählten Ziel.

Im vierten Zustand beginnt die Navigation. Ein großer grüner Banner am oberen Rand des Bildschirms zeigt Anweisungen wie „Weiter geradeaus“ an, um den Benutzer visuell zu unterstützen. Die Route bleibt weiterhin als blaue Linie auf der Karte sichtbar, die den aktuellen Standort des Nutzers mit dem Ziel verbindet. Zusätzlich wird ein roter „Abbrechen“-Button links unten eingeblendet, der es dem Nutzer ermöglicht, die Navigation zu jeder Zeit zu beenden.

## 7 Umsetzung

In diesem Kapitel wird die Implementierung der Software für eine Indoor-Navigations-App beschrieben, die ausschließlich auf den IMU-Daten eines Smartphones basiert. Die App wurde in JavaScript, HTML und CSS entwickelt, um eine Webanwendung zu erstellen, die in nahezu Echtzeit die Position und Orientierung des Nutzers ermittelt und darstellt. In Kapitel 7.1 erfolgt eine Erläuterung der Softwarearchitektur, bei der der Einsatz von sechs externen JavaScript-Bibliotheken zur Realisierung wichtiger Funktionen wie der Kartendarstellung, QR-Code-Lesung und der Glättung der Sensordaten durch einen Kalman-Filter beschrieben wird. In Kapitel 7.2 erfolgt eine Erläuterung der Speicherung von Navigations- und Produktdaten in JSON-Dateien, welche die Navigation und Positionierung innerhalb des Gebäudes ermöglicht. In dem Kapitel 7.3 wird die Verarbeitung der Sensordaten erörtert, welche der Bestimmung der Nutzerbewegungen dient. Der Kalman-Filter wird in Kapitel 7.3.1 eingeführt, um die Sensordaten zu glätten und präzisere Bewegungsinformationen zu gewinnen. Schließlich wird in Kapitel 7.3.2 die PDR-Methode beschrieben, welche anhand der erfassten Schritte die Position des Nutzers Schritt für Schritt berechnet.

### 7.1 Softwarearchitektur

Für die exemplarische Umsetzung des Prototyps wurde eine Webanwendung erstellt, die sechs externe Javascript-Bibliotheken verwendet, um die Anforderungen umsetzen zu können. Die Entscheidung für eine Webapplikation und gegen eine native Applikation fiel aufgrund der einfacheren Umsetzung, da für die Ausführung auf den verfügbaren mobilen Endgeräten (iPhone 13 mini) keine zusätzliche Apple-Hardware (Mac) benötigt wird. Der Nachteil an einer Webapplikation ist jedoch die fehlende Unterstützung der Sensor-APIs und damit des Magnetometers.<sup>50</sup> Beschleunigungssensor und Kompass können jedoch verwendet werden.<sup>51</sup>

Die erste in Abbildung 13 aufgeführte Bibliothek, Leaflet, stellt Klassen für die Darstellung einer Karte zur Verfügung. Durch ihre Verwendung werden die Anforderungen F2 und F9 umgesetzt, d. h. die Anzeige und die Gestensteuerung der Karte. Die Bootstrap-Bibliothek wird für die Gestaltung des UI verwendet. Der Kalman-Filter wird mit Hilfe der kalman-filter-Bibliothek implementiert. Die Fuse Bibliothek

---

<sup>50</sup> Vgl. Mozilla 2024a.

<sup>51</sup> Vgl. Mozilla 2024b und Mozilla 2024c.

stellt eine Klasse zur Verfügung, die fuzzy searching ermöglicht, d. h. die Suche nach ähnlichen statt exakt gleichen Zeichenketten. Damit können die Zielvorschläge aus der Eingabe im Suchfeld der Anforderung F11 umgesetzt werden. Für den Vergleich zweier Objekte wird die Bibliothek lodash verwendet. Die sechste Bibliothek html5-qrcode schließlich stellt eine Klasse zum Scannen von QR-Codes zur Verfügung. Damit kann die Anforderung F5 umgesetzt werden, indem ein Positionsobjekt im JSON-Format, z. B. {"lat":55,"lng":500}, mittels QR-Codes kodiert wird.

