HAW HAMBURG

MASTERTHESIS
Pascal Stahr

Navigation eines Assistenzsystems für Blinde auf Basis von geographischen Informationsdaten

FAKULTÄT TECHNIK UND INFORMATIK
Department Maschinenbau und Produktion

Faculty of Engineering and Computer Science
Department of Mechanical Engineering and Production Management

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Pascal Stahr

Navigation eines Assistenzsystems für Blinde auf Basis von geographischen Informationsdaten

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Maschinenbau/Berechnung und Simulation im Maschinenbau
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Henner Gärtner
Zweitprüfer: Prof. Dr. Jochen Maaß

Eingereicht am: 20.07.2021

Pascal Stahr

Thema der Arbeit
Navigation eines Assistenzsystems für Blinde auf Basis von geographischen Informationsdaten

Stichworte
Shared Guide Dog 4.0, mobiles Assistenzsystem, Navigation, Pfadplanung, Open Street Map

Kurzzusammenfassung
In der vorliegenden Masterthesis wird die Pfadplanung und Navigation eines mobilen Assistenzsystems für Blinde auf Basis eines geographischen Informationssystems entwickelt. Für blinde Personen ist die Orientierung und Mobilität eine große Herausforderung, die durch das mobile Assistenzsystem *Shared Guide Dog* minimiert werden soll. Die grundlegende Voraussetzung ist die sichere Lokalisation und Pfadplanung. Im Rahmen dieser Arbeit wird das Augenmerk auf die Pfadplanung gelegt und eine grundlegende Implementierung der Lokalisation durchgeführt. Dazu wird ein Prozess erarbeitet, der die Überprüfung und Verarbeitung der Kartendaten, Pfadplanung anhand der Kartendaten und die Integration auf den Shared Guide Dog beinhaltet.

Es kann gezeigt werden, dass sich die Kartendaten des geographischen Informationssystems *Open Street Map* gut zur Erfüllung der Aufgabe eignen, wobei die Kartendaten vor der Verwendung überprüft und mit weiteren Daten angereichert werden müssen. Auf Basis der Daten wird ein Pfadplanungsalgorithmus implementiert, der unterschiedliche Nutzerprofile sowie Eigenschaften der Wege berücksichtigen kann.

Pascal Stahr

Title of the paper
Navigation of an assistance system for the blind based on geographic information data

Keywords
Shared Guide Dog 4.0, mobile assistance system, navigation, path planning, Open Street Map

Abstract
This master thesis deals with the development of path planning and navigation of a mobile assistance system for blind people based on a geographic information system. Orientation and mobility is a major challenge for blind persons, which is to be minimized by the mobile assistance system called *Shared Guide Dog*. The basic requirement is reliable localization and path planning. This thesis focuses on path planning and performs a basic implementation of localization. For this purpose, a process is developed that includes the verification and processing of map data, path planning based on the map data, and integration on the Shared Guide Dog.

It can be shown that the map data from the geographic information system *Open Street Map* is well suited to accomplish the task, although the map data must be reviewed and enriched with additional data before use. Based on the data, a path planning algorithm is implemented that can take into account different user profiles as well as characteristics of the paths.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Glossar

API	Application Programming Interface. 46
BT	Behaviour Tree. 34f.
CEP	Circular Error Probable (dt. Streukreisradius). 29, 69
DGPS	Differential GPS. 22
DOF	Degrees of Freedom. 25
DOP	Dilution of Precision. 20f.
EKF	Extended Kalman Filter. 68
GIS	Geographisches Informationssystem. 8
GNSS	Global Navigation Satellite System (dt. Globales Navigationssatellitensystem). 19f.
HDOP	Horizontal Dilution of Precision. 21
IMU	Inertial Measurement Unit (dt. Inertiale Messeinheit). 24f., 28ff., 61, 63, 68f., 76
JSON	JavaScript Object Notation. 59
LiDAR	Light Detection And Ranging. 22
MEMS	Micro-Machined Electromechanical System. 26
OSM	Open Street Map. 4ff., 8, 11, 40, 46, 53, 73, 78
REP	ROS Enhancement Proposals. 34
ROS	Robot Operating System. 33
SA	Selective Availability (dt. Selektive Verfügbarkeit). 20
SAPOS	Satellite Positioning System. 22
SBAS	Satellite Based Augmentation System (dt. Satellitenbasiertes Ergänzungssystem). 22, 29
SLAM	Simultaneous Localization and Mapping. 14
TCP/IP	Transmission Control Protocol/Internet Protocol. 59
UERE	User Equivalent Range Errors. 20f.
UWB	Ultra-Wideband. 79

1 Einleitung

Für sehende Menschen sind die Orientierung und individuelle Mobilität in den meisten Fällen keine Schwierigkeit. Um zu wissen, wo sich eine Person befindet, schaut sich diese um und sucht visuelle Hinweise, zum Beispiel Straßenschilder oder öffentliche Gebäude. Auch die Mobilität, also die sichere und selbständige Fortbewegung, ist kein Problem, denn Hindernisse und Unebenheiten des Weges lassen sich meist leicht erkennen.

Einer blinden Person fehlt die Fähigkeit, sich umschauen zu können. Stattdessen muss sie durch andere Sinne und Hilfsmittel ersetzt werden. Viele Blinde orientieren sich mit dem Gehör, dabei ist das Gehör blinder Personen nicht besser entwickelt als das der sehenden Personen. Sie haben nur gelernt die aufgenommenen Sinneseindrücke besser zu verarbeiten und zu deuten. Trotzdem bleiben die Orientierung und Mobilität eine große Herausforderung, in der Stadt wie auf dem Land. [72]

Von vielen Blinden- und Sehbehindertenvereinen werden Schulungen zum Thema Orientierung und Mobilität angeboten. Dabei werden Tricks und Tipps vermittelt und der Umgang mit Hilfsmitteln wie beispielsweise dem Langstock gelehrt. Darüber hinaus können häufig benutzte Wege trainiert werden, wodurch diese allein begangen werden können. Aber diese Strecken erfordern ein ständiges Training, um beispielsweise Veränderungen an den Wegen zu erkennen. Schwierig wird es bei selten benutzten Wegen oder im Urlaub. Dort sind eine Orientierung und sichere Fortbewegung nur zusammen mit einer sehenden oder ortskundigen Person möglich. Aber auch in der Umgebung um den Wohnort lauern viele Gefahren. Große Kreuzungen stellen ein besonderes Problem dar, vor allem wenn die Ampelanlage nicht mit einem akustischen Signal ausgestattet ist. Dann bleibt oft nur die Möglichkeit zu hören, ob die Autos fahren oder ob sie stehen und dann zu gehen. Das ist nicht nur gefährlich, sondern fordert von der blinden Person auch die volle Aufmerksamkeit.

Als Hilfsmittel kommt in den meisten Fällen ein Langstock zum Einsatz. Er wird vor dem Körper geschwenkt und erlaubt dem Nutzer oder der Nutzerin die Struktur des Weges und Hindernisse zu erkennen. Außerdem können damit zum Beispiel Ampeln gefunden werden. Ein weiteres Hilfsmittel ist ein Blindenführhund. Ein Blindenführhund wird speziell ausgebildet, um die blinde Person sicher zu führen. Zudem kann er auf Kommando bestimmte Ziele finden, die jedoch vorher trainiert werden müssen. Von der Antragstellung bis zum fertig ausgebildeten Blindenführhund vergehen jedoch schnell 1,5 bis 2 Jahre und die Kosten können nur bei einem Restsehvermögen von unter 5 % von der Krankenkasse übernommen werden [21, S. 48 f.]. Daneben werden immer mehr elektronische Hilfsmittel eingesetzt, wie die OrCam, die Texte im Alltag vorlesen kann [56], und Apps auf dem Smartphone, zum Beispiel die Navigationsapp Routago, die speziell für Blinde entwickelt wurde [15]. Alle diese Hilfsmittel lösen einzelne Problem, die bei der Orientierung und Mobilität auftreten. Eine Hilfe, wie sie der Blindenführhund in der bekannten Umgebung bieten kann, gibt es jedoch nicht.

Das Projekt *Shared Guide Dog 4.0* hat das Ziel, diese Lücke zu schließen. Dafür wird ein mobiler Roboter, der sogenannte *Shared Guide Dog*, entwickelt, der sich für den Nutzer oder die Nutzerin wie ein Blindenführhund verhalten und die sichere Orientierung und Mobilität in bekannten wie auch unbekannten Gegenden ermöglichen soll.

Der Name des Projekts setzt sich aus den Begriffen „Shared“, „Guide Dog“ und „4.0“ zusammen. „Shared“ steht für die gemeinsame Nutzung eines Fahrzeugs von mehreren Nutzern, ähnlich wie es bei Fahrrädern oder Autos in vielen deutschen Städten bereits möglich ist. Der Begriff „Guide Dog“ bezeichnet einen Blindenführhund. Ein Blindenführhund bietet derzeit von allen Hilfsmitteln die umfassendste Hilfe bei der Orientierung und Mobilität an. Eine ähnliche Hilfe soll auch der Shared Guide Dog ermöglichen. Die „4.0“ steht für die Vernetzung und den Datenaustausch von Geräten, angelehnt an die Industrie 4.0. Durch die Vernetzung ist es möglich, mehrere Shared Guide Dogs in einem großen Gebiet fahren zu lassen, die sich gegenseitig vor Gefahren warnen.

1.1 Projektplan

Das Projekt gliedert sich in fünf Projektphasen wie in Abbildung 1.1 gezeigt. Zu Beginn wurde eine Anforderungsanalyse durchgeführt. Bereits in diesem Schritt wurde mit einigen blinden Menschen zusammengearbeitet, um die Bedürfnisse und Vorbehalte zu verstehen. Aus den Anforderungen wurden in der zweiten Projektphase unterschiedliche Mock-Ups entwickelt. Darunter waren eine mit Sensorik ausgestattete Weste, die durch Vibrationen vor Hindernissen warnt, und ein selbstfahrender Rollstuhl auf Basis eines Fahrerlosen Transportfahrzeugs. Bei der Erprobung der Mock-Ups hat sich ein intelligenter Rollator, der die Umgebung erfassen und den Nutzer oder die Nutzerin daraufhin steuern kann, als vielversprechendstes Design herausgestellt. Mit diesem Design wurde die dritte Projektphase „Entwicklung“ gestartet. Das Projekt befindet sich zum Zeitpunkt dieser Arbeit in der Entwicklungsphase. Ziel dieser Phase ist die Entwicklung der Komponenten für einen autonom fahrenden Shared Guide Dog. Diese Phase geht fließend in die Integrationsphase über, in der die entwickelten Komponenten in funktionsfähige Prototypen integriert werden. Den Abschluss des Projekts bildet die Überführung in die Lehre. In dieser Phase werden die Projektergebnisse abschließend dokumentiert und die Nutzung in der Lehre vorbereitet.

Abbildung 1.1: Plan des Projekts Shared Guide Dog 4.0 (Quelle: eigene Darstellung)

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung und Umsetzung eines Navigationssystems, das die Daten aus einem geographischen Informationssystem zur Navigation eines mobilen Assistenzroboters im innerstädtischen Umfeld nutzt. Für die Umsetzung des Navigationssystems sind die Entwicklung einer Kartendatenverwaltung, die Auswahl einer internen Datenrepräsentation und die Implementierung eines Pfadplanungsalgorithmus erforderlich. Ziel des Pfadplanungsalgorithmus ist die Berechnung einer optimalen Route abhängig vom Nutzer oder der Nutzerin und den Straßeneigenschaften, sodass sich zum Beispiel Straßen mit Kopfsteinpflaster vermeiden lassen. Zum Abschluss der Arbeit soll das Navigationssystem auf einem Prototyp des *Shared Guide Dog* umgesetzt werden und eine Fahrt durch den Lohmühlenpark demonstriert werden.

1.3 Projektumgebung an der HAW Hamburg

Neben dem Projekt *Shared Guide Dog 4.0* gibt es an der HAW Hamburg noch weitere Projekte, die sich mit der innerstädtischen Mobilität beschäftigen. Diese Projekte sind unter dem Namen *Test Area Intelligent Quatier Mobility (TIQ)* zusammengefasst. [58] Eines dieser Projekte ist das Projekt *Sensorknoten*, das sich mit der Entwicklung von Multi-Sensor Systemen beschäftigt, um Informationen zur Mobilität im Sensorumfeld zu erhalten. Zum Beispiel soll die Unterscheidung von Fußgängern und Fahrradfahrern möglich sein, sowie deren Bewegung zu verfolgen. Neben der Beobachtung bietet ein solcher Sensorknoten eine Infrastruktur zur Kommunikation autonomer Kleinstfahrzeuge und zur Lokalisierung und Verfolgung dieser Fahrzeuge. Damit soll der Verkehrsraum für alle Verkehrsteilnehmer sicherer gestaltet werden. [5, S. 3 f.]

Mit der Entwicklung eines autonomen mobilen Roboters beschäftigt sich das Projekt *Husky*. Der Husky soll autonom über den Campus Berliner Tor der HAW Hamburg fahren und dort kleine Aufgaben, wie das Abliefern von Paketen, erfüllen können. [31]

Alle Daten der entwickelten Roboter werden von den Sensorknoten gesammelt und verarbeitet. Anschließend können die Daten den mobilen Systemen, dazu gehört auch der Shared Guide Dog, wieder zur Verfügung gestellt werden. Ein Ziel des Sensorknoten-Projekts ist, die Datenverarbeitung auf einige stationäre Systeme zu verlagern und viele mobile Plattformen mit einfacher Hardware einsetzen zu können.

Für den Shared Guide Dog wurde der Lohmühlenpark am Campus Berliner Tor der HAW Hamburg als Testgebiet ausgewählt. In Abbildung 1.2 ist der Lohmühlenpark beispielhaft abgebildet. Im Park gibt es mehrere große Grünflächen, einen Spielplatz und ein Basketballfeld. Am nördlichen und südlichen Ende des Testgebiets sind die U-Bahn-Haltestellen Lohmühlenstraße und Berliner Tor zu finden. Der Park ist von mehreren Wegen durchzogen, von denen ungefähr die Hälfte gepflastert sind. Die andere Hälfte besteht aus Kopfsteinpflaster und festen Sandwegen. Der Park wird jeden Tag von vielen Menschen zu Fuß und auf dem Fahrrad besucht und durchquert. Der Autoverkehr ist im Park nicht erlaubt, lediglich Sonderfahrzeuge, wie die Müllabfuhr, die Stadtreinigung oder die

Feuerwehr und Polizei, dürfen durch den Park fahren. Das Gebiet wurde ausgewählt, da es deutlich weniger Gefahren als an einer stark befahrenen Straße birgt.

Abbildung 1.2: Beispielhafte Bilder des Testgebiets im Lohmühlenpark (Quelle: eigene Darstellung)

1.4 Vergleichbare Projekte

Mit der fortschreitenden Entwicklung von autonom fahrenden Fahrzeugen, rückt die Kombination von lokalen Sensordaten und einer globalen Umgebungsrepräsentation durch eine vorgegebene Karte immer mehr in den Fokus der Forschung. Als Umgebungsrepräsentation hat sich dabei in vielen Forschungsprojekten (vgl. [16], [1], [64], [12]) die Open Street Map (OSM) (vgl. Kapitel 2.2) als nützliche Umgebungsrepräsentation herausgestellt. Dieser Abschnitt gibt einen Überblick über einige der Projekte und deren Verwendung von Open Street Map.

In [16] wird die Open Street Map zur Lokalisierung, Pfadplanung und Steuerung eines autonomen, mobilen Roboters genutzt. Als mobiler Roboter kommt ein straßenzugelassenes Kawasaki Mule 3010 Fahrzeug zum Einsatz. Die Größe ist in etwa vergleichbar mit der eines Golfcarts. Die Lokalisierung wird mit einer Kombination aus GPS und Lidar auf Basis der Gebäudeumrisse aus der Karte durchgeführt. Für die Pfadplanung werden die Straßendaten verwendet, um die schnellste oder kürzeste Route zu finden. Darüber hinaus werden auf Grundlage der Kartendaten Lichter, wie etwa Blinker beim Abbiegen, gesteuert. Bei dem vorgestellten Ansatz werden die OSM Kartendaten vor der Verwendung nicht angepasst oder verändert. [16, S. 2 ff.]

In [64] wird ein Ansatz vorgestellt, bei dem Kartendaten von Open Street Map für die globale Navigation eines mobilen Outdoor-Roboters genutzt werden. Ziel des Ansatzes ist eine robuste Lokalisierung und Pfadplanung zu implementieren. Die Lokalisierung wird auf Basis von GPS-Daten durchgeführt. Anschließend findet eine Korrektur der Position durch das Versetzen auf den nächstgelegenen Weg statt. Die Wegerkennung wird mit einem 3D-Lidarsensor durchgeführt, der eine Klassifizierung der Umgebung vornimmt. Dieser Ansatz wird während der gesamten Strecke angewendet und erlaubt es, Fehler im GPS-Signal oder in der Karte auszugleichen. [64]

Floros, van der Zander und Leibe [12] stellen in ihrem Ansatz eine Methode zur globalen Lokalisierung eines PKW vor. Der Ansatz nutzt die Informationen eines Kamerasystems und kombiniert diese mit den Kartendaten aus Open Street Map, um die Position des Fahrzeugs zu bestimmen. Für die Lokalisierung wird ein Monte-Carlo Algorithmus eingesetzt. [12]

Das Projekt beschrieben in [1] legt den Fokus auf die Verarbeitung der OSM Kartendaten, um diese für Outdoor-Roboter aufzubereiten. Das beschriebene Beispielszenario beinhaltet einen Roboter, der kleine Pakete liefern kann. Zur Verarbeitung der Daten wird eine Toolchain entwickelt, die die Aufgaben Datenbeschaffung, Preprocessing, Kreuzungserkennung, Polygonbearbeitung, Netzgenerierung und Evaluation beinhaltet. Die Umsetzung auf einem mobilen Roboter findet nicht statt. [1]

Die Kartendaten der OSM werden in den vorgestellten Forschungsprojekten mit unterschiedlichen Schwerpunkten genutzt. In [64] werden die Daten lediglich zur globalen Orientierung genutzt und die lokale Orientierung wird mit einem 3D-Lidarsensor durchgeführt. Die lokale Orientierung erfolgt nicht anhand der Kartendaten. In [16] und [12] werden die Kartendaten hingegen auch zur lokalen Orientierung des mobilen Roboters genutzt. Bei den verwendeten Fahrzeugen handelt es sich jedoch um straßenzugelassene Fahrzeuge, die keine Lokalisierung im Zentimeterbereich erfordern. Der Ansatz in [1] setzt den Schwerpunkt auf die Verarbeitung der Kartendaten, um diese anschließend für einen Lieferroboter nutzen zu können. Das verwendete Fahrzeug entspricht dem Shared Guide Dog in der Größe, es findet jedoch keine Integration in einen mobilen Roboter statt.

Für die Lokalisierung werden in allen Arbeiten GPS-Sensoren mit einem weiteren System kombiniert, da ein GPS-Empfänger allein keine Position mit ausreichender Genauigkeit liefern kann. Zum Einsatz kommen Lidar-Systeme (vgl. [16], [64]) oder ein Kamerasystem (vgl. [12]). Die Lokalisation erfolgt anschließend mit einem Monte-Carlo Algorithmus. Damit unterscheiden sich die vorgestellten Projekte von dieser Arbeit, in der die Lokalisation mit einem GPS-Empfängers, einer IMU und den Odometriedaten durchgeführt werden soll.

1.5 Aufbau der Arbeit

Navigation besteht aus den drei Teilaufgaben Lokalisation, Pfadplanung und Steuerung. Die Kapitel 2, 3 und 4 geben eine technische Einführung in diese drei Aufgabengebiete. In Kapitel 5 wird eine Anforderungsanalyse durchgeführt. Die entwickelten Anforderungen werden für die Erstellung eines Lösungsansatzes in Kapitel 6 genutzt. Kapitel 7 beschäftigt sich mit der Umsetzung der Navigationslösung. Dazu gehören eine kritische Betrachtung der Kartendaten und Entwicklung eines Pfadplanungsalgorithmus, der die Anforderungen des Shared Guide Dogs erfüllen kann. In Kapitel 8 wird die Lösung in den Shared Guide Dog integriert. Eine Überprüfung der entwickelten Lösung findet in Kapitel 9 statt. Die Arbeit endet mit einem Fazit und Ausblick in Kapitel 10.

2 Grundlagen der Pfadplanung

In diesem Kapitel soll zunächst ein Überblick über verschiedene Typen von Karten gegeben werden, die eine Repräsentation der Umgebung darstellen. Anschließend folgt eine detaillierte Betrachtung der Kartendaten aus dem geographischen Informationssystem Open Street Map (OSM). Den Abschluss für dieses Kapitel bildet eine Einführung in verschiedene Pfadplanungsalgorithmen.

2.1 Roboterkarten

Die Wahl der internen Karte ist ein wichtiger Baustein bei der Entwicklung eines mobilen Roboters. Die Karte wird für die Lokalisation des Roboters in der Umgebung und die sichere und robuste Pfadplanung zu einem Ziel genutzt [18, S. 156]. Dabei müssen drei Herausforderungen betrachtet werden. Die Karte soll möglichst kompakt gespeichert werden, damit sie effizient von anderen Komponenten genutzt werden kann. Darüber hinaus muss sie an die Aufgabe und die Umgebung angepasst sein. Eine Karte, die darauf ausgelegt ist, strukturierte Umgebungen abzubilden, kann nicht in einer unstrukturierten Umgebung eingesetzt werden. Zuletzt muss die Karte die Repräsentation von Sensor- und Lokalisierungsungenauigkeiten ermöglichen. [61, S. 853]

Nach [18] lassen sich die Karten in metrisch kontinuierliche, metrisch diskrete und topologische Karten unterteilen. Metrisch kontinuierliche Karten speichern die Daten in der Genauigkeit, mit der sie aufgenommen wurden oder mit der vom Computer maximal speicherbaren Genauigkeit, sofern diese geringer ist. Der Raum, in dem die Koordinaten gespeichert werden, kann als kontinuierlich angesehen werden. Metrisch diskrete Karten speichern Objekte und Hindernisse nur in bestimmten Punkten. Der Raum wird damit diskretisiert. Topologische Karten speichern die Daten in einem kontinuierlichen Raum, stellen jedoch die Beziehungen der Objekte untereinander in den Vordergrund. Ein Beispiel ist eine Straßenkarte, bei der Objekte wie Kreuzungen durch Beziehungen, zum Beispiel Straßen und Wege, verbunden werden. [18, S. 156 ff.]

Bereits seit den 1980er Jahren werden Rasterkarten (engl. Occupancy Grids) als Repräsentation der Umgebung eingesetzt. Rasterkarten bestehen im einfachsten Fall aus einem Gitter, dessen Zellen mit den Werten 0 oder 1 gefüllt werden. Steht eine 0 in der Zelle, ist die Zelle frei von Hindernissen, eine 1 steht für eine belegte Zelle. Der Nachteil dieses Ansatzes ist die fehlende Möglichkeit Sensorungenauigkeiten in der Karte zu repräsentieren. Eine Zelle kann nur belegt oder nicht belegt sein. Eine Weiterentwicklung dieses Ansatzes besteht darin, die Zellen mit Belegungswahrscheinlichkeiten zu füllen. Die Karte wird deutlich flexibler und kann auch in Umgebungen mit dynamischen Hindernissen eingesetzt werden.

Eine Rasterkarte gehört zur Gruppe der diskreten Karten. Die Vorteile dieser Karte sind, dass keine Features benötigt werden, aus denen die Karte aufgebaut ist, der Zugriff auf die Kartenzellen in einer konstanten Zeit möglich ist und nicht besuchte Regionen gut dargestellt werden können. Nachteilig ist, dass die Karte einen hohen Speicherbedarf hat und durch den Diskretisierungsfehler Einbußen in der Genauigkeit nicht zu vermeiden sind. Rasterkarten können sowohl in strukturierten wie auch in unstrukturierten Umgebungen als zweidimensionale oder dreidimensionale Karten eingesetzt werden. [61, S. 855]

Der Nachteil des hohen Speicherbedarfs kann durch die Verwendung von Quadtrees im zweidimensionalen und Octrees im dreidimensionalen Raum verringert werden. Die Idee bei der Erstellung von Quadtrees und Octrees ist, dass Karten adaptiv anhand der Merkmale im Raum unterteilt werden. Bereiche, in denen keine Merkmale vorhanden sind, die leer sind oder noch nicht exploriert wurden, werden durch einige wenige große Zellen repräsentiert. Sind in einem Bereich viele Merkmale vorhanden, verringert sich die Zellgröße und damit auch der Diskretisierungsfehler. [18, S. 165 f.]

Eine Quadtree-Rasterkarte ist in Abbildung 2.1 auf der linken Seite gezeigt. Die Erstellung erfolgt nach [18, S. 165 f.]. Im ersten Schritt wird zunächst eine große Zelle über die gesamte Karte gelegt. Enthält diese Zelle Merkmale, wird sie in vier gleichmäßige Zellen zerlegt. Dieser Vorgang wird so lange durchgeführt bis eine vorgegebene kleinste Zellgröße erreicht ist. In diesem Beispiel ist die gewählte Zellgröße groß und damit ist auch der Diskretisierungsfehler groß. Der Vergleich der ursprünglichen Geometrien (in rot dargestellt) und der belegten Zellen zeigt, dass insbesondere Strukturen, die eine andere Orientierung als das Raster haben, einen großen Diskretisierungsfehler besitzen. [18, S. 165 f.]

Die zweite betrachtete Kartenrepräsentation sind Linienkarten (engl. line maps). Die Merkmale werden durch Linien mit Koordinaten repräsentiert und sind damit nicht wie bei Rasterkarten an ein Gitter gebunden. Es handelt sich bei Linienkarten um kontinuierliche Karten. Sie haben die Vorteile, dass sie keinen Diskretisierungsfehler besitzen und dass sie einen geringeren Speicherbedarf haben. Die Repräsentation großer Gebiete gelingt mit Linienkarten besser als mit Rasterkarten. Ein Nachteil ist der höhere Berechnungsaufwand bei der Erstellung der Karte, da die Linien aus den Messpunkten errechnet werden müssen. Linienkarten eignen sich nur bedingt für unstrukturierte Umgebungen, da alle Hindernisse durch Linien angenähert werden. [61, S. 857] [18, S. 161 f.]

In Abbildung 2.1 ist auf der rechten Seite eine beispielhafte Linienkarte dargestellt. Sie wurde in Anlehnung an den Split-and-Merge Algorithmus von Douglas and Peucker, wie in [61, S. 857 f.] beschrieben, erstellt. Die Idee des Algorithmus ist, dass eine Reihe von Messpunkten durch eine Linie angenähert wird. Anschließend wird die Distanz zwischen dem am weitesten entfernt liegenden Punkt und der Linie berechnet. Ist die Distanz kleiner als ein Schwellenwert, terminiert der Algorithmus, ist sie größer, wird die Punktemenge an dieser Stelle geteilt und beide Teilmengen durch eine Linie angenähert. Dieser Vorgang wird so lange durchgeführt, bis kein Punkt weiter als der Schwellenwert entfernt liegt. Deutlich zu sehen ist dieser Vorgang an dem ehemals runden Hindernis in der rechten unteren Ecke. Der Algorithmus terminiert mit einer Annäherung des Kreises durch ein Oktagon. [61, S. 857 f.]

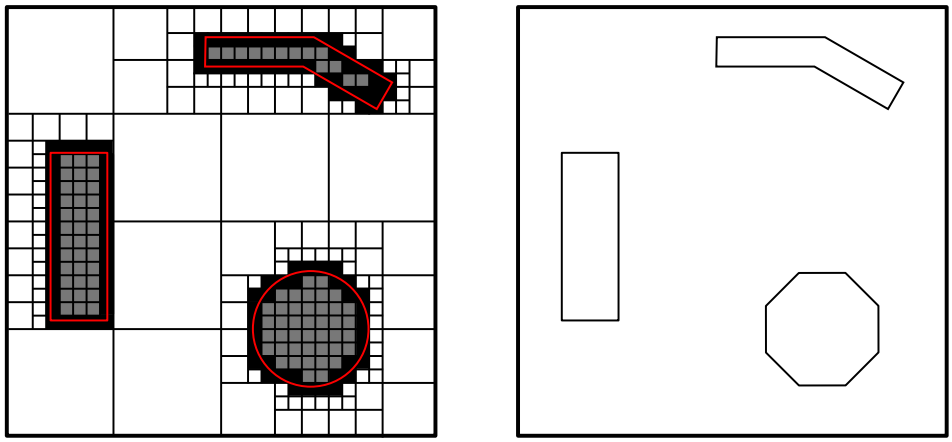

Abbildung 2.1: Beispielhafte Darstellung einer Rasterkarte und einer Linienkarte (Quelle: eigene Darstellung)

2.2 Geographische Informationssysteme

Als Geographisches Informationssystem (GIS) wird ein System bezeichnet, das geographische Daten erfassen und verwalten kann. Darüber hinaus werden diese Systeme zur Analyse von räumlichen Zusammenhängen von Daten verwendet. [8] Bekannte GIS sind beispielsweise *Google Maps*, *Bing Maps*, *HERE* und *OSM*. Die meisten dieser GIS haben jedoch den Nachteil, dass die Kartendaten zwar für den Endnutzer kostenfrei genutzt werden können, die Daten jedoch nicht für eigene Zwecke verändert werden dürfen. Lediglich OSM ist unter einer Lizenz lizensiert, die die kostenfreie Nutzung der Karten und der Daten erlaubt. [43]

Im Open Street Map Projekt wird eine kostenlose, für jeden frei zugängliche Weltkarte erstellt. Die Karte wird durch viele Freiwillige auf der ganzen Welt erstellt. Im Rahmen des Projektes sind viele Softwarelösungen entstanden, die jedoch nicht Ziel des Projektes sind, sondern nur als Unterstützung dienen. Die Kartendaten von OSM haben sich bereits in verschiedenen Forschungsprojekten als nützlich herausgestellt und werden auch von kommerziellen Transportunternehmen wie Uber und Grab verwendet. Dabei nutzt Grab die Daten nicht nur, sondern nimmt auch neue Daten auf und fügt diese zur OSM hinzu. [44] [52]

2.2.1 Datenrepräsentation

Die Open Street Map ist aus den drei Grundelementen Knoten (engl. nodes), Wegen (engl. ways) und Relationen (engl. relations) aufgebaut. Das Speichern von Informationen und Eigenschaften der drei Grundelemente erfolgt mit Attributen (engl. tags), die zu den Elementen hinzugefügt werden können. OSM Kartendaten werden im XML-Schema gespeichert. Mit diesem Aufbau gehört die OSM zur Kategorie der topologischen Karten.

Knoten sind das Kernelement von Open Street Map. Jeder Knoten repräsentiert einen Punkt auf der Erdoberfläche. Zur vollständigen Beschreibung eines Knotens werden die Position als Längen- und Breitengrad und eine Nummer zur eindeutigen Identifikation benötigt. Der Längen- und Breitengrad

werden als Dezimalzahl mit 7 Nachkommastellen Genauigkeit angegeben. Das ermöglicht eine Positionsangabe mit einer Abweichung von weniger als $\pm 5{,}6\,\text{mm}$ (vgl. 3.2.1). Weitere Eigenschaften wie zum Beispiel die Höhe werden durch das Hinzufügen weiterer Attribute angegeben. Beispielhaft ist im folgenden Codeblock die Definition eines Knotens angegeben. Ein Knoten wird durch das XML-Element mit dem Namen *node* eingeleitet. Innerhalb dieses Elements werden die Eigenschaften angegeben, die den Knoten definieren. Es müssen mindestens der Längen- und Breitengrad und die ID angegeben werden, wobei die ID durch das System vergeben wird. Darüber hinaus ist die Angabe von optionalen Eigenschaften möglich, wie in diesem Beispiel die Sichtbarkeit des Knotens (Z. 1: *visible*). [47]

Das Knotenelement kann weitere Elemente beinhalten, die Eigenschaften über den Knoten speichern. Diese Elemente werden als Attribute bezeichnet und mit dem Namen *tag* eingeleitet. Ein Attribut besteht immer aus einem Schlüssel (engl. key) und einem Wert (engl. value). Entsprechend der englischen Bezeichnung werden die Schlüssel mit *k* und die Werte mit *v* identifiziert. Prinzipiell können beliebige Bezeichnungen als Schlüssel und Werte verwendet werden. Um das Arbeiten mit Open Street Map und die Darstellung der Attribute zu vereinfachen, gibt es jedoch Leitfäden zur Bezeichnung von Objekteigenschaften (vgl. [45]). Die freie Wahl der Bezeichnung birgt das Risiko, dass gleiche Objekte mit unterschiedlichen Tags gekennzeichnet werden und somit nur aufwendig automatisiert erkannt werden können. [47] [54]

```
1 <node id='253416079' visible='true'  lat='53.5571377' lon='10.0197035'>
2     <tag k='check_date:wheelchair' v='2021-03-24' />
3     <tag k='name' v='Lohmühlenstraße' />
4     <tag k='railway'  v='subway_entrance' />
5     <tag k='ref'  v='3' />
6     <tag k='wheelchair' v='no' />
7 </node>
```

Werden mehrere Knoten verbunden, handelt es sich um einen Weg, das zweite Grundelement von Open Street Map. Es wird zwischen offenen und geschlossenen Wegen unterschieden. Bei geschlossenen Wegen sind der erste und letzte Punkt identisch. Mit geschlossenen Wegen lassen sich Wege mit einem gemeinsamen Start- und Endpunkt, beispielsweise Kreisverkehre, und Flächen darstellen. Stellt ein Weg eine Fläche dar, wird dies mit dem Attribut *area* gekennzeichnet. Das Einsatzgebiet von Wegen ist sehr vielfältig. Es ist naheliegend, dass jede Straße und jeder Weg durch einen OSM-Weg repräsentiert wird. Es können jedoch auch Baumreihen, Gebäudeumrisse, Flüsse und viele weitere Strukturen durch Wege repräsentiert werden. [55]

Die Unterscheidung, welches Objekt einen Weg darstellt, wird durch Attribute angegeben. Der folgende Codeblock zeigt beispielhaft die Definition eines Weges. Ein Weg wird durch das XML-Element mit dem Namen *way* eingeleitet. Um einen validen Weg zu erzeugen, müssen mindestens zwei Knoten angegeben werden. Hierzu wird das XML-Element mit dem Namen *nd* (Abkürzung für node) mit dem Schlüssel *ref* verwendet. Der Wert von *ref* verweist auf die ID des Knotens. Es ist

möglich einen Knoten zu mehreren Wegen hinzuzufügen, sodass sich kreuzende Wege entstehen. Ein Weg kann aus maximal 2000 Knoten bestehen. [55]

```
<way id='5229522' visible='true' changeset='100256842'>
    <nd ref='26678245' />
    <nd ref='7156477920' />
    <nd ref='26678243' />
    <nd ref='775863169' />
    <tag k='highway' v=' residential ' />
    <tag k='lit ' v='yes' />
    <tag k='maxspeed' v='30' />
    <tag k='name' v='Stiftstraße' />
    <tag k='sidewalk' v='both' />
    <tag k='surface' v='asphalt' />
</way>
```

Das dritte und letzte Grundelement sind die Relationen. Sie werden durch ein XML-Element mit dem Namen *relation* angegeben. In ihrem Aufbau ähneln sie Wegen sehr stark. Die Eigenschaften und weitere Informationen werden in Attributen gespeichert. Im Gegensatz zu Wegen können zu Relationen nicht nur Knoten hinzugefügt werden, sondern auch Wege und andere Relationen. Dazu wird ein XML-Element mit dem Namen *member* und dem Schlüssel *ref* verwendet. [51]

Ein häufiges Einsatzgebiet ist die Kennzeichnung von U-Bahnlinien. Zur Darstellung einer U-Bahnlinie in einer Karte werden die Gleise, Stationen und eventuell weitere Objekte wie zum Beispiel Signale benötigt. Die Gleise können durch einen Weg repräsentiert werden, während sich für die Stationen Gebiete (geschlossene Wege) anbieten. Signale sind Knoten entlang der Strecke, die die Eigenschaft *signal* besitzen. Um diese unterschiedlichen Elemente der U-Bahnlinie zuzuordnen, werden Relationen genutzt. Beispielhaft ist im folgenden Codeblock eine solche Relation dargestellt. Der Übersichtlichkeit halber werden nur die Gleise (Z. 4), eine Station (Z. 3) und ein Signal (Z. 2) gezeigt. [51]

```
<relation id='2872789' visible ='true' changeset='104594697'>
    <member type='node' ref='6055658161' role='stop_entry_only' />
    <member type='way' ref='466451524' role='platform_entry_only' />
    <member type='way' ref='383422087' role='' />
    <tag k='from' v='Elbbrücken' />
    <tag k='name' v='U4: Elbbrücken =&gt; Billstedt' />
    <tag k='network' v='HVV' />
    <tag k='operator' v='Hamburger Hochbahn AG' />
    <tag k='ref' v='U4' />
    <tag k='route' v='subway' />
    <tag k='to' v=' Billstedt ' />
</relation>
```

2.2.2 Qualität der Daten

Für die Navigation des Shared Guide Dog, der sich hauptsächlich auf Bürgersteigen und Fußwegen bewegt, ist die Qualität der Kartendaten von großer Bedeutung. Bereits eine geringe Abweichung kann ausreichen, damit der in Open Street Map eingezeichnete Weg nicht mit dem realen Weg übereinstimmt. Da die Qualität der Open Street Map Kartendaten für unterschiedliche Regionen der Welt sehr unterschiedlich ausfallen kann, soll in diesem Abschnitt ein Überblick über Kartierungsmöglichkeiten als Grundlage der Karte gegeben werden. Eine Bewertung der Datenqualität, Datenaktualität und des Detailgrades im Testgebiet wird in Kapitel 7.2 durchgeführt.

