Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Jürgen Niklas Plaggemeyer

Entwicklung eines Straßenqualitätsindikators mit Anbindung an einen Geoserver

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Jürgen Niklas Plaggemeyer

Entwicklung eines Straßenqualitätsindikators mit Anbindung an einen Geoserver

Bachelorarbeit eingereicht im Rahmen des Studiums

im Studiengang Mechatronik
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüfer: Prof. Dr. Rasmus Rettig
Zweitprüferin: Prof. Dr. Zhen Dai

Abgabedatum: 05.12.2022

Jürgen Niklas Plaggemeyer

Thema der Ausarbeitung

Entwicklung eines Straßenqualitätsindikators mit Anbindung an einen Geoserver

Stichworte

Auto, Fahrrad, ROS, TAVF, EDDY

Kurzzusammenfassung

Die Arbeit umfasst die Konzeption, Entwicklung, Aufbau und Test einer Messeinrichtung zur Bestimmung eines Straßenqualitätsindikators mittels Inertialsensorik. Hierbei wird die Straße auf ihre Rauigkeit und punktuelle Störungen (z.B. Schlaglöcher) untersucht. Es wird zwischen dem Anwendungsfall Automobil und Fahrrad unterschieden. Für das Fahrrad wird ein Messsystem entwickelt. Die Messdaten werden an einen Geoserver übermittelt und können graphisch in einer Karte dargestellt werden.

Jürgen Niklas Plaggemeyer

Title of the paper

Development of a road quality indicator with connection to a geoserver

Keywords

Car, Bike, ROS, TAVF, EDDY

Abstract

The these includes the conception, development, construction and test of a measuring system for the determination of a road quality indicator by means of inertial sensor technology. The road is examined for its roughness and punctual disturbances (e.g. potholes). A distinction is made between the automotive and bicycle use cases. A measurement system is developed for the bicycle. The measurement data are transmitted to a geoserver and can be displayed graphically in a map.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

DIN	Deutsches Institut für Normung
DoF	Freiheitsgrade (engl. degrees of freedom)
EN	europäische Norm
GNSS	Globales Navigationssatellitensystem
GPS	Global Positioning System
IMU	Inertiale Messeinheit (englisch inertial measurement unit)
IRI	International Roughness Index
ISO	Internationale Organisation für Normung
MIV	Motorisierter Individualverkehr
PKW	Personenkraftwagen
Pos	Position
RMS	Root Mean Square
ROS	Robot Operating System
WLAN	Wireless Local Area Network

1. Einführung

1.1. Motivation

Das Automobil ist das wichtigste Fortbewegungsmittel in Deutschland. Zurzeit sind etwa 48,5 Millionen Personenkraftwagen (PKW) in Deutschland zugelassen [1]. Besonders in ländlichen Regionen mit einer schlechten Anbindung an den öffentlichen Nahverkehr kann auf den motorisierten Individualverkehr (MIV) nicht verzichtet werden. Im modalen Split für ländliche Regionen hat der MIV mit über 63 % den größten Anteil des Verkehrsaufkommens [2].

In Großstädten und Metropolen wird das Auto häufiger stehen gelassen, dennoch erreicht der MIV eine Nutzung von knapp 40 %. Die Strecken werden häufiger zu Fuß oder mit dem Fahrrad zurückgelegt. Durchschnittlich werden etwa 29 km pro Tag mit dem MIV zurückgelegt [2].

Des Weiteren wurde die Warenlagerung durch die Just-in-Time Lieferungen erfolgreich auf die Straße verlegt. Dieses bedeutet ein massiv höheres Aufkommen an Lastkraftwagen. Bei dieser Nutzung der Straßen bleiben Schäden nicht aus. Daher sind nur 11,7 % der Autobahnen und 5,7 % der Bundesstraßen in Deutschland in einem sehr guten Zustand, insbesondere mit Auswirkungen auf den Fahrkomfort und die Fahrsicherheit [3].

Straßen mit Beschädigungen und schlechter Oberflächenbeschaffenheit gehen nicht nur zu Lasten des Fahrkomforts. Auch nimmt der Verschleiß an Achsen und Fahrwerk durch die erhöhte Beanspruchung zu. Personenkraftfahrzeuge können durch ihr Fahrwerk Unebenheiten in der Straße gut dämpfen und somit den Komfort der Insassen gewährleisten. Anders sieht es bei Fahrrädern aus. Etwa 62 % aller genutzten Fahrräder in Deutschland sind herkömmliche Trecking- oder Cityfahrräder [4]. Diese besitzen entweder keine oder nur eine Gabelfederung. In den meisten Fällen sind diese mit einer Sattelfederung gepaart. Schlechte Straßen und Fahrradwege haben bei Fahrradfahrern nicht nur einen negativen Einfluss auf den Komfort, es kostet auch mehr Kraft diese zu befahren im Gegensatz zu einer unbeschädigten Straße.

Durch die Kartografierung eines Straßenqualitätsindikators wäre es möglich, die Bodenbeschaffenheit in die Navigation und Planung der Fahrstrecke mit einzubeziehen.

Besonders für Fahrradfahrer ist eine solche Karte sinnvoll, da schlechte Straßen oft den Komfort am alltäglichen Fahrradfahren beeinträchtigen und die Unfallgefahr erhöhen.
In der Verbindung mit Wetterdaten können Senken oder Spurrillen erkannt werden, in denen sich bei Regen Wasser sammelt Welches zu Aquaplaning führt. Dieses kann im Navigationssystem integriert werden, um auf diese Gefahrenstellen hinzuweisen.
Die Daten zur Straßenqualität sind auch wichtig für die Instandhaltung der Infrastruktur. Durch das Überprüfen der Straßen in bestimmten zeitlichen Abständen können Veränderungen erfasst werden. Dieses könnte die Grundlage für einen Indikator bilden, wann eine Straße erneuert werden muss.

1.2. Aufgabenstellung

Basierend auf einer Recherche von wissenschaftlicher Literatur und Vorarbeiten ist ein Indikator zu definieren, der eine quantitative Beschreibung des Zustands der Straßenqualität erlaubt. Zu berücksichtigen ist einerseits eine integrale Beschreibung von Bereichen (z.B. Rauigkeit) und andererseits eine Beschreibung von punktuellen Störungen (z.B. Schlaglöcher). Die entwickelte Systematik ist anzuwenden auf zwei Anwendungsfälle:

1. Testfahrzeug des Urban Mobility Labs mit der Inertialsensorik im SBG-Apogee [5]
2. Anwendungsfall eines Fahrrads. Hierfür ist ein Messystem zu konzipieren und umzusetzen. Der erstellte Indikator soll als Robot Operating System (ROS) Node in einer ROS-Umgebung arbeiten.

Das System ist an einen Geoserver (www.geoserver.org) anzubinden und die gemessene Information zu übermitteln. Eine Abfrage der ermittelten Informationen mit Darstellung in einer Karte ist zu implementieren.

1.3. Ziel und Struktur der Arbeit

Ziel dieser Arbeit ist es ein System zu entwickeln welches die Straßenqualität bewertet und Events finden kann. Dieses soll anhand von einer Inertialsensorik geschehen.
Kapitel 2: Durch eine Literaturrecherche wird sich mit bereits vorhandenen Systemen vertraut gemacht.
Kapitel 3: Es werden die Anforderungen definiert.
Kapitel 4: Durch die Erstellungen geeigneter Konzepte werden die Anforderungen abgedeckt. Diese beinhalten den Algorithmus, welcher die eigentliche Bewertung durchführt.
Kapitel 5: Die Umsetzung und der Konzepte bezüglich des Aufbaues der Hardware und Software wird beschrieben.
Kapitel 6: Für die Verwendung eines Algorithmus werden verschiedene Paramater für den Einsatz im Messystems des Testfahrzeuges und des Fahrrades bestimmt.
Kapitel 7: Durch verschieden Tests wird das Messystem geprüft.
Kapitel 8: Es werden alle Ergebnisse zusammengefasst und die ein Ausblick auf Zukünftiges gegeben.

2. Grundlagen

In diesem Kapitel werden die Dynamiken von Automobilen und Fahrrädern zur Erstellung eines Straßenqualitätsindikator beschrieben. Es wird auf die Beschaffenheiten von Straßen und Wege eingegangen und welche punktuellen Störungen auftreten können. Es wird eine Übersicht der bereits bestehenden Techniken zur Straßenqualitätsmessung erstellt.

2.1. Dynamik von Automobilen und Fahrrädern

Automobile und Fahrräder haben eine Vielzahl von Dynamiken. Besonders bei extremen Fahrmanövern entstehen Effekte, die bei einer normalen Fahrt nicht auftauchen. Diese Sonderfälle werden nicht betrachtet. Das Fahrwerk ist sehr entscheiden für das Fahrverhalten. Es lässt sich vereinfacht mit dem „Quarter-Car Model“ beschrieben.

2.1.1 Automobile

Ein Automobil besitzt sechs Freiheitsgrade. Dieses sind drei translatorische und drei rotatorische Bewegungsmöglichkeiten. Diese werden nach DIN ISO 8855 bezeichnet und sind in Abbildung 1 dargestellt.

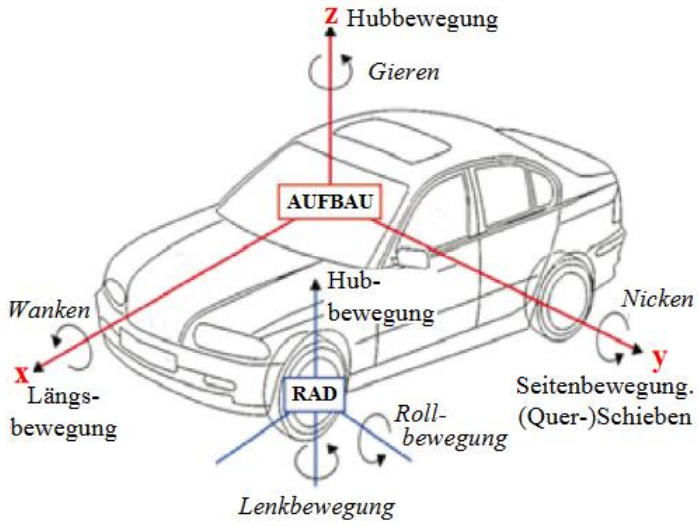
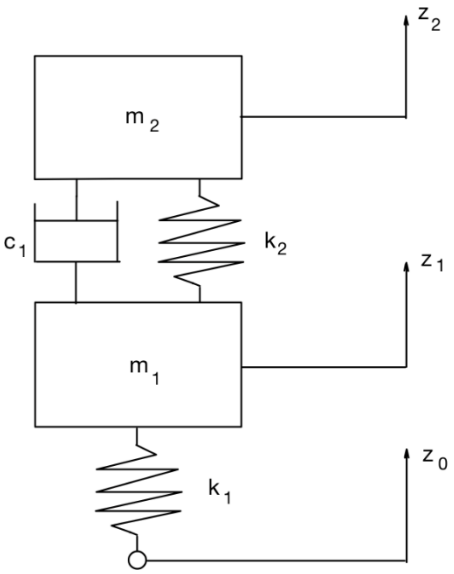


Abbildung 1: Bewegungsmöglichkeiten nach DIN ISO 8855 [6]

Fährt ein Fahrzeug auf einer idealen glatten Straße mit konstanter Geschwindigkeit grade aus, ohne Berücksichtigung des Luftwiderstandes, so wirkt nur die Erdanziehungskraft auf das Fahrzeug. Nun taucht auf dieser idealen Straße eine kurze Erhöhung über die gesamte Breite der Fahrbahn auf. In dem Moment des Überfahrens beginnt das Auto zu nicken. Gleiches gilt auch für Vertiefungen über die gesamte Fahrbahnbreite. Bei einer Erhöhung auf dieser idealen Straße, die nur auf einer Hälfte der befahrenen Straße auftritt, beginnt das Fahrzeug zu wanken. Das Gieren des Fahrzeuges entsteht in Kurvenfahrten.
In der Realität treten alle drei Fahreigenschaft überlagert auf. Diese werden durch das Höhenprofil der Straße hervorgehoben. Die Hubbewegung des Rades wird über das Fahrwerk an die Karosserie übertragen.
Ein Fahrwerk besteht aus einem Feder- und Dämpferpaket. Die Feder kann dabei eine Stahlfeder oder ein Federbalg sein. Mit dem Quarter-Car Modell ist es möglich das Federverhalten eines Fahrzeuges stark vereinfacht darzustellen [5]. In Abbildung 2 ist das Quarter-Car Modell zu sehen. Dieses ist der vereinfachte Aufbau eines Reifens und der Karosserie welches aus zwei quadratischen Körpern bestehen. Mit diesem Modell wird nur ein Reifen des Fahrzeuges beschrieben. Die Dämpfungs- und Federrate des Fahrwerkes werden mit c_1 und k_2 beschrieben. Die Federrate des Reifens ist k_2. Grundlegend gilt:

$$m_2 \cdot \ddot{z}_2 = k_2 \cdot (z_1 - z_2) + c_1 \cdot (\dot{z}_1 - \dot{z}_2)$$
$$m_1 \cdot \ddot{z}_1 = k_1 \cdot (z_0 - z_1) - k_2 \cdot (z_1 - z_2) - c_1 \cdot (\dot{z}_1 - \dot{z}_2)$$



z_0	Änderung des Straßenprofils
z_1	Verschiebung des Reifens
z_2	Verschiebung der Karosserie
m_1	Masse des Reifens
m_2	Masse der Karosserie
c_1	Dämpfungsrate des Fahrwerkes
k_1	Federrate des Reifens
k_2	Federrate des Fahrwerkes

Abbildung 2: Quarter-Car Modell [13]

In der Realität verhält sich das Fahrzeug anders, da weitere Faktoren das Quarter-Car Modell des Fahrzeuges beeinflussen. Die Masse des Fahrzeuges ist abhängig von der Beladung bzw. der Anzahl an Insassen. Zudem erzeugt die Aerodynamik des Fahrzeuges, in Abhängigkeit von

der Geschwindigkeit, einen Anpressdruck. Die Federrate des Reifens ist abhängig vom Luftdruck, der sich durch die Temperatur des Reifens verändern kann.

2.1.2 Fahrräder

Das Koordinatensystem für Fahrräder hat die gleiche Orientierung wie das Koordinatensystem für Automobile (Abbildung 3). Ebenso kann ein Fahrrad gieren und nicken. Ist der Koordinatenursprung in der Mitte des Fahrrades, so ist ein Wanken nicht möglich. Das Fahrrad kann sich nur zur Seite neigen. Dabei befindet sich die Drehachse zwischen Reifen und Straßenoberfläche. Das Fahrrad neigt sich bei Kurvenfahrten (Dynamisches Gleichgewicht) und bei starkem Treten in die Pedale.

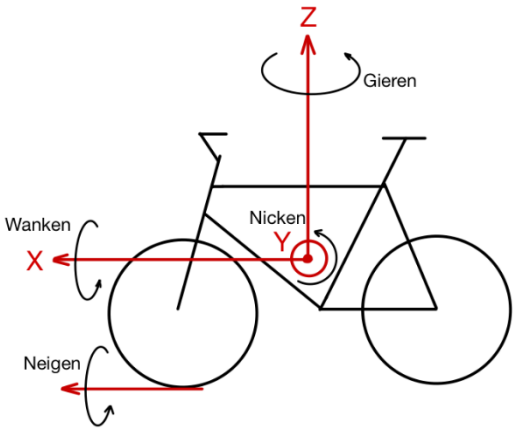
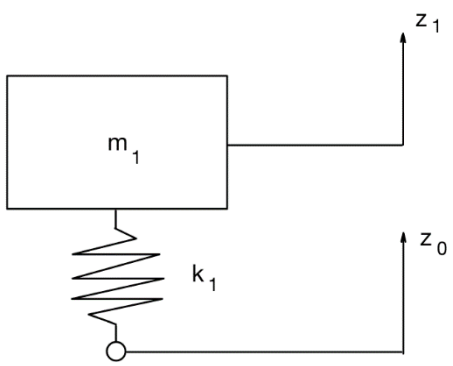


Abbildung 3: Bewegungsmöglichkeiten eines Fahrrades

Die meisten Fahrräder besitzen heutzutage ebenfalls ein Dämpfer-Feder-Paket. Diese lässt sich ebenfalls mit dem Quarter-Car Modell beschrieben (Abbildung 2). Fahrräder ohne Federung eigenen sich besonders gut für die Ermittlung eines Straßenqualitätsindikators, da das Höhenprofil der Straße nahezu direkt an den Rahmen übertragen wird. Das Quarter-Car Modell eines Fahrrades besteht daher nur aus einer Masse und aus der Federkonstante des Reifens. Hierbei gilt:

$$m_1 \cdot \ddot{z}_1 = k_1 \cdot (z_0 - z_1)$$



z_0 Änderung des Straßenprofils
z_1 Verschiebung des Rahmens
m_1 Masse des Rahmens
k_1 Federrate des Reifens

Abbildung 4: „Quarter-Fahrrad“ Modell eines Fahrrades ohne Federung

2.2. Straßen und Wege

Straßen und Wege haben unterschiedliche Beschaffenheiten und Vorgaben je nach Verwendungszweck. Für die Straßenqualität ist das Höhenprofil, die Rauigkeit und die Neigung der Straße ausschlaggebend. In Deutschland sind etwa 95 % der öffentlichen Fahrbahndecken asphaltiert [6]. Der Rest teilt sich auf in betonierte oder gepflasterte Fahrbahndecken. Ein sehr geringer Teil der öffentlichen Straßen besteht aus Sand- oder Schotterwegen.

2.2.1 Asphaltstraßen

In Abhängigkeit von der Einordnung der Straße (Autobahn, Bund- oder Landesstraße, etc.) und der Belastung gelten unterschiedliche Vorschriften. Diese Vorschriften reichen von den Abmaßen der Straße über den Frostschutz bis hin zur Tragfähigkeit.

Da die TAVF nur im Innenstadtbereich liegt sind die Richtlinien nach RASt 06 in Tabelle 1 [7].

Grundlegend ist eine Straße in verschiedenen Schichten aufgebaut. In Abbildung 5 ist das Schichtenmodell einer schwerbelasteten Straße zu sehen. Die Dicke und Anzahl der unterschiedlichen Schichten sind abhängig von dem Einsatzgebiet der Straße. Für die Straßenqualität ist im Wesentlichen die Deckschicht verantwortlich. Die Toleranzen der Deckschicht ist abhängig vom Untergrund und der verwendeten Asphaltart. In Tabelle 2 ist eine Übersicht der Toleranzen dargestellt.

Bezeichnung	**Wert**
höchste Längsneigung	6 %
höchste Querneigung	2,5 %
an Überquerungen, Bordsteinsenkungen	≤ 3 cm

Tabelle 1: Toleranzen von neugebauten Stadtstraßen [7]

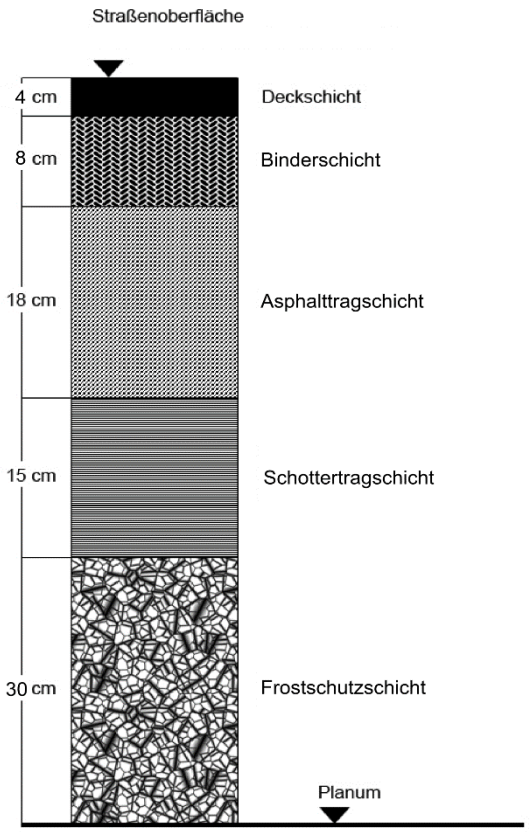


Abbildung 5: Schichtenmodell einer schwerbelasteten Straße nach RStO 01 [9]

Offenporiger Asphalt wird auch als Flüsterasphalt bezeichnet. Durch die hohe Anzahl an Hohlräumen wird weniger Luft zwischen Straße und Reifen verdichtet. So wird eine Lärmpegel--Reduzierung von etwa 10 dB erreicht [8].
Durch das Benutzen der Fahrbahn verschleißt die Oberfläche der Asphaltdecke, wodurch die Güte der Oberfläche verändert wird. Die hier genannten Toleranzen beziehen sich auf eine neu gebaute Straße.

Asphaltdeckschicht Typ	Hohlraumgehalt	Unebenheit bei maschinellem Einbau
Asphaltbeton	max. 5,5 Vol.-%	max. 6 mm/4m (auf Asphalttragschicht) max. 4 mm/4m (auf Asphaltbinderschicht)
Splittmastixasphalt	max. 5,0 Vol.-%	max.6 mm/4m (auf Asphalttragschicht) max. 4 mm/4m (auf Asphaltbinderschicht)
Gussasphalt	----------	max.6 mm/4m (auf Asphalttragschicht) max. 4 mm/4m (auf Asphaltbinderschicht)
Offenporigem Asphalt	min. 22 Vol.-% max. 28 Vol.-%	---------- max. 3 mm/4m (auf Asphaltbinderschicht)

Tabelle 2:Asphaltdeckschichten nach ZTV SoB-StB [7]

2.2.2 Gepflasterte Straßen

In Deutschland werden verschiedene Materialien für gepflasterte Straßen eingesetzt. Diese sind z.B.: Platten aus Beton, Pflasterziegel, spaltraum Naturstein, bearbeiteter und unbearbeiteter Naturstein.
Der Aufbau einer Pflasterstraße ist in Abbildung 6 dargestellt.

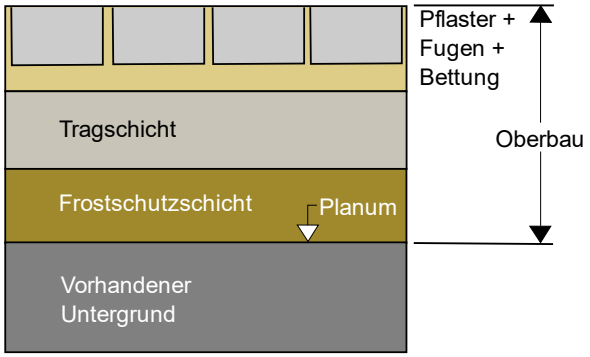


Abbildung 6: Aufbau einer Pflasterstraße [33]

Maßgeblich für die Oberflächenqualität der Deckschicht, ist die verwendete Art der Steine und der Pflasterverband. Dieses ist die Art in welchem Muster die Steine verlegt werden. Abhängig von Fahrtrichtung und Pflasterverband werden mehr oder weniger Fugen überfahren. Einzelne. Pflasterdecken für Straßen und Flächen werden in DIN 18318 geregelt. Die Anforderungen an die Toleranzen der Oberflächen sind in Tabelle 3 dargestellt. Diese Anforderungen gelten für neu gebaute Straßen.

Bezeichnung	Toleranz
Querneigung befahrbarer Straßen	2,0 - 2,5 %
Höhenversatz zwischen Betonsteinen	2 mm
Höhenversatz zwischen Natursteinen	5 mm

Tabelle 3: Toleranzen nach DIN 18318

2.2.3 Punktuelle Störungen

Maßgeblich für die Straßenqualität ist auch die Anzahl an Störquellen auf der Fahrbahn. Dieses sind z.B. Schlaglöcher, Kanalschachtabdeckungen, Bahnübergänge, Schwellen zur Verkehrsberuhigung, abgesenkte Bordsteine oder Dehnungsfugen an Brücken.
Schlaglöcher haben dabei den größten Effekt auf die Straßenqualität, da diese die größten Beschleunigungen im Fahrzeug hervorrufen. Kanalschachtabdeckungen etc. sind so konzipiert, dass sie in Straßenoberfläche eingelassen werden.

2.3. International Roughness Index

Der International Roughness Index (IRI) ist eine weitverbreitete Methode, um die Qualität einer Straßenoberfläche zu bestimmen. Dabei wir die Oberfläche der Straße als ein 2D-Profil angenommen. Der IRI gibt die Summe aus Aufwärtsbewegungen eines Reifens für eine bestimmte Teilstrecke an. Meistens wird der IRI in m/km oder in/mi (Inch pro Meile) angegeben. Die zu messende Teilstrecke ist dabei meistens 10 m oder 20 m lang.
Früher wurde der IRI über Messlatten bestimmt, die auf die Straße gelegt wurden. Heutzutage kann das Profil über optische Sensoren oder Inertailsensoren bestimmt werden. Bei der Verwendung von Inertailsensoren gibt es verschieden Positionen zur Befestigung von diesen. In Bezug auf das Quarter-Car Modell kann die Inertialsensorik an den verschiedenen „Quadraten“ befestigt werden. In den meisten Fällen wird eine Inertialsensorik im Inneren des Fahrzeugs befestigt. Durch die Verwendung des Quarter-Car Modell wird dann auf das Höhenprofil der Straße zurückgerechnet und der IRI bestimmt.
Der IRI kann auch durch verschieden Näherungsfunktionen bestimmt werden. Hierfür dient der Root Mean Square (RMS) als Basis. [10] [11]

2.4. Stand der Technik

Zur Messung der Straßenqualität bzw. zum genauen Aufzeichnen der Straßenoberfläche gibt es zwei Grundprinzipen. Die Vermaßung der Straße mithilfe von optischen Sensoren und die Aufzeichnung und Auswertung von Inertialsensoren aus dem Fahrzeug. Die optischen Sensoren liefern eine vollständigere Bewertung, da ein Großteil der gesamten Fahrbahnoberfläche gemessen wird.

2.4.1 Optische Sensoren

Durch den Einsatz von optischen Sensoren kann, dass Straßenprofil genau nachgebildet werden. Ein Beispiel für eine solches System ist der Pavement Profile Scanner (PPS) des Fraunhofer-Instituts für Physikalische Messtechnik [9]. Hierbei wird die Straße auf einer Breite von etwa 4 m durch einen LIDAR abgetastet, welcher sich am Heck des Messfahrzeuges befindet. Es können, je nach Auflösung, Straßenunebenheiten bis zu 0,2 mm erkannt werden. In Verbindung mit einer Kamera können sogar Risse im Asphalt erkannt werden. Die Daten werden vollautomatisch durch ein Deep Learning Framework mit

automatisierter Interpretation von 2D- und 3D-Daten ausgewertet. Für dieses Messsystem ist allerding ein Aufbau notwendig, welcher über die Länge des Fahrzeuges hinausreicht.
Ähnliche Systeme werden von dem Bundesamt für Straßenwesen verwendet. Das Erfassungssystem zur Fahrbahnoberflächenanalyse (EFA) [10] bestimmt die Querebenheit mit 41 Lasertriangulationssensoren und die Längsebenheit durch 5 Lasertriangulationssensoren. Die Fahrbahn wird auf 4 m Breite abgetastet. Unabhängig von der gefahrenen Geschwindigkeit wird alle 0,1 m eine Messung durchgeführt. Für die Auswertung wird die Fahrbahn in 1 m Segmente unterteilt. Das EFA untersucht, neben der Neigung, folgende Aspekte: allgemeine Unebenheit, Spurrinnentiefe und fiktive Wassertiefe.

2.4.2 Inertialsensoren

Bei der Nutzung von Inertialsensorik ist die Beschleunigung des Fahrzuges die ausschlaggebende Größe. Die Reifen sind die einzigen Berührungspunkte zur Straße. Somit wird nur ein geringer Teil der Straße in die Messung einbezogen. In der Regel werden Inertialsensorik und GPS-Modul zur Datenerfassung genutzt. Bei den Ansätzen wie diese Daten verarbeiten werden, gibt es verschiedene Herangehensweisen. Dieses ist entweder einfach Algorithmen oder maschinelles Lernen. In Tabelle 4 werden verschiedene Literaturen und deren Herangehensweise zur Datenverarbeitung aufgelistet.

Referenz	**Sensorik**	**Datenverarbeitung**
[11]	Inertialsensorik und GPS-Position von Smartphones und Tablets	**Straßenqualität:** Näherungsfunktion an den International Roughness Index (IRI) **Schlagloch Erkennung:** Keine
[14]	Inertialsensor um z_1 und z_2 des Quarter-Car Modells zu bestimmen. GPS-Modul.	**Straßenqualität:** IRI **Schlagloch Erkennung:** Keine
[10]	Inertialsensor und GPS-Modul im Fahrzeug.	**Straßenqualität:** Näherungsfunktion an den IRI **Schlagloch Erkennung:** Algorithmus definierter geschwindigkeits-abhängigen Schwellwert
[15]	Inertialsensor und GPS-Modul im Fahrzeug.	**Straßenqualität:** Keine **Schlagloch Erkennung:** Maschinelles Lernen zur Erkennung von unterschiedlichen Ereignissen

Tabelle 4: Bestehende Systeme Teil 1

Referenz	Sensorik	Datenverarbeitung
[16]	Inertialsensorik und GPS-Position von Smartphones	**Straßenqualität:** Keine **Schlagloch Erkennung:** K-Means-Algorithmus und Support Vector Machine zur Erkennung von Schlaglöchern
[17]	Intertialsensorik und GPS-Position von Smartphones	**Straßenqualität:** Algorithmus abhängig von der Magnitude IRI als Klassifikator **Schlagloch Erkennung:** Keine
[18]	Intertialsensorik und GPS-Position von Smartphones im Fahrzeug	**Straßenqualität:** Algorithmus mit maschinellem Lernen im Zeitbereich, Frequenzbereich oder durch Wavelet-Transformation **Schlagloch Erkennung:** Algorithmus mit maschinellem Lernen im Zeitbereich, Frequenzbereich oder durch Wavelet-Transformation

Tabelle 5 Bestehende Systeme Teil 2

3. Anforderungsentwicklung

Die Anforderungsentwicklung teilt sich auf in Anforderungen an die Software, die Hardware und den Algorithmus. Die Software beschreibt die grundlegende Umgebung, in der der Algorithmus betrieben werden soll. Da ein ROS-System zum Einsatz kommen soll, kann im Tesla und in der Messeinrichtung fürs Fahrrad auf die gleiche Software zurückgegriffen werden. Der Algorithmus ist für die Ermittlung der Straßenqualität und Eventfindung zuständig und soll sowohl für den Tesla als auch für das Fahrrad funktionieren. Die Hardware für den Tesla ist bereits vorhanden. Die Hardware für die Messeinrichtung muss komplett konzipiert werden. In folgender Tabelle ist eine Übersicht der zu konzipierender Bereiche:

Bereich	**Tesla**	**Fahrrad**
Software	Wird konzipiert	Wird konzipiert
Algorithmus	Wird konzipiert	Wird konzipiert
Hardware	Vorhanden	Wird konzipiert

Tabelle 6: Übersicht zu konzipierender Bereiche

3.1. Software

Für die Software wurden die nachstehend Anforderungen festgelegt. Dabei soll noch Möglichkeit die gleiche Software im Tesla und im Fahrrad eingesetzt werden. Im Tesla wird zurzeit Ubuntu 16.04 eingesetzt. Ein späteres Update auf Ubuntu 20.04 soll mitberücksichtigt werden.

Pos.	**Beschreibung der Anforderung**
1	Verwendung von ROS
2	Hochladen der Daten an einem Geoserver
3	Kompatibel für Ubuntu 20.04 und 16.04
4	Einfache Bedienung des Systems ermöglich (Starten, Stoppen etc.)