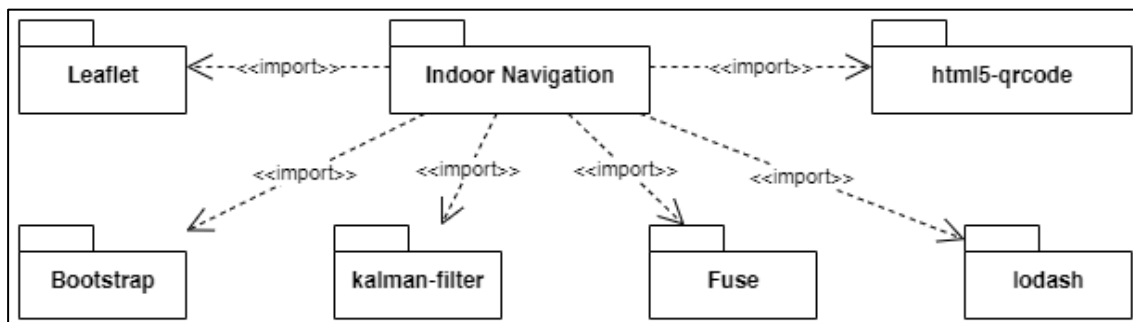


Abbildung 13: Paketdiagramm der Anwendung.

## 7.2 Speicherung der Daten

Grundlage für die Navigation von Punkt zu Punkt ist der Navigationsgraph mit den zugehörigen Produktpositionen und einem Bild der Umgebungskarte. Dies entspricht den Anforderungen F1 und F3. Das Bild ist als PNG-Datei hinterlegt, es können aber auch andere Bildformate verwendet werden. Die Graph- und Produktdaten liegen im JSON-Format vor, das in Abbildung 14 dargestellt ist.



Abbildung 14: Beispiel für JSON-Objekte des Graphen und der Produkte.

Die Datei „graph.json“ enthält ein Array von Node-Objekten. Jedes Node-Objekt hat ein Positionsobjekt „latlng“, einen Index und ein Array mit den Indizes der verknüpften

Nodes namens „links“. Die Datei „products.json“ enthält ein Array von Produktobjekten. Dazu gehören die siebenstellige interne Artikelnummer „nan“, der Produktname und der Index der Node, an dem sich das Produkt im Markt befindet.

### 7.3 Auslesen und Umrechnen der Sensordaten

Um die Sensordaten der IMUs eines Smartphones nutzen zu können, muss zunächst die Berechtigung zum Zugriff auf diese Sensoren angefordert werden. Anschließend werden die Daten der einzelnen Sensoren, um die Bewegungen und die Ausrichtung des Geräts zu bestimmen. Auf Geräten mit iOS 13 und höher ist es notwendig, eine explizite Berechtigung zur Nutzung der Bewegungssensoren anzufordern. Diese Berechtigung wird über die Funktion `requestSensors()` erteilt. (Vgl. Abbildung 15)

```
function requestSensors () {
  if (
    DeviceOrientationEvent &&
    typeof DeviceOrientationEvent.requestPermission === 'function'
  ) {
    DeviceOrientationEvent.requestPermission()
  }
}
```

Abbildung 15: Code der Funktion `requestSensors()`.

Diese Funktion prüft, ob eine Autorisierung erforderlich ist und fordert diese gegebenenfalls an. Ohne diese Zustimmung wäre der Zugriff auf die Sensoren der iOS-Geräte blockiert, was die Erfassung von Bewegungs- und Orientierungsdaten verhindern würde. Sobald die Berechtigung erteilt wurde, kann über die Funktion `handleMotion()` auf die Daten des Beschleunigungssensors zugegriffen werden. (Vgl. Abbildung 16)

```
function handleMotion (event) {
  if (motionArray.length >= motionArrayLength) {
    motionArray.shift()
  }
  motionArray.push([event.acceleration.x, event.acceleration.y, event.acceleration.z])
}
```

Abbildung 16: Code der Funktion `handleMotion()`.

Diese Funktion sammelt die Beschleunigungswerte entlang der x-, y- und z-Achse. Die betreffenden Werte werden durch den Beschleunigungsmesser bereitgestellt, welcher die lineare Beschleunigung des Geräts misst. Die gemessenen Daten werden in ein Array (`motionArray`) eingefügt, welches die Bewegungen des Geräts im Raum aufzeichnet.