Das Open Street Map Projekt sieht drei unterschiedliche Methoden zum Kartieren vor. Das sind das Aufnehmen von Wegen mit GPS, die Verwendung von Luft- und Satellitenbildern und die Nutzung von alten Karten. Bei Luft- und Satellitenbildern sowie alten Karten ist darauf zu achten, dass kein Copyright die Verwendung einschränkt, wie es beispielsweise bei Luftbildern von Google Maps der Fall ist. [46]

Neue Wege in OSM werden in der Regel mit einem GPS-Empfänger abgelaufen und die Positionen aufgenommen. Dieser Vorgang wird mehrere Male an unterschiedlichen Tagen und zu unterschiedlichen Zeiten wiederholt, um anschließend daraus einen Mittelwert zu bilden. Damit lassen sich die Fehlerquellen im GPS-Signal minimieren und die Genauigkeit der Daten erhöhen. Es bleibt jedoch die Gefahr von systematischen Fehlern. Diese treten etwa in der Nähe von hohen Gebäuden auf. Das GPS-Signal kann von diesen reflektiert werden, was eine Abweichung von mehreren Metern zur Folge haben kann (vgl. Abschnitt 3.2.4).

Die zweite Möglichkeit, eine Gegend für Open Street Map zu überprüfen oder zu kartographieren, sind Luft- und Satellitenbilder. Der Prozess, bei dem mehrere Datenquellen genutzt werden, um eine Karte zu erstellen, wird als Georektifikation (engl. georectification) bezeichnet. Grundsätzlich haben Satelliten- und Luftbilder Unsicherheiten durch die Positionierung und durch die Verzerrung. Die Positionierungsunsicherheit entsteht durch ungenaue Referenzpunkte, an denen das Bild ausgerichtet wird. Eine Verzerrung kann durch einen schrägen Aufnahmewinkel, durch ungenaue topographische Daten, zum Beispiel Erhebungen, und durch Fehler beim Zusammenfügen der Bilder entstehen.

Eine Datenquelle, die in der Regel qualitativ sehr hochwertige Luftbilder und Karten zur Verfügung stellt, sind die Katasterämter. Die Daten sind georeferenziert und entzerrt, können also ohne weitere Bearbeitung genutzt werden. Für viele Regionen in Deutschland stehen die Karten zur Verfügung, teilweise kostenfrei. Jedoch bieten nicht alle Länder der Erde diese Möglichkeit. Neben den Katasterämtern gibt es noch weitere Quellen für Luftbilder. Eine Nutzung ist jedoch schwierig, da oft nicht bekannt ist, ob die Bilder korrekt positioniert und entzerrt sind.

Als letzte Möglichkeit zur Überprüfung der Kartendaten, werden ältere Karten genutzt. Dabei handelt es sich um offizielle Karten, deren Copyright abgelaufen ist und dementsprechend genutzt werden können. Der Nachteil dieser Karten ist jedoch, dass oftmals Straßen oder Gebäude eingezeichnet sind, die es nicht mehr gibt oder die bereits baulich verändert wurden. Eine Nutzung dieser Karten ist selten.

2.3 Pfadplanungsalgorithmen

Das Ziel eines Pfadplanungsalgorithmus ist die Berechnung eines Weges in einer gegebenen Umgebung nach bestimmten Kriterien. Die Kriterien können beispielsweise die Weglänge oder die benötigte Zeit für den Weg sein. Eine einfache Klasse der Pfadplanungsalgorithmen stellen die Forward-Search Algorithmen (dt. Vorwärtssuche) dar. Diese Algorithmen gehen von einem bekannten Startpunkt aus und suchen von diesem einen Weg zum Ziel. Damit die Pfadplanungsalgorithmen einen Pfad berechnen können, werden Informationen über die Erreichbarkeit von Punkten ausgehend von anderen Punkten benötigt. Als Datenstruktur bietet sich ein ungerichteter Graph an. Die Ecken des Graphen repräsentieren Kreuzungen und Punkte auf der Karte, die Kanten verbinden die Ecken und stellen die Wege und Straßen dar. [23, S. 1 ff.]

Die generellen Schritte bei Forward-Search Algorithmen sind in der folgenden Auflistung dargestellt. Q bezeichnet die Liste der zu untersuchenden Knoten. [26, S. 32 f.]

1. Füge den Startknoten zu Q hinzu
2. Nimm den ersten Knoten aus Q
3. Wenn der aktuelle Knoten dem Zielknoten entspricht, beende den Algorithmus
4. Prüfe für alle Nachbarknoten, ob sie bereits durchlaufen wurden
   a) Wenn ja, führe eine zu definierende Aktion aus
   b) sonst füge die Nachbarknoten zu Q hinzu
5. Führe die Schritte 2 bis 4 so lange aus, bis das Ziel gefunden ist oder alle Knoten untersucht wurden

Im Folgenden sollen Dijkstras Algorithmus und der A* Algorithmus als bekannte Vertreter der Forward-Search Algorithmen erläutert werden. Als Abschluss des Kapitels wird ein bidirektionaler Ansatz vorgestellt, der eine Weiterentwicklung der vorherigen Algorithmen darstellt. Die Forward-Search Algorithmen befolgen alle grundsätzlich das vorgestellte Schema, besitzen jedoch unterschiedliche Implementierungen der Liste der zu untersuchenden Knoten und behandeln bereits untersuchte Knoten unterschiedlich. [26, S. 35 ff.]

Einer der bekannten Pfadplanungsalgorithmen ist Dijkstras Algorithmus. Q wird als sortierte Liste implementiert. Als Sortierkriterium werden die Kosten bis zum jeweiligen Knoten verwendet. Dafür werden die Kanten des Graphen mit Kosten versehen, die im einfachsten Fall nur die Weglänge berücksichtigen. Der nächste zu untersuchende Knoten wird anhand der Kosten zu diesem Knoten ausgewählt. Die Gesamtkosten des Plans sind die kumulierten Kosten aller Kanten. Eine Eigenschaft von Dijkstras Algorithmus ist, dass er immer den optimalen Plan bezüglich der Kosten findet. [26, S. 36 f.]

Der A*-Algorithmus ist eine Erweiterung von Dijkstra's Algorithmus mit dem Ziel, die Anzahl der zu prüfenden Knoten zu reduzieren. Dazu wird eine weitere Kostenfunktion eingeführt. Diese

Kostenfunktion wird als Heuristik bezeichnet und schätzt die Kosten vom aktuellen Knoten zum Ziel. Da die realen Kosten vor der vollständigen Exploration nicht bekannt sein werden können die Kosten nur geschätzt und nicht berechnet werden. Die Eigenschaften der Heuristik sind, dass die Kosten zum Ziel möglichst genau an den realen Kosten liegen, ohne diese jemals zu überschätzen. Die Einführung der Heuristik kann als Orientierung für den Algorithmus angesehen werden. Da die Entfernung vom Zielpunkt teurer als die Annäherung ist, wird zuerst in Richtung des Ziels gesucht, bevor weitere Wege evaluiert werden. Kann gewährleistet werden, dass die Heuristik die Kosten nicht überschätzt, ist das Ergebnis des A*-Algorithmus ein optimaler Weg. Eine häufig verwendete Kostenfunktion ist die euklidische Distanz, da der Weg zwischen einem Knoten und dem Zielknoten niemals kleiner sein kann. Die Auswahl des nächsten zu untersuchenden Knotens wird wie bei Dijkstra's Algorithmus anhand der Kosten bestimmt. Die Kosten setzen sich jedoch aus der Heuristik und den Kosten zum aktuellen Knoten zusammen. [26, S. 37 f.]

Die bidirektionale Suche gehört nicht zu den Forward-Search Algorithmen, sondern stellt einen neuen Ansatz dar. Die Idee ist, dass ein Suchalgorithmus nicht nur vom Startknoten, sondern auch vom Zielknoten aus gestartet wird. Das Ziel ist erreicht, wenn sich beide Algorithmen treffen. Als Suchalgorithmus können zum Beispiel Dijkstra oder A* eingesetzt werden, die eine optimale Lösung erzeugen. Die Verwendung der bidirektionalen Suche kann die Anzahl der untersuchten Knoten in vielen Fällen deutlich reduzieren. Ein Beispiel ist ein Graph mit vielen Sackgassen. Wird nur von einer Seite aus gesucht, besteht eine hohe Wahrscheinlichkeit, dass zuerst eine benachbarte Sackgasse exploriert wird, bevor der Weg zum Ziel gefunden wird. Durch die Suche von beiden Seiten wird dieses Risiko reduziert. [26, S. 41]

3 Technische Grundlagen der Lokalisation

Dieses Kapitel gibt einen kurzen Überblick über verschiedene Möglichkeiten der Lokalisation. Dazu werden zunächst die technischen Grundlagen erläutert und anschließend häufig verwendete Sensoren vorgestellt.

3.1 Lokalisation in bekannten und unbekannten Karten

Die Lokalisation eines Roboters lässt sich in zwei große Teilbereiche einteilen. Die erste Gruppe fasst die Lokalisation eines Roboters auf einer unbekannten Karte zusammen, die zweite Gruppe beinhaltet Methoden zur Lokalisierung eines Roboters auf bereits bekannten Karten, auch als a-priori Karten bezeichnet.

Die Lokalisation in unbekannten Karten hat autonome Roboter lange Zeit vor eine große Herausforderung gestellt. Es ist nicht ausreichend, Hindernisse zu erkennen und zu umfahren, gleichzeitig müssen eben diese Hindernisse in der Karte verzeichnet werden. Dieser Prozess des gleichzeitigen Lokalisierens und Kartieren wird als Simultaneous Localization and Mapping (SLAM) bezeichnet. Die Schwierigkeit ist, dass die Position des Roboters für eine präzise Kartierung exakt bekannt sein müsste, für die exakte Lokalisation ist hingegen eine präzise Karte erforderlich. Heute sind verschiedene Methoden verfügbar, die es einem Roboter in einer strukturierten und statischen Umgebung ermöglichen, sich zu lokalisieren und gleichzeitig die Kartierung durchzuführen. Unstrukturierte, dynamische oder sehr große Regionen können jedoch nicht robust mit diesen Methoden abgebildet werden. [61, S. 871 ff.]

Alle SLAM-Algorithmen müssen mit zwei großen Herausforderungen umgehen. Zum einen sind das fehlerbehaftete Sensordaten, zum anderen wiederkehrende Strukturen. Fehlerbehaftete Sensordaten lassen sich nicht vermeiden, da es keinen perfekten Sensor gibt. Als Beispiel wird die Lokalisierung mittels Odometrie und die Kartierung mit einem Lidar-Sensor betrachtet. Mit der Odometrie lässt sich eine Position relativ zur Startposition bestimmen. Diese Positionsbestimmung wird als Koppelnavigation (engl. dead reckoning) bezeichnet und hat die Eigenschaft, dass der Fehler in der Positionsbestimmung bei längeren Strecken immer größer wird. Dieser Fehler wird als *Drift* bezeichnet und muss durch den SLAM-Algorithmus kompensiert werden. [61, S. 872 ff.]

Wiederkehrende Strukturen als zweite Herausforderung sind ein Folgeproblem der fehlerbehafteten Positionsbestimmung. Viele SLAM-Algorithmen versuchen Strukturen, die bereits kartiert sind, in neuen Sensordaten wiederzuerkennen, um eine zusammenhängende Karte zu erstellen. Dies ist jedoch in strukturierten Umgebungen schwierig umzusetzen. Ein Beispiel ist ein langer Gang, von dem viele gleichartige Türen abgehen, wie es etwa bei einem Krankenhausflur der Fall ist. Solch ein Gang

besitzt kaum Strukturen, die ein SLAM-Algorithmus nutzen kann, um bereits besuchte Regionen wieder zu erkennen. Ein Beispiel für die aufgenommenen Daten und die mit einem SLAM-Algorithmus erzeugte Karte ist in Abbildung 3.1 gezeigt. Auf der linken Seite ist gut zu erkennen, dass durch den fortschreitenden Fehler der Odometriedaten, die aufgenommene Karte verzerrt wird. Die Karte auf der rechten Seite wurde anhand der aufgenommenen Strukturen korrigiert und zeigt eine Karte mit geringen Abweichungen. [61, S. 883 f.]

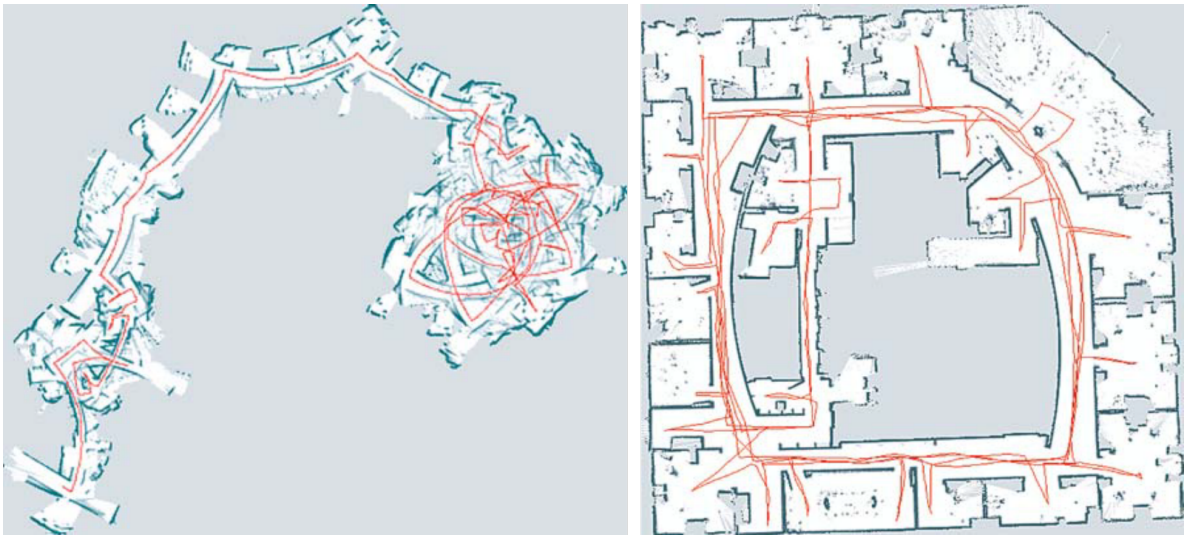

Abbildung 3.1: Links: Karte erstellt mit Laserscanner und Odometrie, rechts: Karte nach der Fehlerbereinigung (Quelle: [61, S. 884])

Die Lokalisation in bekannten Karten wird in vielen Fällen durch Triangulation durchgeführt. Für die Triangulation werden mehrere Referenzpunkte benötigt, deren Position bekannt sein muss. Durch die Messung der Entfernung des Roboters zu den Referenzpunkten kann die Position bestimmt werden. Für eine eindeutige Positionsbestimmung im zweidimensionalen Raum müssen mindestens drei Referenzpunkte sichtbar sein, die Positionsbestimmung im dreidimensionalen Raum benötigt vier Referenzpunkte. Zudem muss die Position der Hindernisse in einem Referenzsystem bekannt sein. Die Referenzpunkte lassen sich in aktive und passive Referenzpunkte einteilen. Passive Referenzpunkte senden keine Signale zur eigenen Position, sondern müssen von Systemen auf dem Roboter erfasst werden. Häufig werden Hindernisse genutzt, die in der Umgebung erkannt werden und auf der Karte eingezeichnet sind. Aktive Referenzpunkte senden hingegen eigene Signale, die vom Roboter empfangen werden können. Aus der Laufzeit des Signals kann die Position bestimmt werden. Die bekanntesten Vertreter der aktiven Systeme sind globale Satellitennavigationssysteme (GNSS).

3.2 Systeme zur Lokalisation

Der folgende Abschnitt beschäftigt sich mit verschiedenen Systemen, die eine Lokalisation eines mobilen Roboters ermöglichen. Dabei wird zwischen der globalen und der lokalen Lokalisation unterschieden. In der Literatur wird die globale Lokalisation auch als absolute Lokalisation und die

lokale Lokalisation als relative Lokalisation bezeichnet (vgl. [18, S. 110 f.]). Das Ergebnis einer globalen Lokalisation ist die Orientierung und Position, bezeichnet als *Pose*, in einem globalen Koordinatensystem. Bei der lokalen Lokalisation wird hingegen die Pose in einem lokalen Koordinatensystem bestimmt. [18, S. 110 f.]

3.2.1 Koordinatensysteme

Bevor eine Lokalisierung durchgeführt werden kann, müssen Koordinatensysteme definiert werden, in denen die Bewegungen des Roboters abgebildet werden können. Jeder mobile Roboter besitzt ein Koordinatensystem, das fest mit der Umgebung und der Karte verbunden ist. Dieses Koordinatensystem wird als globales Koordinatensystem bezeichnet. Zusätzlich wird ein lokales Koordinatensystem definiert, in dem die lokale Lokalisation abgebildet wird. Das globale und lokale Koordinatensystem wird meist als kartesisches Koordinatensystem definiert. Ein Roboter mit Aktuatoren besitzt üblicherweise für jeden Aktuator ein eigenes Koordinatensystem, in dem die Bewegungen abgebildet werden. Neben dem globalen, lokalen und roboterbezogenen Koordinatensystem kann es weitere Koordinatensysteme geben. Ein besonders bei Outdoor-Robotern genutztes Koordinatensystem ist ein weiteres globales Koordinatensystem, in dem Positionen im erdnahen Umfeld abgebildet werden können. Eine Übersicht über das globale und lokale Koordinatensystem und die Definition ist in Abbildung 3.2 gegeben. Die globale Lokalisation referenziert die Pose des Roboters im globalen Koordinatensystem (x, y). Die lokale Lokalisation gibt hingegen nur die Pose in einem lokalen Koordinatensystem (x_L, y_L) an. Die Position des lokalen Koordinatensystems im globalen Koordinatensystem ist nicht bekannt. [61, S. 23]

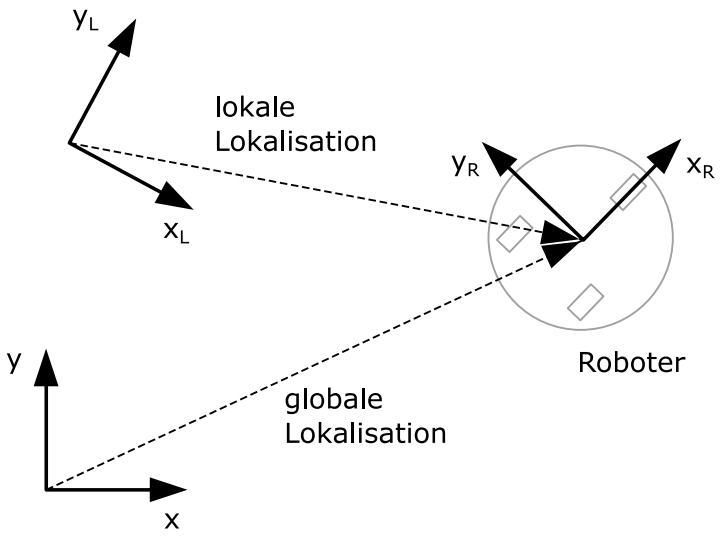

Abbildung 3.2: Definition verschiedener Koordinatensysteme (Quelle: eigene Darstellung)

3.2.2 World Geodetic System 1984

Der meistgenutzte Vertreter der globalen Koordinatensysteme ist das WGS 84 (World Geodetic System 1984). Dieses Koordinatensystem wird als Standard-Referenzsystem zur Positionsbestimmung durch Globale Satellitennavigationssysteme und für die Navigation verwendet. Es ist eng mit weiteren Erdmodellen, beispielsweise dem World Magnetic Model (WMM) und dem Earth Gravitational Model (EGM), verknüpft. Dies hat den Vorteil, dass beispielsweise das Erdmagnetfeld an jeder Stelle direkt mit den Koordinaten berechnet und zum Beispiel für ein Magnetometer verwendet werden kann. Für die Definition des WGS 84 werden zunächst ein erdgebundenes, kartesisches Koordinatensystem und anschließend in diesem ein Ellipsoid, das die Erdoberfläche beschreibt, festgelegt. Der Koordinatenursprung liegt im Massenschwerpunkt der Erde. Die Richtung der z-Achse entspricht der Richtung des IERS Reference Pole (IRP, Nordpol) und die x-Achse dem IERS Reference Meridian (IRM) (vgl. [19] für eine Definition des IRP und IRM). Der IRM wird auch als Nullmeridian bezeichnet und verläuft durch Greenwich, einen Stadtteil von London. Die y-Achse steht senkrecht auf der von x- und z-Achse aufgespannten Ebene, sodass sich ein rechtshändiges Koordinatensystem ergibt. Die Definition des WGS 84 Koordinatensystems ist in das Referenzellipsoid in Abbildung 3.3 eingezeichnet. [36, S. 2–1 f.]

Um die irreguläre Oberfläche der Erde mathematisch beschreiben zu können, wird vom WGS 84 ein Referenzellipsoid definiert. Dieses Ellipsoid ist so gelegen, dass es die Irregularitäten der Erdoberfläche mittelt und die Oberfläche eine Äquipotentialfläche zur Anziehungskraft der Erde bildet. Es ist rotationssymmetrisch um die Rotationsachse der Erde. Für die mathematisch vollständige Beschreibung dieses Ellipsoids sind vier Parameter nötig. Definiert sind die große Halbachse a, der Abplattungsfaktor der Erde $1/f$, die geozentrische Gravitationskonstante GM und die nominale mittlere Rotationsgeschwindigkeit der Erde ω. In Tabelle 3.1 sind alle Parameter mit ihren Werten aufgelistet. [36, S. 3–1 ff.]

Tabelle 3.1: Parameter zur Beschreibung des WGS 84 Referenzellipsoids

Parameter	Bezeichnung	Wert	Einheit
Große Halbachse	a	$6.378.137,0$	m
Abplattung	$1/f$	$298,257223563$	-
Gravitationskonstante	GM	$3,986004418 \cdot 10^{14}$	m^3/s^2
Rotationsgeschwindigkeit	ω	$7,292115 \cdot 10^{-5}$	rad/s

Für die eindeutige Identifizierung eines Punktes auf dem Referenzellipsoid wird dieser in Längen- und Breitengrade eingeteilt. Die Längengrade (engl. longitude) verlaufen in Nord-Süd-Richtung, die Breitengrade (engl. latitude) verlaufen in Ost-West-Richtung. In Abbildung 3.3 ist ein Ellipsoid mit Längen- und Breitengraden, sowie den beiden Halbachsen a und b gezeigt. Die kleine Halbachse b lässt sich mit dem Abplattungsfaktor $1/f$ und der großen Halbachse a nach Gleichung 3.1 berechnen. [36, B-3]

$$b = a \cdot (1 - f) = 6.356.752,3142\,m \tag{3.1}$$

Die Bezeichnung der Längen- und Breitengrade erfolgt anhand des bereits beschriebenen Koordinatensystems. Demzufolge ist der Äquator der 0. Breitengrad, da er in der x-y-Ebene liegt und der IERS Reference Meridian der 0. Längengrad. Ausgehend vom 0. Breitengrad sind alle Breitengrade nördlich positiv bezeichnet, in Richtung Süden negativ. Alternativ werden die Breitengrade mit den Buchstaben *N* in nördlicher Richtung und *S* in südlicher Richtung versehen. Die Breitengrade können Werte zwischen -90 ° und +90 ° annehmen. Die Längengrade werden anhand der positiven Drehrichtung um die z-Achse des Koordinatensystems, also in Richtung Osten, positiv bezeichnet, in die andere Richtung entsprechend negativ. Da die Längengrade nur von Pol zu Pol verlaufen, können sie Werte von -180 ° bis +180 ° annehmen. Auch hier ist ein alternative Bezeichnung mit *E* in Richtung Osten und *W* in Richtung Westen möglich.

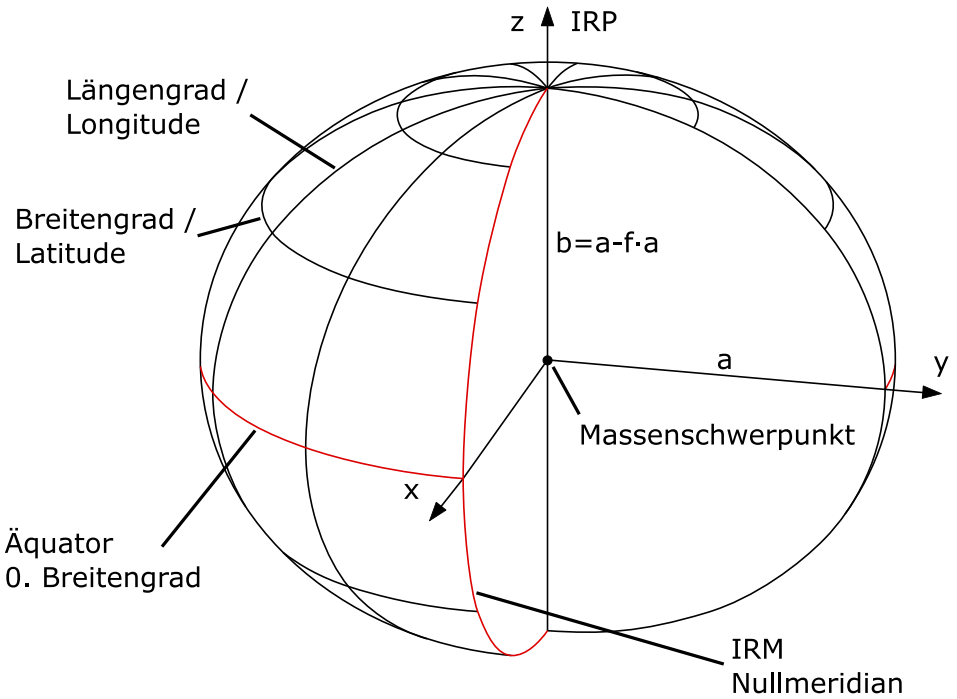

Abbildung 3.3: Bezeichnungen am WGS 84 Referenzellipsoid (Quelle: [36, S. 2–1], eigene Darstellung)

3.2.3 Entfernungsberechnung

Die Entfernungsberechnung zwischen zwei Punkten auf dem WGS 84 Referenzellipsoid ist nicht eindeutig. Es gibt drei verschiedene Verbindungen, die zwischen zwei Punkten berechnet werden können. Als erstes eine Verbindung, bei der eine Himmelsrichtung exakt eingehalten wird. Diese Verbindung wird als Loxodrome bezeichnet. Die zweite Verbindung verläuft entlang der kürzesten Verbindung auf der Oberfläche des Ellipsoid, die Orthodrome. Die Berechnung dieser beiden Verbindungen ist jedoch aufwendig. Für kleine Distanzen kann sich der große Erdumfang zu Nutze gemacht werden und die Strecke durch die dritte Verbindung, die direkte Verbindung, angenähert

werden. Bei der Berechnung der direkten Verbindung ist zu beachten, dass der Abstand der Längengrade d_L zueinander zu den Polen immer weiter abnimmt. Eine Berücksichtigung dieser Tatsache ist mit dem Kosinus möglich, damit ergibt sich Gleichung 3.2 zur Berechnung des Abstands der Längengrade.

$$d_L(\psi) = \frac{2\pi \cdot a}{360} \cdot \cos\psi \tag{3.2}$$

Unter Berücksichtigung von Gleichung 3.2 ergibt sich Gleichung 3.3 zur Berechnung der euklidischen Distanz zwischen zwei Punkten auf der Erdoberfläche in Metern. Die Längengrade werden mit l und die Breitengrade mit b bezeichnet. Der Faktor 111.319 wird zur Umrechnung von Grad in Meter verwendet und ergibt sich aus dem Erdumfang am Äquator dividiert durch 360 °.

$$d = 111.319 \cdot \sqrt{(b_1 - b_2)^2 + \left((l_1 - l_2) \cdot \cos\frac{(b_1 + b_2) \cdot \pi}{2 \cdot 180}\right)^2} \tag{3.3}$$

3.2.4 Globale Navigationssatellitensysteme

Häufig wird zur globalen Lokalisation ein Global Navigation Satellite System (dt. Globales Navigationssatellitensystem) (GNSS) verwendet. Mit einem solchen System ist es theoretisch möglich, eine absolute Position bis auf wenige Millimeter genau anzugeben. Dazu muss jedoch ein hoher technischer Aufwand getrieben werden, sodass sich diese Werte in der Praxis nicht erreichen lassen. Das am weitesten verbreitete GNSS ist NAVSTAR-GPS, das von den U.S.A. betrieben wird. Daneben gibt es noch weiter Länder, die eigene satellitengestützte Navigationssysteme betreiben. Die Europäische Union betreibt GALILEO, Russland GLONASS und China BEIDOU. Weitere Systeme, die nur kleine Bereiche der Erde abdecken oder für spezielle Zwecke konzipiert wurden, sollen hier nicht weiter betrachtet werden.