Tabelle 7: Anforderungen an das System im Testfahrzeug

3.2. Algorithmus

Der Algorithmus zur Errechnung der Straßenqualität soll dabei sowohl auf dem System im Auto als auch auf dem Fahrrad funktionieren. Für Algorithmus und Events wurden folge Anforderungen festgelegt:

Pos.	Beschreibung der Anforderung
1	Ermittlung der Straßenqualität
2	Ermittlung von punktuellen Störungen (Schlaglöcher etc.)
3	Geschwindigkeitsabhängige Verarbeitung der Daten
4	Erkennen und Herausfiltern von Fahrszenarien die einen Einfluss auf die Beschleunigungswerte unabhängig von der Straßenqualität haben
5	Identischer Algorithmus im Testfahrzeug und für das Fahrrad
6	Ortsabhängigkeit der Daten herstellen
7	Grundlegend gleicher Algorithmus im Testfahrzeug und für das Fahrrad

Tabelle 8: Anforderung an den Algorithmus der Messeinrichtung

3.3. Hardware im Testfahrzeug

Für die Hardware im Testfahrzeug wurden folgende Anforderungen festgelegt:

Pos.	Beschreibung der Anforderung
1	Verwendung des SBG-Apogee D
2	Verwendung der vorhandenen Hardware im Fahrzeug

Tabelle 9: Anforderungen an die Hardware im Testfahrzeug

3.4. Hardware für die Messeinrichtung im Fahrrad

Für die Hardware der Messeinrichtung für Fahrräder wurden folge Anforderungen festgelegt:

Pos.	Beschreibung der Anforderung
1	Geeignete Inertialsensorik
2	Geeignete GSN-Sensorik
3	Staub und wasserdichtes Gehäuse
4	Akkulaufzeit von min 5 h
5	Erschütterungsfester Aufbau
6	Minimale Größe und minimales Gewicht
7	Flexible einsetzbar an mehreren Fahrrädern
8	Zerstörungsfreie Montage und Demontage innerhalb von 15 min
9	Nach Möglichkeit Verwendung von bereitgestellter Hardware

Tabelle 10: Anforderung an die Hardware der Messeinrichtung

4. Konzeption und Architektur

Diese Kapitel beinhaltet die Konzeption und Architektur für die Software. den Algorithmus und die Hardware für den Einsatz am Fahrrad. Da das als Anforderung eine ROS-Umgebung gefordert wird, lässt sich eine allgemeine Konzeption der Software für mehrere Systeme erstellen. Der Algorithmus wird so konzipiert, dass dieser auf unterschiedlichen Fahrzeugen funktioniert. Die Adaptivität auf verschiedene Fahrzeuge wird durch die Abhängigkeit des Algorithmus von fahrzeugspezifischen Parametern hergestellt. Die Wahl der Hardwarekomponenten für die Messeinrichtung am Fahrrad wird aus zur Verfügung gestellten Komponenten ausgewählt.

4.1. Software

Die zu konzipierende Software bezieht sich auf eine ROS-Umgebung, welche unter verschiedenen Ubuntu-Versionen (16.04 und 20.04) lauffähig sein soll. In Abbildung 7 wird ein Blockdiagramm der Software dargestellt. In dem System können ein oder mehrere Sensoren mit entsprechendem Treiber zum Einsatz kommen. Dieses ist abhängig von der Auswahl der Hardware. Der Algorithmus benötigt die Beschleunigungsdaten des Fahrzeuges, welche im folgenden Text als IMU-Data bezeichnet werden und die GPS-Position und GPS-Geschwindigkeit, welche im folgenden Text als GNSS-Data bezeichnet werden. Damit der Algorithmus möglichst unabhängig von der Auswahl der Sensoren funktioniert, wird ein Adapter-Modul eingeführt. Dieser Adapter kann die Daten von mehreren Sensoren vereinen und bringt diese in das Format, welches vom Algorithmus verwendet wird. Sollte ein Sensortreiber bereits Daten im richtigen Format liefern, wird der Adapter nicht benötigt. Der Algorithmus ist die zu entwickelnde Softwarekomponente. Dieser ermittelt durch die Geschwindigkeits-, Positions- und Beschleunigungsdaten die Straßenqualität und findet Events. Die Schnittstelle dient zum Hochladen von ROS-Nachrichten an den EDDY-Geoserver. Diese Schnittstelle kann auch Daten speichern, falls keine Internetverbindung besteht und bei erneuter Internetverbindung hochladen.

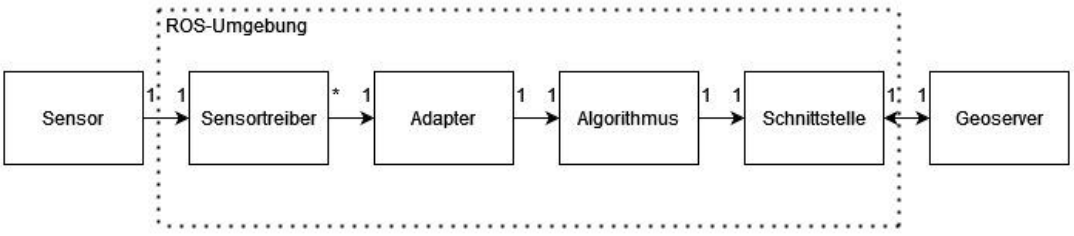


Abbildung 7: Software Blockdiagram

4.2. Algorithmus

Der Algorithmus zur Ermittlung der Straßenqualität und zum Auffinden von Events wird sowohl im Testfahrzeug als auch in der Messeinrichtung fürs Fahrrad eingesetzt. Hierfür werden verschiedene Parameter und Funktionen für das verwendet Fortbewegungsmittel bestimmt. Die Parameter sind abhängig von der verwendeten Hardware und Aufbau und Eigenschaften des Fortbewegungsmittels. Besonders wichtig hierbei ist die Federung des Fortbewegungsmittels. Somit würde sich der Algorithmus auch auf andere Fahrzeuge oder Fahrräder anwenden lassen, wenn die entsprechenden Parameter ermittelt werden.
Zur Klassifizierung wird die befahrene Strecke in Teilstücke unterteilt. Diese besitzen die Länge $l_{Teilstrecke}$. Innerhalb dieser Teilstrecke wird anhand der Beschleunigungen in Z-Richtung die Straßenqualität berechnet und Events ermittelt. Die Geschwindigkeit wird zum Prüfen der Teilstrecken und Skalieren der Straßenqualität benötigt. Über die GPS-Position wird die Ortsabhängigkeit der Straßenqualität ermittelt.
Damit die Straßenqualität beurteilt werden, kann müssen einige Fahrszenarien erkannt und herausgefiltert werden. Diese Fahrszenarien haben einen Einfluss auf die Beschleunigungswerte unabhängig von der Straßenqualität. Teilstücke, bei denen diese Fahrszenarien erkannt wurden, werden nicht bei der Straßenqualitätsberechnung und Eventfindung berücksichtigt.
Grundsätzlich werden alle Beschleunigungswerte im Fahrzeug durch den Untergrund beeinflusst. Hauptsächlich spiegelt sich die Straßenqualität in der Beschleunigung der Z-Achse wider. Je nach Neigung der Straße und des Fahrzeuges wird die Erdbeschleunigung unterschiedlich auf die einzelnen Beschleunigungen im Fahrzeug aufgeteilt. Je Teilstrecke wird daher die Erdbeschleunigung ermittelt, welche auf die Z-Achse wirkt. Die Straßenqualität durch den RMS (englisch: root mean square) der Z-Beschleunigung bestimmt. Events werden über den Ruck der Gesamtbeschleunigung ermittelt.

4.2.1 Daten sammeln und Wertepaare bilden

Für den Algorithmus relevant sind die IMU-Daten, welchen die Beschleunigung in Z-Richtung enthalten und die GNSS-Daten, welche die GPS-Position und die die GPS-Geschwindigkeiten enthalten. Das Datensammeln beginnt mit dem Bilden von Wertepaaren der Messgrößen. Diese werden meistens mit unterschiedlichen Frequenzen gemessen. Auch bei gleicher Messfrequenz könne die Datensätze der Messgrößen asynchron beim Algorithmus ankommen.
Es werden Wertepaare der eingehenden Datensätze gebildet. Dabei wird immer dann ein neues Wertpaar erzeugt, wenn ein neuer IMU-Datensatz beim Algorithmus ankommt. In dem Wertepaar wird nur der Datensatz erneuert, der unterschiedlich zum vorherigen Datensatz ist. Über das Verhältnis der Messfrequenzen lässt sich ermitteln, wie viele Wertepaare zu einem Datensatz mit der niedrigeren Messfrequenz gebildet werden:

$$n_{Wertepaare} = \frac{f_{hoch}}{f_{niedrig}}$$

Nachdem die Messdaten durch synchronisiert wurden, wird noch der Sensor-Offset von den zu benutzenden Messgrößen entfernt. Der Sensor-Offset wird nicht im Algorithmus bestimmt. Dieses muss vorher ermittelt werden.

4.2.2 Teilstrecke bilden

Die Daten werden so lange gesammelt bis die gefahrene Strecke, anhand der GNSS-Koordinaten, in eine Teilstrecke mit der Länge $l_{Teilstrecke}$ unterteilt werden kann. Die Berechnung der Straßenqualität und die Eventfindung wird nur für die jeweilige Teilstrecke durchgeführt. Es gibt einen Bereich zwischen Minimalgeschwindigkeit V_{Min} und Maximalgeschwindigkeit in dem Daten gesammelt werden. Die Minimalgeschwindigkeit wird benötigt, da der Algorithmus eine Bewegung des Fahrzeuges voraussetzt. Oberhalb der Maximalgeschwindigkeit kann eine zuverlässige Funktion des Algorithmus nicht mehr gewährleistet werden.
In den GNSS-Daten wird die Fahrzeuggeschwindigkeit als Vector in Nordrichtung, Ostrichtung und Höhenrichtung übermittelt. Die aktuelle Geschwindigkeit des Fahrzeuges wird aus dem Betrag der einzelnen Geschwindigkeiten gebildet. Hierbei reicht es, nur die Geschwindigkeiten in Nord- und Ostrichtung zu verwenden:

$$V_{Fahrzeug} = \sqrt{{V_{Nord}}^2 + {V_{Ost}}^2}$$

Zur Berechnung der Länge der Teilstücke werden die Positionsangaben in den GNNS-Daten genutzt. Bei der Längenberechnung der Teilstücke $l_{Teilstrecke}$ ist der Abstand der Breitengrade zu beachten. Dieser ist am Äquator am größten und wir zu den Polen kleiner [16]. Der Abstand zwischen zwei GNSS-Koordinaten mit $P_1 = (Lat_1|Lon_1)$ und $P_2 = (Lat_2|Lon_2)$ in Meter wird näherungsweise wie folgt berechnet [20]:

$$l_x = 111{,}3\ km \cdot \cos\left(0{,}01745 \cdot \frac{Lat_1 + Lat_2}{2}\right) \cdot (Lon_1 - Lon_2)$$
$$l_y = 111{,}3\ km \cdot (Lat_1 - Lat_2)$$
$$l_{P12} = \sqrt{{l_x}^2 + {l_y}^2} \cdot 1000\ m$$

Die Länge der Teilstrecke wird durch den ersten und letzten Punkt innerhalb der Teilstrecke bestimmt. Der Sollwert der Länge der Teilstrecke soll dabei $l_{Teilstrecke}$ sein. Abhängig von der Fahrzeuggeschwindigkeit und der Messfrequenz der GNSS-Daten ergibt sich eine effektive Länge der Teilstrecke $l_{Teilstrecke\ Effekitv}$. Damit fehlerhafte GNSS-Daten keine zu langen Teilstrecken erzeugen, muss folgende Bedingung erfüllt sein:

$$l_{Teilstrecke} \leq l_{Teilstrecke\ Effekitv} \leq l_{Teilstrecke_{Maximal}}$$

Die maximale Länge der Teilstrecke ist dabei an die verwendeten Sensoren und Messbedingungen anzupassen.
Die Teilstrecke ist der Messbereich, der von dem Algorithmus untersucht wird. Die Teilstrecke beinhaltet die einzelnen GNSS und IMU-Daten für die Länge der Teilstrecke. Die Anzahl der GNSS-Datensätze $n_{GNSS\,Data}$ und der IMU-Datensätze $n_{Imu\,Data}$ innerhalb der Teilstücke ist variiert und abhängig von der gefahrenen Geschwindigkeit. Je langsamer gefahren wurde, je mehr Datensätze können innerhalb der Teilstrecke gesammelt werden.

4.2.3 Geschwindigkeit prüfen

Es wird überprüft, ob innerhalb der Teilstücke eine konstante Geschwindigkeit gefahren wurde. Bei einer konstanten Geschwindigkeit innerhalb der Teilstrecke wurde das Fahrzeug innerhalb der Teilstrecke nicht beschleunigt oder abgebremst. Die Beschleunigungswerte in Z-Richtung können durch das Abbremsen oder Beschleunigen des Fahrzeuges verfälscht werden. Beim Abbremsen des Fahrzeuges taucht das Fahrzeug in die fordere Federung ein, wodurch die Z-Achse nicht mehr orthogonal zur Straßenoberfläche steht. Zudem sind die auftretenden Beschleunigungen, die durch die Straßenoberfläche erzeugt werden, abhängig von der Fahrzeuggeschwindigkeit. Da die Straßenqualität später in Abhängigkeit von der Fahrzeuggeschwindigkeit skaliert wird, ist eine möglichst konstante Geschwindigkeit in der Teilstrecke nötig.
Um die konstante Geschwindigkeit innerhalb eines Teilstückes zu prüfen, wird zunächst die Durchschnittsgeschwindigkeit des Teilstückes $V_{Teilstück}$ berechnet:

$$V_{Teilstück} = \sum_{k=1}^{n_{GNSS\,Data}} \sqrt{{V_{Nord}}_k^{\,2} + {V_{Ost}}_k^{\,2}} \cdot \frac{1}{n_{GNSS\,Data}}$$

Danach wird geprüft, wie groß die Differenz der einzelnen Geschwindigkeiten von der Durchschnittsgeschwindigkeit des Teilstückes ist. Da die Geschwindigkeit nicht immer exakt gehalten werden kann, gibt es hierfür einen Schwellwert $V_{Mittelabweichung}$ der nicht überschirrten werden darf:

$$\left|V_{Teilstrecke} - V_{Fahrzeug_k}\right| \leq V_{Mittelabweichung} \qquad D_f = \{k = \mathbb{N} \mid \mathrm{k} \leq n_{GNSS\,Data}\}$$

Abhängig vom verwendeten Fahrzeug kann die Geschwindigkeit unterschiedlich gut gleichmäßig gehalten werden. Die verwendeten Sensoren haben ebenfalls einen Einfluss auf die Gleichmäßigkeit der Geschwindigkeit. Daher ist $V_{Mittelabweichung}$ für unterschiedliche Fahrzeuge anzupassen.

4.2.4 Erdbeschleunigung entfernen

Zur Ermittlung der Straßenqualität wird die Beschleunigung in Z-Richtung benutzt. In Abhängigkeit von der Neigung der Straße und somit auch von der Neigung des Fahrzeuges wirkt die Erdbeschleunigung unterschiedlich stark in Z-Richtung des Fahrzeuges. Die wirkende Erdbeschleunigung wird für jede Teilstrecke bestimmt. Es wird dabei die Annahme getroffen, dass die Straße innerhalb der Teilstrecke die Neigung in Längs- und Querrichtung konstant ist. Der Mittelwert der Beschleunigung der Z-Achse entspricht ca. der Erdbeschleunigung. Innerhalb der Teilstrecke wird der Mittelwert der Z-Beschleunigung wie folgt gebildet:

$$\overline{a}_z = \frac{1}{n_{Imu\ Data}} \cdot \sum_{k=1}^{n_{Imu\ Data}} a_{z_k}$$

Ausgehend davon, dass die Erdbeschleunigung negativ auf die Z- Achse wirkt, kann der Mittelwert auf jeden einzelnen Beschleunigungswert addiert werden, um die Erdbeschleunigung zu entfernen:

$$a_{z\ effektiv_k} = a_{z_k} - \overline{a}_z \quad D_f = \{k = \mathbb{N} \mid 1 < \mathrm{k} \leq n_{IMU\ Data}\}$$

4.2.5 Straßenqualität ermitteln

Die Straßenqualität wird durch die Rauheit der Straße bestimmt. Große Rauheiten und Unebenheiten auf der Straße erzeugen auch dementsprechende Beschleunigungen auf der Z-Achse. Events wie Schlaglöcher etc. erzeugen hohe Beschleunigungen und verschlechtern somit die Straßenqualität. Die Straßenqualität wird innerhalb einer Teilstrecke über den RMS der Z-Beschleunigung bestimmt. Je höher der RMS in der Teilstrecke ist, desto schlechter ist die Straßenqualität [11].

$$RMS_{a_z} = \sqrt{\frac{1}{n_{Imu\ Data}} \cdot \sum_{k=1}^{n_{Imu\ Data}} {a_{z\ effektiv_k}}^2}$$

Die Straßenqualität, also der RMS_{a_z} ist abhängig von der Geschwindigkeit des Fahrzeuges. Wird dieselbe Straße mit unterschiedlichen Geschwindigkeiten abgefahren, so sind die resultierenden Beschleunigungen größer, je höher die Fahrzeuggeschwindigkeit ist. Um Straßenqualitäten unabhängig von der Fahrzeuggeschwindigkeit zu beurteilen, wird eine selbst entwickelte Korrekturfaktorfunktion (KFF) eingeführt. Diese ist abhängig von der Fahrzeuggeschwindigkeit in der Teilstrecke und der ermittelten Straßenqualität. Mit dieser Funktion werden alle RMS-Werte auf ein 50 km/h äquivalent angepasst:

$$RMS_{50} = KFF\left(V_{Teilstück}, RMS_{a_z}\right)$$

Wird derselbe Straßenabschnitt mit 30 km/h und mit 50 km/h überfahren, so ist der RMS_{50}, mit einer gewissen Toleranz, für beide Geschwindigkeiten gleich. Die KFF ist für jedes Fahrzeug unterschiedlich und muss dementsprechend ermittelt werden.
Zur Positionierung der Straßenqualität wird der Mittelpunkt der Teilstrecke bestimmt. Da es sich bei den Teilstrecken um kurze Abschnitte handelt, kann der Mittelpunkt ohne Berücksichtigung der Erdkrümmung und Abstand der Breitengarde berechnet werden. Der Mittelpunkt der Teilstrecken wird durch die erste und letzte Position der Teilstrecke berechnet:

$$Lat_{Mitte} = \frac{(Lat_0 + Lat_{n_{GNSS\ Data}})}{2}$$
$$Lon_{Mitte} = \frac{(Lon_0 + Lon_{n_{GNSS\ Data}})}{2}$$

4.2.6 Events finden

Punktuelle Störungen wie z.B. Bahnschwellen, Schlaglöcher oder Kanalschachtabdeckungen werden als Events bezeichnet. Abhängig von der Größe des Events kann diese zwei punktuelle Störung erzeugen. Es wird immer nur in einem Teilabschnitt nach Events gesucht. Für das Finden von Events wird der Ruck der Z-Achse herangezogen. Durch das Ermitteln von Maximalwerten im Ruck, welche über einem variablen Schwellwert liegen, werden Events erkannt. Der Schwellwert ist dabei abhängig von Mittelwert des Betrags des Rucks. Des Weiteren wird eine Totzeit benötigt. Die Totzeit sorgt dafür, dass ein Event beim Überfahren der Vorderachse und der Hinterachse als nur ein Event detektiert wird. Innerhalb dieser Totzeit kann kein weiteres Event gefunden werden. Die Totzeit wird bei aufeinander folgenden Teilstrecken an die nächste Teilstreck übergeben. Für jedes Fahrzeug wird ein passender Schwellwert und Totzeit benötigt. Die Totzeit ist abhängig vom Radstand des Fahrzuges.
Zunächst wird der Ruck der Z-Achse bestimmt. Hierfür wird angenommen, dass die IMU-Daten immer exakt mit der eingestellten Frequenz aufgezeichnet werden. Damit durch die Bildung des Rucks keine Informationen verloren gehen, sind bei aufeinanderfolgenden Teilstrecken die letzten IMU-Daten gleichzeitig die ersten IMU-Daten der darauffolgenden Teilstrecke.

$$j_{z_k} = \left(a_{z_{k-1}} - a_{z_k}\right) \cdot \frac{1}{f_{IMU\ Daten}} \quad D_f = \{k = \mathbb{N}\ |1 <\ \mathrm{k} \leq n_{IMU\ Data}\}$$

Damit nur nach einem maximalen Ruck und nicht nach einem minimalen Ruck gesucht werden muss, wird der Betrag des Rucks gebildet:

$$j_{z\ Betrag_k} = \left|j_{z_k}\right| \qquad D_f = \{k = \mathbb{N}\ |1 <\ \mathrm{k} \leq n_{IMU\ Data}\}$$

Als nächstes wir der Mittelwert des Betrages des Rucks innerhalb der Teilstrecke gebildet. Dieser wird zu Schwellwertfindung benötigt, ab welchem Betrag ein Event erkannt wird.

$$\overline{Jz\ Betrag_k} = \sum_{k=2}^{n_{Imu\ Data}} j_{z\ Betrag_k} \cdot \frac{1}{n_{Imu\ Data} - 1}$$

Ein Event wird erkannt, wenn der Ruck in einem Messpunkt j_{z_k} den Mittelwert des Betrages des Rucks um den Faktor $j_{Schwellwert}$ überschreitet. Dieser Schwellwert ist abhängig von dem verwendeten Fahrzeug. Ein Event wird erkannt, wenn folgendes gilt:

$$j_{z_k} > \overline{Jz\ Betrag_k} \cdot F_{Ruck} \qquad D_f = \{k = \mathbb{N} \mid \mathrm{k} \leq n_{IMU\ Data}\}$$

Die Abhängigkeit der Eventfindung durch den Mittelwert des Betrages des Rucks soll vermieden werden, dass auf schlechten Straßen fälschlicherweise Events erkannt werden. Auf einer glatten Straße erzeugt daher ein kleiner Ruck schon ein Event, während bei einer sehr schlechten Straße ein größerer Ruck benötigt wird, um ein Event zu finden.
Ein Event wird über die maximal auftretende Beschleunigung klassifiziert. Hierfür wird zunächst aus dem Wert des Rucks in einem Event die Beschleunigung zurückgerechnet:

$$a_{z_{event\ 1}} = j_{z_{event\ 1}} \cdot \frac{1}{f_{IMU\ Daten}}$$

Dieser Beschleunigungswert ist dann noch abhängig von der gefahrenen Geschwindigkeit. Dieser Beschleunigungswert wird ebenfalls auf ein 50 km/h äquivalenten Wert angepasst. Es wird davon ausgegangen, dass die Beschleunigungen eines Events ebenfalls die gleiche lineare Abhängigkeit von Fahrzeuggeschwindigkeit haben, wie die RMS-Werte der Straßenqualität. Daher wird aus dem Quotienten vom RMS_{50} und RMS_{a_z} der Korrekturfaktor für die Eventbeschleunigung gebildet:

$$a_{50_{event\ 1}} = a_{z_{event\ 1}} * \frac{RMS_{50}}{RMS_{a_z}}$$

Diese Korrektur der Beschleunigung ist so möglich, da sich die Events in der gleichen Teilstrecke befinden wie die ermittelte Straßenqualität. Die Totzeit soll dafür sorgen, dass nur ein Event erkannt wird, wenn dieses mit der Vorder- und Hinterachse des Fahrzeugs überfahren wird. Um dieses zu gewährleisten, muss die Totzeit abhängig vom Radstand und der Fahrzeuggeschwindigkeit bestimmt werden. Daher gilt:

$$t_{tot} = \frac{l_{Raadsatnd}}{V_{Fahrzeug}}$$

Die Totzeit muss anschließend abhängig von der Messfrequenz passend gerundet werden. Dem Event wird die Position der zugehörigen Beschleunigungsdaten zugewiesen. Sollten die GNSS-Daten deutlich langsamer abgetastet werden als die IMU-Daten, können an einer GPS-Position mehrere Events lokalisiert werden.

4.3. Hardware der Messeinrichtung für das Fahrrad

Für die Hardware der Messeinrichtung für Fahrräder gab es eine Auswahl an Hardwarekomponenten. Diese werden gegenübergestellt und anhand ihrer Kriterien verglichen. Hierfür gibt es folgendes Punktesystem: Die Gewichtung (G) der Kriterien wird prozentual nach ihrer Wichtigkeit aufgeteilt. Die Kriterien bekommen eine Wertung (W) von 0 bis 5 Punkten. Je höher die Punktzahl, je besser ist ein Kriterium erfüllt.

4.3.1 Gehäuse

Für die Auswahl eines passenden Gehäuses stehen zwei Konzepte zur Auswahl: Die Nutzung eines selbst erstellten 3D-Druck-Gehäuses oder die Nutzung einer fertigen wasserfesten Box/Schaltschrank.

Kriterium	**Gewichtung**	**3D-Druck Gehäuse**		**Wasserfeste Box / Schaltschrank**	
	G	**W**	**G·W**	**W**	**G·W**
Wasserdichtigkeit nach IP-Zertifizierung	0,30	3	0,9	5	1,5
Einfaches Öffnen und Schließen möglich	0,20	2	0,4	5	1
Stabilität	0,20	4	0,8	5	1
Befestigungsmöglichkeit der Komponenten	0,10	5	0,5	3	0,3
Integration von Befestigungsmöglichkeiten zur Befestigung am Fahrrad	0,10	5	0,5	2	0,2
Anpassung der Größe an die Komponenten	0,10	5	0,5	3	0,3
Gesamt:	**1,00**	**-**	**3,6**	**-**	**4,3**

Tabelle 11: Kriterien vergleich des Gehäuses

Durch die Bewertung der Kriterien erfolgt die Umsetzung durch eine/n Wasserfeste Box/Schaltschrank.

4.3.2 Befestigungsmöglichkeit

Für die Auswahl einer passenden Befestigungsmöglichkeit der Messeinrichtung an einem Fahrrad stehen drei Konzepte zur Auswahl: Befestigung mit Hilfe von Spanngurten auf dem Gepäckträger, Befestigung mit Hilfe von Rohrschellen am Rahmen des Fahrrades oder Befestigung an der Zweilochhalterung für Getränkehalter am Unterrohr oder Sattelrohr des Fahrrades.

Kriterium	**Gewichtung**	**Spanngurte am Gepäckträger**		**Rohrschellen am Fahrradrahmen**		**Zweilochhalterung für Getränkehalter**	
	G	**W**	**G·W**	**W**	**G·W**	**W**	**G·W**
Geringer Montageaufwand	0,3	4	1,2	3	0,9	3	0,9
Universell an mehreren Fahrrädern einsetzbar	0,2	4	0,8	4	0,8	4	0,8
Stabile Verbindung zwischen Messeinrichtung und Fahrrad	0,2	4	0,8	4	0,8	5	1
Keine Einschränkung beim Fahren	0,2	5	1	4	0,8	4	0,8
Flexible Positionierung	0,1	2	0,2	4	0,4	1	0,1
Gesamt:	**1,00**	**-**	**4**	**-**	**3,7**	**-**	**3,6**

Tabelle 12: Kriterien vergleich der Befestigungsmöglichkeit

Durch die Bewertung der Kriterien erfolgt die Umsetzung durch eine Befestigung mit Spanngurten am Gepäträger.

4.3.3 Einplatinencomputer

Für die Auswahl des passenden Einplatinencomputer stehen zwei Konzepte zur Auswahl: Ein Raspberry Pi 3B+ oder ein NVIDIA Jetson Nano.

Kriterium	**Gewichtung**	**Raspberry PI**		**Jetson Nano**	
	G	**W**	**G·W**	**W**	**G·W**
Rechenleistung passend für den Algorithmus	0,5	5	2,5	5	2,5
Geringe Größe	0,3	5	1,5	4	1,2
ROS-Treiber für Sensoren vorhanden	0,2	5	1	4	0,8
Gesamt:	**1,00**	**-**	**5**	**-**	**4,5**

Tabelle 13: Kriterien vergleich der Einplatinencomputer

Durch die Bewertung der Kriterien erfolgt die Umsetzung durch einen Raspberry Pi.

4.3.4 Internetverbindung

Für die Bereitstellung einer Internetverbindung stehen zwei Konzepte zur Auswahl: Verbindung bereitstellen durch einen UMTS-Stick oder die Nutzung eines WLAN-Hotspots.

Kriterium	**Gewichtung**	**UMTS-Stick**		**WLAN-Hotspot**	
	G	**W**	**G·W**	**W**	**G·W**
Einfache SSH-Verbindung möglich	0,4	3		5	
Keine Berücksichtigung bei der Konzeption des Gehäuses	0,3	0		5	
Unabhängig von anderen Geräten	0,3	5		0	
Gesamt:	**1,00**	**-**	**2,7**	**-**	**3,5**

Tabelle 14: Kriterien vergleich der Möglichkeiten zur Internetverbindung

Durch die Bewertung der Kriterien erfolgt die Umsetzung durch einen WLAN-Hotspot.

4.3.5 Inertialsensorik

Für die Auswahl der passenden Inertialsensorik stehen drei IMUs zur Auswahl. Dieses ist Hardware, die in anderen Projekten bereits genutzt und zur Verfügung gestellt wurde:

GY-521 6-DoF [19]	**Adafruit 9-DoF** [20]	**Adafruit 10-DoF** [21]
Beschleunigungssensor: MPU-6050 3-Achsen ±2 g / ±4 g / ±8 g / ±16 g 1,25 Hz – 40 Hz	Beschleunigungssensor: ICM-20948 3-Achsen ±2 g / ±4 g / ±8 g / ±16 g 1 Hz – 100 Hz	Beschleunigungssensor: LSM303 3-Achsen ±2 g / ±4 g / ±8 g / ±16 g 1 Hz – 100 Hz
Gyroskop: MPU-6050 3-Achsen ±250 dps / ±500 dps / ±1000 dps / ±2000 dps	Gyroskop: ICM-20948 3-Achsen ±250 dps / ±500 dps / ±1000 dps / ±2000 dps	Gyroskop: L3GD20H 3-Achsen ±250 dps / ±500 dps / ±2000 dps
-	Kompass: ICM-20948 3-Achsen ±4900 µT	Kompass: LSM303 3-Achsen ±1300 µT bis ±8100 µT
-	-	Barometer: BMP180 -40 bis 85 °C 300 bis 1100hPa

Tabelle 15: Technische Daten der IMUs

Aufgrund der technischen Daaten wurde sich für das Adafruit 9 DoF IMU-Modul entschieden, da diese Beschleunigungswerte mit 100 Hz aufzeichnen kann. Des Weiteren ist für den ICM-20948 ein ROS-Treiber vorhanden, welcher die Einbindung in das System erleichtert.

4.3.6 GPS-Sensor

Als GPS-Sensor steht nur das SparkFun Venus GPS-Modul [20] zur Verfügung. Dieses besitzt folgende technische Daten:

SparkFun Venus GPS		
Genauigkeit	Position Geschwindigkeit Zeit	2,5 m CEP 0,1 m/s 60 ns
Update Rate	1 / 2 / 4 / 5 / 8 / 10 / 20 Hz (default 1Hz)	
Interface	UART LVTTL level	
Betriebslimits	Höhe Geschwindigkeit	< 18000 m < 515 m/s
Baud Rate	4800 / 9600 / 38400 / 115200	
Betriebstemperatur	-40 °C bis 85 °C	

Tabelle 16: Technische daten GPS-Sensor

4.3.7 Akkumulator

Als Akku steht die TL-PB15600 Powerbank von TP-Link zur Verfügung. Diese besitzt folgende technische Daten:

TL-PB15600 Powerbank	
Kapazität	15600 mAh
Eingang	DC 5 V / 2,4 A
Ausgang	Max. DC 5 V/2,4 A pro Port
Schnittstelle	1 Micro-USB-Port 2 USB2-Ports 6-stufige Ladestatus-LED-Anzeige
Größe	61,2 mm x 22,5 mm x 150 mm
Gewicht	344 g

Tabelle 17: Technische Daten Akkumulator

5. Umsetzung

In diesem Kapitel wird die Umsetzung der Konzeption beschrieben. Im Testfahrzeug als auch in der Messeinrichtung für das Fahrrad kommen unterschiedliche Softwarekomponenten zum Einsatz. Die Unterschiede beziehen sich auf die verwendeten Sensoren und die daraus resultierenden Treiber. Der Algorithmus ist in beiden Anwendungsfällen der gleiche. Zudem wird der Aufbau der Messeinrichtung für das Fahrrad beschrieben. Die Hardware im Testfahrzeug war bereits vorhanden.

5.1. Hardware im Fahrrad

Für den Aufbau der Hardware wurde als Grundbaustein eine wasserfeste Transportbox von Mil-Tec gewählt. Diese besteht aus ABS-Kunststoff. Durch die angebrachten Scharniere lässt sich die Box einfach öffnen und schließen. Die Box hat Außenmaße von 228x130x46 mm. Der gesamte Aufbau der Hardware ist in Abbildung 8 dargestellt. Zur Befestigung der Komponenten in der Box wurde eine passgenaue Acrylscheibe in den Boden der Box eingeklebt. Diese Acrylscheibe hat eine passgenaue Vertiefung für den Akku. Beim Schließen des Deckels wird der Akku festgeklemmt und kann sich somit nicht mehr bewegen. Für die Befestigung des Raspberry PI sind drei Gewindestifte in der Acrylscheibe angebracht. Aus platztechnischen Gründen musste der Raspberry PI verkehrtherum eingebaut werden. Zwei USB-Anschlüsse sind im eingebauten Zustand weiterhin benutzbar. Die Adafruit 9-Dof IMU ist mithilfe von Gewindeabstandshaltern auf der Acrylscheibe angebracht. Die IMU sitzt genau mittig im Gehäuse. Das Sparkfun Venus GPS-Modul besitzt keine Schraublöcher zur Befestigung. Daher wurde mithilfe einer Lochrasterplatine und Buchsenleisten eine Befestigungsmöglichkeit geschaffen. Die GNSS-Antenne ist mit doppelseitigem Klebeband auf dem Akku befestigt.