Die Funktion `handleOrientation()` greift auf die Gyroskop- und Kompassdaten zu, um die Orientierung und Drehbewegungen des Geräts zu bestimmen. (Vgl. Abbildung 17)

Die Ereignisse, die diese Daten liefern, werden über Eventlistener überwacht. Diese Listener reagieren auf Änderungen der Bewegungs- und Orientierungsdaten des Geräts und rufen entsprechende Funktionen zur Verarbeitung dieser Daten auf.

```
function handleOrientation (event) {
  const bias = 120 // rotation of png
  const orientation = 360 - event.webkitCompassHeading
  map.setBearing(orientation + bias)
  motionArray[motionArray.length - 1].push(event.webkitCompassHeading, event.beta, event.gamma)
  const newPosition = calculatePosition(motionArray, userPosition, bias)
  userPosition = L.latLng(newPosition.lat, newPosition.lng)
  circle.setLatLng(userPosition)
  centerPosition()
}
```

Abbildung 17: Code der Funktion `handleOrientation()`.

Der Wert `webkitCompassHeading` repräsentiert die Daten des Kompasses. Diese Daten werden verwendet, um die Ausrichtung des Geräts zu bestimmen. Die Variablen `beta` und `gamma` werden vom Gyroskop geliefert. Sie messen die Rotationsgeschwindigkeit und die Neigung des Geräts entlang der Achsen, um genaue Informationen über die Rotationsbewegungen zu liefern.

### 7.3.1 Kalman-Filter Implementierung

Der Kalman-Filter wird in der Funktion `kFilter()` eingesetzt, um die Sensordaten, die die Bewegung des Benutzers aufzeichnet, zu glätten. (Vgl. Abbildung 18)

```
export function kFilter (arr) {
  const kFilter = new KalmanFilter({
    observation: {
      name: 'sensor',
      sensorDimension: 6,
      sensorCovariance: [1, 1, 1, 1, 1, 1]
    },
    dynamic: {
      name: 'constant-speed'
    }
  })
  return kFilter.filterAll(arr)
}
```

Abbildung 18: Code der Funktion `kFilter()`.

In dieser Implementierung wird der Kalman-Filter so konfiguriert, dass er Sensordaten in Form von Beschleunigung in drei Richtungen und weiteren Parametern verarbeitet. Die Funktion filtert die Sensordaten, um rauschärmere Bewegungsdaten zu erhalten, die später in der PDR-Implementierung genutzt werden.

### 7.3.2 PDR-Implementierung

Der PDR-Algorithmus berechnet die Position des Benutzers, indem er die Bewegungen Schritt für Schritt verfolgt. Die Berechnung der Bewegungen des Benutzers in x- und y-Richtung erfolgt ausgehend von einer bekannten Startposition, einem QR-Code, unter Zuhilfenahme trigonometrischer Funktionen.

Um die Schritte zu erkennen wird die Funktion `detectPeak()` verwendet. (Vgl. Abbildung 19). Die Funktion überprüft, ob ein Schritt aufgetreten ist, indem sie die Beschleunigungswerte analysiert. Sobald ein Peak erkannt worden ist, was bedeutet, dass der Mittelwert der Beschleunigungsdaten größer ist als die umliegenden Werte, wird die Position entsprechend aktualisiert.

```
export function detectPeak (data) {  
  const len = data.length  
  if (len < 3) return false  
  
  const after = data[len - 1]  
  const middle = data[len - 2]  
  const before = data[len - 3]  
  
  return (middle > after && middle > before && middle > stepThreshold)  
}
```

Abbildung 19: Code der Funktion `detectPeak()`.

Nachdem ein Schritt erkannt wurde, berechnet die Funktion `updatePosition()` die neue Position des Benutzers, indem sie die aktuelle Orientierung und Schrittlänge verwendet. Die Schrittlänge wird dabei mit dem Kosinus und Sinus des Winkels multipliziert, um die Verschiebung in x- und y-Richtung zu ermitteln. Diese Verschiebungen werden anschließend zu den aktuellen Koordinaten des Benutzers hinzugefügt, um die neue Position zu bestimmen. (Vgl. Abbildung 20)

```
function updatePosition (lastOrientation) {
  const directionRad = toRadians(lastOrientation)
  const cosDirection = Math.cos(directionRad)
  const sinDirection = Math.sin(directionRad)
  // Calculate new position
  position.lat += stepLength * cosDirection
  position.lng += stepLength * sinDirection

  return position
}
```

Abbildung 20: Code der Funktion updatePosition().