Für eine Positionsbestimmung werden Bodenstationen, Satelliten im Erdorbit und ein GNSS-fähiger Empfänger benötigt. Die Bodenstationen beobachten und überwachen die Satelliten im Erdorbit und können bei Bedarf Kurs- und Zeitkorrekturdaten senden. Die Satellitenkonstellation besteht aus mehreren Satelliten, die die Erde umkreisen. Sie senden in regelmäßigen Abständen ihre Position und die aktuelle Uhrzeit. Der Empfänger kann die Signale der Satelliten empfangen. Aus dem Laufzeitunterschied der einzelnen Signale ist es möglich eine genaue Position zu bestimmen. Wird davon ausgegangen, dass sich das Signal der Satelliten gleichmäßig ausbreitet, spannt jeder Satellit eine Kugeloberfläche auf, deren Radius durch die Laufzeit des Signals bestimmt werden kann. Sind nur zwei Satelliten verfügbar, schneiden sich die beiden Kugeln in einem Kreis. Durch den dritten Satelliten lassen sich die möglichen Positionen auf zwei Punkte verringern. Der aufgrund der Höhe unwahrscheinlichere Punkt wird vom Empfänger verworfen. In der Praxis wird jedoch noch ein vierter Satellit benötigt, da die Satellitenuhren zu große Abweichungen untereinander haben, als dass die Laufzeitunterschiede ohne externe Zeitsynchronisation bestimmt werden könnten. Aus den vier vorhandenen Signalen können die Position mit den drei Unbekannten Längengrad, Breitengrad und Höhe sowie die Zeit berechnet werden. [70]

Das älteste der GNSS ist NAVSTAR-GPS, als GPS bekannt, das seit 1973 betrieben wird. Im Kalten Krieg wuchs der Bedarf der US Air Force nach einem System, mit dem präzise Positionen mit einer globalen Abdeckung bestimmt werden können. Aus diesem Bedürfnis entstand NAVSTAR-GPS. Auch wenn es bereits seit 1973 in Benutzung ist, wurde es erst im Jahr 1993 mit 24 Satelliten im Orbit als global einsatzbereit erklärt. Für die nichtmilitärische Nutzung wurden Fehler in die Zeit- und Positionssignale eingestreut, die sogenannte Selective Availability (dt. Selektive Verfügbarkeit) (SA), damit nur das US-Militär Zugang zu den genauesten Positionsdaten hat. Seit dem Jahr 2000 gibt es diese Einschränkung nicht mehr. [70] [11]

Bei den Bodenstationen wird zwischen Beobachtungsstationen (engl. Monitor Stations), Bodenantennen (engl. Ground Antennas) und der Hauptsteuerungsstation (engl. Master Control Station) unterschieden. Die Beobachtungsstationen überwachen die Satelliten und sammeln dabei Daten, die sie an die Master Control Station übermitteln. Aus diesen Daten berechnet die Master Control Station die genauen Positionen der Satelliten und kann bei Bedarf Steuerungskommandos generieren, damit eine optimale Verteilung der Satelliten im Erdorbit gewährleistet ist. Die Bodenantennen senden diese Korrekturdaten anschließend an die Satelliten. [35]

Für NAVSTAR-GPS waren ursprünglich 24 Satelliten vorgesehen. Sie sind in sechs gleichmäßig verteilten Erdorbits angeordnet und umfliegen die Erde zweimal am Tag. Die Satelliten bewegen sich auf einer mittleren Höhe von 20.200 km. Heute werden 32 Satelliten betrieben, um bei Ausfällen oder Wartungsarbeiten die Verfügbarkeit sicherzustellen. [70]

Als Fehlerquellen im GPS-Signal werden in diesem Abschnitt die sogenannten User Equivalent Range Errors (UERE) vorgestellt. Neben den UERE haben Rundungsfehler einen kleinen Einfluss auf die Positionsbestimmung, der hier jedoch nicht weiter betrachtet werden soll. Als Maß für die Abweichung einer GPS-Position wird anschließend die Dilution of Precision (DOP) behandelt. Als UERE werden sechs Fehlerquellen bezeichnet, die auf dem Weg des Signals vom Satelliten bis zum Empfänger auftreten. Das sind in absteigender Relevanz für den Praxiseinsatz sortiert:

- Ablenkung durch Ionosphäre (upper atmosphere (ionosphere))
- Mehrwegefehler (multipath)
- Satellitenumlaufbahn (satellite orbit)
- Untere Atmosphäre (lower atmosphere)
- Ungenauigkeit der Empfängeruhr (receiver clock)
- Ungenauigkeit der Satellitenuhr (satellite clock)

Die Ungenauigkeit der Satellitenuhren spielen in der Praxis kaum eine Rolle, da sie durchgehend ausgeglichen werden. Die Satelliten empfangen Korrekturdaten von den Kontrollstationen am Boden und passen dementsprechend die interne Zeit an. Die Uhrzeit des Empfängers wird durch den vierten Satelliten berechnet.

Damit bleiben vier praxisrelevante Fehlerquellen übrig. Den größten Fehler verursacht die Ablenkung und Verzögerung des Signals in der Atmosphäre. Die Ionosphäre hat dabei einen deutlich größeren Einfluss als die Troposphäre und Stratosphäre. Die Dichte der Erdatmosphäre variiert je nach Tages- und Jahreszeit und in Abhängigkeit von der Sonneneinstrahlung. Je nachdem wie hoch der Satellit aus Sicht des Empfängers über dem Horizont steht, ist der Weg länger oder kürzer. Ein langer Weg bedeutet, dass eine größere Distanz in der Erdatmosphäre zurückgelegt wird und somit auch eine größere Ablenkung und Verzögerung stattfindet als bei einem Signal, das von einem Satelliten direkt über dem Empfänger ausgesendet wird. Diese Fehlerquelle kann durch Korrekturdaten zu ungefähr 3/4 korrigiert werden, es bleibt jedoch ein Restfehler.

Eine Fehlerquelle, die besonders in dicht bebauten Städten, aber auch in Wäldern auftreten kann, ist die Reflektion des Signals von Gebäuden und anderen festen Gegenständen. Der Empfänger erhält ein Signal, das eine größere Strecke zurückgelegt hat und berechnet daraufhin eine fehlerhafte Position. Dieser Fehler wird als Mehrwegefehler (engl. multipath error) bezeichnet. Diese Fehlerquelle kann je nach Ort der Positionsbestimmung einen größeren Einfluss als die Ablenkung durch die Ionosphäre haben und lässt sich nicht korrigieren.

Eine weitere Fehlerquelle ist die Positionsgenauigkeit der Satelliten auf der Umlaufbahn. Die Satelliten folgen der Umlaufbahn nie exakt, sondern werden durch Gravitationskräfte der Erde, des Mondes und der Sonne beeinflusst. Zu einem bestimmten Teil können diese Einflüsse bei der Positionsbestimmung berücksichtigt werden, jedoch bleibt ein Restfehler, der die Positionsbestimmung des Empfängers beeinflusst. [6]

Die DOP ist ein Maß, für die Verteilung der Satelliten über den Himmel und wird in verschiedene Werte unterschieden. Je größer der Wert wird, desto schlechter ist die Positionsbestimmung, ein kleiner Wert wird bei kleinen Fehlern erreicht. Damit der GPS-Empfänger eine Position bestimmen kann, werden vier Satelliten benötigt. Als Beispiel wird eine gleichmäßige Verteilung der Satelliten über den Himmel und eine Verteilung, bei der alle Satelliten aus Sicht des Empfängers dicht beieinanderstehen, betrachtet. Im ersten Fall wäre die Horizontal Dilution of Precision (HDOP), also die DOP bezogen auf die horizontale Position, niedrig, da durch die Fehler nur ein kleines Gebiet aufgespannt wird. Eine Konstellation, bei der die Satellitensignale aus einer ähnlichen Richtung kommen, wird ein größeres Gebiet aufgespannt und damit ist die HDOP größer. Zur Berechnung der DOP werden die Varianzen der Position in Ost-West-Richtung σ_E, Nord-Süd-Richtung σ_N und Höhenrichtung σ_U verwendet. Es ergeben sich die Gleichungen 3.4 bis 3.6 mit der Varianz σ durch die UERE. [25, S. 56] [71, Lesson 3]

$$PDOP = \frac{\sqrt{\sigma_E^2 + \sigma_N^2 + \sigma_U^2}}{\sigma} \tag{3.4}$$

$$HDOP = \frac{\sqrt{\sigma_E^2 + \sigma_N^2}}{\sigma} \tag{3.5}$$

$$VDOP = \frac{\sqrt{\sigma_U^2}}{\sigma} \tag{3.6}$$

Unter dem Begriff Ergänzungssysteme (engl. augmentation system) werden alle Systeme zusammengefasst, die das GPS-Signal in irgendeiner Weise verbessern. Der häufigste Einsatzzweck ist die Verbesserung der Positionierung und der Verfügbarkeit des GPS-Signals. Der am weitesten verbreitete Ansatz sind differentielle Verfahren, bei denen eine georeferenzierte Bodenstation die GPS-Signale empfängt, diese Position mit der realen Position abgleicht und daraus Korrektursignale berechnet. Diese Verfahren werden unter dem Begriff Differential GPS (DGPS) zusammengefasst. Die Gruppe der DGPS-Systeme lässt sich in Wide Area DGPS und Local Area DGPS aufteilen. Das Wide Area DGPS zählt zu den Satellite Based Augmentation System (dt. Satellitenbasiertes Ergänzungssystem) (SBAS). Dabei werden die Korrektursignale über separate Satelliten zur Verfügung gestellt. Beim Local Area DGPS werden die Signale direkt von den Bodenstationen gesendet und haben demnach eine kleinere Reichweite. In Deutschland ist Local Area DGPS über den staatlichen Dienst Satellite Positioning System (SAPOS) verfügbar. [35] [10]

3.2.5 LiDAR

Der Begriff Light Detection And Ranging (LiDAR) ist angelehnt an den Begriff RADAR (Radio Detection And Ranging), das bereits seit langer Zeit eingesetzt wird. Mit einem LiDAR-Sensor lässt sich die Entfernung zu einem Objekt berührungslos messen. Grundsätzlich ist ein LiDAR-Sensor aus einer Sende-, einer Empfangseinheit, der Ansteuerelektronik und einer Signalauswertung aufgebaut. In Abbildung 3.4 wird ein typischer Aufbau eines LiDAR-Sensors gezeigt. Die Ansteuerelektronik sendet ein Signal an die Sendeeinheit, die daraufhin einen Laserstrahl emittiert. Üblicherweise wird für das menschliche Auge unsichtbares Infrarotlicht in einem Wellenlängenbereich von 780 nm bis 950 nm verwendet. Der ausgesendete Laserstrahl trifft auf ein Objekt, wird durch dieses reflektiert und von der Empfängereinheit detektiert. Mit der Signalauswertung kann aus dem empfangenen Signal die Entfernung zum Objekt bestimmt werden. Voraussetzung für die genaue und erfolgreiche Messung eines Objekts ist eine direkte Sichtverbindung zum Objekt und die Fähigkeit des Objekts, den Laserstrahl zu reflektieren. Durch das aktive Aussenden eines Laserstrahls, ist eine Messung unabhängig der Beleuchtung möglich. [73] [74, S. 175 f.]

Für die Distanzmessung kommen zwei verschiedene Messverfahren zum Einsatz. Das am häufigsten verwendete Verfahren ist die Time-of-Flight (TOF) Messung. Ein weiteres Verfahren, das insbesondere bei eindimensionalen Sensoren zum Einsatz kommt, ist die Phasenkorrelation. [73, S. 5 f.]

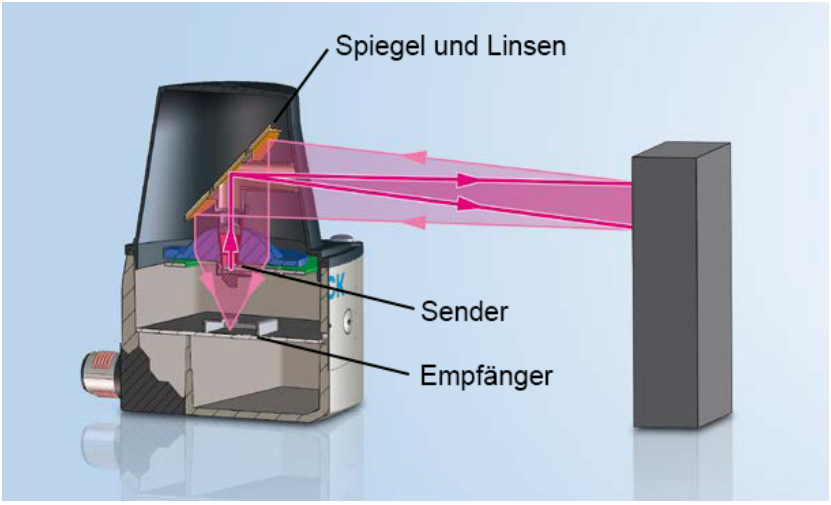

Abbildung 3.4: Beispielhafter Aufbau eines LiDAR-Sensors (Quelle: [73, S. 9], eigene Beschriftung)

Die TOF-Messung basiert auf der Messung der Zeit, die das ausgesendete Licht braucht, bis es wieder beim Empfänger ankommt. Da die Lichtgeschwindigkeit in Luft bekannt ist, kann aus der gemessenen Zeit die Entfernung zum Objekt nach Gleichung 3.7 berechnet werden [74, S. 172]. Es besteht also ein direkter Zusammenhang zwischen der gemessenen Zeit und der Entfernung.

$$r = \frac{c_0 \cdot t_{of}}{2} \tag{3.7}$$

mit

r	Abstand in m
c_0	Lichtgeschwindigkeit in Luft in m/s
t_{of}	Laufzeit des Lichts in s

Aufgrund der hohen Geschwindigkeit des Lichts werden hohe Anforderungen an den Sensor gestellt. Mit der für Luft spezifischen Lichtgeschwindigkeit von 299.705.518 m/s benötigt das Licht bis zu einem Objekt in 1 m Entfernung und zurück zum Sensor nur ungefähr 66,7 ns. Trotz dieser kurzen Zeit erreichen die Sensoren einen systematischen Fehler von $\pm$ 1 mm (vgl. Sick LMS4000).

Beim Messverfahren der Phasenkorrelation wird ein kontinuierlicher Laserstrahl ausgesendet, auf den ein Signal mit einer bestimmten Frequenz aufgeprägt ist. Das empfangene Signal wird mit dem ausgesendeten Signal verglichen und die Phasenverschiebung berechnet. Aus der Frequenz und der Phasenverschiebung lässt sich die Entfernung zum Objekt bestimmen. Dieses Verfahren hat den Nachteil, dass nur bis zu einer Phasenverschiebung von 360° eine eindeutige Lösung vorliegt. Darüber hinaus kann keine Distanz berechnet werden. In der Praxis wird dieser Nachteil durch das Aufprägen mehrerer unterschiedlicher Frequenzen umgangen. Im Vergleich zum Pulslaufzeitverfahren liegt die Intensität des ausgesendeten Laserstrahls um mehrere Größenordnungen niedriger. Das Verfahren

ist somit deutlich anfälliger gegenüber Störeinflüssen, zum Beispiel durch Sonneneinstrahlung. [73, S. 5] [66]

LiDAR Sensoren unterliegen zwei zu beachtenden Störeinflüssen. Der erste ist eine starke Umgebungsbeleuchtung, beispielsweise starker Sonnenschein. Da das Sonnenlicht im kontinuierlichen Spektrum Wellenlängen zwischen 140 nm und 10 cm aufweist, werden auch Wellenlängen, wie sie vom LiDAR verwendet werden, emittiert. Treffen diese Lichtstrahlen auf die Empfangseinheit des Sensors, kann es zu fehlerhaften Messungen kommen. Durch Filter und Linsen, kann dieser sogenannte Gleichlichtanteil wirkungsvoll herausgefiltert werden. Der zweite Störeinfluss tritt auf dem Weg des Lichts vom Sender zum Objekt und zurück auf. Durch Regen, Nebel, Rauch oder andere Partikel in der Luft wird das Signal bereits auf dem Weg reflektiert und von der Empfangseinheit eine zu geringe Entfernung detektiert. Bei mehreren Messungen kann dieser Effekt jedoch herausgefiltert werden. Aufgrund der teildurchlässigen Struktur der Hindernisse wird bei jeder Messung eine andere Distanz gemessen. Werden die Distanzen mehrerer Messungen miteinander kombiniert, ergibt sich ein sehr breites Echo mit geringer Intensität, das herausgefiltert werden kann. [74, S. 175] [73, S. 7 f.]

Lidar-Sensoren sind als 1D-, 2D- und 3D-Sensoren verfügbar. In der Navigation und Lokalisation von mobilen Robotern sind heutzutage 2D- und 3D-Sensoren üblich. EIn 2D-LiDAR Sensor besteht aus einer Sende- und Empfangseinheit und einem drehbar gelagerten Spiegel. Der Sender sendet einen Lichtimpuls aus, der von dem Spiegel abgelenkt wird. Anschließend trifft er auf ein Objekt, wird reflektiert und trifft wieder auf den Spiegel. Dieser lenkt das empfangene Licht in Richtung des Empfängers. Der Vorteil dieser Konstruktion ist, dass für eine zweidimensionale Abtastung der Umgebung nur der Spiegel bewegt werden muss und nicht die gesamt Sender- und Empfängereinheit. Übliche Öffnungswinkel für 2D-LiDAR Sensoren sind zwischen 180° und 270°. Es gibt jedoch auch Sensoren mit noch kleineren Öffnungswinkeln und auch welche, die volle 360° abtasten können. [18, S. 42 ff.]

3D-LiDAR Sensoren sind eine Weiterentwicklung der 2D-Sensoren. Sie können neben der horizontalen Ebene auch die vertikale Ebene erfassen. Dabei gibt es zwei unterschiedliche Bauformen. Zum einen kann ein 2D-Scanner um eine Achse geschwenkt werden, um die dritte Dimension zu erfassen. Zum anderen wird ein zweiter Spiegel im Sensor verbaut, der das Licht in einer weiteren Achse ablenken kann und so die dritte Dimension abtastet. Der Vorteil beim Schwenken eines 2D-Scanners ist die günstigere Hardware, da nur ein 2D-Scanner und ein Motor zum Schwenken benötigt wird. Demgegenüber steht jedoch ein hoher Aufwand bei der Synchronisierung der Rotation und den Scans. Ein 3D-Scanner hat diesen Nachteil nicht, dafür sind die Kosten höher. [18, S. 45 ff.]

3.2.6 Inertiale Messeinheit

Eine Inertial Measurement Unit (dt. Inertiale Messeinheit) (IMU) wird genutzt, um die Orientierung eines Objektes im Raum zu bestimmen und zu verfolgen. Eine IMU ist aus zwei oder drei unterschiedlichen Sensoren aufgebaut. Bei einem Aufbau aus zwei Sensoren werden ein Accelerometer, das die Beschleunigung in drei Achsen, und ein Gyroskop, das die Rotationsgeschwindigkeit um drei Achsen

aufnehmen kann, verwendet. Durch die Integration der Beschleunigung kann die Momentangeschwindigkeit berechnet werden und durch einen weiteren Integrationsschritt die Position. Die Integration der Rotationsgeschwindigkeit liefert die Richtung. Heutzutage wird oftmals ein Magnetometer als dritter Sensor verbaut. Dieser misst die magnetische Feldstärke der Erde. Daraus kann eine Orientierung relativ zum Erdmagnetfeld berechnet werden. Sensoren, die ein Magnetometer, ein Accelerometer und ein Gyroskop mit jeweils drei Achsen besitzen, werden als 9-DOF IMU (Degrees of Freedom (DOF)) bezeichnet. Sind nur ein Accelerometer und Gyroskop integriert, handelt es sich um eine 6-DOF IMU. [75]

Die näherungsweise Bestimmung der Position über die Richtung und Geschwindigkeit wird als Koppelnavigation (engl. dead reckoning) bezeichnet. Um die Position in einem Referenzsystem zu berechnen, ist es daher von entscheidender Bedeutung, dass die Startposition, -ausrichtung und -geschwindigkeit in diesem System bekannt sind. [67]

Eine IMU wird in die zwei Bauformen *Stable Platform System* und *Strapdown System* unterteilt. Ein Stable Platform System bezeichnet einen Aufbau, bei dem eine Plattform auf einer festen Position in einem Referenzsystem gehalten wird. Ein einfaches Beispiel ist ein Kreiselinstrument, bei dem ein sich drehender Kreisel in allen Ebenen beweglich gelagert wird. Durch die Drehimpulserhaltung ändert sich die Lage im Raum nicht und die Position kann relativ zu einem Objekt gemessen werden (vgl. mechanisches Gyroskop). Eine andere Möglichkeit ist die Stabilisierung durch drei Motoren, die die Plattform in ihrer Position halten. Üblicherweise stimmt diese Lage mit der horizontalen Ebene eines ortsfesten Koordinatensystems überein. Bei der Stabilisierung mit Motoren werden die Beschleunigungs- und Drehratenwerte der IMU ausgelesen und die Plattform stabilisiert. Die Lage des Fahrzeugs kann dann mit Winkelaufnehmern an den Achsen abgelesen werden. Bei einem Strapdown System ist die IMU direkt am Fahrzeug montiert. Die IMU nimmt dann alle Bewegungen des Fahrzeugs direkt auf. Dieses Verfahren benötigt jedoch deutlich mehr Rechenleistung, da die Bewegungen des Fahrzeugs nicht direkt in einem ortsfesten Referenzsystem ermittelt werden, sondern erst über die Lage der IMU berechnet werden müssen. Heutzutage stellt das jedoch kaum ein Problem dar, da jeder handelsübliche Mikrocontroller in der Lage ist, diese Berechnungen in Echtzeit durchzuführen. Zudem ist der Aufbau verglichen mit dem Stable Platform System deutlich einfacher und kostengünstiger. [67]

Für die Orientierungs- und Drehratenmessung und die Beschleunigungsmessung können unterschiedliche Verfahren genutzt werden. Eines der ältesten Verfahren die Orientierung aufzunehmen ist, einen sich drehenden Kreisel in allen drei Achsen beweglich zu lagern und die Bewegung relativ zum Körper zu messen. Durch den Drehimpuls besitzt der Kreisel eine hohe Lagestabilität und kann als Referenz zu einem sich bewegenden Körper genutzt werden. Dieses, als mechanisches Gyroskop bezeichnete Messinstrument, kommt heutzutage insbesondere in der Luftfahrt zum Einsatz. Ein mechanisches Gyroskop hat jedoch den Nachteil, dass es viele bewegliche Teile beinhaltet und durch kleinste Ungenauigkeiten ein Drift in der Rotation entsteht. Eine andere Möglichkeit der Drehratenmessung ist mit einem optischen Gyroskop. Dabei werden zwei Laserstrahlen in unterschiedlicher Richtung in eine Spule ausgesendet. Bewegt sich der Sender, verlängert sich der Weg für einen

Laserstrahl, während er sich für den anderen verkürzt. Aus der entstehenden Interferenz kann die Drehrate berechnet werden. Dieser Sensoraufbau liefert bessere Messergebnisse, je länger der Weg des Lichts ist. Mit großen Sensoraufbauten lassen sich demnach bessere Messergebnisse erzielen. [75, S. 8 f.]

Auch für die Beschleunigungsmessung kommen mehrere Verfahren in Frage. Ein einfacher Aufbau beinhaltet eine gefedert gelagerte Masse und einen Positionsaufnehmer. Wirkt keine Kraft auf die Masse, befindet sie sich in der Ruheposition. Sobald jedoch eine Kraft wirkt, wird die Masse ausgelenkt und die Position kann mit dem Positionsaufnehmer gemessen werden. Mit der Federkonstante und dem 2. Newtonschen Gesetz lässt sich die Beschleunigung berechnen. [75, S. 14]

Heute üblich sind Micro-Machined Electromechanical System (MEMS) Sensoren, bei denen alle Sensoren in einen Chip integriert sind. Für die Messung der Drehrate wird der Coriolis-Effekt genutzt, der bei Drehung eines Systems eine Kraft auf eine Masse bewirkt. In den Chip sind vibrierende Elemente integriert, mit denen sich diese Kraft messen lässt. Das in ein MEMS IMU integrierte Accelerometer basiert auf den gleichen Prinzipien wie das mechanische Accelerometer. MEMS IMUs besitzen mit der geringen Größe, geringen Fertigungskosten, geringem Energiebedarf und geringen Wartungskosten viele Vorteile, die Messergebnisse sind jedoch nicht so genau wie bei einem klassischen Gyroskop oder Accelerometer. Durch weitere technische Entwicklungen werden auch die MEMS IMUs immer genauer. [75, S. 9 ff.] [67, S. 190 ff.]

3.2.7 Odometrie

Im Bereich der mobilen Roboter wird *Odometrie* als Positionsbestimmung auf Basis der Aktuatorposition verstanden. Der Shared Guide Dog besitzt zwei angetriebene Räder, mit denen er sich fortbewegen kann (vgl. 3.3.3). Die Umdrehungsmessung der Räder kann mit verschiedenen Methoden erfolgen. Weit verbreitet sind Impulsgeber und Hall-Sensoren. Ziel der Odometrie ist es, Bewegungen der Aktuatoren des Roboters aufzunehmen und mittels eines mathematischen Modells eine Positionsabschätzung zu erhalten. Wie schon die Positionsschätzung mittels IMU, wird die Positionsschätzung durch die Odometrie als Koppelnavigation bezeichnet. Da es sich bei dem Shared Guide Dog um einen mobilen Roboter mit differentiellem Antrieb handelt, soll nur dieser hier betrachtet werden. Die Positionsbestimmung mit Odometrie eines differentiellen Antriebs ist mathematisch einfach zu beschreiben, da nur zwei Aktuatoren, nämlich die beiden Räder, einen Einfluss auf die Bewegung des Roboters haben. [61, S. 477 ff.] [18, S. 28 ff.]

Als Impulsgeber werden Scheiben bezeichnet, die an den Rädern befestigt sind und bei Drehung der Räder Impulse ausgeben. Impulsgeber arbeiten mit optischen, elektrischen oder magnetischen Verfahren. In die Scheiben von optischen Impulsgebern sind Schlitze eingearbeitet, durch die ausgesendetes Licht auf einen Sensor treffen kann. Elektrische Impulsgeber besitzen Kontakte anstatt der Schlitze, die einen Kontakt schließen oder öffnen. Bei magnetischen Impulsgebern sind Magnete verbaut, deren Position aufgenommen werden kann.

Es wird unterschieden in Absolutimpulsgeber und Inkrementalgeber. Absolutimpulsgeber geben zu jeder Position des Rades einen eindeutigen Impuls aus. Dazu sind mehrere Reihen von Schlitzen

nebeneinander angeordnet, sodass zu jeder Position ein eindeutiges Muster entsteht. Die Auflösung lässt sich durch die Anzahl der Reihen variieren. Inkrementalgeber geben bei Drehungen nur ein Inkrement aus. Die Scheibe besitzt nur eine Schlitzreihe und durch das Zählen der Impulse lässt sich die Position des Rades bestimmen. Die Auflösung kann durch die Anzahl der Schlitze variiert werden. Ein Nachteil der Inkrementalgeber ist, dass nicht zwischen Vorwärts- und Rückwärtsbewegungen unterschieden werden kann. Dafür sind sie einfacher aufgebaut als Absolutimpulsgeber. [18, S. 28 ff.]

Mit Hall-Sensoren können Änderungen im magnetischen Feld gemessen werden. In bürstenlosen Gleichstrommotoren (BLDC Motor) mit ihrem umlaufenden Magnetfeld kann dadurch die Lage des Rotors gemessen werden. Hall-Sensoren werden oftmals benutzt, um den Strom in BLDC-Motoren zu kommutieren. Die Anordnung der Sensoren erfolgt in einem äquidistanten Abstand zueinander. Dadurch ergibt sich je nach Drehrichtung und -geschwindigkeit des Rotors ein spezifisches Signal.

3.3 Aufbau der Hardwareplattform

Dieser Abschnitt gibt einen Einblick in die Hardware und Aktuatoren des Shared Guide Dog 4.0. Der Shared Guide Dog ist so konzipiert, dass Behinderungen durch Hindernisse im öffentlichen Verkehr autonom abgearbeitet werden können. Das können beispielsweise unebene Wege, Steigungen und Gefälle sein. Nicht möglich ist das Steigen von Treppen, die mehr als zwei Stufen haben. Im aktuellen Entwicklungsstadium handelt es sich bei dem Shared Guide Dog um einen Prototyp, der ständig mit weiteren Funktionen und Sensoren ausgestattet wird. In diesem Kapitel soll der Stand, der für diese Arbeit maßgeblich ist, gezeigt werden.

Eine Übersicht über die Hardwareplattform ist in Abbildung 3.5 dargestellt. Als zentrale Recheneinheit kommt ein Notebook zum Einsatz. Für die Entwicklung des Prototypen bietet es ausreichend Rechenleistung und ist gleichzeitig portabel. Zudem bietet das Notebook Bluetooth und WiFi für die drahtlose Datenübertragung. Das Notebook liest alle Sensordaten über USB- und Ethernet-Schnittstellen. Zur Hinderniserkennung in der Umgebung ist ein Lidar-Sensor vorgesehen. Ein Netzteil stellt die externe Stromversorgung aus dem Akku sicher. Die Sensordaten werden per Ethernet an das Notebook übertragen. Die Steuerbefehle des Computers an die Antriebe werden per USB an einen Arduino Mikrocontroller übertragen, der die Signale in ein UART-Signal übersetzt. Der GPS-Empfänger kann direkt per USB an das Notebook angeschlossen werden. Die IMU, die Berührungssensoren und der 1D-Lasersensor sind an einen Adafruit Feather M0 Mikrocontroller angeschlossen, der die Berührungssensoren und den Lasersensor auswertet und dann die Daten aller Sensoren per USB an das Notebook sendet.

3.3.1 Chassis

Als Basis für den Shared Guide Dog 4.0 kommt ein sogenanntes Veloped des schwedischen Hersteller Trionic zum Einsatz. Der Hersteller bezeichnet das Gerät als „Gelände-Rollator“ [68]. Die Räder sind in einem Dreieck mit einem Rad vorne und zwei Rädern hinten angeordnet. Der Nutzer bewegt sich zwischen den hinteren beiden Rädern. Bei dem vorderen Rad handelt es sich um ein sogenanntes

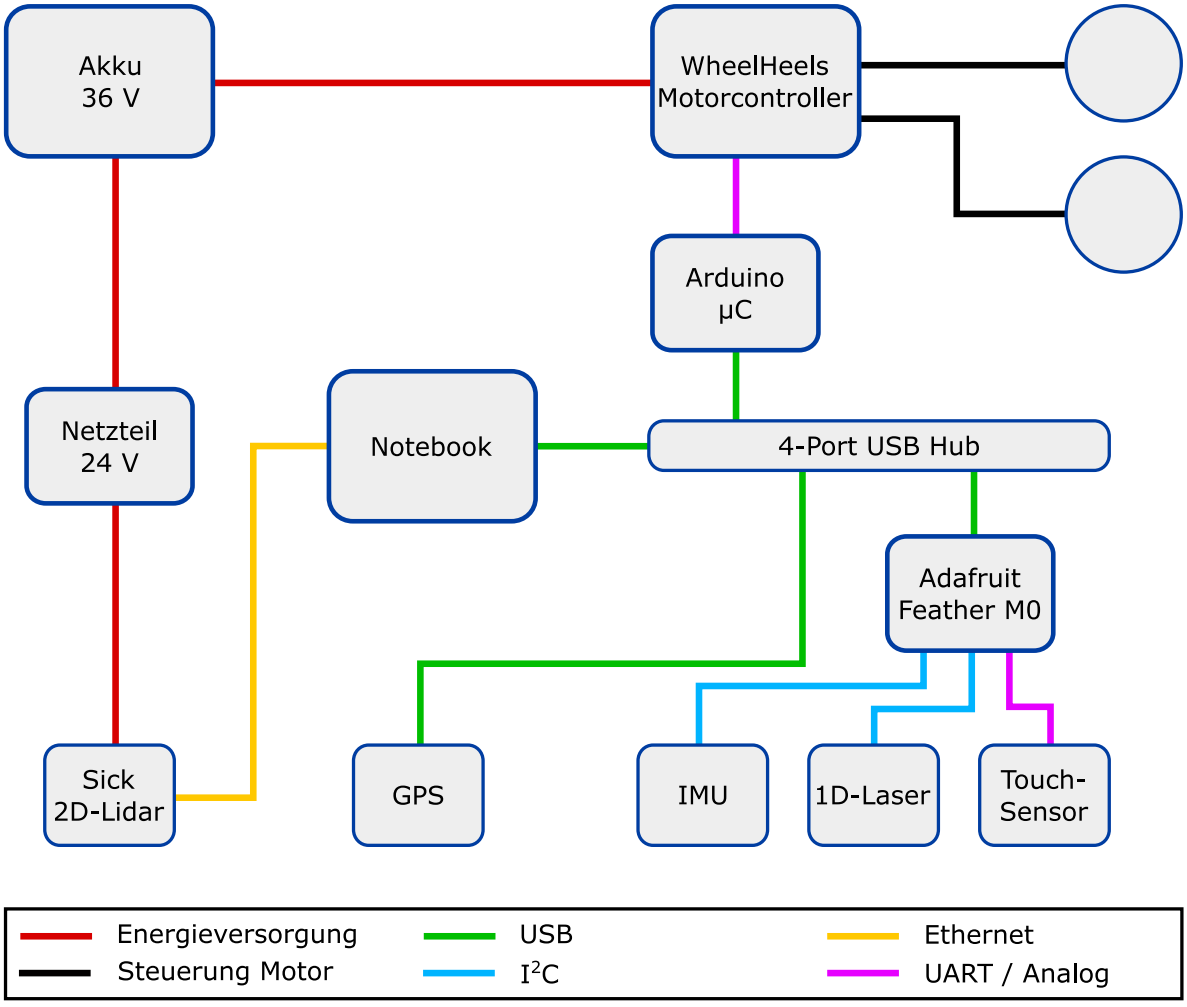

Abbildung 3.5: Schematische Darstellung der Hardwareplattform (Quelle: eigene Darstellung)

Trionic Kletterrad. Dabei ist ein zusätzliches Rad nach vorne-oben versetzt angebracht, das es ermöglicht, Kanten von bis zu 13 cm zu überwinden. Die Außenmaße des Trionic Veloped betragen 76 cm in der Breite und 102 cm in der Länge. [68]

3.3.2 Sensorik

Der Prototyp des Shared Guide Dog 4.0 ist mit einer Grundausstattung an Sensorik versehen. Der Shared Guide Dog besitzt einen GPS-Empfänger, eine IMU und nutzt zusätzlich die Odometriedaten des Motorcontollers zur Positionsbestimmung. Der 1D-Lasersensor und die Berührungssensoren werden als Sicherungssystem und die Anpassung des Shared Guide Dog an den Nutzer oder die Nutzerin eingesetzt. Für die Umgebungserkennung und Hindernisvermeidung wird ein 2D-LiDAR Sensor genutzt.

GPS-Empfänger

Auf dem Shared Guide Dog 4.0 kommt der GPS-Empfänger NL-602U des Herstellers NaviLock mit dem Chipsatz u-blox 6 UBX-G6000-BT zum Einsatz. Er kann bis zu 50 Satelliten auf den GPS-Frequenzen 1575 und 4200 MHz empfangen. Die Update-Rate beträgt bis zu 5 Hz und die Datenausgabe erfolgt im NMEA 0183 Format und wahlweise im herstellereigenen UBX-Format. Das NMEA Protokoll ist ein von der National Marine Electronics Association entwickeltes Standardprotokoll für GPS-Empfänger. Mit dem UBX-Format lassen sich zusätzliche Informationen übertragen, wie beispielsweise die Konfiguration oder der Hardwarestatus des GPS-Empfängers. [38] [69, S. 84 ff.]

Die Positionsgenauigkeit wird mit 2,5 m Circular Error Probable (dt. Streukreisradius) (CEP) und 2,0 m CEP mit SBAS angegeben. Der Streukreisradius bezeichnet den Radius eines Kreises, in dem 50 % aller Messwerte liegen. Neben dem SBAS kann das herstellereigene Aided-GPS System AssistNow in der Online- und Offline-Version zur Verbesserung der Positionsgenauigkeit verwendet werden. AssistNow ist ein kostenloser Service, der mithilfe von Bodenstationen Korrekturdaten für das GPS-Signal zur Verfügung stellt. Die Daten können über das Internet heruntergeladen werden. Der Unterschied zwischen AssistNow Online und AssistNow Offline ist, dass bei der Online-Version eine dauerhafte Internetverbindung bestehen muss, bei der Offline-Version werden die Daten heruntergeladen und zwischengespeichert. Die Daten sind dann bis zu 4 Stunden gültig, können jedoch jederzeit aktualisiert werden. [38] [69, S. 34 ff.]