In Abbildung 9 ist das Blockschaltbild der Hardware dargestellt. Der Raspberry Pi wird über den 2,1 A Ausgang des Akkus mit Spannung versorgt. Das Sparkfun Venus GPS-Modul ist über eine serielle Schnittstelle mit dem Raspberry Pi verbunden. Die Adafruit 9-Dof IMU ist über I^2C mit dem Raspberry Pi verbunden. Das Smarthone, welches zur Steuerung des Raspberry Pi genutzt wird, ist über WLAN verbunden.

Die Messeinrichtung für das Fahrrad wurde auf dem Gepäckträger des Eskute E-Bikes montiert (Abbildung 10). Zum Schutz des Gepäckträgers und für einen besseren Halt wurde ein rutschhemmender Schaumstoff verwendet. Die Befestigung selbst wurde durch einen Gummiexpander und einen Spanngurt realisiert. Bei der Montage ist darauf zu achten, dass die Ratsche des Spanngurtes wie auf Abbildung 10 positioniert ist, da es sonst zu Störungen

des GPS-Empfangs kommt. Die Messeinrichtung ist so auszurichten, dass die IMU möglichst mittig über dem Hinterreifen platziert ist.

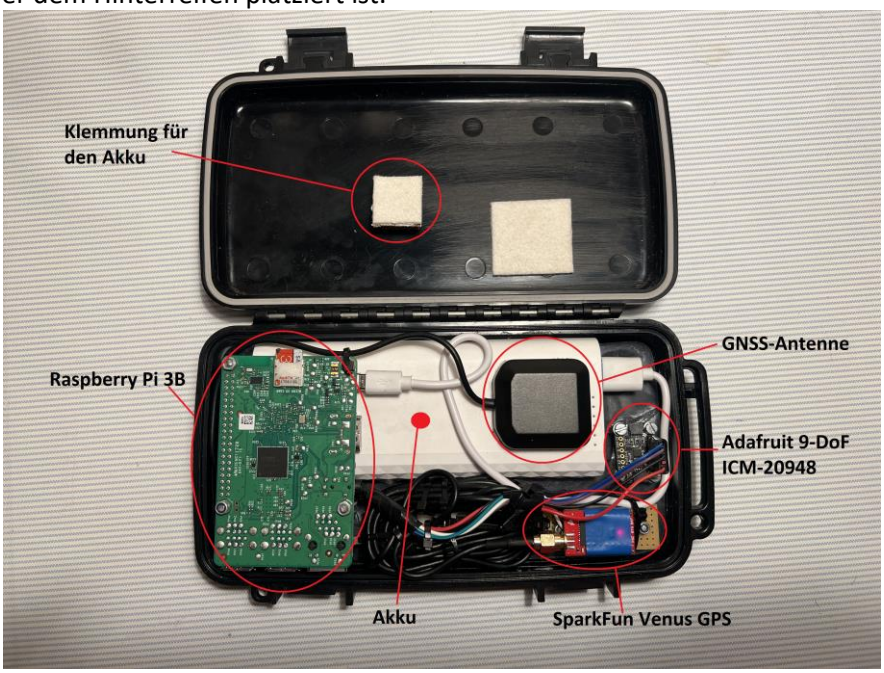


Abbildung 9: Aufbau der Hardware

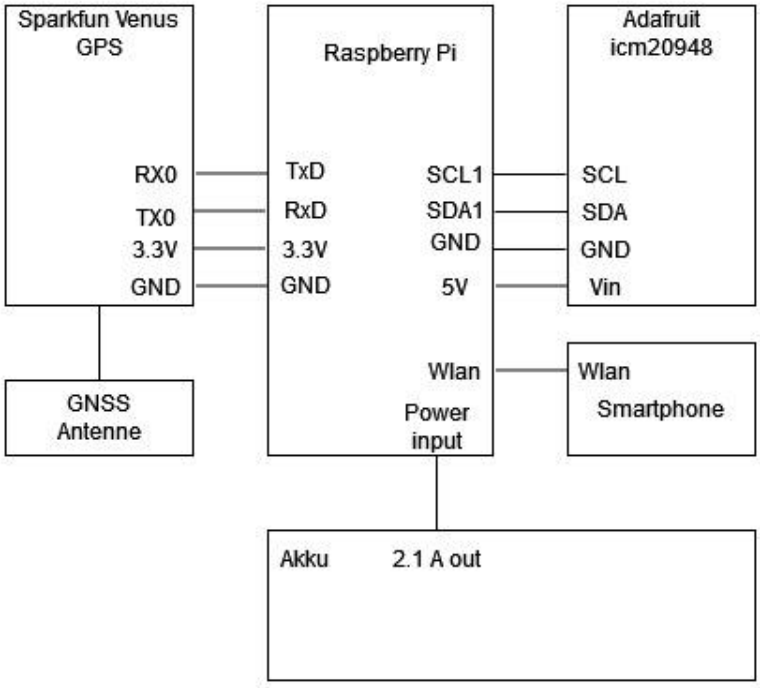


Abbildung 8: Blockschaltbild der Hardware

Abbildung 10: Befestigung der Messeinrichtung für das Fahrrad

5.2. Software für den Einsatz im Testfahrzeug

Im Fahrzeug besteht die Aufgabe darin, die verwendeten Module in die vorhandene Software zu integrieren. Es ist bereits ein Ubuntu 16.04 mit ROS Kinetic Kane aufgesetzt. Ebenso ist der SBG-Treiber [23] installiert der die IMU- und GNSS-Daten in der ROS-Umgebung zur Verfügung stellt. In Abbildung 8 ist die gesamte ROS-Struktur dargestellt.

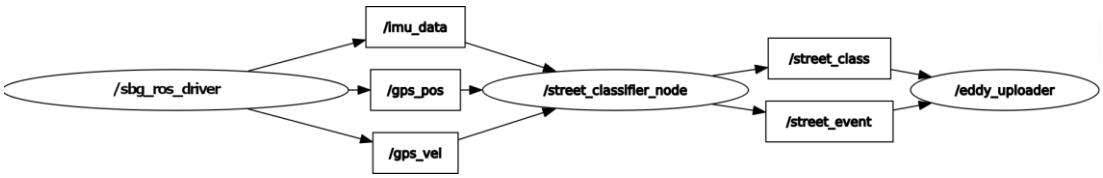


Abbildung 11: ROS im Testfahrzeug

Der „street_classifier_node“-Knoten wurde mit den Nachrichtentypen des SBG-Treibers entwickelt und daher kann der „street_classifier_node“-Knoten die Daten direkt verwenden. Es wird kein Adapter benötigt, wie in der Konzeption beschrieben, da der Algorithmus direkt mit den Daten des „sbg_ros_driver“-Konten arbeiten kann. In Tabelle 18 sind die für den

Algorithmus relevanten Daten aus den Topics des SBG-Treibers aufgeführt. Für dieses Projekt wurden die voreingestellten Abtastfrequenzen übernommen.

ROS-Topic	Inhalt	Frequenz
/gps_vel	Geschwindigkeit des Fahrzeuges aufgeteilt in Nord und Süd-Richtung	5 Hz
/gps_pos	GNSS-Status, Breitengrad, Höhengrad und Höhe	5 Hz
/imu_data	Beschleunigung des Fahrzeuges in XYZ-Richtung	25 Hz

Tabelle 18: Daten des SBG System Apogee-D Treiber

Im „street_classifier_node“-Knoten wurde der Algorithmus umgesetzt. Dieser veröffentlicht über das „street_calss“-Topic die ermittelte Straßenqualität und über das „street_event“-Topic die gefunden Events. Für beide Topics wurde jeweils ein eigener ROS-Nachrichtentyp erstellt. Der „eddy_uploader“-Knoten ist eine Entwicklung aus dem Projekt „EDDY“ [24] und ist die Schnittstelle zwischen ROS und dem EDDY-Geoserver. Diese Schnittstelle lädt bei einer Internetverbindung die ortsabhängigen Straßenqualitäten und Events z.B. Schlaglöcher an den Geoserver hoch. Bei einer Unterbrechung kann der „eddy_uploader“-Knoten die Daten speichern und bei erneuert Internetverbindung hochladen.
Für die Steuerung der Software im Testfahrzeug wird sich mit einer Remote-Desktop-Verbindung von einem weiteren Computer auf den Computer im Fahrzeug aufgeschaltet.

5.3. Software für den Einsatz im Fahrrad

Für die Software im Fahrrad wurde ein Ubuntu 20.04 Image von Ubiquity Roborics gewählt. Auf diesem Image ist ROS Noetic Ninjemys vorinstalliert. In Abbildung 9 ist die ROS-Struktur für die Messeinrichtung im Fahrrad dargestellt.

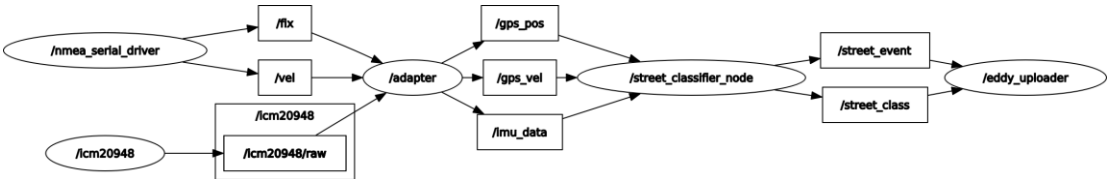


Abbildung 12: ROS-Struktur in der Messeinrichtung für Fahrrad

Da für die Messeinrichtung im Fahrrad die Messdaten von anderen Sensoren bezogen werden, kommt hier ein Adapter zum Einsatz. Dieser Adapter wandelt die Messdaten in die SGB-Treiber Nachrichtentypen um. Diese benötigt der „street_classifier_node“-Knoten. Als Treiber für das GPS-Modul wird der „nmea_navsat_driver“-Konten eingesetzt [25]. Dieses ist ein fertig entwickeltes Treiberpaket für das SparkFun Venus GPS-Modul, welches NMEA-Nachrichten über eine Serielle Schnittstelle sendet.
Der „icm20948“-Knoten ist der Treiber für die ICM20948-Sensorik, welche in der Adafruit 9-DoF IMU verbaut ist [26]. Hierdurch werden die IMU-Daten bereitgestellt.

Die Messfrequenz der GNSS-Daten des SparkFun Venus GPS-Moduls sind standartmäßig auf 1 Hz eingestellt und wurden auf 5 Hz hochgesetzt. Hierzu wurde eine entsprechende Nachricht, gemäß des Datenblattes, über die serielle Schnittstelle an das SparkFub Venus GPS-Modul gesendet [22]. Die Messfrequenz des ICM20948 wir bei den standardmäßig eingestellten 100 HZ belassen.
Der Adapter konvertiert die Nachrichtentypen des „nmea_navsat_driver“-Knoten und des „icm20948“-Knoten in das SBG-Treiber Nachrichtenformat. Dieses ist nötig, da der „street_classifier_node“-Knoten für die SBG-Treiber programmiert wurde. Eine Veränderung der Frequenzen findet nicht statt. In Tabelle 19 ist eine Auflistung er umgewandelten ROS-Topics dargestellt.

Adapter Input Topic	Adapter Output Topic	Inhalt	Frequenzen
/vel	/gps_vel	Geschwindigkeit des Fahrzeuges aufgeteilt in Nord- und Süd Richtung	5 Hz
/fix	/gps_pos	Position des Fahrzeuges: Breitengrad, Höhengrad und Höhe	5 Hz
/icm20948/raw	/imu_data	Beschleunigung des Fahrzeuges in XYZ-Richtung	100 Hz

Tabelle 19: ROS-Topic Umwandlung im Adapter

Im Adapter wir zusätzlich die Einbauposition der IMU korrigiert, da die Achsen in der Messeinrichtung nicht nach DIN ISO 8855 ausgerichtet sind. Zudem werden die Beschleunigungswerte von ICM20948 in LSB/g angegeben und werden daher umgerechnet. Der ICM ist im Treiber auf einen Bereich von ±8 g eingestellt. Es gilt daher:

$$X_{Output} = \frac{Y_{Inputt}}{4096} \cdot 9{,}813\ \frac{m}{s}$$
$$Y_{Output} = -\frac{X_{Inputt}}{4096} \cdot 9{,}813\ \frac{m}{s}$$
$$Z_{Output} = -\frac{Z_{Inputt}}{4096} \cdot 9{,}813\ \frac{m}{s}$$

Die Daten aus dem Adapter werden im „street_classifier_node“-Knoten mit den entsprechenden Parametern fürs Fahrrad verarbeitet. Auch im Fahrrad kommt der „eddy_uploader“-Knoten zum Einsatz, um die Straßenqualitäten auf dem EDDY-Geoserver hochzuladen.
Um die Steuerung der Messeinrichtung zu realisieren, wird in Ubuntu 20.04 die Verbindung zu dem WLAN-Hotspot eines Smartphones konfiguriert. Über das Smartphone kann mit einer entsprechenden SSH-App die Verbindung hergestellt werden. Zum Erleichtern des Startens aller ROS-Knoten wurde ein Bash-Skript entwickelt. Durch Ausführen dieses Skriptes werden alle erforderlichen ROS-Knoten gestartet.

5.4. Algorithmus

Der Algorithmus ist im „street_classifier_node“-Knoten umgesetzt. Dieser ROS-Knoten ist in #C programmiert. Der Ablauf des Algorithmus wird durch eine State Maschine realisiert. Hiervon ausgenommen ist die Initialisierung von ROS. In Abbildung 13 wird die State Maschine dargestellt. Diese wird mit einer Frequenz von 500 Hz ausgeführt, um sicherzustellen, dass alle ankommenden Daten erfasst werden. Die IMU und GNSS-Daten werden außerhalb der State Maschine durch eine Callbag-Funktion bereitgestellt.
Im „Init“- State werden alle verwendeten Variablen und Arrays zurückgesetzt. In diesem State befinde sich die State Maschine, wenn die Geschwindigkeit nicht im voreingestellten Bereich liegt, oder kein GPS-Signal vorhanden ist.

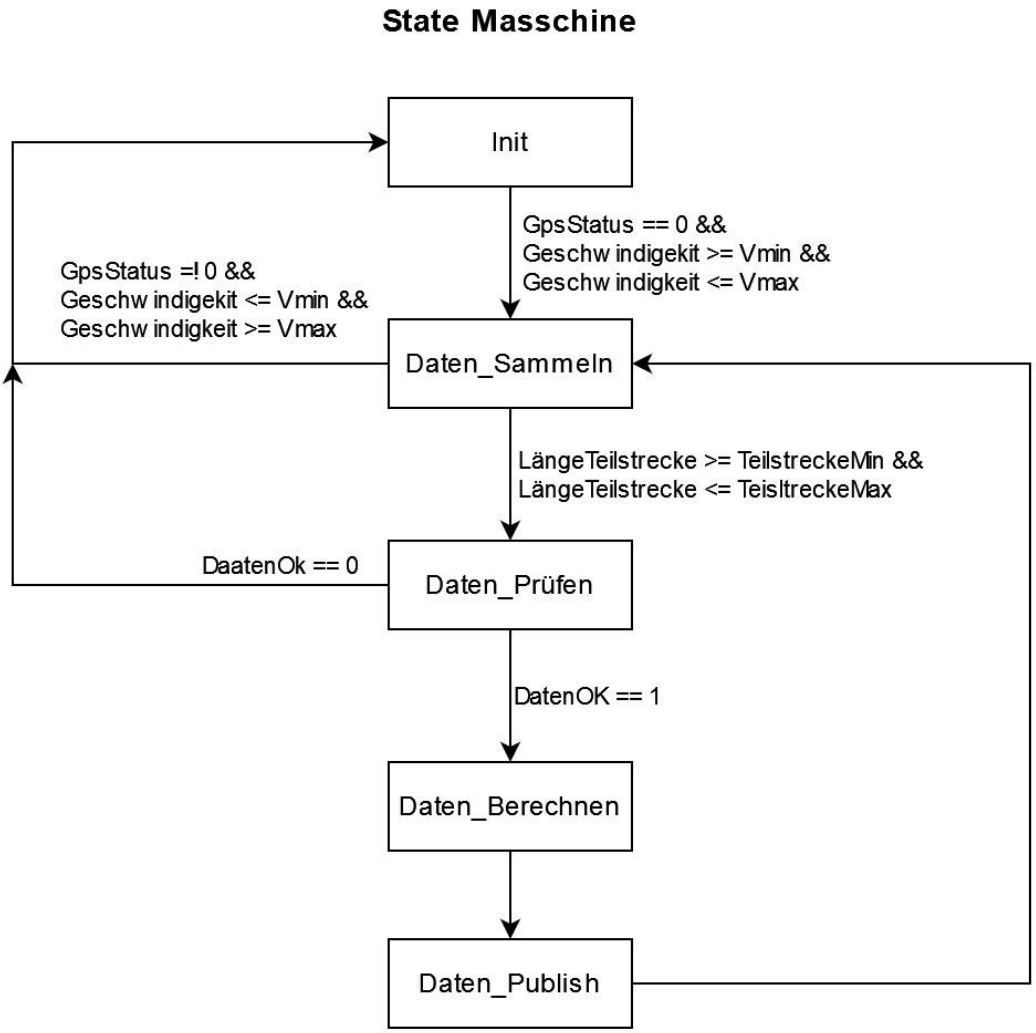


Abbildung 13: State Maschine des „street_classifier_node“-Knoten

Im „Daten Sammeln“-State werden die IMU und GNSS-Daten je in ein separates Array gespeichert. Es werden immer dann neue Daten in die Arrays gespeichert, wenn neue IMU-Daten eingegangen sind, da diese die höchste Abtastfrequenz haben. Bei den GNSS-Daten werden so lange die gleichen Daten in das Array geschrieben bis neue GNSS-Daten am „street_classifier_node“-Knoten ankommen. Es wird in jedem Durchlauf die Distanz zwischen der ersten und aktuell eingelesen GPS-Position berechnet. Liegt die Distanz zwischen $l_{Teilstrecke}$ und $l_{Teilstrecke_{Maximal}}$ wird in den nächsten State gewechselt. Liegt die Distanz unter $l_{Teilstrecke}$ werden weiterhin Daten gesammelt. Liegt die Distanz über $l_{Teilstrecke_{Maximal}}$ wird in den „Init“- State gewechselt. Sollte die GPS-Signal fehlerhaft sein oder die Fahrzeuggeschwindigkeit nicht in dem Arbeitsbereich liegen wird ebenfalls in den „Init“- State gewechselt.
Im „Daten_Prüfen“-State wird der Mittelwert der einzelnen Fahrzeuggeschwindigkeiten im Array gebildet und Geschwindigkeitsmittelwertabweichung überprüft. Liegen die Fahrzeuggeschwindigkeiten innerhalb der Toleranz geht es in den nächsten State. Andernfalls geht es zurück in den „Init“- State.
Im „Daten_Berechnen“-State wird zuerst die wirkende Erdbeschleunigung auf die Z-Achse bestimmt und von den Beschleunigungsdaten subtrahiert. Danach wird der RMS und der RMS50 berechnet. Anschließend wird er Mittelpunkt der Teilstrecke berechnet. Für das Finden von Events wird der Ruck, der Betrag des Rucks und der Mittelwert des Rucks in der Teilstrecke gebildet. Danach wir die Totzeit bestimmt und die einzelnen Ruckwerte mit dem Mittelwert und dem Ruckfaktor verglichen. Danach wird automatisch in den nächsten State gewechselt.
Im „Daten_Publish“-State werden die Straßenqualität und die Events über das „street_class“-Topic und das „street_event“-Topic veröffentlicht. Danach werden alle Variablen und Arrays zurückgesetzt. Die Totzeit ist hiervon ausgenommen.
Für das „street_class“-Topic und das „street_event“-Topic wurden jeweils eigenen ROS-Nachrichten erstellt. Die Inhalte dieser Nachrichten sind in Tabelle 20 und Tabelle 21 dargestellt.

Datentyp	**Name**	**Einheit**	**Beschreibung**
geometry_msgs/Vector3	position	°	Mittelpunkt der Teilstrecke (Latitude, Longitude und Altitude)
float32	distance	m	Effektive Länge der Teilstrecke
float32	vel	m/s	Durchschnittsgeschwindigkeit in der Teilstrecke
float32	anzpkt	-	Anzahl an Messpunkten in der Teilstrecke
float32	rms	m/s²	Straßenqualität bei Durchschnittsgeschwindigkeit
float32	rms50	m/s²	Korrigierte Straßenqualität bei 50 Km/h äquivalent

Tabelle 20: Inhalt „Street_Class“-Nachricht

Datentyp	Name	Einheit	Beschreibung
geometry_msgs/Vector3	position	°	Mittelpunkt der Teilstrecke (Latitude, Longitude und Altitude)
float32	vel	m/s	Durchschnittsgeschwindigkeit in der Teilstrecke
float32	lvl	m/s²	Stärke des Events bei Durchschnittsgeschwindigkeit
float32	lvl50	m/s²	Korrigierte Stärke des Events bei 50 Km/h äquivalent

Tabelle 21: Inhalt "Street_Event"-Nachricht

5.5. Eddy-Uploader

Der Eddy-Uploader aus dem Projekt Eddy dient als Schnittstelle zwischen ROS und dem Geoserver. Die Daten auf dem Geoserver werden in JSON-Objekten gespeichert. In diesen JASO-Objekten befinden sich alle Informationen aus den ROS-Nachrichten und zusätzlich spezifische Informationen für den Server. Die JSON-Objekte enthalten alle Informationen aus den jeweiligen ROS-Nachrichten. Beim Hochladen an den Server werden noch weitere Informationen durch den Server hinzugefügt. Diese zusätzlichen Inhalte sind in Tabelle 22 aufgeführt.

Variable	Beschreibung
createdAt	Datum und Zeitstempel als das JSON-Objekt erstellt wurde. Entspricht dem Datum der Messfahrt.
createdBy	Abhängig von dem verwendetem API-Key wird der zugehörige Name eingetragen.
object_class	Wird zur Unterscheidung der verschiedenen Objekte auf dem Geoserver benutzt. Z.B. Straßenschilder, Straßenqualität oder Events.
id	Einzigartige Objekt-ID

Tabelle 22: Spezifische JSON-Objekte Inhalte

Durch den für diese Ausarbeit benutzten API-Key lautet der „createdBY“-Name "haw-team8". Für die „object_class“ werden in diesem Projekt verschiedene Bezeichnungen benutzt. Diese sind in Tabelle 23 aufgeführt.

object_class	Beschreibung
Street_Class_Auto	Ermittelte Straßenqualität durch das Testfahrzeug
Street_Event_Auto	Ermittelte Events durch das Testfahrzeug
Street_Class_Fahrrad	Ermittelte Straßenqualität durch das Fahrrad
Street_Event_Fahrrad	Ermittelte Events durch das Fahrrad

Tabelle 23: Objekt-Klassen auf dem Geoserver

Die allgemeine Funktion des Eddy-Uploader lässt sich wie folgt beschreiben: „Läuft der Node so sucht er nach den konfigurierten Topics und versucht, diese auch zu abonnieren. Die Nachrichten werden für den Upload vorbereitet und in eine Warteschlange gesteckt. Diese kann bei vielen Nachrichten sehr lang werden. Wird das Debug Level für ROS-Log aktiviert wird die Anzahl auch mit ausgegeben. Ein Upload erfolgt nach dem Prinzip First Come First Serve. Sollte der Upload - warum auch immer - nicht funktionieren, wird für eine Sekunde pausiert und danach erneut ein Versuch gestartet.
Wird der Node beendet, so wird versucht, den Rest noch hochzuladen. Sollte das nicht gelingen, wird die Warteschlange als JSON-Datei abgespeichert. Der Upload kann später durch ein Skript erfolgen.
Es ist möglich, die IDs der hochgeladenen Elemente zu speichern. Dies ist in der Config-Datei einstellbar. Dies ist besonders bei Tests sinnvoll, wenn die Daten danach wieder aus der Cloud gelöscht werden sollen.“ [24]

5.6. Visualisierung der Geoserver-Daten

Für die Visualisierung der Geoserver-Daten wird ein Python-Skript benutzt. Dieses ist nicht in ROS eingebunden und ermöglicht eine einfache Abfrage und Visualisierung der Serverdaten. Die Visualisierung basiert auf der Anleitung „Introduction_EDDY_Cloud“ [27] und der „Dokumentation zum Bau und Inbetriebnahme des HAW Airviews“ [28]. In Abbildung 14 ist der Ablauf der Visualisierung dargestellt.

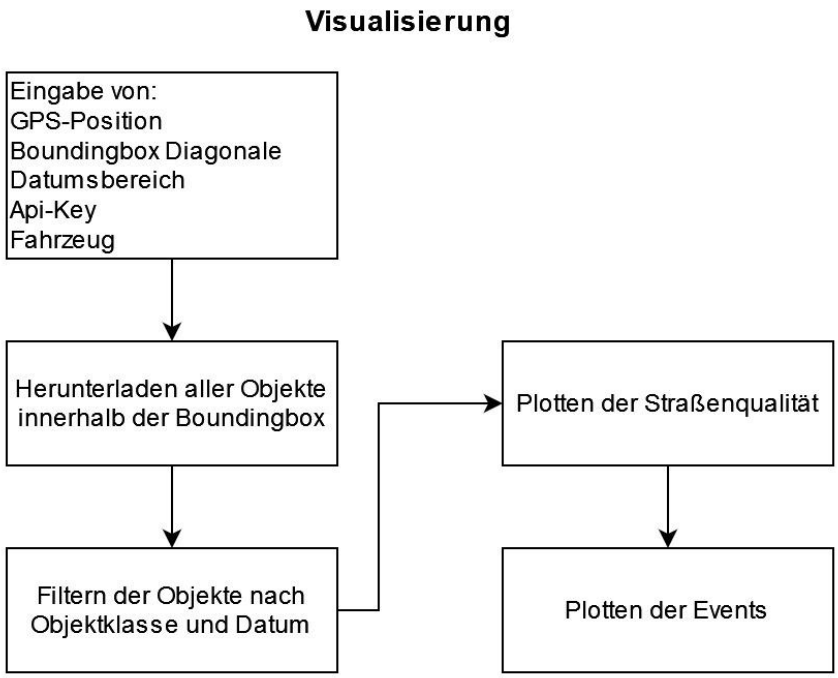


Abbildung 14: Ablauf der Visualisierung der Geoserver Daten

Die Visualisierung erzeugt um die eingegebene GPS-Position eine quadratische Boundingbox. Die Größe der Boundingbox wird durch die eingegebene Diagonale bestimmt. Innerhalb dieser Boundingbox werden alle vorhandenen Objekte heruntergeladen. Im Anschluss werden die Objekte nach der eingegeben Objektklasse und dem Datumsbereich gefiltert. Die Filterung nach dem Datumsbereich ermöglicht die Unterscheidung von mehreren Messfahrten auf der gleichen Strecke.
Für die Straßenqualität wird der RMS50-Wert in einer Karte geplottet. Dabei werden die Werte durch eine Farbskala unterschieden. Für die Events wird der lvl50-Wert geplottet. Die Größe der geplotteten Karte wird an die Verteilung der gefundenen Daten angepasst. In Abbildung 15 ist ein Beispiel dargestellt.

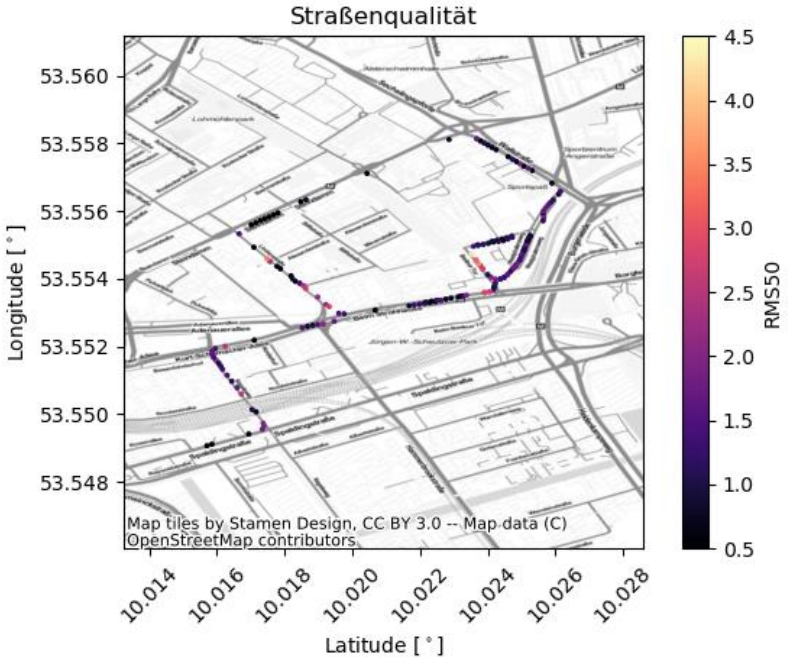


Abbildung 15: Beispiel für die Visualisierung

6. Bestimmung von Parametern

In diesem Kapitel werden die Parameter für den Betrieb des Algorithmus im Testfahrzeug und in der Messeinrichtung bestimmt. Durch die unterschiedlichen Aufbauten und Eigenschaften zwischen Testfahrzeug und Fahrrad müssen die Parameter getrennt bestimmt werden.

6.1. Testfahrzeug

In der nachfolgenden Tabelle befindet sich eine Übersicht aller Parameter, die in diesem Kapitel bestimmt werden.