Die Funktion calculatePosition() kombiniert die gefilterten Sensordaten aus dem Kalman-Filter und die PDR-Berechnung, um die Position des Benutzers zu ermitteln. Diese Funktion ruft zuerst den Kalman-Filter auf, um die Beschleunigungsdaten zu glätten und erkennt dann den Peak mithilfe der detectPeak()-Funktion. Sobald ein Schritt erkannt wird, wird die Position des Benutzers aktualisiert. (Vgl. Abbildung 21)

```
export function calculatePosition (motionArray, userPosition, bias) {
  position.lat = userPosition.lat
  position.lng = userPosition.lng
  const rotationBias = 360 - bias
  const filteredArrays = kFilter(motionArray)

  const lastIndex = filteredArrays.length - 1
  const [xFiltered, yFiltered, zFiltered] = filteredArrays[lastIndex].slice(0, 3)
  const lastOrientation = motionArray[lastIndex][3] + rotationBias
  // Calculation of magnitude
  const magnitude = Math.sqrt(xFiltered ** 2 + yFiltered ** 2 + zFiltered ** 2)
  if (magnitudeArray.length >= magnitudeArrayLength) {
    magnitudeArray.shift()
  }
  magnitudeArray.push(magnitude)
  // Update position if step is detected
  if (detectPeak(magnitudeArray)) {
    return updatePosition(lastOrientation)
  }
}
```

Abbildung 21: Code der Funktion calculatePosition().

Diese Funktion kombiniert die verschiedenen Schritte der Berechnung. Sie filtert die Sensordaten mit dem Kalman-Filter, erkennt die Schritte und verwendet die aktuelle Orientierung, um die neue Position des Benutzers zu bestimmen.



## 8 Fazit

Im Rahmen dieser Fallstudie wurde die Funktionsweise sowie die Bedeutung der Inertial Measurement Unit in Smartphones untersucht. Diese Sensoren, bestehend aus einem Beschleunigungssensor, einem Gyroskop sowie einem Magnetometer, liefern Daten über die Bewegung und die Ausrichtung des Gerätes, was insbesondere für Anwendungen wie Indoor-Navigation sowie Bewegungserkennung von Bedeutung ist.

Im Rahmen dieser Arbeit wurde ein Prototyp einer Indoor-Navigations-App als Webanwendung entwickelt. Dabei konnte das Magnetometer nicht verwendet werden, da der Zugriff auf diesen Sensor eine native Applikation erfordert hätte. Aus diesem Grund wurde auf die Implementierung des Madgwick-Algorithmus, der auf die Verwendung von Magnetometerdaten angewiesen ist, verzichtet. Durch die Entwicklung einer nativen Anwendung, die das Magnetometer vollständig integriert, könnten zukünftige Forschungsarbeiten die Präzision und Robustheit der Sensordatenverarbeitung weiter verbessern. Die Anwendung des Madgwick-Algorithmus würde eine Sensordatenfusion ermöglichen, wodurch eine native Anwendung widerstandsfähiger gegen Drifts und Bias wäre.

Die wesentlichen Anforderungen an den Prototyp konnten erfüllt werden. Eine abschließende Beurteilung der intuitiven und benutzerfreundlichen Gestaltung des User Interfaces kann zum jetzigen Zeitpunkt noch nicht erfolgen. Um fundierte Aussagen zur User Experience und Benutzerfreundlichkeit treffen zu können, ist eine Validierung durch Nutzerumfragen und umfangreiche Tests notwendig.

Die zentrale Forschungsfrage, wie eine Indoor-Navigation mit Inertial Measurement Units mobiler Endgeräte realisiert werden kann, konnte erfolgreich beantwortet werden. Die Ergebnisse zeigen, dass IMUs in Verbindung mit Algorithmen wie dem Kalman-Filter und Pedestrian Dead Reckoning eine Positionsbestimmung auch ohne GPS-Signale ermöglichen. Eine native Anwendung, die zusätzlich den Madgwick-Algorithmus integriert, könnte die Genauigkeit und Zuverlässigkeit der Sensordaten weiter optimieren und eine robustere Indoor-Navigation gewährleisten.

Weiterer Forschungsbedarf besteht in der Überprüfung und Verbesserung der Genauigkeit der berechneten Position des Nutzers. Dies kann z. B. durch die Integration eines Map-Matching-Algorithmus erfolgen.