Der GPS-Empfänger ist am Griff des Trionic Veloped angebracht. Damit wird den Empfehlungen aus [69, S. 45] gefolgt, soweit es die technischen Gegebenheiten zulassen. In Abbildung 3.6 ist die Befestigung des GPS-Empfängers am Prototypen des Shared Guide Dog 4.0 dargestellt.

IMU

Auf dem Shared Guide Dog ist eine Bosch Sensortec BNO055 IMU auf Basis der MEMS Technologie installiert. Sie beinhaltet ein Magnetometer, Accelerometer und Gyroskop in einem Gerät. Zusätzlich ist ein 32-bit Cortex M0 Mikrocontroller integriert, der die Sensordatenfusion übernimmt. Der Chip sitzt auf einer Platine von Adafruit, die alle Anschlüsse als Pins zur Verfügung stellt und eine Ansteuerung mit 3,3 V und 5 V erlaubt. Die IMU kann Beschleunigungswerte in benutzerdefinierten Bereichen zwischen ± 2 g und ± 16 g mit 14 bit Genauigkeit aufnehmen. Das Gyroskop kann Drehraten in einem einstellbaren Bereich zwischen 125 °/s und 2000 °/s mit 16 bit Genauigkeit aufnehmen. Zusätzlich ist ein Magnetometer, das eine absolute Orientierung mit einer Genauigkeit von $\pm 2{,}5$ ° ausgibt, verbaut. Der integrierte Mikrocontroller erlaubt mehrere Betriebsmodi, um die für den Anwendungszweck benötigten Daten auszugeben. Für dieses Projekt wird der *NDOF* Modus verwendet, bei dem alle Sensordaten fusioniert werden und eine absolute Orientierung im Raum ausgegeben wird. Ein Nachteil ist, dass die Messbereiche nicht mehr frei gewählt werden können, sondern für das Accelerometer ein Bereich von ± 4 g und für das Gyroskop ein Bereich von ± 2000 °/s fest vorgegeben wird. Die Datenrate, mit der die fusionierten Sensordaten ausgegeben werden, liegt

für das Accelerometer, das Gyroskop und fusionierte Daten bei 100 Hz, für das Magnetometer bei 20 Hz. [3]

Die IMU reagiert empfindlich auf magnetische Felder in der Nähe des Sensors. Die Positionierung auf dem Shared Guide Dog erfolgt möglichst weit entfernt von anderen Geräten, die magnetische Felder verursachen. Dazu zählen insbesondere das Notebook und der Motorcontroller. Nach verschiedenen Versuchen an mehreren Stellen wurde sich entschieden, die IMU zusammen mit dem GPS-Empfänger in einem Gehäuse am Griff, wie in Abbildung 3.6 gezeigt, unterzubringen.

1D-Laser

Der 1D-Abstandssensor wird zur Messung des Abstands zwischen dem Shared Guide Dog und dem Nutzer oder der Nutzerin eingesetzt. Anhand der Messung wird die Geschwindigkeit an die Gehgeschwindigkeit des Nutzers oder der Nutzerin angepasst. Es kommt ein VL53L1X Time-of-Flight (ToF) Laser-Abstandssensor von ST Microelectronics zum Einsatz. Der Sensor ist auf einem Pololu PCB mit Spannungsregelung und Pinouts für den Anschluss an einen Mikrocontroller installiert [59]. Der Sensor nutzt einen 940 nm Laser, mit dem sich Distanzen von bis zu 4 m und 50 Hz messen lassen. [63]

Die Installation auf dem Shared Guide Dog erfolgt in der Mitte des Lenkers, sodass der Bereich zwischen den Handgriffen gemessen werden kann. Der Sensor ist über I^2C an den Adafruit Feather M0 angeschlossen, der die Auswertung der Daten und Kommunikation mit dem Notebook übernimmt.

Berührungssensoren

Neben dem 1D-Laser kommen zwei Berührungssensoren an den Handgriffen als zweites Sicherungssystem für den Shared Guide Dog zum Einsatz. Die Funktionsweise basiert auf dem Prinzip der Kapazitätsmessung. Die Berührungssensoren bestehen aus jeweils einem Streifen Aluminiumfolie, der an den Adafruit Feather M0 angeschlossen ist. Wird der Aluminiumstreifen von einer Person angefasst, lässt sich eine Änderung der Kapazität messen. Die Auswertung der Kapazitätsmessung erfolgt durch den Adafruit Mikrocontroller, der die Daten anschließend per USB an das Notebook sendet.

Lidar

Zur Überwachung der Umgebung und der Erkennung von Hindernissen, ist auf dem Shared Guide Dog ein Lidar-Sensor vorgesehen. Bei dem Sensor handelt es sich um einen Sick TiM571-2050101 Outdoor-2D-Lidar Sensor. Er arbeitet mit einem für den Menschen ungefährlichen und unsichtbaren 850 nm Infrarotlaser. Mit einer Frequenz von 15 Hz kann ein Bereich von 270 ° mit einer Auflösung von 0,33 ° abgedeckt werden. Bei der maximalen Messdistanz von 25 m liegen die Messpunkte ungefähr 14,5 cm auseinander. [62]

Der Lidar Sensor ist ganz vorne am Shared Guide Dog oberhalb des Trionic Kletterrads befestigt. In einer Höhe von ungefähr 50 cm über dem Boden tastet der Sensor eine horizontale Ebene ab.

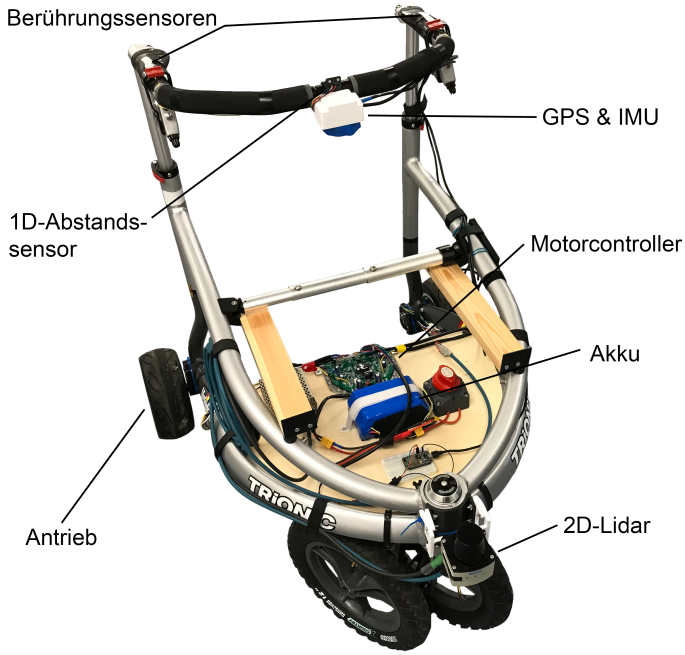

Abbildung 3.6: Übersicht über die Hardwareplattform des Shared Guide Dog 4.0 (Quelle: eigene Darstellung)

Die Nullachse des Sensors ist an der Fahrrichtung des Shared Guide Dog ausgerichtet, so dass auf beiden Seiten ein Feld von 135 °detektierbar ist.

3.3.3 Antrieb

Der Antrieb des Shared Guide Dog erfolgt über die beiden Hinterräder. Damit ergibt sich ein differentieller Antrieb mit dem Vorteil, dass ein einfaches kinematisches Modell zur Beschreibung genutzt werden kann. Bei der Bewegungsfreiheit ergeben sich Einschränkungen, da sich der Roboter nicht seitlich bewegen kann. Auf diese Bewegungsrichtung kann jedoch verzichtet werden, da sich der Mensch überwiegend nach vorne bewegt und auch Kurven oft mit einer Vorwärtsbewegung verknüpft sind.

Als Motoren kommen zwei Brushless DC Motoren aus einem Hoverboard des Herstellers WheelHeels zum Einsatz. Sie sind in die Reifen integriert und leisten bei 36 V jeweils bis zu 350 W. Damit kann eine Geschwindigkeit von bis zu 20 km/h erreicht werden. Die Ansteuerung erfolgt mit dem mitgelieferten Controller und einem Arduino Mikrocontroller, der den Anschluss an einen USB-Port des Computers ermöglicht. Über die in die Motoren integrierten Hall-Sensoren, kann die aktuelle Drehgeschwindigkeit ausgelesen werden. Die Platzierung der Motoren und des Controllers am Shared Guide Dog ist in Abbildung 3.6 gezeigt.

4 Roboterkontrollarchitekturen

Die Navigation eines Roboters beinhaltet die Lokalisation, die Pfadplanung und die Steuerung. Die letzten beiden Kapitel haben eine Einführung in die Themen Lokalisation und Pfadplanung gegeben. Das folgende Kapitel beschäftigt sich mit der noch fehlenden Aufgabe der Steuerung.

4.1 Entwicklung von Roboterkontrollarchitekturen

Das Ziel bei der Entwicklung einer Steuerung ist, alle auftretenden Aufgaben, die der Roboter bei der Bearbeitung seiner Aufgabe bewältigen muss, zu einem einzigen Kontrollprogramm zusammenzufassen. Die Softwarestruktur dieses Programms wird als Roboterkontrollarchitektur (engl. Robot Control Architecture) bezeichnet. [18, S. 317]

Die erste Architektur bestand aus den drei Elementen sensing (dt. Erfassung), planning (dt. Planen) und acting (dt. Ausführen) und wird als sense-plan-act (SPA) bezeichnet. Beim Erfassen wird die Umgebung mithilfe von Sensoren erfasst und in ein internes Modell übersetzt. Das Modell wird vom Planungselement genutzt, um einen Plan zur Erfüllung eines Ziels zu erzeugen. Dieser Plan wird vom Ausführungselement verwendet, um die Aktuatoren zu steuern. Aus dieser Architektur ergeben sich zwei Nachteile. Zum einen wird der Roboter während der Planungsphase blockiert bis der Plan erstellt ist. Zum anderen wird bei der Ausführung eines Plans die Umgebung nicht dauerhaft beobachtet. Das kann zu gefährlichen Situationen in dynamischen Umgebungen führen. [61, S. 189]

In der Folge wurden immer mehr Architekturen entwickelt, die eine schnelle Reaktion auf geänderte Umgebungsbedingungen zulassen. Durchgesetzt hat sich die Subsumption Architektur. Eine Subsumption Architektur besteht aus mehreren Schichten von Zustandsmaschinen (engl. finite state machine), die hierarchisch geordnet sind. Die Idee ist, dass höhere Schichten die Befehle von niedrigeren Schichten überschreiben können. Die unterste Schicht kann beispielsweise den Befehl senden, dass der Roboter geradeaus fahren soll. Darüber liegt eine Schicht, die die Umgebung mit Sensoren überwacht und wenn ein Hindernis im Weg des Roboters liegt, einen Befehl zum Drehen sendet. Sobald das Hindernis nicht mehr im Weg liegt, übernimmt wieder die untere Schicht und der Roboter fährt weiter geradeaus. Die Planung und Optimierung von komplexen Aufgaben ist mit einer Subsumption Architektur nur schwer möglich, da durch die Schichten nur die nähere zeitliche und räumliche Umgebung betrachtet wird. [61, S. 189 f.]

Heutzutage wird oftmals ein hybrider Ansatz als Roboterkontrollarchitektur verwendet. Für die Planung und Optimierung langer und komplexer Aufgaben kommt eine SPA Architektur zum

Einsatz. Die Reaktion auf die dynamische Umgebung übernimmt eine Subsumption Architektur. [61, S. 191 ff.]

4.2 Robot Operating System 2

In diesem Kapitel soll ein kurzer Einblick in die Konzepte und den Aufbau des Robot Operating System (ROS) in der zweiten Version gegeben werden. Dabei sollen die technischen Hintergründe und Implementierungen grundlegend erläutert werden, sofern es für das Verständnis dieser Arbeit erforderlich ist.

Das ROS ist ein Open-Source-Software-Framework für Roboter. Entwickelt seit 2007 für den Willow Garage PR2 Roboter, wird es seitdem für viele verschiedene Roboterprojekte eingesetzt. Seit 2014 befindet sich ROS2 in der Entwicklung. Es bringt viele Vorteile und Neuerungen, beispielsweise Sicherheitsverbesserungen in der Nachrichtenübetragung und die Echtzeitfähigkeit. ROS liegt als Middleware-Layer zwischen dem Betriebssystem (OS) und der benutzerspezifischen Anwendung und stellt verschiedene Tools und Softwarebibliotheken zur Anwendungsentwicklung bereit. Eines dieser Pakete ist Navigation 2, das während dieser Arbeit verstärkt genutzt wird und deshalb in Abschnitt 4.3 ausführlicher behandelt werden soll. [14]

4.2.1 Kommunikationsstrukturen

Die Architektur eines ROS2-Programms basiert auf dem ROS-Graphen. Als ROS-Graph werden alle Knoten (engl. Nodes) und deren Verbindungen, über die sie kommunizieren können, zusammengefasst. Knoten sind der Hauptbestandteil eines jeden ROS Programms. Es handelt sich dabei um kleine Programmbausteine, die über die Verbindungen zu anderen Knoten kommunizieren können. Darüber hinaus können sie weitere Dienste anbieten, wie beispielsweise Services und Actions. Knoten sind üblicherweise für eine bestimmte Funktionalität zuständig, zum Beispiel für die Steuerung eines Motors. [40]

Der ROS-Graph basiert auf einer sogenannten Publisher-Subscriber Architektur. Knoten können Daten in sogenannten *Topics* veröffentlichen (engl. publish), die von anderen Knoten abonniert (engl. subscribe) werden können. Die Kommunikation zwischen zwei Knoten erfolgt mit klar definierten Datentypen, den *Messages*, die an die Topics gesendet oder von diesen empfangen werden können. [40]

Neben der Kommunikation über Topics können Knoten Services und Actions anbieten. Services werden genutzt, wenn zwischen zwei Knoten nur für bestimmte Aufgaben Daten ausgetauscht werden müssen. Dazu sendet ein Knoten eine Service-Anfrage (engl. service request) an einen Service-Server eines Knotens. Dieser Knoten führt die Aufgabe aus und sendet anschließend eine Service-Antwort (engl. service response) an den anfragenden Knoten. Typische Aufgaben für Services sind zum Beispiel der Reset eines Roboters oder die Berechnung einer globalen Route. [41]

Die letzte hier vorgestellte Art der Kommunikation erfolgt über Actions. Als Actions werden lange laufende Aufgaben bezeichnet, wie zum Beispiel die Navigation entlang eines Weges. Eine Action

besteht aus einem Anfrage-Service, regelmäßigen Zwischenberichten (engl. feedback) und einem Ergebnis-Service. Die Anfrage und das Ergebnis werden an einen Service-Server gesendet, da Actions auch abgelehnt werden können. [39]

4.2.2 ROS Enhancement Proposals (REP)

ROS2 besitzt ein System der sogenannten ROS Enhancement Proposals (REP), in denen Designrichtlinien, technische Spezifikationen oder neue Funktionen zusammengefasst werden. Das System ist vergleichbar mit einer Normenreihe für ROS. Für diese Arbeit werden REP 103 und REP 105 verwendet. REP 105 definiert Koordinatensysteme für mobile Roboter. Es wird spezifiziert, dass mindestens die drei Koordinatensysteme *map*, *odom* und *base_link* und die Transformationen zwischen diesen vorhanden sein müssen. Das *map* Koordinatensystem ist fest mit der Karte verknüpft und das *base_link* Koordinatensystem mit der Roboterplattform. Das *odom* Koordinatensystem ist ein lokales Koordinatensystem, in dem die Roboterposition kontinuierlich abgebildet wird. Ein GPS-Empfänger, dessen Signal diskrete Sprünge haben kann, kann somit nicht für die Positionsbestimmung im *odom* Koordinatensystem eingesetzt werden. Stattdessen werden beispielsweise Odometriedaten oder Daten einer IMU verwendet. Mit dem GPS-Empfänger kann die Position im *map* Koordinatensystem bestimmt werden, die diskrete Sprünge aufweisen darf. [32]

REP 103 definiert Einheiten und Konventionen für Koordinatensysteme. Als Standardeinheit für alle Berechnungen in ROS werden SI-Einheiten und die daraus abgeleiteten Einheiten festgelegt. Für Koordinatensysteme wird die Ausrichtung auf einem mobilen Roboter spezifiziert. Es handelt sich bei allen Koordinatensystemen um Rechte-Hand-Systeme, die auf einem mobilen Roboter mit der x-Achse nach vorne zeigen. Die z-Achse zeigt nach oben und die y-Achse nach links. Neben diesem Koordinatensystem werden weitere Systeme zum Beispiel für Kameras definiert. [13]

4.3 Navigation 2

Navigation 2 ist ein Paket mit nützlichen Funktionen zur Navigation und Steuerung eines Roboters in ROS2. Das Ziel des Navigation 2 Pakets ist ein Software-Framework zu erstellen, mit dem sich unterschiedliche Arten von autonomen, mobilen Robotern steuern lassen. Das Paket stellt unter anderem Funktionen zur Speicherung und Verwaltung von Karten, Lokalisierung, Pfadplanung, Steuerung verschiedener Antriebe und Verhaltensmodellierung bereit. Mit diesen Funktionen bietet das Navigation 2 Paket einen einfachen Einstieg in die Roboternavigation, bleibt aber gleichzeitig vielseitig einsetzbar und hochgradig konfigurierbar. [28]

Das Navigation 2 Paket ist in einen *Behaviour Tree (BT) Navigator Server*, *Recovery Server*, *Controller Server* und *Planner Server* aufgeteilt. Der BT Navigator Server koordiniert den Planner Server, der für das Planen eines Weges zuständig ist, den Controller Server, der den mobilen Roboter steuert und den Recovery Server, der Wiederherstellungsstrategien implementiert. Ein Schema der Architektur ist in Abbildung 4.1 gezeigt. Oben in der Mitte steht der BT Navigator Server mit Verbindungen zu dem Recovery Server, Controller Server und Planner Server. Jeder

dieser Aufgabenbereiche wird von einem Server behandelt, der eine Standard-Plugin-Schnittstelle implementiert. Die Schnittstelle erlaubt es Algorithmen und Verhaltensweisen während der Laufzeit auszutauschen. [30, S. 1 ff.]

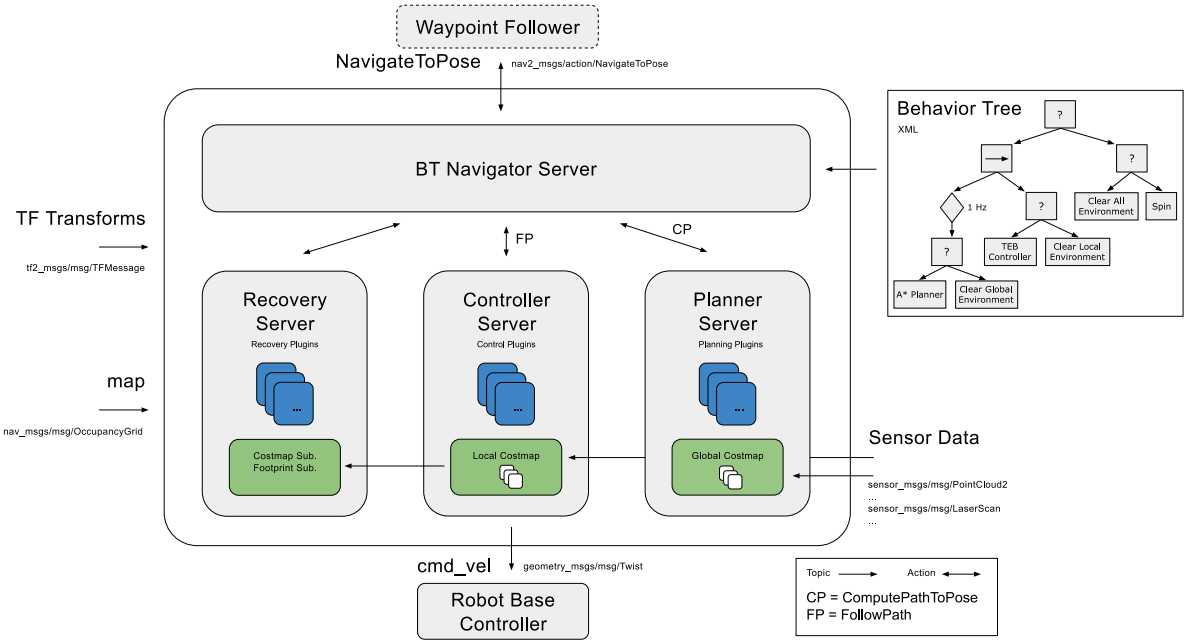

Abbildung 4.1: Navigation 2 Architektur (Quelle: [28], [30, S. 4])

Grundlage für die Koordinierung ist ein BT. Ein BT strukturiert Verhaltensweisen in einer Baumstruktur. Ein beispielhafter BT ist in der Abbildung auf der rechten Seite als Eingabe für den BT Navigator Server gezeigt. Die Kästen mit einem Fragezeichen sind Rückfallknoten und der Kasten mit dem Pfeil ein sequenzieller Knoten. Ein Behavior Tree wird von links nach rechts gelesen. Der gezeigte BT wird mit einer Rate von 1 Hz den A* Planner aufrufen, um einen Pfad zum nächsten Zielpunkt zu berechnen. Schlägt diese Operation fehl, wird die nächste Funktion innerhalb des Rückfallknotens ausgeführt, in diesem Fall ist das *Clear Global Environment*. Der sequenzielle Knoten im BT bewirkt eine Ausführung der linken Knoten und anschließend der rechten Knoten. Nach dem Planen des Pfades wird der *TEB Controller* aufgerufen, der für die Steuerung des Roboters entlang des Pfades verantwortlich ist. Schlägt eine dieser Funktionen fehl, fällt der Algorithmus zurück auf den obersten Knoten und wird dann anfangen die Recovery-Strategien auf der rechten Seite auszuführen. Je nach aufgerufener Funktion des Behavior Trees wird eine Funktion der drei Server gestartet. [30, S. 3 f.]

Damit das Navigation 2 Framework verwendet werden kann, müssen einige Sensordaten, eine Karte und sogenannte *Transforms* bereitgestellt werden. Die Sensordaten werden benötigt, um auf Hindernisse reagieren zu können und einen lokalen Plan zum Umfahren zu generieren. Zudem kann mit dem Navigation 2 Paket eine Lokalisation auf Basis der Sensordaten durchgeführt werden. Eine Karte kann optional vorgegeben werden. Wird keine Karte vorgegeben, können SLAM Algorithmen

verwendet werden, um eine Karte zu erstellen. Die *TF Transforms* beinhalten die Koordinatentransformation zwischen den verschiedenen in REP 105 definierten Koordinatensystemen. Der Output des Navigation 2 Pakets ist ein Geschwindigkeitsbefehl für die Antriebe. Dieser Befehl wird auf dem Topic *cmd_vel* veröffentlicht. [30, S. 5] [29] [32]

5 Anforderung an die Navigationslösung

Die Anforderungen wurden während mehrerer Treffen mit einigen blinden Bewohnern der Wohngruppe Stiftstraße und dem Vorsitzenden des Blinden- und Sehbehindertenvereins Hamburg (BSVH) erarbeitet. Zudem werden einige weitere Anforderungen entwickelt, die die Sicherheit gewährleisten sollen. Die Anforderungen werden ähnlich wie die drei vorigen Kapitel in die Bereiche Lokalisation, Pfadplanung und Steuerung eingeteilt.

5.1 Anforderungen an die Pfadplanung

Für die Berechnung der Route gibt es zwei entscheidende Sicherheitsanforderungen. Zum einen darf die Navigation nicht starten, wenn der Start- oder Endpunkt zu weit von einem Kartenpunkt entfernt liegt (P-2). Bei einer zu großen Entfernung besteht die Gefahr, dass auf dem Weg zum Start- oder Endpunkt Hindernisse oder Absätze nicht erkannt werden und sich somit eine Gefährdung für den Nutzer oder die Nutzerin ergibt. Zum anderen müssen bestimmte Strecken temporär von der Routenplanung ausgeschlossen werden können, wie zum Beispiel Baustellen (P-3). Bereits bei der Routenplanung sollen entsprechende Wege umgangen werden, um die Gefahren einer Baustelle, wie Baustellenverkehr durch große Fahrzeuge, stark verschmutzte Wege oder schlecht abgesicherte Baustellen zu vermeiden. Zudem stellt die Anforderung einen Komfortgewinn für den Nutzer oder die Nutzerin dar, da die Route nicht während der Fahrt umgeplant werden muss und somit Umwege vermieden werden können. Durch das Rechtshalten auf dem Weg (P-4) wird die Sicherheit weiter erhöht, da es in Deutschland üblich ist, sich auf der rechten Seite eines Weges zu bewegen. Der Shared Guide Dog bewegt sich demnach in die gleiche Richtung wie andere Personen und muss seltener auf entgegenkommende Personen reagieren.

Bei der Eingabe eines Ziels soll der Nutzer oder die Nutzerin die Möglichkeit haben, eine Adresse einzugeben (P-8), da nicht davon auszugehen ist, dass die Längen- und Breitengrade des Ziels bekannt sind. Nach der Zieleingabe soll innerhalb einer kurzen Zeit die Route berechnet (P-6) und über die voraussichtliche Wegzeit informiert werden (P-7).

Tabelle 5.1 listet alle Anforderung an die Pfadplanung mit den zu prüfenden Zahlenwerten auf.

5.2 Anforderungen an die Lokalisation

Die wichtigste Anforderung an die Lokalisation ist die Abweichung von der Sollposition. Diese Anforderung geht von einem Weg mit 1,5 m Breite aus. Bei einer Breite des Shared Guide Dogs von 0,76 m (vgl. Kapitel 3.3) und einer Positionierungsabweichung von 0,30 m in jede Richtung benötigt

Tabelle 5.1: Anforderungen an die Pfadplanung und die Karte

		Zahlenwert			
Lfd. Nr.	Definition	min	max	Einheit	F / W
P-1	Zuverlässigkeit der Routenfindung				F
P-2	Abweichung des Start-/Endpunkts von einem Kartenpunkt.		5	m	F
P-3	Die Navigation muss bestimmte Strecken meiden können.				F
P-4	Der Shared Guide Dog soll sich auf der rechten Weghälfte bewegen.				W
P-5	Die Karte soll einfach aktualisierbar sein.				W
P-6	Zeit für die Berechnung der Route.		5	s	W
P-7	Der Nutzer soll über die voraussichtliche Wegzeit informiert werden.				W
P-8	Die Eingabe des Ziels erfolgt über die Adresse.				W
P-9	Es sollen verschiedene Nutzer berücksichtigt werden können.				F
P-10	Die Karte soll nur Punkte aus dem Testgebiet berücksichtigen.				F

der Shared Guide Dog bereits 1,36 m der Wegbreite. Die verbleibenden 14 cm sollen nicht befahren werden, da an den Wegrändern oftmals Gras wächst und dieses nicht überfahren werden soll. Daraus ergibt sich Anforderung L-1.

Die Anforderung L-2 ergibt sich aus Kapitel 3.3. Da auf dem Prototyp des Shared Guide Dog nur ein GPS-Empfänger, eine IMU und Odometriedaten verfügbar sind, soll die Lokalisation nur mit diesen Sensoren arbeiten. Der 2D-Lidar Sensor wird während dieser Arbeit nicht zur Lokalisierung, sondern nur für die Hinderniserkennung verwendet.

In Tabelle 5.2 sind alle Anforderungen an die Lokalisation aufgelistet.

Tabelle 5.2: Anforderungen an die Lokalisation

		Zahlenwert			
Lfd. Nr.	Definition	min	max	Einheit	F / W
L-1	Abweichung von der Sollposition		0,3	m	F
L-2	Lokalisierung nur mit GPS, IMU und Odometrie				F

5.3 Anforderungen an die Steuerung

Unabhängig von der Navigation darf der Energiespeicher des Shared Guide Dog einen gewissen Ladezustand nicht unterschreiten. Da es bei Li-Ionen Akkus nicht empfehlenswert ist, diese vollständig zu entladen, soll am Ende der Fahrt noch eine Restladung von mindestens 20 % vorhanden sein.

Damit bleibt noch ein Puffer, um auf unerwartete Ereignisse reagieren oder wieder zurück zur Ladestation fahren zu können.

Während der Abarbeitung des Weges ist besonders die Hinderniserkennung und Vermeidung von Zusammenstößen entscheidend. Die Hinderniserkennung muss Objekte mit einer für den Shared Guide Dog sichtbaren Fläche von mindestens 50 mm erkennen können. Dieses Maß entspricht der Kante einer handelsüblichen Leitbake, wie sie oft für Baustellenabsicherungen genutzt wird. Absperrpfosten, wie sie beispielsweise für Parkplätze genutzt werden, haben üblicherweise eine Breite von 60 mm und mehr und fallen somit auch unter dieses Maß (vgl. [2] und [22]). Nach der Identifizierung eines Hindernisses, darf ein Sicherheitsabstand von 100 cm nicht unterschritten werden. Wird dieser doch unterschritten, muss der Shared Guide Dog die Fahrt unterbrechen und kann erst weiterfahren, wenn das Hindernis nicht mehr im Weg ist.

Während der Abarbeitung der Route, soll der Nutzer möglichst gleichmäßig geführt werden. Daraus ergibt sich Anforderung S-5, dass auf Hindernisse frühzeitig reagiert wird und die Geschwindigkeit entsprechend angepasst wird.

Tabelle 5.3: Anforderungen an die Steuerung

		Zahlenwert			
Lfd. Nr.	Definition	min	max	Einheit	F / W
S-1	Batterieladezustand am Ende der Fahrt.	20	-	%	F
S-2	Stopp bei Annäherung an ein Hindernis		1,0	m	F
S-3	Das Fahrzeug darf nur fahren, wenn beide Handgriffe festgehalten werden.				F
S-4	Größe von Objekten, die erkannt werden müssen	50		mm	F
S-5	Verlangsamung des Geräts bei einer Hinderniserkennung ab einer gewissen Distanz.	10		m	W

6 Lösungsansatz für die Navigation

Die Entwicklung des Lösungsansatzes erfolgt ausgehend vom Shared Guide Dog, denn das Ziel ist, dass sich der Shared Guide Dog im Testgebiet (vgl. Kapitel 1.3) lokalisieren und ausgehend von der Startposition zu einem Ziel navigieren kann. Für die Lokalisation und Navigation wird als Grundlage eine Karte benötigt. Anhand dieser Karte soll es möglich sein, einen Pfad zu berechnen.

Bei der Wahl einer Karte gibt es die Möglichkeit eine eigene Karte zu erstellen, zum Beispiel durch SLAM Algorithmen, es können aber auch Karten aus geographischen Informationssystemen genutzt werden. Die Erstellung einer eigenen Karte für den Shared Guide Dog ist nur mit hohem Aufwand umzusetzen. Auch wenn das Testgebiet mit knapp 40.000 m^2 nicht übermäßig groß ist, nimmt eine eigene Kartierung viel Zeit für die Erstellung und Wartung in Anspruch. Insbesondere die Berücksichtigung von dynamischen Hindernissen, wie beispielsweise parkenden Autos, ist aufwendig. Sinnvoller ist die Nutzung eines geographischen Informationssystems. Es bietet die Vorteile, dass Kartendaten sofort verfügbar sind und die Karten bereits sehr viele Informationen über die Umgebung enthalten. Die Wahl des Informationssystems fällt auf die Karte von Open Street Map (OSM). Im Gegensatz zu vielen anderen Anbietern ist sie kostenlos und kann frei für die eigenen Anwendungszwecke verwendet werden. [43]

Preprocessing der Kartendaten

Bevor die OSM Karte verwendet werden kann, muss sie auf Fehler überprüft, eventuell fehlende Informationen hinzugefügt und in ein Format übersetzt werden, das eine effiziente Pfadplanung ermöglicht. Anhand der Daten wird ein Pfad berechnet, den der Shared Guide Dog abfahren kann. Um alle diese Anforderungen zu erfüllen, wird ein Lösungsansatz entwickelt, der in Abbildung 6.1 dargestellt ist. Der Lösungsansatz ist in die Datenverwaltung (Kapitel 6.1), die Pfadplanung (Kapitel 6.2) und die Kartennutzung auf dem Shared Guide Dog (Kapitel 6.3) geteilt.

6.1 Datenverwaltung

Das Ziel der Datenverwaltung ist, dass dem Shared Guide Dog ein konsistenter Datensatz zur Verfügung gestellt wird. Dazu werden zwei Teilbereiche spezifiziert. Der eine Teilbereich beinhaltet die Aufgaben Überprüfen, Augmentieren und Bereitstellen der Daten für die Pfadplanung. Der andere Teilbereich übernimmt die Hindernisverwaltung.

Bevor mit den Open Street Map Kartendaten gearbeitet werden kann, müssen diese heruntergeladen werden. Dabei wird eine Datei im XML-Schema mit den Kartendaten erstellt (vgl. Kapitel

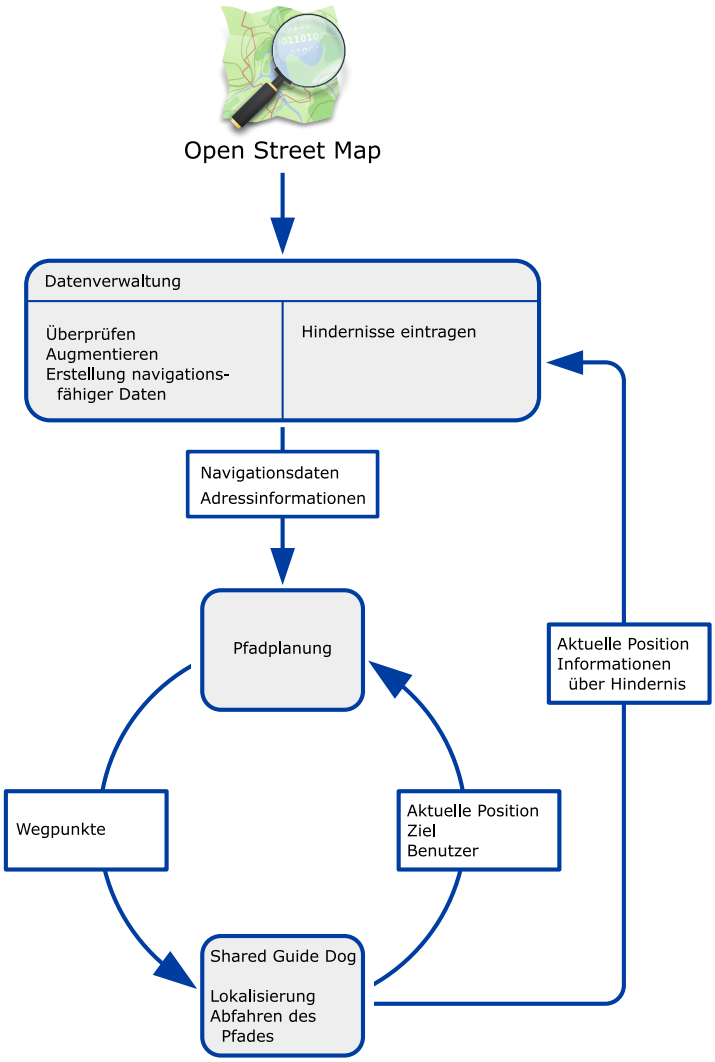

Abbildung 6.1: Lösungsansatz für die Umsetzung der Navigationslösung (Quelle: OSM Logo: OSM contributors, eigene Darstellung)

2.2.1). Diese Datei ist die Eingabe für die Datenverwaltung. Der erste Schritt zur Erstellung einer Karte für den Shared Guide Dog sieht die Überprüfung der OSM Daten vor. Zur Überprüfung der Karte werden die folgenden, in DIN EN ISO 19157 [7, S. 8 ff.] spezifizierten Kriterien, verwendet.