Parameter	Wert
Offset X-Achse	0,1826 m/s²
Offset Y-Achse	-0,0467 m/s²
Offset Z-Achse	0,0076 m/s²
Minimalgeschwindigkeit	3 m/s
Maximalgeschwindigkeit	15 m/s
Länge der Teilstrecke	10 m
Maximale Länge der Teilstrecke	15 m
Geschwindigkeitsmittelwertabweichung	0,45 m/s
Korrekturfaktorfunktion	f1=0,0555*V+0,618; f2=0,0176*V+0,173
Ruck Faktor	2,5

Tabelle 24:Parameter für das Testfahrzeug

6.1.1 Offset SBG Systems Apogee-D

Zur Bestimmung des Offsets der Beschleunigungswerte des SBG Systems Apogee-D wurde eine Stelle gewählt, an der das Auto grade steht. In Abbildung 10, 11 und 12 wird die Messung grafisch dargestellt. Zur Offset-Bestimmung wird der Mittelwert für jede Achse über den Messzeitraum bestimmt. Für die X und Y-Achse ist der Offset, der Mittelwert mit umgekehrten Vorzeichen. Für die Z-Achse muss die Gravitationszone beachtet werden. Diese gibt für Hamburg eine Erdbeschleunigung von -9,8130 m/s² vor [17]. Daher ergibt sich für den Offset der Z-Achse:

$$a_{z_{Offset}} = -9{,}8130\ \frac{m}{s^2} - \left(-9{,}8054\ \frac{m}{s^2}\right) = 0{,}0076\ \frac{m}{s^2}$$

Beschleunigung	X-Achse	Y-Achse	Z-Achse
Mittelwert	-0,1826 m/s²	0,0467 m/s²	-9,8054 m/s²
Offset	0,1826 m/s²	-0,0467 m/s²	0,0076 m/s²
Standartabweichung	0,0152 m/s²	0,0157 m/s²	0,0238 m/s²

Tabelle 25: Offset und Standartabweichung der XYZ-Beschleunigungen

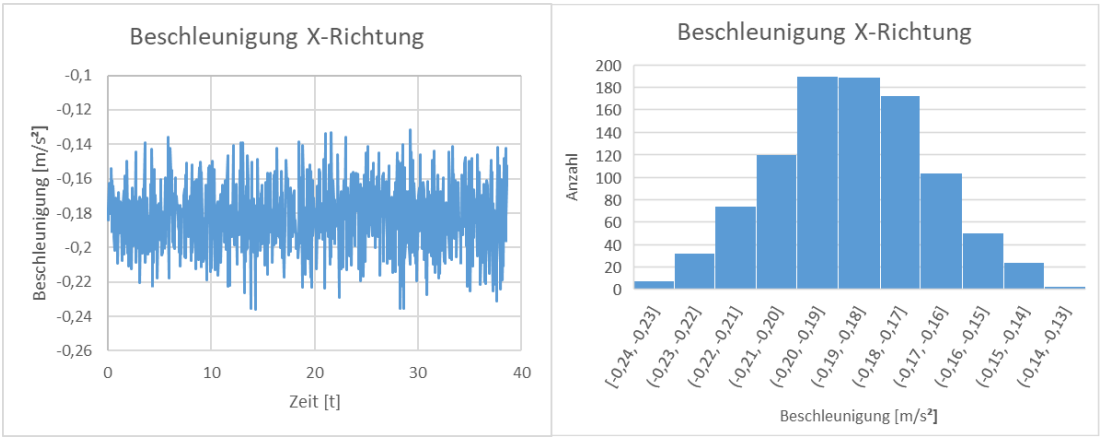


Abbildung 16: Offset X-Achse SBG Apogee-D

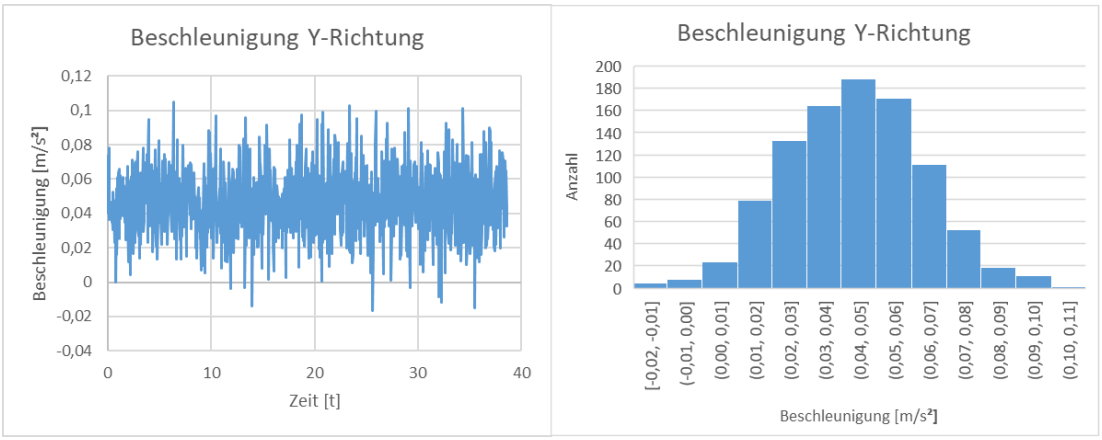


Abbildung 17: Offset y-Achse SBG Apogee-D

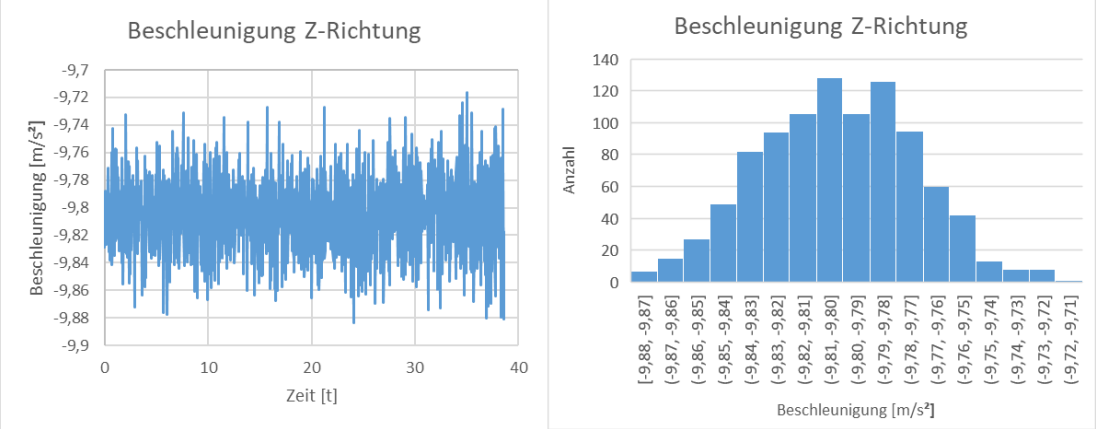


Abbildung 18: Offset Z-Achse SBG Apogee-D

6.1.2 Minimalgeschwindigkeit

Die Minimalgeschwindigkeit V_{Min} ist jene Geschwindigkeit ab der begonnen wird, Daten aufzuzeichnen. Dieses ist notwendig da eine Bewegung des Fahrzeuges zur Bestimmung der Straßenqualität Voraussetzung ist. Des Weiteren werden im Stand Erschütterungen durch das Ein und Aussteigen gemessen. Zudem verhält sich die Geschwindigkeit unter 10 km/h nicht linear zu der Beschleunigung in Z-Richtung [13]. Dieses ist wichtig für die Umsetzung der Korrekturfaktorfunktion. Daher gilt für die Minimalgeschwindigkeit:

$$V_{Min} = 3\,\frac{m}{s} = 10{,}8\,\frac{km}{h}$$

6.1.3 Maximalgeschwindigkeit

Die Maximalgeschwindigkeit V_{Max} ist jene Geschwindigkeit ab wann keine Messdaten mehr vom Algorithmus aufgezeichnet und verarbeitet werden. Die Messung soll hauptsächlich auf der TAVF und im Innenstadtbereich durchgeführt werden. Daher wird die Maximalgeschwindigkeit V_{Max} auf 54 km/h festgelegt. Ebenso ist die Abtastrate der Sensoren ab 54 km/h zu gering, um die Straßenqualität und Events sicher zu ermitteln. Bei 54 km/h ergeben sich folgende Abstände zwischen den Daten:

$$\Delta l_{GNSS\ Daten} = \frac{f_{GNSS\ Data}}{V_{max}} = \frac{15\,\frac{m}{s}}{5\ Hz} = 3\ m$$

$$\Delta l_{IMU\ Daten} = \frac{f_{GNSS\ Data}}{V_{max}} = \frac{15\,\frac{m}{s}}{25\ Hz} = 0{,}6\ m$$

6.1.4 Länge und maximale Länge der Teilstrecke

Die Länge der Teilstrecke ist abhängig vom Radstand des Fahrzeuges, der Messfrequenz der Sensoren und der Minimal- und Maximalgeschwindigkeit. Durch die Einteilung der Straßen in Teilstrecken entsteht am Anfang und Ende von aufeinanderfolgenden Teilstrecken eine Überschneidung der gemessenen Straße. Die Beschleunigung im Tesla wird sowohl von den Vorderreifen als auch von den Hinterreifen beeinflusst. Daher entsteht eine Überschneidung in der Länge des Radstandes des Tesla. Daher sollte die Teilstrecke mindestens dem dreifachen des Radstandes entsprechen. Daher gilt:

$$l_{Teilstrecke} \geq 3 \cdot l_{Radstand} = 3 \cdot 2{,}96\ m = 8{,}88\ m$$

Abhängig von der Messfrequenz und der Minimal- und Maximalgeschwindigkeit ergeben sich folgende Distanzen zwischen zwei GPS-Koordinaten:

$$\Delta l_{min} = \frac{f_{GNSS\ Data}}{V_{min}} = \frac{3\frac{m}{s}}{5\ Hz} = 0{,}6\ m$$
$$\Delta l_{max} = \frac{f_{GNSS\ Data}}{V_{max}} = \frac{15\frac{m}{s}}{5\ Hz} = 3\ m$$

Die horizontale Genauigkeit der GPS-Position des SBG System Apogee-D ist im SBAS-Modus mit 0,6 m angegeben. Ausgehend von diesen Daten werden folgende Werte festgelegt:

$$l_{Teilstrecke} = 10\ m$$
$$l_{Teilstrecke_{Maximal}} = 15\ m$$

6.1.5 Geschwindigkeitsmittelwertabweichung

Die Geschwindigkeitsmittelwertabweichung $V_{Mittelabweichung}$ gibt an, wie weit die einzelnen Geschwindigkeiten der 10 m Teilstrecke vom quadratischen Mittel abweichen dürfen, damit die Geschwindigkeit in der Teilstrecke als konstant gewertet wird. Im Bereich der Innenstadt

ist es durch das hohe Verkehrsaufkommen und die zahlreichen Ampeln kaum möglich, mit konstanter Geschwindigkeit zu fahren.

In Abbildung 19 wurde eine Testfahrt auf der TAVF hinsichtlich der Fahrzeuggeschwindigkeit analysiert. Dazu wurde die Testfahrt in 10 Sekunden Intervalle eingeteilt und die

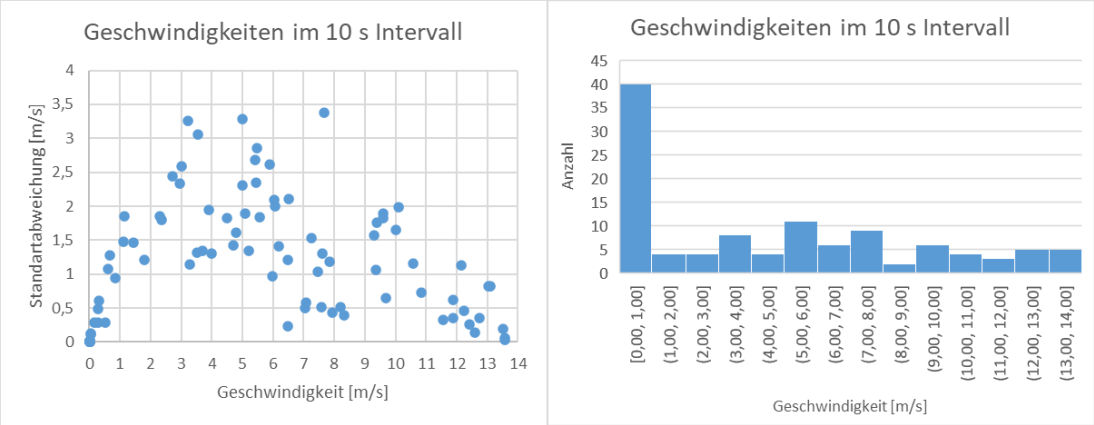


Abbildung 19: Geschwindigkeitsverteilung einer Testfahrt

Durchschnittsgeschwindigkeit und Standardabweichung ermittelt. Je größer die Standartabweichung in einem Intervall ist, desto inkonstanter ist die Geschwindigkeit in diesem Intervall. Ab ca. 6 m/s ist zu erkennen, dass sich einige Intervalle eine Standartabweichung von ca. 0,5 m/s hat. Da die Standardabweichung die mittlere Abweichung angibt, aber jeder punkt vergleichen wird gilt:

$$V_{Mittelabweichung} = \frac{Standardabweichung}{2} + 0{,}2\frac{m}{s} = \frac{0{,}5\frac{m}{s}}{2} + 0{,}2\frac{m}{s} = 0{,}45\ \frac{m}{s}$$

Die 0,2 m/s sind eine zusätzliche Toleranz, um die Spitzenwerte abzufangen.

6.1.6 Korrekturfaktorfunktion

Zur Ermittlung der Korrekturfaktorfunktion wird zunächst ein Verhältnis zwischen Fahrzeuggeschwindigkeit und der entstehen Beschleunigungen durch die Straßenqualität bestimmt. Hierfür wurde eine Straße mit guter Oberfläche und eine Straße mit schlechter Oberfläche mehrmals mit verschiedenen Geschwindigkeiten abgefahren. Die Geschwindigkeit wurde dabei möglichst konstant gehalten. Für diese Ermittlung wurde die Straße „Berliner Tor“ (Kopfsteinpflaster) und die Straße „Bei der Hauptfeuerwache“ (Asphaltstraße) genutzt. Aus den Messdaten wurden Fahrszenarien herausgefiltert, welche die Beschleunigungen unabhängig von der Straßenqualität beeinflussen. Zu jeder gefahrenen Geschwindigkeit wurde die passende Straßenqualität über den RMS bestimmt.
In Abbildung 20 ist das Ergebnis der Messung zu sehen. Es lässt sich sehr gut ein linearer Zusammenhang zwischen Straßenqualität und Fahrzeuggeschwindigkeit erkenne. Es wird die

Annahme getroffen, dass dieses Verhältnis zwischen 10 km/h und 60 km/h gültig ist. Eine Linearität bis 45 km/h wurde bereits bewiesen [13]. Somit ergeben sich folgende Funktionen:

$$f_1 = 0{,}0555 \cdot V_{Teilstück} + 0{,}618$$
$$f_2 = 0{,}0176 \cdot V_{Teilstück} + 0{,}173$$

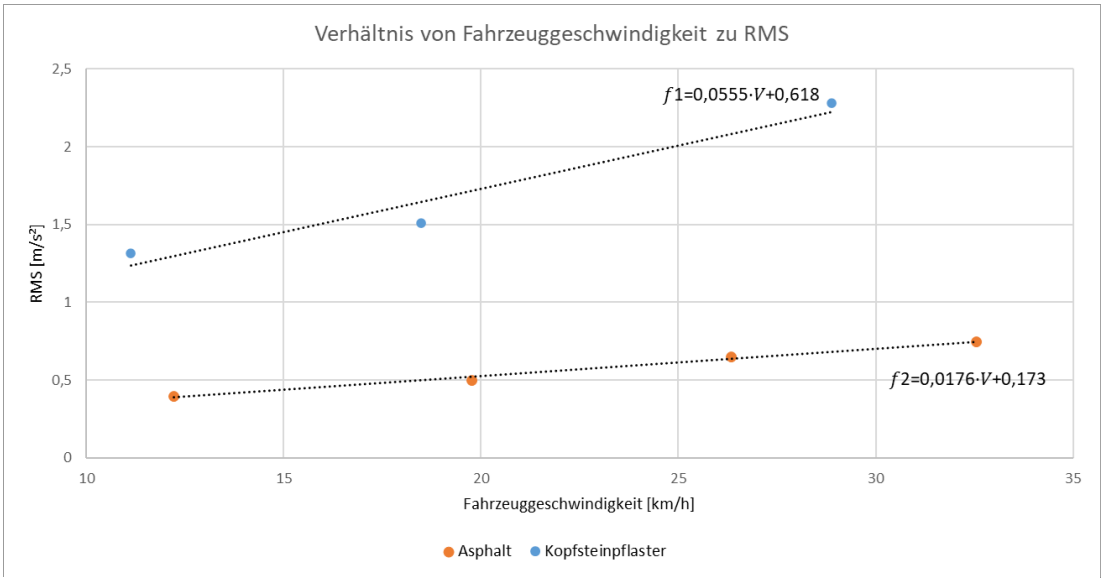


Abbildung 20: Verhältnis von Fahrzeuggeschwindigkeit zu RMS

Um nun den RMS_{50} zu ermitteln wird die Annahme getroffen, dass es für jede Straßenqualität eine passende Funktion gibt, die ebenfalls linear verläuft. Diese Funktionen sind in Abbildung 21 orange dargestellt. Um nicht jede Funktion messtechnisch zu ermitteln, kann über das Verhältnis zwischen f1 und f2 jede beliebige Korrekturfaktorfunktion ermittelt werden. Voraussetzung hierfür ist, dass sich die Starßenqualitäten von f1 und f2 stark unterscheiden.
Angenommen, es ist die $V_{Teilstrecke}$, $RMS_{Teilstrecke}$, *f1* und *f2* gegeben, so kann der passende RMS_{50} mithilfe folgenden Verhältnisses bestimmt werden:

$$\frac{f_1(V_{Teilstrecke}) - RMS_{Teilstrecke}}{RMS_{Teilstrecke} - f_2(V_{Teilstrecke})} = \frac{f_1(50) - RMS_{50}}{RMS_{50} - f_2(50)}$$

Aufgelöst und umgestellt ergibt sich für RMS_{50} folgendes:

$$RMS_{50} = \left(\frac{RMS_{Teilstrecke} - f_1(V_{Teilstrecke})}{f_1(V_{Teilstrecke}) - f_2(V_{Teilstrecke})}\right) \cdot \left(f_1(50) - f_2(50)\right) + f_1(50)$$

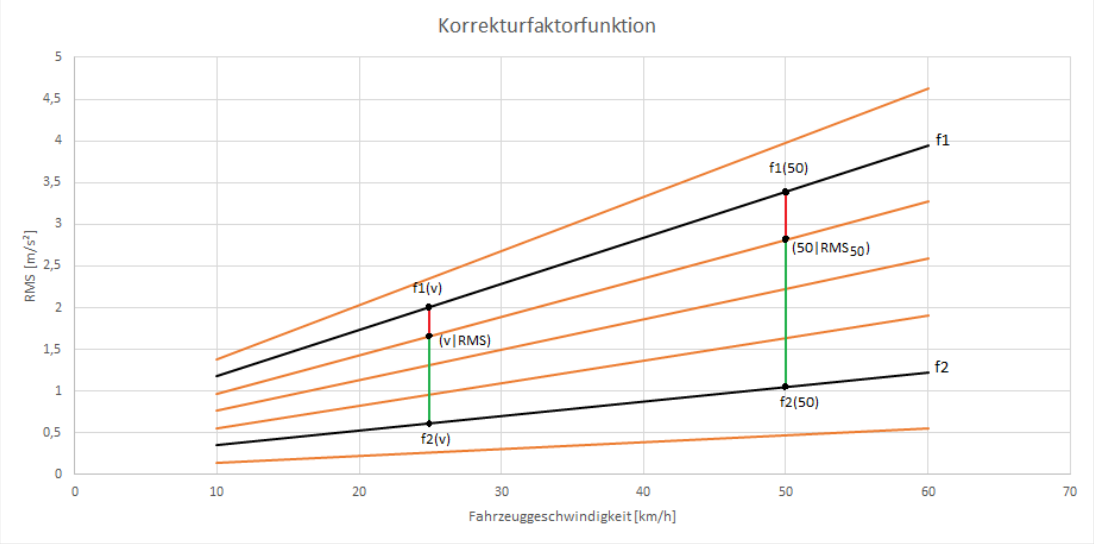


Abbildung 21: Korrekturfaktorfunktionen für das Tatfahrzeug

6.1.7 Ruck-Faktor

Der Faktor F_{Ruck} gibt an, um wie viel höher der Ruck im Messpunkt zum RMS-Ruck der Teilstrecke sein muss, damit ein Event erkannt wird. Zunächst werden verschieden Events analysiert. In Abbildung 22 ist der Ruck von drei charakteristischen Events bei verschiedenen Geschwindigkeiten dargestellt und zusätzlich der Ruck beim Überfahren einer Kopfsteinpflasterstraße.

Beim der Kanalschachtabdeckung und den Dehnungsfugen sind die Events klar erkennbar. Bei der Straße mit Flicken im Asphalt existieren durch die hohe Anzahl an Flicken auch viele Events. Bei der Kopfsteinpflasterstraße können Events aufgrund der Straßenoberfläche nicht genau bestimmt werden.

Diese Events wurden so ausgewählt, dass sie den minimalen Ruck für dieses Event wiedergeben. Der Ruck eines Schlagloches verhält sich in den untersuchten Fällen ähnlich zu dem einer Kanalschachtabdeckung und wurde daher nicht extra aufgeführt.

Der Faktor $j_{Schwellwert}$ wird so ausgelegt, dass Events sicher erkannt werden, aber nicht zu viele fehlerhafte Events beim Überfahren einer sehr rauen Straße erzeugt werden. Daher gilt:

$$F_{Ruck} = 2{,}5$$

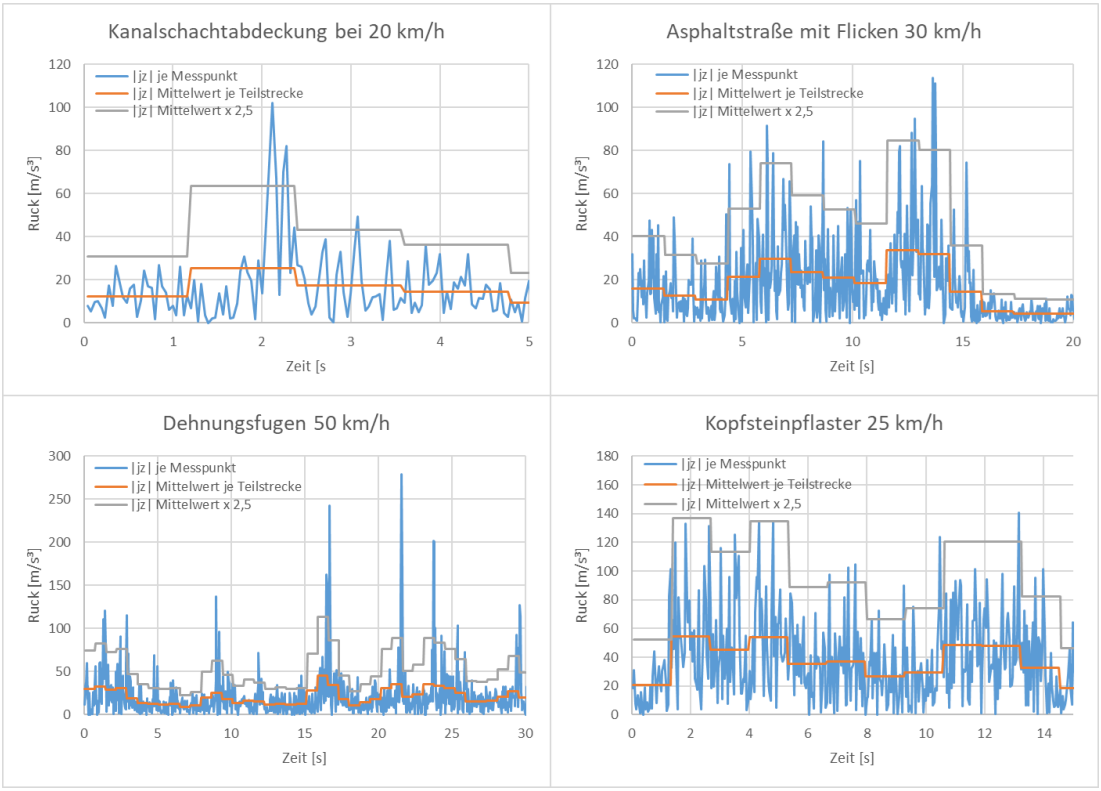


Abbildung 22: Eventanalyse für das Testfahrzeug

6.2. Fahrrad

In diesem Kapitel werden die benötigen Parameter und Funktionen für den Betrieb der Messeinrichtung auf dem Fahrrad entwickelt. Zum Teil orientieren sich einige davon an den Parametern des Testfahrzeuges, um eine Vergleichbarkeit zu gewährleisten.
In der nachfolgenden Tabelle befindet sich eine Übersicht aller Parameter, die in diesem Kapitel bestimmt werden.

Parameter	Wert
Offset X-Achse	0,0609 m/s²
Offset Y-Achse	-0,0607 m/s²
Offset Z-Achse	0,7022 m/s²
Minimalgeschwindigkeit	3 m/s
Maximalgeschwindigkeit	15 m/s
Länge der Teilstrecke	10 m
Maximale Länge der Teilstrecke	18 m
Geschwindigkeitsmittelwertabweichung	0,55 m/s
Korrekturfaktorfunktion	f1=1,144*V+0,8892; f2=0,0750*V+17,291
Ruck Faktor	4

Tabelle 26: Parameter für das Fahrrad

6.2.1 Offset ICM-20948

Zum Ermitteln des Offsets der Beschleunigungswerte vom ICM-20948 wird die Messeinrichtung auf einen graden Untergrund gelegt. In Abbildung 17, Abbildung 18 und Abbildung 19 ist der Verlauf der Messung und ein Histogramm der Messwerte dargestellt. Zur Offset-Bestimmung wird der Mittelwert für jede Achse über den Messzeitraum bestimmt. Für die X und Y-Achse ist der Offset, der Mittelwert mit umgekehrten Vorzeichen. Für die Z-Achse muss die Gravitationszone beachtet werden. Diese gibt für Hamburg eine Erdbeschleunigung von -9,8130 m/s² vor [17]. Daher ergibt sich für den Offset der Z-Achse:

$$a_{z_{Offset}} = -9{,}8130\ \frac{m}{s^2} - \left(-9{,}1108\ \frac{m}{s^2}\right) = 0{,}7022\ \frac{m}{s^2}$$

Beschleunigung	X-Achse	Y-Achse	Z-Achse
Mittelwert	0,0609 m/s²	0,0607 m/s²	-9,1108 m/s²
Offset	-0,0609 m/s²	-0,0607 m/s²	0,7022 m/s²
Standartabweichung	0,0447 m/s²	0,0435 m/s²	0,0570 m/s²

Tabelle 27: ICM-20948 Offset und Standartabweichung der XYZ-Beschleunigungen

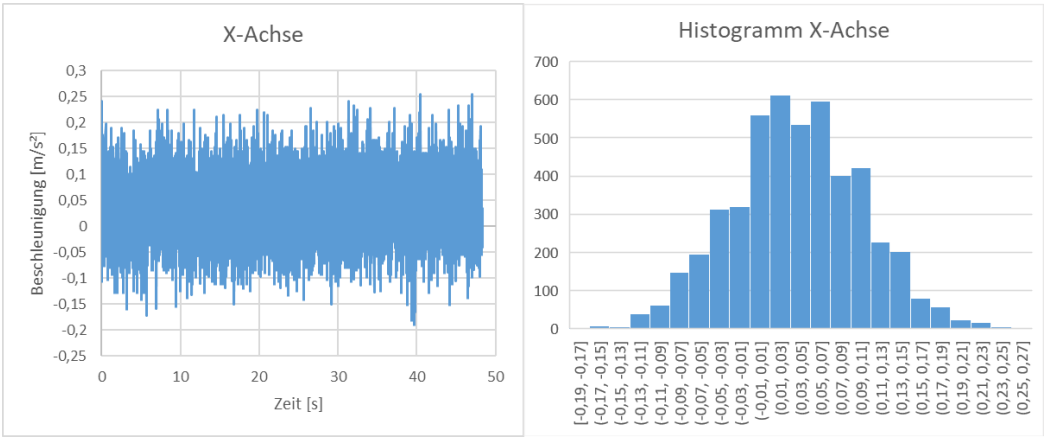


Abbildung 23: Offset X-Achse ICM-20948

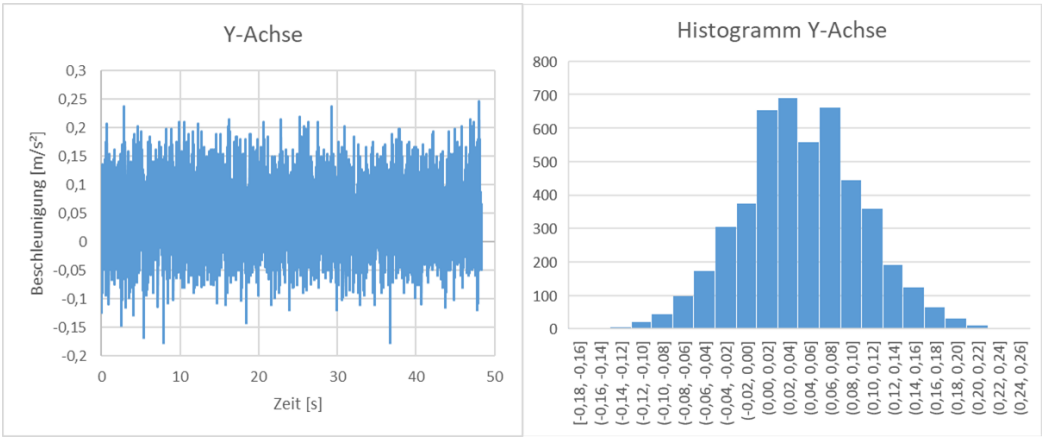


Abbildung 24: Offset Y-Achse ICM-20948

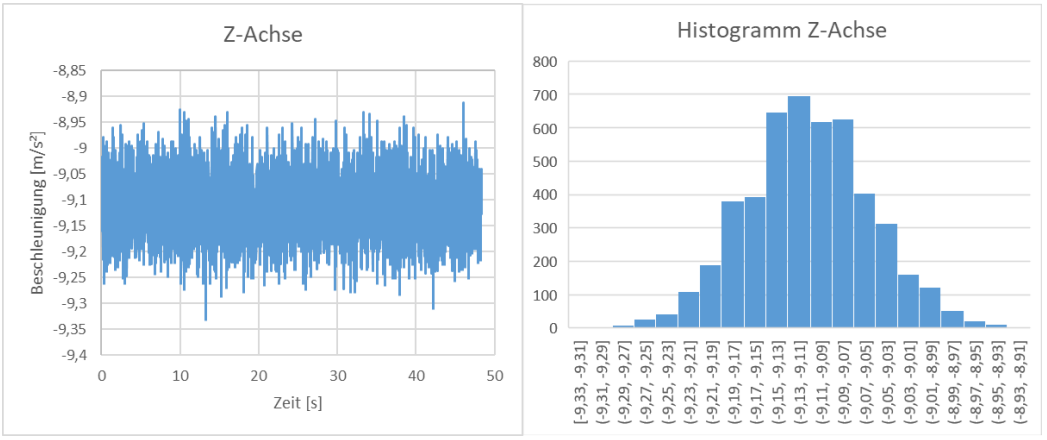


Abbildung 25: Offset Z-Achse ICM-20948

6.2.2 Minimalgeschwindigkeit

Eine Minimalgeschwindigkeit wird benötigt, da eine Bewegung des Fahrrades für den Algorithmus eine Voraussetzung ist. Mit dem Fahrrad ist es nach eigenen Testfahrten auf gerader Strecke möglich, etwa 10 km/h als minimale Geschwindigkeit zu fahren. Zudem verhält sich die Geschwindigkeit unter 10 km/h nicht linear zu der Beschleunigung in Z-Richtung [13]. Dieses ist wichtig für die Umsetzung Korrekturfaktorfunktion. Daher gilt für die Minimalgeschwindigkeit:

$$V_{min} = 3\ \frac{m}{s} = 10{,}8\ \frac{km}{h}$$

6.2.3 Maximalgeschwindigkeit

Die Maximalgeschwindigkeit V_{Max} ist jene Geschwindigkeit ab wann keine Messdaten mehr vom Algorithmus aufgezeichnet und verarbeitet werden. Eine Maximalgeschwindigkeit wird beim Fahrrad prinzipiell nicht benötigt da es mit dem zur Verfügung gestellten Fahrrad kaum möglich ist, über 30 km/h zu fahren. Damit keine fehlerhaften Daten aufgezeichnet werden, falls dennoch höhere Geschwindigkeiten erzielt werden, gilt hier wie im Testfahrzeug:

$$V_{max} = 15\ \frac{m}{s} = 54\ \frac{km}{h}$$

6.2.4 Länge und maximale Länge der Teilstrecke

Wie auch beim Testahrzeug entsteht eine Überschneidung der Beschleunigungswerte beim Einteilen der Straße in Teilstrecken. Um einen Vergleich mit den Messwerten des Testfahrzeuges zu haben, wird die Länge der Teilstrecke ebenfalls auf 10 m festgelegt. Bei einem Radstand von ca. 1 m ist folgende Bedingung erfüllt:

$$l_{Teilstrecke} \geq 3 \cdot l_{Radstand} = 3 \cdot 1\ m = 3\ m$$

Abhängig von der Messfrequenz und der Minimal- und Maximalgeschwindigkeit ergeben sich folgende minimalen und maximalen Distanzen zwischen zwei GPS-Koordinaten:

$$\Delta l_{min} = \frac{f_{GNSS\ Data}}{V_{min}} = \frac{3\frac{m}{s}}{5\ Hz} = 0{,}6\ m$$

$$\Delta l_{max} = \frac{f_{GNSS\ Data}}{V_{max}} = \frac{15\frac{m}{s}}{5\ Hz} = 3\ m$$

Die horizontale Genauigkeit der GPS-Position Genauigkeit des Sparkfun Venus GPS-Moduls wird mit 2,5 m angegeben. Ausgehend von diesen Daten werden folgende Werte festgelegt:

$$l_{Teilstrecke} = 10\ m$$

$$l_{Teilstrecke_{Maximal}} = l_{Teilstrecke} + \Delta l_{max} + 2 \cdot GPS_{Toleranz} = 18\,m$$

6.2.5 Geschwindigkeitsmittelwertabweichung

Zum Ermitteln der Geschwindigkeitsmittelwertabweichung wurden die gefahrenen Geschwindigkeiten einer Testfahrt analysiert. Die Geschwindigkeit wurde dabei der Standardabweichung innerhalb eines 10 s Intervalls gegenübergestellt. In Abbildung 26 sind die gefahrenen Geschwindigkeiten dargestellt. Gut zu erkennen ist, dass die meisten Geschwindigkeiten im Bereich zwischen 3 m/s und 5 m/s liegen. Die Standardabweichung in diesem Bereich, bei denen die Geschwindigkeit annähernd konstant gehalten wurde, beträgt im Durchschnitt ca. 0,7 m/s. Da die Standardabweichung die mittlere Abweichung nach oben und unten angibt, gilt:

$$V_{Mittelabweichung} = \frac{Standartabweichung}{2} + 0{,}2\frac{m}{s} = \frac{0{,}7\frac{m}{s}}{2} + 0{,}2\frac{m}{s} 0{,}55\ \frac{m}{s}$$

Die 0,2 m/s sind eine zusätzliche Toleranz, um die Spitzenwerte abzufangen.