## Literaturverzeichnis

### Agafonkin 2024

Agafonkin, Volodymyr: Leaflet – a Javascript library for interactive maps. <https://leafletjs.com/>, 2024, Abruf am 27. September 2024.

### Av 2023

Av, Minhaz: html5-qrcode. A cross platform HTML5 QR code reader. <https://github.com/mebjas/html5-qrcode>, 2023, Abruf am 27. September 2024.

### Borovica-Gajic et al. 2020

Borovica-Gajic, Renata; Qi, Jianzhong; Wang, Weiqing (Hrsg.): Databases Theory and Applications. 31<sup>st</sup> Australasian Database Conference, ADC 2020, Melbourne, VIC, Australia, February 3 – 7, 2020, Proceedings. Melbourne 2020.

### Chao et al. 2020

Chao, Pingfu; Xu, Yehong; Hua, Wen; Zhou, Xiaofang: A Survey on Map-Matching Algorithms. In: Borovica-Gajic et al. 2020, S. 121 - 133.

### Colle 2023

Colle, Pierre: kalman-filter. Kalman filter in javascript. <https://github.com/piercus/kalman-filter>, 2023, Abruf am 27. September 2024.

### Dalton 2016

Dalton, John-David: Lodash. A modern JavaScript utility library delivering modularity, performance & extras. <https://lodash.com/>, 2016, Abruf am 27. September 2024.

### DIN 2020

DIN Deutsches Institut für Normung e. V.: EN ISO 9241-210:2019 (D): Ergonomie der Mensch-System-Interaktion – Teil 210: Menschzentrierte Gestaltung interaktiver Systeme (ISO 9241-210:2029). 2. Aufl. Berlin 2020.

### Ebert 2022

Ebert, Christof: Systematisches Requirements Engineering. Anforderungen ermitteln, dokumentieren, analysieren und verwalten. 7. Aufl., Heidelberg 2022.

### **European GNSS Agency 2020**

European GNSS Agency: GNSS user technology report. Issue 3 (2020).

### **Google LLC 2024**

Google LLC: Google Maps – Transit & Essen (Version 6.130.1) [Mobile App]. AppStore. <https://apps.apple.com/de/app/google-maps-transit-essen/id585027354>, 2024, Abruf am 27. August 2024.

### **ISO/IEC 2023**

ISO/IEC: ISO/IEC 25010:2023(E). Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model. 2. Aufl. Vernier, Genf 2023.

### **Jahns 2013**

Jahns, Robert: Untersuchung und Optimierung von Empfindlichkeit und Rauschverhalten magnetoelektrischer Sensoren, [https://macau.uni-kiel.de/servlets/MCRFileNodeServlet/dissertation\\_derivate\\_00004908/Dissertation\\_Robert\\_Jahns.pdf](https://macau.uni-kiel.de/servlets/MCRFileNodeServlet/dissertation_derivate_00004908/Dissertation_Robert_Jahns.pdf), 2013, Abruf am 23. September 2024.

### **JGraph Ltd 2024**

JGraph Ltd: draw.io – Diagramming Tool (Version 24.7.16) [Web App]. draw.io. <https://www.draw.io>, 2024, Abruf am 28. August 2024.

### **Jiang et al. 2022**

Jiang, Changhui; Chen, Yuwei; Chen, Chen; Jia, Jianxin; Sun, Haibin; Wang, Tinghuai; Hyypä, Juha: Implementation and performance analysis of the PDR/GNSS integration on a smartphone. <https://link.springer.com/article/10.1007/s10291-022-01260-0>, 2022, Abruf am 23. September 2024.

### **Köster 2018**

Köster, Uwe: Entwicklung eines MEMS-basierten Low-Cost-Sensors für geodätische Überwachungsmessungen. [https://digibib.hs-nb.de/resolve/id/dbhsnb\\_thesis\\_0000002243](https://digibib.hs-nb.de/resolve/id/dbhsnb_thesis_0000002243), 2018, Abruf am 20. September 2024.

**Kuang et al. 2018**

Kuang, Jian; Niu, Xiaoji; Chen, Xingeng: Robust Pedestrian Dead Reckoning Based on MEMS-IMU for Smartphones. <https://www.mdpi.com/1424-8220/18/5/1391>, 2018, Abruf am 24. September 2024.