- Vollständigkeit der Daten (completeness)
- Logische Konsistenz (logical consistency)
- Lagegenauigkeit (positional accuracy)
- Thematische Genauigkeit (thematic accuracy)
- Zeitliche Genauigkeit (temporal accuracy)

Eine ausführliche Analyse der einzelnen Kriterien unter Berücksichtigung des Testgebiets wird in Kapitel 7.2 durchgeführt. Aus dieser Analyse können konkrete Maßnahmen zur Verbesserung der Qualität abgeleitet werden, wenn ein Kriterium nicht zur Zufriedenheit erfüllt werden kann. Nach der Überprüfung der Daten folgt die Anreicherung der Karte mit für den Shared Guide Dog benötigten Daten. Die Anreicherung ist notwendig, da Anforderung P-3 fordert, dass es möglich sein soll bestimmte Strecken zu meiden. Die Erfüllung von Anforderung P-9 verlangt darüber hinaus die Möglichkeit zur Berücksichtigung unterschiedlicher Nutzerprofile bei der Routenplanung. Zur Umsetzung der ersten Anforderung gibt es die Möglichkeiten, die zu meidende Strecke aus der Karte zu löschen oder die Strecke mit einer Kennzeichnung zu versehen, dass sie nicht befahren werden darf. Das Löschen der Strecke hat den Nachteil, dass dieser Vorgang nicht einfach rückgängig gemacht werden kann. Eine Kennzeichnung der Strecke hat hingegen den Vorteil, dass Strecken nur temporär gesperrt werden können, beispielsweise bei kurzzeitigen Baustellen oder Pfützen durch starken Regen. Das Open Street Map Datenformat bietet mit Attributen, die zu Knoten, Wegen und Relationen hinzugefügt werden können, eine einfache und elegante Möglichkeit zur Umsetzung dieser Anforderung, ohne dass das Datenformat angepasst werden muss (vgl. Kapitel 2.2.1). Auch die Berücksichtigung unterschiedlicher Nutzerprofile kann mit OSM Attributen ermöglicht werden. Bei der Auswahl des Pfadplanungsalgorithmus muss darauf geachtet werden, dass Wege mit unterschiedlichen Eigenschaften und Attributen in der Kostenfunktion berücksichtigt werden können.

Beim Augmentieren der Daten gibt es eine weitere Anforderung, die für den Lösungsansatz beachtet werden muss. Es handelt sich dabei um Anforderung P-4, in der gefordert wird, dass der Shared Guide Dog nach Möglichkeit rechts auf dem Weg fährt. Während der Entwicklung wurden zwei Möglichkeiten zur Lösung dieser Anforderung identifiziert. Die erste Möglichkeit sieht das Einfügen weiterer Wege parallel zu den bestehenden Wegen vor, sodass ein Netz aus Wegen entlang der realen Wege entsteht. Die zweite Möglichkeit ist, diese Aufgabe dem lokalen Controller zu übertragen. Dieser kennt die Breite und Mitte des Weges aus der Karte und kann den Shared Guide Dog auf der rechten Seite fahren lassen. Die Wahl für den Prototypen fällt auf die erste Variante. Sie bietet den Vorteil, dass eine Hindernisumfahrung bereits bei der Routenplanung berücksichtigt werden kann. Ein lokaler Controller würde das Hindernis erst kurz vorher erkennen und dann eine Ausweichstrategie starten. Das kann für den Nutzer oder die Nutzerin unerwartet und unangenehm sein. Im weiteren Verlauf des Projekts muss der lokale Controller jedoch weiter betrachtet werden, da er für die Vermeidung von Zusammenstößen mit dynamischen und neuen Hindernissen benötigt wird. Die Entwicklung eines lokalen Controllers wird in dieser Arbeit nicht behandelt.

Die Auswahl eines Pfadplanungsalgorithmus erfolgt in Kapitel 6.2 auf Grundlage der in Kapitel 2.3 vorgestellten Algorithmen. Die Daten für diese Algorithmen werden als Graph benötigt. Das bedeutet, dass von jedem Knoten der Karte sofort ersichtlich ist, welche anderen Knoten direkt erreicht werden können.

Hindernisverwaltung

Die Aufgabe der Hindernisverwaltung besteht im Eintragen von Hindernissen in die Karte, dem Entfernen von Hindernissen und Veröffentlichen der Position von Hindernissen für andere Shared Guide Dogs. Während der Fahrt des Shared Guide Dogs wird die Umgebung durch den Laserscanner überwacht. Befindet sich ein Hindernis auf dem Weg, wird dieses detektiert und der Shared Guide Dog stoppt seine Fahrt. Das weitere Vorgehen hängt von der Art des Hindernisses ab. Handelt es sich um ein dynamisches Hindernis, wie beispielsweise einen Fußgänger oder einen Fahrradfahrer, wird sich das Hindernis nach kurzer Zeit weiterbewegen und den Shared Guide Dog nicht behindern. Bei einem statischen Hindernis, darunter werden hier beispielsweise parkende Autos und stehende Menschen verstanden, bewegt sich das Hindernis nicht weiter und der Shared Guide Dog muss eine Ausweichroute finden. Zur Unterscheidung eines dynamischen von einem statischen Hindernis, soll eine gewisse Zeit gewartet werden, bevor das Hindernis an die Hindernisverwaltung gemeldet wird. Für die Meldung eines Hindernisses ist die aktuelle Position und der nächste Punkt auf dem Pfad erforderlich. Die Hindernisverwaltung sucht diese beiden Knoten in der Karte und sperrt sie für weitere Routenberechnungen durch das Hinzufügen eines Attributes. Das Hinzufügen des Attributes bietet den Vorteil, dass die Kostenfunktion nicht angepasst werden muss, da sie schon aufgrund der Anforderungen aus dem letzten Abschnitt gesperrte Knoten behandeln kann. Zudem sind alle geographischen Daten aus der Karte verfügbar und es muss keine Synchronisierung zwischen den Wegpunkten und Hindernissen durchgeführt werden.

Die Nutzung von Attributen erlaubt es darüber hinaus, Hindernisse auch wieder zu entfernen. Dieser Aspekt ist wichtig, damit die Pfadplanung keine unnötigen Umwege einplant. Zur Auswahl stehen die Möglichkeiten, das Hindernis nach einer gewissen Zeit automatisch zu entfernen oder einen Shared Guide Dog ohne Nutzer oder Nutzerin das Hindernis kontrollieren zu lassen. Als Ausblick kann eine Überwachung der Straßen und Wege des Testgebiets als dritte Möglichkeit betrachtet werden. Das Projekt *Sensorknoten* arbeitet an einer Lösung, mit der diese Aufgabe erfüllt werden kann. Da zur Zeit der Erstellung dieser Arbeit jedoch kein Sensorknoten zur Verfügung steht, wird diese Möglichkeit vorerst verworfen. Aus den verbleibenden zwei Möglichkeiten findet die Löschung nach einer definierten Zeit Anwendung. Die Möglichkeit einen Shared Guide Dog zum Kontrollieren fahren zu lassen hat, den entscheidenden Nachteil, dass das Betreiben eines autonomen Roboters in Deutschland an viele Auflagen und Regularien geknüpft ist, die mit dem Prototyp nicht erfüllt werden können. Diese Lösung kann im weiteren Verlauf des Projekts erneut betrachtet werden.

Der Fokus bei der Umsetzung für den Prototypen liegt auf dem Eintragen und Entfernen von Hindernissen aus der Karte. Das Veröffentlichen ist für den in dieser Arbeit genutzten Prototypen vorerst nicht wichtig, da es nur ein Fahrzeug gibt. Sobald mehrere Fahrzeuge eingesetzt werden, bekommt diese Aufgabe eine größere Bedeutung. Für eine Erfüllung dieser Anforderung, müssen jedoch noch weitere Aspekte betrachtet werden, wie beispielsweise die Kommunikation mehrerer Shared Guide Dogs untereinander und die Sicherheit der Datenübertragung.

6.2 Pfadplanung

Das Ziel der Pfadplanung ist, zwischen einem gegeben Start- und Zielpunkt eine für den Nutzer oder die Nutzerin optimale Strecke zu berechnen. Bei der Auswahl eines Pfadplanungsalgorithmus sind zwei Aspekte zu betrachten. Zum einen, ob die gewählte Kartenrepräsentation vom Algorithmus genutzt werden kann und zum anderen, ob die Berechnung der Strecke an den Nutzer oder die Nutzerin angepasst werden kann. Die Datenverwaltung übersetzt das Open Street Map Datenformat in einen Graphen und speichert diesen als Datei, die im weiteren Verlauf der Arbeit als navigationsfähige Datei bezeichnet wird. Damit kommen alle in Kapitel 2.3 vorgestellten Algorithmen in Frage. Eine einfache Umsetzung, die ein optimales Ergebnis liefert, ist mit Dijkstras Algorithmus möglich. Wird als Kostenfunktion nicht allein die Länge des Weges betrachtet, sondern zusätzlich die Eigenschaften des Weges, ist die Berücksichtigung unterschiedlicher Nutzerprofile möglich. Durch die ungerichtete Suche kann es bei großen Karten jedoch zu längeren Berechnungszeiten kommen, die zu vermeiden sind (Anforderung P-6). Der A* Algorithmus umgeht diesen Nachteil durch die Nutzung der Heuristik als Orientierung zum Ziel. Gleichzeitig gelten die gleichen Aussagen zur Kostenfunktion wie schon bei Dijkstras Algorithmus. Die bidirektionale Suche mit dem A* Algorithmus besitzt dieselben Vorteile wie der A* Algorithmus in der unidirektionalen Suche, bietet tendenziell jedoch noch einen Zeitvorteil bei der Berechnung. Die Wahl für einen Pfadplanungsalgorithmus fällt auf den A* Algorithmus. Er bietet mit der Anpassung der Kostenfunktion eine einfache Möglichkeit zur Optimierung der Pfadplanung. Zudem ist die Erweiterung zu einem bidirektionalen A* Algorithmus jederzeit möglich, sofern dadurch ein Vorteil zu erwarten ist.

6.3 Kartennutzung auf dem Shared Guide Dog

Der letzte Teil des Lösungsansatzes beschäftigt sich mit der Nutzung der Daten auf dem Shared Guide Dog. Der folgende Abschnitt bezieht sich auf Abbildung 6.2, in der die Datenverarbeitung und die Kommunikation der Knoten auf dem Shared Guide Dog dargestellt sind.

Im Zentrum der Steuerung des Shared Guide Dog steht das Navigation 2 Paket, das den BT Navigator Server implementiert, der die Planung, Steuerung und Wiederherstellungsstrategien koordiniert (vgl. Kapitel 4.3). Für die Anfragen an die Pfadplanung und die Bearbeitung der empfangenen Wege wird der *Waypoint Follower* Knoten implementiert. Er bekommt vom Nutzer oder der Nutzerin ein Ziel mitgeteilt und fragt daraufhin die Pfadplanung an. In der Pfadplanung wird der für den Nutzer bzw. die Nutzerin optimale Pfad berechnet und als Menge von Wegpunkten an den Waypoint Follower Knoten gesendet. Der Knoten veröffentlicht die Punkte dann auf dem *NavigateToPose* Topic für das Navigation 2 Paket.

Innerhalb des Navigation 2 Pakets werden die Wegpunkte an den Planner Server übergeben, der einen kontinuierlichen Pfad berechnen, dem der Controller Server folgen kann. Die Berechnung des kontinuierlichen Pfades soll mit einer konfigurierbaren Rate aktualisiert werden. Die Aktualisierung ist notwendig, um den Pfad an kleine Hindernisse anpassen zu können. Steht beispielsweise in der Mitte des Weges ein Poller, plant die Pfadplanung den Weg zwar um das Hindernis herum, durch

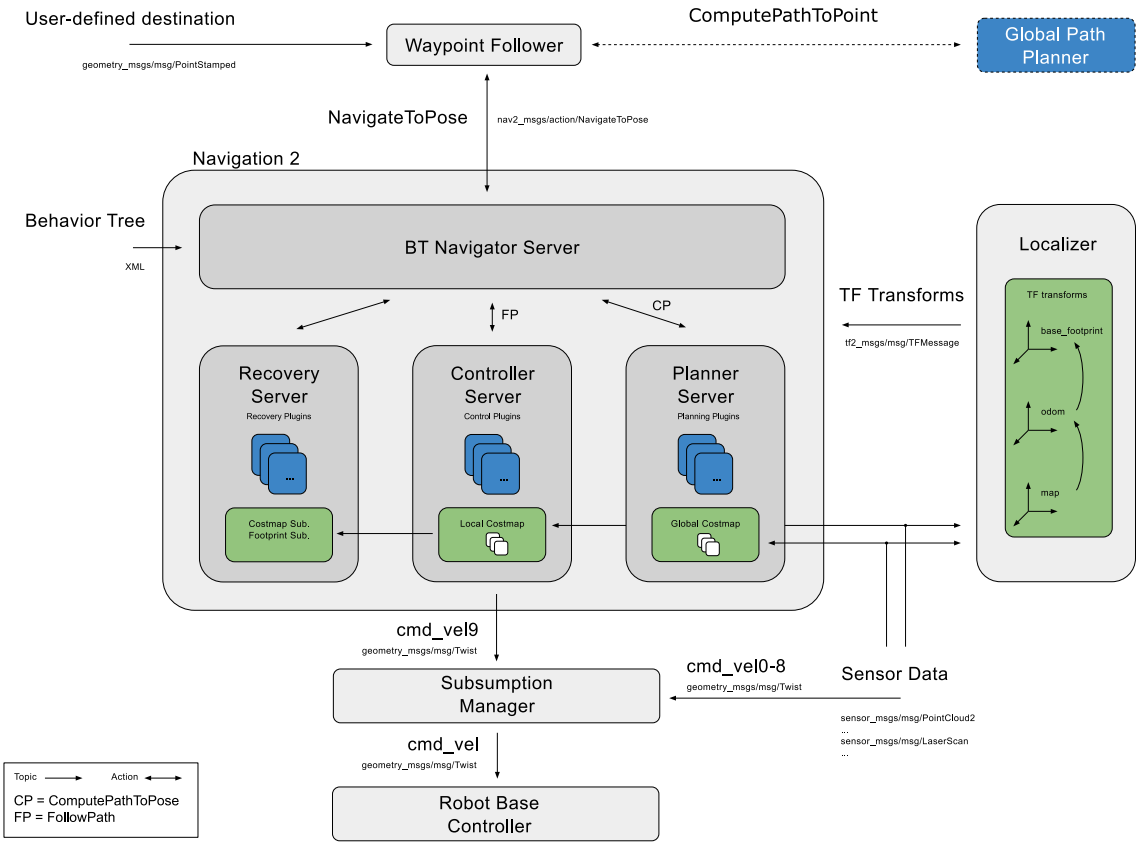

Abbildung 6.2: Integration des Navigation 2 Pakets auf dem Shared Guide Dog (Quelle: [28], eigene Darstellung)

Lokalisierungsfehler kann sich das Hindernis aber trotzdem im Weg des Shared Guide Dog befinden. Diese Situationen sollen durch eine lokale Umgehung gelöst werden, was jedoch nicht Teil dieser Arbeit ist.

Zusätzlich zu den Wegpunkten benötigt Navigation 2 die Koordinatentransformation zwischen dem *map* und *odom* und dem *odom* und *base_link* System. Diese Transformationen werden vom Localizer bereitgestellt. Da der Fokus dieser Arbeit nicht auf der Implementierung der Sensordatenfusion und Berechnung einer Position liegt, wird diese Aufgabe vom *robot_localization* Paket für ROS2 gelöst. Die Implementierung und Konfiguration wird in Kapitel 8.2 behandelt.

Die Ausgabe des Navigation 2 Pakets ist eine Nachricht mit der Sollgeschwindigkeit in x-Richtung und der Soll-Drehrate um die z-Achse. Bevor diese Nachricht an den Motorcontroller geschickt wird, muss sie vom *Subsumption Manager* verarbeitet werden. Der Subsumption Manager ist ein Knoten, der eine Subsumption-Architektur implementiert und als Sicherungssystem genutzt wird.

7 Umsetzung der Navigationslösung

Im Laufe dieses Kapitels werden die Datenverwaltung und die Pfadplanung für die Navigation auf dem Shared Guide Dog entwickelt und implementiert.

7.1 Herunterladen der Kartendaten von OSM

Für das Herunterladen der Kartendaten von Open Street Map gibt es mehrere unterschiedliche Möglichkeiten. Da im Rahmen dieser Arbeit nur lesende und keine schreibenden Operationen durchgeführt werden, sollen die beiden Möglichkeiten *Planet.osm/Extracts* und die *Overpass API* vorgestellt werden. Ziel des Herunterladens ist eine Karte des Testgebiets im Open Street Map (OSM) Kartenformat (vgl. Kapitel 2.2.1). Die Gebiete außerhalb des Testgebiets sollen nicht in der Karte enthalten sein, um die räumliche Eingrenzung sicherzustellen. Mit Blick auf die Zukunft ist diese Auswahl nicht endgültig. Werden die Shared Guide Dogs in Gebieten eingesetzt, wo die Kartierung nicht so gut ist, können sie auch zum Kartieren verwendet werden. Dann muss eine Überprüfung der gewählten Möglichkeit zum Herunterladen der Karte stattfinden, damit auch schreibende Zugriffe auf die Karte möglich sind.

Als Planet.osm wird der vollständige Datensatz des gesamten Planeten verstanden, in dem alle Daten enthalten sind. Extracts sind hingegen kleinere Ausschnitte aus Planet.osm. Das können beispielsweise einzelne Kontinente, Länder oder noch kleinere Ausschnitte sein. Für Deutschland sind die Bundesländer als Extracts verfügbar, wobei auch diese teilweise noch unterteilt werden. Für die Stadt Hamburg steht keine weitere Unterteilung zur Verfügung. Es besteht somit nur die Möglichkeit das gesamte Hamburger Stadtgebiet herunterzuladen. Um das Ziel zu erfüllen, dass nur das Testgebiet in der Karte enthalten sein soll (Anforderung P-10), müssen entweder viele Daten gelöscht oder die außerhalb liegenden Gebiete gesondert während des Preprocessings berücksichtigt werden. Extracts haben den weiteren Nachteil, dass sie nicht im OSM Kartenformat heruntergeladen werden können, sondern nur als pbf-Archiv und vor der Bearbeitung entpackt werden müssen. [49]

Die Overpass API ist die zweite Möglichkeit die Daten von Open Street Map herunterzuladen. Es handelt sich dabei um eine Application Programming Interface (API), die speziell für das Abrufen von Daten aus OSM entwickelt wurde. Für die Overpass API sind zwei Abfragesprachen verfügbar. Zum einen kann die Anfrage per XML geschrieben werden, zum anderen über die eigens entwickelte Overpass Query Language (QL). Zwei Vorteile der Overpass API sind, dass genau angegeben werden kann, welches Gebiet exportiert werden soll und die Daten schon gefiltert werden können. Insbesondere das Filtern bietet viele Möglichkeiten die Dateigröße für den Download zu reduzieren. [48]

Die Wahl für das Herunterladen der Kartendaten fällt auf die Overpass API. Sie bietet die Möglichkeit nur einen ausgewählten Bereich herunterzuladen und diesen bei Bedarf auch noch zu filtern. Da das Testgebiet im Vergleich zum Hamburger Stadtgebiet sehr klein ist, würden bei der Nutzung der Extracts viele Daten heruntergeladen werden, die nicht benötigt werden.

Die Abfrage der Daten ist im folgenden Codeblock dargestellt. Mit dem Element *bbox-query* wird ein Gebiet definiert, aus dem alle Knoten und Wege geladen werden. Anschließend folgt eine Vereinigung (*union*) einer Abfrage (*query*), bei der alle Wege mit dem Attribut *highway*, die jedoch nicht das Attribut *highway=secondary* besitzen, mit einer rekursiven Suche nach den zugehörigen Knoten verbunden wird.

```
<osm-script>
  <bbox-query s="53.5533" w="10.019" n="53.558" e="10.025"/>
  <recurse type="node-way"/>
  <union>
    <query type="way">
      <item/>
      <has-kv k="highway" regv="."/>
      <has-kv k="highway" modv="not" v="secondary"/>
    </query>
    <recurse type="way-node"/>
  </union>
  <print mode="meta"/>
</osm-script>
```

7.2 Fehler in den Daten

Der zweite Schritt der Datenverarbeitung ist die Fehlerüberprüfung. Um eine Aussage über die Qualität der Daten und die zu erwartetenden Fehler treffen zu können, wird zunächst eine Analyse der Kartendaten des Lohmühlenparks durchgeführt.

Abbildung 7.1 zeigt den Vergleich der Open Street Map Daten mit den Kartendaten des Landesbetriebs Geoinformation und Vermessung (LGV) der Stadt Hamburg. Die Karten können kostenfrei aus dem Geoportal der Stadt Hamburg (vgl. [24]) bezogen werden. Auf der linken Seite wird ein Stadtplan mit dem Maßstab 1:500 als Hintergrund verwendet, auf der rechten Seite ist es ein Luftbild mit einer Auflösung von 20 cm. Überlagert werden die Bilder von den OSM Daten, die in Rot dargestellt sind. Die Ausrichtung erfolgt anhand der Eckpunkte des Basketballplatzes in der Mitte der Bilder. Anhand der beiden Punkte wird die Skalierung und Drehung durchgeführt, denn obwohl beide Karten nach Norden ausgerichtet sein sollten, sind die Karten zueinander um ungefähr 1,3 ° verdreht. Der Vergleich der dargestellten Bilder zeigt, dass es nur sehr geringe Abweichungen zwischen den Kartendaten gibt. Diese können auch nicht zweifelsfrei auf die Daten zurückgeführt werden, sondern können auch bei der Skalierung und Drehung entstanden sein.

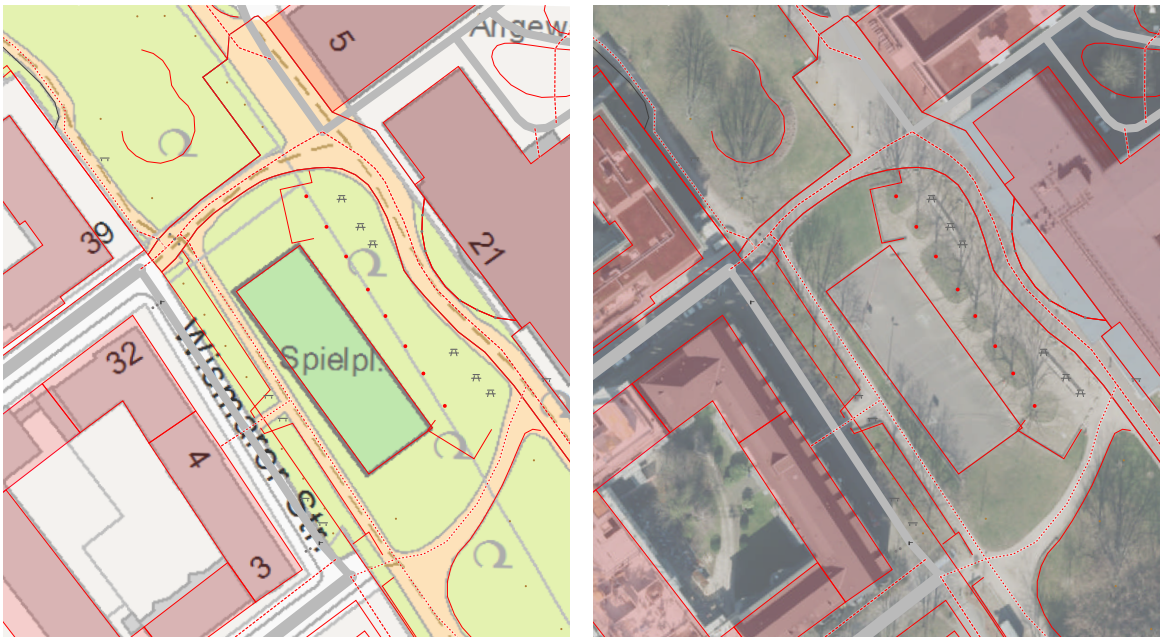

Abbildung 7.1: Vergleich der Karten des Katasteramts mit den OSM Kartendaten (Quelle: Hintergrund: [24], [42])

Während sich der vorige Abschnitt besonders auf die Positionsgenauigkeit der Open Street Map Daten konzentriert hat, soll nun das Augenmerk auf die Vollständigkeit von Attributen und Objekten gelegt werden. In [27] wurden die OSM Karten mit Karten des kommerziellen Anbieters Navteq (inzwischen HERE [17]) verglichen. Betrachtet wurden nur Straßen und Wege. Insgesamt fehlen in den Open Street Map Daten bei ungefähr 6 % der Wege in besiedelten Gebieten die Straßennamen. Fehlende Attribute sind insbesondere bei kleineren Wegen aufgetreten. Fußwege sind zu ungefähr 53 % vollständig eingetragen. Die Betrachtung der Objektvollständigkeit liefert für Hamburg das Ergebnis, dass 96,18 % aller Objekte mit den Navteq Daten übereinstimmen. [27]

Bei den vorgestellten Daten ist zu beachten, dass der Vergleich bereits im Jahr 2009 durchgeführt und keine Aussage über die Vollständigkeit oder Korrektheit der Navteq Daten getroffen wurde. Die Open Street Map Karte bestand zu der Zeit aus etwas mehr als 500 Mio. Knoten, während es heute bereits über 7 Mrd. sind [53]. Es muss also davon ausgegangen werden, dass sich die Abdeckung der Karte deutlich verbessert hat. Insbesondere in dicht besiedelten Gebieten kann von einer nahezu vollständigen Abdeckung aller Wege und Straßen ausgegangen werden. Schwieriger zu prüfen sind hingegen die Attribute. Hier muss davon ausgegangen werden, dass einige Attribute fehlerhaft sein können oder sogar ganz fehlen.

Als Fazit zum Thema Datenqualität und -genauigkeit sollen die in DIN EN ISO 19157 (vgl. [7]) definierten Datenqualitätselemente betrachtet werden. Die Ergebnisse der Bewertung nach DIN EN ISO 19157 sind in Tabelle 7.1 zusammengefasst. Zu beachten ist, dass diese Bewertung nur für den Lohmühlenpark gilt und für weitere Einsatzgebiete des Shared Guide Dog erneut durchgeführt werden muss.

Das erste Element bewertet die Vollständigkeit der Daten. Der Vergleich der Luftbilder mit den OSM Daten hat gezeigt, dass sich nordöstlich des Spielplatzes Bäume, Bänke und eine Steinreihe als Abgrenzung zur Straße befinden. Die Bäume und Bänke sind in Open Street Map sehr genau und vollständig eingezeichnet. Die Steinreihe fehlt jedoch vollständig. Bis zu einer vollständigen Darstellung fehlen demnach noch einige Objekte.

Das zweite Element ist die logische Konsistenz. Hier wird zusammengefasst, ob die Regeln für die Datenstruktur eingehalten werden oder nicht. Die Open Street Map Datenstruktur lässt dem Nutzer sehr viele Freiheiten bezüglich der Daten, die eingetragen werden können. Häufig vorkommende Objekte sollten trotzdem mit den gleichen Attributen versehen werden. Das ist bei Open Street Map nicht immer der Fall.

Als drittes Element wird die Positionsgenauigkeit aufgeführt. Es konnte bereits im vorigen Abschnitt gezeigt werden, dass diese bei Open Street Map Daten sehr hoch ist.

Das vierte Element ist die thematische Genauigkeit, die die Korrektheit von Attributen und Features betrachtet. Teilweise fehlen Attribute in den OSM Daten oder es sind fehlerhafte Attribute eingetragen, was jedoch seltener vorkommt. Insgesamt kann dieses Element als gut bewertet werden.

Die zeitliche Genauigkeit ist das fünfte betrachtete Element. Es wird die Aktualität der Kartendaten bewertet. Open Street Map bezeichnet die eigene Karte als oftmals aktueller und qualitativ hochwertiger in Bezug auf neue und geänderte Wege verglichen mit kommerziellen Kartenanbietern [50]. Eine Überprüfung dieser Aussage ist jedoch nicht möglich.

Das letzte Element betrachtet die Benutzbarkeit der Daten in anderen Projekten. Für diese Bewertung werden alle vorher betrachteten Elemente im Hinblick auf die Verwendbarkeit betrachtet. Für das Projekt Shared Guide Dog ergibt sich eine sehr gute Verwendbarkeit. Die wichtigsten Punkte sind die Positionsgenauigkeit und die Vollständigkeit der Attribute. Auch wenn die Vollständigkeit der Attribute nicht immer mit sehr gut bewertet werden kann, ist sie derzeit ausreichend für das Projekt.

Tabelle 7.1: Bewertung der OSM Daten anhand der DIN EN ISO 19157

Element	Beschreibung	Bewertung
Vollständigkeit	Vorhandensein von Features, Attributen und Beziehungen	gut
Logische Konsistenz	Regeln des Datensets wurden eingehalten	gut
Positionsgenauigkeit	Abweichung von der realen geographischen Position	sehr gut
Thematische Genauigkeit	Korrektheit von Attributen und Klassifikation von Features	gut
Zeitliche Genauigkeit	Darstellung des aktuellen Stands	sehr gut
Benutzbarkeit	Verwendbarkeit für das eigene Projekt	sehr gut

Die Open Street Map Kartendaten können als sehr genau bewertet werden, trotzdem muss eine Überprüfung der Daten vorgenommen stattfinden. Dabei wird zwischen Attributfehlern und Kartierungsfehlern unterschieden. Kartierungsfehler sind beispielsweise doppelt gesetzte Knoten,

nicht verbundene oder fehlende Wege und falsch gesetzte Knoten. Zu den Attributfehler gehören Fehler, bei denen die Werte der Attribute falsch oder nicht gesetzt sind.

Im Open Street Map Projekt sind Qualitätssicherungstools entstanden mit denen sowohl eine automatische Überprüfung der Karte durchgeführt wie auch manuell Fehler eingetragen werden können. In den Prozess der Qualitätssicherung für das Projekt Shared Guide Dog soll der *JOSM Validator* zur Anwendung kommen. Dabei handelt es sich um eine Kernfunktion des JOSM Editors, mit dem sich die aktuelle Karte überprüfen lässt. Mit dem JOSM Validator lassen sich hauptsächlich Kartierungsfehler erkennen, während falsch gesetzte Attribute nicht erkannt werden. Dieser Prozess muss manuell durchgeführt werden. In Abbildung 7.2 sind beispielhaft Fehler dargestellt, die am Berliner Tor 21 gefunden wurden. Ein automatisierter Prozess zum Erkennen der Fehler ist nicht möglich, da die Ortskenntnis erforderlich ist. [50]

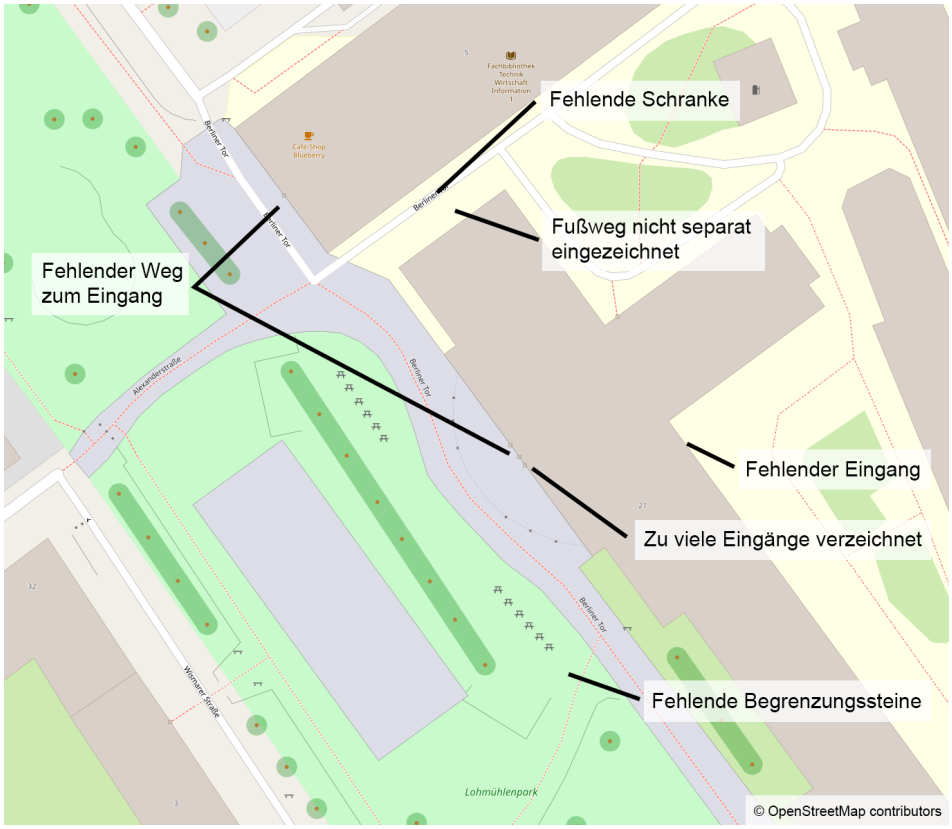

Abbildung 7.2: Mögliche Fehler in den Open Street Map Kartendaten (Quelle: [42], eigene Darstellung)

7.3 Augmentieren der Daten

Bevor die Daten für die Navigation genutzt werden können, müssen sie augmentiert werden. Für den Shared Guide Dog können einige Attribute, die in den OSM-Daten vorhanden sind, weiter genutzt werden, andere Attribute müssen neu hinzugefügt werden. Ziel der Augmentation ist es,

alle für den Navigationsalgorithmus benötigten Daten in der Karte zu hinterlegen. Dabei wird zwischen Attributen, die sich auf den ganzen Weg beziehen und mit Wegen verknüpft werden, und Attributen, die sich nur auf einen einzigen Knoten beziehen und mit diesem Knoten verknüpft werden, unterschieden. Eine Übersicht über die Attribute ist in Tabelle 7.2 gegeben.