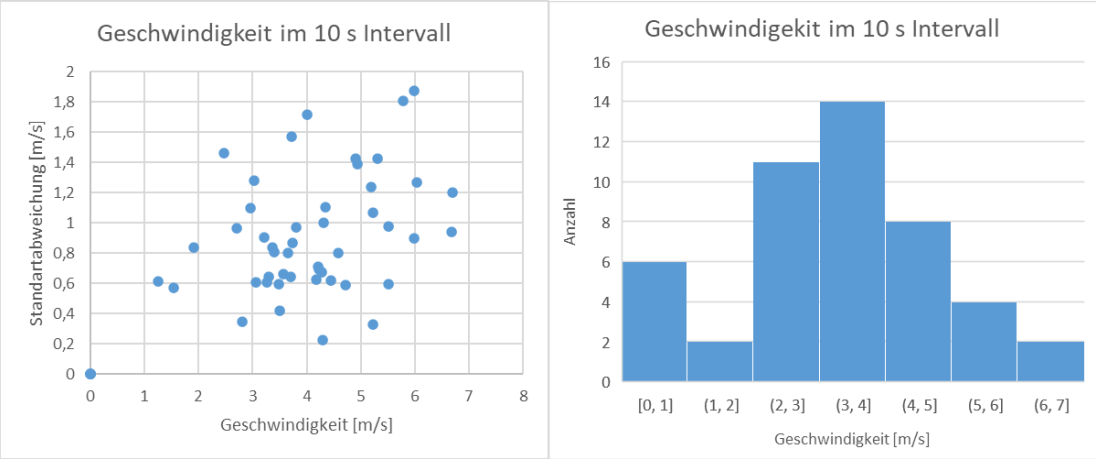


Abbildung 26: Geschwindigkeitsverteilung einer Testfahrt

6.2.6 Korrekturfaktorfunktion

Zur Entwicklung der Korrekturfaktorfunktion für das Fahrrad wird gleich vorgegangen wie beim Testfahrzeug (vgl. Kapitel 6.1.6). Hierfür wurde eine Straße mit guter Oberfläche und eine Straße mit schlechter Oberfläche mehrmals mit verschiedenen Geschwindigkeiten abgefahren. Die Geschwindigkeit wurde dabei möglichst konstant gehalten. Für die Ermittlung der Korrekturfaktorfunktion wurde ebenfalls die Straße Berliner Tor (Kopfsteinpflaster) und die Straße Bei der Hauptfeuerwache (Asphaltstraße) genutzt. In Abbildung 27 werden die gemessenen Punkte dargestellt. Dadurch ergeben sich folgende Funktionen:

$$f_1 = 1{,}511 \cdot V_{Teilstück} - 3{,}662$$
$$f_2 = 0{,}431 \cdot V_{Teilstück} - 1{,}973$$

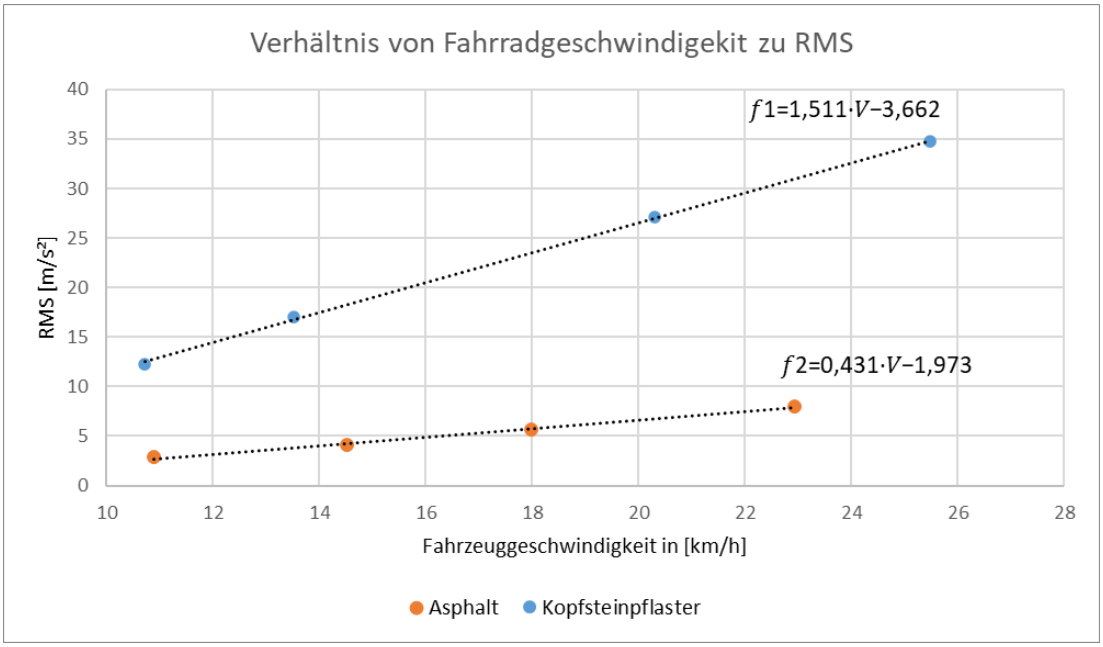


Abbildung 27: Verhältnis von Fahrradgeschwindigkeit zu RMS

Auch hier kann über die Anwendung des Strahlensatzes jede weitere Korrekturfunktion ermittelt werden. Es gilt somit:

$$RMS_{50} = \left(\frac{RMS_{Teilstrecke} - f_1(V_{Teilstrecke})}{f_1(V_{Teilstrecke}) - f_2(V_{Teilstrecke})}\right) \cdot \left(f_1(50) - f_2(50)\right) + f_1(50)$$

6.2.7 Ruck-Faktor

Der Faktor F_{Ruck} gibt an, um wie viel höher der Ruck im Messpunkt zum RMS-Ruck der Teilstrecke sein muss, damit ein Event erkannt wird. Zunächst werden verschieden Events analysiert. Die in Abbildung 23 geziegelten Events repräsentieren, die am häufigsten vorkommenden Events bzw. den Ruck, der durch diese Events entstanden ist. Abbildung 23 zeigt ein Schlagloch bei ca. 11 km/h und mehrere Schlaglöcher bei etwa 23 km/h. Zudem wird die Fahrt von einem Radweg über einen Bordstein auf die Straße gezeigt. Dieser Bordstein ist nicht richtig eingeebnet und erzeugt als Event einen sehr großen Ruck. Ebenso ist eine Kopfsteinpflasterstraße dargestellt. Diese erzeugt aufgrund ihrer Bauweise durchgängig hohe Spitzenwerte im Ruck. Der Faktor F_{Ruck} sollte so ausgelegt werden, dass diese Spitzen

nicht als Event erkannt werden. Der Faktor darf auch nicht zu hoch ausgewählt werden, so dass keine kleineren Events übersehen werden. Daher gilt:

$$F_{Ruck} = 4{,}0$$

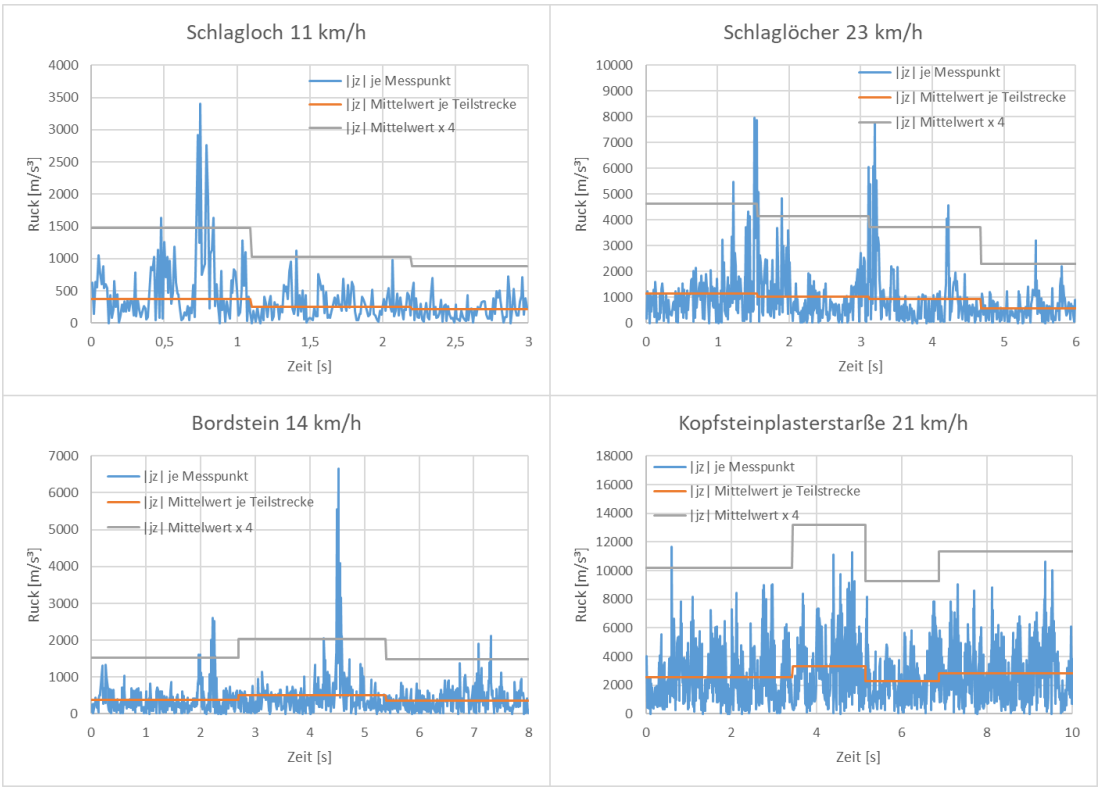


Abbildung 28: Eventanalyse für die Messeinrichtung im Fahrrad

7. Tests und Validierungen

In diesem Kapitel werden unterschiedliche Tests für das System im Testfahrzeug und die Messeinrichtung für den Einsatz am Fahrrad durchgeführt und bewertet. Es wird überprüft, wie gut die Qualität der Eingangsdaten ist und ob der Algorithmus richtig funktioniert. Es wird die Serververbindung getestet. Zudem wird überprüft ob zuvor festgelegten Anforderungen erfüllt werden.

7.1. Testfahrzeug

Im Testfahrzeug werden die Daten des SBG Systems Apogee-D überprüft. Der Algorithmus wird auf seine korrekte Funktion und Genauigkeit beim Bestimmen von Straßenqualitäten und Events geprüft.

7.1.1 Ablauf der Tests

Die Tests des Systems im Testfahrzeug wurden mithilfe von ROS-Bag Dateien durchgeführt. Durch das Aufsetzten eines ROS-Systems mit allen notwendigen ROS-Knoten auf einem lokalen Computer wurde der Tesla simuliert. Für die nachfolgenden Testes wurden zwei Testfahrten durchgeführt. Diese sind in Abbildung 29 und 30 zu sehen.

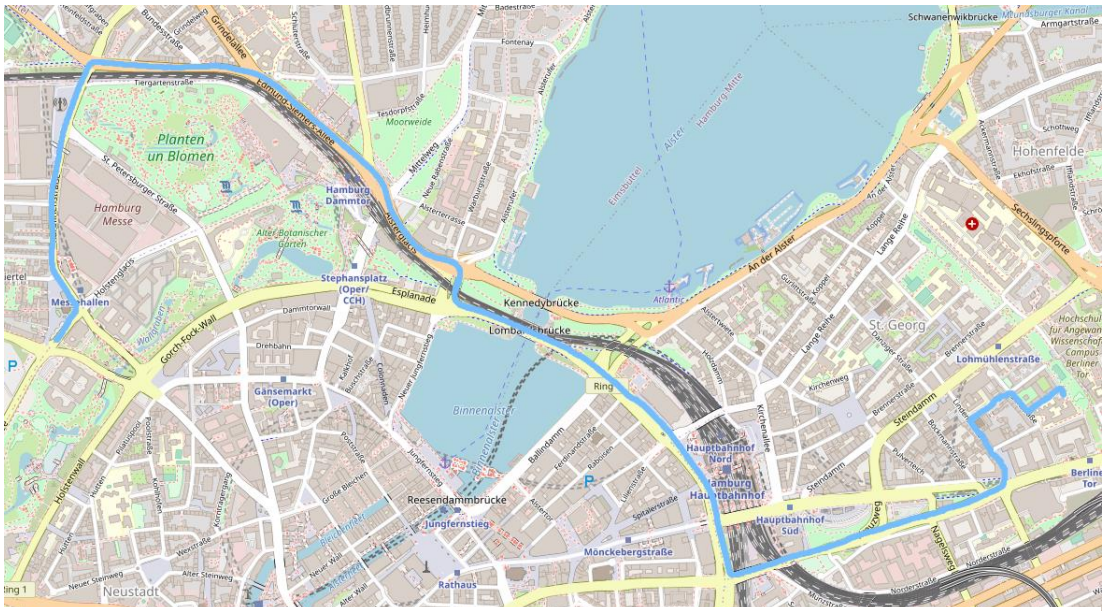


Abbildung 29: Testfahrt 1 im Tesla

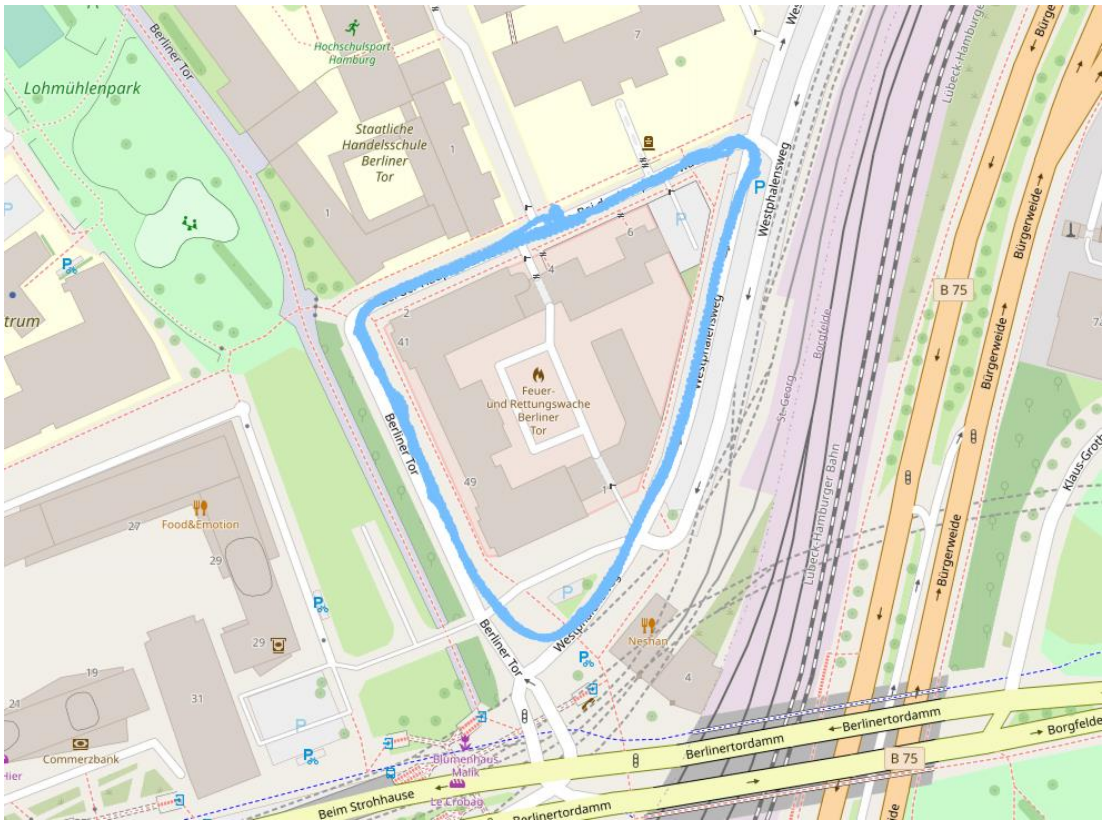


Abbildung 30: Testfahrt 2 im Tesla mit insgesamt 4 Runden

In folgender Tabelle ist dargestellt welche Testfahrt für welche Tests verwendet wurde.

Test	**Verwendete Testfahrt**
Daten Prüfen	Testfahrt 1
Ausnutzung der Daten	Testfahrt 1
Genauigkeit des Algorithmus	Testfahrt 2

Tabelle 28: Verwendung der Testfahrten

7.1.2 Daten überprüfen

Es werden zuerst die IMU-Daten überprüft. Zunächst wird geprüft, wie weit die eingestellte Messfrequenz von der tatsächlichen Messfrequenz abweicht. Hierzu wurde das Δt zwischen den einzelnen IMU-daten ermittelt. Bei einer Messfrequenz von 25 Hz sollte das Δt zwischen den Messungen 40 ms betragen. In Abbildung 31 ist das passende Histogramm dargestellt. Wie zu erkennen ist, gibt es kleinere Auffälligkeiten. Bei 175 IMU-Daten betrug das Δt zwischen den Nachrichten etwa 8 ms. Etwa 98,5 % der IMU-Daten haben eine Toleranz von 40±0,3 ms.

Etwa 1 % der IMU-Daten lag in einem Bereich von 40±5 ms und ca. 0,5 % der IMU-Daten wurden mit einem Δt von ca. 80 ms ausgesendet.

Als nächstes wird überprüft, in welchem Bereich die Beschleunigungswerte für die Z-Achse. liegen. In Abbildung 31 ist die Anzahl der Beschleunigungen der Z-Achse dargestellt. Die Werte lagen während der gesamten Testfahrt zwischen -3,92 m/s² und -15,64 m/s². Der SBG Systems Apogee-D hat einen Messbereich von ±10 g. Es gibt keine Messwerte, die außerhalb des Messbereiches liegen.

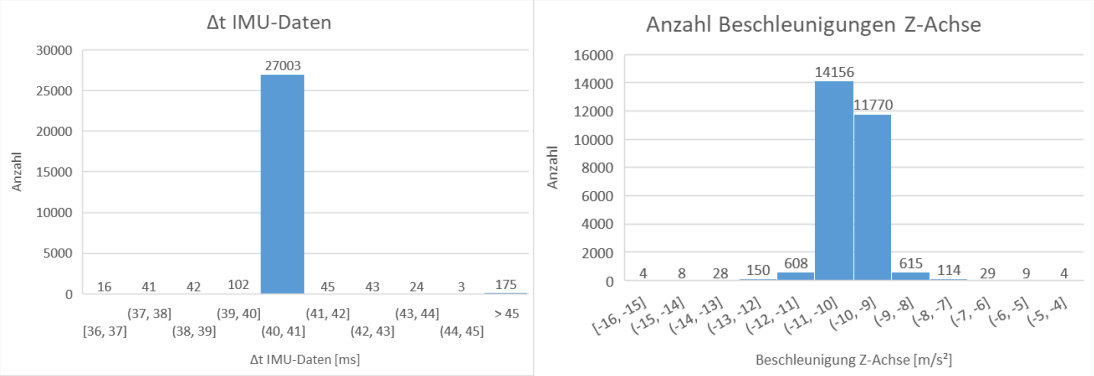


Abbildung 31: IMU-Daten in der Testfahrt 1

Des Weiteren werden die GNSS-Daten überprüft. Diese werden mit eine Messfrequenz von 5 Hz aufgezeichnet. Daher sollte das Δt zwischen den GNSS-Daten 200 ms betragen. In Abbildung 32 ist die Verteilung der Δt zwischen den GNNS-Daten in einem Histogramm dargestellt.

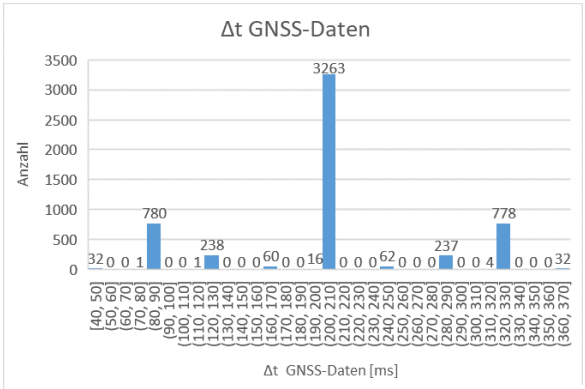


Abbildung 32: Δt der GNSS-Daten in der Testfahrt

Auffällig bei den GNSS-Daten ist die große Anzahl an GNSSS-Daten, die nicht mit der richtigen Messfrequenz aufgezeichnet wurden. Es wurde festgestellt, dass so bald eine Δt Zwischen zwei GNSS-Daten zu groß ist, ist das darauffolgende Δt zu klein. Dabei ergibt die Summe dieser Δts immer ca. 400 ms. Also wird versucht, das fehlerhafte Δt zu korrigieren. Es lassen sich drei Wertepaare erkennen, bei denen dieses Phänomen auftritt. Ein Zusammenhang zwischen diesem Phänomen und der Fahrzeuggeschwindigkeit wurde nicht festgestellt.

Es wird zudem die GPS-Position während der Messfahrt überprüft. Hierbei wird verglichen wie weit die GPS-Positionen von der Straße abweicht. Dabei gibt es keine größeren Auffälligkeiten. Es gibt typische Abweichung der GPS-Position wie in Abbildung 33 dargestellt. Diese entstehen in Häuserschluchten durch die Reflexion der GPS-Signale an den Häuserfronten.

Abbildung 33: Abweichung der GPS-Position

Fazit: Die IMU-Daten entsprechen den Erwartungen und können ohne Einschränkung für die Klassifizierung genutzt werden. Die Verzögerung der GNSS-Daten kann in bestimmten Fällen zu einer Verlängerung der Teilstrecke führen. Bei einer Geschwindigkeit von 50 km/h und einer Verzögerung von 320 ms beträgt die zurückgelegt Distanz 4,4 Meter. Tritt eine solche Verzögerung am Ende eine Teilstrecke auf, kann es unter Umständen dazu führen, dass die maximale Länge der Teilstrecke überschritten wird.

7.1.3 Nutzung der Daten

Die gesamte Testfahrt hatte eine Länge von etwa 5,2 km. Es wurden ca. 1,7 km der Strecke durch 154 Teilstrecken für die Klassifizierung der Straße genutzt. In Abbildung 34 sind die Positionen dargestellt, an denen eine Teilstrecken zur Klassifizierung gebildet wurden.

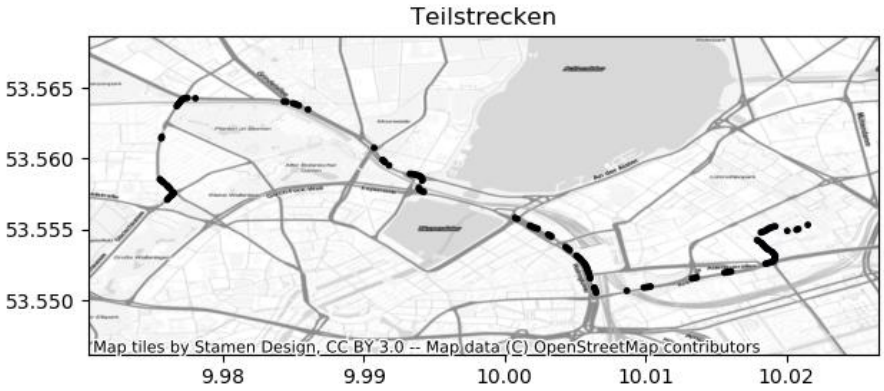


Abbildung 34: Teilstrecken der Messfahrt

In Abbildung 35 sind die Längen der Teilstrecken und die Verteilung der Teilstrecken abhängig von der Fahrzeuggeschwindigkeit dargestellt. Die meisten Teilstrecken besitzen eine Länge im Bereich von 10 m bis 12 m. Die Länge der Teilstrecke ist dabei unabhängig von der Fahrzeuggeschwindigkeit.

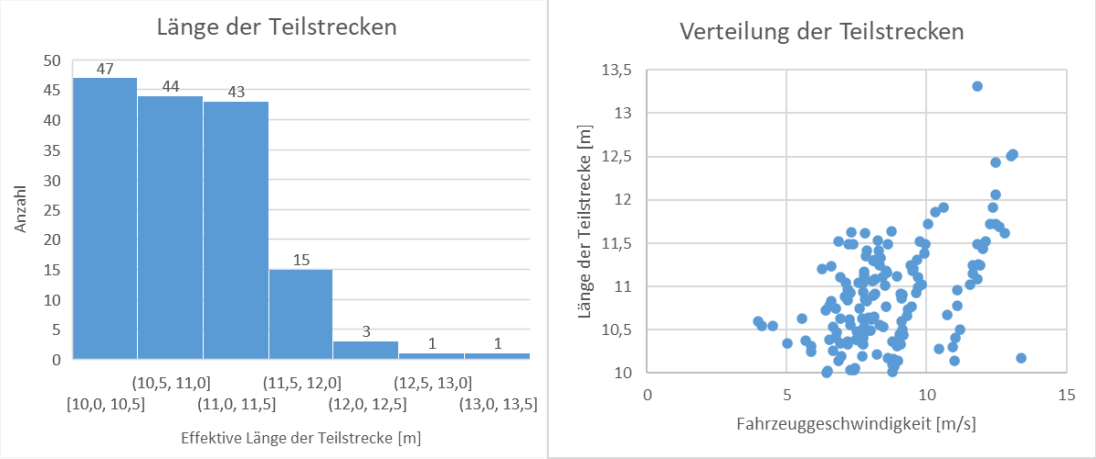


Abbildung 35: Länge der Teilstrecke in der Testfahrt 1

In Abbildung 36 ist die Anzahl der IMU-Messpunkte innerhalb einer Teilstrecke und die durchschnittliche Distanz zwischen zwei aufgenommenen IMU-Messpunkten dargestellt. Mit zunehmender Geschwindigkeit erhöht sich die Distanz zwischen den IMU-Messpunkten. Bei

einer Geschwindigkeit von 12 m/s beträgt die Distanz zwischen zwei IMU-Messpunkten im Durchschnitt etwa 0,6 Meter.

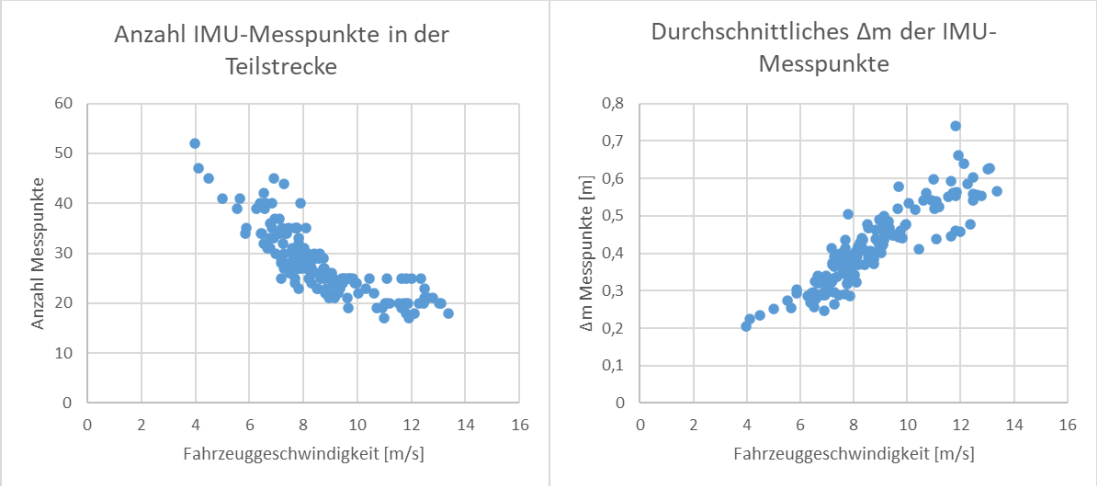


Abbildung 36: IMU-Messpunkte in der Testfahrt 1

Fazit: Durch das einmalige Abfahren einer Strecke mit dem Auto wird nur ein kleiner Teil zur Klassifizierung genutzt. Abhängig ist dieses vom Verkehrsaufkommen und der Anzahl an Ampeln. Hierdurch wird beim Fahren das Halten eine konstante Geschwindigkeit erschwert. Bei höheren Geschwindigkeiten ist die Distanz zwischen zwei IMU-Mespunkten groß. Dieses kann unter Umständen dazu führen, dass Events nicht erkannt werden oder die ermittelte Straßenqualität nicht repräsentativ für den Straßenabschnitt ist.

7.1.4 Genauigkeit des Algorithmus

Um die Genauigkeit des Algorithmus zu prüfen, wurden mehrere Fahrten um das Gebäude der Feuerwache am „Berliner Tor“ durchgeführt (Testfahrt 2). Auf dieser Testrecke sind eine Kopfsteinpflaster- und zwei Asphaltstraßen vorhanden. Auf den Asphaltstraßen befinden sich Events in Form von Kanalschachtabdeckungen und Flicken auf der Straße.

Zunächst wird die Ermittlung der Straßenqualität geprüft. Hierfür wurden vier Runden auf der oben beschrieben Teststrecke mit gleichbleibenden Geschwindigkeiten gefahren. Die Straßenqualitäten sind durch den RMS50 in Abbildung 37 und 38 dargestellt.

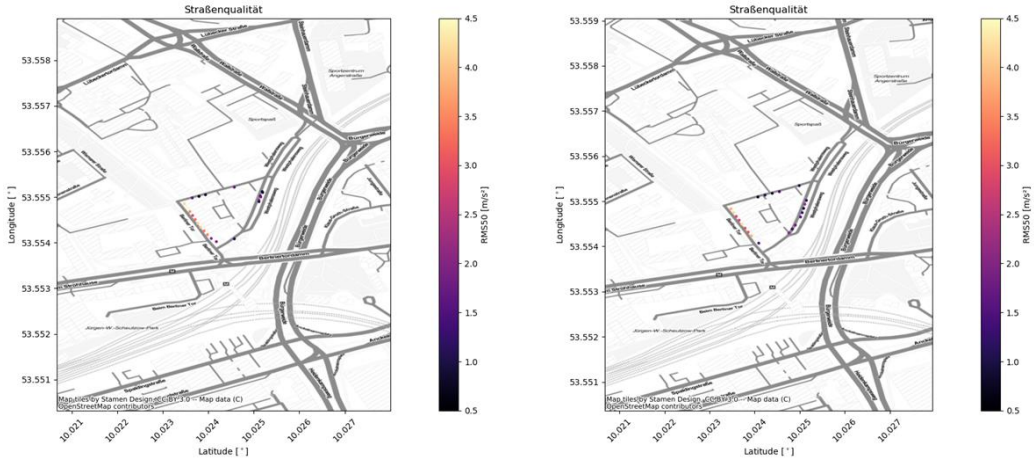


Abbildung 37: Straßenqualität der Testfahrt 2 Runde 1 und 2

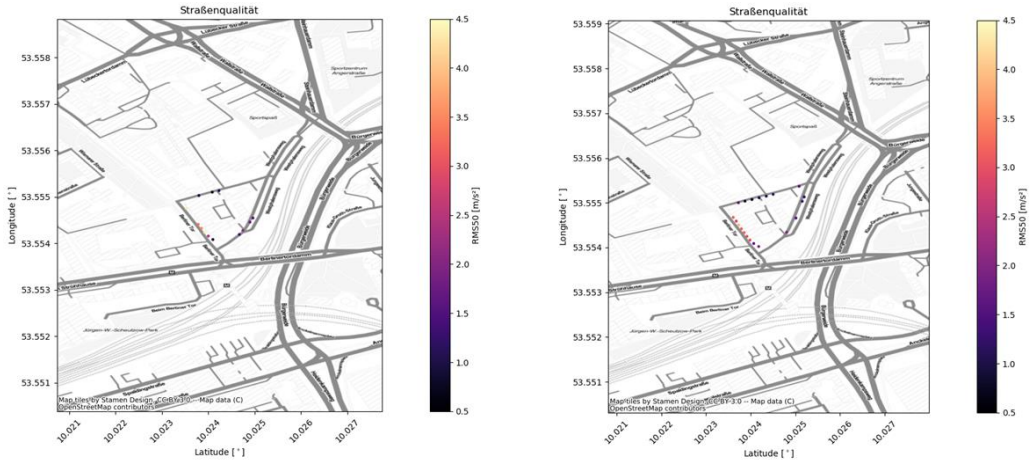


Abbildung 38: Straßenqualität der Testfahrt 2 Runde 3 und 4

Zu beachtet ist, dass nicht die gesamte Strecke der Testfahrt zur Klassifizierung genutzt wird, da zu einigen Zeitpunkten die Mindestgeschwindigkeit nicht erreicht wurde oder die Geschwindigkeit in der Teilstrecke nicht konstant gehalten wurde. Gut zu erkennen ist die Kopfsteinpflasterstraße am Berliner Tor durch den hohen RMS50. Dieser liegt trotz gleichbleibender Geschwindigkeiten zwischen 3,1 m/s² und 4,3 m/s². Der größte Teil der gefahrenen Geschwindigkeit auf der Kopfsteinpflasterstraße lag dabei in einem Bereich von 18,5 km/h bis 20,5 km/h.

Auf den Asphaltstraßen liegt die Straßenqualität bei gleichbleibender Geschwindigkeit zwischen 1,2 m/s² und 2,5 m/s². Der größte Teil der Geschwindigkeiten lag hierbei in einem Bereich von 23 km/h bis 26 km/h.
Es lassen sich sehr gut Bereiche erkennen, an denen sich die Straßenqualität verändert. So lässt sich ein Bereich von etwa 20 m auf der Straße „Berliner Tor“ erkennen, welcher etwa in der Mitte liegt, an dem die Kopfsteinpflasterstraße einen geringeren RMS50 und somit eine höhere Straßenqualität hat.
Bei den Asphaltstraßen sind solche Bereiche ebenfalls zu erkennen. Auffällig ist hierbei die Straße „Westphalensweg“ vor der Ausfahrt der Feuerwache. Hier wird ein erhöhter RMS50, im Gegensatz zu den anderen Teilen der Straße, gemessen. Bei einer optischen Betrachtung der Straße in diesem Bereich wurde festgestellt, dass der Asphalt hier stark abgenutzt ist.
Besonders gut sind diese Bereiche zu erkennen, wenn die Messdaten der einzelnen Runden in einer Karte übereinandergelegt werden, da so die fehlenden Bereiche ergänzt werden und ein Gesamtbild entsteht. Dieses ist dargestellt in Abbildung 39.

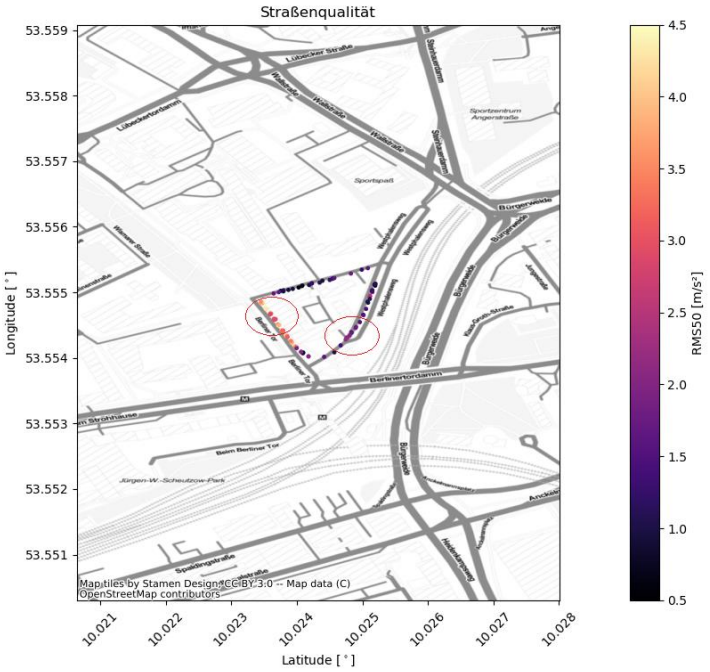


Abbildung 39: Überlagerte Straßenqualitäten der Testfahrt 2

Zur Betrachtung der Events werden zunächst die einzelnen gefunden Events der jeweiligen Runden betrachtet. Im Vorfeld wurden Events auf der Straße „Bei der Hauptfeuerwache“ durch eine optische Betrachtung herausgesucht. In Abbildung 40 und 41 sind die gefunden Events der einzelnen Runden dargestellt.

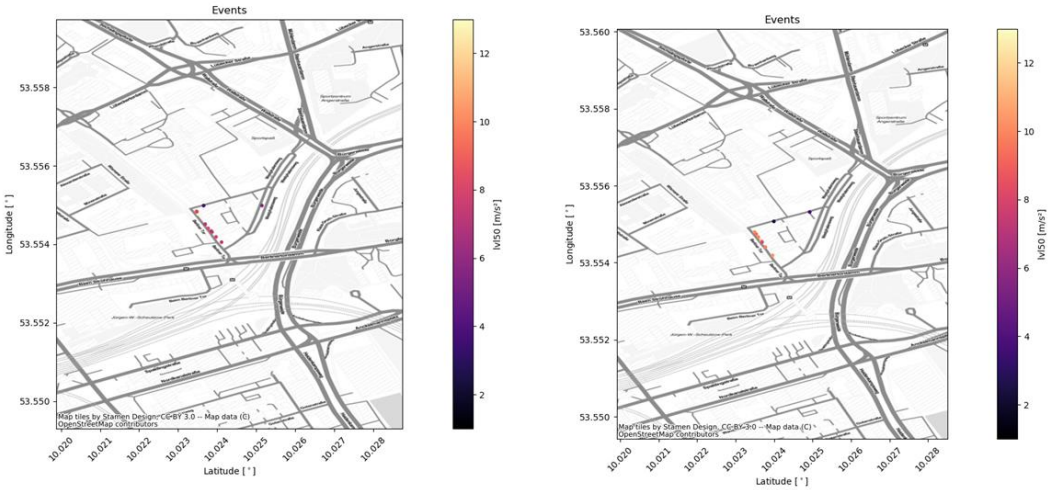


Abbildung 40: Events der Testfahrt 2 Runde 1 und 2

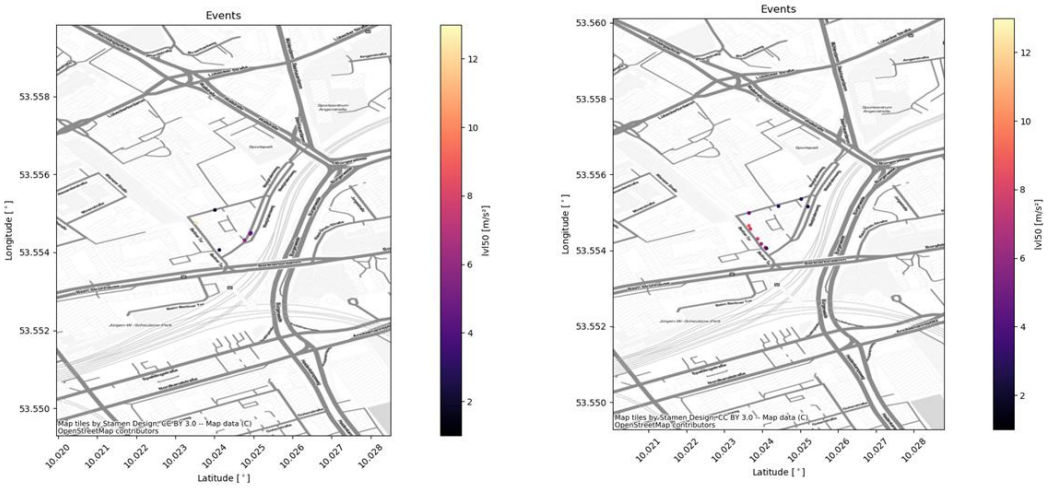


Abbildung 41: Events der Testfahrt 2 Runde 3 und 4

Auffällig hierbei ist, dass in drei von vier Runden viele Events auf der Kopfsteinpflasterstraße gefunden wurden. Ein Event auf eine Kopfsteinpflasterstraße gestaltet sich als schwierig zu definieren. Dennoch ist dieses ein Hinweis darauf, dass der Faktor F_{Ruck} zu gering gewählt wurde. Auf der Straße „Bei der Hauptfeuerwache“ wurden im Vorfeld vier Events durch eine optische Betrachtung der Straße herausgesucht, die sicher erkannt werden sollten. Diese

Events sind gleichmäßig über die Straße verteilt. Die Entfernung der Events kann zwischen den beiden mittleren Events dazu führen, dass diese aufgrund der Totzeit nicht gefunden werden. Bei den Events handelt es sich um Kanalschachtabdeckungen und Flicken auf der Straße. Nach einer Analyse der Daten wurde festgestellt, dass alle möglichen Events in diesem Abschnitt gefunden wurden. Durch eine nicht gleichbleibende Geschwindigkeit und die Totzeit konnte in einigen Runden kein Event gefunden werden. In Abbildung 42 ist die Überlagerung der Events dargestellt. Zu erkennen ist hierbei, dass die Position der beiden letzten Events etwas verstreut ist. Die beiden vorderen Events wurden mehrfach gut lokalisiert. Dieses Verhalten ist auf die Events zurückzuführen. Die beiden ersten Events sind sehr eindeutig. Die beiden letzten Events sind sehr verstreut, da die Flicken in der Straße über eine größere Fläche verstreut sind. Abhängig von der genauen Fahrspur werden die Events unterschiedlich lokalisiert. Zudem werden bei 25 km/h nur etwa alle 1,4 m eine GPS-Position

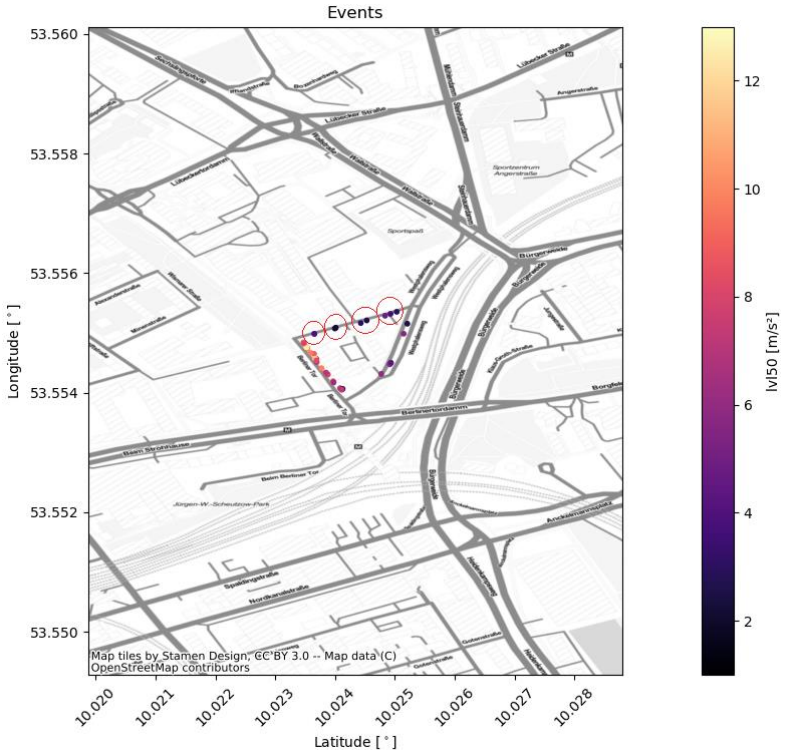


Abbildung 42:Überlagerung der Events der Testfahrt 2

empfangen.

Desweitern ist aufgefallen, dass zum Teil Events gefunden werden, an denen sich keine Events befinden. Dieses kommt vermehrt bei sehr glatten Straßen vor, da der Ruckmittelwert dort sehr gering ist. Daher reicht ein verhältnismäßig kleiner Ruck aus, um ein Event zu finden.

Fazit: Die Straßenqualität lässt sich sehr gut über den RMS50 beurteilen. Bereits kleinere Veränderungen in der Straßenoberfläche sind zu messen. Die Wiederholgenauigkeit bezogen auf die gleiche Straße kann nie genau gegeben werden, da die gleiche Strecke niemals exakt gleich abgefahren werden kann. Dennoch ist eine gewisse Genauigkeit erkennbar. Eine Beurteilung einer Straße kann nur durch mehrere Teilstrecken erfolgen. Dabei können die Teilstrecken von unterschiedlichen Testfahrten kombiniert werden. Aus den vorliegenden Messdaten ist der RMS50 für das Testfahrzeug wie folgt zu bewerten:

RMS50 [m/s²]	**Straßenqualität**	**Beschreibung**
<0,5 bis 1,5	gut	Gute Asphaltstraßen mit wenig Beschädigungen und einer glatten Oberfläche
1,5 bis 3,0	mittel	Schlechte Asphaltstraßen mit Beschädigungen und höher Oberflächen Rauheit.
3,0 bis >4,5	schlecht	Kopfsteinpflasterstraßen oder anderweitig gepflasterte Straßen

Tabelle 29: Kategorisierung des RMS50 beim Tesla

Das Auffinden eines Events wird durch den Algorithmus mit den eingestellten Parametern gut bewerkstelligt. Das einzelne auffinden eines Events ist dabei immer möglich. Das gleiche Event mehrmals aufzufinden ist dabei nicht immer möglich. Hierbei muss auch berücksichtig werden, ob das Event überhaupt überfahren wurde. Bei Teilstrecken mit einem geringen Ruck werden zum Teil Events fehlerhaft ermittelt. Es könnte daher sinnvoll sein, einen Minimalwert einzuführen, über dem ein Ruck mindestens liegen muss, um als Event gewertet zu werden. Eine Klassifizierung der Events über die Stärke findet nicht statt.

7.2. Messeinrichtung für den Einsatz im Fahrrad

Die durchzuführenden Tests für die Messeinrichtung beziehen sich auf die Hardware mit Hinblick auf die Anforderungen. Es wird die Qualität der IMU und GNSS-Daten überprüft. Der Algorithmus wird auf seine korrekte Funktion und Genauigkeit beim Bestimmen von Straßenqualitäten und Events geprüft.

7.2.1 Ablauf der Tests

Es wurden zwei Testfahrten durchgeführt. Dabei wurden die Messdaten in einer ROS-Bag Dateien aufgezeichnet. Durch die Verwendung eines lokalen Computers wurde die Messeinrichtung auf dem Fahrrad simuliert. Die Messfahrten sind in Abbildung 43 und Abbildung 44 dargestellt.

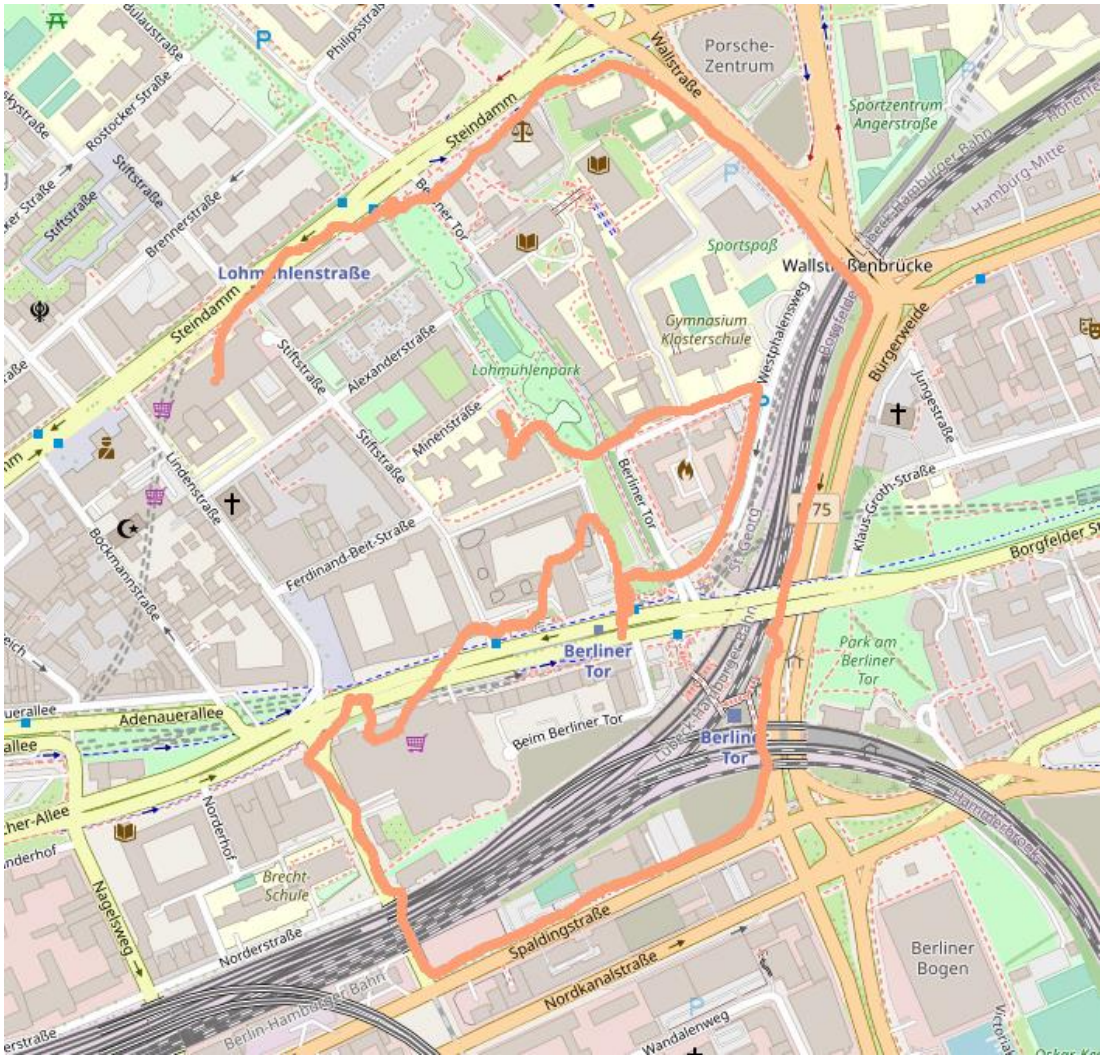


Abbildung 43: Testfahrt 1 mit der Messeinrichtung für das Fahrrad

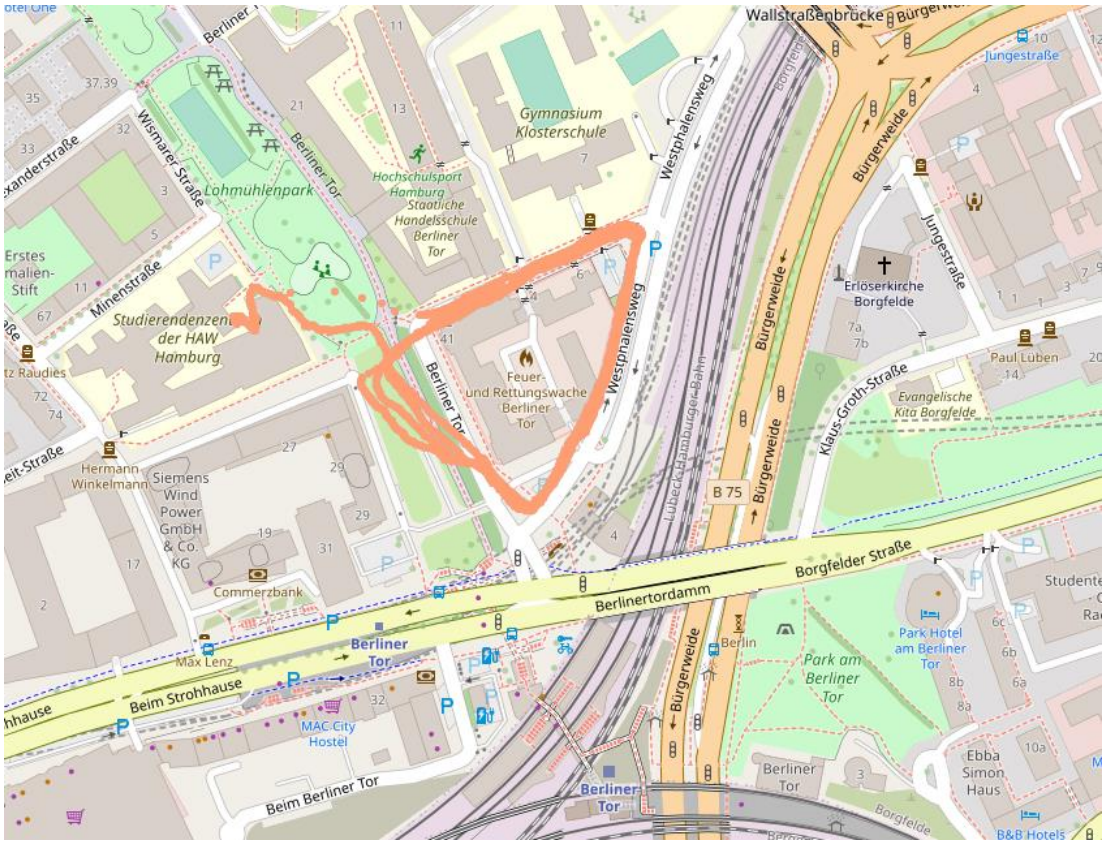


Abbildung 44: Testfahrt 2 mit der Messeinrichtung für das Fahrrad

In folgender Tabelle ist dargestellt welche Testfahrt für welche Tests verwendet wurde.

Test	**Verwendete Testfahrt**
Daten Prüfen	Testfahrt 1
Ausnutzung der Daten	Testfahrt 1
Genauigkeit des Algorithmus	Testfahrt 2

Tabelle 30: Verwendung der Testfahrten

7.2.2 Daten überprüfen

Für die Validierung der Daten wird zuerst überprüft, ob die eingestellte Messfrequenz der IMU-Daten von der tatsächlichen Messfrequenz abweicht. Der ICM-20948 ist auf eine Frequenz von 100 Hz eingestellt, welches einem Δt zwischen den IMU-Daten von 10 ms entspricht. In Abbildung 45 ist die Anzahl der Δt für die Testfahrt 1 dargestellt. Der größte Teil liegt hierbei bei 10 ms. Die größte Abweichung liegt im Bereich von 10±2 ms. Andere Abweichungen sind so gering, dass diese nicht berücksichtigt werden müssen.

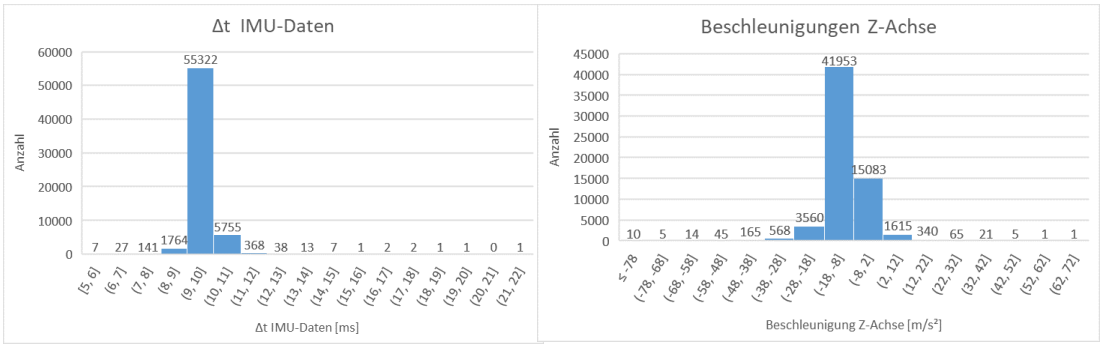


Abbildung 45: IMU-Daten in der Testfahrt 1

Als nächstes werden die IMU-Werte überprüft. Ebenfalls in Abbildung 45 sind diese abgebildet. Der ICM-20948 ist auf einen Arbeitsbereich von ±78,48 m/s² eingestellt. Dieses entspricht ±8 g. Zu erkennen ist, dass während der gesamten Testfahrt 10 Messpunkte außerhalb dieses Wertebereiches lagen. Der größte Anteil der Messwerte liegt in einem Bereich von -28 m/s² bis 12 m/s².
Die GNSS-Daten werden zuerst auf die Messfrequenz überprüft. Hierzu wird ebenfalls ermittelt, in welchen Bereichen das Δt der GNSS-Daten liegt. In Abbildung 46 ist die Verteilung der Δt der GNSS-Daten dargestellt. Bei dem Δt der GNSS-Daten ist das gleiche Phänomen wie bei den GNSS-Daten im Tesla vorzufinden. Der größte Teil liegt im Bereich der eingestellten 200 ms. Allerdings kommt es auch hier vor, dass auf ein zu großes Δt sofort ein zu kleines Δt folgt. In Summe haben diese beiden Δt etwa 400 ms. Es wird also versucht, das fehlerhafte Δt zu korrigieren.

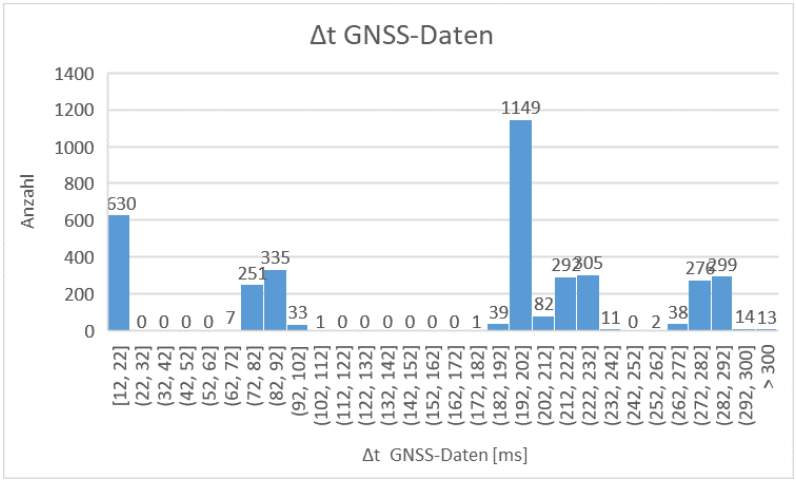


Abbildung 46: Δt GNSS-Daten

Die GPS-Position wird optisch kontrolliert, indem verglichen wird, wie weit die Position von der eigentlich gefahrenen Strecke abweicht. Bereits in der Übersicht der Testfahrt 2 (Abbildung 44, Seite 61) ist zu erkennen, dass die GPS-Position sehr weit von der eigentlich

gefahrenen Position abweicht. Das Sparkfun Venus GPS Modul wird mit einer Genauigkeit von 2,5 m angegeben. In Abbildung 47 ist die Fahrt zwischen zwei Häuserschluchten dargestellt. Die rote Linie zeigt die tatsächlich gefahrene Strecke und die orange Linie die GPS-Positionen. Diese weichen sehr stark von einender ab. Auf der gesamten Testfahrt kommt es zu solchen Abweichungen. Die in Abbildung 47 dargestellte Abweichung ist dabei die Größte.

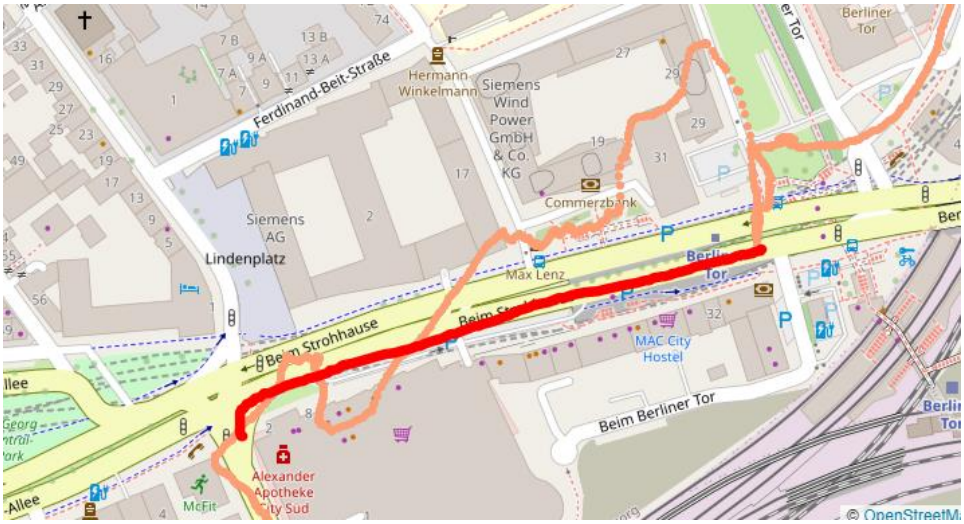


Abbildung 47: Fehlerhafte GPS-Positionen

Fazit: In sehr geringen Fällen wird, der maximale Arbeitsbereich des der IMU überschritten. Eine Erhöhung des Messbereiches von ±8 g auf ±16 ist dabei nicht nötig. Die Verzögerungen der GNSS-Daten können zu einer Verlängerung der Teilstrecke führen. Ausgehen von einer maximalen Geschwindigkeit von 30 km/h kann dieses nicht zu einem Überschreiten der maximalen Länge der Teilstrecke führen.
Die zum Teil fehlerhaften GPS-Positionen sind auf die Position der Messeinrichtung zurückzuführen. Durch die niedrige Position auf dem Gepäckträger wird ein Großteil der Messeinrichtung durch den Körper des Fahrers abgeschirmt.

7.2.3 Ausnutzung der Daten

Die gesamte Testfahrt hatte eine Länge von etwa 3,0 km. Es wurden ca. 2,4 km der Strecke durch 224 Teilstrecken für die Klassifizierung der Straße genutzt. In Abbildung 48 sind die Positionen dargestellt, an denen eine Teilstrecken zur Klassifizierung gebildet wurden.

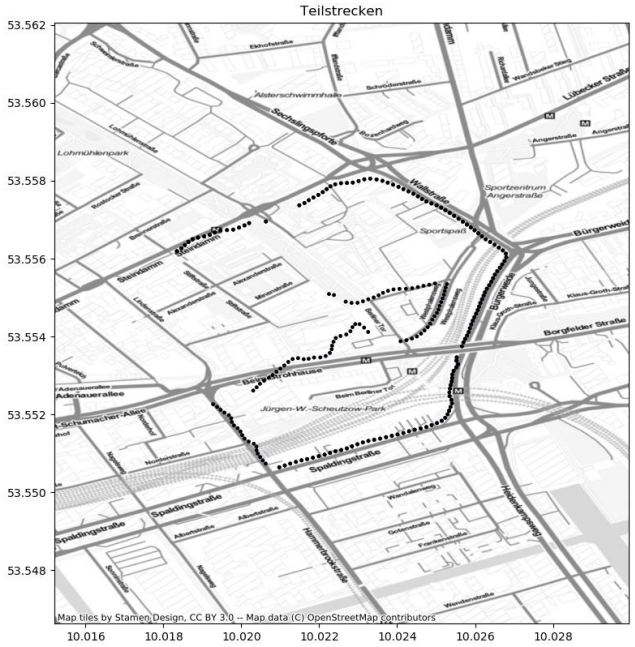


Abbildung 48: Teilstrecken der Testfahrt 1

In Abbildung 49 sind die Längen der Teilstrecken und die Verteilung der Teilstrecken, abhängig von der Fahrzeuggeschwindigkeit, dargestellt. Die meisten Teilstrecken haben eine Länge im Bereich von 10 m bis 12 m ein Die Länge der Teilstrecke ist dabei unabhängig von der Geschwindigkeit.

In Abbildung 50 ist die Anzahl der IMU-Messpunkte innerhalb einer Teilstrecke und die durchschnittliche Distanz zwischen zwei aufgenommenen IMU-Messpunkten dargestellt. Auffällig hierbei ist, dass die Anzahl der Messpunkte pro Teilstrecke bei geringen Geschwindigkeiten stark variiert. Ab einer Geschwindigkeit von etwa 5 m/s verringert sich die Streuung. Bei einer konstanten Geschwindigkeit von 4 m/s ist eine Anzahl von etwa 250 Messpunkten pro Teilstrecke zu erwarten.

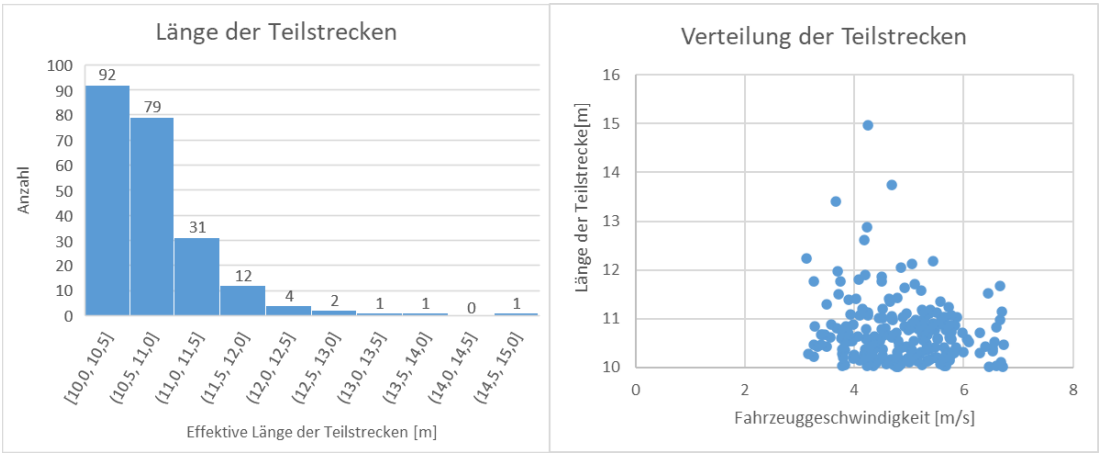


Abbildung 49: Länge der Teilstrecke in der Testfahrt 1

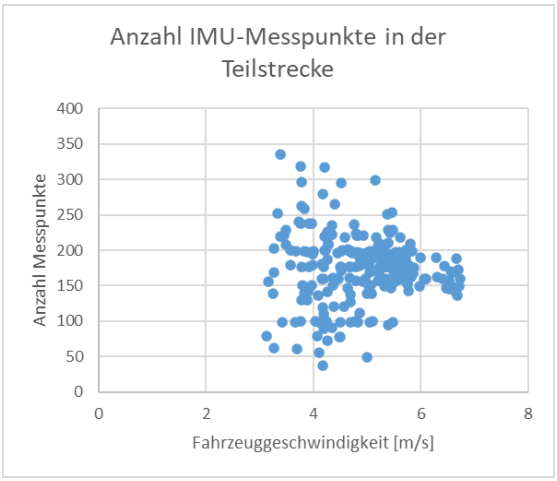


Abbildung 50: IMU-Messpunkte in der Testfahrt 1

Fazit: Durch einmaliges Befahren einer Strecke mit dem Fahrrad wird bereits ein Großteil der Strecke zum Klassifizieren genutzt. Mit dem Fahrrad ist man weniger abhängig von anderen Verkehrsteilnehmern, insbesondere beim Fahren einer konstanten Geschwindigkeit. Ein mehrfaches Abfahren einer Strecke ist in den meisten Fällen nicht nötig.
Die nicht von der Fahrzeuggeschwindigkeit abhängige Anzahl der IMU-Messpunkte in der Teilstrecke verdeutlicht nochmal die ungenauen GPS-Positionen. Die ermittelte Länge der Teilstrecke über die GNSS-Daten weicht in vielen Fällen von der tatsächlich zurückgelegten Entfernung ab.

7.2.4 Genauigkeit des Algorithmus

Um die Genauigkeit des Algorithmus zu prüfen, wurden drei Runden um das Gebäude der Feuerwache an der Straße „Berliner Tor“ gefahren. Dieses ist die gleiche Teststrecke, die beim Testfahrzeug verwendet wurde. Auf dieser Testrecke sind eine Kopfsteinpflaster- und zwei Asphaltstraßen vorhanden. Es wird zunächst die Ermittlung der Straßenqualität getestet. In Abbildung 51 und Abbildung 52 sind die Straßenqualitäten der drei Runden auf der Teststrecke dargestellt.

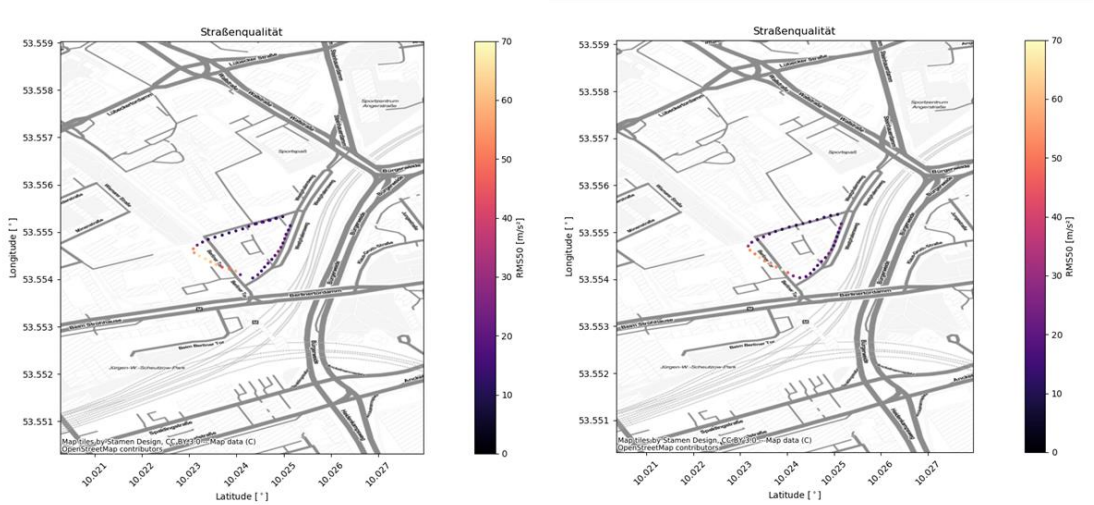


Abbildung 51: Straßenqualität der Testfahrt zur Genauigkeit Runde 1 und 2

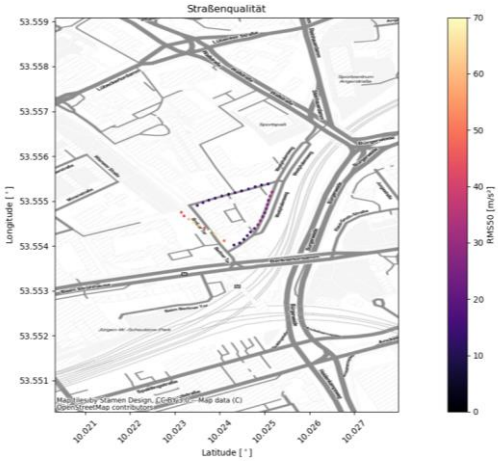


Abbildung 52: Straßenqualität der Testfahrt zur Genauigkeit Runde 3

In allen drei Runden wurde eine gleichmäßige Geschwindigkeit von ca. 15 km/h gefahren. Die Kopfsteinpflasterstraße ist in allen drei Runden gut an dem hohen RMS50 zu erkennen. Auf den Asphaltstraßen ist eine gewisse Variation des RMS50 zu erkennen. Es lassen sich auch einige wiederkehrende Bereiche erkennen. So ist der RMS50 auf der Straße am Parkplatzen vor der Feuerwehr in Runde 1 und 2 an der gleichen Stelle etwas erhöhter. In Runde 3 weist die gesamte Straße auf dem Parkplatz vor der Feuerwache einen erhöhten RMS50 auf. Durch eine Überlagerung der drei Runden lässt sich auch hier eine bessere Darstellung generieren. Die zuvor angesprochene markante Stelle auf dem Parkplatz vor der Feuerwache wird dadurch noch deutlicher sichtbar. In Abbildung 53 ist die Überlagerung der 3 einzelnen Runden dargestellt.

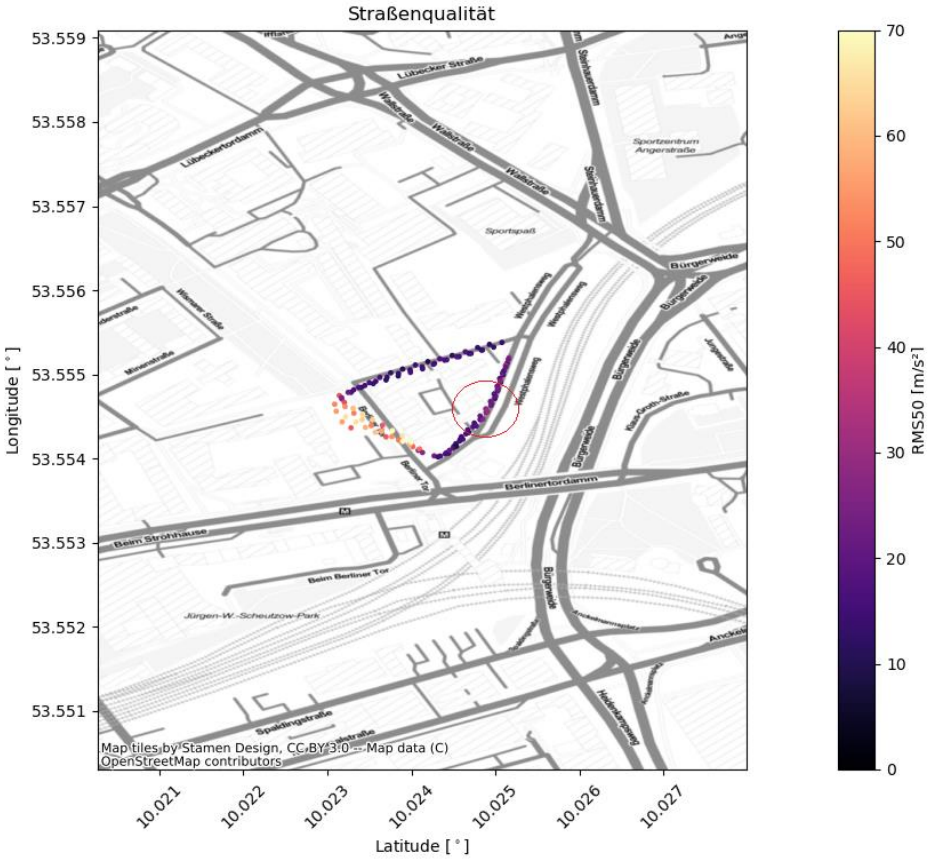


Abbildung 53: Überlagerte Straßenqualitäten der Testfahrt 2

Zur Betrachtung der Events werden zunächst die einzelnen gefunden Events der jeweiligen Runden betrachtet. Im Vorfeld wurden Events auf der Straße „Bei der Hauptfeuerwache“ durch eine optische Betrachtung herausgesucht. In Abbildung 54 und 55 sind die gefunden Events der einzelnen Runden dargestellt.

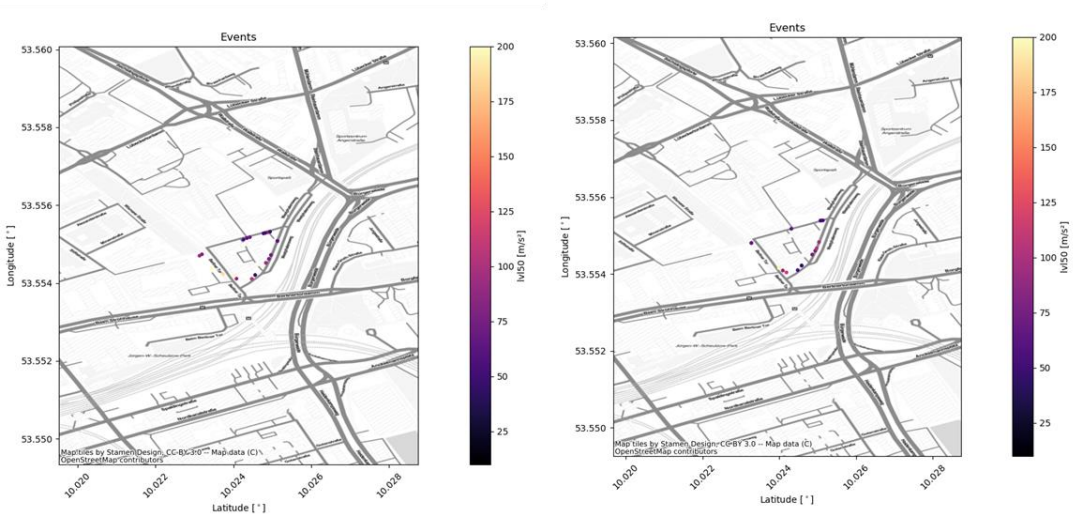


Abbildung 54: Events der Testfahrt 2 Runde 1 und 2

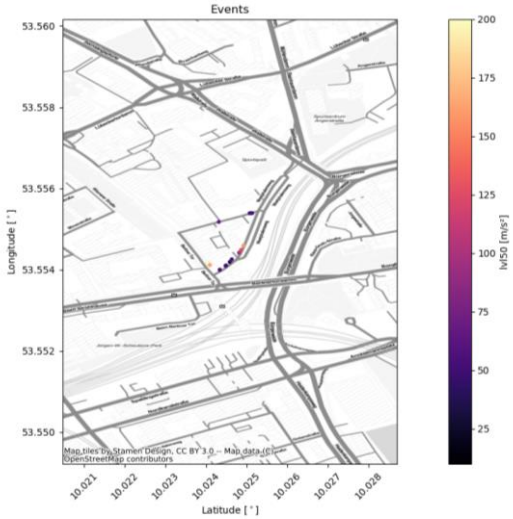


Abbildung 55: Events der Testfahrt 2 Runde 3

Auffällig hierbei ist, dass bei den drei Runden der Testfahrt 2 nur wenig Events auf dem Kopfsteinpflaster erkannt wurde. Auf der Straße „Bei der Hauptfeuerwache“ wurden im Vorfeld drei Events durch eine optische Betrachtung der Straße gesucht, die sicher erkannt werden sollten. Diese Events sind gleichmäßig über die Straße verteilt. Bei den Events

handelt es sich um Kanalschachtabdeckungen und Flicken auf der Straße. Nach einer Analyse der Daten wurde festgestellt, dass alle möglichen Events in diesem Abschnitt gefunden wurden. Das erste Event wird dabei durch die fehlerhaften GPS-Positionen nicht an seiner richtigen Position lokalisiert. In Runde 2 und 3 wurden die beiden letzten Events gefunden. Da es sich hierbei um Flicken auf der Straße handelt, wurde an einigen Stellen mehrere Events durch die Verteilung der Flicken auf der Straße gefunden.
Auch hier lässt sich durch eine Überlagerung der einzelnen Runden eine genauere Übersicht der Daten schaffen. In Abbildung 56 ist die Zusammenlegung der Event dargestellt. Die Events, welche zur Analyse verwendet wurden, werden mit einem roten Kreis markiert.

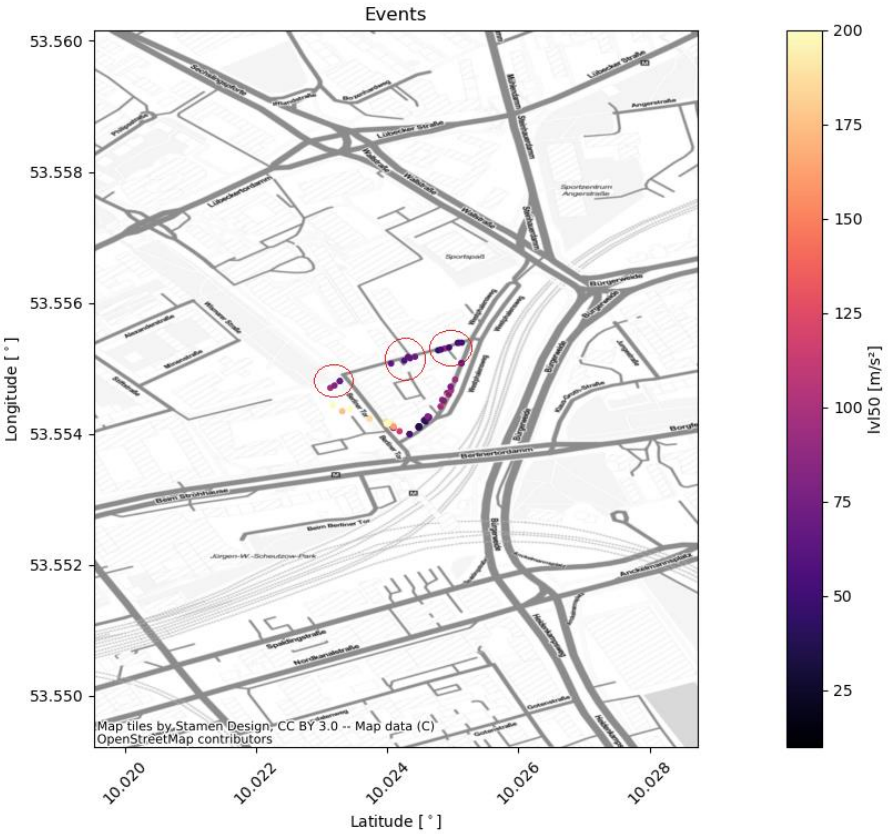


Abbildung 56: Überlagerung der Events der Testfahrt 2

Fazit: Der Algorithmus ist mit den IMU und GNSS-Daten aus der Messeinrichtung in der Lage, die Straßenqualität gut zu bestimmen. Der RMS50 kann daher als Klassifikator genommen werden. Eine Wiederholgenauigkeit ist schwierig zu bestimmen, da mit dem Fahrrad nur ein sehr kleiner Teil der Straße betrachtet wird. Eine Straße kann nicht durch eine Teilstrecke bewertet werden. Es muss immer die Gesamtheit aller Teilstrecken mit einbezogen werden. Aus den vorliegenden Messdaten ist der RMS50 für das Fahrrad wie folgt zu bewerten:

RMS50 [m/s²]	**Straßenqualität**	**Beschreibung**
<4,0 bis 15	gut	Gute Asphaltstraßen mit wenig Beschädigungen und einer glatten Oberfläche
15 bis 35	mittel	Schlechte Asphaltstraßen mit Beschädigungen und höher Oberflächenrauheit.
35 bis >70	schlecht	Kopfsteinpflasterstraßen oder anderweitig gepflasterte Straßen

Tabelle 31: Kategorisierung des RMS50 beim Fahrrad

Das Auffinden eines Events wird durch den Algorithmus mit den eingestellten Parametern gut bewerkstelligt. Das einzelne auffinden eines Events ist dabei immer möglich. Das gleiche Event mehrmals aufzufinden ist dabei nicht immer möglich. Hierbei muss aufgrund der schmalen Reifen des Fahrrades auch berücksichtig werden, ob das Event überhaupt überfahren wurde. Eine Klassifizierung der Events über die Stärke findet nicht statt

7.2.5 Testen der Hardware

Die hier beschrieben Test beziehen sich auf die in 3.4 definierten Anforderungen an die Hardware.

Zur Überprüfung der Staub- und Wasserdichtigkeit der Messeinrichtung wurde das Gehäuse (ohne Hardware im Inneren) für 5 Minuten unter Wasser getaucht. Es konnte kein Eindringen von Wasser festgestellt werden.

Die Messeinrichtung benötig im Betrieb die vollen 2,1 A, die der Akku leisten kann. Mit einer Kapazität von 15,6 Ah steht so eine Laufzeit von etwas mehr als 7 Stunden zur Verfügung.

Der feste Sitz aller Komponenten im Inneren der Messeinrichtung wurde im Zuge der Testfahrten kontrolliert. Es wurde dabei keine Beeinträchtigung der Hardware festgestellt.

Für die Montage und Demontage der Messeinrichtung an einem Fahrrad werden jeweils ca. 5 Minuten benötigt.

Da der Raspberry Pi mit dem CPU nach unten gerichtet betrieben wird, bestand die Vermutung, dass dieser Überhitzen könnte. In Abbildung 57 ist eine Temperaturmessung des internen Temperatursensors des Raspberry Pi dargestellt. Die Umgebungstemperatur betrug dabei 20 °C.

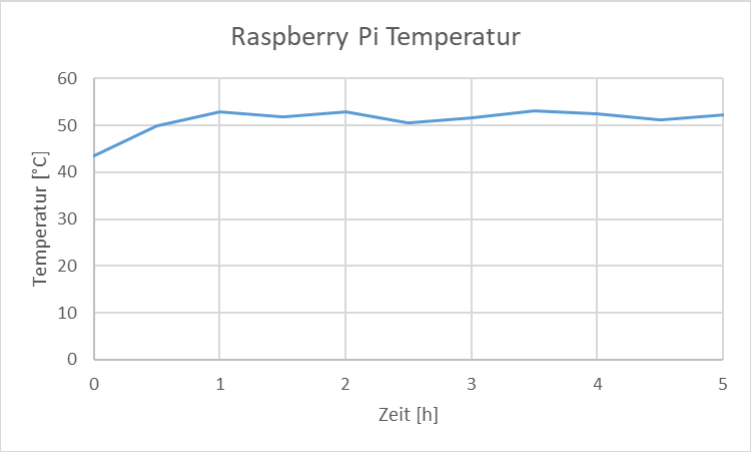


Abbildung 57: Raspberry Pi Temperaturmessung

Fazit: Die zuvor definierten Anforderungen an die Hardware wurden erfüllt. Eine Überhitzung des Raspberry Pi in der Messeinrichtung kann ausgeschlossen werden.

7.2.6 Serververbindung

Um das Hochladen an den EDDY-Server zu testen, wurden der in ROS eingebundene EDDY-Uploader verwendet. Das Hochladen wurde dahingehend getestet, dass die hochzuladenden Daten auf dem jeweiligen System zwischengespeichert wurden. Bei der Simulation auf einem lokalen Computer mit DSL-Anschluss werden alle Daten erfolgreich hochgeladen. Beim Verwenden eines WLAN-Hotspots wird die Verbindung zwischen Smartphone und Raspberry Pi des Öfteren unterbrochen. Wird dann eine erneute Internetverbund hergestellt, werden die Daten automatisch durch den EDDY-Uploader hochgeladen. Wird ein Smartphone mit gedrosseltem Datenvolumen verwendet (64 kbit/s) reicht dieses nicht aus, um die Daten an den Geoserver hochzuladen.
Die Visualisierung, welche die Daten herunterlädt, wird ausschließlich auf einem lokalen Computer mit DSL-Anschluss verwendet. In einigen Tests wurden die heruntergeladenen Daten mit den hochzuladenden Daten überprüft. Dabei wurde kein Unterschied zwischen den Daten festgestellt.

Fazit: Das Hochladen und Runterladen von Daten an den Geoserver funktioniert, solange eine ausreichend gute Internetverbindung besteht.

7.3. Überblick aller Messungen

In diesem Abschnitt werden Karten dargestellt, die alle auf dem Geoserver befindlichen Daten darstellen. Es wurden mehrere Testfahrten durchgeführt. In dieser Übersicht sind auch Testfahrten abgebildet, die nicht direkt in dieser Ausarbeitung verwendet wurden.

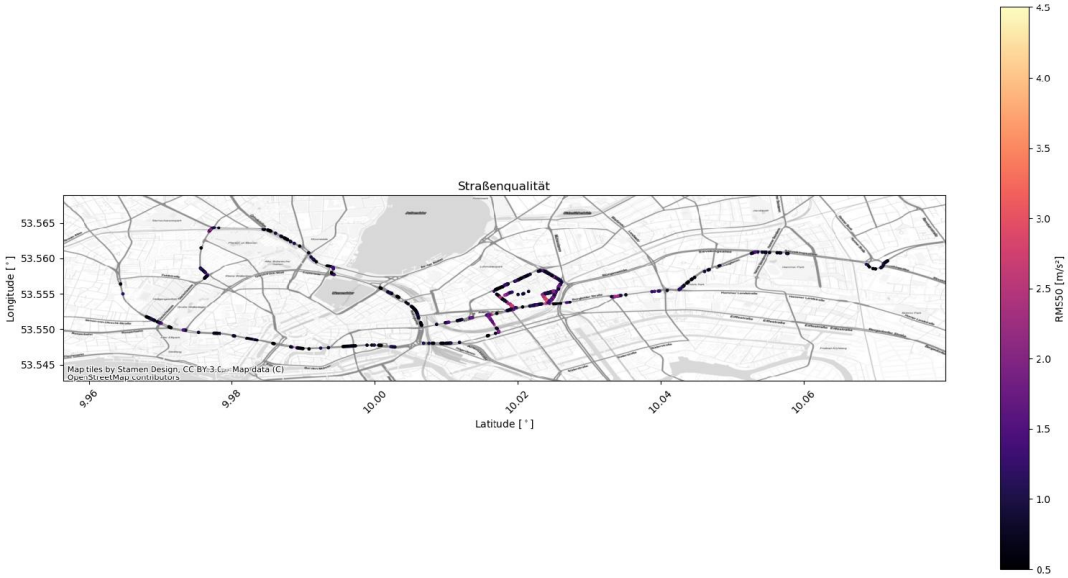


Abbildung 58: Straßenqualitäten ermittelt mit dem Testfahrzeug

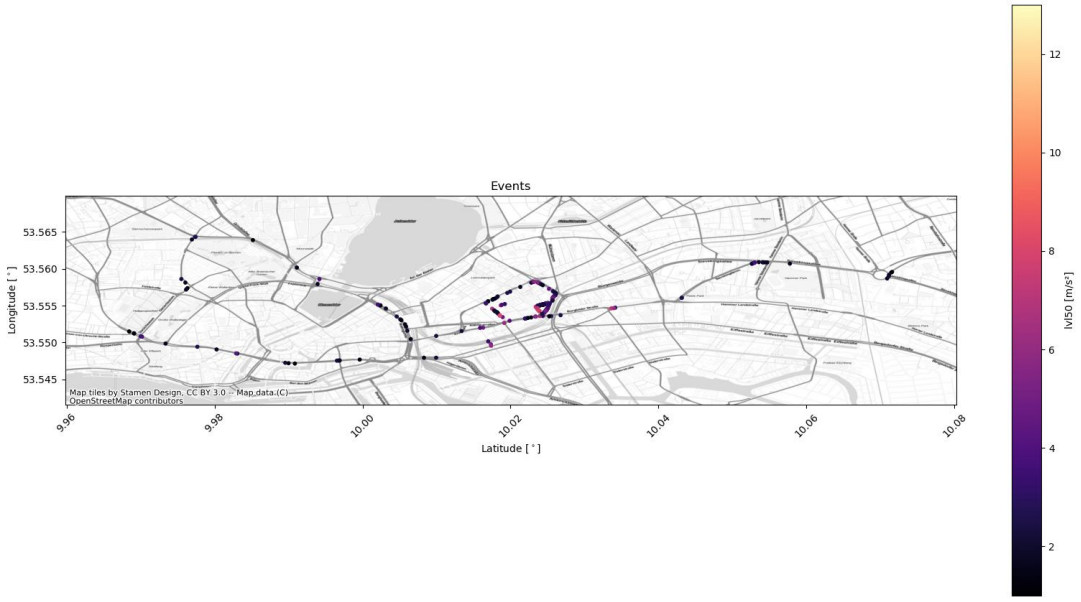


Abbildung 59: Events ermittelt mit dem Testfahrzeug

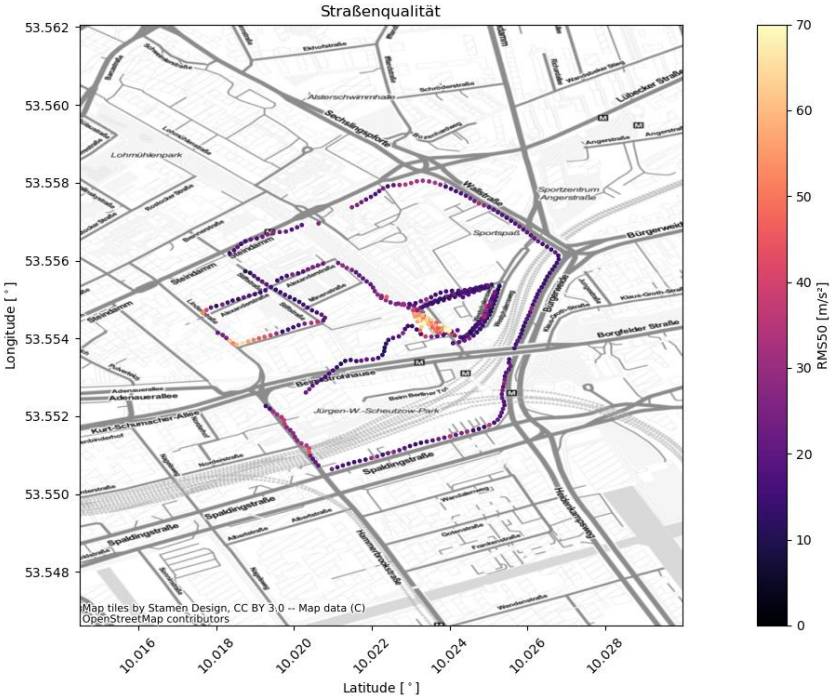


Abbildung 60: Straßenqualitäten ermittelt mit dem Fahrrad

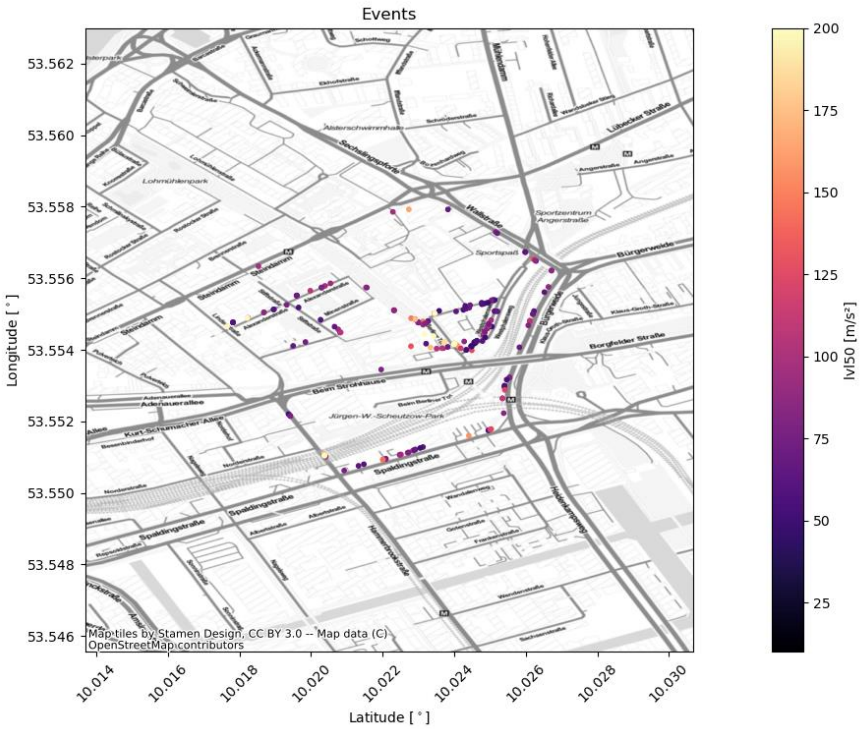


Abbildung 61: Events ermittelt mit dem Fahrrad

8. Zusammenfassung und Ausblick

8.1. Zusammenfassung

In dieser Arbeit wurde ein System entwickelt, welches über die Beschleunigung der Z-Achse eines Fahrzeuges die Straßenqualität ermittelt und Events (z.B. Schlaglöcher) findet. Die Untersuchung der Straße erfolgt dabei in 10 m langen Teilstücken. Durch die Einbindung von GPS-Positionen wird eine Ortsabhängigkeit der Straßenqualitäten hergestellt. In dieser Arbeit wird der Anwendungsfall PKW und Fahrrad unterschieden. Der verwendete Algorithmus wird speziell für unterschiedliche Fahrzeuge durch eine Parameterfindung angepasst. Die ermittelten Straßenqualitäten und Events werden an einen Geoserver übermittelt. Über eine passende Visualisierung können die Daten gefiltert und in einer Karte dargestellt werden.

Um die Straßenqualität unabhängig von der Fahrzeuggeschwindigkeit zu beurteilen, wurde eine Korrekturfaktorfunktion entwickelt, welche die Straßenqualität auf ein 50 km/h Äquivalent anpasst.

Die ermittelten Straßenqualitäten von PKW und Fahrrad werden getrennt voneinander betrachtet, da die Federungen und Aufhängungen der beiden Fahrzeuge grundsätzlich verschieden sind. Kleine Unebenheiten, die im Testfahrzeug kaum messbar sind, erzeugen beim Fahrrad eine deutlich messbare Beschleunigung. Des Weiteren ist der einbezogene Bereich der Straße zur Ermittlung der Straßenqualität beim Fahrrad aufgrund der schmalen Reifen sehr gering. Bei den Testfahrten wurde versucht, exakt die gleiche Strecke zu befahren. Dadurch lässt sich auch beim Fahrrad an einigen Bereichen eine Wiederholgenauigkeit feststellen. Beim PKW wird ein größerer Bereich der Straße aufgrund der breiteren Reifen mit einbezogen. Dadurch sind die ermittelten Straßenqualitäten repräsentativer für die befahrene Straße. Sowohl mit dem PKW als auch mit dem Fahrrad funktioniert die Ermittlung der Straßenqualität gut. Große Unterschiede in der Straßenqualität sind gut zu erkennen. Feine Abstufungen sind nicht erkennbar. Die Beurteilung einer Straße wird durch die Betrachtung mehrerer Teilabschnitte durchgeführt. Eine Straße kann mit gut, mittel oder schlecht bewertet werden.

Die Ermittlung von Events im PKW funktioniert mit Einschränkungen. Starke Events, welche einen hohen Ruck zur Folge haben, werden sicher erkannt. Durch das verwendete Verfahren zur Ermittlung von Events genügt bei sehr glatten Straßen ein verhältnismäßig kleiner Ruck, um ein Event zu lokalisieren. Bei der Ermittlung von Events im Fahrrad tritt das gleiche Problem auf. Hierbei werden durch das nicht gefederte Hinterrad bereits kleine

Unebenheiten zu großen Beschleunigungen. Ein Event wird beim PKW und Fahrrad ermittelt, allerdings nicht weiter kategorisiert.
Zusammenfassend ist zu sagen, dass die Beurteilung der Straßenqualität gut funktioniert und in kommenden Projekten eingesetzt werden kann. Die Eventerkennen benötig noch Anpassungen. Durch das Nutzen der Totzeit ist das Finden von Events eingeschränkt.

8.2. Ausblick

Für die Zukunft sollten noch weitere Straßen, insbesondere von der TAVF, kategorisiert werden, um einen Überblick der gesamten Straßenqualität zu erhalten. Wird dieses in regelmäßigen Abständen durchgeführt können die Straßen auf Verschlechterungen untersucht werden.
Es könnte versucht werden eine allgemeine gültige Korrekturfaktorfunktion für Fahrräder zu finden, indem diese von mehreren Bauartgleichen (Vorderrad gefedert, Hinterrad starr) Fahrrädern untersucht wird.
Ebenso sollte eine bessere Visualisierung entwickelt werden, da in der aktuellen Lösung nicht gezoomt werden kann und die Karten direkt als Bilddatei erzeugt werden. Hierbei sollte eine Lösung gefunden werden, um eine gesamte Straße anhand der einzelnen Teilstrecken beurteilen zu können.

Literaturverzeichnis

[1] Kraftfahrt-Bundesamt, „Bestand," 1 Januar 2022. [Online]. Available: https://www.kba.de/DE/Statistik/Fahrzeuge/Bestand/bestand_node.html. [Zugriff am 06 09 2022].

[2] Bundesministerium für Verkehr und digitale Infrastruktur, „Mobilität in Deutschland – MiD Ergebnisbericht," Dezember 2018. [Online]. Available: https://www.bmvi.de/SharedDocs/DE/Anlage/G/mid-ergebnisbericht.pdf?__blob=publicationFile. [Zugriff am 06 09 2022].

[3] Deutscher Bundestag, „Zustand der Autobahnen und Straßen," 10 April 2019. [Online]. Available: https://www.bundestag.de/webarchiv/presse/hib/2019_04/634830-634830. [Zugriff am 06 09 2022].