**Külls 2016**

Külls, Robert: Systematik eines Beschleunigungssensor-Designkonzepts auf MEMS-Basis. <https://www.bookshop.fraunhofer.de/buch/systematik-eines-beschleunigungssensor-designkonzepts-auf-mems-basis/243929>, 2016, Abruf am 20. September 2024.

**Madgwick 2021**

Madgwick, Sebastian: Madgwick Orientation Filter. <https://ahrs.readthedocs.io/en/latest/filters/madgwick.html>, 2021, Abruf am 24. September 2024.

**Michaelson 2018**

Michaelson, Tobias: Lagebestimmung durch Sensorfusion mittels Kalman-filter. [https://reposit.haw-hamburg.de/bitstream/20.500.12738/8460/1/Masterarbeit\\_Tobias\\_Michaelson.pdf](https://reposit.haw-hamburg.de/bitstream/20.500.12738/8460/1/Masterarbeit_Tobias_Michaelson.pdf), 2018, Abruf am 23. September 2024.

**Möllmann 2014**

Möllmann, Sebastian: Integration einer Inertial Measurement Unit in die autonome Fahrzeugplattform CampusBot. <https://reposit.haw-hamburg.de/handle/20.500.12738/6743>, 2018, Abruf am 23. September 2024.

**Mozilla 2024a**

Mozilla Corporation (Gesellschaft nach US-amerikanischem Recht): Sensor-APIs. Browser compatibility. [https://developer.mozilla.org/en-US/docs/Web/API/Sensor\\_APIs#browser\\_compatibility](https://developer.mozilla.org/en-US/docs/Web/API/Sensor_APIs#browser_compatibility), 2024, Abruf am 27. September 2024.

**Mozilla 2024b**

Mozilla Corporation (Gesellschaft nach US-amerikanischem Recht): DeviceMotionEvent. Browser compatibility. [https://developer.mozilla.org/en-US/docs/Web/API/DeviceMotionEvent#browser\\_compatibility](https://developer.mozilla.org/en-US/docs/Web/API/DeviceMotionEvent#browser_compatibility), 2024, Abruf am 27. September 2024.

**Mozilla 2024c**

Mozilla Corporation (Gesellschaft nach US-amerikanischem Recht): DeviceOrientationEvent. Browser compatibility. [https://developer.mozilla.org/en-US/docs/Web/API/DeviceOrientationEvent#browser\\_compatibility](https://developer.mozilla.org/en-US/docs/Web/API/DeviceOrientationEvent#browser_compatibility), 2024, Abruf am 27. September 2024.

**Oriient New Media Ltd. 2023**

Oriient New Media Ltd.: Oriient Retail Overview v3.6 (Präsentation). Köln 2023.

**Otto et al. 2024**

Otto, Mark; Thornton, Jacob; Lauke, Patrick; Déramond, Julien; Poupard, Gaël; Sharma, Rohit; Cuppens, Martijn; Mazovetskiy, Gleb: Bootstrap · The most popular HTML, CSS, and JS library in the world. <https://getbootstrap.com/>, 2024, Abruf am 27. September 2024.

**Risk 2024**

Risk, Kiro: Fuse.js. Powerful, lightweight fuzzy-search library, with zero dependencies. <https://www.fusejs.io/>, 2024, Abruf am 27. September 2024.

**Schnabel 2017**

Schnabel, Patrick: MEMS - Micro-Electro-Mechanical Systems. <https://www.elektronik-kompodium.de/sites/bau/1503041.htm>, 2017, Abruf am 27. September 2024.

**Statista 2023**

Statista GmbH: Ranking der Top 15 beliebtesten Smartphone-Apps in den USA nach Reichweite im Oktober 2023. <https://de.statista.com/statistik/daten/studie/238358/umfrage/reichweite-der-beliebtesten-smartphone-apps-in-den-usa/>, 2023, Abruf am 28. Februar 2024.

**Valenti et al. 2015**

Valenti, Roberto; Dryanovsji, Ivan; Xiao, Jizhong: Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs. <https://www.mdpi.com/1424-8220/15/8/19302>, 2015, Abruf am 25. September 2024.

**Vasquez 2004**

Vasquez, Daniel: Wireless Zero-Power Ferromagnetic MEMS Magnetometer. [https://www.researchgate.net/publication/255663586\\_Wireless\\_Zero-Power\\_Ferromagnetic\\_MEMS\\_Magnetometer](https://www.researchgate.net/publication/255663586_Wireless_Zero-Power_Ferromagnetic_MEMS_Magnetometer), 2004, Abruf am 24. September 2024.

**Watson 2016**

Watson, Jeff: MEMS Gyroscope Provides Precision Inertial Sensing in Harsh, High Temperature Environments. <https://www.analog.com/en/resources/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html>, 2016, Abruf am 25. September 2024.

**Welch / Bishop 2002**

Welch, Greg; Bishop, Gary: An Introduction to the Kalman Filter. [https://www.mit.edu/course/16/16.070/www/project/PF\\_kalman\\_intro.pdf](https://www.mit.edu/course/16/16.070/www/project/PF_kalman_intro.pdf), 2002, Abruf am 25. September 2024.

## Anhang

### Programmcode auf Github:




<https://github.com/danielgilbers/indoor-navigation>

### Link und QR-Code zur Anwendung:

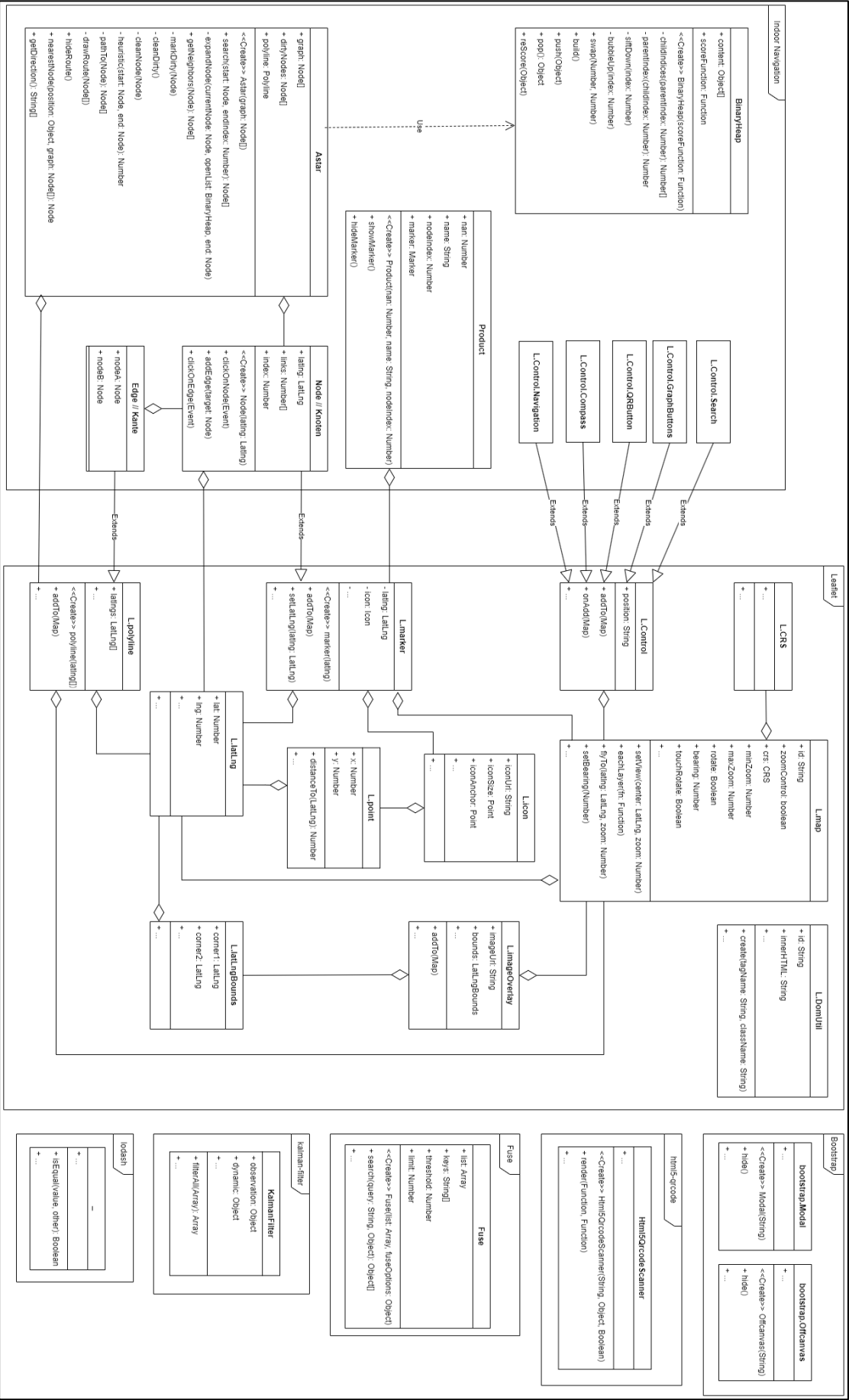
<https://danielgilbers.github.io/indoor-navigation/>



### QR-Codes verschiedener Positionen auf der Karte:

		
Startposition (100, 645)	Variante 1 (55, 500)	Variante 2 (170, 860)

Klassendiagramm:





**KI-Tools & KI-Nutzung**

Kapitel	KI-Tools	Beschreibung der Verwendung
	<a href="https://chatgpt.com/">https://chatgpt.com/</a>	Unterstützung bei Programmierung.
Alle	<a href="https://www.deepl.com/">https://www.deepl.com/</a>	Formulierung einzelner Abschnitte.

## Teilleistungserklärung

<b>Verantwortlich für ... Abschnitte</b>	<b>... den Inhalt (Vor- und Nachname des bzw. der Studierenden)</b>
Inhaltsverzeichnis	<b>Lucas Freier, Maximilian Fröhlich, Daniel Gilbers, Eric Lepp</b>
Abbildungsverzeichnis	<b>Lucas Freier, Maximilian Fröhlich, Daniel Gilbers, Eric Lepp</b>
Tabellenverzeichnis	<b>Lucas Freier, Maximilian Fröhlich, Daniel Gilbers, Eric Lepp</b>
Abkürzungsverzeichnis	<b>Lucas Freier, Maximilian Fröhlich, Daniel Gilbers, Eric Lepp</b>
Kapitel 1	<b>Lucas Freier, Maximilian Fröhlich, Daniel Gilbers, Eric Lepp</b>
Kapitel 2	<b>Maximilian Fröhlich</b>
Abschnitt 2.1	<b>Maximilian Fröhlich</b>
Abschnitt 2.2	<b>Maximilian Fröhlich</b>
Abschnitt 2.3	<b>Maximilian Fröhlich</b>
Kapitel 3	<b>Maximilian Fröhlich</b>
Abschnitt 3.1	<b>Maximilian Fröhlich</b>
Abschnitt 3.2	<b>Maximilian Fröhlich</b>
Abschnitt 3.3	<b>Maximilian Fröhlich</b>
Abschnitt 3.4	<b>Maximilian Fröhlich</b>
Abschnitt 3.5	<b>Maximilian Fröhlich</b>

Kapitel 4	<b>Lucas Freier</b>
Abschnitt 4.1	<b>Lucas Freier</b>
Abschnitt 4.2	<b>Lucas Freier</b>
Abschnitt 4.3	<b>Lucas Freier</b>
Kapitel 5	<b>Daniel Gilbers</b>
Abschnitt 5.1	<b>Daniel Gilbers</b>
Abschnitt 5.2	<b>Daniel Gilbers</b>
Abschnitt 5.3	<b>Daniel Gilbers</b>
Abschnitt 5.4	<b>Daniel Gilbers</b>
Kapitel 6	<b>Eric Lepp</b>
Abschnitt 6.1	<b>Eric Lepp</b>
Abschnitt 6.1.1	<b>Eric Lepp</b>
Abschnitt 6.1.2	<b>Eric Lepp</b>
Abschnitt 6.2	<b>Eric Lepp</b>
Kapitel 7	<b>Lucas Freier</b>
Abschnitt 7.1	<b>Daniel Gilbers</b>
Abschnitt 7.2	<b>Daniel Gilbers</b>
Abschnitt 7.3	<b>Lucas Freier</b>
Abschnitt 7.3.1	<b>Lucas Freier</b>
Abschnitt 7.3.2	<b>Lucas Freier</b>

Fazit	<b>Lucas Freier, Maximilian Fröhlich, Daniel Gilbers, Eric Lepp</b>
Literaturverzeichnis	<b>Lucas Freier, Maximilian Fröhlich, Daniel Gilbers, Eric Lepp</b>
Anhang	<b>Lucas Freier, Maximilian Fröhlich, Daniel Gilbers, Eric Lepp</b>

29. September 2024

Handschriftliche Unterschriften




---

Lucas Freier




---

Maximilian Fröhlich




---

Daniel Gilbers




---

Eric Lepp