Für den Shared Guide Dog werden fünf Attribute benötigt, die mit einem Weg verknüpft werden. Das ist die Oberflächenbeschaffenheit (surface), die Art des Weges (highway), die Breite des Weges (width), die Steigung (slope) und sonstige Gefahren (hazard). Die Oberflächenbeschaffenheit und Art des Weges sind als Attribute bereits in den Open Street Map Daten vorhanden. Bei diesen Attributen muss jedoch kontrolliert werden, ob die Attributwerte in der Liste der erlaubten Werte sind, andernfalls müssen sie angepasst werden (vgl. Tabelle 7.2). Die Attribute Breite, Steigung und sonstige Gefahren müssen hinzugefügt werden. Zu den sonstigen Gefahren zählen beispielsweise Ausfahrten, die häufig von Autofahrern genutzt werden. Der Weg wird dann weiterhin als Fußweg gekennzeichnet, durch die Angabe einer Gefahrenzone wird er jedoch teurer für den Pfadplanungsalgorithmus.

Von den Attributen für die Knoten ist nur das Hindernis (OSM Attribut *barrier*) in den Open Street Map Daten vorhanden. Ein Attribut für Kantsteine oder Absätze gibt es in Open Street Map nicht und muss daher manuell hinzugefügt werden. Die Angaben zur Adresse sind in Open Street Map vorhanden, werden jedoch in Relations gespeichert und nicht in den Knoten. Diese Angaben müssen noch zu den Knoten kopiert werden.

Tabelle 7.2: Attribute für die Augmentation

OSM-Tag	Beschreibung	mögliche Werte	Weg/Knoten
surface	Oberflächenbeschaffenheit	asphalt, paving_stones, compacted, gravel, cobblestone, sand	Weg
highway	Art des Weges	footway, service, living_street, residential, steps	
width	Breite des Weges	Zahl in m	
slope	Steigung / Gefälle	Zahl in Grad	
hazard	sonstige Gefahren	Zahl	
barrier	Hindernis an der Position	bollard, lift_gate, tree	Knoten
curb	Kantstein an der Position	Höhe in m	
addr:street	Straße der Adresse	Zeichenkette	Knoten
addr:housenumber	Nummer des Hauses	Zahl	
addr:housename	Bezeichnung des Gebäudes	Zeichen	

7.4 Datenausgabe

Damit ein Pfadplanungsalgorithmus effizient einen Weg berechnen kann, sollen die augmentierten Kartendaten in ein navigationsfähiges Format gebracht werden (vgl. Kapitel 6.2). Merkmal eines

solchen Formates ist die direkte Verknüpfung von Knoten, also dass aus jedem Knoten direkt die Nachbarknoten und die Verbindungen dorthin ersichtlich sind. Die OSM Kartenrepräsentation stellt kein navigationsfähiges Format dar, da die Knoten nicht direkt miteinander verknüpft sind, sondern separat über Wege (siehe Abschnitt 2.2.1).

Der Aufbau der erstellten Navigationsdatei orientiert sich am Aufbau der OSM Rohdaten. Es wird eine XML-Datenstruktur verwendet, die zur eindeutigen Identifizierung die Endung *.nav* bekommt. Das XML-Wurzelelement ist *nodelist* mit den XML-Attributen *name* und *version*. Der Name ist ein Bezeichner für die Navigationsdaten und über die Version können Änderungen am Datensatz kenntlich gemacht werden. Unter dem Wurzelelement folgen die einzelnen Knotenelemente, bezeichnet mit *node*. Sie besitzen die Attribute *id*, *lat* und *lon* für die Knoten-ID und die Positionsangabe mit Breiten- und Längengrad. Die Knoten können drei unterschiedliche Elemente beinhalten. Das sind Hindernisse (barrier), Kantsteine (curb) und Referenzen zu den Nachbarknoten (nd). Hindernisse können beispielsweise Poller und Schranken sein, die umgangen werden müssen. Kantsteine werden mit ihrer Höhe in Metern angegeben, damit der Pfadplanungsalgorithmus diese berücksichtigen kann. Jeder Knoten muss eine mindestens eine Referenz zu einem anderen Knoten besitzen. In diesem Element werden die ID des Nachbarknotens (ref) und der Weg dorthin über verschiedene Attribute gespeichert. Eine ausführliche Erklärung der Attribute findet in Abschnitt 7.6 statt.

Der folgende Codeausschnitt zeigt beispielhaft den Aufbau der Navigationsdatei.

```
<nodelist name='lohmuehlenpark' version='0.1'>
  <node id='8596122144' lat='53.5571622' lon='10.0207811' >
    <curb>0.1</curb>
    <nd ref='8596122143'>
      <highway>footway</highway>
      <surface>paving_stones</surface>
      <width>3</width>
    </nd>
    <nd ref='-105937'>
      <highway>footway</highway>
      <surface>paving_stones</surface>
      <width>3</width>
    </nd>
  </node>
  <node id='8596109047' lat='53.5570256' lon='10.0207442' >
    <barrier>bollard</barrier>
    <nd ref='8596109046'>
      <highway>residential</highway>
      <surface>asphalt</surface>
      <width>5</width>
    </nd>
  </node>
</nodelist>
```

Damit der Nutzer komfortabel ein Ziel eingeben kann, soll die Möglichkeit geschaffen werden, dass eine Adresse als Ziel angegeben werden kann. Da der Pfadplanungsalgorithmus jedoch nur mit Längen- und Breitengraden rechnen kann, muss eine Zuordnung der Adresse zu den Koordinaten stattfinden. Dafür wird neben der Navigationsdatei eine Adressliste erstellt, die genau diese Aufgabe erfüllt. Der folgende Codeblock zeigt einen Ausschnitt aus der Adressliste für den Lohmühlenpark. Die Zuordnung der Knoten aus OSM erfolgt über die Knoten-ID.

```
1 # Address list
2 Stiftstra ße 69 : 257693056
3 Berliner Tor 21 : 313339524
4 Berliner Tor 5 : 4623635466
5 Wismarer Straße 3 : 5139452006
```

7.5 Vernetzung der Knoten

Wie bereits in 6.3 erläutert, soll im Rahmen dieser Arbeit auf einen lokalen Controller und die lokale Hindernisumfahrung mit diesem verzichtet werden. Stattdessen werden Hindernisse auf Basis der Kartendaten umfahren. Dafür ist es jedoch nicht ausreichend, dass jeder Weg nur durch eine Verbindung zwischen Knoten repräsentiert wird. Stattdessen sollen mehrere parallele Wege nebeneinander über die Breite des Weges erstellt werden. Anschließend müssen die neuen Knoten mit den Originalknoten vernetzt werden. Damit ein Wechsel des Weges in kurzen Abständen möglich ist, werden lange Strecken interpoliert und neue Knoten eingefügt. Zur Berechnung der Anzahl neuer Knoten wird die Distanz zwischen den beiden Knoten durch den Schwellenwert für die Interpolation a geteilt (Gleichung 7.1).

$$n = \frac{\sqrt{(\Delta b)^2 + (\frac{\Delta l}{\cos b})^2}}{a} \tag{7.1}$$

Ist die Anzahl der einzufügenden Knoten bekannt, können aus der Distanz und der Anzahl der Knoten die Inkremente für die Längen- und Breitengrade berechnet werden (Gleichungen 7.2 und 7.3).

$$I_L = \frac{\Delta l}{n} \tag{7.2}$$

$$I_B = \frac{\Delta b}{n} \tag{7.3}$$

Die Berechnung der neuen Knoten für parallele Wege erfolgt auf Basis der Wege zu den Nachbarknoten. Für die vollständige Bestimmung der Position eines neuen Knotens wird eine Richtung und ein Abstand zum Originalknoten benötigt. Für die Berechnung der Richtung wird die Winkelhalbierende zwischen den Wegen verwendet. Im ersten Schritt werden die Winkel der Wege nach Gleichung

7.4 berechnet. Der Faktor $\cos l$ ist notwendig, um die Stauchung des Abstands der Längengrade zu den Polen zu berücksichtigen (vgl. Kapitel 3.2.1). Anschließend werden die Winkel der Größe nach sortiert. In Gleichung 7.4 wird die Funktion $\arctan 2$ genutzt. Es handelt sich dabei um eine in der Programmierung verbreitete Funktion, die den Arkustangens in einem Wertebereich von $-\pi$ bis $+\pi$ ausgeben kann. Dafür müssen der x- und y-Achsenabschnitt als einzelne Parameter angegeben werden. Als x-Achsenabschnitt wird die Differenz der Breitengrade korrigiert mit dem Kosinus und als y-Achsenabschnitt die Differenz der Längengrade verwendet. [4]

$$\phi_n = \arctan 2(b - b_n \cdot \cos(l), l - l_n) \tag{7.4}$$

Vor der Berechnung der Winkelhalbierenden wird der kleinste Winkel mit 2π addiert und zur Liste mit den Winkeln hinzugefügt. Damit wird ein separater Berechnungsschritt für den Winkel zwischen dem kleinsten und größten Winkel vermieden. Für die Berechnung der Winkelhalbierenden werden alle Winkel in der Liste durchlaufen und nach Gleichung 7.5 die Winkelhalbierenden berechnet.

$$\phi_{m,n} = \frac{\phi_m + \phi_n}{2} \tag{7.5}$$

Die Berechnung der Distanz zwischen dem Originalknoten und einem neu hinzugefügten Weg erfolgt nach Gleichung 7.6 mit der Breite b der angrenzenden Wege. Der Divisor 6 ergibt sich aus der Überlegung, dass die Wege nur über 2/3 der Wegbreite verteilt werden sollen. So bleibt auf beiden Seiten des Weges noch Platz für den Shared Guide Dog.

$$a = \frac{b_1 + b_2}{6} \tag{7.6}$$

Nach der Erstellung neuer Knoten folgt die Vernetzung der Knoten mit den Originalknoten. Wenn an einem Knoten mehr als zwei neue Knoten erstellt werden, werden diese auch untereinander vernetzt. Das ist beispielsweise bei Kreuzungen der Fall. Der folgende Codeblock zeigt den Pseudocode zur Erstellung neuer Knoten und Vernetzung der Knoten untereinander. Die Vernetzung neuer Knoten mit alten Knoten erfolgt in einer separaten Funktion.

```
1  declare node as Node          // current node
2
3  declare neighborlist as Set with Nodes     // list containing all neighbor nodes sorted by angle to current node
4  declare childnodes as List with Nodes      // new created nodes
5
6  function mesh_nodes()
7      if neighborlist.size() < 2
8          return
9      else
10         // add smallest angle + 2*PI to list
11         neighborlist.add(neighborlist[0].angle + 2*PI)
12     end
13
14     last_neighbor = null
15     new_id = node.id
16     foreach neighbor in neighborlist
```

```
17          if last_neighbor != null
18              winkel = (last_neighbor.angle + neighbor.angle) / 2
19
20              // get ways to neighbor and last_neighbor
21              way1 = node.get_way(neighbor)
22              way2 = node.get_way(last_neighbor)
23
24              width1 = way1.get_width()
25              width2 = way2.get_width()
26
27              // calculate distance for new point
28              distance = (width1 + width2) / 6
29
30              lat = node.lat + sin(winkel) * 1/111319 * distance
31              lon = node.lon + cos(winkel) * 1/(111319*cos(node.lat/180*PI)) * distance
32              new_id++
33              Node child(new_id, lat, lon)
34
35              // mesh created node with last node
36              if neighborlist.size() > 3 && childnodes.size() > 0
37                  last_node = childnodes.back()
38                  child.add_way_to(last_node)
39                  last_node.add_way_to(child)
40              end
41
42              childnodes.add(child)
43          end
44
45          last_neighbor = neighbor
46      end
47
48      // add way from first to last child node
49      if childnodes.size() > 2
50          Node n1 = childnodes.begin();
51          Node n2 = childnodes.back();
52          n1.add_way_to(n2)
53          n2.add_way_to(n1)
54      end
```

7.6 Navigation mit A*

Für die Pfadplanung wird der A*-Algorithmus ausgewählt, da er als Algorithmus einfach zu implementieren ist und gleichzeitig viele Möglichkeiten für die Optimierung der Wege bietet. Bei der Kostenberechnung des A*-Algorithmus werden die Kosten bis zum aktuellen Knoten $g(x_n)$, die Kosten vom aktuellen zum nächsten Knoten $c(x_n, x_{n+1})$ und die geschätzten Kosten vom nächsten Knoten zum Ziel $h(x_{n+1}, x_d)$, die Heuristik, verwendet (Gleichung 7.7).

$$f(x_{n+1}) = g(x_n) + c(x_n, x_{n+1}) + h(x_{n+1}, x_d) \tag{7.7}$$

Die Funktion zur Berechnung der Heuristik darf die Kosten zum Zielknoten nicht überschätzen, sodass es sich anbietet die euklidische Distanz als Funktion zu nutzen. Es wird dafür Gleichung 3.3 aus Kapitel 3.2.2 verwendet. Die Kosten vom aktuellen zum nächsten Knoten setzen sich aus der Distanz zwischen den beiden Knoten und der Wegfunktion w zusammen (Gleichung 7.9). Die Wegfunktion wird im nächsten Abschnitt entwickelt.

$$g(x_n) = g(x_{-1}) + c(x_{-1}) \tag{7.8}$$

$$c(x_n, x_{n+1}) = d(x_n, x_{n+1}) \cdot w(x_n, x_{n+1}) \tag{7.9}$$

$$h(x_{n+1}, x_d) = d(x_{n+1}, x_d) \tag{7.10}$$

Das Ziel der Wegfunktion ist die Berücksichtigung unterschiedlicher Straßenverhältnisse, Straßenarten, Steigungen oder Gefälle und sonstiger Hindernisse. Zusätzlich sollen die Fähigkeiten und Präferenzen der Nutzer und Nutzerinnen berücksichtigt werden, zum Beispiel, dass eine Straße mit Kopfsteinpflaster nicht befahren werden soll.

Der A* Algorithmus bietet durch die Kostenberechnung eine Möglichkeit, Strecken in Abhängigkeit der Eigenschaften mit unterschiedlichen Kosten zu versehen. Die Umsetzung in der Routenplanung erfolgt mit einer XML-Datei, in der für verschiedene Benutzer individuelle Kosten für die unterschiedlichen Straßeneigenschaften hinterlegt werden können. Standardmäßig ist ein Default-User vorgesehen, der einen durchschnittlichen Nutzer repräsentieren soll. Die Kosten sind so gewählt, dass vorzugsweise auf Fußwegen mit festen Untergründen gefahren wird. Nur wenn dieser Weg einen großen Umweg bedeuten würde, wird auf unbefestigte Wege oder verkehrsberuhigte Straßen ausgewichen.

Beispielhaft ist der Aufbau der Benutzerdatei im folgenden Codeblock dargestellt. Das XML-Wurzelelement ist *<users>*. Darunter können beliebig viele Nutzer definiert werden. Jeder neue Nutzer wird mit dem Element *<user>* eingeleitet und muss das XML-Attribut *name* zur eindeutigen Identifizierung besitzen. Es können keine zwei Nutzer mit dem gleichen Namen angelegt werden. Innerhalb des user-Elements werden die OSM Attribute aufgelistet und mit Kostenfaktoren für den jeweiligen Nutzer versehen. Im gezeigten Beispiel für die Oberfläche des Weges (surface) werden die möglichen Werte *asphalt*, *paving_stones*, *compacted*, *gravel*, *cobblestone* und *sand* definiert. Innerhalb der einzelnen Elemente werden die Kostenfaktoren angegeben. Das Element *default* wird hinzugefügt, um Wege, deren Attribut nicht definiert wurde, mit Kosten zu versehen. Besitzt das OSM Attribut statt einer Bezeichnung einen Zahlenwert, werden keine möglichen Werte angegeben, sondern direkt der Kostenfaktor, wie beispielsweise im Element *<hazard>* gezeigt.

Nur für den Default-Nutzer müssen alle Attribute angegeben werden. Für die Definition weiterer Nutzer ist es ausreichend nur die Werte anzugeben, die geändert werden sollen. Im Beispiel werden die Kostenfaktoren für die Oberfläche der Wege reduziert. Alle anderen Faktoren, die nicht angegeben werden, werden vom Default-User übernommen.

```
<users>
    <user name='default'>
        <surface>
            <asphalt>1</asphalt>
            <paving_stones>1.2</paving_stones>
            <compacted>1.5</compacted>
            <gravel>2.0</gravel>
            <cobblestone>4.0</cobblestone>
            <sand>5.0</sand>
        </surface>
        <way>
            <footway>1.0</footway>
            <service>1.5</service>
            <living_street>2.0</living_street>
            <residential>5.0</residential>
            <steps>1e6</steps>
        </way>
        <slope>1.0</slope>
        <hazard>1.0</hazard>
        <default>10.0</default>
    </user>
    <user name='sand_kein_problem'>
        <surface>
            <asphalt>1</asphalt>
            <paving_stones>1</paving_stones>
            <compacted>1</compacted>
            <gravel>1.5</gravel>
            <cobblestone>1.8</cobblestone>
            <sand>2.0</sand>
        </surface>
    </user>
</users>
```

Die Wegfunktion setzt sich aus den Kostenfaktoren der Nutzer und einem richtungsabhängigen Term zusammen. Die Kostenfaktoren werden anhand der OSM Attribute des Weges ausgewählt. Ein Weg mit dem OSM Attribut *<surface>compacted</surface>* erhält für den Default-Nutzer den Kostenfaktor 1,5. Dieses Vorgehen wird für alle Attribute des Weges wiederholt und die Kostenfaktoren addiert (Gleichung 7.11).

$$u(x, x_{n+1}) = \sum_{n=0}^{4} c(n) \tag{7.11}$$

Zusätzlich zu den Attributen wird die Richtung zum nächsten Knoten berücksichtigt. Damit wird Anforderung P-4 erfüllt, dass der Shared Guide Dog nach Möglichkeit rechts auf dem Weg fahren soll. Die Umsetzung erfolgt mit einer richtungsabhängigen Funktion. Die Richtung zum nächsten Knoten wird relativ zur Richtung zum nächsten Originalknoten berechnet. Dazu wird eine Koordinatentransformation durchgeführt, sodass die Strecke zum nächsten Originalknoten auf der x-Achse liegt. Anschließend kann der Winkel zwischen dem Weg zum nächsten Knoten und dem Weg zum Originalknoten bestimmt werden. Ein Winkel mit einem Wert >0 bedeutet, dass der Weg nach links führt, ein Winkel <0 bedeutet, dass der Weg nach rechts führt.

Die Funktion muss die Eigenschaft aufweisen, dass ein Weg, der auf der rechten Seite des Weges verläuft, günstiger ist als ein Weg auf der linken Seite. Zudem darf der Funktionswert nicht null oder kleiner als null werden und die Funktion soll das Abbiegen nach links, also die Richtungsänderung um mehr als 30 ° nicht bestrafen. Ausgewählt wird die in Gleichung 7.12 gezeigte Funktion, die eine e-Funktion und eine Tangens hyperbolicus Funktion kombiniert. Mit der e-Funktion lassen sich kleine Winkel nach rechts günstiger und kleine Winkel nach links teurer gestalten. Werden die Winkel größer, nähert sich die Funktion null an. Um eine verwendbare Kostenfunktion zu erhalten, muss eine eins addiert werden. Die Tangens hyperbolicus Funktion sorgt für die Erhöhung der Kosten bei größeren Winkeln. Die Funktion muss eingefügt werden, da der Algorithmus ansonsten versucht, häufig kleine Winkel nach rechts zu fahren, bis ein großer Winkel nach links gefahren werden kann. Das ist günstiger als einen kleinen Winkel nach links zu fahren. Mit der Tangens-Hyperbolicus Funktion lässt sich dieses Verhalten unterdrücken.

$$c_r(x) = a \cdot -e^{-\left((b\varphi)^2\right)} \cdot \varphi + 1 + c \cdot \tanh(5\varphi) \tag{7.12}$$

Mit den beiden Faktoren a, b und c ist eine Anpassung der Funktion möglich. Der Faktor a steuert die Höhe der e-Funktion. Mit dem Faktor b lässt sich die Breite anpassen. Dabei muss beachtet werden, dass dieser Faktor auch die Höhe beeinflusst. Der Faktor c ist für die bleibenden Kosten bei großen Winkeln zuständig. Wird er gleich null gesetzt, gibt es keine bleibenden Kosten. Je größer der Wert wird, desto größer werden die bleibenden Kosten. In Abbildung 7.3 sind drei verschiedene Varianten der Funktion mit unterschiedlichen Faktoren dargestellt. Die Variante mit $a = 2$, $b = 3$ und $c = 0,1$ hat sich während der Entwicklung als gute Kombination für die Kostenberechnung herausgestellt.

7.7 Schnittstelle zum Shared Guide Dog 4.0

Bevor die Schnittstelle zum Shared Guide Dog implementiert werden kann, muss entschieden werden, ob das System zentral oder dezentral gesteuert werden soll. Eine zentrale Steuerung hat die Vorteile, dass jeder Shared Guide Dog auf die gleiche Karte zugreift und dass das Verarbeiten der Kartendaten und die Pfadplanung auf einem leistungsfähigen Computer durchgeführt werden kann. Für die entwickelte Lösung ist dieser Aspekt noch nicht entscheidend. Je mehr Wege und Einschränkungen berücksichtigt werden sollen, desto größer wird jedoch der Berechnungsaufwand und damit auch die

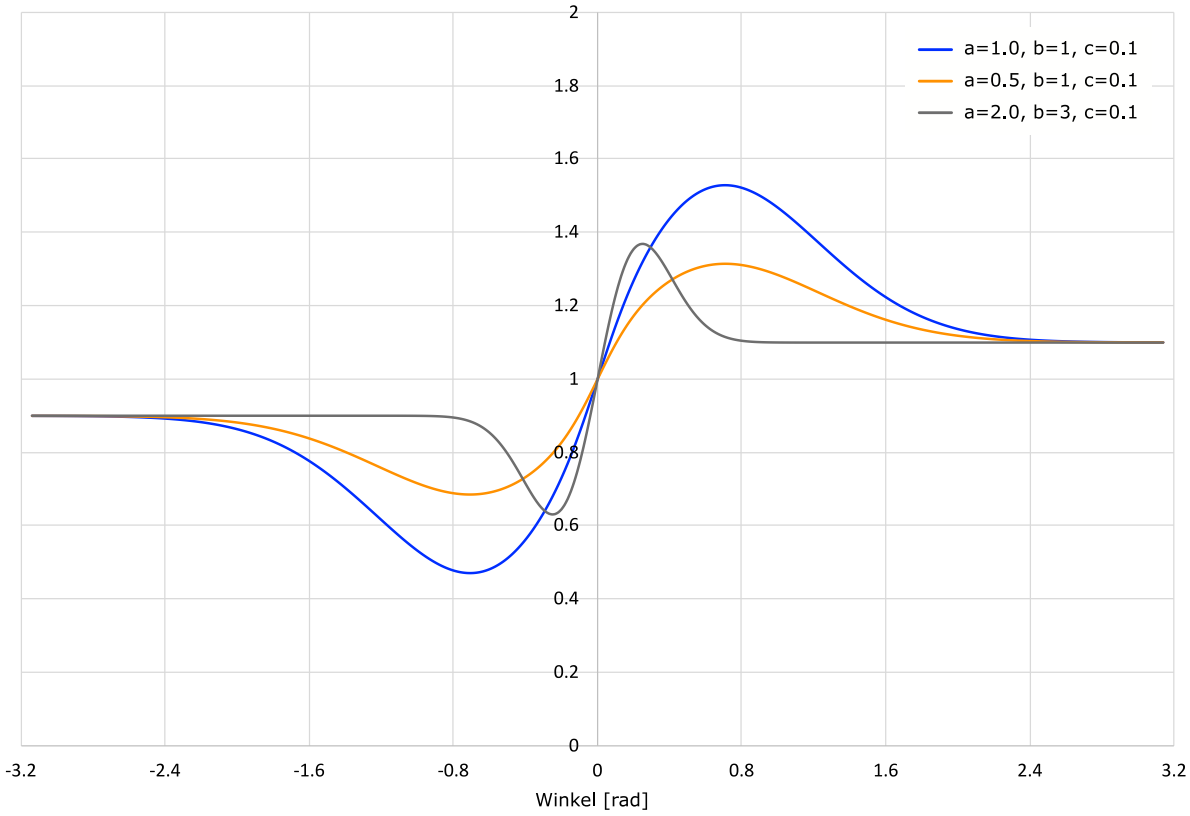

Abbildung 7.3: Werte der Funktion zum Rechtshalten auf Wegen (Quelle: eigene Darstellung)

Berechnungszeit. Der Nachteil einer zentralen Pfadplanung ist die benötigte Datenverbindung zum Shared Guide Dog. Besonders im Hinblick auf einen Einsatz des Shared Guide Dog im Inneren von Gebäuden, kann es hier zu Problemen kommen. Die Vorteile einer dezentralen Pfadplanung sind die Unabhängigkeit von einem zentralen Server und die nicht benötigte zusätzliche Infrastruktur, die bei der zentralen Steuerung benötigt wird. Eine Herausforderung stellt jedoch die Synchronisierung von Kartendaten zwischen den Shared Guide Dogs dar. Hierfür wird, wie bei der zentralen Steuerung, eine dauerhafte Datenverbindung zu anderen Shared Guide Dogs benötigt.

Eine abschließende Auswahl wird hier für kein System getroffen, da die Unsicherheiten an die Anforderungen während der weiteren Laufzeit des Projekts zu groß sind. Umgesetzt wird die Schnittstelle zum Shared Guide Dog eine Transmission Control Protocol/Internet Protocol (TCP/IP) Verbindung. Das Programm zur Pfadplanung agiert als Server und verarbeitet die Anfragen des Shared Guide Dog, läuft jedoch auf dem gleichen Notebook wie auch die Steuerung des Shared Guide Dog.

Das Datenformat zur Informationsübertragung zwischen dem Shared Guide Dog und dem Kartenserver basiert auf dem JavaScript Object Notation (JSON) Format. Es handelt sich dabei um ein Standardformat für den Datenaustausch, bei dem die Daten in Name-Value Pairs übertragen werden. Das Datenformat soll sowohl für eine dezentrale wie auch für eine zentrale Steuerung verwendet

werden. Der Name wird immer als String übertragen, die Values können Zahlen, Strings, Boolsche Werte oder Arrays sein. [9]

Für eine vollständige Anfrage an den Kartenserver müssen eine Start- und Zielposition und der Modus übertragen werden. Der Modus (*mode*) gibt an, ob eine Hinderniseintragung oder die Routenplanung durchgeführt werden soll. Für die Routenplanung ist dann die aktuelle Position als Startposition (*start*) und die Zielposition erforderlich. Dabei ist es möglich die Zielposition in Längen- und Breitengraden (*dest*) anzugeben oder als Adresse (*address*). Eine optionale Angabe ist der Nutzername (*username*). Wird er nicht angegeben, wird von der Pfadplanung automatisch der Default-Nutzer verwendet.

Der folgende Abschnitt zeigt die vollständige Anfrage einer Route von der U-Bahn Berliner Tor bis zur Adresse Berliner Tor 21. Das Feld *dest* wird der Vollständigkeit halber mit aufgeführt, auch wenn es in diesem Fall nicht benötigt wird.

```
{
    "mode": 0,
    "username": "default",
    "start": [53.5537574, 10.0240248],
    "dest": [0.000, 0.000],
    "address": "Berliner Tor 21"
}
```

8 Einbetten der Navigationslösung in den Shared Guide Dog

Für die Einbettung der Navigationslösung auf dem Shared Guide Dog werden in diesem Kapitel die Aufgaben Lokalisation, Planung des lokalen Pfades und Hinderniserkennung und -vermeidung betrachtet.

8.1 Sensordaten

Für die Lokalisation des Shared Guide Dog werden der GPS-Empfänger, die Inertial Measurement Unit (dt. Inertiale Messeinheit) (IMU) und die Odometriedaten verwendet. Dieses Kapitel gibt einen Überblick über die Ausgabe der Sensoren. Anhand der Daten soll das Verhalten der Sensoren diskutiert werden. Die Messwerte der Sensoren werden an drei Position und während drei Fahrten im Lohmühlenpark aufgenommen. Abbildung 8.1 gibt einen Überblick über die verwendeten Punkte und Wege. In den Kästen mit schwarzer Umrandung sind die Höhen der Gebäude nach [24] eingetragen. Die Höhen werden bei der Diskussion des GPS-Signals verwendet.

8.1.1 Sensordaten des GPS-Empfängers

Der GPS-Empfänger wird für die absolute Lokalisation in der Karte verwendet. Wie bereits in Kapitel 3.2.4 erläutert, gibt es jedoch Fehlerquellen, die zu einer schlechten oder sogar unbrauchbaren Positionsangabe führen können. Der Lohmühlenpark ist in Richtung Nordosten und Südwesten von hohen Gebäuden umgeben. Zudem stehen mehrere große Bäume im Park, die den Empfang von Satellitensignalen stören können.

Die Ermittlung der Positionsabweichung erfolgt anhand mehrerer Testläufe im Lohmühlenpark. Beim ersten Testlauf war der GPS-Empfänger vorne am Shared Guide Dog oberhalb des Vorderrades befestigt. In Abbildung 8.2 ist auf der linken Seite die wahre Route (*ground truth*) in schwarz und die gemessene Route in orange und grau eingezeichnet. Der Startpunkt ist unten in der Mitte vor dem Studierendenzentrum der HAW Hamburg. Beim Vergleich des gemessenen mit dem wahren Weg ist bereits zu Beginn der Messung eine deutliche Abweichung der grauen Linie zu erkennen. Die orangefarbene Linie startet am richtigen Startpunkt und hat auch danach deutlich geringere Abweichungen als die graue Linie. Die höchste Genauigkeit lässt sich beim Durchqueren des Parks in Richtung Berliner Tor 21 (BT21) erzielen. Bereits nach der halben Strecke stellt sich jedoch eine so große Abweichung ein, dass die Positionsangabe ab hier unbrauchbar wird. Diese Abweichung

Abbildung 8.1: Positionen und Fahrten im Lohmühlenpark zur Messwertaufnahme (Quelle: [42], eigene Darstellung)

ist wahrscheinlich auf die Annäherung an das Gebäude BT21 zurückzuführen, das mit einer Höhe von 29,1 m [24] das GPS-Signal reflektiert. Erst bei einem großen Abstand des Empfängers zum Gebäude wird der Fehler wieder kleiner, bleibt jedoch bei einer Abweichung von mehreren Metern.

Für den zweiten Testlauf ist der GPS-Empfänger am Lenker befestigt, wie bereits in Kapitel 3.3 gezeigt und zusätzlich werden GPS-Korrekturdaten des Dienstes *AssistNow Offline* verwendet. Die aufgezeichneten Routen sind in Abbildung 8.2 auf der rechten Seite gezeigt. Auf den ersten Blick ist sofort erkennbar, dass die Abweichung deutlich geringer ist. Nur der Startpunkt der orangefarbenen Linie weicht um mehrere Meter vom wahren Startpunkt ab, nach wenigen Metern wird die Position jedoch korrigiert und nähert sich der wahren Strecke an. Größere Abweichungen ergeben sich in dieser Konfiguration nur noch bei schnellen Bewegungen und Richtungswechseln. Deutlich ist das an der südlichsten Ecke des Testlaufs auf dem Basketballplatz zu sehen.

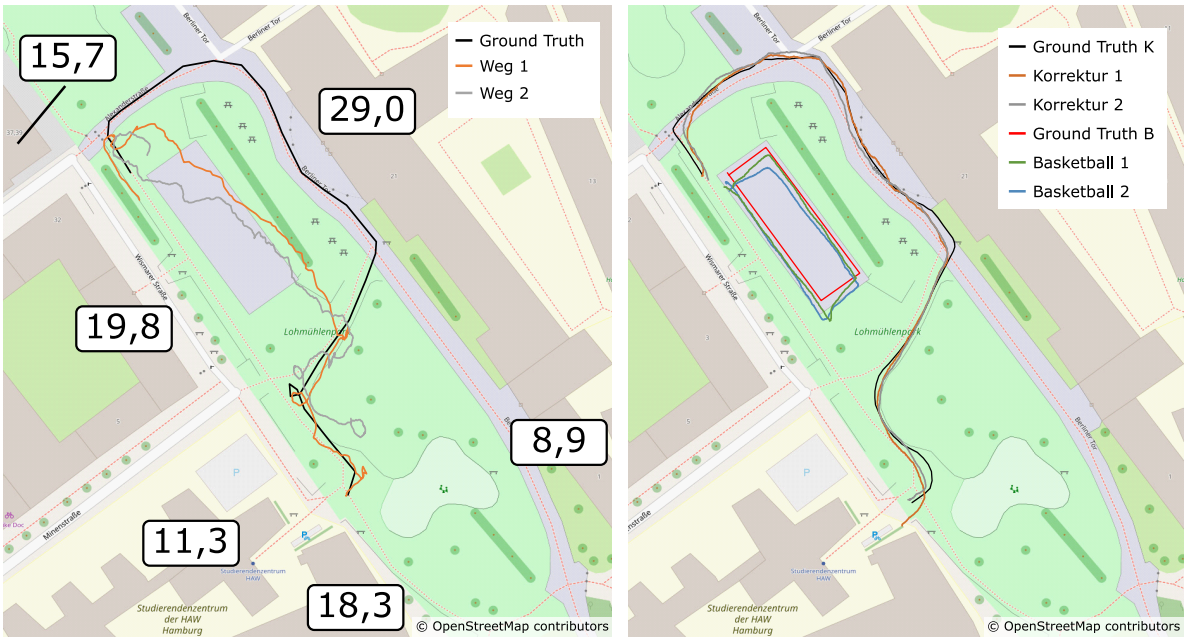

Abbildung 8.2: Links: GPS-Position ohne Korrektur, rechts: GPS-Position mit korrigierter Position (Quelle: [42], eigene Darstellung)

Aus den Abweichungen und den durchgeführten Methoden zur Reduzierung dieser Abweichungen, lassen sich Prozeduren zum Starten des GPS-Empfängers ableiten. Vor dem Beginn jeder Fahrt sollte sichergestellt werden, dass aktuelle Korrekturdaten verfügbar sind. Dadurch lässt sich der Fehler deutlich reduzieren. Bei schnellen Bewegungen und Richtungswechseln sollte zur Lokalisierung kurzzeitig auf andere Sensoren zurückgegriffen werden. Werden längere Strecken in eine Richtung gefahren, ist der Fehler im GPS-Signal kleiner und kann für die Lokalisierung verwendet werden. Eine Herausforderung ergibt sich aus einem kontinuierlich wachsenden Fehler im GPS-Signal. Dieser Fehler kann nicht aus dem GPS-Signal abgeschätzt werden und die IMU- und Odometriedaten unterliegen einem Drift, weshalb auch diese nicht zur Fehlerkorrektur verwendet werden können.