[4] Bundesministerium für Verkehr und digitale Infrastruktur, „Fahrrad-Monitor Deutschland 2019," 30 September 2019. [Online]. Available: https://www.bmvi.de/SharedDocs/DE/Anlage/StV/fahrradmonitor-2019-ergebnisse.pdf?__blob=publicationFile. [Zugriff am 06 09 2022].

[5] SBG Systems, „Apogee Series," [Online]. Available: https://www.sbg-systems.com/products/apogee-series-high-accuracy-ins-gnss/. [Zugriff am 04 10 2022].

[6] M. Guiggiani, The Science of Vehicle Dynamics, Pisa: Springer, 2018.

[7] R. Grimm, „Marktführer: Deutschlands Straßen bestehen größtenteils aus Asphalt," 2014 Juli 08. [Online]. Available: https://www.baustoffwissen.de/baustoffe/baustoffknowhow/garten-landschaftsbau-tiefbau/strassenbelag-marktfuehrer-asphalt/. [Zugriff am 12 09 2022].

[8] Deutscher Asphaltverband, „Technik," [Online]. Available: https://www.asphalt.de/themen/technik/. [Zugriff am 14 09 2022].

[9] Max Wiede GmbH, „Alles rund um die Funktionsweise von Flüsterasphalt," [Online]. Available: https://max-wiede.de/funktionsweise-von-fluesterasphalt/. [Zugriff am 15 09 2022].

[10] Kongyang Chen, Guang Tan, Minigming, Jie Wu, „CRSM: a practical crowdsourcing-based road surface monitoring system," 2013.

[11] Luis Amador-Jiménez, Nagham Matout, „A low cost solution to assess road´s rough¬ness surface condi¬tion for Pavement Management," 2014.

[12] Fraunhofer-Institut für Physikalische Messtechnik, „Pavement Management Systeme," [Online]. Available: https://www.ipm.fraunhofer.de/de/gf/objekterfassung-laserscanning/anw/strassen-messtechnik/querebenheit-laengsebenheit.html. [Zugriff am 14 09 2022].

[13] Bundesanstalt für Straßenwesen, „EFA - Erfassungssystem zur Fahrbahnoberflächenanalyse," [Online]. Available: https://www.bast.de/DE/Strassenbau/Technik/EFA.html. [Zugriff am 02 10 2022].

[14] Neueraihemaitijiang Abulizi, Akira Kawamura, Kazuya Tomiyama, Shun Fujita, „Measuring and evaluating of road roughness conditions with a compact road profiler an ArcGIS," Hokkaido Japan, 2016.

[15] Jakob Eriksson, lewis Girod, Bret Hull, Ryan Newton, „The Pothhole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring," 2008.

[16] Ravi Bhoraskar, „Wolverine: Traffic and Road Condition Estimation using Smartphone Sensors," 2012.

[17] Viengnam Douangphachanh, Hiroyuki Oneyama, „A Study on the Use of Smartphones for Road Rougness Condition Estimation," Graduate School of Civil and Environmental Engineering, Tokyo, Japan, 2013.

[18] Fatjon Seraj, Berend Jan van der Zwaag, Arta Dilo, Tammara Luarasi and Paul Havings, „RoADS: A Road Pavement Monitoring System for Anomaly Detetion," Enschede, The Neatherlands, 2016.

[19] IBM Deutschland GmbH, „Geografisches Koordinatensystem," 01 03 2021. [Online]. Available: https://www.ibm.com/docs/de/db2/10.5?topic=systems-geographic-coordinate-system. [Zugriff am 03 11 2022].

[20] Martin Kompf, „Entfernungsberechnung," [Online]. Available: https://www.kompf.de/gps/distcalc.html. [Zugriff am 04 10 2022].

[21] „GY-521 MPU6050 3-Achsen-Beschleunigungs-Gyroskop 6DOF Module," [Online]. Available: https://arduino-projekte.info/produkt/gy-521-mpu6050-3-achsen-beschleunigungs-gyroskop-6dof-module/. [Zugriff am 27 11 2022].

[22] „Adafruit TDK InvenSense ICM-20948 9-DoF IMU (MPU-9250 Upgrade) - STEMMA QT / Qwiic," [Online]. Available: https://www.adafruit.com/product/4554. [Zugriff am 27 11 2022].

[23] „Adafruit 10-DOF IMU Breakout - L3GD20H + LSM303 + BMP180," [Online]. Available: https://www.adafruit.com/product/1604. [Zugriff am 27 11 2022].

[24] „SparkFun Venus GPS with SMA Connector,“ [Online]. Available: https://www.sparkfun.com/products/retired/11058.

[25] „ sbg_driver,“ [Online]. Available: https://github.com/SBG-Systems/sbg_ros_driver. [Zugriff am 27 11 2022].

[26] Projekt "EDDY", [Online]. Available: https://git.haw-hamburg.de/abf136/eddy/-/tree/main/eddy_cloud.

[27] „nmea_navsat_driver,“ [Online]. Available: http://wiki.ros.org/nmea_navsat_driver.

[28] „ros_icm20948,“ [Online]. Available: https://github.com/GAVLab/ros_icm20948.

[29] Maximilian Weltz, „Introduction_EDDY_Cloud,“ Urban Mobility Lab - HAW Hamburg, 09 10 2022. [Online].

[30] Torben Dierks, Jannik Garbers, Joschua R¨uhaak, Wolf Nickel, „Dokumentation zum Bau und Inbetriebnahme des HAW Airviews,“ 01 02 2022. [Online].

[31] Physikalisch-Technische Bundesanstalt, „Das europäische Gravitationszonenkonzept nach WELMEC für eichpflichtige Waagen,“ [Online]. Available: https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_1/1.1_masse/1.15/gravzonen.pdf. [Zugriff am 18 11 2022].

[32] B. Heißig, Fahrwerkhandbuch, Vieweg+Teubner Verlag, 2008.

[33] G. Dillbahner, „Oberbau (Straßenbau),“ 14 Dezember 2007. [Online]. Available: https://de.wikipedia.org/wiki/Oberbau_%28Stra%C3%9Fenbau%29. [Zugriff am 12 09 2022].

[34] [Online]. Available: https://de.m.wikipedia.org/wiki/Datei:Pflaster_ungebunden.svg.

A Anhang

A.1 street_class.msg

```
# Mittelpunkt der Messstrecke [Latitude (positive North deg), Longitude
(positive East deg), Altitude (Above Mean Sea Level m)]
geometry_msgs/Vector3 position

# Länge der Teilstrecke [m]
float32 distance

# Durchschnittsgeschwindigkeit in der Teilstrecke [m/s]
float32 vel

#anzahl der Messpunkte
float32 anzpkt

# RMS bei Messgeschwindigkeit [m/s^2]
float32 rms

# RMS 50 km/h KorrekturFaktor [m/s^2]
float32 rms50

# RMS 50 km/h KorrekturFaktor ohne Beschleunigungen durch Events
[m/s^2]
float32 rms50NoEvents
```

A.2 street_event.msg

```
# [Latitude (positive North deg), Longitude (positive East deg),
Altitude (Above Mean Sea Level m)]
geometry_msgs/Vector3 position

# Durchschnittsgeschwindigkeit in der Teilstrecke [m/s]
float32 vel

# Stärke es Events bei Messgeschwindigkeit [m/s^2]
float32 lvl

# Stärke es Events bei 50 km/h [m/s^2]
float32 lvl50
```

A.3 adapter_node.cpp

```
#include "ros/ros.h"
#include "sensor_msgs/Imu.h"            //Imu Daten
#include "sensor_msgs/NavSatFix.h"      //GPS Position
#include "geometry_msgs/TwistStamped.h" //GPS Geschwindigkeit
#include "sbg_driver/SbgImuData.h"
#include "adapter/sbg_gps_vel_old.h"
#include "adapter/sbg_gps_pos_old.h"

sbg_driver::SbgImuData ImuDataAktuell;
adapter::sbg_gps_vel_old GpsVelAktuell;
adapter::sbg_gps_pos_old GpsPosAktuell;

sbg_driver::SbgImuData ImuDataAlt;
float c = 0.1;

void ImuCallbag(const sensor_msgs::Imu Data){
  ImuDataAktuell.header = Data.header;
  ImuDataAktuell.accel.x = (Data.linear_acceleration.y/4096.0)*9,81;
  ImuDataAktuell.accel.y = -(Data.linear_acceleration.x/4096.0)*9,81;
  ImuDataAktuell.accel.z = -(Data.linear_acceleration.z/4096.0)*9,81;
}

void FixCallbag(const sensor_msgs::NavSatFix Data){
  GpsPosAktuell.header = Data.header;
  GpsPosAktuell.status.status = (uint8_t)Data.status.status;
  GpsPosAktuell.position.x = Data.latitude;
  GpsPosAktuell.position.y = Data.longitude;
  GpsPosAktuell.position.z = Data.altitude;
}

void VelCallbag(const geometry_msgs::TwistStamped Data){
  GpsVelAktuell.header = Data.header;
  GpsVelAktuell.vel.x = Data.twist.linear.x;
  GpsVelAktuell.vel.y = Data.twist.linear.y;
  GpsVelAktuell.vel.z = Data.twist.linear.y;
}
```

```
int main(int argc, char **argv){
  ros::init(argc, argv, "adapter");


  ros::NodeHandle nh;

  ros::Subscriber ImuData =
nh.subscribe<sensor_msgs::Imu>("/icm20948/raw", 1000, &ImuCallbag);
  ros::Subscriber FixData =
nh.subscribe<sensor_msgs::NavSatFix>("/fix", 1000, &FixCallbag);
  ros::Subscriber VelData =
nh.subscribe<geometry_msgs::TwistStamped>("/vel", 1000, &VelCallbag);

  ros::Publisher pub_SbgImuData =
nh.advertise<sbg_driver::SbgImuData>("imu_data", 2);
  ros::Publisher pub_sbg_gps_pos_old =
nh.advertise<adapter::sbg_gps_pos_old>("gps_pos", 2);
  ros::Publisher pub_sbg_gps_vel_old =
nh.advertise<adapter::sbg_gps_vel_old>("gps_vel", 2);

  sbg_driver::SbgImuData ImuDataAlt;
  adapter::sbg_gps_vel_old GpsVelAlt;
  adapter::sbg_gps_pos_old GpsPosAlt;

  ros::Rate r(500);

  while(ros::ok()){
    r.sleep();
    ros::spinOnce();

    if(ImuDataAktuell != ImuDataAlt){
      ImuDataAktuell.accel.x=(1-c)*ImuDataAktuell.accel.x +
c*ImuDataAlt.accel.x;
      ImuDataAktuell.accel.y=(1-c)*ImuDataAktuell.accel.y +
c*ImuDataAlt.accel.y;
      ImuDataAktuell.accel.z=(1-c)*ImuDataAktuell.accel.z +
c*ImuDataAlt.accel.z;
      ImuDataAlt=ImuDataAktuell;
      pub_SbgImuData.publish(ImuDataAktuell);
    }
```

```
    if(GpsVelAktuell != GpsVelAlt){
      pub_sbg_gps_vel_old.publish(GpsVelAktuell);
      pub_sbg_gps_pos_old.publish(GpsPosAktuell);
    }

    ImuDataAlt = ImuDataAktuell;
    GpsPosAlt = GpsPosAktuell;
    GpsVelAlt = GpsVelAktuell;

  }
  return 0;
}
```

A.4 street_classifier_node.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include "std_msgs/Float64.h"
#include "geometry_msgs/Twist.h"
#include "geometry_msgs/Vector3.h"
#include "sbg_driver/SbgImuData.h"
#include "street_classifier/sbg_gps_vel_old.h"
#include "street_classifier/sbg_gps_pos_old.h"
#include "street_classifier/street_class.h"
#include "street_classifier/street_event.h"
#include <sstream>
#include <math.h>

//Auto
#define Vmin              3                 //Min. Geschwindigkeit ab
wann Daten gesammelt werden in m/s
#define Vmax              20                //Max. Geschwindigkeit bis
wann Daten gesammelt werden in m/s
#define LangeAbschnitt    10                //Länge des Messintervalls
in m
#define LangeAbschnittMax 14                //Max. Länge des
Messintervalls in m
#define GeschwindigkeitMittelAbweichung 0.45 //Max. Abweichung der
Geschiwndigkeit in einem Abschnitt in m/s
#define SensorOffsetAccX  0.1823            //Beschleunigungssensor
Offset in X in m/s^2
#define SensorOffsetAccY  -0.046            //Beschleunigungssensor
Offset in Y in m/s^2
#define SensorOffsetAccZ  0.015             //Beschleunigungssensor
Offset in Z in m/s^2
#define f1m               0.0555            //Korekturfunktion
f=(f=mx+b)
#define f1b               0.618             //Korekturfunktion
f=(f=mx+b)
#define f2m               0.0176            //Korekturfunktion
f=(f=mx+b)
```

```
#define f2b               0.0173            //Korekturfunktion
f=(f=mx+b)
#define RuckFaktur        2.5               //Faktor Schwellwert
#define Radstand          2.96              //Radstand des fahrzeuges
in m

/*
//Fahrrad
#define Vmin              3                 //Min. Geschwindigkeit ab
wann Daten gesammelt werden in m/s
#define Vmax              15                //Max. Geschwindigkeit bis
wann Daten gesammelt werden in m/s
#define LangeAbschnitt    10                //Länge des Messintervalls
in m
#define LangeAbschnittMax 17                //Max. Länge des
Messintervalls in m
#define GeschwindigkeitMittelAbweichung 0.6 //Max. Abweichung der
Geschiwndigkeit in einem Abschnitt in m/s
#define SensorOffsetAccX  0.0609            //Beschleunigungssensor
Offset in X in m/s^2
#define SensorOffsetAccY  -0.0607           //Beschleunigungssensor
Offset in Y in m/s^2
#define SensorOffsetAccZ  7.022             //Beschleunigungssensor
Offset in Z in m/s^2
#define f1m               0.1511            //Korekturfunktion
f=(f=mx+b)
#define f1b               -3.661            //Korekturfunktion
f=(f=mx+b)
#define f2m               0.4305            //Korekturfunktion
f=(f=mx+b)
#define f2b               -1.972            //Korekturfunktion
f=(f=mx+b)
#define RuckFaktur        4                 //Faktor Schwellwert
#define Radstand          1.15              //Radstand des Fahrzeuges
in m
*/
#define ArrayGrosse 600
#define EventMaxAnz 40
```

```
enum States{Init, Daten_Sammeln, Daten_Prufen, Daten_Berechnen,
Daten_Publish};

//Globale Variabeln
sbg_driver::SbgImuData ImuDataAktuell;
street_classifier::sbg_gps_vel_old GpsVelAktuell;
street_classifier::sbg_gps_pos_old GpsPosAktuell;

void ImuDataCallbag(const sbg_driver::SbgImuData Data){
  ImuDataAktuell = Data;
  ImuDataAktuell.accel.x = ImuDataAktuell.accel.x + SensorOffsetAccX;
  ImuDataAktuell.accel.y = ImuDataAktuell.accel.y + SensorOffsetAccY;
  ImuDataAktuell.accel.z = ImuDataAktuell.accel.z + SensorOffsetAccZ;
}

void GpsVelCallbag(const street_classifier::sbg_gps_vel_old Data){
  GpsVelAktuell = Data;
}

void GpsPosCallbag(const street_classifier::sbg_gps_pos_old Data){
  GpsPosAktuell = Data;
}

float GpsAbstand(float lat1, float lat2, float lon1, float lon2){
  float dx = 111.3 * cos((lat1 + lat2) / 2 * 0.01745) * (lon1 - lon2);
  float dy = 111.3 * (lat1-lat2);
  return sqrt(dx*dx + dy*dy)*1000;  //in Meter
}

int main(int argc, char **argv){
  sbg_driver::SbgImuData ImuDataArray[ArrayGrosse];
  street_classifier::sbg_gps_vel_old GpsVelArray[ArrayGrosse];
  street_classifier::sbg_gps_pos_old GpsPosArray[ArrayGrosse];

  sbg_driver::SbgImuData re1;
  street_classifier::sbg_gps_vel_old re2;
  street_classifier::sbg_gps_pos_old re3;
  street_classifier::street_event re4;
  street_classifier::street_class re5;
```

```
  street_classifier::street_class sc;
  street_classifier::street_event se[ArrayGrosse];
  float datax2 = 0.0;
  float datax3 = 0.0;
  float RMStest = 0.0;

  int count = 0;
  float dist = 0.0;
  float RMS = 0.0;
  float RMS50 = 0.0;
  float RMSNoEvents = 0.0;
  int state = Init;
  float GeschwindigkeitAktuell = 0.0;
  float GeschwindigkeitMittel = 0.0;
  int DatenOk = 0;
  float distZwischen = 0.0;
  float Erdbeschleunigung = 0.0;
  float Beschleunigung[ArrayGrosse];
  int EventDauertAn[ArrayGrosse];
  int EventAnz = 0;
  float lvl[ArrayGrosse];
  float RuckZ[ArrayGrosse];
  float RuckZBetrag[ArrayGrosse];
  float RuckBetragMittelwert = 0;
  float IMUf = 25.0;                //Fahrrad 100, Auto 25
  float Totzeit = 0.0;
  float TotzeitSoll = 0.0;

  ros::init(argc, argv, "street_classifier_node");
  ros::NodeHandle nh;

  // Subscriber erstellen
  ros::Subscriber SubImuData =
nh.subscribe<sbg_driver::SbgImuData>("imu_data", 1000,
&ImuDataCallbag);
  ros::Subscriber SubGpsVel =
nh.subscribe<street_classifier::sbg_gps_vel_old>("gps_vel", 1000,
&GpsVelCallbag);
```

```
  ros::Subscriber SubGpsPos =
nh.subscribe<street_classifier::sbg_gps_pos_old>("gps_pos", 1000,
&GpsPosCallbag);

  ros::Publisher PubStreetClass =
nh.advertise<street_classifier::street_class>("/street_class", 1000);
  ros::Publisher PubStreetEvent =
nh.advertise<street_classifier::street_event>("/street_event", 1000);
  ros::Rate r(600);

  //EvalEvents
  while (ros::ok()){
    r.sleep();
    ros::spinOnce();
    GeschwindigkeitAktuell = sqrt(GpsVelAktuell.vel.y
*GpsVelAktuell.vel.y + GpsVelAktuell.vel.x * GpsVelAktuell.vel.x);

    switch(state){
      case Init:
        if((GpsPosAktuell.status.status == (uint8_t)0) &&
GeschwindigkeitAktuell >= Vmin){
          state = Daten_Sammeln;
        }
        break;

      case Daten_Sammeln:
        if((GpsPosAktuell.status.status != (uint8_t)0) ||
GeschwindigkeitAktuell < Vmin || dist >=  LangeAbschnittMax){
          state = Init;
        }
        if(LangeAbschnitt <= dist && dist <=  LangeAbschnittMax){
          state = Daten_Prufen;
        }
        break;

      case Daten_Prufen:
        if(DatenOk == 1){
          state = Daten_Berechnen;
        }
        else{
```

```
        state = Init;
      }
      break;

    case Daten_Berechnen:
      state = Daten_Publish;
      break;

    case Daten_Publish:
      state = Daten_Sammeln;
      break;
  }

  //EvalStates
  switch(state){
  case Init:
    // Daten zurücksetzten
    for(int i=0; i<ArrayGrosse; i++){
      ImuDataArray[i] = re1;
      GpsVelArray[i] = re2;
      GpsPosArray[i] = re3;
      Beschleunigung[i] = 0.0;
      EventDauertAn[i] = 0.0;
      se[i] = re4;
      lvl[i] = 0.0;
    }

    count = 0;
    dist = 0.0;
    GeschwindigkeitMittel = 0.0;
    sc =re5;
    Totzeit = 0.0;
    TotzeitSoll = 0.0;
    break;

  case Daten_Sammeln:
    if(count == 0){                                    //erster durchlauf
      ImuDataArray[count] = ImuDataAktuell;
      GpsVelArray[count] = GpsVelAktuell;
      GpsPosArray[count] = GpsPosAktuell;
```

```
        count++;
      }
      else if((ImuDataAktuell != ImuDataArray[count-1])){
        //Daten Speichern
        ImuDataArray[count] = ImuDataAktuell;
        GpsVelArray[count] = GpsVelAktuell;
        GpsPosArray[count] = GpsPosAktuell;

        //Abstand berechnen
        dist = GpsAbstand(GpsPosArray[0].position.x,
GpsPosArray[count].position.x, GpsPosArray[0].position.y,
GpsPosArray[count].position.y);

        count++;
        if(dist >=LangeAbschnitt){
          count--;
        }
      }

    case Daten_Prufen:
      //Geschwindigkeits abweichungen Prüfen
      for(int i=0; i<count; i++){
        GeschwindigkeitMittel = GeschwindigkeitMittel +
(sqrt(GpsVelArray[i].vel.y *GpsVelArray[i].vel.y + GpsVelArray[i].vel.x
* GpsVelArray[i].vel.x));
      }
      GeschwindigkeitMittel = GeschwindigkeitMittel/count;

      for(int i=0; i<count; i++){
        if(sqrt((sqrt(GpsVelArray[i].vel.y * GpsVelArray[i].vel.y +
GpsVelArray[i].vel.x * GpsVelArray[i].vel.x) - GeschwindigkeitMittel) *
(sqrt(GpsVelArray[i].vel.y *GpsVelArray[i].vel.y + GpsVelArray[i].vel.x
* GpsVelArray[i].vel.x) - GeschwindigkeitMittel)) <=
GeschwindigkeitMittelAbweichung){
          DatenOk = 1;
        }
        else{
          DatenOk = 0;
        }
      }
```

```
      break;

    case Daten_Berechnen:
      //Starßenqualität berechnen
      //Erdbeschleunigung entferenen
      for(int i=0; i<count; i++){
        Erdbeschleunigung = Erdbeschleunigung +
ImuDataArray[i].accel.z;
      }
      Erdbeschleunigung = Erdbeschleunigung/count;

      for(int i=0; i<count; i++){
        Beschleunigung[i] = ImuDataArray[i].accel.z -
Erdbeschleunigung;
      }
      Erdbeschleunigung = 0.0;

      //RMS bilden
      for(int i=0; i<count; i++){
        datax2 = datax2 + ((Beschleunigung[i]) * (Beschleunigung[i]));

      }
      RMS = sqrt(datax2/count);
      datax2 = 0.0;


      //Korrektur Fakor
      RMS50 = ((RMS-
(f1m*(GeschwindigkeitMittel*3.6)+f1b))/((f1m*(GeschwindigkeitMittel*3.6
)+f1b)-(f2m*(GeschwindigkeitMittel*3.6)+f2b)))*((f1m*50+f1b)-
(f2m*50+f2b))+(f1m*50+f1b);

      //Daten zuweisen
      sc.rms = RMS;
      sc.rms50 = RMS50;
      sc.distance = dist;
      sc.vel = GeschwindigkeitMittel;
      sc.anzpkt = count;
```

```
      //Mittellpunkt berechnen
      sc.position.x = (GpsPosArray[0].position.x +
GpsPosArray[count].position.x)/2;
      sc.position.y = (GpsPosArray[0].position.y +
GpsPosArray[count].position.y)/2;
      sc.position.z = (GpsPosArray[0].position.z +
GpsPosArray[count].position.z)/2;

      //Events finden
      //Ruck berechnen
      for(int i=1; i<count; i++){
        if(i>=1){
          RuckZ[i] = (ImuDataArray[i-1].accel.z -
ImuDataArray[i].accel.z)/(1/IMUf);
        }
      }
      //Ruck Betrag berechnen
      for(int i=0; i<count-1; i++){
        RuckZBetrag[i] = sqrt(RuckZ[i] * RuckZ[i]);
      }
      //Ruck Betrag Mittelwert
      for(int i=0; i<count-1; i++){
        RuckBetragMittelwert = RuckBetragMittelwert + RuckZBetrag[i];
      }
      RuckBetragMittelwert = RuckBetragMittelwert/(count-1);
      //Totzeit bestimmen
      if (Totzeit<=0){
        TotzeitSoll = Radstand/GeschwindigkeitMittel;
      }
      else{
        TotzeitSoll = Totzeit;
      }

      //Vergleich

      for(int i=0; i<count-1; i++){
        if(Totzeit > 0.0){
          Totzeit = Totzeit - (1/IMUf);
        }
```

```
        if(RuckZBetrag[i] > RuckBetragMittelwert * RuckFaktur &&
Totzeit <= 0.0){
          Totzeit = TotzeitSoll;
          se[EventAnz].lvl = RuckZBetrag[i]*(1/IMUf);
          se[EventAnz].lvl50 = RuckZBetrag[i]*(1/IMUf) * (RMS50/RMS);
          se[EventAnz].position.x = GpsPosArray[i].position.x;
          se[EventAnz].position.y = GpsPosArray[i].position.y;
          se[EventAnz].position.z = GpsPosArray[i].position.z;
          se[EventAnz].vel = GeschwindigkeitMittel;
          EventAnz++;
        }

      }
      EventAnz--;
      RuckBetragMittelwert = 0.0;
    break;


    case Daten_Publish:
        //Daten Publischen
        PubStreetClass.publish(sc);
        ROS_INFO("AAnz: %i",EventAnz);

        for(int i=0; i<EventAnz; i++){
          PubStreetEvent.publish(se[i]);
        }
        EventAnz = 0;

        // Daten zurücksetzten
      for(int i=0; i<ArrayGrosse; i++){
        ImuDataArray[i] = re1;
        GpsVelArray[i] = re2;
        GpsPosArray[i] = re3;
        Beschleunigung[i] = 0.0;
        EventDauertAn[i] = 0.0;
        se[i] = re4;
        lvl[i] = 0.0;
      }
      count = 0;
      dist = 0.0;
```

```
      GeschwindigkeitMittel = 0.0;
      sc =re5;

      break;
    }
  }
  return 0;
}
```

A.5 karteplot.py

```
# Bibliotheken importieren
import configparser
import requests
import json
from geographiclib.geodesic import Geodesic
import pandas as pd
import geopandas
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.colors as colors
import contextily
import datetime

# laden der Config Datei mit API-Key
Config = configparser.ConfigParser()
Config.read("/home/dbb/catkin_ws/src/eddy_cloud/configs/config.ini")

def get_bounding_box(latitude, longitude, size):
    geod = Geodesic.WGS84
    g_ne = geod.Direct(latitude, longitude, 45, size)
    g_sw = geod.Direct(latitude, longitude, 225, size)
    bounding_box = [g_sw['lat2'], g_sw['lon2'], g_ne['lat2'],
g_ne['lon2']]
    return bounding_box

latitude = 53.55419020645113
longitude = 10.024050326501525
ZeitStart = "2022-12-03T18:00:30Z"
ZeitEnde =  "2023-12-03T18:04:50Z"
size = 5000# [m] länge eines Pfeiles
Fahrzeug = "Auto"                   #Auto oder Fahrrad

if Fahrzeug == "Auto":
    oc1 = "Street_Class_Auto"
    oc2 = "Street_Event_Auto"
    s_min = 0.5
    s_max = 4.5
```

```
    e_min = 1
    e_max = 13
if Fahrzeug == "Fahrrad":
    oc1 = "Street_Class_Fahrrad"
    oc2 = "Street_Event_Fahrrad"
    s_min = 0
    s_max = 70
    e_min = 10
    e_max = 200

bounding_box = get_bounding_box(longitude, latitude, size) # Bounding-
Box berechnen
limit = 10000 # Limit an Parametern
offset = 0

collection_id = "traffic_signs"
user = "hamburg"

# URL bestehend aus Basisadresse, collection_id und API-Key
url = 'https://eddy01.comloc.net/oaf/collections/' \
    + collection_id \
    + '/items/?user=' \
    + user \
    + '&limit=' \
    + str(limit) \
    + '&offset=' \
    + str(offset) \
    + '&bbox=' \
    + str(bounding_box[0]) + '%2C' \
    + str(bounding_box[1]) + '%2C' \
    + str(bounding_box[2]) + '%2C' \
    + str(bounding_box[3]) \
    + '&apiKey=' \
    + Config.get('OeconCloud', 'API_KEY')

# Header festlegen
headers = {"accept":"*/*",\
           "apiKey":Config.get('OeconCloud', 'API_KEY')}

# Daten anfragen
```

```
response = requests.get(url, headers = headers)

# Anzeigen der informationen
print(f"Response-Code: {response.status_code}\n\n") # 200 bedeutet
Erfolg
json_response = response.json()

ZeitStart = datetime.datetime.strptime(ZeitStart, '%Y-%m-%dT%H:%M:%SZ')
ZeitEnde = datetime.datetime.strptime(ZeitEnde, '%Y-%m-%dT%H:%M:%SZ')

array_Street_Class = []
array_Street_Event = []
for element in json_response["features"]:
    if element["properties"]["object_class"] == oc1 and
datetime.datetime.strptime(element["properties"]["createdAt"], '%Y-%m-
%dT%H:%M:%SZ') >= ZeitStart and
datetime.datetime.strptime(element["properties"]["createdAt"], '%Y-%m-
%dT%H:%M:%SZ') <= ZeitEnde:
        array_Street_Class.append(element)
    if element["properties"]["object_class"] == oc2 and
datetime.datetime.strptime(element["properties"]["createdAt"], '%Y-%m-
%dT%H:%M:%SZ') >= ZeitStart and
datetime.datetime.strptime(element["properties"]["createdAt"], '%Y-%m-
%dT%H:%M:%SZ') <= ZeitEnde:
        array_Street_Event.append(element)
#print(json.dumps(json_response, indent=4))

geodata_class= geopandas.GeoDataFrame.from_features(array_Street_Class)
geodata_event =
geopandas.GeoDataFrame.from_features(array_Street_Event)

geodata_class = geodata_class.set_crs('EPSG:4326')
geodata_event = geodata_event.set_crs('EPSG:4326')

######################
###Stree Class Plot###
######################
#print(geodata_class)

cmap = 'magma' #Farbverlauf
```

```
data = 'rms50' #Messwert
legendlabel= 'RMS50 [m/s²]' #LabelderColorbar

#norm=colors.Normalize(vmin=geodata_class.rms50.min(),
vmax=geodata_class.rms50.max())
norm=colors.Normalize(vmin=s_min, vmax=s_max)

cbar=plt.cm.ScalarMappable(norm=norm,cmap=cmap)

fig, ax = plt.subplots()
geodata_class.plot(
    ax=ax,
    column=data,
    cmap=cmap,
    vmin=s_min,
    vmax=s_max,
    legend=False,
    markersize=6
)
#addcolorbar
ax_cbar=fig.colorbar(cbar, ax=ax)
ax_cbar.set_label(legendlabel)
ax.ticklabel_format(useOffset=False)
plt.xticks(rotation=45)
ax.set_xlabel('Latitude [$^\circ$]')
ax.set_ylabel('Longitude [$^\circ$]')

plt.title("Straßenqualität")
#es wird da Package contextily genutzt, um automatisch eine
Hintergrundkarte herunterzuladen
contextily.add_basemap(ax,
    crs=geodata_class.crs,
    source=contextily.providers.Stamen.TonerLite
)

plt.tight_layout()#Sicherstellen, dass alles auf dem Bild drauf ist
plt.savefig(data+".png",format='PNG',dpi=300)
plt.show()

######################
```

```
###Stree Class Plot###
######################

cmap = 'magma' #Farbverlauf
data = 'lvl50' #Messwert
legendlabel= 'lvl50 [m/s²]' #LabelderColorbar

#norm=colors.Normalize(vmin=geodata_event.lvl50.min(),
vmax=geodata_event.lvl50.max())
norm=colors.Normalize(vmin=e_min, vmax=e_max)

cbar=plt.cm.ScalarMappable(norm=norm,cmap=cmap)

fig, ax = plt.subplots()
geodata_event.plot(
    ax=ax,
    column=data,
    vmin=e_min,
    vmax=e_max,
    cmap=cmap,
    legend=False,
    markersize=10
)
#addcolorbar
ax_cbar=fig.colorbar(cbar, ax=ax)
ax_cbar.set_label(legendlabel)
ax.ticklabel_format(useOffset=False)
plt.xticks(rotation=45)
ax.set_xlabel('Latitude [$^\circ$]')
ax.set_ylabel('Longitude [$^\circ$]')

plt.title("Events")
#es wird da Package contextily genutzt, um automatisch eine
Hintergrundkarte herunterzuladen
contextily.add_basemap(ax,
    crs=geodata_class.crs,
    source=contextily.providers.Stamen.TonerLite
)

plt.tight_layout()#Sicherstellen,dass alles auf dem Bild drauf ist
```

```
#plt.savefig(data+".png",format='PNG',dpi=300)
plt.show()
```

A.6 StartScript.sh

```
#!/bin/bash
roscore &
sleep 6
rosrun ros_icm20948 talker.py &
sleep 1
rosrun nmea_navsat_driver nmea_serial_driver _port:=/dev/ttyAMA0
_baud:=38400 &
sleep 1
rosrun adapter adapter_node &
sleep 1
rosrun street_classifier street_classifier_node &
sleep 1
rosrun eddy_cloud eddy_uploader.py
```

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

--Platzhalter--