8.1.2 Sensordaten der IMU

Die IMU wird für die Aufnahme der Beschleunigungen und der Orientierung verwendet. Für die Berechnung der Orientierung werden die Drehrate des Gyroskops und die absolute Orientierung, berechnet aus den magnetischen Flüssen des Magnetometers, fusioniert. Dieser Prozess wird im internen Mikrocontroller der IMU durchgeführt und kann nicht angepasst werden. Die Messung im Stand wurde an den drei in Abbildung 8.1 eingezeichneten Positionen durchgeführt. Für die Messung während der Fahrt wurden die drei Fahrten abgefahren. Vor jeder Messung wurde die IMU vollständig kalibriert.

In Abbildung 8.4 sind zwei Diagramme mit den gemessenen Beschleunigungen in x-Richtungen an den drei Positionen und während der drei Fahrten dargestellt. Zu beachten ist die Skalierung der Ordinate. Im Stand sind die größten Beschleunigungen kleiner als $\pm 0{,}1\ \frac{m}{s^2}$, während der Fahrt wurden

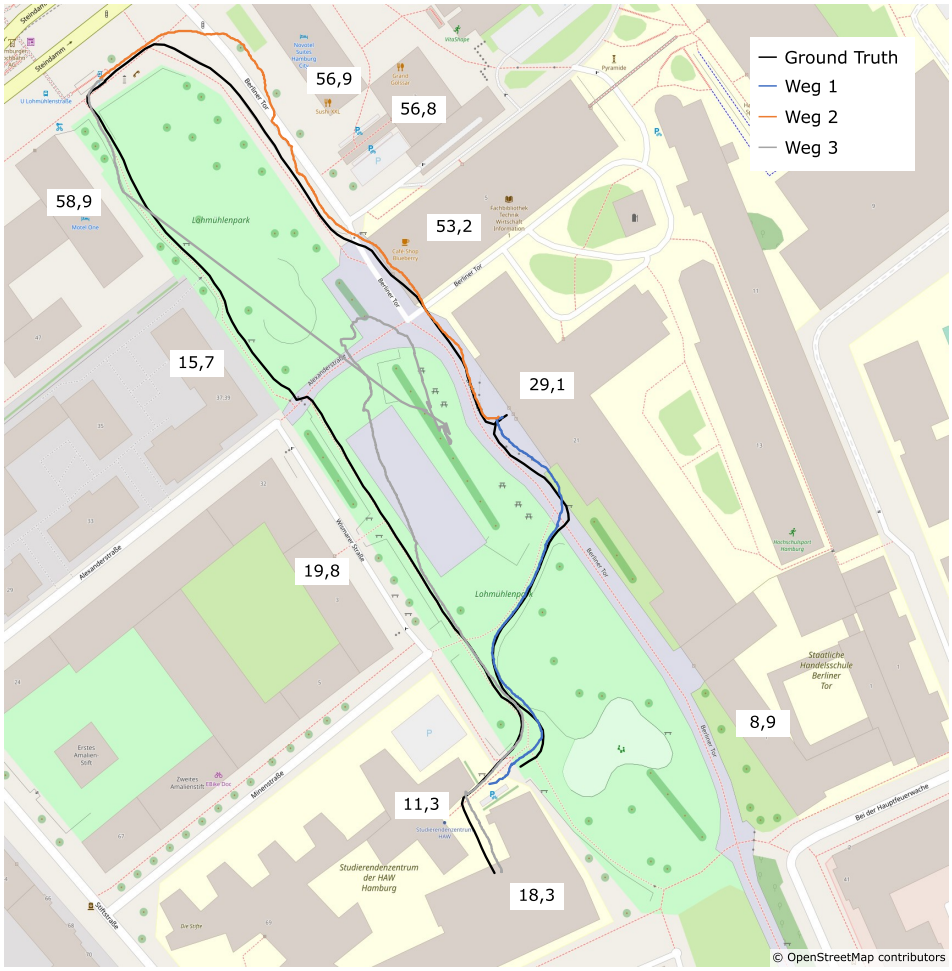

Abbildung 8.3: Aufgenommene GPS-Daten bei drei Fahrten durch den Lohmühlenpark (Quelle: OpenStreetMap, eigene Darstellung)

dagegen Beschleunigungen von bis zu 25,3 $\frac{m}{s^2}$ gemessen. Das Diagramm mit der Beschleunigung im Stand zeigt deutlich die unterschiedlichen Offsets der Messung. Bei keiner Messung wurde das Fahrzeug bewegt und die Messungen wurden alle am gleichen Tag innerhalb von einer Viertelstunde durchgeführt. Äußere Einflüsse können somit nahezu ausgeschlossen werden. Trotzdem liegt der Mittelwert der Messreihe für Position 3 bei ungefähr 0,02 $\frac{m}{s^2}$, während der Mittelwert für die Positionen 1 und 2 bei ungefähr -0,67 $\frac{m}{s^2}$ und -0,69 $\frac{m}{s^2}$ liegt. Unterschiedliche Offsets sind auch bei den Fahrten zu beobachten, durch die Skalierung fallen Sie dort jedoch nicht so stark auf.

Zur Prüfung der gemessenen Werte, wird aus diesen die Geschwindigkeit mit Gleichung 8.1 berechnet. Es ergeben sich die Endgeschwindigkeiten von 24,8 $\frac{m}{s^2}$ für Fahrt 1, 33,5 $\frac{m}{s^2}$ für Fahrt 2 und -2,5 $\frac{m}{s^2}$ für Fahrt 3. Als Offset wurde der Mittelwert der ersten 5 s der Messung verwendet. In diesem Zeitraum wurde die IMU nicht bewegt. Zur Glättung von Spitzen während der Messung

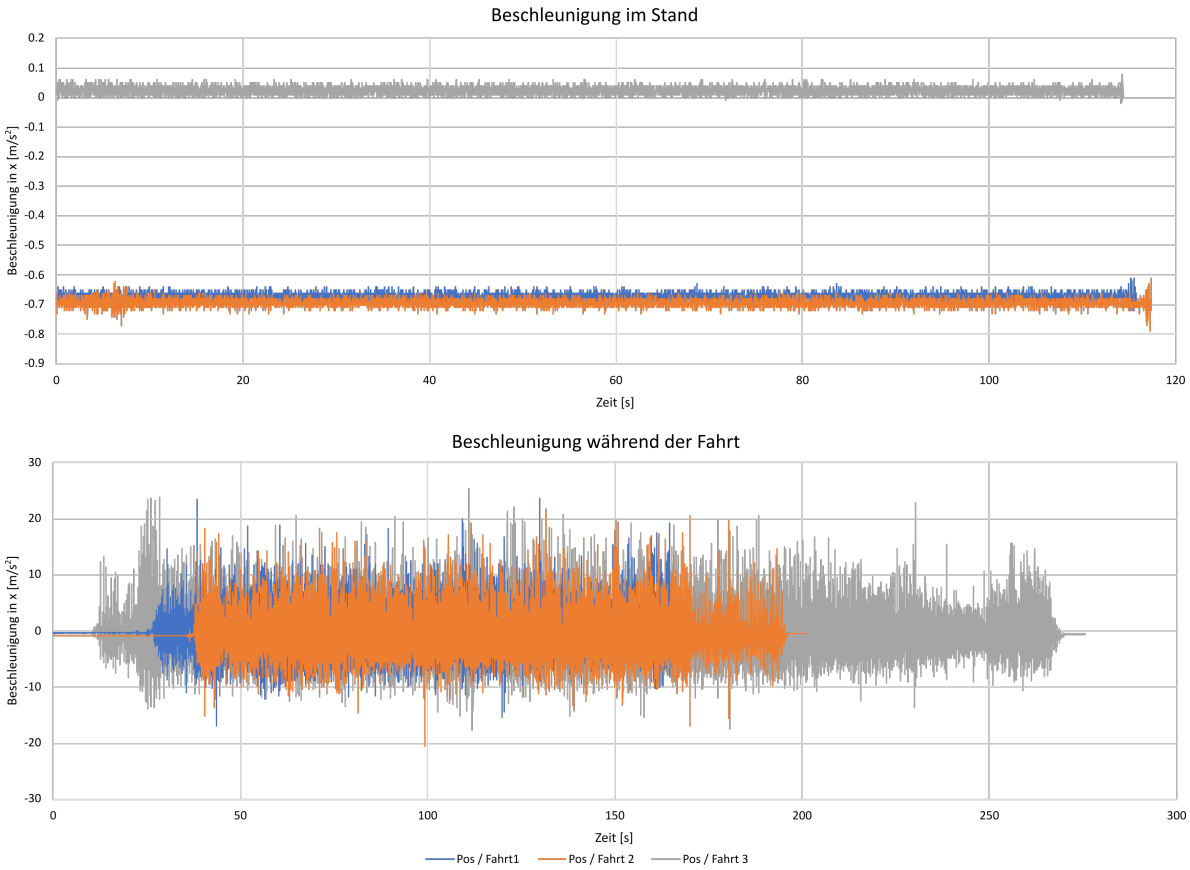

Abbildung 8.4: Gemessene Beschleunigung in x-Richtung im Stand und während der Fahrt (Quelle: eigene Darstellung)

wird bei der Geschwindigkeitsberechnung der gleitende Mittelwert der letzten fünf gemessenen Beschleunigungswerte verwendet.

$$v_i = v_{i-1} + \frac{\sum_{n=i-4}^{i} a_n}{5} \cdot (t_i - t_{i-1}) \tag{8.1}$$

mit

a	Gemessene Beschleunigung in x-Richtung
v	Geschwindigkeit
t	Zeitpunkt der Messung
i	Aktueller Zeitschritt

Die aufgenommenen Fahrten starten im Stand und enden wieder im Stand. Ein perfekter Sensor würde als Endgeschwindigkeit 0 $\frac{m}{s}$ ausgeben. Der Vergleich mit den tatsächlich berechneten Geschwindigkeiten zeigt jedoch deutliche Abweichungen von diesem Wert. Nur die Geschwindigkeitsberechnung für Fahrt 3 liegt, unter der Berücksichtigung von Messfehlern und einem Drift, in dem zu erwartenden Bereich. Der Grund für die Abweichungen konnte im Rahmen dieser Arbeit

nicht abschließend ermittelt werden. Eine Betrachtung der Beschleunigungen deutet auf einen Skalierungsfehler der IMU oder der Sensordatenfusion im Mikrocontroller der IMU hin. Möglich sind diese großen Beschleunigungen jedoch auch durch Vibrationen, die durch uneben Wege in das Fahrzeug eingebracht werden.

8.1.3 Odometriedaten

Die Odometriedaten werden vom Controller der Räder aufgenommen. Für jeden Antrieb wird ein separater Wert zurückgegeben, aus dem die Geschwindigkeit und Drehrate des Fahrzeugs berechnet werden können. Der zurückgegebene Wert des Controllers liegt in einem Bereich zwischen 0 und 1000. Ein Wert von 0 bedeutet keine Bewegung der Räder, bei einem Wert von 1.000 drehen die Räder mit der Maximalgeschwindigkeit. Da die Maximalgeschwindigkeit jedoch nicht bekannt ist, muss der Zusammenhang zwischen dem Wert des Controllers und der Rotationsgeschwindigkeit der Räder experimentell bestimmt werden. Die experimentelle Bestimmung erfolgt anhand der Fahrten aus Abbildung 8.5. Die während der Fahrten aufgenommenen Odometriedaten werden mit einem zunächst beliebigen Faktor multipliziert, um die Rotationsgeschwindigkeiten ω_r und ω_l zu erhalten, und anschließend mit Gleichung 8.4 und Gleichung 8.5 die Geschwindigkeit und Drehrate bestimmt. Durch eine Integration der Geschwindigkeit und Drehrate lässt sich die Position ermitteln. Die Optimierung der Faktoren erfolgt anhand der Distanz zwischen den Endpunkten der gemessenen und der wahren Strecken. Die beiden Umrechnungsfaktoren werden verwendet, um die Strecke zu minimieren. Damit ergeben sich die folgenden Gleichungen für die Umrechnung der Controllerausgabe in die Rotationsgeschwindigkeiten der Räder.

$$\omega_r = m_r \cdot \frac{1}{57,93} \tag{8.2}$$

$$\omega_l = m_l \cdot \frac{1}{59,00} \tag{8.3}$$

Ist ω bekannt, kann die Geschwindigkeit und Drehrate des Fahrzeugs mit Gleichung 8.4 und 8.5 bestimmt werden.

$$v_{odom} = \frac{(\omega_r + \omega_l) \cdot U}{2} \tag{8.4}$$

$$\dot{\omega}_{odom} = \frac{(\omega_r - \omega_l) \cdot U}{s_w} \tag{8.5}$$

mit

$U = 0,68\,m$	Umfang der Räder
$s_w = 0,71\,m$	Abstand der Räder zueinander

Die Integration der Odometriedaten ergibt die Position relativ zum Startpunkt. In Abbildung 8.5 sind die Odometriedaten der drei Testfahrten eingezeichnet. Als Startpunkt werden die Koordinaten der wahren Strecke (*ground truth*) verwendet. Die Anfangsorientierung wird von der IMU übernommen.

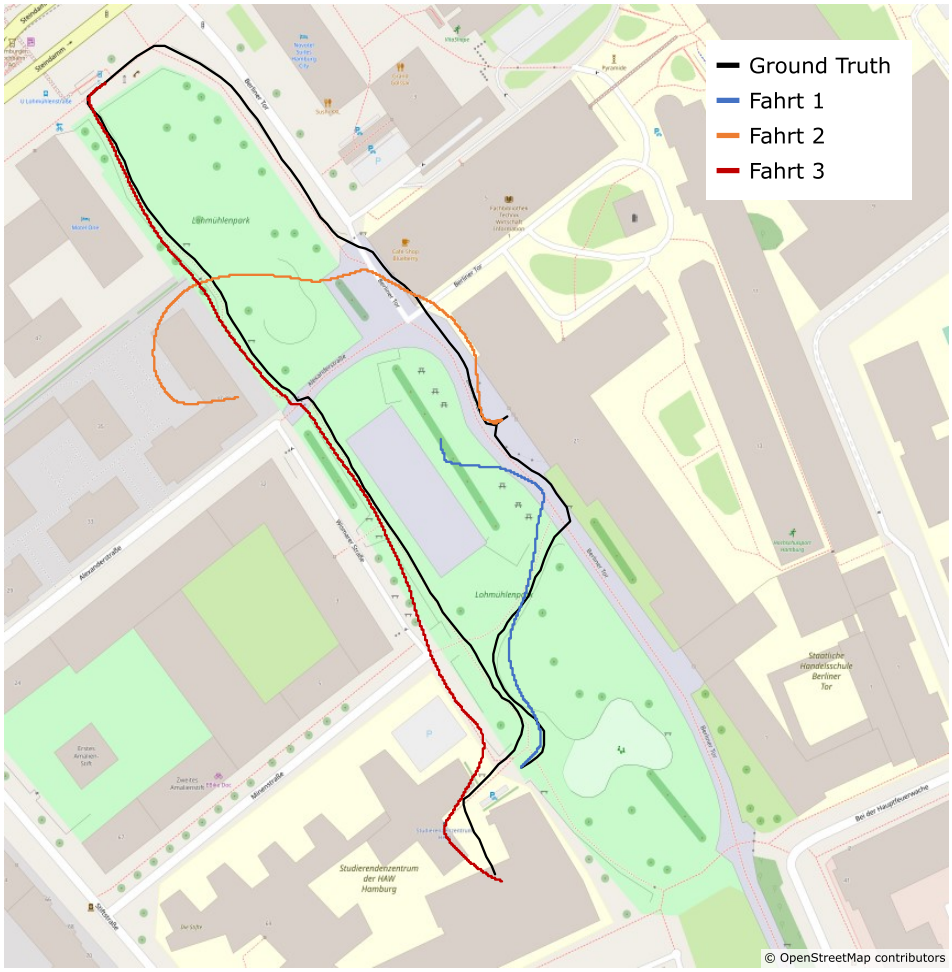

Abbildung 8.5: Position aus den Odometriedaten (Quelle: [42], Routen eigene Darstellung)

8.2 Sensordatenfusion mit Kalman Filter

Für die Positionsbestimmung im Navigation 2 Paket müssen die Transformationen zwischen drei Koordinatensystemen (*frames*) bereitgestellt werden. Das ist zum einen die Transformation zwischen der Karte (*map*) und dem Odometriesystem (*odom*) und zum anderen die Transformation zwischen dem Odometriesystem und dem Shared Guide Dog (*base_footprint*). Bei der Transformation zwischen dem Odometriesystem und dem Shared Guide Dog muss beachtet werden, dass diese Transformation kontinuierlich ist und keine diskreten Sprünge aufweist.

Die Berechnung der Transformation zwischen dem Karten-Koordinatensystem und dem Odometriesystem erfolgt durch eine Sensordatenfusion des GPS-Empfängers und der IMU mit einem Extended Kalman Filter (EKF). Das Kalman Filter wird häufig zur Filterung von verrauschten Messwerten und zur Fusion von Sensordaten verwendet. Dieses Kapitel stellt kurz die Schritte eines Kalman Filters und die Erweiterungen für den EKF dar. Anschließend folgt die Umsetzung und Einstellung auf dem Shared Guide Dog.

8.2.1 Vorgehen beim Kalman Filter

Die Berechnung eines Kalman Filters teilt sich in die Bereiche Vorhersage (engl. prediction) und Schätzung (engl. estimation) auf. Während der Vorhersage wird der aktuelle Systemzustand verwendet, um eine Vorhersage zum nächsten Systemzustand abzugeben. Für die Schätzung werden Messwerte verwendet, um den nächsten Systemzustand zu schätzen. Diese beiden Bereiche werden zusammengeführt und daraus ein wahrscheinlicher Systemzustand berechnet. Das Zusammenführen der Systemzustände erfolgt mit den Fehlerkovarianzen der Sensoren. [20, S. 48 ff.]

Für die Implementierung des Kalman Filter wird das *robot_localization* Paket für ROS2 verwendet. Das Paket implementiert einen EKF. Das EKF erweitert das ursprüngliche Kalman Filter, das nur lineare Systeme abbilden kann, sodass es auch für nichtlineare Systeme eingesetzt werden kann. Mit dem enthaltenen EKF kann die Pose des mobilen Roboters im dreidimensionalen Raum abgeschätzt werden. Benötigt werden dazu Sensordaten, die die Bewegungen des Fahrzeugs beschreiben, und die Fehlerkovarianzen der Sensoren, damit eine Abschätzung über die Güte der Positionsangabe getroffen werden kann. Das Paket erlaubt eine unbegrenzte Anzahl an Sensoren, von denen jeder eingestellt werden kann, inwiefern die gemessenen Daten einen Einfluss auf die berechnete Pose haben. [34]

Für das Systemmodell werden die allgemeinen Newtonschen Bewegungsgleichungen angewendet. Der GPS-Empfänger liefert eine Position in Längen- und Breitengrad und die IMU die Beschleunigung in drei Richtungen, die Drehrate und die Ausrichtung relativ zur Erdoberfläche. Bei der Modellierung müssen zusätzlich die Positionen der Sensoren berücksichtigt werden. Als Ursprung für das Koordinatensystem wird der Mittelpunkt zwischen den beiden hinteren Rädern verwendet. Die Achsenausrichtung erfolgt nach REP103 [13]. In ROS2 wird dieses Koordinatensystem als *base_footprint* bezeichnet. Der GPS-Empfänger und die IMU sind in einem gemeinsamen Gehäuse am Lenker untergebracht. Die Verschiebung in x-Richtung ist bei beiden Sensoren identisch. In y-Richtung liegen beide Sensoren in der Mitte des Fahrzeugs. In der Höhe liegen die Sensoren ungefähr 40 mm auseinander. Alle Werte sind in Tabelle 8.2.1 aufgelistet. Die Berücksichtigung der Positionen der Sensoren erfolgt im *robot_localization* Paket automatisch, sofern das Robotermodell in ROS korrekt definiert wurde. [65]

	GPS	IMU
x [m]	0,43	0,43
y [m]	0,00	0,00
z [m]	0,93	0,89

Tabelle 8.1: Position der Sensoren auf dem Shared Guide Dog

8.2.2 Berechnung der Fehlerkovarianzmatrizen

Ein wichtiger Parameter zur Berechnung des Kalman Filters ist die Fehlerkovarianz der Sensoren. Dazu werden für jeden Sensor Fehlerkovarianzmatrizen definiert. Im *robot_localization* Paket sind diese Matrizen als Diagonalmatrizen spezifiziert.

$$Q = \begin{bmatrix} var(x) & cov(x,y) \\ cov(y,x) & var(y) \end{bmatrix} \tag{8.6}$$

Die Varianz ist definiert als Quadrat der Standardabweichung [57, S. 299 f.]. Die Kovarianz setzt die Messwerte von x und y in Beziehung. Bei allen hier verwendeten Sensoren wird davon ausgegangen, dass sich die Abweichungen der Einzelmessungen nicht gegenseitig beeinflussen, die Verteilung der Messwerte ist unabhängig voneinander. Die Kovarianz wird damit null. [33, S. 199 ff.]

$$var(x) = \frac{\sum(x - \tilde{x})}{n-1} = \sigma^2 \tag{8.7}$$

Zur Bestimmung der Fehlerkovarianzmatrix des GPS-Empfängers wird der Circular Error Probable (dt. Streukreisradius) (CEP) betrachtet. Im Datenblatt wird dieser Wert mit 2,5 m angegeben (vgl. Kapitel 3.3). Der Streukreisradius ist als Radius eines Kreises, in dem 50 % aller Messwerte liegen, definiert. Wird von einer Gaußschen Normalverteilung ausgegangen, ist der Bereich, in dem 50 % aller Messwerte liegen ungefähr $0,67\sigma$ [33, S. 290]. Daraus lässt sich mit Gleichung 8.8 die Varianz, also das Quadrat der Standardabweichung, bestimmen.

$$var(x) = \left(\frac{CEP}{0,67}\right)^2 = 13,738 \tag{8.8}$$

Für die IMU ist kein typischer Fehler angegeben. Zur Ermittlung der Fehlerkovarianzmatrix werden die Messwerte an den drei im vorigen Kapitel Kapitel behandelten Positionen verwendet. Da sich der Shared Guide Dog während der Messung im Stillstand befand, ist der Erwartungswert $\tilde{x} = 0$. Mit Gleichung 8.7 und der Definition der Fehlerkovarianzmatrix nach Gleichung 8.6 ergibt sich die Fehlerkovarianzmatrix der IMU wie in Gleichung 8.9 angegeben. Für die Positionsbestimmung sind nur die Beschleunigung in x-Richtung, Orientierung und Drehrate um die z-Achse entscheidend.

$$R_{IMU} = \begin{bmatrix} var(\omega) & 0 & 0 \\ 0 & var(\dot{\omega}) & 0 \\ 0 & 0 & var(\ddot{x}) \end{bmatrix} = \begin{bmatrix} 1,657 \cdot 10^{-4} & 0 & 0 \\ 0 & 5,923 \cdot 10^{-3} & 0 \\ 0 & 0 & 1,301 \cdot 10^{-4} \end{bmatrix} \tag{8.9}$$

Bei dieser Fehlerkovarianzmatrix ist zu beachten, dass die Messwerte im Stand aufgenommen wurden. Während der Fahrt hat die IMU einen deutlich größeren Fehler. Basierend auf den Messwerten in Abbildung 8.4 wird die Abschätzung getroffen, dass die Fehler ungefähr 2 Größenordnungen größer sind als im Stillstand. Die Abschätzung basiert lediglich auf Erfahrungswerten, die während der Testfahrten gesammelt wurden. Damit ergibt sich die folgende Kovarianzmatrix, die für die Berechnung des Kalman Filters verwendet wird.

$$R_{IMU} = \begin{bmatrix} 1,657 \cdot 10^{-2} & 0 & 0 \\ 0 & 5,923 \cdot 10^{-1} & 0 \\ 0 & 0 & 1,301 \cdot 10^{-2} \end{bmatrix} \tag{8.10}$$

Eine experimentelle Ermittlung der Fehlerkovarianzmatrix für die Odometriedaten ist im Rahmen dieser Arbeit nicht möglich. Ein Vorgehen, wie es bei der IMU gewählt wurde, bietet sich nicht an, da im Stand der Wert 0 ohne Rauschen zurückgegeben wird. Die Fehlerkovarianzmatrix der Odometrie muss während der Testfahrten ermittelt und immer wieder angepasst werden. Für die Lokalisation wurde die folgende Fehlerkovarianzmatrix verwendet.

$$R_{odom} = \begin{bmatrix} 1 \cdot 10^{-3} & 0 & 0 \\ 0 & 1 \cdot 10^{-3} & 0 \\ 0 & 0 & 1 \cdot 10^{-3} \end{bmatrix} \tag{8.11}$$

8.3 Verarbeitung der Wegpunkte

Der Shared Guide Dog kriegt auf die Anfrage eines Weges eine Liste mit Wegpunkten vom Pfadplanungsalgorithmus geliefert. Anhand dieser kann jedoch noch nicht navigiert werden. Es muss zunächst ein kontinuierlicher Pfad durch die Wegpunkte geplant werden. Die einfachste Möglichkeit Punkte zu verbinden ist durch gerade Linien. Zu diesem Zweck wird das Navigation 2 Plugin *Straight Path Planner* auf Grundlage der Navigation 2 Tutorials [37] implementiert. Das Plugin nutzt Gleichung 8.12 für die Berechnung der Anzahl neuer Punkte mit dem Abstand a zueinander. In einer Schleife können diese neuen Punkte anschließend erstellt werden und dem Controller Server als Pfad zur Verfügung gestellt werden. [37]

$$n = \frac{\sqrt{(\Delta x)^2 + (\Delta y)^2}}{a} \tag{8.12}$$

8.4 Hinderniserkennung und Fahrzeugüberwachung

Die Sicherheit des Shared Guide Dog wird durch zwei Mechanismen sichergestellt. Im Normalbetrieb übernimmt das Navigation 2 Paket die Steuerung des Fahrzeugs und über die Verhaltensweisen des Behaviour Trees wird das Fahrzeug zum Beispiel bei Annäherung an ein Hindernis langsamer und stoppt schließlich. Treten jedoch Ereignisse auf, die eine Fortsetzung des Normalbetriebs nicht

zulassen, beispielsweise der Absturz eines Teils des Navigation 2 Pakets, wird eine Subsumption-Architektur als Rückfallebene implementiert (vgl. Kapitel 6.3). Die Redundanz der Systeme hat das Ziel, das System sicherer bei Ausfällen zu gestalten.

Eine Subsumption-Architektur basiert auf mehreren Schichten (engl. layer), die sich gegenseitig überschreiben können. Die Schichten werden dafür nach Priorität sortiert. Schichten mit einer hohen Priorität können Schichten mit niedrigerer Priorität überschreiben, jedoch nicht andersherum. Die Nachrichtentypen von ROS2 bieten keine Möglichkeiten, Nachrichten mit Prioritäten zu versehen und sie dementsprechend zu sortieren. Deshalb wird ein neuer Knoten mit der Bezeichnung *Subsumption Manager* implementiert, der diese Aufgabe übernimmt. Der Knoten ist an letzter Stelle, bevor der Befehl zum Antrieb gesendet wird, vorgesehen. Der Aufbau des Subsumption Managers ist in Abbildung 8.6 gezeigt.

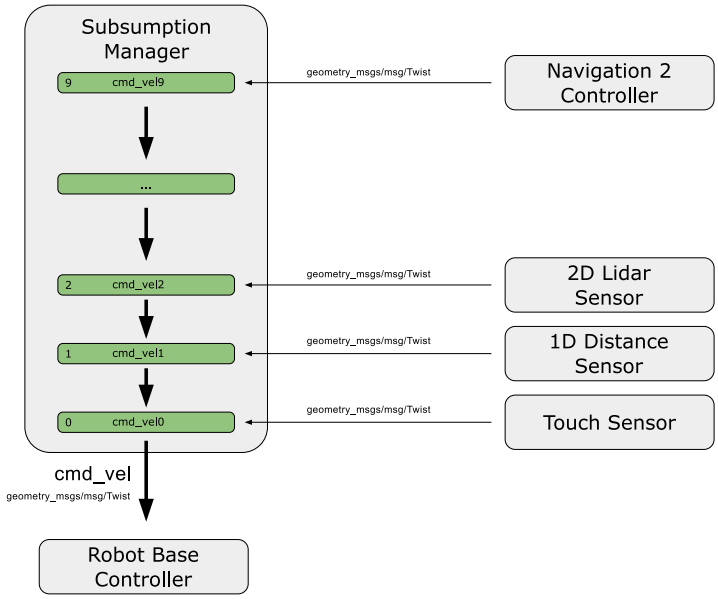

Abbildung 8.6: Implementierung der Subsumption Architektur auf dem Shared Guide Dog (Quelle: eigene Darstellung)

Der Subsumption Manager kann die Nachrichten auf den Topics *cmd_vel0* bis *cmd_vel9* überwachen. Das Topic *cmd_vel0* hat die höchste Priorität, das Topic *cmd_vel9* die niedrigste. Mit der niedrigsten Priorität werden die Steuerungsbefehle des Navigation 2 Pakets versehen. Die Prioritäten acht bis drei sind nicht dargestellt, da sie für den Prototypen derzeit nicht benötigt werden. Diese Schichten sind für Erweiterungen in Zukunft vorgesehen. Mit der Priorität zwei wird der 2D Lidarsensor versehen, der zur Hindernisvermeidung eingesetzt wird. Die beiden höchsten Prioritäten sind für Sensoren, die den Shared Guide Dog auf den Nutzer oder die Nutzerin reagieren lassen, reserviert. Der 1D Abstandssensor erfasst den Abstand zwischen dem Nutzer oder der Nutzerin und dem Fahrzeug. Wird der Abstand zu groß, bremst das Fahrzeug automatisch ab. Die höchste Priorität besitzt der Berührungssensor. Nur wenn beide Hände des Nutzers oder der Nutzerin

erkannt werden, darf das Fahrzeug fahren. Andersherum heißt das, dass bei einem Kontaktabbruch das Fahrzeug sofort stoppen soll.

Die Hindernisvermeidung erfolgt im Navigation 2 Paket mit einer lokalen Rasterkarte, die mit Daten des Lidar-Sensors gefüllt wird. Die Anforderung, dass der Shared Guide Dog bei einem Hindernis stoppen soll, kann auch ohne Karte umgesetzt werden. Die Karte bietet im weiteren Verlauf des Projekts jedoch große Vorteile bei der lokalen Hindernisvermeidung und wird daher nicht ersetzt.

9 Validierung der erarbeiteten Navigationslösung

In diesem Kapitel wird die entwickelte Navigationslösung getestet. Die Validierung teilt sich in die drei Bereiche Interpolation und Vernetzung, Pfadplanung und Umsetzung auf dem Shared Guide Dog.

9.1 Interpolation und Vernetzung

Bei der Interpolation und Vernetzung liegt das Augenmerk auf der korrekten Berechnung neuer Knoten und deren Vernetzung mit den alten Knoten. Dazu wird eine Datei aus Open Street Map (OSM) heruntergeladen, manuell korrigiert und augmentiert. Bei der Bearbeitung der Karte sollen die Punkte Korrektheit der Datenverarbeitung, also die Überprüfung, ob Rundungsfehler oder ähnliches auftreten, die Prüfung der Interpolation mit mehreren Schwellenwerten und die Prüfung der Vernetzung mit mehreren Schwellenwerten überprüft werden.

Bei der Interpolation der Strecke zwischen zwei Punkten soll zum einen die Anzahl der eingefügten Punkte und zum anderen die Übereinstimmung mit der originalen Strecke überprüft werden. Die Anzahl der eingefügten Punkte lässt sich mit Gleichung 9.1 leicht überprüfen. Die Anzahl der eingefügten Punkte n_P muss gleich der Anzahl der Sollpunkte n_S, die aus der Distanz d und dem Schwellenwert t für das Einfügen neuer Punkte berechnet wird, sein.

$$n_P = \lfloor \frac{d}{t} \rfloor = n_S \tag{9.1}$$

Diese Überprüfung wird für einen Teil der Straße Berliner Tor durchgeführt. Die Berechnung der Distanz erfolgt mit dem JOSM Measurement Plugin und zur Überprüfung mit Gleichung 9.2. Der Längengrad wird mit l bezeichnet und der Breitengrad wird mit b bezeichnet.

$$d = 111.319 \cdot \sqrt{(b_1 - b_2)^2 + \left((l_1 - l_2) \cdot \cos \frac{b_1 + b_2}{2}\right)^2} \tag{9.2}$$

Die Distanz zwischen den Punkten ergibt sich zu 9,272 m zwischen Punkt 1 und 2 und zu 67,570 m zwischen Punkt 2 und 3. Die Anzahl der eingefügten Punkte ist in Tabelle 9.1 eingetragen. Die Werte sind alle identisch mit der nach Gleichung 9.1 berechneten Anzahl an Punkten. Neben der Anzahl der Punkte ist die Richtung der neu eingefügten Punkte wichtig. Die eingefügten Punkte müssen alle auf der direkten Verbindung zwischen den beiden ursprünglichen Punkten liegen. Werden

weitere Punkte über der Breite des Weges eingefügt, sollen diese in einem rechten Winkel zum Weg angeordnet sein.

Tabelle 9.1: Eingefügte Punkte während der Interpolationstests

	Anzahl der Punkte	
Interpolation	Strecke 1	Strecke 2
2,0 m	4	33
5,0 m	1	13
15,0 m	0	4

Die Anzahl der Punkte ist in diesem Test korrekt und auch die optische Überprüfung mit Abbildung 9.1 der neu eingefügten Punkte ergibt, dass diese korrekt eingefügt wurden. Es kann also davon ausgegangen werden, dass die Interpolation korrekt arbeitet.

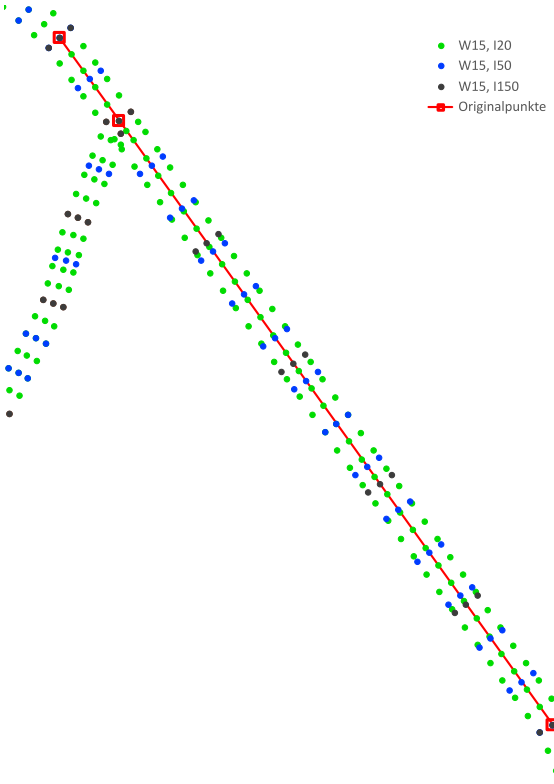

Abbildung 9.1: Ergebnis der Interpolation von zwei Strecken (Quelle: eigene Darstellung)

Nach der Interpolation muss das Einfügen und Vernetzen neuer Knoten überprüft werden. Es wird dazu eine Karte mit der Interpolationsweite von 5 m und einer minimalen Breite des Weges für das Einfügen neuer Punkte von 1,5 m erstellt. Die Karte ist in Abbildung 9.2 auf der linken Seite vor der Vernetzung und auf der rechten Seite nach der Vernetzung dargestellt.

Der Weg in Richtung Norden wurde nicht vernetzt, da die Breite mit 1,5 m gleich dem Schwellenwert ist und damit keine neuen Knoten eingefügt werden. Bei der optischen Überprüfung der Verbindungen kann kein Fehler festgestellt werden. Knoten, die innerhalb eines Weges liegen, werden

Abbildung 9.2: Links: Navigationsdaten vor der Vernetzung, rechts: Daten nach der Vernetzung (Quelle: [42], eigene Darstellung)

nicht untereinander vernetzt, während die neuen Knoten bei Kreuzungen Verbindungen untereinander besitzen. Die genaue Betrachtung der rechten Kreuzung zeigt jedoch, dass zwischen dem Weg nach Süden und dem Weg nach Osten ein spitzer Winkel entsteht. An dieser Stelle stellt der Winkel kein Problem dar, doch es muss im weiteren Verlauf des Projekts überprüft werden, ob sich dadurch Probleme ergeben.

Abschließend kann zu der Interpolation, Erstellung neuer Knoten und deren Vernetzung während der Tests keine Fehler aufgetreten sind. Für die Überprüfung wurde jedoch immer das Testgebiet im Lohmühlenpark verwendet und somit kann nicht abgeschätzt werden, wie sich der Algorithmus bei der Anwendung auf ein anderes Gebiet verhält. Für das Projekt ist dieser Aspekt derzeit jedoch nicht entscheidend, da der Shared Guide Dog zurzeit nur im Lohmühlenpark operieren soll.

9.2 Pfadplanung

Die Validierung der Pfadplanung wird zwischen fünf ausgewählten Punkten durchgeführt. Diese Punkte werden nacheinander als Startpunkt und die anderen Knoten als Ziel verwendet. Es ergeben sich somit 20 Strecken, die überprüft werden. In Abbildung 9.3 sind die vier Strecken vom Startpunkt *Campus Berliner Tor* in Rot dargestellt. Die Berechnung der Strecken wurde mit dem Default-User durchgeführt. An diesen Strecken werden die Anforderungen *rechts fahren* (P-4) und Vermeiden gefährlicher Strecken (P-3) überprüft. Beispielhaft ist in Abbildung 9.3 ein Startpunkt verbunden mit allen Zielpunkten dargestellt. Als Navigationskarte wird eine Karte mit einer Interpolation über 5 m und einer minimalen Wegbreite für neue Knoten von 1,5 m verwendet.

In der Abbildung ist zu erkennen, dass alle Wege auf der rechten Seite des Weges verlaufen. Trotzdem werden keine überflüssigen Rechtsschlenker eingebaut. Das Vermeiden gefährlicher Strecken wird mit einem zweiten Nutzer überprüft. Dieser Nutzer hat hohe Kosten für Service-Wege, wie etwa

Auffahrten auf Parkplätze oder zu Gebäuden, hinterlegt. In der Abbildung ist die berechnete Strecke in grün dargestellt. Zu Beginn des Weges verläuft der Weg noch genauso wie der rote Weg des Default-Users. Der Default-User wählt im Süden jedoch die Strecke über den Parkplatz, während der zweite Nutzer einen Weg mit möglichst kurzer Strecke auf Service-Straßen wählt.

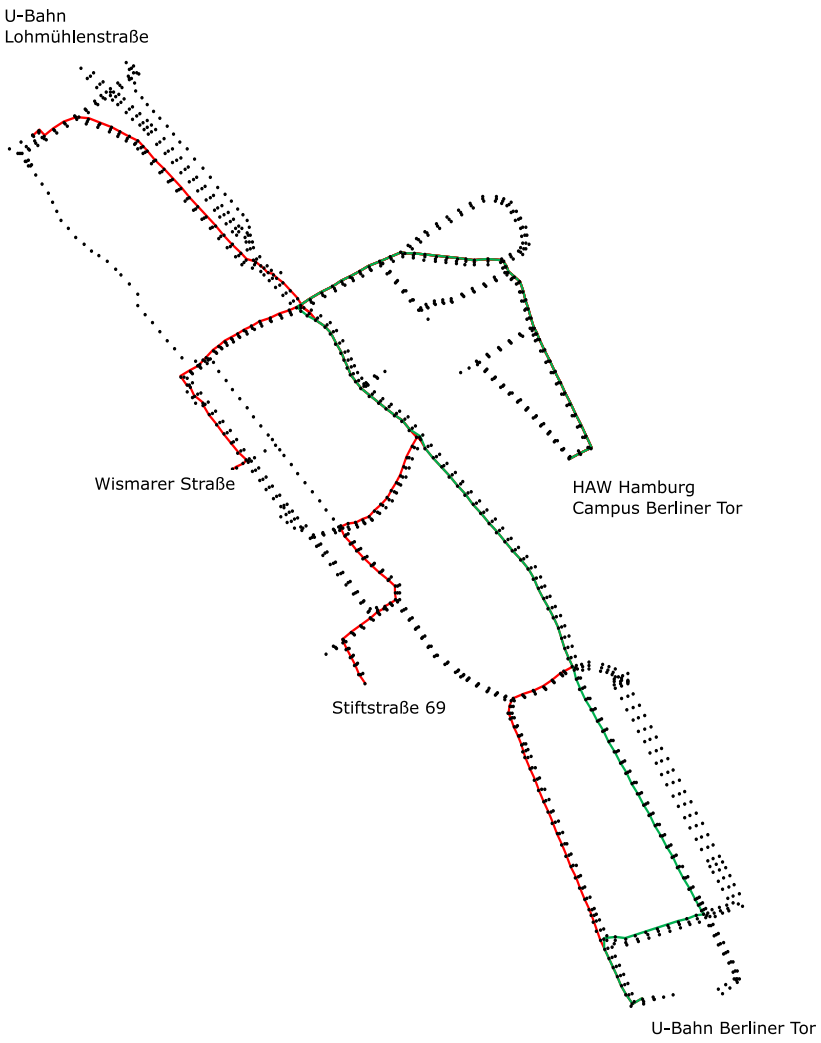

Abbildung 9.3: Beispielhafte Strecken von einem Startpunkt zu allen Zielpunkten (Quelle: eigene Darstellung)

9.3 Integration in den Shared Guide Dog 4.0

Bei den Tests in der Realität hat sich herausgestellt, dass die auf dem Shared Guide Dog verbauten Sensoren nicht für eine Lokalisierung und Navigation verwendet werden können. Im Stillstand des Shared Guide Dog arbeiten alle Sensoren wie im vorigen Kapitel beschrieben. Sobald sich der Shared Guide Dog bewegt, werden die Abweichungen der Sensoren so groß, dass keine robuste Lokalisierung mehr möglich ist. Die größte Herausforderung stellt die Inertial Measurement Unit (dt. Inertiale

Messeinheit) (IMU) dar. Zum einen besitzen die Messwerte ein Offset, das durch eine Kalibrierung größtenteils ausgeglichen werden kann. Zum anderen sind die Messwerte bei einer langsamen Gehgeschwindigkeit mit über 25 $\frac{m}{s^2}$ deutlich größer als es zu erwarten wäre. Für diese Messwerte können zwei mögliche Gründe identifiziert werden. Es kann sich um einen Skalierungsfehler der IMU oder der Sensordatenfusion auf der IMU handeln. Möglich ist jedoch auch, dass durch die steife Bauweise des Fahrzeugs Bodenunebenheiten ohne Dämpfung auf die IMU übertragen werden und zu diesen großen Ausschlägen führen. Eine abschließende Analyse der Messwerte konnte im Rahmen dieser Arbeit nicht durchgeführt werden.

Die Integration in den Shared Guide Dog 4.0 wird in der Simulation überprüft, da die Auswahl der Sensoren für den Prototypen keine robuste Lokalisierung zulässt. In der Simulation soll die Funktionalität der Datenübertragung vom Pfadplanungsalgorithmus zum Shared Guide Dog, die Planung eines Weges zwischen den Wegpunkten und das Erkennen von Hindernissen validiert werden. Für die Validierung der Pfadplanung und Steuerung entlang des Weges werden erneut die bereits für die Validierung der Pfadplanung definierten Strecken verwendet.

Die Berechnung der Route vom Shared Guide Dog hat während der Tests fehlerfrei funktioniert. Auch die Steuerung des Shared Guide Dog entlang der Strecke hat funktioniert. Dabei ist jedoch zu beachten, dass die Sensoren in der Simulation zwar ein Fehlermodell hinterlegt haben, aber keinem Drift oder einem Mehrwegefehler beim GPS-Signal unterliegen. Die Ergebnisse sind also nur bedingt in die Praxis übertragbar.

Aussagekräftigere Tests können mit der Hinderniserkennung durchgeführt werden. Dazu wurden während der Fahrt des Shared Guide Dogs Hindernisse im Weg eingefügt. Getestet wurden Hindernisse mit einer Breite von 0,5 m und größer. Die Hinderniserkennung hat in allen Testfällen gebremst und den Shared Guide Dog zum Stillstand gebracht. Nur bei Hindernissen, die in kurzem Abstand vor dem Shared Guide Dog eingefügt wurden, konnte der Shared Guide Dog nicht mehr rechtzeitig bremsen. In Abbildung 9.4 ist der Shared Guide Dog in der Simulation vor einem Hindernis zu sehen. In blau sind die Laserstrahlen des Lidarsensors dargestellt. Sie dienen als optische Kontrolle, ob ein Hindernis erkannt wurde oder nicht.

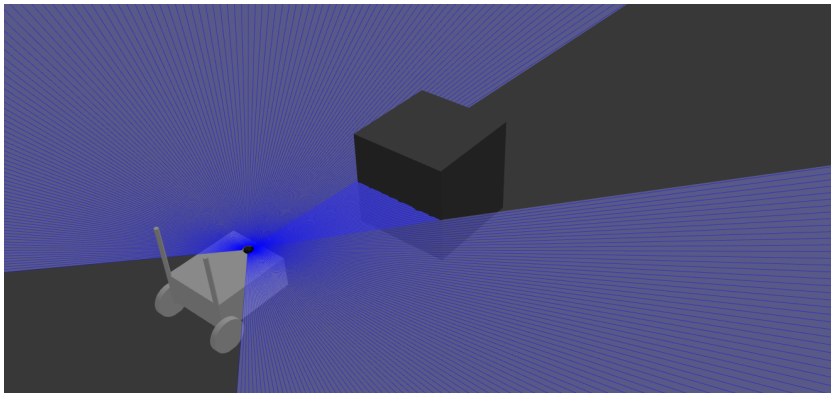

Abbildung 9.4: Der Shared Guide Dog in der Simulation vor einem Hindernis (Quelle: eigene Darstellung)

10 Fazit und Ausblick

In dieser Arbeit wurde eine Software zur Bereitstellung von Navigationsdaten aus dem offenen Kartendienst Open Street Map (OSM) entwickelt. Es konnte gezeigt werden, dass sich die Kartendaten von Open Street Map im innerstädtischen Bereich aufgrund der hohen Detaildichte und Genauigkeit gut für die Navigation eines mobilen Roboters eignen. Auf Basis der Informationen aus der Karte wurde eine Kostenfunktion für einen A*-Algorithmus entwickelt, die die Gegebenheiten für Fußgänger, wie die Art des Weges und die Oberfläche, und verschiedene Anforderungen von unterschiedlichen Nutzern und Nutzerinnen berücksichtigen kann. Zuletzt wurde die Steuerung entlang des geplanten Weges in den Shared Guide Dog integriert.

Die Software wurde für den aktuellen Forschungsstand des Shared Guide Dogs entwickelt. Durch das flexible OSM-Kartenformat mit den frei belegbaren Attributen und der Anpassbarkeit der Kosten für die Attribute, kann die Software einfach an weitere Anforderungen angepasst werden. Das System zum Eintragen der Hindernisse ist für den Prototyp ausreichend, jedoch sollte im weiteren Verlauf des Projekts die Entfernung von Hindernissen aus der Karte angepasst werden, damit sichergestellt werden kann, dass die Hindernisse entfernt wurden. Es bietet sich zudem an verschiedene Klassen von Hindernissen zu implementieren, die kurzfrisitge, mittelfristige und langfristige Hindernisse berücksichtigen. Kurzfristige Hindernisse können zum Beispiel Pfützen sein. Zu den mittelfristigen Hindernissen lassen sich parkende Autos zählen. Als langfristige Hindernisse können Baustellen klassifiziert werden.

Der A*-Algorithmus bietet durch die Konfigurierbarkeit der Nutzer eine hohe Flexibilität bei der Routenplanung. Denkbar sind in diesem Bereich weitere Einstellmöglichkeiten, beispielsweise die temporäre Überschreibung des Nutzerprofils oder einen *Zeitmodus*, in dem immer die schnellste Route gewählt wird.

Die grundsätzliche Funktionalität der Steuerung und Hindernisvermeidung konnte in der Simulation gezeigt werden. Durch die Integration weiterer Sensoren kann die Sicherheit des Nutzers oder der Nutzerin jedoch in Zukunft noch weiter gesteigert werden. Besondere Herausforderungen stellen Treppenabgänge oder herabhängende Hindernisse dar. Weitere Sensorsysteme lassen sich einfach an den Subsumption Manager in ROS2 anbinden, sodass auch diese Sensoren als Sicherheitssystem verwendet werden können. In diesem Bereich sind jedoch noch weitere Arbeiten und Tests notwendig, um die Sicherheitsfunktionen in vielen unterschiedlichen Situationen zu testen.

Im Laufe der Arbeit haben sich besonders zwei Aspekte als weitere Forschungsgebiete herausgestellt. Das sind die lokale Hindernisvermeidung und die Lokalisation des Shared Guide Dog. Für die Vermeidung von statischen Hindernisse reicht die Umfahrung auf Basis der Kartendaten. Im Testgebiet sind jedoch viele Menschen zu Fuß und auf dem Fahrrad unterwegs. Diese dynamischen Hindernisse

können nicht auf Basis der OSM-Kartendaten umfahren werden, sondern erfordern eine robuste lokale Hindernisvermeidung.

Eine Lokalisierung und Navigation des Shared Guide Dogs in der Realität war im Rahmen dieser Arbeit nicht möglich, da die Abweichungen der Sensoren zu groß waren. Unter der Annahme, dass ein Gebiet mit Sensorknoten überwacht werden kann und diese eine exakte Position an den Shared Guide Dog liefern können, erscheint die Lokalisation mit der verwendeten Hardware jedoch als möglich. Soll der Shared Guide Dog auch als Lösung, die nicht auf externe Daten zurückgreifen kann, entwickelt werden, müssen andere Systeme zur Lokalisierung in Betracht gezogen werden. Vielversprechend erscheinen Systeme mit der Ultra-Wideband (UWB)-Technologie, die lokal installiert werden und Daten zur Position liefern. Nachteile sind jedoch die Abhängigkeit von einem weiteren System und die damit verbundenen hohen Installationskosten. Eine andere Möglichkeit der Lokalisation ist die Positionsbestimmung per Triangulation. Dazu werden Hindernisse, die mit dem Lidar-Sensor erfasst werden, mit der Karte abgeglichen und so eine Position bestimmt. In [60] ist ein Ansatz vorgestellt, bei dem sich ein Fahrzeug mit einem 3D-Scanner in einer Karte lokalisieren kann. Ein ähnliches Vorgehen ist auch für das Projekt Shared Guide Dog 4.0 vorstellbar.

Literatur

[1] Oleksii Bashkanov u. a. “Exploiting OpenStreetMap-Data for Outdoor Robotic Applications”. In: *2019 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*. 2019, S. 1–7. DOI: 10.1109/ROSE.2019.8790418.

[2] Bohmeyer & Schuster, Hrsg. *Absperrbaken / Warnbaken*. 2021. URL: https://www.schilder-versand.com/warnbaken (besucht am 19. 07. 2021).

[3] Bosch Sensortec, Hrsg. *BNO055: Intelligent 9-axis absolute orientation sensor*. 2020. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf (besucht am 18. 07. 2021).

[4] cppreference.com, Hrsg. *atan2, atan2f, atan2l*. 2020. URL: https://en.cppreference.com/w/cpp/numeric/math/atan2 (besucht am 19. 07. 2021).

[5] Maximilian A. De Muirier, Tim Tiedemann und Stephan Pareigis. *Development of a Multi-Sensor System for Smart Quartier Mobility Applications*. 2021. URL: https://autosys.informatik.haw-hamburg.de/papers/2021MaximilianDeMuirier.pdf (besucht am 10. 07. 2021).

[6] David DiBiase. *The Nature of Geographic Information: An Open Geospatial Textbook*. URL: https://www.e-education.psu.edu/natureofgeoinfo/ (besucht am 21. 06. 2021).

[7] *DIN EN ISO 19157:2014-04, Geoinformation_ - Datenqualität (ISO_ 19157:2013); Englische Fassung EN_ ISO_ 19157:2013*. Berlin. DOI: 10.31030/2090178.

[8] Eberhard Karls Universität Tübingen, Hrsg. *Was ist ein GIS? "GIS makes the world go round"*. URL: https://uni-tuebingen.de/fakultaeten/mathematisch-naturwissenschaftliche-fakultaet/fachbereiche/geowissenschaften/arbeitsgruppen/geographie/institut/gis-zentrum/was-ist-ein-gis/ (besucht am 13. 07. 2021).

[9] ECMA International. *ECMA-404: The JSON Data Interchange Syntax*. Hrsg. von ECMA International. 2017. URL: https://www.ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf (besucht am 27. 06. 2021).

[10] European Union Agency for the Space Programme, Hrsg. *About EGNOS*. URL: https://egnos-user-support.essp-sas.eu/new_egnos_ops/egnos-system/about-egnos (besucht am 28. 06. 2021).

[11] EUSPA. *What is GNSS?* Hrsg. von EUSPA. 2021. URL: https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss (besucht am 17. 05. 2021).

[12] Georgios Floros, Benito van der Zander und Bastian Leibe. *OpenStreetSLAM: Global Vehicle Localization Using OpenStreetMaps*. 2013. URL: https://www.vision.rwth-aachen.de/media/papers/florosicra13.pdf (besucht am 10. 07. 2021).

[13] Tully Foote und Mike Purvis. *REP103: Standard Units of Measure and Coordinate Conventions*. 2014. URL: https://ros.org/reps/rep-0103.html (besucht am 22. 06. 2021).

[14] Brian Gerkey. *Why ROS2?* Hrsg. von Open Source Robotics Foundation, Inc. 2019. URL: https://design.ros2.org/articles/why_ros2.html (besucht am 19. 07. 2021).

[15] Routago GmbH, Hrsg. *Routago Assist*. 2021. URL: https://www.routago.de/ (besucht am 17. 07. 2021).

[16] Matthias Hentschel und Bernardo Wagner. "Autonomous robot navigation based on OpenStreetMap geodata". In: *13th International IEEE Conference on Intelligent Transportation Systems*. 2010, S. 1645–1650. DOI: 10.1109/ITSC.2010.5625092.

[17] HERE Technologies, Hrsg. *HERE Technologies: The world's #1 location platform*. 2021. URL: https://www.here.com/ (besucht am 19. 07. 2021).

[18] Joachim Hertzberg, Kai Lingemann und Andreas Nüchter. *Mobile Roboter: Eine Einführung aus Sicht der Informatik*. eXamen.press. Berlin: Springer Vieweg, 2012. ISBN: 978-3-642-01726-1.

[19] International Earth Rotation and Reference Systems Service, Hrsg. *The Earth Orientation Parameters*. 2013. URL: https://www.iers.org/IERS/EN/Science/EarthRotation/EOP.html (besucht am 28. 06. 2021).

[20] Phil Kim. *Kalman filter for beginners: With MATLAB examples*. s.l.: Createspace, 2011. ISBN: 978-1463648350.

[21] Renate Kokartis u. a. *Der Führhund als Mobilitätshilfe*. Hrsg. von Deutscher Blinden- und Sehbehindertenverband e.V. München, 2006. URL: https://www.dbsv.org/broschueren.html?file=files/ueber-dbsv/publikationen/broschueren/DBSV-Broschuere-Der-Blindenfuehrhund.pdf (besucht am 17. 07. 2021).

[22] Kroschke sign-international GmbH, Hrsg. *Parkpfosten Bonn*. 2021. URL: https://www.kroschke.com/parkpfosten-bonn--p-5644.html (besucht am 19. 07. 2021).

[23] Sven Oliver Krumke und Hartmut Noltemeier. *Graphentheoretische Konzepte und Algorithmen*. 3. Aufl. Leitfäden der Informatik. Wiesbaden: Springer Vieweg, 2012. ISBN: 978-3-8348-1849-2.

[24] Landesbetrieb Geoinformation und Vermessung, Hrsg. *Geoportal Hamburg*. 2020. URL: https://geoportal-hamburg.de/geo-online/ (besucht am 31. 05. 2021).

[25] Richard B. Langley. "Dilution of Precision". In: *GPS World* 10 (1999), S. 52–59. URL: http://www2.unb.ca/gge/Resources/gpsworld.may99.pdf (besucht am 07. 07. 2021).

[26] Steven Michael LaValle. *Planning algorithms*. Cambridge: Cambridge University Press, 2006. ISBN: 9780511546877. DOI: 10.1017/CBO9780511546877.

[27] Ina Ludwig, Angi Voss und Maike Krause-Traudes. "A Comparison of the Street Networks of Navteq and OSM in Germany". In: *Advancing Geoinformation Science for a Changing World*. Hrsg. von Stan Geertman, Wolfgang Reinhardt und Fred Toppen. Bd. 1. Lecture Notes in Geoinformation and Cartography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 65–84. ISBN: 978-3-642-19788-8. DOI: 10.1007/978-3-642-19789-5{\textunderscore}4.

[28] Steve Macenski, Carl Delsey und Ruffin White. *Navigation 2 documentation*. 2020. URL: https://navigation.ros.org/ (besucht am 30. 05. 2021).

[29] Steve Macenski, Carl Delsey und Ruffin White. *Navigation Concepts*. 2020. URL: https://navigation.ros.org/concepts/index.html (besucht am 06. 07. 2021).

[30] Steve Macenski u. a. *The Marathon 2: A Navigation System*. 1. März 2020.

[31] Maximilian Mang und Nils Schönherr. *Husky*. 2020. URL: https://autosys.informatik.haw-hamburg.de/platforms/2020husky/ (besucht am 18. 07. 2021).

[32] Wim Meeussen. *REP105: Coordinate Frames for Mobile Platforms*. 2010. URL: https://ros.org/reps/rep-0105.html (besucht am 22. 06. 2021).

[33] Hans-Joachim Mittag. *Statistik: Eine Einführung mit interaktiven Elementen*. 3., überarb. u. erw. Aufl. Lehrbuch. Berlin: Springer Spektrum, 2014. ISBN: 978-3-642-54387-6.

[34] T. Moore und D. Stouch. "A Generalized Extended Kalman Filter Implementation for the Robot Operating System". In: *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, Juli 2014.

[35] National Coordination Office for Space-Based Positioning, Navigation, and Timing. *GPS: The Global Positioning System*. Hrsg. von National Coordination Office for Space-Based Positioning, Navigation, and Timing. 2021. URL: https://www.gps.gov (besucht am 18. 05. 2021).

[36] National Geospatial-Intelligence Agency, Hrsg. *World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems*. 2014. URL: https://earth-info.nga.mil/index.php?dir=wgs84&action=wgs84 (besucht am 28. 06. 2021).

[37] Navigation 2, Hrsg. *Writing a New Planner Plugin*. 2020. URL: https://navigation.ros.org/plugin_tutorials/docs/writing_new_nav2planner_plugin.html (besucht am 18. 07. 2021).

[38] Navilock. *Navilock NL-602U USB 2.0 GPS Empfänger u-blox 6 1,5 m*. Hrsg. von Navilock. 2021. URL: https://www.navilock.de/produkt/61840/pdf.html?sprache=de (besucht am 21. 05. 2021).

[39] Open Robotics, Hrsg. *Understanding ROS 2 actions*. 2021. URL: https://docs.ros.org/en/foxy/Tutorials/Understanding-ROS2-Actions.html (besucht am 19. 07. 2021).

[40] Open Robotics, Hrsg. *Understanding ROS 2 nodes*. 2021. URL: https://docs.ros.org/en/foxy/Tutorials/Understanding-ROS2-Nodes.html (besucht am 19. 07. 2021).

[41] Open Robotics, Hrsg. *Understanding ROS 2 services*. 2021. URL: https://docs.ros.org/en/foxy/Tutorials/Services/Understanding-ROS2-Services.html (besucht am 19. 07. 2021).

[42] OpenStreetMap contributors, Hrsg. *OpenStreetMap*. 2021. URL: https://www.openstreetmap.org/#map=18/53.55532/10.02180 (besucht am 19. 07. 2021).

[43] OpenStreetMap Wiki, Hrsg. *About OpenStreetMap*. 2021. URL: https://wiki.openstreetmap.org/wiki/About_OpenStreetMap (besucht am 19. 07. 2021).

[44] OpenStreetMap Wiki, Hrsg. *Grab*. 2021. URL: https://wiki.openstreetmap.org/wiki/Grab (besucht am 19. 07. 2021).

[45] OpenStreetMap Wiki, Hrsg. *Map features*. 2021. URL: https://wiki.openstreetmap.org/wiki/Map_features (besucht am 13. 07. 2021).

[46] OpenStreetMap Wiki, Hrsg. *Mapping techniques*. 2021. URL: https://wiki.openstreetmap.org/wiki/Mapping_techniques (besucht am 18. 07. 2021).

[47] OpenStreetMap Wiki, Hrsg. *Node*. 2021. URL: https://wiki.openstreetmap.org/wiki/Node (besucht am 13. 07. 2021).

[48] OpenStreetMap Wiki, Hrsg. *Overpass API*. 2021. URL: https://wiki.openstreetmap.org/wiki/Overpass_API (besucht am 18. 07. 2021).

[49] OpenStreetMap Wiki, Hrsg. *Planet.osm*. 2021. URL: https://wiki.openstreetmap.org/wiki/Planet.osm (besucht am 18. 07. 2021).

[50] OpenStreetMap Wiki, Hrsg. *Quality assurance*. 2021. URL: https://wiki.openstreetmap.org/wiki/Quality_assurance (besucht am 31. 05. 2021).

[51] OpenStreetMap Wiki, Hrsg. *Relation*. 2021. URL: https://wiki.openstreetmap.org/wiki/Relation (besucht am 13. 07. 2021).

[52] OpenStreetMap Wiki, Hrsg. *Research*. 2021. URL: https://wiki.openstreetmap.org/wiki/Research (besucht am 19. 07. 2021).

[53] OpenStreetMap Wiki, Hrsg. *Stats*. 2021. URL: https://wiki.openstreetmap.org/wiki/Stats (besucht am 18. 07. 2021).

[54] OpenStreetMap Wiki, Hrsg. *Tags*. 2021. URL: https://wiki.openstreetmap.org/wiki/Tags (besucht am 13. 07. 2021).

[55] OpenStreetMap Wiki, Hrsg. *Way*. 2021. URL: https://wiki.openstreetmap.org/wiki/Way (besucht am 13. 07. 2021).

[56] OrCam, Hrsg. *Revolutionäres Gerät für blinde Menschen*. 2021. URL: https://www.orcam.com/de/ (besucht am 17. 07. 2021).

[57] Lothar Papula. *Mathematische Formelsammlung: Für Ingenieure und Naturwissenschaftler ; mit zahlreichen Rechenbeispielen und einer ausführlichen Integraltafel*. 11., überarb. Aufl. Wiesbaden: Springer Vieweg, 2014. ISBN: 978-3-8348-1913-0.

[58] Stephan Pareigis, Tim Tiedemann und Maximilian A. De Muirier. *Test Area Intelligent Quartier Mobility (TIQ)*. 2021. URL: https://autosys.informatik.haw-hamburg.de/project/smartmobility/ (besucht am 16. 07. 2021).

[59] Pololu Corporation, Hrsg. *VL53L1X Time-of-Flight Distance Sensor Carrier with Voltage Regulator*. URL: https://www.pololu.com/product/3415.

[60] Philipp Ruchti u. a. "Localization on OpenStreetMap data using a 3D laser scanner". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, S. 5260–5265. DOI: 10.1109/ICRA.2015.7139932.

[61] Bruno Siciliano und Oussama Khatib, Hrsg. *Springer handbook of robotics: With ... 84 tables*. Berlin: Springer, 2008. ISBN: 978-3-540-30301-5.

[62] Sick AG. *Produktdatenblatt TiM571-2050101*. Hrsg. von Sick AG. 2021. URL: https://cdn.sick.com/media/pdf/4/44/444/dataSheet_TiM571-2050101_1075091_de.pdf (besucht am 21. 06. 2021).

[63] STMicroelectronics, Hrsg. *VL53L1X: A new generation, long distance ranging Time-of-Flight sensor based on ST's FlightSense™ technology*. 2018. URL: https://www.st.com/resource/en/datasheet/vl53l1x.pdf (besucht am 08. 07. 2021).

[64] Benjamin Suger und Wolfram Burgard. "Global outer-urban navigation with OpenStreetMap". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, S. 1417–1422. DOI: 10.1109/ICRA.2017.7989169.

[65] T. Moore. *Preparing Your Data for Use with robot_ localization*. 2016. URL: http://docs.ros.org/en/melodic/api/robot_localization/html/preparing_sensor_data.html (besucht am 17. 07. 2021).

[66] Thorsten Theilig. *Sick AG Whitepaper: HDDM+ - Innovative Technologie von Sick für die Distanzmessung*. Hrsg. von Sick AG. 2017. URL: https://cdn.sick.com/media/docs/0/10/510/whitepaper_hddm_innovative_technology_for_distance_measurement_from_sick_de_im0076510.pdf (besucht am 27. 05. 2021).

[67] D. H. Titterton und J. L. Weston. *Strapdown Inertial Navigation Technology, 2nd Edition*. IEE radar, sonar, navigation, and avionics series. Stevenage: The Institution of Engineering and Technology, 2004. ISBN: 0-86341-358-7. URL: http://gbv.eblib.com/patron/FullRecord.aspx?p=432568.

[68] Trionic Sverige AB. *Trionic Veloped*. Hrsg. von Trionic Sverige AB. 2021. URL: https://www.trionic.de/de/der-gelaendegaengige-alternative-zum-outdoor-rollator-i-18 (besucht am 18. 05. 2021).

[69] u-blox AG. *u-blox 6 Receiver Description: Including Protocol Specification*. Hrsg. von u-blox AG. 2013. URL: https://www.u-blox.com/sites/default/files/products/documents/u-blox6_ReceiverDescrProtSpec_%5C%28GPS.G6-SW-10018%5C%29_Public.pdf?utm_source=en%5C%2Fimages%5C%2Fdownloads%5C%2FProduct_Docs%5C%2Fu-blox6_ReceiverDescriptionProtocolSpec_%5C%28GPS.G6-SW-10018%5C%29.pdf (besucht am 21. 05. 2021).

[70] U.S. Office of the Department of Defense, Hrsg. *Global Positioning System: Standard Positioning Service Performance Standard*. 2020. URL: https://www.gps.gov/technical/ps/2020-SPS-performance-standard.pdf (besucht am 18. 05. 2021).

[71] Jan Van Sickle. *GPS and GNSS for Geospatial Professionals*. Hrsg. von The Pennsylvania State University. URL: https://www.e-education.psu.edu/geog862/home.html (besucht am 28. 06. 2021).

[72] Was machen die da?, Hrsg. *Interview mit Heiko Kunert, Blinden- und Sehbehindertenverein Hamburg*. 2014. URL: https://wasmachendieda.de/2014-12-16/heiko-kunert-blinden-und-sehbehindertenverein-hamburg/ (besucht am 18. 07. 2021).

[73] Harald Weber. *Sick AG Whitepaper: Funktionsweise und Varianten von LiDAR-Sensoren*. Hrsg. von Sick AG. 2018. URL: https://cdn.sick.com/media/docs/5/25/425/whitepaper_lidar_de_im0079425.pdf (besucht am 19. 05. 2021).

[74] Hermann Winner, Hrsg. *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort ; mit 45 Tabellen*. 2., korrigierte Aufl. Kraftfahrzeugtechnik. Wiesbaden: Vieweg + Teubner, 2012. ISBN: 978-3-8348-1457-9.

[75] Oliver J. Woodman. *An introduction to inertial navigation*. Techn. Ber. UCAM-CL-TR-696. University of Cambridge, Computer Laboratory, Aug. 2007. DOI: 10.48456/tr-696. URL: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf.