Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Alexander Günther

Konzeption eines Messsystems für die kombinierte Vermessung des Fahrzeugumfelds und der Mobilfunknetze

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

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

Alexander Günther

Konzeption eines Messsystems für die kombinierte Vermessung des Fahrzeugumfelds und der Mobilfunknetze

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

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

Erstprüfer/in: Prof. Dr. rer. nat. Rasmus Rettig
Zweitprüfer/in : Prof. Dr. Zhen Dai

Abgabedatum: 20.09.2023

Zusammenfassung

Alexander Günther

Thema der Bachelorthesis

Konzeption eines Messsystems für die kombinierte Vermessung des Fahrzeugumfelds und der Mobilfunknetze

Stichworte

Objekterkennung, ROS, Bildverarbeitung, Python, Messtechnik Lidar & Mobilfunk, Lokalisierung, neuronale Netze zur Objekterkennung, Umfeldsensorik

Kurzzusammenfassung

In dieser Arbeit steht die Konzeption eines Messsystems im Fokus, welches sowohl die Erfassung der Umgebung eines Trägerfahrzeugs als auch die Leistungsfähigkeit der Mobilfunknetze kombiniert. Die Konzeption basiert auf einer Analyse der Anforderungen, die von den Stakeholdern des Projekts gestellt werden. Die Ergebnisse dieser Analyse dienen als Leitfaden für die Entwicklung einer individuellen Softwarelösung, die darauf abzielt, den Informationsbedarf der beteiligten Interessengruppen umfassend zu erfüllen. Abschließend werden aus dieser Softwarelösung die wesentlichen Hardwareanforderungen extrahiert, um die Grundlage für die Entwicklung eines erweiterten Prototypen zu schaffen.

Abstract

Alexander Günther

Title of the paper

Design of a measurement system for the surveying of the vehicle surrounding and mobile networks.

Keywords

Objectdetection, ROS, Image processing, Python, measurement technology for Lidar and mobile networks, localisation, neural networks for object detection, environmental sensors

Abstract

This work focuses on the conception of a measurement system that combines both the recording of the environment of a carrier vehicle and the performance of the mobile phone networks. The conception is based on an analysis of the requirements presented by the stakeholders of the project. The results of this analysis serve as a guide for the development of a customized software solution that aims to comprehensively meet the information needs of the stakeholders involved. Finally, the essential hardware requirements are extracted from this software solution to create the basis for the development of an extended prototype.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ROS	Robotic Operating System
CPU	Central Processing Unit
YOLO	You Only Look Once
GPU	Graphics Processing Unit
HAW	Hochschule für angewandte Wissenschaften
FPS	Frames per Second
OpenCV	Open Computer Vision
DDR4	Double Data Rate 4
RAM	Random Access Memory
SSD	Solid State Drive
Lidar	Light detection and ranging
SOR	Statstic Outlier Removal
WGS84	World Geodetic System 1984
UTM	Universal Transverse Mercator
SIM	Subscriber Identity Module
NUC	Next Unit of Computing
GNSS	Global Navigation Satellite System
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Receive Quality
RSSI	Received Signal Strength Indicator
SINR	Signal to Interference plus Noise Ratio

1 Einführung

1.1 Motivation

Mobile Datenverbindungen werden in der heutigen Zeit immer wichtiger und besitzen im Alltag ein relevante, aber dennoch passive Rolle. Nicht nur Mobiltelefone nutzen mobile Datenverbindungen, sondern auch andere technische Geräte wie Messeinrichtungen von Mobilfunk- oder Stromnetzbetreibern. Durch die Verwendung intelligenter Stromzähler werden zum Beispiel Zählerstände online an die Stromnetzbetreiber übermittelt. Aufgrund steigender Anwendungsmöglichkeiten der Mobilfunksysteme und Datenmengen ist es notwendig die Mobilfunknetze weiter auszubauen. Dabei spielt auch die Qualität der Netze eine zentrale Rolle.

Im Projekt UrbanRFMap soll eine Messeinrichtung konzeptioniert und realisiert werden, welche die Qualitätskontrolle von Mobilfunknetzen mit übernehmen soll. Dabei liegt die Priorität auf der Kombination von Mobilfunkmesstechnik und Umfeldsensorik.

Die im Verlauf des Projekts erfassten Umfelddaten urbaner Gebiete, darunter Kamerabilder und Lidarmessungen, dienen dazu, Erkenntnisse über die Wechselwirkung zwischen der Mobilfunknetzqualität und der räumlichen Umgebung zu gewinnen. Dies kann durch die Kartierung von Mobilfunk- und Umfelddaten visualisiert werden.

Die Qualität von Mobilfunknetzen verändert sich durch den zukünftig flächendeckenden Ausbau des 5G/6G-Mobilfunknetzes. Die Risiken von geringerer Reichweite der angestiegenen Trägerfrequenzen und die Abschattung der Netze durch Reflexion an Gebäuden können durch die Kartierung aufgezeigt werden. Diese Daten ermöglichen einen gezielten Ausbau der Infrastruktur für unterschiedlichste Branchen und sind sowohl von wissenschaftlichem als auch wirtschaftlichem Interesse.

1.2 Aufgabenstellung & Ziel

Diese Arbeit beschäftigt sich mit der Entwickelung eines effizienten Messsystems zur kombinierten Vermessung der Mobilfunkqualität und Umfelddaten. Dabei soll das neue System auf Basis eines vorhandenen Prototypen konzipiert werden, mit der Vorgabe,

eine kosteneffiziente Umsetzung im Rahmen des UrbanRFMap-Projekts zu ermöglichen. [1]

Hierfür werden die Bedürfnisse und Anforderungen verschiedener Interessengruppen analysiert. Zur Auswertung der Messdaten wird eine Softwarelösung entwickelt, die dem Zweck dient, die Hardwarekomponenten eines zukünftigen Prototypen zu ermitteln. Diese Software soll im späteren Verlauf des Projektes zur Datenauswertung mitgenutzt werden.

1.3 Struktur der Arbeit

Das folgende Kapitel beschreibt die Struktur der Bachelorarbeit, wie in Abbildung 1 dargestellt.

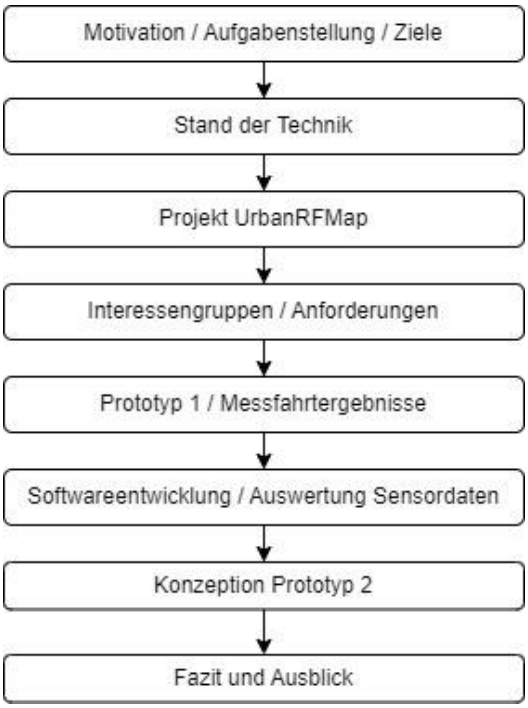


Abbildung 1: Ablauf der Arbeit [2]

Für die fachliche Einordnung der Aufgabenstellung befasst sich die Arbeit im ersten Kapitel mit der Motivation für das Projekt, einer Kurzbeschreibung der Aufgabenstellung und der Ziele.

In Kapitel zwei folgt der Überblick über den Stand der Technik. Im Zentrum stehen die technische Grundlagen für die Mobilfunkvermessung und die Umfeldsensorik. Hier werden einzelne Messgrößen und Instrumente näher erläutert.

Das Projekt UrbanRFMap und dessen für die Bachelorarbeit elementare Ziele werden in Kapitel drei beschrieben. Dies bildet die Grundlage für die Bestimmung und Analyse der Interessengruppen des Projektes in Kapitel vier. Am Ende dieses Kapitels werden die Ergebnisse der Anforderungsanalyse in projektspezifische Größen der Soft- und Hardwareentwicklung abgeleitet und zusammengefasst.

Anschließend wird in Kapitel fünf die Hardware des Prototyp 1 beschrieben und es wird auf die, durch eine Testfahrt generierten, Daten eingegangen.

Die gesammelten Daten der Testfahrt werden in Kapitel sechs analysiert. Aus den Ergebnissen dieser Analyse wird eine Softwarelösung entwickelt, welche den Informationsbedarf der Interessengruppen befriedigen soll.

In Kapitel sieben werden aus der geschriebenen Software die notwendigen Eigenschaften für die Sensorik eines neuen Prototypen extrahiert. Die veränderten Hardwareanforderungen an den neuen Prototypen werden exemplarisch dargestellt und bilden den abschließenden Teil des Kapitels. Die Konzeption stellt das Ergebnis des Projektes dar.

Zum Schluss wird in Kapitel acht ein Fazit der Arbeit gezogen und ein Ausblick in die Zukunft gegeben.

2 Stand der Technik

Das folgende Kapitel beschreibt zum einen die wichtigsten Mobilfunkparameter und die Messtechnik, welche für die Beurteilung der Mobilfunkqualität notwendig sind. Zum anderen den aktuellen Stand der Umfeldmesstechnik und Lokalisierung.

2.1 Mobilfunkparameter und Messtechnik

Die Mobilfunk-Vermessung spielt, wie bereits in Kapitel 1.1 beschrieben, eine immer größere Rolle. Elementare Kenngrößen zur Qualitätsermittlung sind die Werte von RSRP, RSRQ, RSSI und SINR. [3]

Das Referenzsignal der Empfangsfeldstärke am Endgerät (RSRP) ist ein Maß für die Signalqualität des Mobilfunks am Endgerät. Es gibt die Dämpfung des Empfangssignals in der Einheit Dezibel Milliwatt an. Endgeräte des Mobilfunknetzes vermessen dauerhaft die Referenzsignale der Funkzellen in der Umgebung. Zur Bewertung dieser Referenzsignale wird der RSRP-Wert berechnet. Die Endgeräte verbinden sich mit derjenigen Funkzelle, welche die geringste Dämpfung, bzw. den größten RSRP-Wert aufweist. [4]

RSRP		
> -70 dBm bis -79 dBm	Sehr guter Pegel	Telefonie und Internet funktionieren ohne Störungen
-80 dBm bis -89 dBm	Guter, solider Pegel	Internet und Telefonie mit kleinen Störungen
-90 dBm bis -100 dBm	Mittelmäßiger Pegel	Internet und Telefonie mit größeren Störungen
-101 dBm bis -110 dBm	Schlechtes Signal	Internet nur mit Abbrüchen, Telefonie nicht möglich
< -110 dBm	Wenig bis kein Signal	Keine Verbindung oder Verbindungsabbruch

Tabelle 1: RSRP [3]

Durch eine Ausstrahlleistungsgrenze von -23 dBm zu den Endgeräten eines Nutzers ist der Wert auf max. -23 dBm begrenzt. Die Tabelle 1 zeigt, dass die Verbindung bis -79 dBm ohne Störungen ist. Bis -110 dBm wird die Qualität der Verbindung immer schlechter und bei RSRP-Werten kleiner -110 dBm bricht die Verbindung ab. Folglich definiert ein hoher RSRP-Wert ein starkes Signal und eine ideale Verbindung, während ein niedriger RSRP-Wert auf eine geringe Signalstärke und eine schlechte Verbindung hinweist. [3]

Das Referenzsignal der Empfangsqualität (RSRQ) ist ein Maß für die Bewertung der Signalqualität innerhalb des Mobilfunknetzes. Der Wert wird in Dezibel angegeben. Das Endgerät bestimmt den RSRQ-Wert und überträgt ihn an die Basisstation. Auf dieser Grundlage wird die Übertragungsgeschwindigkeit festgelegt und beurteilt, ob ein Zellenwechsel notwendig ist. [3] Die Tabelle 2 zeigt, dass die Signalqualität von großen zu kleinen RSRQ-Werten abnimmt. [5]

RSRQ		
-3 dB	Keine Störungen vorhanden	Exzellente Qualität
-4 dB bis -5 dB	Kaum Störungen	Keine Beeinflussung der Verbindung
-6 dB bis -8 dB	Störungen vorhanden	Störungen wirken sich auf die Verbindung aus
-9 dB bis -11 dB	Große Störungen	Spürbare Leistungsverluste vorhanden
-12 dB bis -15 dB	Starke Störungen	Keine Telefonie mehr möglich
-16 dB bis -20 dB	Extrem starke Störungen	Kein Nutzsignal vorhanden

Tabelle 2: RSRQ [3]

Der Indikator für die Empfangsfeldstärke (RSSI) ist ein Maß für die absolute Signalstärke, welche die Empfangsantenne misst. Darin enthalten sind Signale zur Steuerung, Datensignale, aber auch Interferenzsignale benachbarter Zellen, Hintergrundrauschen und Störungen. [3]

RSSI	
-23 dB	beste, mögliche Verbindung
-70 dB bis -80 dB	Keine Störungen
-80 dB bis -90 dB	Einigermaßen stabil
-90 dB bis -100 dB	Starke Probleme
-100 dB bis -110 dB	Extrem langsame Internetverbindung, keine Telefonie mehr
< -110 dBm	Keine Verbindung möglich

Tabelle 3: RSSI [6]

Eine Aussage über die Qualität der Verbindung ist mit dem RSSI nicht möglich. Selbst wenn der Wert hoch ist, kann die Qualität der Verbindung schlecht sein. In diesem Fall enthält das empfangenen Frequenzband viele Störungen bzw. Rauschen. [7]

Der RSSI-Wert wird benötigt, um den „signal-to-interference-plus-noise-ratio“-Wert (SINR) zu berechnen. Dieser setzt den RSSI-Wert in ein Verhältnis mit den Störungen und dem Rauschen der Funkverbindung. [3]

SINR Wert und Datendurchsatz	
> 10 dB	Exzellent
6 dB bis 10 dB	Gut
0 dB bis 5 dB	Mittelmäßig
< 0 dB	Kein Datenverkehr möglich

Tabelle 4: SINR [3]

Tabelle 4 zeigt, dass, je größer der SINR-Wert ist, umso hochwertiger ist die Verbindungsqualität. Ist der SINR-Wert hoch kann durch wenig Störungen und Rauschen ein höherer Datendurchsatz realisiert werden. [8]

Geeignete Gerätschaften, um die Mobilfunkqualität zu ermitteln, werden unter anderem von dem Unternehmen ENQT GmbH entwickelt. Die Abbildung 2 zeigt das von ihr entwickelte AMS-System. Dieser Netztester bestimmt die zuvor beschriebenen Werte und speichert diese in einem Messdatenportal. Durch eine Georeferenzierung ist es möglich, großflächige Messungen durchzuführen und zu visualisieren, um die Mobilfunkqualität einschätzen zu können. [9]

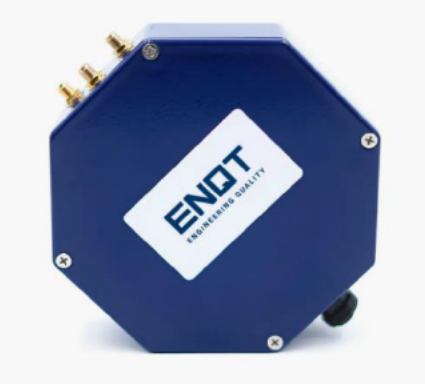


Abbildung 2: ENQT GmbH - AMS [9]

2.2 Umfeldmesstechnik

Die Umfeldmesstechnik bezieht sich auf Technologien und Methoden welche verwendet werden, um verschiedene Eigenschaften der Umgebung zu messen [10]. Dabei wird zwischen bildgebenden und nicht-bildgebenden Technologien unterschieden. Um flächige Abtastungen zu realisieren, werden bildgebende Sensoren wie z.B. Lidar-Sensoren, Kameras oder Infrarotkameras verwendet. Zu den nicht-bildgebenden Sensoren zählen Ultraschallsensoren oder Umfeldmesstechniken zur Vermessung der Luftqualität.

Abbildung 3: Velodyne HDL32-E [11]

Lidar-Sensoren, beispielhaft dargestellt in Abbildung 3, sind bildgebende Sensoren, welche den Abstand und Winkel eines Messpunktes zum Sensor bestimmen. Der Begriff „Lidar“ ist die Abkürzung für „Light Detection and Ranging“ und beschreibt eine optische Messmethode, die auf der Erfassung von Lichtwellen basiert. [12] Der Lidar erzeugt ein digitales Abbild des Umfeldes. Ein Laserstrahl tastet die Umgebung ab. Reflektiert dieser an einer Oberfläche, detektiert der Lidar dies und kann über die Zeit zwischen dem Aussenden des Strahls und dem Erfassen der Reflexion den Abstand ermitteln. Diese Messmethode wird „Time of Flight“ genannt. [13]

Lidar-Sensoren unterscheiden sich in der Anzahl der Laserstrahlen und der Option zur Rotation des Sensors um seine eigene Achse. Rotiert der Lidar, besitzt der Sensor Kenntnis über seine Orientierung. Folglich ist es möglich, dem gemessenen Abstand einen Winkel im sensoreigenen Koordinatensystem zuzuordnen. Geschieht dies fortlaufend, erhält man Datensätze, in denen sich die Koordinaten der Messpunkte des Lidar-Umfeldes befinden. Durch mehrere gleichzeitig messende Laserstrahlen ist es möglich, dreidimensionale Punktwolken zu erzeugen, hier in Abbildung 4 dargestellt.

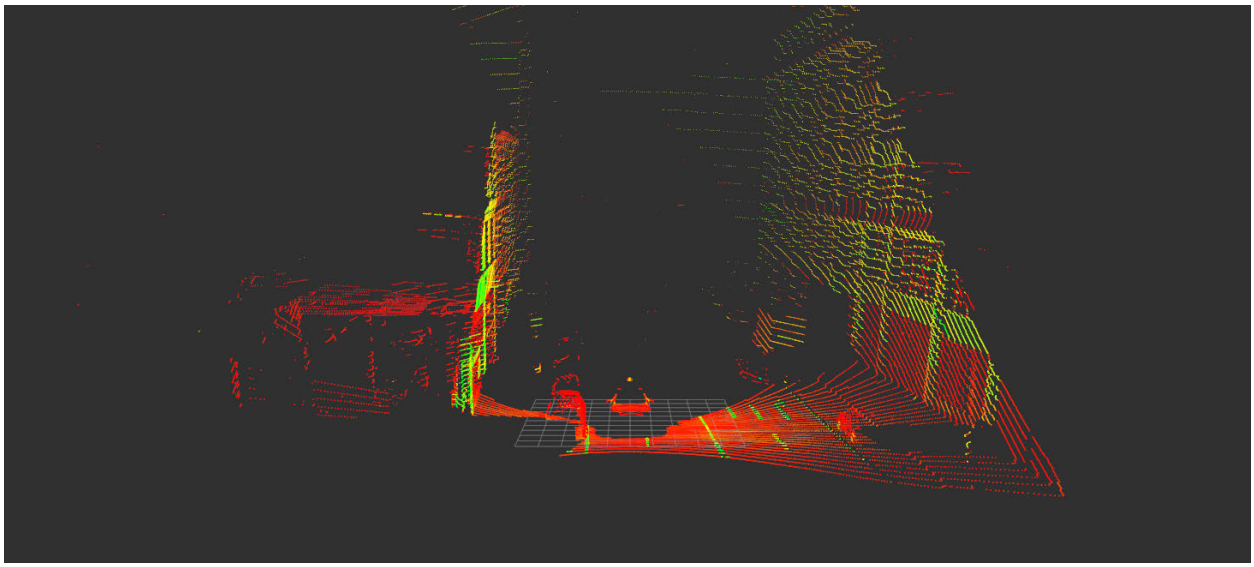


Abbildung 4: Visualisierte Punktwolke des Velodyne HDL-32E [2]

Neben der dreidimensionalen Abbildung der Umgebung kann die Lidartechnik für Distanzmessungen zwischen Objekten genutzt werden. Diese Technik findet unter anderem Anwendung im Bereich des autonomen Fahrens.

Weitere bildgebende Sensoren sind verschiedene Typen von Kameras mit variabler Verwendung. Die vielfältige Auswahl auf dem Markt beinhaltet Farb- und Grauwertkameras, Infrarotkameras, Kameras mit vielen und wenigen Bildern die Sekunde oder Kameras mit Global-Shutter oder Rolling-Shutter. Global-Shutter bedeutet, dass alle Pixel des Bildaufnahmesensors der Kamera gleichzeitig belichtet und ausgelesen werden. Der Vorteil daran ist die zeitliche Synchronität der Pixel eines Bildes. Durch den Global-Shutter werden Verzerrungen während der Aufnahme eines Bildes verhindert. Bei einem Rolling-Shutter werden die Pixel des Bildaufnahmesensors zeilenweise belichtet und nacheinander eingelesen. Dies ist kostengünstiger in der Anschaffung, führt jedoch bei schnellen Bewegungen zu Verzerrungen. [14] Es ist möglich, für jeden Verwendungszweck einen passenden Kameratyp auszuwählen und einzeln oder in Kamerasystemen zu verwenden.

Im weiteren Verlauf werden ausschließlich Kamerasysteme beschrieben, die in dieser Arbeit zur Anwendung kommen.

Das erste System, welches zur Anwendung kommt, ist ein Stereokamerasystem. Dieses besteht aus zwei Einzelkameras, welche in fest definiertem Abstand und fest definierter Orientierung zueinander aufgebaut sind. [15]

Abbildung 5: Aufnahmen eines Stereokamerasystems [2]

Abbildung 5 zeigt die Bilder eines Stereokamerasystems. Durch die synchrone Aufnahme der Bilder ist es möglich, Tiefeninformationen des Bildinhaltes zu ermitteln. Durch spezielle Algorithmen und Techniken wird mit Stereokameras das dreidimensionale Sehen des menschlichen Auges imitiert.

Ein weiteres Kamerasystem ist die sphärische Rundumkamera. Diese erzeugt 360° Aufnahmen mittels mehrerer im Kreis angeordneter Einzelkameras mit überschneidenden Bildbereichen. Ein Beispiel dafür ist das in Abbildung 6 dargestellte Ladybug 5+ System des Unternehmens Teledyne FLIR mit sechs Einzelkameras. [16]

Abbildung 6: Teledyne FLIR Ladybug 5+ [16]

Ein 360° Aufnahme der Ladybug ist in Abbildung 7 zu sehen.

Abbildung 7: Ladybug 5+ Bild [2]

Zu den nicht-bildgebenden Technologien der Umfeldvermessung gehören beispielsweise Sensoren zur Messung von Lufttemperatur, Luftdruck, Partikeln in der Luft, Geräuschpegel, Rauchentwicklung, Lichtintensität und Radioaktivität.

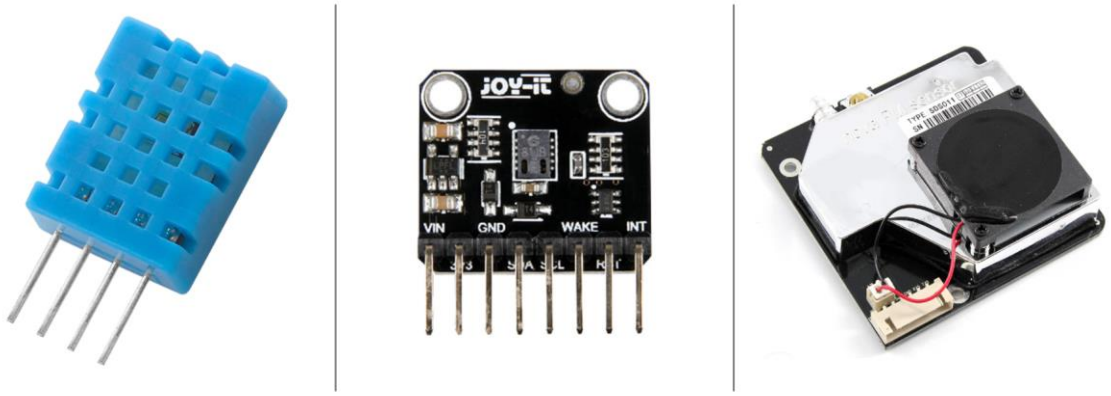


Abbildung 8: DHT11 [17] | CCS881 [18] | PM SDS011 [19]

In Abbildung 8 werden drei Sensoren dargestellt. Der DHT11-Sensor führt mittels kapazitivem Feuchtigkeitsfühler und einem Heißleiter Temperatur- und Luftfeuchtigkeitsmessungen durch. [17]

Der CCS881-Sensor detektiert flüchtige organische Verbindungen in der Luft [18]. Zu den flüchtigen organischen Verbindungen gehören Stoffe wie Flüssigbrennstoffe, Lösungsmittel, Alkohole oder Kohlenwasserstoffe. [20] Zusätzlich kann der Sensor aus seinen Messwerten die Äquivalenzkonzentration von Kohlenstoffdioxid ermitteln. [21]

Ein weiterer Wert der Luftgüte ist die Partikelkonzentration. Mit dem SDS011-Sensor ist es möglich Partikel mit einer Größe von 0,3 bis 10 µm zu messen. Die Partikelmessung findet mittels des Streulichtverfahrens statt. [19] Dabei wird eine Lichtquelle auf die Probe

gerichtet. Aufgrund der Wechselwirkung der Probe kann durch die Streuung des Lichtes die Partikelgröße ermittelt werden. [22]

Die beschriebenen bildgebenden Sensoren zur Umfeldvermessung werden im Verlauf dieser Arbeit als Messtechnik verwendet. Die nicht-bildgebenden Sensoren sind exemplarisch mit aufgeführt und werden in dieser Arbeit nicht primär betrachtet.

2.3 Lokalisierung und Koordinatensysteme

Die eigene Positionsbestimmung ist ein wichtiger Bestandteil vieler technischer Systeme. Navigationssysteme benötigen die aktuelle Position, um den optimalen Routenverlauf zu berechnen oder die aktuelle Umgebung anzuzeigen. Ein hochpreisiges Präzisionsgerät dieser Art ist zum Beispiel der in Abbildung 9 dargestellte Apogee-D des Unternehmens SBG Systems.

Abbildung 9: SBG Apogee-D [23]

Dieses Koppelnavigationssystem, welches GNSS-Signale mit einer Präzisions-Inertialsensorik verknüpft, besteht aus Beschleunigungssensoren und Gyroskopen, um die genaue Lage und Bewegung eines Fahrzeugs zu überwachen. Bei vorübergehendem GNSS-Signalausfall ist bei diesem System eine zuverlässige Lokalisierung, mit einer Lokalisierungsgenauigkeit von wenigen Zentimetern, möglich. [23]

Der Begriff Lokalisierung bezieht sich auf die georeferenzierte Positionsbestimmung. Für die Festlegung der Georeferenz gibt es verschiedene Systeme.

Ein System ist das World Geodetic System von 1984 (WGS84). Dieses teilt die Erde, wie in Abbildung 10 zu sehen, in Längen- und Breitengrade auf. GNSS-Systeme können diese Größen bestimmen und für die Nutzung bereitstellen. [24]

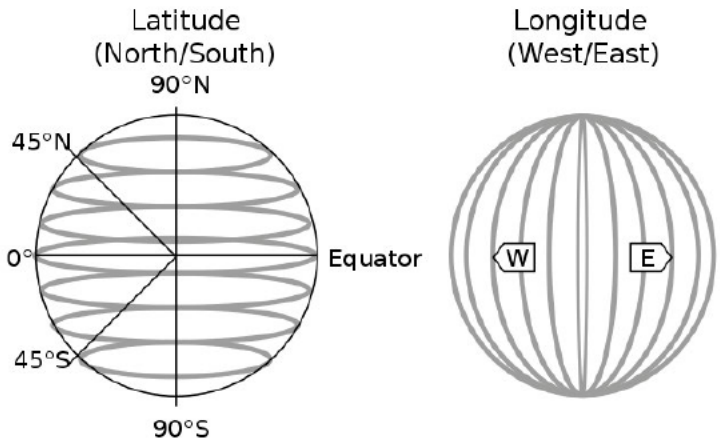


Abbildung 10: WGS84 [25]

Ein weiteres globales Koordinatensystem ist das „Universal Transverse Mercator“ (UTM). Der Vorteil dieses Koordinatensystems ist, wie in Abbildung 11 zu erkennen, die Einteilung der Erde in verschiedene Zonen.

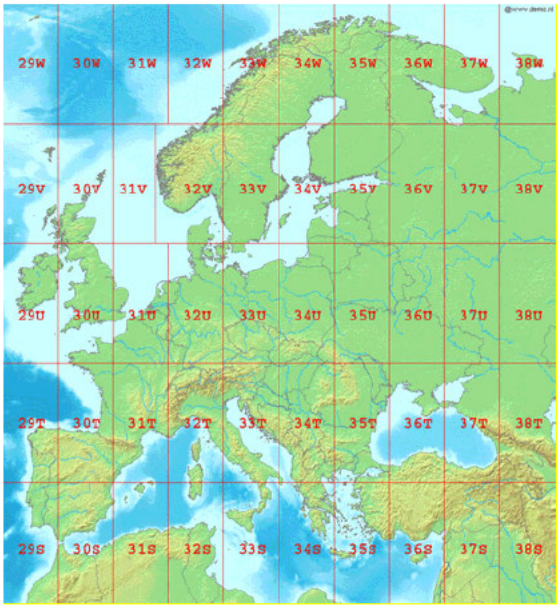


Abbildung 11: UTM-Koordinatensystem [26]

Innerhalb jeder Zone wird ein kartesisches Koordinatensystem in Metern zugrunde gelegt. Der Nullpunkt liegt jeweils unten links und die Achsen northing, in Richtung Norden, und easting, in Richtung Osten. Dieses Koordinatensystem ermöglicht es linear zwischen Positionen zu rechnen. [27] Diesen Vorteil bietet ein globales Koordinatensystem wie das WGS84-Koordinatensystem nicht, da dort Längen- und Breitengrade verwendet werden, welche keinen exakt linearen Zusammenhang bieten.

3 Projekt UrbanRFMap

Das folgende Kapitel bildet eine Grundlage für diese Bachelorarbeit. Dabei wird auf den Förderantrag des Projektes eingegangen. [1, S. 2]

3.1 Projektbeschreibung

Das Projekt UrbanRFMap soll als erstes System seiner Art die Leistungsfähigkeit der Mobilfunknetze sowie das Umfeld in dichten urbanen Gebieten analysieren. Dabei wird der Fokus primär auf die Eigenschaften des Messsystems gelegt. Die Projektpartner ENQT GmbH und die Hochschule für Angewandte Wissenschaften Hamburg haben sich gemeinsam als übergeordnetes Ziel gesetzt ein in sich geschlossenes, autarkes, hoch automatisiertes und robustes Messsystem zu entwickeln. [1, S. 2]

Die großflächige Vermessung des Umfeldes und des Mobilfunknetzes urbaner Gebiete ist derzeit äußerst kostspielig. Hochqualifiziertes Personal wird benötigt, um kostenintensive Messverfahren durchzuführen. [1, S. 4]

Diese Bachelorarbeit ist Teil des Projektes UrbanRFMap. Sie befasst sich mit der Software- und Hardwarekonzeption eines Prototypen. Dabei bezieht sich die Konzeption der Hardware auf das Messsystem selbst. Die Konzeption der Software bezieht sich auf die Auswertung von Sensordaten.

3.2 Ziele des Projektes

Ein elementares Ziel ist sowohl die Kosteneffizienz des Messsystems selbst, sowie die Kosten bei der Datenbeschaffung. Beide Faktoren sollen, im Vergleich zur Konkurrenz, günstiger sein. Realisiert werden kann dies zum einen durch die Verwendung kostengünstiger und dennoch präziser Messtechnik. Zum anderen durch die Nutzung von Open-Source-Software, welche eine kostengünstige Realisierung, mit einem hohen Automatisierungsgrad, ermöglicht. Folglich kann im Betrieb des Messystems auf den kostenintensiven Einsatz qualifizierten Personals verzichtet werden. [1, S. 6]

Dies führt zu einem autarken Messsystem. Durch den Einsatz einer höheren Stückzahl des Messsystems ist die Grundlage für eine effektive und ökonomische Vermessung

urbaner Gebiete geschaffen. Die Installation des Systems auf Trägerfahrzeugen, wie zum Beispiel Müllsammlern der Stadtreinigung, ermöglicht eine großflächige und langfristige Datenerhebung. [10.2]

Ein weiteres Ziel ist die Entwicklung eines modularen Systems, welches durch diverse Sensoren erweitert werden kann. Dies dient dem Zweck, das Gesamtbild der Messdaten zu erweitern und ein möglichst reales Abbild der urbanen Gebiete durch diverse Messungen zu schaffen. [1]

Aus den Zielen des Projektes UrbanRFMap ergeben sich Anforderungen unterschiedlicher Interessengruppen, welche im folgenden Kapitel beschrieben sind.

4 Stakeholder und Anforderungen

In diesem Kapitel sollen die Interessengruppen, Stakeholder, für das zu entwickelnde Messsystem herausgearbeitet werden. Anschließend werden die Interessen dieser Gruppen abgeleitet und Anforderungen an die Bachelorarbeit definiert.

4.1 Stakeholder Analyse

In der Stakeholder Analyse stellt sich zum einen die Frage, welche Gruppen Interesse an Mobilfunk- und Umfelddaten aus dem Bundesland Hamburg haben, zum anderen, welche Gruppen eine flächige Datenerhebung über das gesamte Stadtgebiet in vergleichsweise kurzen Intervallen gewährleisten können.

4.1.1 Identifikation von Stakeholdern

Durch Brainstorming und Diskussionsrunden wurde eine Liste möglicher Stakeholder erstellt. Zunächst wurden generelle Interessengruppen aufgelistet. Dazu gehören kommunale Reinigungsdienste, Mobilfunknetzbetreiber, Energieversorger, die Bundesnetzagentur, sowie weitere Unternehmen aus den unterschiedlichsten Branchen. [10.1]

Aufgrund der Projektförderung durch ein Hamburger Institut werden mit den, in der Liste enthaltenen, Branchen regionale Interessengruppen ermittelt. Diese Interessengruppen sind die Hamburger Stadtreinigung, die Behörde für Umwelt, Klima, Energie und Agrarwirtschaft (Bukea), Vay ein Unternehmen für fahrerlose Mobilität und HPA, die Hamburg Port Authority. Als Projektpartner stellen das Unternehmen ENQT GmbH und die Hochschule für Angewandte Wissenschaften Hamburg weitere Interessengruppen da. Diese Bachelorarbeit betrachtet primär die Projektpartner sowie den Stakeholder Hamburger Stadtreinigung, eine Anstalt des öffentlichen Rechts.

Die Fahrer der Trägerfahrzeuge sind als passive Stakeholder zu betrachten. Sie stehen in keinem direkten Kontakt mit dem Messsystem, müssen dennoch beachtet werden.

4.1.2 Interessen und Erwartungen

Im bisherigen Verlauf des Projektes UrbanRFMap wurde von den Projektpartnern erfolgreich Kontakt mit der Hamburger Stadtreinigung hergestellt. Die Absicht der Projektpartner besteht darin, das Messsystem auf den Fahrzeugen der Stadtreinigung zu installieren. Die Fahrzeugflotte der Hamburger Stadtreinigung umfasst ca. 180 Müllsammler, welche in einem zweiwöchigen Intervall alle Straßen des Hamburger Stadtgebietes befahren. Diese Flotte ermöglicht die Erzeugung von intervallaktuellen Messdaten.

In Gesprächen mit diesem Unternehmen wurden deren Interessen und Erwartungen identifiziert. Die Stadtreinigung stellt die Forderung, dass das Müllsammlerpersonal durch das Messsystem in seiner Arbeit nicht beeinträchtigt wird. Zusätzlich will das Unternehmen die vom Messsystem generierten Daten zur Unternehmensoptimierung nutzen.

Die Hamburger Stadtreinigung ist ein städtisches Unternehmen und somit ausschreibungspflichtig. Daher müssen alle vier Jahre Ausschreibungen für die Mobilfunkverträge der Stadtreinigung gemacht werden. Wird durch das zu entwickelnde Messsystem die Mobilfunkqualität in Hamburg aufgezeigt, ist es dem Unternehmen möglich den für sie optimalen Anbieter zu bestimmen.

Die vom Messsystem erfassten Daten können zur Optimierung logistischer Prozesse genutzt werden. Mit der Kenntnis über den Standort bereitgestellter Mülltonnen kann der Entsorgungsprozess optimiert werden. Mit der Kenntnis über die Durchfahrtbreite des Fahrwegs kann der Fahrer eine Entscheidung über die Passierbarkeit der Straße treffen.

Im Verlauf der Gespräche mit der Stadtreinigung wurden weitere Fragen identifiziert, die in dieser Bachelorarbeit nicht weiter vertieft werden. Diese Fragen umfassen unter anderem die Themen rückwärtsfahrende Müllsammler, Aufzeichnen des Fahrzeugdatenbusses (Controller-Area-Network-Bus (CAN)), Verschmutzungen auf Straßen und Gehwegen und die Erkennung von Straßenschildern.

Der Projektpartner ENQT hat ein primäres Interesse daran, die gesammelten Daten zu verarbeiten, um sie externen Organisationen zur Verfügung zu stellen. Dabei wurde eine

Liste möglicher Themen erstellt. Diese reicht von Anlagen der Energieversorger über Anzahl und Ort von Elektroautos bis hin zu der Beschichtung von Gebäudefenstern.

In dieser Bachelorarbeit wird der Fokus, für den Stakeholder ENQT, auf die Ermittlung von Gebäudehöhen gelegt. Das zu entwickelnde System soll mit dieser Information eine Korrelation aus Gebäudehöhe und Mobilfunkqualität ermöglichen, um herauszufinden, ob die Gebäudehöhe ein Maß für eine gute oder schlechte Mobilfunkqualität ist.

Die Interessen der HAW Hamburg als Projektpartner sind bildungsorientiert. Das Projekt UrbanRFMap sorgt für diverse Forschungsthemen und ihre Ergebnisse, welche innerhalb von wissenschaftlichen Veröffentlichungen publiziert werden können.

4.2 Anforderungsanalyse

Im Folgenden werden die verschiedenen Anforderungen der Stakeholder Hamburger Stadtreinigung, ENQT und HAW Hamburg aufgelistet und ausgearbeitet.

Aus den Erwartungen der Hamburger Stadtreinigung ergeben sich verschiedene Anforderungen an das Messsystem. Kein Teil des Messsystems darf in den Innenraum der Fahrerkabine montiert werden. Die Kabine ist ausschließlich für die Mitarbeiter reserviert um persönliche Dinge, Equipment und Lebensmittel zu verstauen. Das Messsystem muss soweit automatisiert sein, dass die Datenaufzeichnung ohne menschliche Interaktion beginnt, sobald sich das Trägerfahrzeug in Bewegung setzt. Um Mülltonnen für die Hamburger Stadtreinigung zu erkennen ist es notwendig, dass mindestens eine Kamera verwendet wird. Für die Messung von Durchfahrtbreiten in der realen Welt, bieten sich Lidar- Punktwolken an. Folglich ist für die Durchfahrtsbreitenvermessung ein Lidar notwendig. [10.2]

Die Projektpartner ENQT und HAW-Hamburg definieren für das zu entwickelnde Messsystem eigene Anforderungen. Das System soll robust und widerstandsfähig sein. Dies bezieht sich auf jegliche Witterungsbedingungen, sowie thermische und mechanische Belastungen. Die Rahmenbedingungen hierfür werden durch die Klimazone Hamburg festgelegt. Hamburg befindet sich in einer gemäßigten Klimazone. Das System muss Temperaturen von -10 °C bis 40 °C, Niederschlägen und mechanischen Einwirkungen, wie zum Beispiel Wind und Vibrationen, standhalten.

Hierfür soll der Messaufbau die IP-Schutzklasse IP 65 erfüllen. Der Messaufbau soll somit staubdicht, vollständig berührungssicher und vor Strahlwasser geschützt sein. [28]

Für die Gebäudehöhenbestimmung ist, wie bei der Durchfahrtsbreitenvermessung, ein Lidar notwendig. Aufgrund der unterschiedlichen Betrachtungsbereiche, von Durchfahrtbreite auf der Straße und den Gebäudehöhen neben der Straße kann aus kostentechnischen Gründen nur ein Lidar verwendet werden. Daher muss es sich dabei um einen 3D Lidar handeln, welcher einen Sichtbereich von mindestens 180° besitzt. Die Reichweite der Messung soll bis zu 70 Metern möglich sein, um hohe Gebäudefronten vermessen zu können.

Die georeferenzierte Lokalisierung des Systems darf maximal 50 cm von seiner realen Position abweichen. Dafür ist ein GNSS-System notwendig.

Die Auslegung des Systems soll einem modularen Ansatz unterliegen. Es ist gefordert, dass die Option zur Erweiterung des Systems, durch zusätzliche Sensorik, gegeben ist.

Das Messsystem muss weitestgehend autark und wartungsarm sein. Es wird dafür eine Spannungsversorgung über das Trägerfahrzeug zur Verfügung gestellt. Alle weiteren Verbindungen innerhalb und außerhalb des Systems sind von dem Trägerfahrzeug unabhängig. Das Messsystem ist soweit zu automatisiert, dass keine weiteren Wartungen vor Ort notwendig sind. Zum Ausführen von Wartungen, zum Beispiel für Softwareupdates, ist ein Remotezugriff über eine dauerhafte Serververbindung zu implementiert.

Es ist gefordert, dass das Messsystem selbstüberwacht und ausfallsicher ist. Dies bedeutet, dass das System im Fehlerfall automatisch einen Statusbericht an den Server übermittelt und anschließend einem definierten Verhalten folgt. Dies kann der automatisierte Neustart des Systems zur Fehlerbehebung sein.

Eine weitere Anforderung, welche sich aus den Projektzielen ergibt, ist die Einhaltung der Kostenkalkulation des Projektantrags.

5 Prototyp 1

Im folgenden Kapitel wird zunächst das Fahrzeugsystem beschrieben, welches als erster Prototyp des zu entwickelnden Messsystems dient. Anschließend wird ein Überblick über die erzeugten Messdaten des Prototypen gegeben. In Kapitel sechs wird anhand dieser Messdaten Software entwickelt, um die Anforderungen der Interessengruppen zu erfüllen.

5.1 Aufbau Prototyp 1

Die Hochschule für Angewandte Wissenschaften Hamburg besitzt im Urban Mobility Lab als Testfahrzeug einen Tesla Model S, wie in Abbildung 12 dargestellt. Dieses Testfahrzeug ist mit einer Rahmenstruktur als Dachgepäckträger ausgestattet. An diesem Rahmen befinden sich diverse Sensoren zur Umfeldvermessung. Das Gesamtkonstrukt wird in dieser Arbeit als Prototyp 1 definiert.

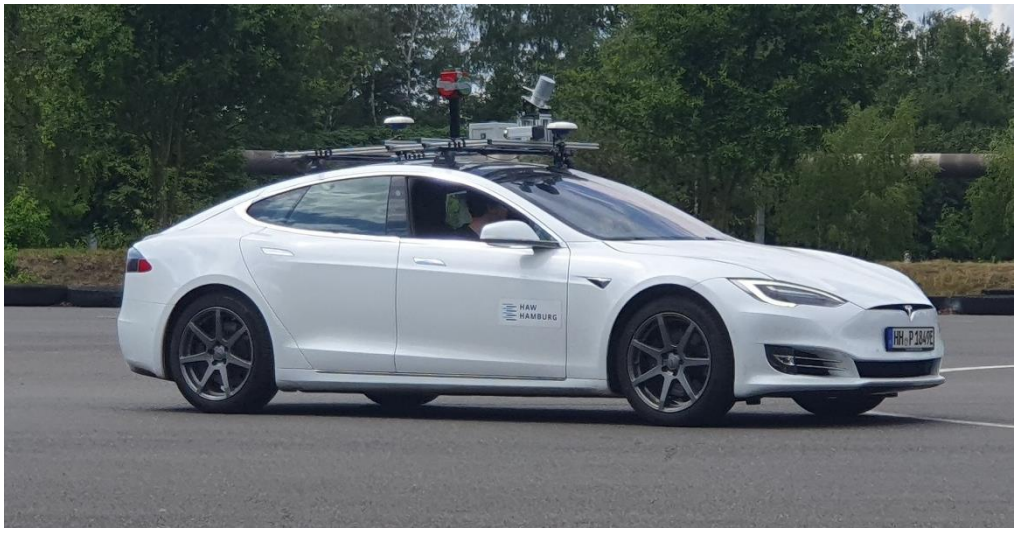


Abbildung 12: Tesla Verkehrsübungsplatz [2]

Im Kofferraum des Prototyp 1 befindet sich ein Computer zur Datenverarbeitung und Datenspeicherung. Zur Steuerung des Computers wird das Betriebssystem Ubuntu 20.04 LTS und das Metabetriebsystem ROS (Robot Operating System) genutzt [29]. Basierend auf dem Publisher / Subscriber – Modell können Programme in ROS Daten veröffentlichen, auf welche andere Programme Zugriff haben. Zur Datensicherung werden Rosbags genutzt. Innerhalb eines ROS-Systems können Rosbags abgespielt

und wiederholt angesehen werden. Der Computer besitzt eine Intel I7-7700k CPU (Central Processing Unit), zwei Mal ein Terrabyte SSD-Speicher (Solid State Disk) und 64 Gigabyte RAM (Random Access Memory).

Eine Anforderung der Stadtreinigung Hamburg ist die Bereitstellung der CAN-Bus-Daten. Um dies zu realisieren, wurde eine Schnittstelle zum CAN-Bus implementiert. Durch eine Software in ROS wird der CAN-Bus mitgelesen und die Nachrichten gespeichert. Dabei ist es sicherheitsrelevant, dass im Fahrbetrieb der CAN-Bus ausschließlich gelesen werden kann.

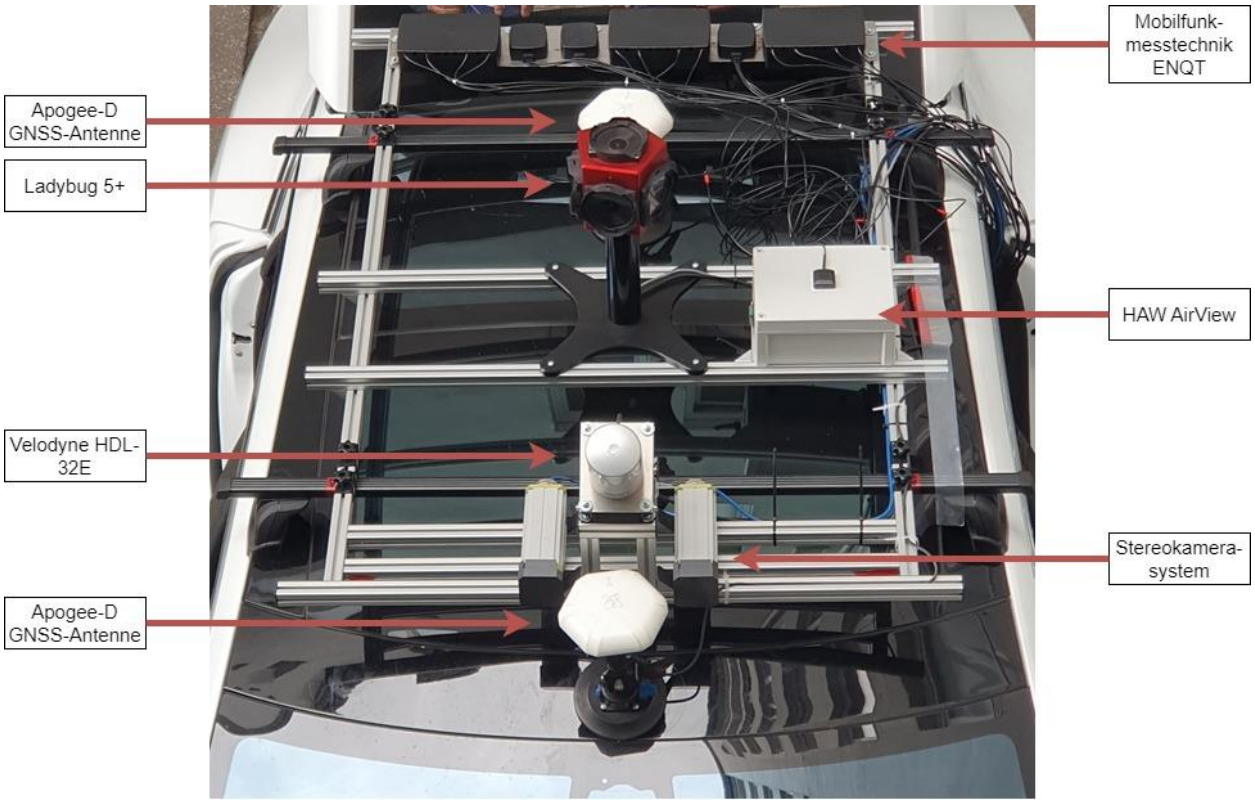


Abbildung 13: Sensoraufbau Dachträger [2]

Im folgenden Absatz werden die Sensoren des Prototyp 1 beschrieben, hier in Abbildung 13 sichtbar. Als bildgebenden Umfeldsensor besitzt der Tesla eine Ladybug 5+ Kamera. Die Kamera wird mit einem Bild pro Sekunde betrieben und überträgt die Bilder mittels einer USB3.1-Schnittstelle an den Computer im Kofferraum des Fahrzeugs. Dort werden die Bilder von ROS verarbeitet und gespeichert. Zusätzlich verfügt das Testfahrzeug über ein Stereokamerasystem, welches in Fahrtrichtung orientiert sind. Die Kameras sind ebenfalls mit einer Bildrate von einem Hertz (Hz) konfiguriert und übermitteln ihre Daten,

mittels einer USB3.0-Schnittstelle, an den Fahrzeugcomputer. Ein weiterer Umfeldsensor des Testfahrzeugs ist der Velodyne HDL-32E Lidar. Dabei handelt es sich um einen rotierenden Laserscanner mit 32 Einzelstrahlen, einem Scanbereich von 41.33° x 360° und einer Reichweite von 100 Metern. Die Datenerhebung findet mit einer Frequenz von 10 Hz statt. Pro Lidarmessung werden zwischen 695.000 und 1.390.000 Datenpunkte erzeugt. Der Lidar ist an das ROS-System angebunden und veröffentlicht dort die Umgebungsvermessungen in Form von PointCloud2 Punktwolken. Für diese Arbeit wurde der Lidar-Scanner um 30° nach vorne geneigt um Bereiche der Straße, sowie Gebäudefronten und Gebäudehöhen, vermessen zu können.

Ergänzend ist das Koppelnavigationssystem Apogee-D des Unternehmens SBG Systems an dem Fahrzeug angebracht.

Der Prototyp 1 ist mit dem hochschuleigenen Messsystem HAW-AirView, zur Messung von Luftgüte und Luftqualität, ausgestattet. Die Messdaten sind Lufttemperatur, Luftfeuchtigkeit, Co2-Gehalt, flüchtige organische Verbindungen und verschiedene Partikelgrößen unter 10 µm. Zur Bereitstellung der Messdaten kommuniziert das Messsystem über WLAN mit dem ROS-System.

Der Prototyp 1 ist durch den Projektpartner ENQT mit Mobilfunkmesstechnik ausgestattet. Das Messsystem wird, bis auf eine Spannungsversorgung, autark betrieben. Die aufgezeichneten Daten der Mobilfunkqualität werden nicht auf dem Fahrzeugcomputer gespeichert, sondern auf den Servern des Projektpartners.

Abbildung 14 veranschaulicht die beschriebenen Datenverbindungen der Sensoren sowie deren ungefähre räumliche Anordnung am Fahrzeug.

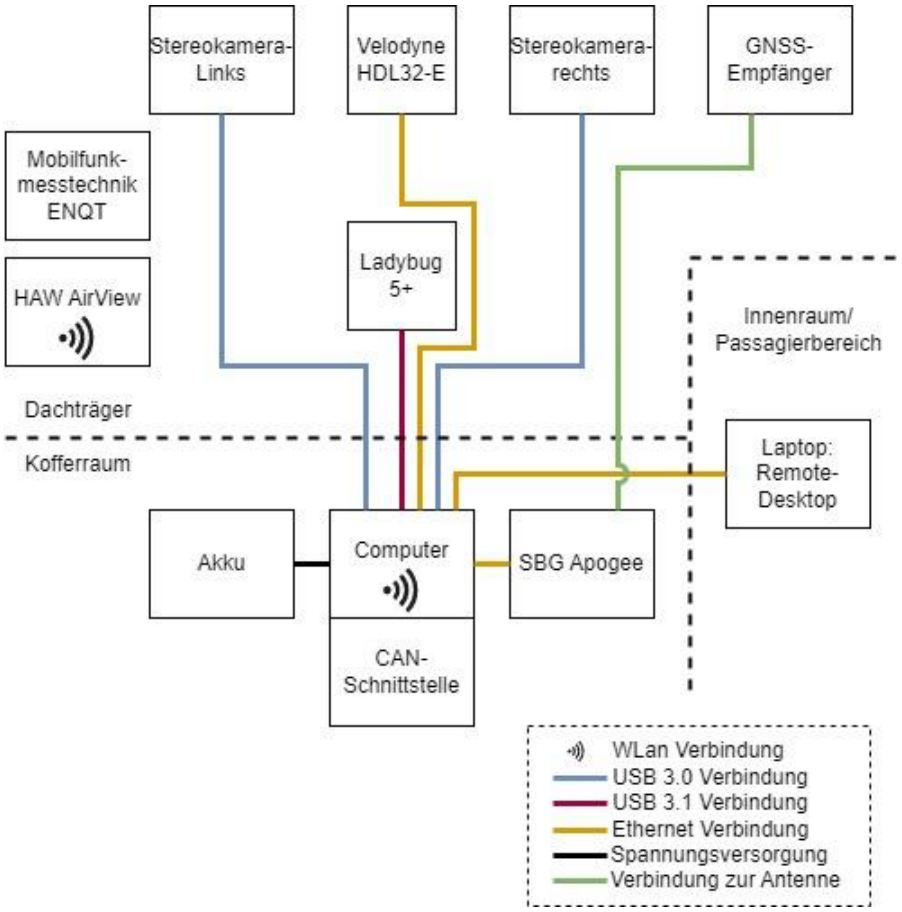


Abbildung 14: Verbindungen des Fahrzeugaufbaus [2]

5.2 Messfahrt Prototyp 1

Die Verfügbarkeit von Messdaten zu Beginn des Entwicklungsprozesses eröffnet die Möglichkeit einer präziseren Abstimmung des Prototyp 2 auf die im Vorfeld definierten Anforderungen. Dieser Vorgang vereinfacht die Entwicklung der Software für das Messsystem, indem die Notwendigkeit wiederholter Testfahrten für jeden Entwicklungsabschnitt vermieden wird. Dieser Ansatz des Rapid-Prototyping optimiert den Entwicklungsprozess und spart Zeit und Ressourcen.

Die Messfahrt des Prototyp 1 wurde auf der Teststrecke für automatisiertes und vernetztes Fahren in Hamburg (TAVF) durchgeführt. Die Abbildung 15 zeigt einen Überblick über die, während der Messfahrt, erzeugten Daten. Der abgebildete Datensatz enthält 14 Sekunden der Messfahrt mit den unter „topics:“ aufgenommenen Messwerten. Zusätzlich enthält diese Rubrik die Anzahl der jeweils aufgenommenen Messwerte pro Messsystem, sowie deren Nachrichtentyp in ROS.

```
duration:    14.0s
start:       May 24 2023 14:32:10.60 (1684931530.60)
end:         May 24 2023 14:32:24.57 (1684931544.57)
size:        1.6 GB
messages:    38827
compression: none [182/182 chunks]
types:       can_msgs/Frame                    [64ae5cebf967dc6aae4e78f5683a5b25]
             diagnostic_msgs/DiagnosticArray   [60810da900de1dd6ddd437c3503511da]
             rosgraph_msgs/Log                 [acffd30cd6b6de30f120938c17c593fb]
             sbg_driver/SbgEkfEuler            [4862d08d71471abacf02c798eaa2e18f]
             sbg_driver/SbgEkfNav              [16e5ed53c5544dda563fc67fb816d9b9]
             sbg_driver/SbgGpsHdt              [8c984f2f80abb760d3b4701ef29aeceb]
             sbg_driver/SbgGpsPos              [6b214c87825603003c01f4e03d945a32]
             sbg_driver/SbgGpsVel              [dc36a4705c96041ace5f0875af58a725]
             sbg_driver/SbgImuData             [59cc541d794c367e71030fa700720826]
             sbg_driver/SbgStatus              [1b73c890bd111d40339f4be9a7495e96]
             sbg_driver/SbgUtcTime             [89495f07708fa38e487b6509c4edabaa]
             sensor_msgs/Image                 [060021388200f6f0f447d0fcd9c64743]
             sensor_msgs/PointCloud2           [1158d486dd51d683ce2f1be655c3c181]
             std_msgs/String                   [992ce8a1687cec8c8bd883ec73ca41d1]
             tf2_msgs/TFMessage                [94810edda583a504dfda3829e70d7eec]
             velodyne_msgs/VelodyneScan        [50804fc9533a0e579e6322c04ae70566]
topics:      /camera/image_raw                14 msgs    : sensor_msgs/Image
             /ccs811_co2                      28 msgs    : std_msgs/String
             /ccs811_tvoc                     28 msgs    : std_msgs/String
             /dht11_humidity                  27 msgs    : std_msgs/String
             /dht11_temperature               27 msgs    : std_msgs/String
             /diagnostics                     36 msgs    : diagnostic_msgs/DiagnosticArray
             /gnss_gga_string                  2 msgs    : std_msgs/String
             /gnss_latitude                    2 msgs    : std_msgs/String
             /gnss_longitude                   2 msgs    : std_msgs/String
             /gnss_qual                        2 msgs    : std_msgs/String
             /gnss_sats                        2 msgs    : std_msgs/String
             /gnss_time                        2 msgs    : std_msgs/String
             /left_cam_node/image_raw         14 msgs    : sensor_msgs/Image
             /received_messages            34922 msgs    : can_msgs/Frame
             /right_cam_node/image_raw        14 msgs    : sensor_msgs/Image
             /rosout                          59 msgs    : rosgraph_msgs/Log
             /rosout_agg                      59 msgs    : rosgraph_msgs/Log
             /sbg/ekf_euler                  698 msgs    : sbg_driver/SbgEkfEuler
             /sbg/ekf_nav                    698 msgs    : sbg_driver/SbgEkfNav
             /sbg/gps_hdt                     70 msgs    : sbg_driver/SbgGpsHdt
             /sbg/gps_pos                     70 msgs    : sbg_driver/SbgGpsPos
             /sbg/gps_vel                     70 msgs    : sbg_driver/SbgGpsVel
             /sbg/imu_data                  1397 msgs    : sbg_driver/SbgImuData
             /sbg/status                      14 msgs    : sbg_driver/SbgStatus
             /sbg/utc_time                    14 msgs    : sbg_driver/SbgUtcTime
             /tf                             278 msgs    : tf2_msgs/TFMessage
             /velodyne_packets               139 msgs    : velodyne_msgs/VelodyneScan
             /velodyne_points                139 msgs    : sensor_msgs/PointCloud2
```

Abbildung 15: Messfahrtdaten [2]

Zu den Umfelddaten gehören, die topics „/velodyne_points“, „/camera/image_raw“, „/left_cam_node/image_raw“ und „/right_cam_node/image_raw“. Die Variable „/velodyne_points“ beinhaltet 139 aufeinanderfolgende Messungen des Lidar in einem PointCloud2-Format. In der Variablen „/camera/image_raw“ sind 14 Kameraaufnahmen der Ladybug 5+ im ROS-Image-Format gespeichert. „/left_cam_node/image_raw“ und „/right_cam_node/image_raw“ bilden zusammen die Variablen der Stereokamera, mit jeweils 14 Bildaufnahmen im ROS-Image-Format.

Während der 14 Sekunden der Datenaufnahme wurden 34922 Nachrichten des CAN-Bus, in „/received_messages“, als CAN-Nachricht-Format gespeichert.

Die Messdaten des HAW AirView befinden sich als String in den topics „/ccs811_co2“, „/ccs811_tvoc“, „dht11_temperature“ und „dht11_humidity“.

Das GNSS-System des Apogee-D erzeugte 70 Positionsbestimmungen mit Längen- und Breitengraden, welche in der topic „/sbg/gps_pos“ gespeichert sind. Die Inertialsensorik generierte in der topic „/sbg/imu_data“ 1397 Messwerte. Die Kombination beider Informationen und somit die präziseste Positionsbestimmung befindet sich in den topics „/sbg/ekf_nav“ und „/sbg/ekf_euler“.

6 Softwareentwicklung

Die in Kapitel 5 erzeugten und beschriebenen Messwerte dienen in diesem Kapitel als Grundlage für die Softwareentwicklung. Anhand der Daten soll herausgearbeitet werden, mit welchen Algorithmen und Sensoren die Informationsanforderungen der Stakeholder bedient werden können.

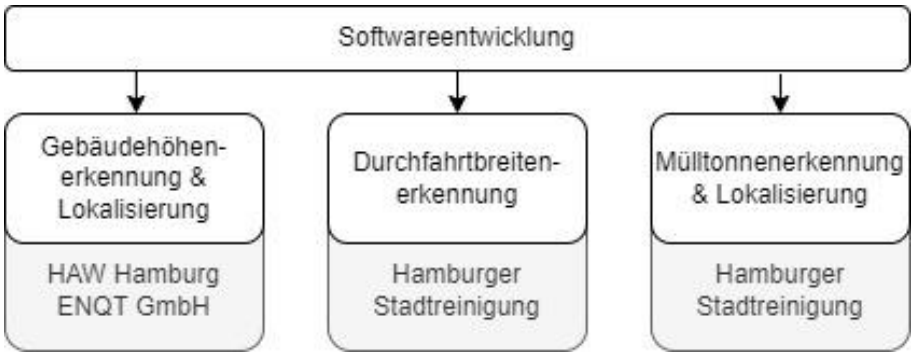


Abbildung 16: Überblick Softwareentwicklung [2]

Wie in der Abbildung 16 ersichtlich, befasst sich die Softwareentwicklung mit den Themen Mülltonnenerkennung und Lokalisierung, Durchfahrtbreitenerkennung und der Bestimmung und Lokalisierung der Gebäudehöhen. Innerhalb eines Themas wird zuerst die Fragestellung erwähnt und im Anschluss das Vorgehen erläutert. Darauf folgt die Umsetzung, das Testen und die Bewertung.

6.1 Gebäudehöhenerkennung und Lokalisierung

Folgend ist die Software für eine Gebäudehöhenerkennung zu entwickeln. Dabei ist die Frage zu beantworten, wie es möglich ist, die Gebäudehöhe mittels einer Lidarmessung zu bestimmen. Methodisch wird ein experimenteller Ansatz gewählt.

6.1.1 Vorgehensweise

Inhalt der experimentellen Untersuchung ist die manuelle Sichtung der Lidardaten zur Identifikation markanter Stellen. Bei diesen Stellen handelt es sich um Abschnitte der Messfahrt mit besonders hohen, kleinen oder keinen Gebäuden. Abschnitte mit Hindernissen, wie zum Beispiel Bäume zwischen Lidar und Gebäude, werden zusätzlich

untersucht. Mit der Bestimmung dieser Abschnitte werden Grenzfälle identifiziert. Dies ist für die Entwicklung einer universal anwendbaren Applikation notwendig.

Im Anschluss an die Sichtung wird mit der Entwicklung experimenteller Softwareansätze gestartet. Durch das wiederholte Abspielen der Softwareansätze in ROS, mit den Daten der Testfahrt, werden die Ansätze getestet. Durch die Bewertung der Testergebnisse kann der Softwareansatz optimiert werden. Dieses Vorgehen wird wiederholt, bis eine zielerreichende Software entwickelt ist. Die finale Version wird mit den Daten der Rosbags qualitativ in Bezug auf die Zielerreichung getestet und bewertet.

6.1.2 Umsetzung

Die Software startet mit der Integration des Scripts in ROS. Es werden drei Subscriber erstellt. Einen für die Apogee-D-Position, einen für die Apogee-D-Ausrichtung und ein dritter für die Velodyne Punktwolke. Immer wenn eine Velodyne Punktwolke veröffentlicht wird und das Programm im Idle Zustand ist, beginnt der Programmablauf.

Die Abbildung 17 zeigt den systematischen Ablauf der Gebäudehöhenerkennung und Lokalisierung. Die Lidardaten werden eingelesen und in ein bearbeitbares Format umgewandelt. Anschließend wird die Punktwolke bearbeitet. Datenpunkte unterhalb von fünf Metern über der Straße und Datenpunkte, die neben dem Testfahrzeug liegen, werden entfernt. Datenpunkte, die zwischen -10 m und -80 m hinter dem Fahrzeug liegen und eine Mindesthöhe von 5 m aufweisen, bleiben erhalten. Im nächsten Schritt wird die Punktwolke an der Fahrzeuglängsachse, also durch den Ursprung des Koordinatensystems, in eine linke und ein rechte Punktwolke aufgeteilt.

Das Datenformat der Punktwolken wird in Open3d-PointClouds umgewandelt, um die Funktionen der Bibliothek Open3d nutzen zu können. Im Anschluss wird eine statistische Ausreißerentfernung durchgeführt. Die „Statistic Outlier Removal“ (SOR) berechnet die Entfernung der Datenpunkte untereinander, sowie die Standardabweichung. Liegen Werte nicht in der Nähe des Durchschnitts, werden sie als Ausreißer klassifiziert und aus der Punktwolke entfernt. Dieser Prozess ist, durch die Angabe der Standardabweichung und gewöhnlicher Wegstrecken zwischen Datenpunkten, konfigurierbar.

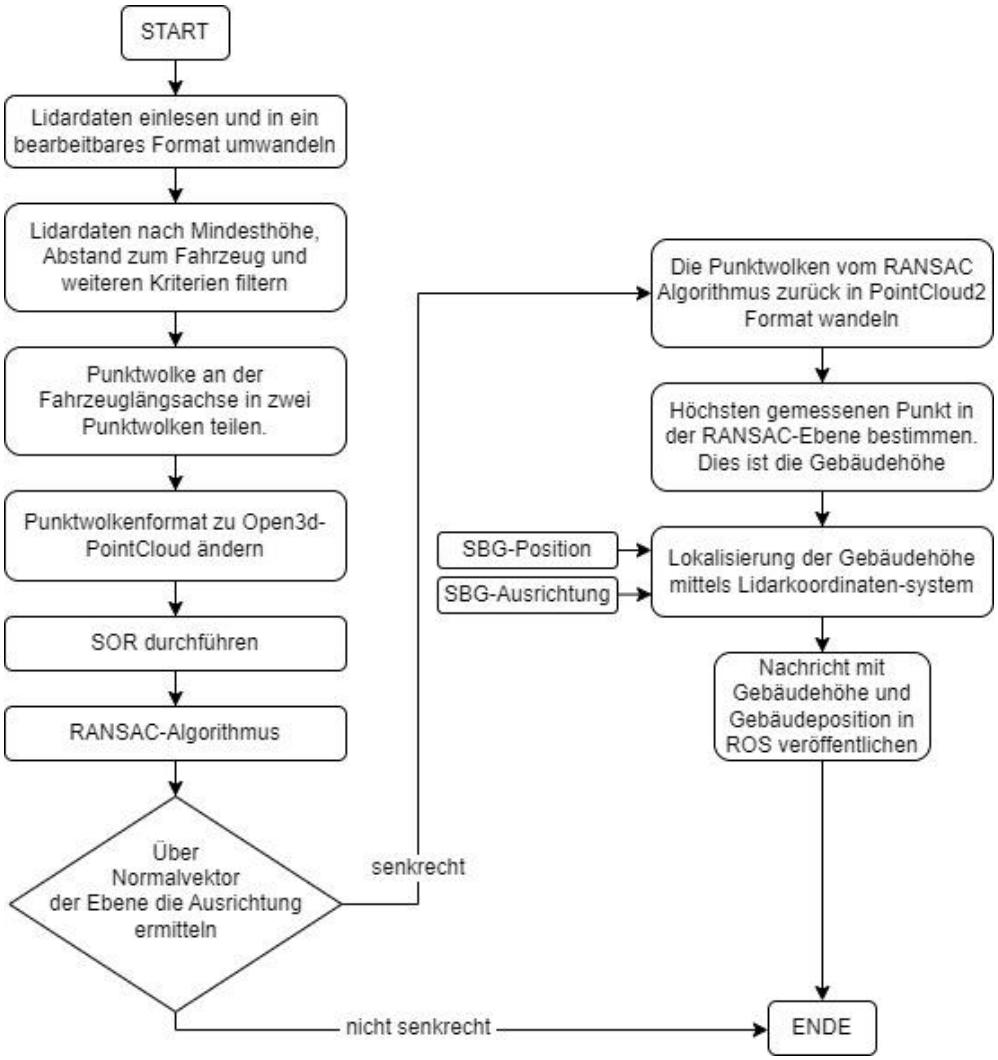


Abbildung 17: Ablauf Gebäudehöhenerkennung [2]

Danach wird ein Random-Sample-Consensus-Algorithmus (RANSAC) verwendet, um Ebenen in den Punktwolken zu lokalisieren. Dabei startet der Algorithmus mit einer festgelegten Anzahl zufällig ausgewählter Datenpunkte. Diese Datenpunkte werden verwendet, um Ebenen durch sie hindurchzulegen. Die Anzahl der zu erstellenden Ebenen ist konfigurierbar. Anschließen ermittelt der Algorithmus weitere Datenpunkte, welche sich in diesen Ebenen befinden. Die Ebene, welche die meisten Datenpunkte beinhaltet, wird abschließend als Ergebnis der Funktion, in Form von Ebenenparametern zurückgegeben. Zusätzlich wird eine neue Punktwolke erstellt, die die Datenpunkte innerhalb dieser Ebene enthält.

Die Ebenenparameter bilden einen Normalvektor der Ebene. Ist der Normalvektor parallel zur X,Y-Ebene befindet sich die Ebene senkrecht im Raum. Senkrechte Strukturen mit

glatten Oberflächen in Urbanen Bereichen, in einer Höhe größer 5 m, sind in den meisten Fällen Gebäudefassaden.

Ist das Ergebnis der Normalvektoruntersuchung eine senkrechte Ebene, wird die neu erstellte Punktwolke in ein bearbeitbares Array-Format transformiert. Anschließend wird der höchste gemessene Punkt innerhalb der Punktwolke ermittelt. Dieser Punkt ist die, durch den Algorithmus bestimmte, Gebäudehöhe.

Für die Lokalisierung der Gebäudeposition wird zunächst in dem Lidarkoordinatensystem der Abstand des Gebäudehöhenmesspunktes zum Lidar und der Winkel zur Fahrzeuglängsachse bestimmt [30]. Im Anschluss wird die GNSS-Position und Ausrichtung des Apogee-D eingelesen. Diese Informationen bestimmen, in Kombination mit dem UTM-Koordinatensystem, die georeferenzierte Gebäudeposition. Für die Visualisierung der Positionen werden diese in WGS84-Koordinaten umgewandelt. Abschließend werden die gemessenen Gebäudehöhen und Gebäudepositionen in einer ROS-Nachricht veröffentlicht [10.7].

Die nachfolgenden Abbildungen visualisieren Teile des Entwicklungsverlaufs.

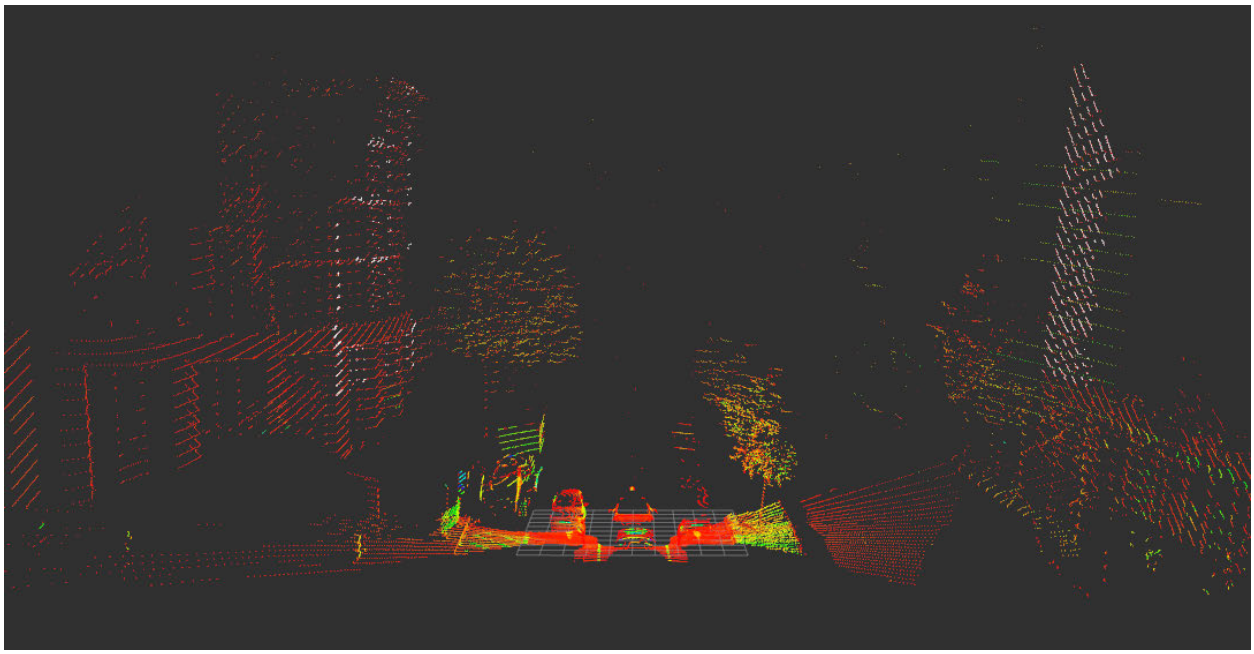


Abbildung 18: Gebäudehöhenbestimmung Lidarbild [2]

Die Abbildung 18 zeigt eine Visualisierung der Messpunkte des Lidar. Die farbigen Datenpunkte stellen die Originalpunktwolke der Lidarmessung dar. Die weißen

Datenpunkte zeigen die durch den RANSAC-Algorithmus bestimmten Ebenen nach der Normalvektoruntersuchung.

Abbildung 19: Gebäudehöhenbestimmung Ladybug 5+ Referenzbild [2]

Um einen Eindruck über das vermessene Umfeld zu erlangen, zeigt die Abbildung 19 das zur Lidaraufnahme passende Ladybug 5+ Bild. Für die verbesserte Erkennbarkeit wurde ein Rahmen um die Einzelbilder gelegt. Vergleicht man beide Visualisierungen miteinander, zeigt das dritte und vierte Einzelbild der Ladybug 5+ ein Gebäude mit abgerundeter Fassade. Dieses Gebäude ist in Abbildung 18 auf der linken Seite zu erkennen. Die Gebäudefront der rechten Lidarbildhälfte ist in dem ersten und zweiten Ladybug 5+ Bild zu sehen.

6.1.3 Test

Zum Testen der Verfahrens wurde ein Abschnitt des Straßennetzes in der Hamburger Hafencity ausgewählt. Die Abbildung 20 zeigt, in Violett, die Messpositionen des Fahrzeugs, sowie, in Rot, die durch den Algorithmus ermittelten Gebäudehöhen in dieser Umgebung.

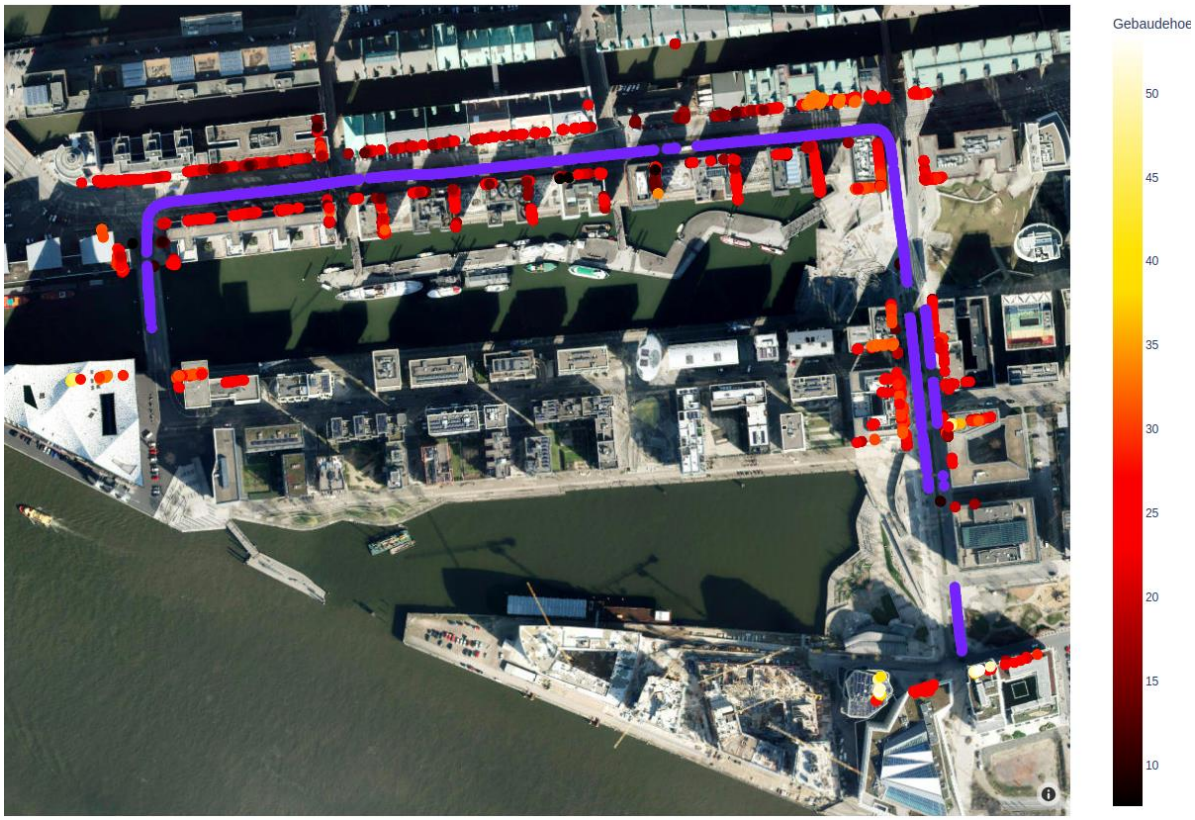


Abbildung 20: Gebäudepositionen und Gebäudehöhen [2]

Die erfassten Gebäudefassaden sind deutlich erkennbar und stimmen mit den Positionen auf der Satellitenkarte überein.

6.1.4 Bewertung

Für die Bewertung der Höhenerkennung wurden vier Gebäude des Tests exemplarisch ausgewählt.

Abbildung 21: Gebäudeauswahl zur Bewertung [2]

Die Abbildung 21 zeigt diese vier Gebäude. Das Gebäude 1 besitzt eine Höhe von 24 Metern. Der von der Software ermittelte Höhenwert beträgt minimal 20,31 Meter und maximal 23,73 Meter. Der Mittelwert der Messungen beträgt 23,18 Meter mit einer Standardabweichung von 0,62 Metern.

Das Gebäude 2 besitzt eine Höhe von 30 Metern. Der ermittelte Höhenwert beträgt minimal 20,02 Meter und maximal 30,28 Meter. Der Mittelwert der Messungen beträgt 25,19 Meter mit einer Standardabweichung von 2,87 Metern.

Gebäude 3 besitzt ebenfalls eine Höhe von 30 Metern. Hier beträgt der ermittelte Höhenwert minimal 21,40 Meter und maximal 30,28 Meter. Der Mittelwert der Messungen beträgt 26,40 Meter mit einer Standardabweichung von 3,17 Metern.

Das letzte Gebäude besitzt eine Höhe von 30 Metern. Die minimal und maximal ermittelten Höhenwerte betragen 24,27 und 31,33 Meter. Der Mittelwert beträgt 29,69 Meter mit einer Standardabweichung von 1,81 Metern.

Die Ergebnisse zeigen, dass der maximal ermittelte Höhenwert pro Gebäude das beste Maß für die Höhenbestimmung darstellt. Der Lidar ist nicht in der Lage höhere Datenpunkte als die maximale Gebäudehöhe zu detektieren, da dort keine Reflektion mehr auftritt. Schlussfolgernd kann der maximale Höhenwert als maximale Gebäudehöhe gewertet werden. Die Schwankungen der Messungen ergeben sich durch die Ebenenerkennung und die Filterung der Daten. Liegt zum Beispiel das obere Gebäudestockwerk zur Fassade nach hinten versetzt, ist dieses nicht Teil der optimal gefundenen Ebene und fließt nicht in die Gebäudehöhenberechnung mit ein. Werden bei der Messung Datenpunkte ermittelt, die außerhalb der Standardabweichung liegen, werden diese von dem SOR-Algorithmus herausgefiltert. Ist ein solcher Datenpunkt eine maximale Gebäudehöhe, wird dieser in die Höhenberechnung nicht mit einbezogen.

Dem Algorithmus ist es möglich, Gebäudefassaden von anderen Strukturen zu unterscheiden. Die Einschränkung, dass die ermittelte Ebene des RANSAC-Algorithmus senkrecht im Raum stehen muss, führt zu einer minimalen Anzahl an Fehldetektionen anderer Strukturen. Die Abbildung 22 zeigt die Visualisierung einer Punktwolke, in welcher sich keine Gebäude in der unmittelbaren Umgebung zum Fahrzeug befinden. Lediglich Bäume und Teile der Straße sind zu erkennen. Die Gebäudehöhenerkennung ordnet dieser Punktwolke keine vertikale Ebene zu. Folglich wird kein gefundenes Gebäude als Programmergebnis zurückgegeben.

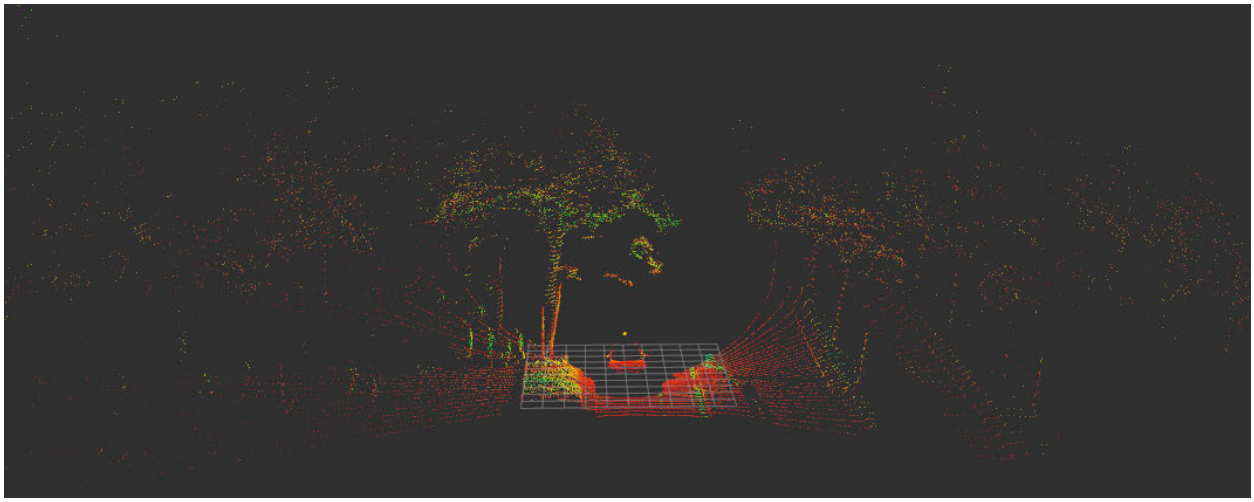


Abbildung 22: Gebäudehöhenerkennung Verhalten ohne Gebäude [2]

Die Gebäudehöhenerkennung hat in 75% der Tests mit einer Toleranz von 0,5 Metern und in 25% der Tests mit einem Fehler von 1,5 Metern zufriedenstellend funktioniert.

Auch die Lokalisierung des Messpunktes mit anschließender Positionsbestimmung funktioniert zufriedenstellend. In Abbildung 18 kann festgestellt werden, dass die erfassten WGS84-Koordinaten mit den projizierten WGS-Karten-Längen- und Breitengraden übereinstimmen.

Abschließend kann die Aussage getroffen werden, dass der experimentelle Softwareansatz erfolgreich ausgearbeitet wurde und die Software zur Gebäudehöhenerkennung und Lokalisierung eingesetzt werden kann.

6.2 Durchfahrtbreitenerkennung

Die folgend zu entwickelnde Software soll die aktuelle Durchfahrtbreite vor dem Trägerfahrzeug bestimmen. Dabei handelt es sich um den freien Bereich vor dem Fahrzeug, welcher durch Bordsteine oder andere Verkehrsteilnehmer begrenzt wird. Die Fragestellung besteht darin, wie die Lidarsensorik zur Ermittlung der maximalen Durchfahrtbreite einer Straße genutzt werden kann. Die Software wird auf der Grundlage einer experimentellen Untersuchung von Lidardaten entwickelt.

6.2.1 Vorgehensweise

Die Vorgehensweise ähnelt der Entwicklung der Gebäudehöhenerkennung. Zuerst werden die Daten der Messfahrt manuell gesichtet und Grenzfälle identifiziert. Die Grenzfälle werden als breite Straßenabschnitte, schmale Straßenabschnitte und Abschnitte mit Hindernissen vor dem Fahrzeug, definiert. Um die Erkennung von Grenzfällen in den Lidardaten zu vereinfachen, werden die Kameradaten der Messfahrt zuerst gesichtet. Sind dort Grenzfälle enthalten, werden anschließend die zugehörigen Lidardaten betrachtet. Die Grenzen der Durchfahrtbreite sind als Sprünge aufeinanderfolgender Datenpunkte in den Lidarkoordinaten sichtbar. Der weitere Verlauf der Vorgehensweise ist mit dem der Gebäudehöhenerkennung identisch. Experimentelle Softwareansätze werden herausgearbeitet, durch Zwischenergebnisse optimiert und bei Zielerreichung getestet und bewertet.

6.2.2 Umsetzung

Die Software startet mit der ROS-Integration. Dabei werden zwei Subscriber erstellt. Einen für die Apogee-D-Position und einen für die Velodyne Punktwolke. Der Programmablauf beginnt, sobald sich das Programm im Wartezustand befindet und eine Velodyne Punktwolke in ROS veröffentlicht wird. Im Folgenden werden drei verschiedene experimentelle Herangehensweisen erläutert.

Variante 1:

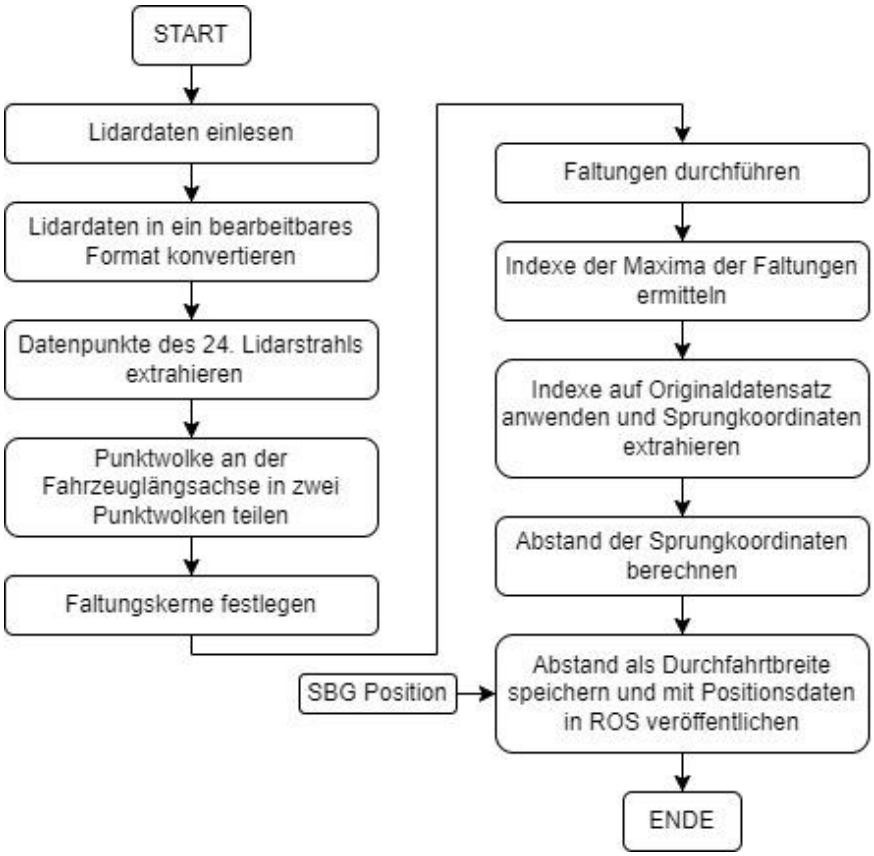


Abbildung 23: Ablauf Durchfahrtbreitenerkennung mit Faltung [2]

Die Abbildung 23 visualisiert den Ablauf der Variante 1 einer Durchfahrtbreitenerkennung. Dazu wird die Faltung des Originaldatensatzes mit einem Faltungskern zur Sprungerkennung durchgeführt. Im ersten Schritt werden die Lidardaten eingelesen und in ein bearbeitbares Format konvertiert. Anschließend wird der Datensatz durch die Extraktion der Daten des 24ten Lidarstrahls verkleinert und auf den zu betrachtenden Bereich angepasst. In Abbildung 24 ist zu erkennen, dass die Motorhaube des Tesla einen Schatten im Lidarbild erzeugt. Der 24te Lidarstrahl, hier in größeren Datenpunkten dargestellt, ist der erste Strahl, der nicht an der Motorhaube reflektiert, sondern die Straße davor vermisst. Aufgrund seiner Eigenschaften wird der Lidarstrahl für die weitere Auswertung ausgewählt.

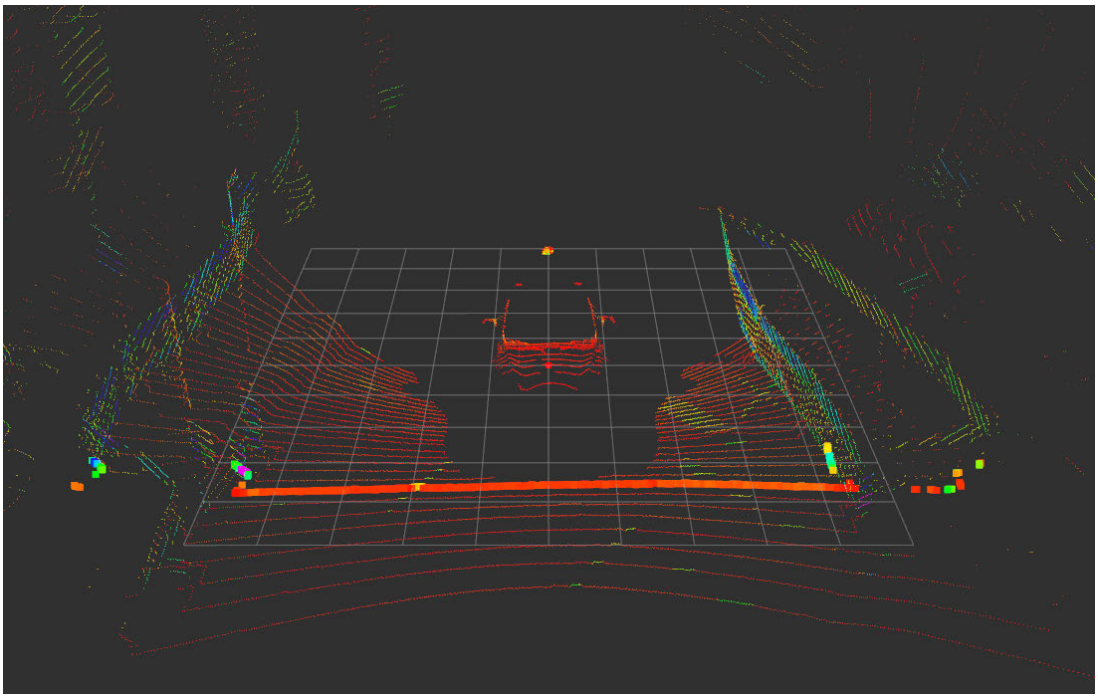


Abbildung 24: Lidarbild Durchfahrtsbreitenerkennung [2]

Folgend wird der Datensatz an der Fahrzeuglängsachse in den linken und rechten Bereich aufgeteilt. Dieses Vorgehen ist notwendig, da die Sprungdetektion ausschließlich den größten Sprung zurückgibt. Danach wird mit Hilfe eines Faltungskerns eine Faltung mit dem Datensatz durchgeführt. Ein neues Faltungssignal wird berechnet. Dieses Signal besitzt Maxima an den Sprungstellen des Originalsignals. Die Indexe der Maxima für den linken und rechten Bereich werden gespeichert. Die Koordinaten des markantesten Sprungs im linken und rechten Bereich befinden sich an den gespeicherten Indexen im Originalsignal. Mit diesen Koordinaten wird der Abstand der beiden ermittelten Sprünge berechnet. Abschließend wird dieser Abstand als Durchfahrtbreite, mit Positionsdaten des Apogee-D, als ROS-Nachricht veröffentlicht [10.7].

Variante 2:

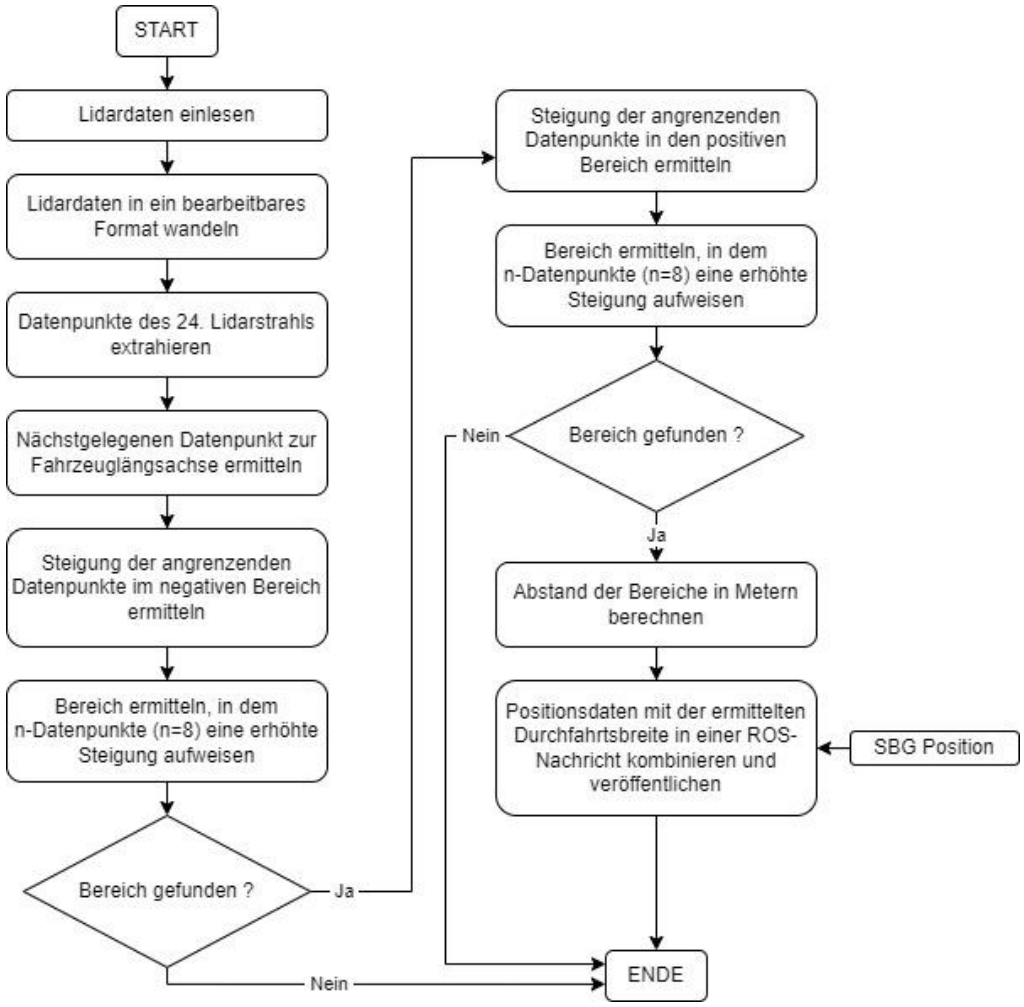


Abbildung 25: Ablauf Durchfahrtbreitenerkennung [2]

Die Abbildung 25 zeigt den Verlauf der Durchfahrtbreitenerkennung Variante 2. Wie in Variante 1 werden die Lidardaten eingelesen, in ein bearbeitbares Array-Format umgewandelt und auf die Daten des 24ten Lidarstrahls reduziert.

Aus den Daten des 24ten Lidarstrahls wird der nächstgelegene Punkt zur Fahrzeuglängsachse ermittelt. Die Fahrzeuglängsachse bildet die Y-Achse. Nach links, in Fahrtrichtung orientiert, befinden sich die negativen und nach rechts die positiven Messwerte der X-Achse. Anschließend wird, beginnend an der Fahrzeuglängsachse, die Steigung aufeinanderfolgender Datenpunkte berechnet. Zuerst wird diese Berechnung im Bereich der negativen X-Werte durchgeführt. Wird ein Steigungswert größer als der empirisch ermittelte Schwellwert (m=1,2) berechnet, wird dieser zwischengespeichert. Liegen die folgenden 8 Werte (n=8) ebenfalls über dem Schwellwert, wird dieser Bereich

als Grenze festgelegt. In dem Fall, dass von dem Programm keine Grenze ermittelt werden kann, beginnt der Programmablauf mit dem nächsten Lidardatensatz erneut.

Wird im negativen Bereich eine Grenze ermittelt, werden die Steigungsberechnungen im positiven Bereich der X-Werte durchgeführt und auf Schwellwertüberschreitung geprüft. Ist diese Überprüfung positiv hat die Software eine weitere Grenze ermittelt. Der Abstand der beiden Grenzwerte zueinander bildet die Durchfahrtbreite. Ist die Überprüfung negativ, beginnt die Software mit dem nächsten Lidardatensatz erneut. Ist ein Durchfahrtbreitenwert berechnet, wird eine ROS-Nachricht mit der aktuellen Position und der Durchfahrtbreite erstellt und veröffentlicht [10.7].

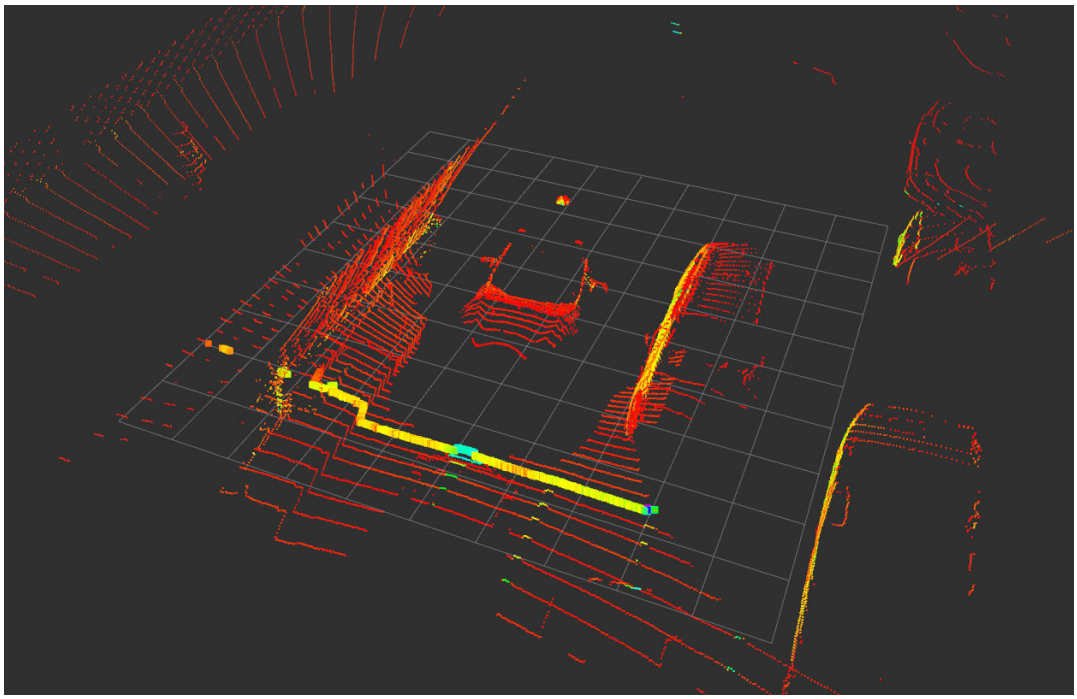


Abbildung 26: Durchfahrtbreiten Grenzenerkennung Lidarbild [2]

In Abbildung 26 sind die gefundenen Grenzen des Algorithmus erkennbar. Auf der linken Bildhälfte ist der Koordinatensprung, der hervorgehobenen Datenpunkte, gut zu erkennen. Der Bordstein wird detektiert und die Koordinaten des ersten Steigungswertes über dem Schwellwert zwischengespeichert. Auf der rechten Bildhälfte ist zu erkennen, dass sich dort kein Bordstein, sondern ein weiteres Kraftfahrzeug befindet. Durch den Anstieg der Messpunktkoordinaten entlang der Karosserie des Kraftfahrzeugs, wird auch hier eine Grenze ermittelt. Der Abstand der beiden Grenzwerte wird als maximale Durchfahrtbreite festgelegt und in ROS mit Positionsdaten veröffentlicht.

Variante 3:

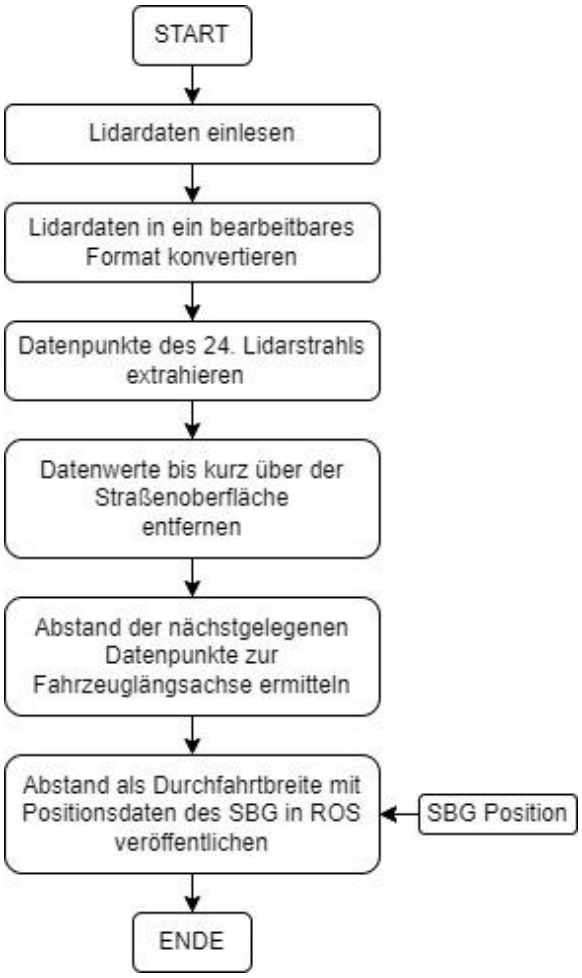


Abbildung 27: Ablauf Durchfahrtsbreitenerkennung ohne Straßenoberfläche [2]

Die Abbildung 27 zeigt den dritten Ansatz der Durchfahrtbreitenerkennung. Dieser beginnt, wie in Variante 1 und 2, mit dem Einlesen und der Aufbereitung der Daten des 24ten Lidarstrahls. Folgend soll die Straßenoberfläche aus den Daten entfernt werden, so dass nur Datenpunkte oberhalb der Straßenoberfläche erhalten bleiben. Maßstab für dieses Vorgehen ist die Vorgabe eines Schwellwertes zwischen 5 cm und 15 cm. Alle Lidardaten unterhalb dieses Wertes werden aus der Punktwolke entfernt. Die Datenpunkte dieser Punktwolke, die am nächsten links und rechts zur Fahrzeuglängsachse liegen, bilden die Durchfahrtbreite ab. Folgend wird der Abstand dieser Datenpunkte berechnet und als Durchfahrtbreite mit den Positionsdaten des Apogee-D in ROS veröffentlicht [10.7].

6.2.3 Test

Für den Test aller drei Varianten wurde, wie in Abbildung 28 ersichtlich, ein Abschnitt des Straßennetzes am Hamburger Hafen ausgewählt, da für diesen Bereich Rosbags zur Verfügung standen.

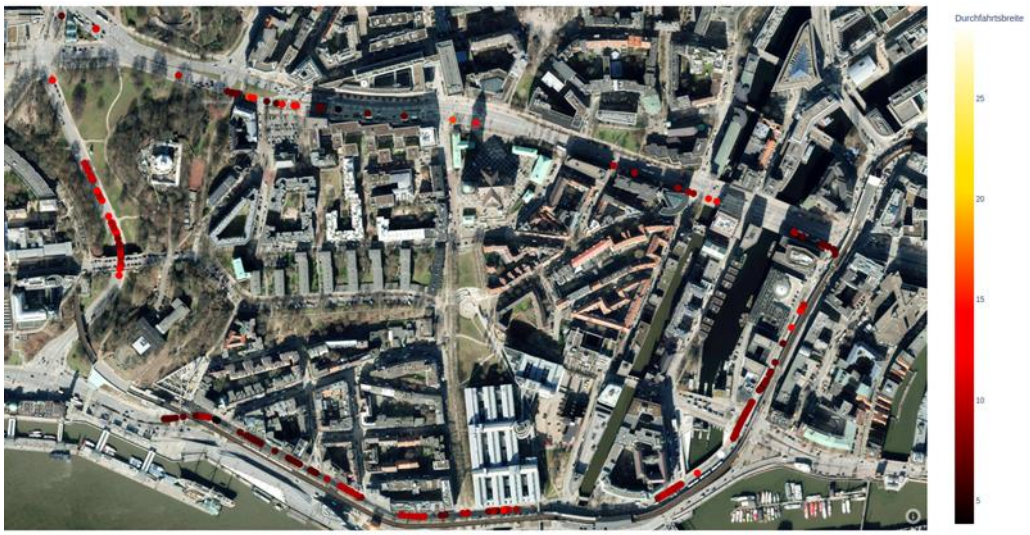


Abbildung 28: Variante 2 Durchfahrtbreite [2]

Die Varianten 1 und 3 liefern keine guten Ergebnisse. Während der Tests ist durch den Abgleich der gemessenen und vorliegenden Durchfahrtbreiten, aufgefallen, dass die ermittelten Werte deutlich von den realen Werten abweichen.

Variante 1 versucht mittels Faltung einen Sprung im Datensatz zu erkennen. Die Problematik dabei ist, dass der Sprung der Bordsteine nicht zwangsläufig der maximale Sprung im Signal ist. Dadurch werden die Grenzstellen falsch identifiziert und folglich falsch vermessen. Die zurückgegebenen Durchfahrtbreiten entsprechen nicht der wirklichen Durchfahrtbreite.

Die Variante 3 soll die Durchfahrtbreite durch das Entfernen der Datenpunkte der Straße berechnen. Anschließend befinden sich nur Datenpunkte im Datensatz, die über einem Schwellwert zur Straße liegen. Problematisch hierbei ist die Unebenheit der Straße und der nichtlineare Verlauf im Querschnitt. Eine Straße wölbt sich häufig mittig, damit Niederschläge seitlich ablaufen können. Diese nicht linearen Querschnitte verhindern die optimale Bestimmung des Schwellwertes. Zu hohe Schwellwerte verursachen das Entfernen benötigter Datenpunkte der Bordsteine. Zu geringe Schwellwerte führen dazu,

dass die Straße nicht, wie beabsichtigt, aus der Punktwolke entfernt wird. Die Software interpretiert folgend die Straße als Bordstein und erzeugt falsche Durchfahrtbreiten.

Beide Softwareansätze führen vermehrt zu falschen Durchfahrtbreiten und sind aufgrund dessen keine geeignete Methodik für deren Ermittlung.

Variante 2 lieferte bei den Tests nachvollziehbare Ergebnisse, weshalb diese näher betrachtet werden. Abbildung 28 zeigt die Positionen, an denen diese Softwarevariante eine Durchfahrtbreite ermitteln konnte.

6.2.4 Bewertung

Aufgrund des Testabbruches der Varianten 1 und 3 ist eine Bewertung dieser nicht möglich.

Variante 2 hat vielversprechende Testresultate erzeugt. Abbildung 29 zeigt ein Histogramm über die erkannten Durchfahrtbreiten der Messstrecke aus Abbildung 28.

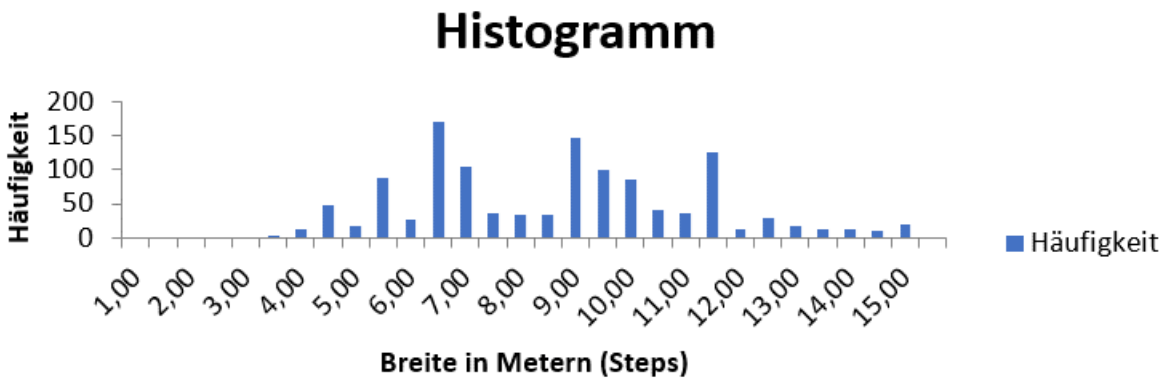


Abbildung 29: Histogramm der gemessenen Durchfahrtsbreite [10.9]

Im Histogramm ist die Anzahl der gemessenen Fahrspuren zu erkennen. Das Histogramm wurde mittels 1347 Durchfahrtbreitenbestimmungen erstellt und in 1-Meter-Schritten aufgeteilt.

Durchschnittlich sind Fahrspuren in Deutschland etwa 2,75 m bis 3,75 m breit [31]. Die Messungen häufen sich in den Bereichen vom Vielfachen dieser Werte. Bei Messwerten von 4,5 m bis 5 m wird eine Fahrspur vermessen. Bei Werten von 6,5 m bis 7 m werden zwei Fahrspuren vermessen und bei 9 m bis 9,5 m drei Fahrspuren. Das lokale Maximum

bei 5,5 m bis 6 m ist auf die zusätzliche Vermessung von abgesetzten Fahrradwegen neben der Straße zurückzuführen. Der Vergleich der Messwerte mit den Kameradaten der Testfahrt unterstreicht die Plausibilität der Messwerte. Die Durchfahrtbreite schwankt durch die Präsenz anderer Verkehrsteilnehmer im Messbereich, welche die Durchfahrtbreite um eine Fahrspur einschränken.

Für Variante 2 gelten zwei Einschränkungen. Zum einen werden die Durchfahrtbreiten nur dann ermittelt und zurückgegeben, wenn beide Grenzen bestimmt werden können. Zum anderen variieren, durch die Ermittlung der Durchfahrtbreite parallel zur Fahrzeugquerachse, die Messergebnisse in Kurvenbereichen stärker.

Der Softwareansatz der Variante 2 erfüllt die Anforderungen der Hamburger Stadtreinigung in Bezug auf die Erkennung der Durchfahrtbreite für die Einschätzung der Passierbarkeit der Wegstrecke. Die Frage, ob und wie die Lidarsensorik zur Ermittlung der maximalen Durchfahrtbreite einer Straße genutzt werden kann, wird durch die Software und die Ergebnisse der Variante 2 beantwortet.

6.3 Mülltonnenerkennung und Lokalisation

Die hier zu implementierende Mülltonnenerkennung soll dem Stakeholder Hamburger Stadtreinigung aufzeigen, dass es möglich ist, verschiedene Objekte aus dem Stadtbild zu erkennen und zu lokalisieren.

In der Entwicklung ist die Frage zu beantworten, wie die Präsenz von Mülltonnen ermittelt werden kann. Die Grundlage dafür bilden die, während der Testfahrt des Prototyp 1 aufgenommenen, Kamerabilder. Als Entwicklungsansatz wurde die Methodik einer Objekterkennung mittels neuronaler Netze ausgewählt.

6.3.1 Vorgehensweise

Das Vorgehen zur Entwicklung dieser Software beruht auf einer experimentellen Untersuchung der Kamerabilder. Zum Training eines neuronalen Netzes muss ein, dem gesuchten Objekt angepasster, Datensatz vorliegen. Die darin enthaltenen Bilder müssen das gesuchte Objekt erkennbar darstellen.

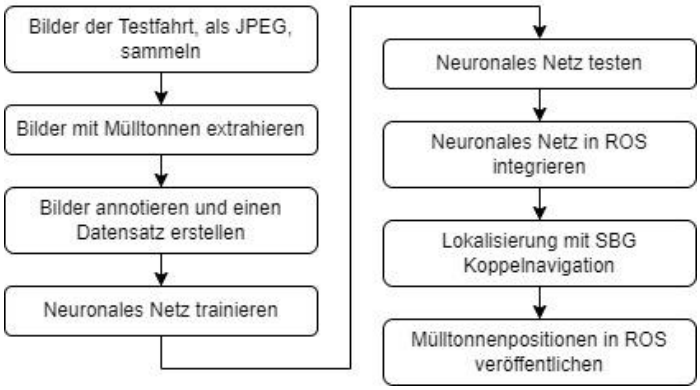


Abbildung 30: Vorgehensweise Mülltonnenerkennung & Lokalisierung [2]

Die Abbildung 30 zeigt den Ablauf der Softwareentwicklung. Aus 40.000 Bildern der Testfahrt werden manuell 7066 Bilder extrahiert, auf denen eine rote Stadtmülltonne zu erkennen ist. Im Anschluss daran werden die Bilder mit dem Online-Tool Roboflow annotiert, indem die Mülltonnen manuell durch Begrenzungsrahmen (Bounding Boxen) markiert werden. Roboflow erzeugt zu jedem Bild eine Textdatei, in welcher sich die Koordinaten der Bounding Boxen befinden. Die Kombination von Bild und Textdatei bildet die Basis für das Training des neuronalen Netztes. [32] Dafür wird die webbasierte

Entwicklungsumgebung Google-Colab verwendet [33]. Die dortige Python-Laufzeitumgebung ist kostenlos und enthält einen Zugang zu Grafikkarten, wodurch die Trainingszeit deutlich reduziert wird. [10.5]

Das trainierte Netz wird im Anschluss getestet und bewertet. Sind die Testergebnisse nicht zufriedenstellend, muss der gesamte Vorgang mit neuen Trainingsdaten so oft wiederholt werden, bis ein zufriedenstellendes Testergebnis vorliegt. Dieses neuronale Netz wird folgend in das ROS-System eingebunden. Anschließend ist für jede ermittelte Stadtmülltonne eine Georeferenz zu bestimmen.

Als Ergebnis dieser Software soll die rote Stadtmülltonne erkannt und georeferenziert kartiert werden.

6.3.2 Umsetzung

Die Software startet mit der ROS-Integration. Dafür werden drei Subscriber erstellt. Jeweils einen für die Apogee-D-Position, die Apogee-D-Orientierung und das Ladybugbild. Der Programmablauf beginnt, sobald sich das Programm im Wartezustand befindet und ein Ladybugbild in ROS veröffentlicht wird.

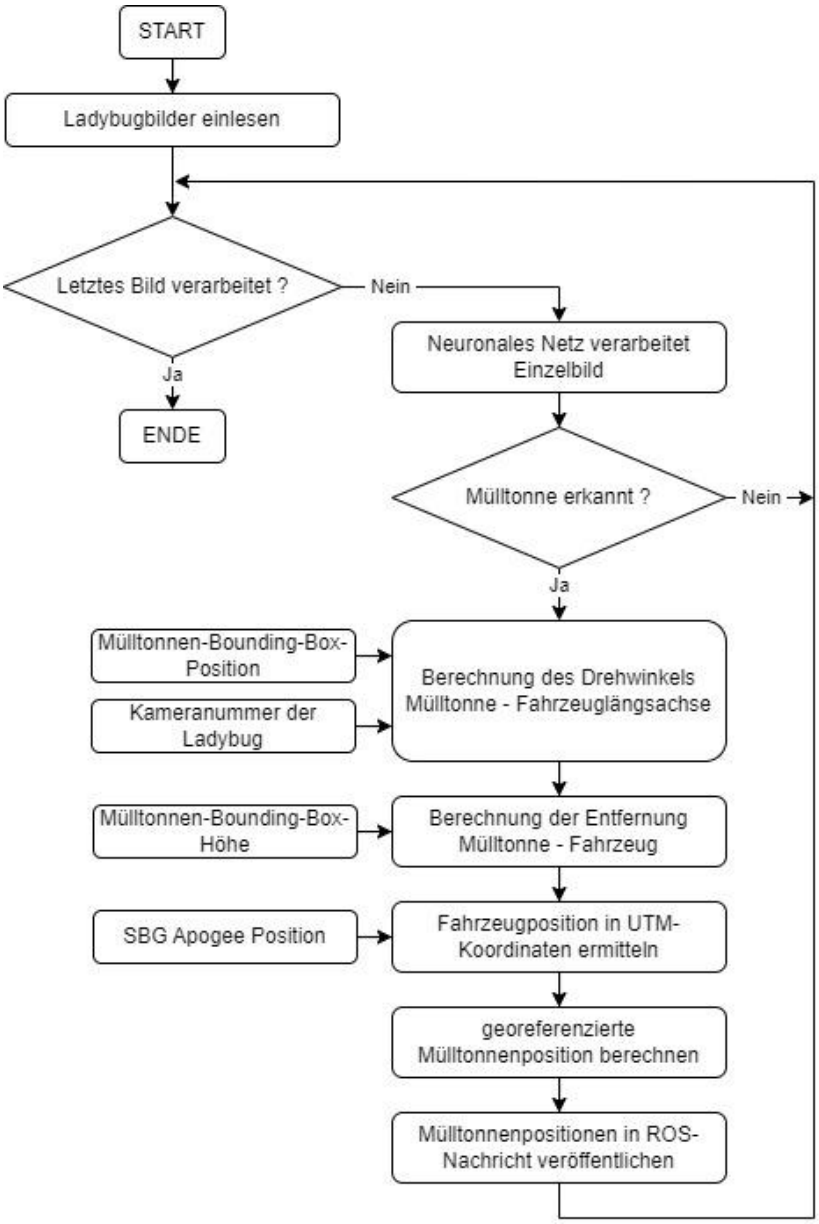


Abbildung 31: Ablauf Mülltonnenerkennung und Lokalisierung [2]

Die Abbildung 31 zeigt den Programmablauf der Mülltonnenerkennung und Lokalisierung. Das Programm beginnt mit dem Importieren der Ladybug 5+ Bilder. Funktionen der Cv-Bridge-Bibliothek wandeln die Bilder von einem ROS- in ein bearbeitbares OpenCV-Format um [34]. Im Anschluss analysiert das neuronale Netz die Einzelbilder. Der Rückgabewert dieser Analyse ist zum einen eine Liste mit gefundenen Mülltonnen, als Bounding Boxen im Bild, zum anderen eine Liste mit der jeweils zugehörigen Confidence. Dieser prozentuale Wert gibt an, mit welcher Wahrscheinlichkeit das Objekt im Bild richtig erkannt ist. Die korrekte Klassifizierung ist ab einem empirisch ermittelten Wert von 82 % gegeben. Wird in einem Bild keine

Mülltonne erkannt oder ist die Confidence kleiner als 82 % startet die Analyse mit einem neuen Bild.

Wird eine Mülltonne erkannt geht der Programmablauf weiter. Für jede detektierte Mülltonne wird die Höhe der Bounding Box geprüft. Der Minimalwert der Höhe ist auf 30 Pixel festgelegt. Geringere Höhenwerte werden als Fehldetektion gekennzeichnet und nicht weiter betrachtet. Enthält das Bild ausschließlich Fehldetektionen, wird mit der Analyse eines neuen Bildes begonnen. Sind die Bedingungen der Mindesthöhe und Confidence erfüllt beginnt der Lokalisierungsprozess.

Durch die Information, welche Ladybug 5+ Kamera das aktuelle Bild aufgenommen hat, sowie Kameraeigenschaften wie den Sichtbereich, ist es möglich, den Winkel der Mülltonne zur Fahrzeuglängsachse zu berechnen. Für die Entfernungsberechnung der Mülltonne zum Fahrzeug wird das Verhältnis der Bounding Box Höhe zu der realen Mülltonnenhöhe verwendet. [30]

Die Fahrzeugposition und -ausrichtung in UTM-Koordinaten wird zusammen mit dem Winkel und der Entfernung der Mülltonne zum Fahrzeug verwendet, um die Position der Mülltonne in UTM-Koordinaten zu berechnen.

Zum Abschluss wird die Mülltonnenposition in WGS84-Koordinaten umgewandelt und als Nachricht in ROS veröffentlicht [10.7]. Dieser Vorgang wiederholt sich für alle fünf horizontal aufgenommenen Einzelbilder der Ladybug5+.

Abbildung 32: Visualisierung der Mülltonnenerkennung [2]

Die Visualisierung in Abbildung 32 zeigt die Rückgabewerte des neuronalen Netzes. Zu erkennen ist die Bounding Box der detektierten Mülltonne, sowie deren Confidence von 99 %.

6.3.3 Test

Folgend wird die Implementierung der Mülltonnenerkennung und Lokalisierung getestet und bewertet.

6.3.3.1 Mülltonnenerkennung

Die Testdaten der Mülltonnenerkennung bestehen aus ca. 17.000 beliebigen Bildern der aufgenommenen Rosbags, welche nicht für den Trainingsprozess genutzt wurden.

Eine geeignete Methode, um neuronale Netze zu testen und zu bewerten, ist eine Konfusionsmatrix. Diese Matrix wird durch den manuellen Vergleich der Objekterkennungsergebnisse mit den Testbildern erzeugt. Tabelle 5 zeigt, dass eine Konfusionsmatrix durch vier verschiedene Faktoren gebildet wird. True Positives sind Objekterkennungsergebnisse, welche der Realität entsprechen und daher positiv sind. False Positives sind Ergebnisse, welche fälschlicherweise als positiv erkannt wurden.

False Negatives sind nicht erkannte gesuchte Objekte. True Negatives sind nicht gesuchte Objekte, welche auch als solche erkannt werden. [35]

	Actual	
Predicted	True Positive (TP)	False Positive (FP)
	False Negativ (FN)	True Negativ (TN)

Tabelle 5: Konfusionsmatrix [35]

Die Tabelle 6 zeigt in der Konfusionsmatrix das Ergebnis der Mülltonnendetektion. Es werden 9115 Mülltonnen korrekt erkannt. 266 Objekte werden fälschlicherweise als Mülltonne erkannt. 37 reale Mülltonnen werden nicht als solche identifiziert und 8000 Bilder enthalten keine Mülltonnen und werden richtig erkannt.

	Actual True	Actual False
Predicted True	TP: 9115	FP: 266
Predicted False	FN: 37	TN: 8000

Tabelle 6: Konfusionsmatrix Mülltonnenerkennung [2]

6.3.3.2 Lokalisierung

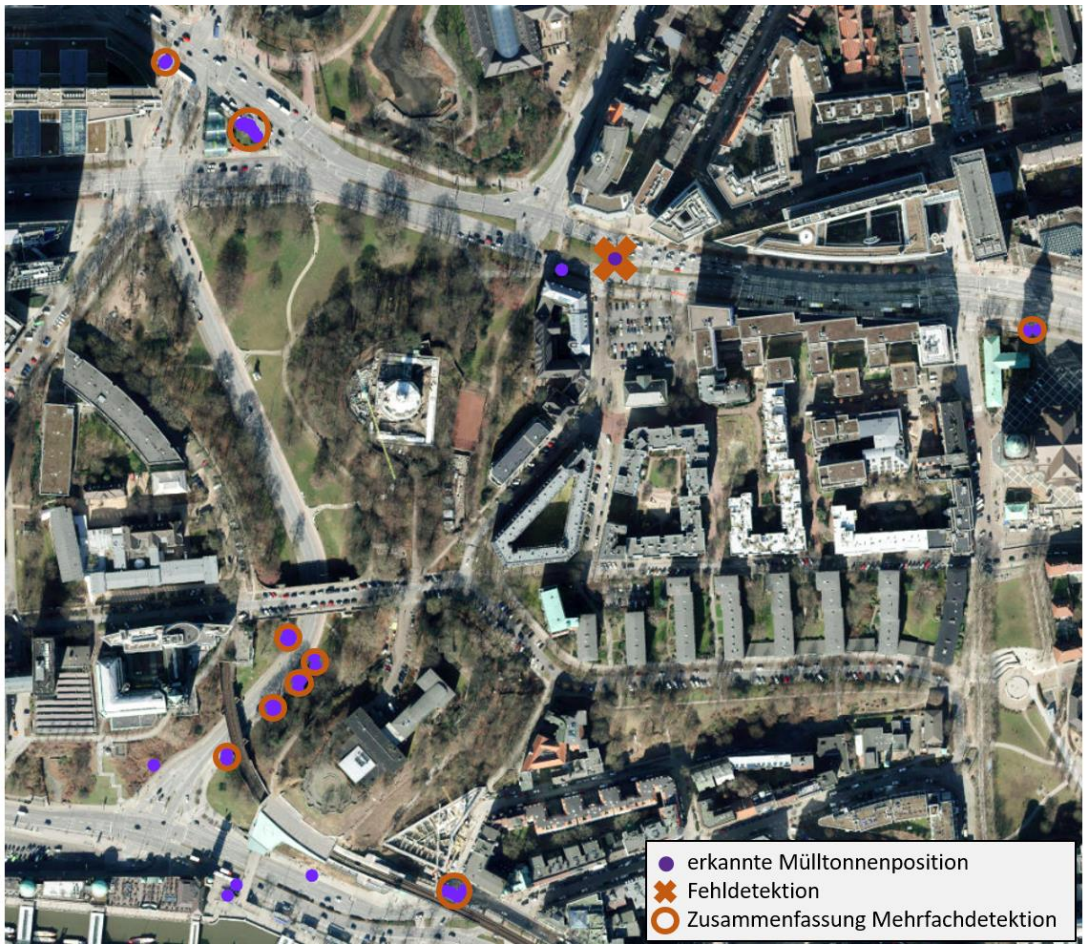


Abbildung 33: Mülltonnenpositionen Hamburg Bismarck-Denkmal [2]

Die Abbildung 33 zeigt die um das Bismarck-Denkmal in Hamburg detektierten Mülltonnen. Die violetten Punkte kennzeichnen die ermittelten Mülltonnenpositionen. Die von dem neuronalen Netz erstellte Detektion einer Mülltonne an einer Position, an der sich keine befindet, wird nachträglich durch ein rotes Kreuz markiert. Durch die Präsenz einer Mülltonne in mehreren Einzelbildern der Ladybug 5+ wird diese mehrfach erkannt und lokalisiert. Der nachträglich eingefügte rote Kreis verdeutlicht die Zusammengehörigkeit von Mehrfachdetektionen einer realen Mülltonne.

6.3.4 Bewertung

Zur Bewertung des neuronalen Netzes werden Kriterien durch die Konfusionsmatrix berechnet.

Die Precision gibt den prozentualen Anteil korrekter Objekterkennungsergebnisse auf alle positiven Erkennungsergebnisse an. [36]

$$Precision = \frac{TP}{TP + FP} = \frac{9115}{9115 + 266} = 0{,}9716 \equiv 97{,}16\,\%$$

Die Accuracy gibt den korrekt klassifizierten Anteil aller wahren Objekterkennungsergebnisse im Verhältnis zu allen Erkennungsergebnisse prozentual an. [36]

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{9115 + 8000}{9115 + 8000 + 266 + 37} = 0{,}9826 \equiv 98{,}26\,\%$$

Der Recall Wert gibt den prozentualen Anteil erkannter Mülltonnen zu allen real vorhandenen Mülltonnen an. [36]

$$Recall = \frac{TP}{TP + FN} = \frac{9115}{9115 + 37} = 0{,}9959 \equiv 99{,}59\,\%$$

Für die Erkennung von Mülltonnen in Bildern ist dieses trainierte neuronale Netz ein geeignetes Mittel. Mehr als 98 % aller Erkennungen des neuronalen Netztes sind korrekt. 99,59 % aller realen Mülltonnen in den Bildern werden erkannt. Die Implementierung der Mülltonnenerkennung ist somit als erfolgreich zu betrachten.

Für die Bewertung der Lokalisierung müssen unter anderem die in Abbildung 33 auftretenden Mehrfachdetektionen betrachtet werden. Die Lokalisierungen dieser Detektionen häufen sich in einem Radius von ca. 2,5 Metern um die reale Mülltonnenposition. Daraus kann ein maximaler Messfehler der Lokalisierung von 2,5 Metern abgeleitet werden. Die Validierung der Einzeldetektionen, mittels eines Kartenabgleichs, ergibt ebenso einen Messfehler von maximal 2,5 Metern.

Mit einer Fehlertoleranz von 2,5 Metern bei der Lokalisierung der Mülltonnen kann die Erkennung als erfolgreich gewertet werden.

6.4 Zusammenfassung der Ergebnisse

Mithilfe der implementierten Verfahren lassen sich die Anforderungen der Stakeholder zufriedenstellend erfüllen. Es wurden zielführende experimentelle Softwareansätze identifiziert und potenzielle Fehlerquellen ergründet. Die erzielten Resultate weisen Fehlerraten auf, die sich innerhalb eines tolerierbaren Rahmens bewegen, wodurch eine erfolgreiche Nutzung der Software möglich ist.

7 Konzeption Prototyp 2

Das folgende Kapitel befasst sich mit der konzeptionellen Auslegung des Prototyp 2. Dieser Prototyp muss zwei Anforderungen erfüllen. Zum einen müssen die Hardwareanforderungen umgesetzt werden, welche sich aus der Softwareentwicklung ergeben. Zum anderen muss ein monetäres Ziel für das Projekt UrbanRFMap eingehalten werden [1].

7.1 Auswahl der Hardware

Die Umfeldvermessung des Trägerfahrzeugs findet weiterhin mittels Lidar- und Kameratechnik statt. Der Lidar muss dafür bestimmte Anforderungen erfüllen. Für die Durchfahrtbreiten- und Gebäudehöhenbestimmung muss dieser verschiedene Bereiche vermessen können. Mindestens ein Lidarstrahl muss direkt vor dem Fahrzeug den Querschnitt der Straße vermessen und für die Gebäudefassaden müssen ganze Punktwolken erzeugt werden. Es ist nicht ausreichend, ausschließlich lineare Strukturen oder einzelne starre Abstände zu vermessen. Der Lidar muss einen großen Messbereich in drei Dimensionen abbilden können.

Abbildung 34: Livox Mid-360 [37]

Als Lidar für Prototyp 2 wird der Mid-360 des Unternehmens Livox ausgewählt. Der in Abbildung 34 gezeigte Mid-360 besitzt einen Sichtbereich von 360° horizontal und -7° bis 52° vertikal. Der Sensor liefert Umfelddaten mit einer Frequenz von 10 Hz und besitzt zusätzlich eine integrierte Inertialsensorik. Durch die kompakte Bauart von 65 mm x 65 mm x 65 mm und einer Reichweite von maximal 70 Metern erfüllt dieser Sensor die Anforderungen. Der Sensor ist mit der Schutzklasse IP76 ausgestattet und daher staubdicht und gegen Wassereindringung geschützt. Mit einer zulässigen

Betriebstemperatur von -20 °C bis 55 °C kann der Sensor in gemäßigten Klimazonen, wie Hamburg, eingesetzt werden. Für die Inbetriebnahme muss der Lidar mit einer Spannung von 9 V bis 27 V versorgt werden. Zur Datenübertragung wird eine RJ45-Ethernet-Schnittstelle verwendet. [38] Mit Anschaffungskosten von ca. 650 Euro befindet sich der Mid-360 innerhalb des Projektbudgets. [37]

Die ausschlaggebende Anforderung an die Kameratechnik ist eine monetäre, da ein Großteil der handelsüblichen Systeme die technischen Anforderungen erfüllen. Eine hohe Anzahl Bilder pro Sekunde ist technisch interessant, jedoch für dieses System nicht notwendig. In urbanen Bereichen mit einer maximalen Trägerfahrzeuggeschwindigkeit von 70 km/h, legt das Trägerfahrzeug bei 10 Hz Videoaufnahmen, zwischen den Bildern maximal 1,94 m Strecke zurück. Dies ist ausreichend, um einen genügenden Informationsgehalt einer gewissen Position zu erhalten. Um die Datenströme von Lidar und Kamera zu synchronisieren, bietet es sich an, dass die Kamerafrequenz mindestens der des Lidar entspricht.

Abbildung 35: MER2-160-227U3C-L Kamera [39]

Für den zu konzipierenden Prototyp 2 wurde eine Kamera mit der Typenbezeichnung MER2-160-227U3C-L ausgewählt. Diese ist in Abbildung 35 zu sehen. Der Hersteller dieser Kamera ist das Unternehmen Daheng Imaging. Mit Anschaffungskosten von 228 Euro wird der monetäre Rahmen des Projektes eingehalten und mit 227 Bildern pro Sekunde auch die Anforderung an die Bildaufnahmefrequenz. Der Bildaufnahmesensor ermöglicht Farbbilder mit einer Auflösung von 1440 x 1080 Pixeln. Die Kamera besitzt einen Global-Shutter für die Erzeugung von Farbbildern. [14] Mit einer Bauformgröße von 29 mm x 29 mm x 45,5 mm ist diese Kamera gut geeignet, um in ein kompaktes Messsystem integriert zu werden. [39]

Für die Mobilfunkvermessung wird die Messtechnik des Projektpartners ENQT in das System eingebunden. Essenzielle Komponenten sind Mobilfunkantennen sowie Modems. Die Mobilfunkantennen werde mit den Modems verbunden. Diese führen mit entsprechender Software die Mobilfunkqualitätsprüfung durch. Pro Netzanbieter wird ein Modem mit jeweils einer, dem Mobilfunknetzanbieter zugehörigen SIM-Karte (Subscriber Identity Module) benötigt. Die benötigten Komponenten liegt bereits in Form des in Abbildung 36 ersichtlichen AMS der Firma ENQT vor. [9]

Abbildung 36: ENQT GmbH - AMS [9]

Durch Bluetooth Low Energy (BLE) können die gemessenen Mobilfunkdaten an die Auswerteelektronik des Prototyp 2 übermittelt werden. [9]

Für die Positionsbestimmung des Prototyp 2 ist ein GNSS-Modul notwendig. Für die Positionsbestimmung von Objekten im Trägerfahrzeugumfeld, ist die Orientierung des Prototyp 2 eine notwendige Information. Dafür ist ein zweites GNSS-Module erforderlich. Die Orientierung kann durch die Kombination beider Positionssignale ermittelt werden. Im AMS-System ist ein GNSS-Modul integriert. Durch die Verwendung mehrerer AMS-Systeme kann die Positions- und Orientierungsbestimmung durchgeführt werden.

Für die zentrale Erfassung der Sensordaten, soll ein „Next Unit of Computing“ (NUC) verwendet werden. NUC´s zeichnen sich durch ihre Leistungsfähigkeit und kompakte Bauform aus. [40] Der NUC benötigt mindestens eine RJ45-Ethernet-Schnittstelle und eine noch zu definierende Anzahl an USB-Schnittstellen. Für die Anbindung der Mobilfunkmesstechnik ist eine BLE-Schnittstelle erforderlich. Weitere Anforderungen an

den NUC müssen noch spezifiziert werden. Aufgrund dessen wurde noch kein bestimmtes Modell ausgewählt.

7.2 Orientierung der Sensorik

Die Orientierung der Messtechnik innerhalb des Prototyp 2 nimmt eine relevante Rolle in der Hardwarekonzeptionierung ein. Für die optimale Nutzung der Sensorik muss diese entsprechend ausgerichtet sein.

Die Positionierung des Lidar in einem 45° Winkel in Fahrtrichtung erfüllt die Anforderung an die zu messenden Größen Straßenoberfläche und Gebäudefronten. Die Positionierung der Kamera in Fahrtrichtung hat den Vorteil, dass Informationen aus dem Straßenverkehr, sowie Mülltonnen in Straßennähe, aufgezeichnet werden. Durch die Positionierung der einzelnen GNSS-Module kann die Orientierung des Fahrzeugs ermittelt werden. Um die Orientierungsabweichung so gering wie möglich zu halten, ist es von Vorteil, die zweite GNSS-Empfangsantenne so weit wie möglich auf der Fahrzeuglängsachse, hinter der ersten Antenne, zu positionieren. Die Mobilfunkantennen werden oben auf dem Prototyp 2 montiert, um die Sende- und Empfangsqualität nicht durch den eigenen Messaufbau zu beeinflussen.

7.3 Hardwarekonzept

Das folgende Kapitel zeigt eine konzeptionelle Ausarbeitung der Hardware für den Prototyp 2.

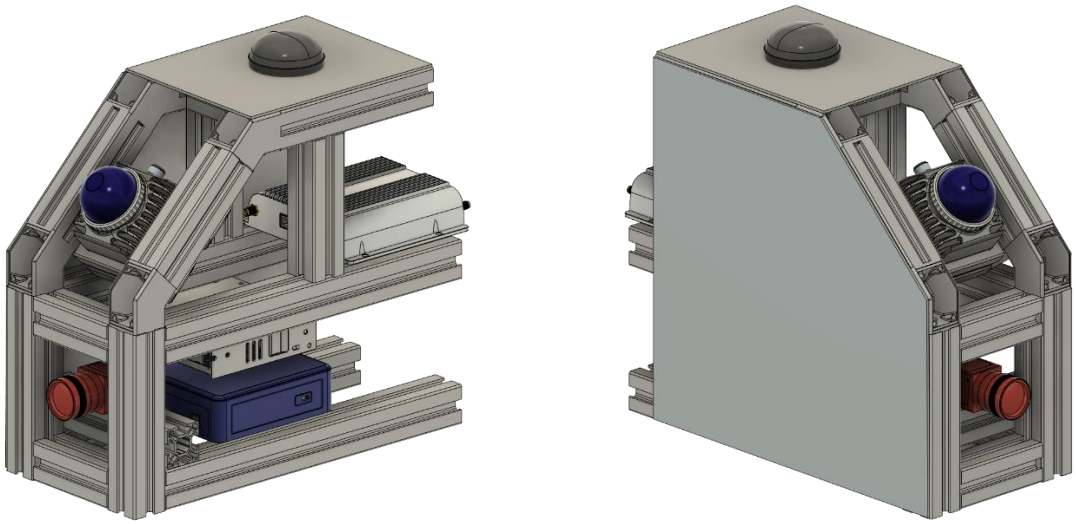


Abbildung 37: Hardwarekonzept Prototyp 2 [2]

Die Abbildung 37 zeigt den konzeptionellen Aufbau des Prototyp 2. Die Rahmenstruktur besteht aus Aluminiumprofilen. Damit ist es möglich, den Bauraum so kompakt wie möglich zu gestalten und an die Hardware anzupassen. An der 45° Kante zur Fahrzeuglängsachse ist der Livox Mid-360 montiert. Ebenfalls in Fahrzeuglängsachse ist die Kamera montiert, hier in Rot dargestellt. Aufgrund der nichtspezifizierten Schutzklasse wird die Optik der Kamera durch eine aufgesetzte Linse vor Umwelteinflüssen geschützt.

Der NUC wird im unteren Bereich montiert, hier in Blau dargestellt. Oberhalb dessen befindet sich die Spannungsversorgung und darüber das Mobilfunkmodem. Dieses wurde exemplarisch als Platzhalter für die Modems von ENQT platziert. Die Mobilfunkantenne, auf dem Aufbau in Schwarz dargestellt, ist ein Platzhalter für die Mobilfunkantennen und GNSS-Empfänger.

Der Prototyp 2 ist ca. 14 cm breit und 27 cm hoch. Die endgültige Höhe und Tiefe kann noch nicht bestimmt werden, weil die Maße einzelner Komponenten noch nicht bekannt sind. Die Höhe kann durch eine geschickte Optimierung der Bauweise auf schätzungsweise 20 cm reduziert werden.

7.4 Konzeptioneller Verdrahtungsplan

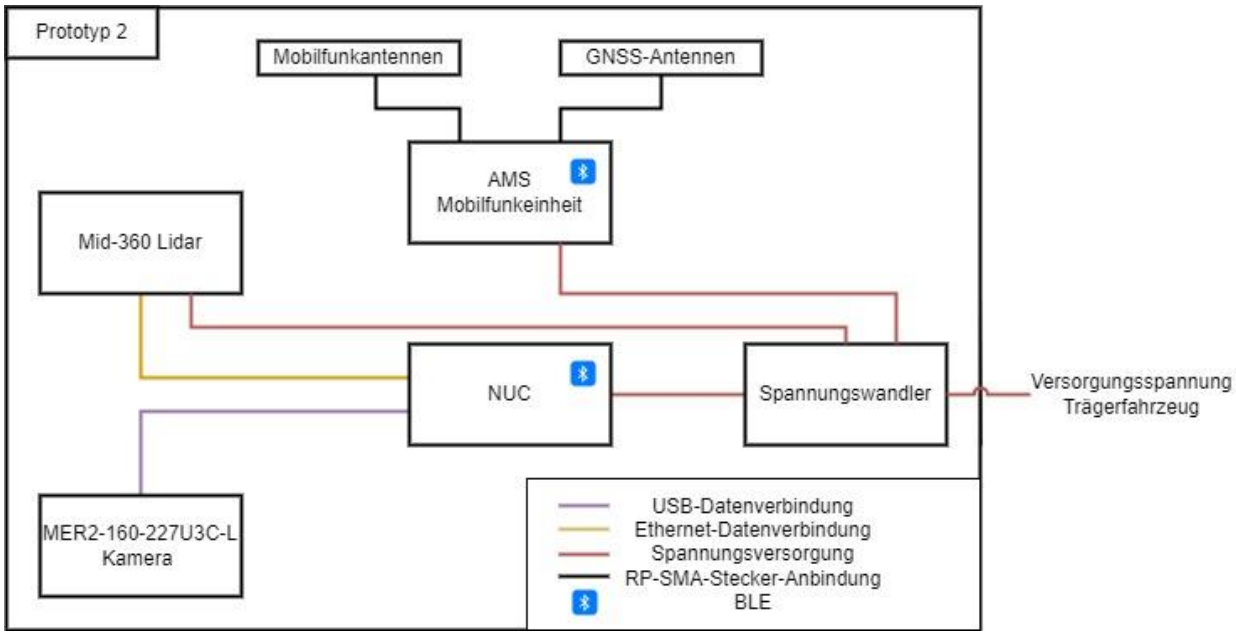


Abbildung 38: konzeptioneller Verdrahtungsplan [2]

Die Abbildung 38 zeigt den konzeptionellen Verdrahtungsplan des Prototyp 2. Die Versorgungsspannung des Systems wird von dem Trägerfahrzeug bereitgestellt und in einer internen Einheit in die benötigten Spannungen umgewandelt. Mit dieser sind der NUC, das AMS-System und der Lidar verbunden. Die Kamera ist zur Datenübertragung und Spannungsversorgung mittels USB-Verbindung mit dem NUC verbunden. Der Livox Lidar ist mit einer RJ45-Ethernet-Schnittstelle mit dem NUC verbunden und übermittelt auf dieser die aufgezeichneten Punktwolken. Die Mobilfunkmesstechnik verwendet BLE für die Datenübertragung an den NUC.

8 Fazit und Ausblick

Die Zielsetzung dieser Arbeit ist die konzeptionelle Erstellung des Prototyp 2 zur Messung von Mobilfunk- und Umfelddaten. Hierfür wurden Anforderungen der eingebundenen Stakeholder ausgearbeitet, die aus Informations- sowie hardwaretechnischen Anforderungen bestehen. Anschließend wurde der derzeitige technische Stand der Umfeld- und Mobilfunktechnik beschrieben. Die durch Prototyp 1 erfassten Daten dienten als Grundlage für die Entwicklung der Software. In dieser wurden die Informationsanforderungen der Stakeholder, bestehend aus Durchfahrtbreitenerkennung, Mülltonnenerkennung und Lokalisierung und Gebäudehöhenerkennung umgesetzt. Durch umfassende Tests der Softwareergebnisse konnte ihre erfolgreiche Umsetzung bestätigt werden.

Die getestete Durchfahrtbreitenerkennung ist eine solide Grundlage für die Bestimmung der Durchfahrtbreite. Um eine Aussage darüber zu treffen, ob ein Fahrzeug oder ein Bordstein die aktuelle Grenze darstellt, kann die Software zukünftig optimiert werden. Dafür ist die Höhenbestimmung der ermittelten Grenzen notwendig. Damit kann nicht nur die Durchfahrtbreite, sondern auch die gesamte Straßenbreite eindeutig ermittelt werden.

Die Ergebnisse der Mülltonnenerkennung und Lokalisierung zeigen, dass es möglich ist mit einer Kamera Objekte zu erkennen und zu lokalisieren. Diese Technik kann auf verschiedene Objekte angewendet werden. Dazu ist ein neuronales Netz notwendig, welches die Objekte erkennt. Für die Lokalisierung erkannter Objekte müssen die realen Maße bekannt sein. Bei Mehrfachdetektionen ist es möglich die beste Positionsvorhersage, durch Nachbearbeitung der Ergebnisse, herauszufiltern oder zu berechnen. Effektive Mittel hierfür sind eine nearest-neigbour-analyse oder eine Mittelwertbildung in UTM-Koordinaten.

Die entwickelte Software führte zu hardwaretechnischen Anforderungen an die Entwicklung des Prototyp 2. Diese Anforderungen wurden ausgearbeitet und in der Konzeption des Prototyp 2 berücksichtigt. Aufgrund der modularen Auslegung des Prototyp 2 kann dieser zukünftig um weitere Sensoren ergänzt werden. Beispiele hierfür sind Sensoren zur Messung der Luftgüte. Dies dient dem Zweck das digitale Abbild des Umfeldes zu erweitern.

Für die Nutzung des Prototyp 2 in der Praxis sind folgende Schritte notwendig. Die fehlenden Komponenten für die Umsetzung des Prototyp 2 sind zu ermitteln. Anschließend muss die Hardware gebaut und die Sensoren mit dem NUC in Betrieb genommen werden. Die Implementierung der NUC-Software zur Datenspeicherung ist erforderlich, um Messfahrten durchführen zu können. Die Ergebnisse dieser Fahrten bilden die Grundlage für die Bewertung des Prototyp 2

9 Literatur

[1] Christopher E. Niemöller, Rasmus Rettig, „Projektantrag UrbanRFMap,“ Hochschule für Angewandte Wissenschaften Hamburg, ENQT, Okt. 2022.

[2] *Erstellt von Alexander Günther*, 2023. Zugriff am: 19. September 2023.

[3] FTS Hennig GmbH. „LTE Signalstärke dbm | LTE Leistung verbessern mit FTS Hennig: Was sind RSRP, RSRQ, RSSI und SINR?” https://www.fts-hennig.de/antennen/blog/lte-leistung/ (Zugriff am: 7. September 2023).

[4] Sebastian Schöne, Maik Enrico Wildemann. „RSRP | Interpretation und Bestimmung.” https://www.lte-anbieter.info/technik/rsrp.php (Zugriff am: 14. September 2023).

[5] Sebastian Schöne und Maik Enrico Wildemann. „RSRQ | Bestimmung und Bedeutung.” https://www.lte-anbieter.info/technik/rsrq.php (Zugriff am: 14. September 2023).

[6] M. Kratzenberg, „RSSI: Welcher Wert ist gut?,“ *Giga*, 16. Februar 2017. https://www.giga.de/ratgeber/tipps/rssi-welcher-wert-ist-gut/ (Zugriff am: 14. September 2023).

[7] D. Morelo, „Was RSSI und seine Beziehung zu einem WiFi-Netzwerk ist,“ *Netspotapp software*, 12. September 2023. https://www.netspotapp.com/de/wifi-signal-strength/what-is-rssi-level.html#:~:text=RSSI%2DPegel%20und%20Signalst%C3%A4rke,-RSSI%2DMessungen%20repr%C3%A4sentieren&text=Je%20h%C3%B6her%20der%20RSSI%2DWert,100%20w%C3%A4re%20%C3%BCberhaupt%20kein%20Signal. (Zugriff am: 14. September 2023).

[8] Sebastian Schöne und Maik Enrico Wildemann. „SINR und SNR | Relevanz für Mobilfunknutzer.” https://www.lte-anbieter.info/technik/sinr.php (Zugriff am: 14. September 2023).

[9] „SignalInsight AMS 2G/3G/4G/5G.” https://enqt.de/signalinsight/ams-2g-4g-5g/ (Zugriff am: 14. September 2023).

[10] „Umfelderfassung/Umfeldsensorik | Mein Autolexikon.” https://www.mein-autolexikon.de/fahrerassistenzsysteme/umfelderfassungumfeldsensorik.html (Zugriff am: 9. September 2023).

[11] I. Velodyne Lidar, „HDL-32E: HIGH RESOLUTION REAL-TIME 3D LIDAR SENSOR,” [Online]. Verfügbar unter: https://velodynelidar.com/wp-content/uploads/2019/12/97-0038-Rev-N-97-0038-DATASHEETWEBHDL32E_Web.pdf

[12] M. Large, „Lidar: Was es ist, wie es funktioniert und was es kann,“ *all-electronics.de*, 06. Juni 2017. https://www.all-electronics.de/automotive-transportation/lidar-sensoren-automotive-575.html#:~:text=auf%20Lidar%2DSensoren%3F-,Was%20ist%20Lidar%3F,Bilder%20der%20erkannten%20Objekte%20liefern. (Zugriff am: 14. September 2023).

[13] U. M. Goran Pandza, „Nanosekunden-Laserpulse für ToF und mehr,“ *all-electronics.de*, 12. Oktober 2021. https://www.all-electronics.de/elektronik-entwicklung/nanosekunden-laserpulse-fuer-tof-und-mehr-128.html (Zugriff am: 14. September 2023).

[14] GeT Cameras, industriekameras und objektive. „Rolling versus Global shutter.” https://www.machinevisionkamera.de/FAQ-ROLLING-VS-GLOBAL-SHUTTER (Zugriff am: 14. September 2023).

[15] „Industrielle Kameras, ihr Markt und ihre technischen Merkmale | wileyindustrynews.com.” https://www.wileyindustrynews.com/news/industrielle-kameras-ihr-markt-und-ihre-technischen-merkmale-2 (Zugriff am: 14. September 2023).

[16] Teledyne FLIR. „Ladybug5+ | Teledyne FLIR.” https://www.flir.de/products/ladybug5plus/?vertical=machine+vision&segment=iis (Zugriff am: 14. September 2023).

[17] AZ-Delivery. „DHT11 Temperatursensor und Luftfeuchtigkeitssensor kompatibel mit Arduino und Raspberry Pi.” https://www.az-delivery.de/products/5-x-dht11-temperatursensor?variant=5559032250395&utm_source=google&utm_medium=cpc&utm_campaign=19229855661&utm_content=147170319769&utm_term=&gclid=Cj0KCQjwmICoBhDxARIsABXkXlLEgdPDPe9Lft0xvT3OhGqsQPezrdL5iSV1PVaNvemWVQzRzpqwT0saAhFtEALw_wcB (Zugriff am: 14. September 2023).

[18] reichelt elektronik GmbH. „DEBO SENS CCS811 - Entwicklerboards - Sensor für Luftqualität, CCS811.” https://www.reichelt.de/de/de/entwicklerboards-sensor-fuer-luftqualitaet-ccs811-debo-sens-ccs811-p253655.html?r=1&gclid=Cj0KCQjwmICoBhDxARIsABXkXlKVnhxwIch3-9Od_CZ7OHwiX-ZPoREKfER_LpdqBq93XigeIsH2_3kaAt8KEALw_wcB (Zugriff am: 14. September 2023).

[19] „Nova Fitness SDS011 Feinstaub Sensor – MAKERSHOP.DE.” https://www.makershop.de/sensoren/sds011-feinstaub-sensor/ (Zugriff am: 14. September 2023).

[20] Umweltbundesamt. „Flüchtige organische Verbindungen.” https://www.umweltbundesamt.de/themen/gesundheit/umwelteinfluesse-auf-den-

menschen/chemische-stoffe/fluechtige-organische-verbindungen#was-sind-die-quellen-fur-voc (Zugriff am: 14. September 2023).

[21] W. Ewald, „TVOC und eCO2 Sensoren,“ *Wolfgang Ewald*, 02. Juli 2021. https://wolles-elektronikkiste.de/tvoc-und-eco2-sensoren (Zugriff am: 14. September 2023).

[22] PAMAS. „Die Methoden der optischen Partikelmessung – Lichtblockade- und Streulichtverfahren.” https://blog.pamas.de/die-methoden-der-optischen-partikelmessung-lichtblockade-und-streulichtverfahren/ (Zugriff am: 14. September 2023).

[23] SBG Systems. „Apogee Series - Hochpräzise Trägheitsnavigationssensoren: Apogee-D All-in-one INS/GNSS.” https://www.sbg-systems.com/de/produkte/apogee-series-hohe-genauigkeit-ins-gnss/#apogee-d_high_performance_ins-gnss (Zugriff am: 17. September 2023).

[24] Wikipedia. „World Geodetic System 1984.” https://de.wikipedia.org/w/index.php?title=World_Geodetic_System_1984&oldid=229816560 (Zugriff am: 10. September 2023).

[25] Wikipedia. „Geographische Koordinaten.” https://de.wikipedia.org/w/index.php?title=Geographische_Koordinaten&oldid=237376253 (Zugriff am: 17. September 2023).

[26] Wikipedia. „UTM-Koordinatensystem.” https://de.wikipedia.org/w/index.php?title=UTM-Koordinatensystem&oldid=234006636 (Zugriff am: 17. September 2023).

[27] Bayerische Vermessungsverwaltung- Landesamt für Digitalisierung, Breitband und Vermessung, „UTM-Abbildung-und-Koordinaten,” [Online]. Verfügbar unter: https://www.ldbv.bayern.de/file/pdf/1910/UTM-Abbildung-und-Koordinaten.pdf

[28] Osram. „Schutzarten (IP-Codes) nach DIN EN 60529 und äußere Umwelteinflüsse.” file://C:/Users/aligu/Downloads/technischer-anwendungsleitfaden---schutzarten-ip-codes-nach-din-en-60529-d.pdf (Zugriff am: 14. September 2023).

[29] MIURA Yasuyuki. „Distributions - ROS.” http://wiki.ros.org/Distributions (Zugriff am: 29. März 2023).

[30] Y. Liu, *Positionserkennung mit einer sphärischen Kamera für das autonome Fahren* (Masterthesis), 2020. [Online]. Verfügbar unter: https://reposit.haw-hamburg.de/cris/explore/theses

[31] Wikipedia. „Richtlinien für die Anlage von Straßen – Querschnitt.” https://de.wikipedia.org/w/index.php?title=Richtlinien_für_die_Anlage_von_Straßen_–_Querschnitt&oldid=231778529 (Zugriff am: 17. September 2023).

[32] Roboflow. „Roboflow Universe: Open Source Computer Vision Community.” https://universe.roboflow.com/ (Zugriff am: 17. September 2023).

[33] „Google Colaboratory.” https://colab.research.google.com/?hl=de (Zugriff am: 17. September 2023).

[34] „Converting between ROS images and OpenCV images (Python).” http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython (Zugriff am: 17. September 2023).

[35] Schmetterlingsgedanke, „Was ist eine Konfusionsmatrix beim maschinellen Lernen?,“ *Geekflare*, 27. Januar 2023. https://geekflare.com/de/confusion-matrix-in-machine-learning/ (Zugriff am: 29. März 2023).

[36] A. Oppermann. „Accuracy, Precision, Recall, F1-Score und Specificity.” https://artemoppermann.com/de/accuracy-precision-recall-f1-score-und-specificity/ (Zugriff am: 29. März 2023).

[37] Livox. „Livox Mid-360.” https://www.livoxtech.com/de/mid-360 (Zugriff am: 14. September 2023).

[38] Livox. „Specs - Mid-360 LiDAR Sensor - Livox.” https://www.livoxtech.com/de/mid-360/specs (Zugriff am: 14. September 2023).

[39] China Daheng Group. „MER2-160-227U3C-L - MER2-U3-L Series - DAHENG IMAGING—Dedicated to Machine Vision.” https://en.daheng-imaging.com/show-107-2031-1.html (Zugriff am: 14. September 2023).

[40] Intel. „Was ist ein NUC?” https://www.intel.de/content/www/de/de/products/docs/boards-kits/nuc/what-is-nuc-article.html (Zugriff am: 17. September 2023).

10 Anhang

10.1 Anforderungsentwicklung mit ENQT

HAW HAMBURG

UrbanRFMap Anforderungsentwicklung

Datum: 09.03.2023 , 12.30 – 15.00 Uhr

Teilnehmer: ENQT, HAW-Hamburg

Version: 0.1 Autor: Alexander Günther

(Bei weiterer Bearbeitung bitte die Versionsnummer fortsetzen und den Autor ergänzen)

Ablauf:

1. Vorstellung der Teilnehmer
2. Formulierung des Projektzieles
3. Identifikation möglicher Stakeholder
4. Stakeholder-Analyse anhand ermittelter Stakeholder
5. Entwurf einer möglichen Implementierung

2. Ziel des Projektes

- „Rohde & Schwarz zu Aldi Preisen“ (HAW)
- Robustes System : Vibrationsfest, Temperaturfest, Witterungsbedingungen, Wasserdicht (ENQT / HAW)
- Autarkes System, keine vor Ort Bedienung durch Personal, lediglich eine Spannungsversorgung vom Fahrzeug (ENQT / HAW)
- Ausfallsicher, Schutzmaßnahmen gegen äußere Einflüsse, vorbestimmte Maßnahmen in Fehlerfällen (ENQT)
- Laufzeit von 45000 Stunden -> Abdecken des Lebenszyklus eines Fahrzeugs (ENQT)
- Modularer erweiterbarer Hardwareaufbau (ENQT / HAW)
- Software aus der Ferne konfigurierbar, Remote-updates (ENQT)
- Kommunikationssicherheit (ENQT / HAW)
- Cybersicherheit (ENQT / HAW)
- Möglichst wartungsfrei (ENQT)
- Komplett selbstüberwacht, vorprogrammiertes Verhalten im Fehlerfall (ENQT)
- Kostenziel 1000 Euro für ein Serienprodukt von 500 – 1000 Stück (ENQT)
- Aufnahme von (ENQT)
  - Mobilfunkdaten
  - 3D-Lidardaten
  - Kameradaten
  - Position (GPS / GNSS)
  - Subfunknetze mit Frequenzen < 1GHz („Subs“)
  - WLAN 802.11p
- Weiter Funktionen:
  - WLAN
  - Bluetooth
  - Heizen / Kühlen spezifischer Komponenten
  - CAN-Bus Daten mit auslesen
  - Automatisierte Auswertungen
- Datenschutzkonformität (HAW)
- Zulassungsfähigkeit (E-Prüfzeichen der verwendeten Bauteile)
- EMV (Elektromagnetische Verträglichkeit)

- Einbettung in die Infrastruktur von ENQT (ENQT)
- Diebstahl-Sicherheit, geeignete Befestigungsmöglichkeit am Fahrzeug (HAW)

3. Mögliche Stakeholder:

Allgemeine Stakeholder:

- Stadtreinigung
- Mobilfunknetzbetreiber
- Energieversorger
- Bundesnetzagentur

Stakeholder in Hamburg und Umgebung:

- Stadtreinigung Hamburg
- Bukea (Behörde für Umwelt, Klima, Energie und Agrarwirtschaft)
- Vay
- HPA

Projektpartner:

- ENQT
- HAW-Hamburg

Weitere:

- Fahrer des betreffenden Fahrzeugs, auf welchem die Messtechnik montiert ist.

HAW HAMBURG

4. Stakeholder-Analyse: Stromnetz Hamburg

Use-Case 1: Smart-Meter-Rollout

- Mobilfunkempfang an Installationslokationen
  - Keller / Wohngebäude / Ampelanlagen / Litfaßsäulen / Mobilfunk / Ladestationen
- „Überall wo der entsprechende Netzbetreiber Zähler verbaut hat"-Christopher Niemöller
- Daten im Sekundenbereich bereitstellen (normalerweise 15 minütig) (Gateways)
- Relevante Messdaten
  - RSRP
  - Datenrate
  - Latenz
  - 3D Lidardaten
  - Kamerabilder
  - GPS

Use-Case 2: Errichtung von E-Ladestationen

- Während der Datenaufnahme, Bilddaten aufnehmen um Ladestationen zu planen
- Planungskosten reduzieren, indem die Mitarbeiter am Computer die Daten vorort betrachten können -> Mehrwert für das Unternehmen
- Korrelation mit externen Daten
- Ist genügend Platz für eine lokale Trafo-Station / Parkplatzsituation mit aufnehmen
- Relevante Messdaten
  - RSRP
  - Datenrate
  - Latenz
  - 3D Lidardaten
  - Kamerabilder
  - GPS

Geschäftsmodell

- ENQT stellt Daten für Stromnetz-Hamburg bereit (ENQT API)
- Stromnetz-Hamburg als Kunde, bezahlt als Endkunde für die Daten
- ENQT also Provider der Daten

Weiteres

- momentan gibt es schon Mobilfunkdaten mit Positionsdaten. Diese durch bildgebende Daten erweitern, um einen Eindruck von der Lokation zu bekommen (Klassifikation von Gebäudehöhen, etc.)

HAW HAMBURG

Stakeholder: Mobilfunknetzbetreiber

Use-Case 1: Das eigene Netz optimieren

- Mobilfunkmessdaten / GPS
- Einschätzung der Netzabdeckung und Qualität des Mobilfunknetzes / Hand-over / verifizierte Daten
- Kostentechnische Optimierung
  - Simulationen für die Netzausbreitung anhand von realen 3D-Daten
  - 5G / 6G Ausbau
  - Gebäude vereinfacht darstellen
- Problematik
  - hohe Frequenzen und hohe Gebäude
  - Gebäude reflektieren den Mobilfunk
  - „Mobilfunk verhält sich in diesen Frequenzen wie Licht. Somit werfen Gebäude einen Schatten"-Christopher Niemöller
- Relevante Messdaten
  - GPS
  - RSRP
  - Datenrate
  - Latenz
  - 3D Lidardaten
  - Kamerabilder

Stakeholder: Bundesnetzagentur

Use-Case 1: Überwachung des Mobilfunk-Marktes

- Werden Ausbauvorgaben erfüllt?
- Werden Frequenznutzungspläne eingehalten?
  - Einhalten von Mindestbandbreiten
- Relevante Messdaten
  - GPS
  - RSRP
  - Datenrate
  - Latenz

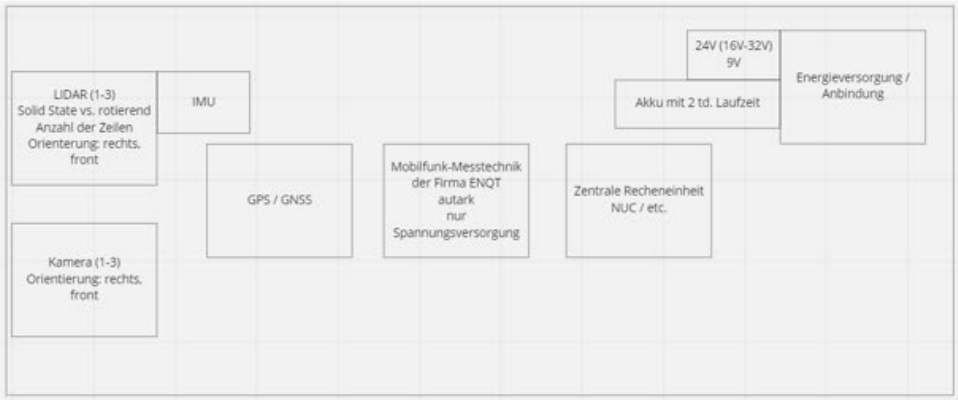
Use-Case 2: EMF-Überwachung

- Gesundheitsschutz (Grenzwerte an Basisstationen)
- Werden max. Leistungen von Basisstationen eingehalten
- Mit bildgebender Sensorik verbunden
- Relevante Messdaten
  - GPS
  - Kamerabilder
  - Leistung (Sendeleistung) einer Basisstation

HAW HAMBURG

5. Entwurf einer Implementierung

Stromnetz-Hamburg Use-Case 2 / Errichtung von E-Ladestationen	
Betreiber	(Stadtreinigung) / ENQT
Kunde	Stromnetz-Hamburg
Welcher Mehrwert entsteht ?	Rein digitale Planung von E-Ladestationen anhand von aktuellen 3D- und Kameradaten der Stadt
Welche Daten ?	• 3D Lidardaten • Fotos für jede Straße in Hamburg zur Planung von E-Ladestationen (kolorierte Punktwolke) • GPS • Mobilfunkmesstechnik
Welche Sensoren ?	Kamera / Lidar / Mobilfunk / GPS (GNSS) /
Welche Anforderungen ?	• Positionsgenauigkeit von 0.5m • Punktauflösung des LIDAR 1cm • Genauigkeit +/- 5 cm
Beschreibung des Ablaufs	Erstmal nur Daten über einen längeren Zeitraum aufnehmen und Speichern. Folgend eine Entwicklung von Software um Parkplätze und Hindernisse zu erkennen. Digitaler Zugang der Mitarbeiter (Stromnetz-Hamburg) auf die Daten von ENQT



10.2 Gesprächsprotokoll Besuch Hamburger Stadtreinigung

Gesprächsprotokoll 15.03.2023: Besuch Stadtreinigung

Erstellt durch Alexander Günther

- Ziel: flächige Datenerfassung
- „Wann fegt die Kehrmaschine, wann sind Kontrollfahrten, wann wird die Fahrt beendet, etc.
- Interesse an folgenden Daten: Straßenzustände, Beschilderung (auch im Winter), Verschmutzung der Straße, Wo befinden sich die Mülltonnen genau ? ,Wo sind häufig große Müllablagerungen, Verstehen des Qualitätsniveaus Mobilfunk (Ausschreibung der Stadtreinigung alle 4 Jahre), Fahrbahntemperatur, Rückwärtsfahrten dokumentieren
- **Wo ist der Nutzen für die Stadtreinigung? Direkte Anwendung für die Stadtreinigung?**
- Ablagerungen → Verunreinigungs-App
- Interner Bedarf: „Wie sieht es draußen aus, im Einsatz“
- Dienstleistungsbetrag für das Sammeln von Daten
- Arbeitnehmer dürfen in den Müllsammlern nicht eingeschränkt werden
- Ein System, welches viele (alle) Funktionen vereint → keine Inselprojekt Lösung
- Erfahrungen sammeln, um zu lernen, was man für die Müllsammelfahrzeuge, bzw. für ein Endprodukt übernehmen kann.
- GPS +/- 1.5 m - 2 m wäre als Ortsangabe für die Stadtreinigung ausreichend
  - wäre es möglich einzelne Fahrspuren zu erkennen (Genauigkeit < 1m)
  - „Wo genau wurde gekehrt ?“
  - Datenschutz ! Betriebsrat ?
- Müllsammler oder Kehrmaschine?
  - Tendenz Müllsammler
- Tests vorerst nur an einem Fahrzeug
- Thema Mobilfunkmessung für Hamburg auf den Weg bringen
  - Gibt es ein städtisches Unternehmen, welches ein solches Projekt bereits durchführt ?
- Momentan Projekte zur Optimierung der Stadtreinigung (Kehrmaschinen).
  - „Wo genau ist der Fahrer gefahren?“ + Bild am Ende der Reinigungsfahrt
- Daten sammeln und intern nutzen.
- Prototypisches Testen auch in Kehrmaschinen?
- SAP & GIS-System
  - „standardisiertes“ modulares System, einfach integrierbar & leichter Zugriff

- Weiteres Vorgehen:
  - Systementwurf grob
  - Stadtreinigung arbeitet Use-Cases aus und meldet sich innerhalb von 2 Wochen zurück
  - Kerninformationen, DSGVO, weiterleiten
  - Kooperationsvertrag

10.3 Gebäudehöhenerkennung – Python-Code

```
#!/usr/bin/env python3
#author: Alexander Günther

import rospy
from sensor_msgs.msg import PointCloud2 as pc2
from sensor_msgs.point_cloud2 import read_points
#import sensor_msgs.point_cloud2 as pc22
import numpy as np
import math as math
import ros_numpy
from sensor_msgs.msg import PointField
from sensor_msgs import point_cloud2
from sbg_driver.msg import SbgEkfNav
from sbg_driver.msg import SbgEkfEuler
from geodesy import utm
from canbus_identifikation.msg import Gebaudehoehe
import csv
import time
import open3d
import copy
from sensor_msgs.msg import NavSatFix

global sbg_ekf_lat
global sbg_ekf_lon
global sbg_ekf_altitude
global Location_in_UTM
global angle

sbg_ekf_lat = 0
sbg_ekf_lon = 0
sbg_ekf_altitude = 0
angle = 0
Location_in_UTM = []

#with open('/home/workstation3/Desktop/Gebaudehoehen.csv', 'w', newline='') as
csvfile:
#    datawriter = csv.writer(csvfile)
#    datawriter.writerow(['Time', 'lat_links', 'lon_links', 'alt_links',
'Gebaudehoehe_links', 'lat_rechts', 'lon_rechts', 'alt_rechts',
'Gebaudehoehe_rechts', 'Fahrzeug_lat', 'Fahrzeug_lon', 'Fahrzeug_alt'])

def rospc_to_o3dpc(rospc, remove_nans=False):
    """ covert ros point cloud to open3d point cloud
    Args:
        rospc (sensor.msg.PointCloud2): ros point cloud message
        remove_nans (bool): if true, ignore the NaN points
```

```
    Returns:
        o3dpc (open3d.geometry.PointCloud): open3d point cloud
    """
    field_names = [field.name for field in rospc.fields]
    is_rgb = 'rgb' in field_names
    cloud_array = ros_numpy.point_cloud2.pointcloud2_to_array(rospc).ravel()
    if remove_nans:
        mask = np.isfinite(cloud_array['x']) & np.isfinite(cloud_array['y']) &
np.isfinite(cloud_array['z'])
        cloud_array = cloud_array[mask]
    if is_rgb:
        cloud_npy = np.zeros(cloud_array.shape + (4,), dtype=float)
    else:
        cloud_npy = np.zeros(cloud_array.shape + (3,), dtype=float)

    cloud_npy[...,0] = cloud_array['x']
    cloud_npy[...,1] = cloud_array['y']
    cloud_npy[...,2] = cloud_array['z']
    o3dpc = open3d.geometry.PointCloud()

    if len(np.shape(cloud_npy)) == 3:
        cloud_npy = np.reshape(cloud_npy[:, :, :3], [-1, 3], 'F')
    o3dpc.points = open3d.utility.Vector3dVector(cloud_npy[:, :3])

    if is_rgb:
        rgb_npy = cloud_array['rgb']
        rgb_npy.dtype = np.uint32
        r = np.asarray((rgb_npy >> 16) & 255, dtype=np.uint8)
        g = np.asarray((rgb_npy >> 8) & 255, dtype=np.uint8)
        b = np.asarray(rgb_npy & 255, dtype=np.uint8)
        rgb_npy = np.asarray([r, g, b])
        rgb_npy = rgb_npy.astype(np.float)/255
        rgb_npy = np.swapaxes(rgb_npy, 0, 1)
        o3dpc.colors = open3d.utility.Vector3dVector(rgb_npy)
    return o3dpc

BIT_MOVE_16 = 2**16
BIT_MOVE_8 = 2**8

def o3dpc_to_rospc(o3dpc, frame_id=None, stamp=None):
    """ convert open3d point cloud to ros point cloud
    Args:
        o3dpc (open3d.geometry.PointCloud): open3d point cloud
        frame_id (string): frame id of ros point cloud header
        stamp (rospy.Time): time stamp of ros point cloud header
    Returns:
        rospc (sensor.msg.PointCloud2): ros point cloud message
    """
```

```
    cloud_npy = np.asarray(copy.deepcopy(o3dpc.points))
    is_color = o3dpc.colors


    n_points = len(cloud_npy[:, 0])
    if is_color:
        data = np.zeros(n_points, dtype=[
        ('x', np.float32),
        ('y', np.float32),
        ('z', np.float32),
        ('rgb', np.uint32)
        ])
    else:
        data = np.zeros(n_points, dtype=[
            ('x', np.float32),
            ('y', np.float32),
            ('z', np.float32)
            ])
    data['x'] = cloud_npy[:, 0]
    data['y'] = cloud_npy[:, 1]
    data['z'] = cloud_npy[:, 2]
    if is_color:
        rgb_npy = np.asarray(copy.deepcopy(o3dpc.colors))
        rgb_npy = np.floor(rgb_npy*255) # nx3 matrix
        rgb_npy = rgb_npy[:, 0] * BIT_MOVE_16 + rgb_npy[:, 1] * BIT_MOVE_8 +
rgb_npy[:, 2]
        rgb_npy = rgb_npy.astype(np.uint32)
        data['rgb'] = rgb_npy

    rospc = ros_numpy.msgify(pc2, data)
    if frame_id is not None:
        rospc.header.frame_id = frame_id

    if stamp is None:
        rospc.header.stamp = rospy.Time.now()
    else:
        rospc.header.stamp = stamp
    rospc.height = 1
    rospc.width = n_points
    rospc.fields = []
    rospc.fields.append(PointField(
                            name="x",
                            offset=0,
                            datatype=PointField.FLOAT32, count=1))
    rospc.fields.append(PointField(
                            name="y",
                            offset=4,
                            datatype=PointField.FLOAT32, count=1))
    rospc.fields.append(PointField(
```

```
                                name="z",
                                offset=8,
                                datatype=PointField.FLOAT32, count=1))
    if is_color:
        rospc.fields.append(PointField(
                        name="rgb",
                        offset=12,
                        datatype=PointField.UINT32, count=1))
        rospc.point_step = 16
    else:
        rospc.point_step = 12

    rospc.is_bigendian = False
    rospc.row_step = rospc.point_step * n_points
    rospc.is_dense = True
    return rospc


def Lidar_Callback(Lidar_data):
    start = time.time()
    hoeheDesLidar = 1.87
    ignoreDataOnStreetLevel = 5 # Entfernen der Daten zwischen Straße und 5 cm
darüber
    sbg_ekf_lat1 = sbg_ekf_lat
    sbg_ekf_lon1 = sbg_ekf_lon
    sbg_ekf_altitude1 = sbg_ekf_altitude
    Location_in_UTM1 = Location_in_UTM

    header = Lidar_data.header

    ## XYZ-Koordinaten aus dem PointCLoud2-Format erstellen
    pc = ros_numpy.numpify(Lidar_data)
    points=np.zeros((pc.shape[0],5))
    points[:,0]=pc['x']
    points[:,1]=pc['y']
    points[:,2]=pc['z']
    points[:,3]=pc['intensity']
    points[:,4]=pc['ring']

    arra = points[:,4].astype(int)
    # Erstellen eines leeren Arrays für das Erzeugen von einer Ausgabe, im
PointCloud2-Format
    data = np.zeros(len(pc['x']),dtype = [('x',np.float32),('y',
np.float32),('z', np.float32),('intensity', np.float32),('ring',
np.uint16),('null',np.uint16)])
    data1 = np.zeros(len(pc['x']),dtype = [('x',np.float32),('y',
np.float32),('z', np.float32),('intensity', np.float32),('ring',
np.uint16),('null',np.uint16)])
```

```
    # Transformationsmatrix aufstellen, umd die Lidar Daten zu korrigieren
    Drehwinkel_um_x = np.deg2rad(29)
    Rx = np.array([[1,0,0],
        [0,math.cos(Drehwinkel_um_x),-math.sin(Drehwinkel_um_x)],
        [0,math.sin(Drehwinkel_um_x),math.cos(Drehwinkel_um_x)]])
    Drehwinkel_um_y = np.deg2rad(0.8)
    Ry = np.array([[np.cos(Drehwinkel_um_y), 0.0 ,np.sin(Drehwinkel_um_y)],
                   [0.0, 1.0 , 0.0],
                   [-np.sin(Drehwinkel_um_y), 0.0, np.cos(Drehwinkel_um_y)]])
    transform = np.matmul(Rx , Ry)
    # Zähler für die Neuzuweisung der Daten
    a = 0
    b = 0
    # Lidar Daten nach gewünschten Aspekten filtern
    for i in range(len(arra)):
        # Auswahl der gewünschten Lidar-Strahl-Daten
        if points[i,4].astype(int) >= 0: #24 ist der erste Strahl der vor dem
Fahrzeug auf die Straße gelangt
            # Den Winkel des Lidar korrigieren (Transformationsmatrix)
            coordinates =
np.array([points[i,0].astype(np.float32),points[i,1].astype(np.float32),points[i,
2].astype(np.float32)])
            transformed_coordinates = coordinates.dot(transform)
            # Die neuen x,y,z werte neu zuweisen (Koordinatentransformation)
            if ((transformed_coordinates[2].astype(np.float32)+hoeheDesLidar) >=
ignoreDataOnStreetLevel and #Werte unter Schwellwert ignorieren
                transformed_coordinates[1].astype(np.float32) <= -10 and
                transformed_coordinates[1].astype(np.float32) >= -80 and
                transformed_coordinates[0].astype(np.float32) >= 0):
                data['x'][a] = transformed_coordinates[0].astype(np.float32)
                data['y'][a] = transformed_coordinates[1].astype(np.float32)
                data['z'][a] =
transformed_coordinates[2].astype(np.float32)+hoeheDesLidar # Nullpunkt auf die
Straße legen
                data['intensity'][a] = points[i,3].astype(np.float32)
                data['ring'][a] = points[i,4].astype(np.uint16)
                a = a+1
            elif ((transformed_coordinates[2].astype(np.float32)+hoeheDesLidar)
>= ignoreDataOnStreetLevel and #Werte unter Schwellwert ignorieren
                transformed_coordinates[1].astype(np.float32) <= -10 and
                transformed_coordinates[1].astype(np.float32) >= -80 and
                transformed_coordinates[0].astype(np.float32) <= 0):
                data1['x'][b] = transformed_coordinates[0].astype(np.float32)
                data1['y'][b] = transformed_coordinates[1].astype(np.float32)
                data1['z'][b] =
transformed_coordinates[2].astype(np.float32)+hoeheDesLidar # Nullpunkt auf die
Straße legen
                data1['intensity'][b] = points[i,3].astype(np.float32)
                data1['ring'][b] = points[i,4].astype(np.uint16)
```

```
                b = b+1

    fields = [PointField('x', 0, PointField.FLOAT32, 1),
            PointField('y', 4, PointField.FLOAT32, 1),
            PointField('z', 8, PointField.FLOAT32, 1),
            PointField('intensity', 16, PointField.FLOAT32, 1),
            PointField('ring', 20, PointField.UINT16, 1),
            PointField('null',30,PointField.UINT16, 1)]

    # Ros PointCloud2 msg erstellen
    msg = point_cloud2.create_cloud(header,fields,data)

    # Format der PointCloud zu open3d Pointcloud umwandeln
    pcd = rospc_to_o3dpc(msg,remove_nans=False)
    pcd = pcd.voxel_down_sample(voxel_size=0.02)

    # statistische Ausreißer Entfernung(Statistical outlier removal - SOR)
    pcd_stat, ind_stat =
pcd.remove_statistical_outlier(nb_neighbors=30,std_ratio=0.5)

    # Translate to visualize:
    points = np.asarray(pcd_stat.points)
    points += [0, 0, 0]
    pcd_stat.points = open3d.utility.Vector3dVector(points)

    #Ebene in gefilterter Pointcloud bestimmen (RANSAC)
    plane_model, inliers = pcd_stat.segment_plane(distance_threshold = 0.1,
ransac_n=50, num_iterations=800) # 0.15
    inlier_cloud = pcd_stat.select_by_index(inliers)
    msg = o3dpc_to_rospc(inlier_cloud, frame_id=Lidar_data.header.frame_id,
stamp=Lidar_data.header.stamp)
    plane_pc = ros_numpy.numpify(msg)

    # Kontrolle ob die Ebene annähernd vertical im Raum liegt
    normalen_Vektor = plane_model[0:3]
    Gebaudehoehe_rechts = 0
    if normalen_Vektor[2]<=0.1:
        # Die Gebäudehöhe ist hier der höchste gemessene Punkt in der vertikalen
Ebene
        Gebaudehoehe_rechts = plane_pc['z'].max()
        index_at_highest_rechts = np.argmax(plane_pc['z'])

    # Alle Auswertungsschritte nocheinmal für die linke Seite, weil RANSAC-
Algorithmus immer die beste Ebene zurück gibt
    msg1 = point_cloud2.create_cloud(header,fields,data1)
    pcd1 = rospc_to_o3dpc(msg1,remove_nans=False)
    pcd1 = pcd1.voxel_down_sample(voxel_size=0.02)
    pcd_stat1, ind_stat1 =
pcd1.remove_statistical_outlier(nb_neighbors=50,std_ratio=0.25)
```

```
    points1 = np.asarray(pcd_stat1.points)
    points1 += [0, 0, 0]
    pcd_stat1.points = open3d.utility.Vector3dVector(points1)
    plane_model1, inliers1 = pcd_stat1.segment_plane(distance_threshold = 0.1,
ransac_n=50, num_iterations=800) # 0.15
    inlier_cloud1 = pcd_stat1.select_by_index(inliers1)
    msg1 = o3dpc_to_rospc(inlier_cloud1, frame_id=Lidar_data.header.frame_id,
stamp=Lidar_data.header.stamp)
    plane_pc1 = ros_numpy.numpify(msg1)
    # Kontrolle ob die Ebene annähernd vertical im Raum liegt
    normalen_Vektor1 = plane_model1[0:3]
    Gebaudehoehe_links = 0
    if normalen_Vektor1[2]<=0.1:
        # Die Gebäudehöhe ist hier der höchste gemessene Punkt in der vertikalen
Ebene
        Gebaudehoehe_links = plane_pc1['z'].max()
        index_at_highest_links = np.argmax(plane_pc1['z'])

    if sbg_ekf_new == 1 and Gebaudehoehe_links !=0 and Gebaudehoehe_rechts !=0:
        # Berechnen der Gebäudepositionen in Lat,lon mittels utm-Koordinaten
        x_links = plane_pc1['x'][index_at_highest_links]
        y_links = plane_pc1['y'][index_at_highest_links]
        winkel_links = abs((np.rad2deg(math.atan2(y_links,x_links)))-90) # Winkel
zur Fahrzeuglängsachse
        distanz_links = np.sqrt(x_links**2 + y_links**2)

        wegstrecke_in_north = distanz_links*np.cos(np.deg2rad(360-winkel_links)-
angle)
        wegstrecke_in_east = -distanz_links*np.sin(np.deg2rad(360-winkel_links)-
angle)
        Object_location = utm.UTMPoint()
        Object_location.easting = Location_in_UTM1.easting + wegstrecke_in_east
        Object_location.northing = Location_in_UTM1.northing +
wegstrecke_in_north
        Object_location.altitude = Location_in_UTM1.altitude
        Object_location.zone = Location_in_UTM1.zone
        Object_location.band = Location_in_UTM1.band
        Object_location = Object_location.toMsg()

        x_rechts= plane_pc['x'][index_at_highest_rechts]
        y_rechts= plane_pc['y'][index_at_highest_rechts]
        winkel_rechts = abs((np.rad2deg(math.atan2(y_rechts,x_rechts)))-90) #
Winkel zur Fahrzeuglängsachse
        distanz_rechts = np.sqrt(x_rechts**2 + y_rechts**2)
        wegstrecke_in_north_rechts = distanz_rechts*np.cos(np.deg2rad(360-
winkel_rechts)-angle)
        wegstrecke_in_east_rechts = -distanz_rechts*np.sin(np.deg2rad(360-
winkel_rechts)-angle)
        Object_location_rechts = utm.UTMPoint()
```

```
        Object_location_rechts.easting = Location_in_UTM1.easting +
wegstrecke_in_east_rechts
        Object_location_rechts.northing = Location_in_UTM1.northing +
wegstrecke_in_north_rechts
        Object_location_rechts.altitude = Location_in_UTM1.altitude
        Object_location_rechts.zone = Location_in_UTM1.zone
        Object_location_rechts.band = Location_in_UTM1.band
        Object_location_rechts = Object_location_rechts.toMsg()

        # Nachricht -> Häuser links -> Visu
        Nachricht = NavSatFix()
        Nachricht.header = header
        Nachricht.latitude = Object_location.latitude
        Nachricht.longitude = Object_location.longitude
        Nachricht.altitude = Object_location.altitude
        pub_pos.publish(Nachricht)

        # Nachricht1 -> Häuser rechts -> Visu
        Nachricht1 = NavSatFix()
        Nachricht1.header = header
        Nachricht1.latitude = Object_location_rechts.latitude
        Nachricht1.longitude = Object_location_rechts.longitude
        Nachricht1.altitude = Object_location_rechts.altitude
        pub_pos_rechts.publish(Nachricht1)

        # Nachricht2 -> Position des Fahrzeug zur Datenaufnahme
        Nachricht2 = NavSatFix()
        Nachricht2.header = header
        Nachricht2.latitude = sbg_ekf_lat1
        Nachricht2.longitude = sbg_ekf_lon1
        Nachricht2.altitude = sbg_ekf_altitude1
        pub_pos_auto.publish(Nachricht2)

        # Publisher für die Gebäudehöhen mit zugehörigen koordinaten
        Gebaude = Gebaudehoehe()
        Gebaude.header = header
        Gebaude.latitude_links = Object_location.latitude
        Gebaude.longitude_links = Object_location.longitude
        Gebaude.altitude_links = Object_location.altitude
        Gebaude.Gebaudehoehe_links = Gebaudehoehe_links
        Gebaude.latitude_rechts = Object_location_rechts.latitude
        Gebaude.longitude_rechts = Object_location_rechts.longitude
        Gebaude.altitude_rechts = Object_location_rechts.altitude
        Gebaude.Gebaudehoehe_rechts = Gebaudehoehe_rechts
        pub_Gebaudehoehe.publish(Gebaude)
        #with open('/home/workstation3/Desktop/Gebaudehoehen.csv', 'a',
newline='') as csvfile:
        #    datawriter = csv.writer(csvfile)
```

```
        #    #datawriter.writerow(['Time', 'lat_links', 'lon_links', 'alt_links',
'Gebaudehoehe_links', 'lat_rechts', 'lon_rechts', 'alt_rechts',
'Gebaudehoehe_rechts'])
        #    datawriter.writerow([Gebaude.header, Gebaude.latitude_links,
Gebaude.longitude_links, Gebaude.altitude_links, Gebaude.Gebaudehoehe_links,
Gebaude.latitude_rechts, Gebaude.longitude_rechts, Gebaude.altitude_rechts,
Gebaude.Gebaudehoehe_rechts, sbg_ekf_lat1,sbg_ekf_lon1,sbg_ekf_altitude1])

    # Publisher für Pointclouds, welche zur Höhenberechnung verwendet wurden.
        pub_cloud_rechts.publish(msg)
        pub_cloud_links.publish(msg1)
    stop = time.time()
    Laufzeit = stop - start
    print(Laufzeit)
    return

def sbg_ekf_Callback(data):
    global sbg_ekf_lon
    sbg_ekf_lon = data.longitude
    global sbg_ekf_lat
    sbg_ekf_lat = data.latitude
    global sbg_ekf_altitude
    sbg_ekf_altitude = data.altitude
    global Location_in_UTM
    Location_in_UTM = utm.fromLatLong(sbg_ekf_lat, sbg_ekf_lon, sbg_ekf_altitude)
    global sbg_ekf_new # Change Data Flag
    sbg_ekf_new = 1
    return

def Euler_callback(data): # data angles are in rad
    global angle
    angle = data.angle.z

if __name__ == '__main__':
    #Node initialisieren
    rospy.init_node("Gebaeudehoehe_node")
    # Publisher & Subscriber erstellen
    sub_Velo = rospy.Subscriber("/velodyne_points", pc2,
queue_size=1,buff_size=20000000, callback=Lidar_Callback) #Lidar Bilder sind
veraltet. Wie gelang ich an die aktuellen heran
    sub_ekf_nav = rospy.Subscriber("/sbg/ekf_nav", SbgEkfNav,
callback=sbg_ekf_Callback)
    sub_euler =
rospy.Subscriber("/sbg/ekf_euler",SbgEkfEuler,callback=Euler_callback)

    pub_Gebaudehoehe = rospy.Publisher('/Gebaeudehoehe_msg', Gebaudehoehe,
queue_size=1)
    pub_cloud_links = rospy.Publisher('/Gebaeudehoehe_links', pc2, queue_size=1)
```

```
    pub_cloud_rechts = rospy.Publisher('/Gebaeudehoehe_rechts', pc2,
queue_size=1)
    pub_pos = rospy.Publisher('/Gebaudeposition_links_msg', NavSatFix,
queue_size=1)
    pub_pos_rechts = rospy.Publisher('/Gebaudeposition_rechts_msg', NavSatFix,
queue_size=1)
    pub_pos_auto = rospy.Publisher('/Gebaudeposition_auto_msg', NavSatFix,
queue_size=1)
    #Console info
    rospy.loginfo("Subscriber has been started.")
    rospy.loginfo("Node has been started")
    # Dauerhafte Ausführung, bis zum Abbruch des Nodes
    rospy.spin()
    # Der code befindet sich in der Callback-funktion des Subscribers.
    # Immer wenn etwas empfangen wird, wird es verarbeitet und gepublished.
```

10.4 Durchfahrtbreitenerkennung – Python-Code

```
#!/usr/bin/env python3
#author: Alexander Günther

import rospy
from sensor_msgs.msg import PointCloud2 as pc2
from canbus_identifikation.msg import Durchfahrtsbreite
import numpy as np
import math as math
import ros_numpy
from sensor_msgs.msg import PointField
from sensor_msgs import point_cloud2
from sbg_driver.msg import SbgEkfNav
import csv

#with open('/home/workstation3/Desktop/Durchfahrtsbreite2.csv', 'w', newline='')
as csvfile:
#    datawriter = csv.writer(csvfile)
#    datawriter.writerow(['lat','lon','Durchfahrtsbreite'])


def Lidar_Callback(Lidar_data):
    try:
        hoeheDesLidar = 1.87
        ignoreDataUnderStreet =  0.5 #Wert in Metern unterhalb der Straße
        hoeheDerStraßenkante = 0.30 #Wert in Metern oberhalb der Straßenebene
        maxStraßenbreite = 30 #Maximale Straßenbreite in metern
        latitude_now = lat
        longitude_now = lon
        altitude_now = altitude

        header = Lidar_data.header

        data = np.zeros(100, dtype=[
            ('x', np.float32),
            ('y', np.float32),
            ('z', np.float32),
            ('intensity', np.float32),
            ('ring', np.float32)
        ])
        ## XYZ-Koordinaten aus dem PointCLoud2-Format erstellen
        pc = ros_numpy.numpify(Lidar_data)
        points=np.zeros((pc.shape[0],5))
        points[:,0]=pc['x']
        points[:,1]=pc['y']
        points[:,2]=pc['z']
        points[:,3]=pc['intensity']
        points[:,4]=pc['ring']
```

```
        arra = points[:,4].astype(int)
        # Erstellen eines leeren Arrays für das Erzeugen von einer Ausgabe, im
PointCloud2-Format
        data = np.zeros(len(pc['x']),dtype = [('x',np.float32),('y',
np.float32),('z', np.float32),('intensity', np.float32),('ring',
np.uint16),('null',np.uint16)])

        # Transformationsmatrix aufstellen, umd die Lidar Daten zu korrigieren
        Drehwinkel_um_x = np.deg2rad(29)
        #rotation = np.deg2rad(100)
        Rx = np.array([[1,0,0],
            [0,math.cos(Drehwinkel_um_x),-math.sin(Drehwinkel_um_x)],
            [0,math.sin(Drehwinkel_um_x),math.cos(Drehwinkel_um_x)]])
        Drehwinkel_um_y = np.deg2rad(0.8)
        Ry = np.array([[np.cos(Drehwinkel_um_y), 0.0 ,np.sin(Drehwinkel_um_y)],
                       [0.0, 1.0 , 0.0],
                       [-np.sin(Drehwinkel_um_y), 0.0,
np.cos(Drehwinkel_um_y)]])
        transform = np.matmul(Rx , Ry)
        a = 0
        # Lidar Daten nach gewünschten Aspekten filtern
        for i in range(len(arra)):
            # Auswahl der gewünschten Lidar-Strahl-Daten
            if points[i,4].astype(int) == 24: # and points[i,4].astype(int) <=
32:  #24 ist der erste Strahl der vor dem Fahrzeug auf die Straße gelangt
                # Den Winkel des Lidar korrigieren (Transformationsmatrix)
                coordinates =
np.array([points[i,0].astype(np.float32),points[i,1].astype(np.float32),points[i,
2].astype(np.float32)])
                transformed_coordinates = coordinates.dot(transform)
                # Die neuen x,y,z werte neu zuweisen (Koordinatentransformation)
                if ((transformed_coordinates[2].astype(np.float32)+hoeheDesLidar)
>= -ignoreDataUnderStreet and
                    (transformed_coordinates[2].astype(np.float32)+hoeheDesLidar)
<= hoeheDerStraßenkante and
                    transformed_coordinates[0].astype(np.float32) >= -
maxStraßenbreite/2 and
                    transformed_coordinates[0].astype(np.float32) <=
maxStraßenbreite/2 ): #Werte unter der Straßendecke ignorieren
                    data['x'][a] = transformed_coordinates[0].astype(np.float32)
                    data['y'][a] = transformed_coordinates[1].astype(np.float32)
                    data['z'][a] =
transformed_coordinates[2].astype(np.float32)+1.87 # Nullpunkt auf die Straße
legen
                    data['intensity'][a] = points[i,3].astype(np.float32)
                    data['ring'][a] = points[i,4].astype(np.uint16)
                    a = a+1
                else:
```

```
                    data['x'][a] = np.float32(0)
                    data['y'][a] = np.float32(0)
                    data['z'][a] = np.float32(0)
                    data['intensity'][a] = np.float32(0)
                    data['ring'][a] = np.uint16(0)
            else:
                data['x'][a] = np.float32(0)
                data['y'][a] = np.float32(0)
                data['z'][a] = np.float32(0)
                data['intensity'][a] = np.float32(0)
                data['ring'][a] = np.uint16(0)

        fields = [PointField('x', 0, PointField.FLOAT32, 1),
                PointField('y', 4, PointField.FLOAT32, 1),
                PointField('z', 8, PointField.FLOAT32, 1),
                PointField('intensity', 16, PointField.FLOAT32, 1),
                PointField('ring', 20, PointField.UINT16, 1),
                PointField('null',30,PointField.UINT16, 1)]

        # Neue PointCloud2 erstellen, um in rviz anzeigen zu lassen
        msg = point_cloud2.create_cloud(header,fields,data)

        # PointCloud2 veröffentlichen um zu zeigen, in welchem Gebiet auf der
Straße die Berechnung stattfindet
        pub_cloud.publish(msg)

        #find value and index closest to x = 0 #just process data which ist not
zero
        x0 = 0
        difference_array = np.absolute(data['x'][data['x'] != 0]-x0)
        index_closest_to_zero = difference_array.argmin()

        # Anzahl der nötigen Datenpunkte für Bordstein abhängig machen von der
Anzahl der Datenpunkte. Nach außenhin immer weniger.
        count = 0 # Anzahl der Datenpunkte mit Steigung größer SChwellwert, in
einer folge.
        possible_index_min = 0
        possible_index_max = 0
        Bordstein_rechts = 0
        index = index_closest_to_zero
        # Steigung von Punkt zu Punkt berechnen und merken, ist dieser über einem
Schwellwert -> merken. Sind dies die folgenden auch,ist es ein Bordstein.
        # Steigungen von der Fahrzeuglängsachse nach rechts, in den positiven
Bereich.
        while index < len(data['x'][data['x'] != 0])-1:
            #Steigung zur nächsten Koordinate berechnen
            steigung_Am_Index = (data['y'][data['y'] != 0][index+1]-
data['y'][data['y'] != 0][index])/(data['x'][data['x'] != 0][index+1]-
data['x'][data['x'] != 0][index])
```

```
            #min und max index zwischenspeichern
            if abs(steigung_Am_Index) > 1.2:
                count = count + 1
                if count == 1:
                    possible_index_min = index
                elif count == 8:
                    possible_index_max=index
            else:
                count = 0
            index = index + 1
            if count == 8:
                index = len(data['x'][data['x'] != 0])-1
                # Bordstein festlegen für den ersten Index über dem Schwellwert
                Bordstein_rechts = abs(data['x'][data['x'] !=
0][possible_index_min])
                # Vergleich z Koordinaten sprung von possible index und letztem
index
                Höhe_Bordstein_rechts = data['z'][data['z'] !=
0][possible_index_max]-data['z'][data['z'] != 0][possible_index_min]

        count = 0
        possible_index_min = 0
        possible_index_max = 0
        index = index_closest_to_zero
        # Steigung von Punkt zu Punkt berechnen und merken, ist dieser über einem
Schwellwert -> merken. Sind dies die folgenden auch,ist es ein Bordstein.
        # Steigungen von der Fahrzeuglängsachse nach links, in den negativen
Bereich.
        if Bordstein_rechts != 0:
            while index > 0+1:
                steigung_Am_Index = (data['y'][data['y'] != 0][index+1]-
data['y'][data['y'] != 0][index])/(data['x'][data['x'] != 0][index+1]-
data['x'][data['x'] != 0][index])

                if abs(steigung_Am_Index) > 1.2:
                    count = count + 1
                    if count == 1:
                        possible_index_min = index
                    elif count == 8:
                        possible_index_max=index
                else:
                    count = 0

                index = index - 1
                if count == 8:
                    index = 0+1

                    Bordstein_links = abs(data['x'][data['x'] !=
0][possible_index_min])
```

```
                    Höhe_Bordstein_links = data['z'][data['z'] !=
0][possible_index_max]-data['z'][data['z'] != 0][possible_index_min] # lediglich
die Höhe der n(count) Datenpunkte
                    #print(f'Höhe Bordstein links =
{Höhe_Bordstein_links*100:.4f} cm')
                    #print(f'Höhe Bordstein rechts =
{Höhe_Bordstein_rechts*100:.4f} cm')
                    Bordsteinbreite = Bordstein_links + Bordstein_rechts
                    #print(f'Durchfahrtsbreite = {Bordsteinbreite} m')
                    aktuelle_Durchfahrtsbreite = Durchfahrtsbreite()
                    aktuelle_Durchfahrtsbreite.header = header
                    aktuelle_Durchfahrtsbreite.latitude = latitude_now
                    aktuelle_Durchfahrtsbreite.longitude = longitude_now
                    aktuelle_Durchfahrtsbreite.altitude = altitude_now
                    aktuelle_Durchfahrtsbreite.Durchfahrtsbreite =
Bordsteinbreite
                    pub_Durchfahrtsbreite.publish(aktuelle_Durchfahrtsbreite)
                    #with
open('/home/workstation3/Desktop/Durchfahrtsbreite2.csv', 'a', newline='') as
csvfile:
                    #    datawriter = csv.writer(csvfile)
                    #    datawriter.writerow([aktuelle_Durchfahrtsbreite.latitude
,aktuelle_Durchfahrtsbreite.longitude,aktuelle_Durchfahrtsbreite.Durchfahrtsbreit
e])


    except Exception as error:
        print(error)

    return

def GPS_callback(data):
    global lat
    lat = data.latitude
    global lon
    lon = data.longitude
    global altitude
    altitude = data.altitude


if __name__ == '__main__':
    #Node initialisieren
    rospy.init_node("Durchfahrtsbreite_node")
    # Publisher & Subscriber erstellen
    sub_Velo = rospy.Subscriber("/velodyne_points", pc2, callback=Lidar_Callback)
    sub_gps = rospy.Subscriber("/sbg/ekf_nav",SbgEkfNav,callback=GPS_callback)
    pub_cloud = rospy.Publisher('/Strassenbreite', pc2, queue_size=1)
    pub_Durchfahrtsbreite = rospy.Publisher('/Durchfahrtsbreite_msg',
Durchfahrtsbreite, queue_size=1)
```

```
    #sub_gps = rospy.Subscriber("/sbg/ekf_nav",SbgEkfNav,callback=GPS_callback)
    #Console info
    rospy.loginfo("Subscriber has been started.")
    rospy.loginfo("Node has been started")
    # Dauerhafte Ausführung, bis zum Abbruch des Nodes

    rospy.spin()
    # Der code befindet sich in der Callback-funktion des Subscribers.
    # Immer wenn etwas empfangen wird, wird es verarbeitet und gepublished.
```

10.5 Mülltonnenerkennung – Python Code – Training des Netzes

```
# -*- coding: utf-8 -*-
"""aktuell1_yolov4-tiny-custom_TRAINING.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1q3_6fPTEGfRxnx657kjwxWr9jkkQTIWV

# **TRAIN A CUSTOM YOLOv4-tiny OBJECT DETECTOR**

# **In this tutorial, we will be training our custom detector for mask detection using YOLOv4-tiny and Darknet**
## This notebook is part of this [blog post](https://techzizou.com/train-a-custom-yolov4-tiny-object-detector-using-google-colab-tutorial-for-beginners/)
## **3(d) Upload the *`process.py`* script file to the *`yolov4-tiny`* folder on your drive**

**To divide all image files into 2 parts. 90% for train and 10% for test.**

This *process.py* script creates the files *train.txt* & *test.txt* where the *train.txt* file has paths to 90% of the images and *test.txt* has paths to 10% of the images.

You can download the ***`process.py`*** script from my [GitHub](https://github.com/techzizou/yolov4-tiny-custom_Training).

# **4) Mount drive and link your folder**
"""

# Commented out IPython magic to ensure Python compatibility.
#mount drive
# %cd ..
from google.colab import drive
drive.mount('/content/gdrive')

# this creates a symbolic link so that now the path /content/gdrive/My\ Drive/ is equal to /mydrive
!ln -s /content/gdrive/My\ Drive/ /mydrive

# list contents in yolov4-tiny folder in your drive
!ls /mydrive/yolov4-tiny

"""# **5) Make changes in the `makefile` to enable OPENCV and GPU**"""

# Commented out IPython magic to ensure Python compatibility.
```

```
# change makefile to have GPU and OPENCV enabled
# also set CUDNN, CUDNN_HALF and LIBSO to 1

# %cd /content/darknet/
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile

"""# **6) Run `make` command to build darknet**"""

# build darknet
!make

"""# **7) Copy files from your drive to the darknet directory**"""

# Commented out IPython magic to ensure Python compatibility.
# Clean the data and cfg folders first except the labels folder in data which is
required

# %cd data/
!find -maxdepth 1 -type f -exec rm -rf {} \;
# %cd ..

# %rm -rf cfg/
# %mkdir cfg

#copy the datasets zip file to the root darknet folder
!cp /mydrive/yolov4-tiny/obj.zip ../

# unzip the datasets and their contents so that they are now in /darknet/data/
folder
!unzip ../obj.zip -d data/

#copy the custom cfg file from the drive to the darknet/cfg folder
!cp /mydrive/yolov4-tiny/yolov4-tiny-custom.cfg ./cfg

# copy the obj.names and obj.data files so that they are now in /darknet/data/
folder
!cp /mydrive/yolov4-tiny/obj.names ./data
!cp /mydrive/yolov4-tiny/obj.data  ./data

#copy the process.py file from the drive to the darknet directory
!cp /mydrive/yolov4-tiny/process.py ./

"""# **8) Run the *`process.py`* python script to create the *`train.txt`* &
*`test.txt`* files inside the *data* folder**"""
```

```
# run process.py ( this creates the train.txt and test.txt files in our
darknet/data folder )
!python process.py

# list the contents of data folder to check if the train.txt and test.txt files
have been created
!ls data/

"""# **9) Download the pre-trained *`yolov4-tiny`* weights**"""

# Download the yolov4-tiny pre-trained weights file
!wget
https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/yolov4-
tiny.conv.29

"""# **10) Training**

## **Train your custom detector**

For best results, you should stop the training when the average loss is less than
0.05 if possible or at least constantly below 0.3, else train the model until the
average loss does not show any significant change for a while.
"""

# train your custom detector! (uncomment %%capture below if you run into memory
issues or your Colab is crashing)
# %%capture

!./darknet detector train data/obj.data cfg/yolov4-tiny-custom.cfg yolov4-
tiny.conv.29 -dont_show -map
```

10.6 Mülltonnenerkennung und Lokalisierung – Python-Code

```
#!/usr/bin/env python3

import rospy
import cv2
import numpy as np
import os
from geodesy import utm
from sensor_msgs.msg import NavSatFix
from sensor_msgs.msg import Image
from sbg_driver.msg import SbgEkfNav
from sbg_driver.msg import SbgEkfEuler
from ladybug6.msg import ladybug5_rectified_images
from cv_bridge import CvBridge
import csv
bridge = CvBridge()
# globale Variablen können innerhalb von Klassen verhindert werden, dadurch, dass
Variablen die gesamte Objektlebensdauer erhalten bleiben
global altitude
global lat
global lon
global Location_in_UTM
global angle

#with open('/home/workstation3/Desktop/Muelltonnen_positionen.csv', 'w',
newline='') as csvfile:
#    datawriter = csv.writer(csvfile)
#    datawriter.writerow(['header','lat','lon','alt'])

def detect(frame,cfg,weights,classes):
    # Neuronales Netz laden
    net = cv2.dnn.readNet(weights,cfg)
    net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
    net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i-1] for i in net.getUnconnectedOutLayers()]
    # Bild dimensionen ermitteln
    height, width, channels = frame.shape
    # Bild durch das neuronale Netz verarbeiten
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True,
crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)
    class_ids = []
    confidences = []
    boxes = []
    # Schleife um alle Vorhersagen zu prüfen
```

```
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.82: # Ist die Sicherheit einer Detektion > 82% ->
Mülltonne erkannt
                # Mülltonnenposition ermitteln
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                # Bounding-Bos Koordinaten
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                # gefundene Mülltonnen speichern
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
    return boxes, confidences


def Rectified_Image_callback(data):
    #aktuelle Gegebenheiten speichern
    actual_position_in_utm = Location_in_UTM
    actual_angle_to_utm = angle #winkel in rad. 0 ist Nord
    # Jedes einzelne Ladybugbild verarbeiten
    for camera in range(5):
        if camera == 0:
            frame = data.camera0
        elif camera == 1:
            frame = data.camera1
        elif camera == 2:
            frame = data.camera2
        elif camera == 3:
            frame = data.camera3
        elif camera == 4:
            frame = data.camera4
        # Image von Rosformat zu OpenCV-format konvertieren
        frame = bridge.imgmsg_to_cv2(frame, desired_encoding="bgr8")
        # Neuronales Netz ausführen
        boxes, confidences = detect(frame,cfg,weights,classes)
        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
        # Alle Detektionen verarbeiten
        for i in range(len(boxes)):
            if i in indexes:
                x, y, w, h = boxes[i] # Bounding-Box Eigenschaften
                if h > 30: #Boundingboxhöhe muss mindestens 30 Pixel betragen
```

```
                    cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),3)
                    cv2.putText(frame, "{}%".format(int(confidences[i]*100)),
(x,y), cv2.FONT_HERSHEY_SIMPLEX, 2, (0,0,255), 2, cv2.LINE_AA)
                    #Berechnung des Winkel der Mülltonne zur Fahrzeuglängsachse,
sowie die Entfernung
                    angle_zum_Fahrzeug = get_angle(x,w,camera)
                    object_height = 1.2 # eigentlich 0.8 m = höhe einer Mülltonne
                    distanz_zum_Fahrzeug = get_distance(h,object_height)
                    # Hier fehlt noch der Versatz von ladybug zu Mitte
Hinterachse (SBG-bezugsystem)
                    # Mülltonnenposition in Fahrzeug-UTM-Koordinaten umrechnen
und kombinieren
                    Wegstrecke_in_north =
distanz_zum_Fahrzeug*np.cos(np.deg2rad(360-angle_zum_Fahrzeug)-
actual_angle_to_utm)
                    Wegstrecke_in_east = -
distanz_zum_Fahrzeug*np.sin(np.deg2rad(360-angle_zum_Fahrzeug)-
actual_angle_to_utm)
                    Object_location = utm.UTMPoint()
                    Object_location.easting = actual_position_in_utm.easting +
Wegstrecke_in_east
                    Object_location.northing = actual_position_in_utm.northing +
Wegstrecke_in_north
                    Object_location.altitude = actual_position_in_utm.altitude
                    Object_location.zone = actual_position_in_utm.zone
                    Object_location.band = actual_position_in_utm.band
                    Object_location = Object_location.toMsg()
                    # Nachricht mit der Position der Mülltonne erstellen
                    Nachricht = NavSatFix()
                    Nachricht.header = data.header
                    Nachricht.latitude = Object_location.latitude
                    Nachricht.longitude = Object_location.longitude
                    Nachricht.altitude = Object_location.altitude
                    # Zur Bewertung mit plotly
                    #with
open('/home/workstation3/Desktop/Muelltonnen_positionen.csv', 'a', newline='') as
csvfile:
                    #    datawriter = csv.writer(csvfile)
                    #    datawriter.writerow([Nachricht.header,Nachricht.latitude
,Nachricht.longitude,Nachricht.altitude])

                    # Mülltonnenposition als NavSatFix publishen
                    pub_pos.publish(Nachricht)
                    # frame zurück zu ROS-Image konvertieren
                    frame_copy = bridge.cv2_to_imgmsg(frame, encoding="bgr8")
                    # Frame in ROS publishen, die Detektion sichtbar gemacht
werden soll.
                    pub.publish(frame_copy)
```

```
def GPS_callback(data): #Aktuelle Position des Fahrzeugs ermitteln und
zwischenspeichern
    global lat
    lat = data.latitude
    global lon
    lon = data.longitude
    global altitude
    altitude = data.altitude
    global Location_in_UTM
    Location_in_UTM = utm.fromLatLong(lat, lon, altitude)

def Euler_callback(data): # Winkel des Fahrzeugs Richtung Norden
zwischenspeichern
    global angle
    angle = data.angle.z # data angles are in rad

def get_angle(x, width, camera): # Übernommen aus der Masterarbeit YanLing, mit
Abwandelungen von Maximilian Weltz
    IMAGE_WIDTH = 2047
    HFOV = 101.662896
    middlePixelPosition = width*0.5 + x

    if camera == 0:
        offset_a = 0
        offset_b = 360
    elif camera == 1:
        offset_a = 72
        offset_b = 72
    elif camera == 2:
        offset_a = 144
        offset_b = 144
    elif camera == 3:
        offset_a = 216
        offset_b = 216
    elif camera == 4:
        offset_a = 288
        offset_b = 288

    if(middlePixelPosition >= 1023.5):
        angle = offset_a + ((middlePixelPosition - 1023.5) * HFOV) / IMAGE_WIDTH
    else:
        angle = offset_b - ((1023.5 - middlePixelPosition) * HFOV) / IMAGE_WIDTH
    return ((angle+11.9)%360) # should correct missalignment of camera #11.9

def get_distance(pixel_hight, object_hight): # Übernommen aus der Masterarbeit
YanLing, mit Abwandelungen von Maximilian Weltz
        # aus Arbeit von Yanling
        #define HFOV 101.662896
        VFOV = 111.383104
```

```
        #define IMAGE_WIDTH 2047
        #define IMAGE_HEIGHT 1231
        IMAGE_ORIGINAL_HEIGHT = 2464
        distance = ((object_hight/pixel_hight)*(IMAGE_ORIGINAL_HEIGHT/2))/
np.tan(np.deg2rad(VFOV/2))
        return distance

if __name__ == '__main__':
    ## Load path to model config ##
    dirname = os.path.dirname(__file__)
    cfg = os.path.join('/home/workstation3/Downloads/yolov4-tiny-custom(4).cfg')
    weights = os.path.join('/home/workstation3/Downloads/yolov4-tiny-
custom_best(4).weights')
    classes = ['bake']
    #Node initialisieren
    rospy.init_node("Muelleimer_detection_node")
    # Publisher & Subscriber erstellen
    pub = rospy.Publisher('/Muelleimer', Image, queue_size=1)
    sub = rospy.Subscriber("/ladybug/all_camera/image_rect_color",
ladybug5_rectified_images, callback=Rectified_Image_callback)
    sub_gps = rospy.Subscriber("/sbg/ekf_nav",SbgEkfNav,callback=GPS_callback)
    sub_euler =
rospy.Subscriber("/sbg/ekf_euler",SbgEkfEuler,callback=Euler_callback)
    pub_pos = rospy.Publisher('/Trash_Can_Detect_v2/Muelleimer_Location_msg',
NavSatFix, queue_size=1)
    #Console info
    rospy.loginfo("Publisher & Subscriber has been started.")
    rospy.loginfo("Node has been started")
    rospy.spin()
    # Der code befindet sich in der Callback-funktion des Subscribers.
    # Immer wenn etwas empfangen wird, wird es verarbeitet und gepublished.
```

10.7 Custom Nachrichtentypen in ROS

Gebaudehoehe.msg:

```
# Message type for transmitting a Value with GNSS information and time stamp

Header header

uint32 id

# Koordinaten des gemessenen Gebäudes in WGS84

float64 latitude_links

float64 longitude_links

float64 altitude_links

float32 Gebaudehoehe_links # in Metern

float64 latitude_rechts

float64 longitude_rechts

float64 altitude_rechts

float32 Gebaudehoehe_rechts # in Metern
```

Durchfahrtsbreite.msg:

```
# Message type for transmitting a Value with GNSS information and time stamp

Header header

uint32 id

# Latitude [degrees]. Positive is north of equator; negative is south.

float64 latitude
```

Longitude [degrees]. Positive is east of prime meridian; negative is west.

float64 longitude

Altitude [m]. Positive is above the WGS 84 ellipsoid

(quiet NaN if no altitude is available).

float64 altitude

aktuelle Durchfahrtsbreite in metern

float32 Durchfahrtsbreite

10.8 Gebäudehöhenermittlung Test

Gebäude	Höhe Wirklichkeit	gemessene Höhe	Differenz	Mittelwert	Standardabweichung
Gebäude 1	24	23,47	0,53	23,183125	0,620487647
	24	22,54	1,46		
	24	23,49	0,51		
	24	23,32	0,68		
	24	23,59	0,41		
	24	23,5	0,5		
	24	23,66	0,34		
	24	23,66	0,34		
	24	23,75	0,25		
	24	23,11	0,89		
	24	23,33	0,67		
	24	23,71	0,29		
	24	23,71	0,29		
	24	22,93	1,07		
	24	23,66	0,34		
	24	23,39	0,61		
	24	22,47	1,53		
	24	22,9	1,1		
	24	23,58	0,42		
	24	23,31	0,69		
	24	23,61	0,39		
	24	22,59	1,41		
	24	22,63	1,37		
	24	23,38	0,62		
	24	23,21	0,79		

	24	22,73	1,27			
	24	23,8	0,2			
	24	23,53	0,47			
	24	23,54	0,46			
	24	23,77	0,23			
	24	23,73	0,27			
	24	23,69	0,31			
	24	23,44	0,56			
	24	23,65	0,35			
	24	23,48	0,52			
	24	23,7	0,3			
	24	23,66	0,34			
	24	20,31	3,69			
	24	22,59	1,41			
	24	23,31	0,69			
	24	22,38	1,62			
	24	22,63	1,37			
	24	22,46	1,54			
	24	22,96	1,04			
	24	22,33	1,67			
	24	22,85	1,15			
	24	22,64	1,36			
	24	23,11	0,89			
Gebäude 2	30	24,08	5,92	25,19833333	2,874289767	
	30	26,45	3,55			
	30	29,59	0,41			
	30	23,1	6,9			
	30	24,33	5,67			
	30	22,53	7,47			
	30	28,98	1,02			

	30	30,28	-0,28			
	30	23,42	6,58			
	30	20,02	9,98			
	30	24,92	5,08			
	30	23,34	6,66			
	30	26,21	3,79			
	30	27,175	2,825			
	30	23,55	6,45			
Gebäude 3	30	21,53	8,47	26,40653846	3,176974689	
	30	21,4	8,6			
	30	21,4	8,6			
	30	21,53	8,47			
	30	21,42	8,58			
	30	21,42	8,58			
	30	22,07	7,93			
	30	24,33	5,67			
	30	24,95	5,05			
	30	24,42	5,58			
	30	24,56	5,44			
	30	21,39	8,61			
	30	21,73	8,27			
	30	24,63	5,37			
	30	24,75	5,25			
	30	24,49	5,51			
	30	26,41	3,59			
	30	24,44	5,56			
	30	23,88	6,12			
	30	28,21	1,79			
	30	26,68	3,32			
	30	24,23	5,77			

	30	26,09	3,91		
	30	28,75	1,25		
	30	27,31	2,69		
	30	26,75	3,25		
	30	28,85	1,15		
	30	29,12	0,88		
	30	29,28	0,72		
	30	29,56	0,44		
	30	29,68	0,32		
	30	29,62	0,38		
	30	29,55	0,45		
	30	27,54	2,46		
	30	28,98	1,02		
	30	26,09	3,91		
	30	29,39	0,61		
	30	29,19	0,81		
	30	29,67	0,33		
	30	29,42	0,58		
	30	29,36	0,64		
	30	29,98	0,02		
	30	30,28	-0,28		
	30	29,99	0,01		
	30	30,3	-0,3		
	30	22,48	7,52		
	30	23,42	6,58		
	30	30,08	-0,08		
	30	23,23	6,77		
	30	29,75	0,25		
	30	29,98	0,02		
	30	29,58	0,42		

Gebäude 4	30	24,93	5,07	29,69216216		1,816576657
	30	28,18	1,82			
	30	27,02	2,98			
	30	28,36	1,64			
	30	29,73	0,27			
	30	29,01	0,99			
	30	29,52	0,48			
	30	29,35	0,65			
	30	29,73	0,27			
	30	30,07	-0,07			
	30	29,71	0,29			
	30	29,9	0,1			
	30	30,7	-0,7			
	30	30,3	-0,3			
	30	31	-1			
	30	31	-1			
	30	29,9	0,1			
	30	31,02	-1,02			
	30	30,52	-0,52			
	30	29,99	0,01			
	30	31,13	-1,13			
	30	27,86	2,14			
	30	31,22	-1,22			
	30	31,22	-1,22			
	30	31,3	-1,3			
	30	30,5	-0,5			
	30	30,99	-0,99			
	30	31,5	-1,5			
	30	31,13	-1,13			
	30	30,68	-0,68			
	30	30,99	-0,99			

30	31,33	-1,33	
30	30,04	-0,04	
30	31,23	-1,23	
30	27,15	2,85	
30	26,13	3,87	
30	24,27	5,73	

10.9 Durchfahrtbreite Messwerte des Tests

Durchfahrtbreite gemessen																									
12,47	6,01	9,29	5,73	7,97	16,30	5,14	6,48	11,19	17,62	11,35	6,48	9,55	17,75	7,58	8,94	11,19	17,74	10,80	9,46	10,40	8,29	8,88	11,59	6,80	15,35	5,15
10,42	10,21	9,30	5,69	7,70	11,06	5,15	6,47	11,19	17,76	11,35	6,48	9,55	17,76	7,76	8,95	11,18	7,05	11,89	9,90	10,37	16,97	9,15	4,03	6,81	3,99	5,15
10,17	12,24	9,30	5,50	4,19	16,23	5,15	6,48	11,18	17,85	11,36	6,49	9,54	17,75	9,56	8,94	11,19	13,39	6,52	10,36	10,37	16,90	8,89	9,61	6,81	7,05	5,14
10,02	8,97	9,29	5,53	7,38	14,66	5,15	6,49	11,18	6,53	11,35	6,48	9,54	17,74	18,24	8,95	11,20	9,74	6,54	7,76	8,48	17,01	9,15	11,17	6,79	6,78	5,14
9,75	7,46	9,02	7,19	7,57	4,64	5,14	6,48	11,19	7,51	11,35	6,48	9,53	17,74	18,24	8,95	11,17	13,03	6,53	7,09	8,85	8,31	9,16	11,13	6,82	6,81	5,14
7,87	3,47	9,04	7,10	7,60	4,82	5,14	6,48	9,97	7,65	11,35	6,47	9,50	17,76	6,44	8,93	11,18	12,18	6,51	6,67	9,70	11,21	9,16	10,92	6,80	6,91	5,14
7,69	4,14	9,08	7,19	8,55	7,70	5,15	6,48	7,13	10,67	11,35	6,48	9,17	17,74	13,63	8,94	11,20	12,19	6,53	6,54	9,71	17,64	9,15	11,05	6,81	6,90	5,14
12,67	4,27	8,81	10,46	8,55	4,91	5,15	6,49	6,83	13,82	8,44	6,48	6,53	17,74	9,40	8,96	11,19	4,33	8,20	12,79	9,75	17,76	9,17	11,28	6,80	6,90	5,14
9,70	4,33	11,26	10,46	8,54	7,34	5,16	6,48	6,78	17,66	11,35	6,48	6,52	17,76	9,18	8,94	11,17	9,72	8,18	6,68	9,74	17,76	9,16	8,98	6,82	6,93	5,14
9,71	4,40	8,74	10,45	8,56	7,37	5,17	6,48	6,79	4,72	11,35	6,48	6,51	17,74	3,17	8,93	11,18	4,30	8,20	6,69	9,73	17,76	9,15	8,97	9,81	7,09	5,14
9,82	4,26	11,33	10,40	8,55	7,34	5,18	6,48	6,78	10,06	11,35	6,48	6,51	17,75	1,18	8,94	11,18	4,37	8,21	12,91	9,80	17,75	9,16	8,99	10,42	7,30	5,14
9,73	4,42	8,50	8,81	8,56	6,96	5,20	6,48	6,79	9,94	11,35	6,48	6,49	17,75	4,44	8,94	11,18	4,30	8,21	12,92	12,39	17,75	9,17	8,98	9,78	7,57	5,14
12,36	4,52	8,65	8,83	8,55	9,19	5,21	6,48	16,03	9,94	11,35	6,48	6,51	17,76	6,65	8,95	11,20	9,78	8,21	13,04	4,41	17,76	9,15	8,98	13,86	7,72	5,14
12,36	4,52	8,26	14,56	8,56	7,13	5,86	6,49	14,20	9,80	11,35	6,47	6,51	17,74	6,72	8,95	11,20	4,17	11,72	9,78	7,02	17,75	9,15	8,95	14,98	12,11	5,15
12,37	4,18	11,17	14,58	8,56	6,97	5,26	6,49	14,51	4,14	11,35	6,48	6,50	17,75	6,64	8,95	11,19	16,06	11,64	12,19	7,77	17,75	8,89	8,94	6,56	3,79	5,15
12,31	4,24	11,70	18,21	8,55	9,19	5,32	6,48	6,25	4,25	11,36	6,47	6,48	17,76	6,68	8,95	11,19	15,18	11,48	14,18	9,60	17,75	8,89	8,95	7,39	3,91	5,14
12,24	4,83	9,27	11,93	8,55	7,01	5,41	6,48	6,17	7,13	11,36	6,48	6,49	17,75	6,59	8,95	11,19	13,13	11,43	14,18	7,46	17,75	8,89	8,94	7,60	15,10	5,14
10,11	4,31	11,29	5,51	8,54	6,75	5,65	6,48	12,79	3,93	11,36	6,48	6,54	17,76	6,49	8,93	11,20	7,00	11,40	9,78	9,35	17,75	6,29	8,94	7,70	8,06	5,15
10,10	4,90	11,58	8,41	8,54	9,20	5,89	6,48	13,58	4,15	11,36	6,48	6,46	17,75	6,47	8,94	11,19	13,33	11,37	13,74	9,36	17,76	6,29	8,95	7,59	4,28	5,14
10,05	4,48	11,34	11,84	8,55	17,31	6,16	6,48	13,37	4,15	11,36	6,48	6,47	17,76	7,89	8,94	11,20	13,11	11,36	13,30	9,37	17,75	6,28	8,93	7,53	4,24	5,15

10,0		11,5				11,1				11,3			17,7			11,1	18,8	11,3	13,3		17,7					5,1
7	4,06	7	8,76	8,53	8,28	1	6,48	6,48	4,10	6	6,48	6,48	5	7,34	8,94	8	2	6	7	7,00	5	6,27	8,94	7,45	8,39	4
		15,0								11,3			17,7			11,1	18,8	11,3			17,7				10,9	5,1
6,83	4,67	5	8,68	8,55	6,26	9,18	6,48	6,56	3,85	7	6,48	6,49	6	6,83	8,95	8	8	5	6,50	7,02	5	6,29	8,94	7,46	9	4
11,4		14,9	11,8		13,5					11,3						11,1	18,8	11,3		10,1	17,7				14,5	5,1
7	5,02	9	5	8,55	6	9,17	6,48	4,32	4,22	6	6,48	6,49	5,63	6,57	8,94	9	6	6	6,50	8	5	6,27	8,93	7,36	8	4
11,3										11,3			17,7			11,1	18,8	11,3		10,2	17,7				16,8	5,1
4	5,06	7,26	8,69	8,55	7,33	9,16	6,48	6,59	4,21	6	6,48	6,49	4	6,56	8,93	9	5	6	6,49	4	6	9,22	8,94	7,64	9	5
11,2					15,9					11,3			17,7			11,1	18,7	11,3		10,2	17,7					9,1
1	5,62	7,25	8,71	8,55	0	9,15	6,48	6,73	4,22	6	6,48	6,73	5	6,52	8,94	9	5	6	6,48	6	4	9,22	8,95	9,25	8,50	1
11,2								15,9		11,3			17,6			11,2	18,5	11,3		10,1	17,7					9,7
0	4,85	7,22	8,78	8,56	4,23	9,16	6,48	2	4,22	5	6,48	6,74	9	6,51	8,96	0	9	6	6,48	8	4	7,65	8,93	9,28	5,83	1
11,1		14,1						14,5		11,3			17,7			11,1	10,5	11,3		10,0	17,7			14,2		9,7
7	6,25	5	5,51	8,61	3,87	9,16	6,47	1	4,22	5	6,47	4,03	5	6,53	8,93	7	9	6	6,49	1	5	7,70	8,93	8	5,64	2
11,1		10,8			14,3					11,3			17,7			11,1	17,3	11,3		10,0	17,7			13,6		9,1
6	6,34	0	5,47	8,68	0	7,17	6,48	9,40	4,21	5	6,49	3,83	1	6,52	3,52	9	4	5	6,49	2	5	8,83	8,92	6	5,35	0
11,1		10,8			13,3					11,3						11,1	17,3	11,3		10,0	17,7			13,8		9,1
7	6,62	1	6,03	8,85	9	9,16	6,47	8,75	4,22	4	6,48	8,99	6,01	6,52	3,51	9	6	6	6,48	2	5	9,02	8,95	6	5,36	1
11,1		10,8		28,6												11,1	17,4	11,3			17,7			14,0		9,1
7	6,60	1	6,30	2	4,72	8,89	6,48	8,62	4,22	5,10	6,48	9,01	6,25	6,54	8,94	9	6	6	6,48	7,87	4	9,60	8,94	9	5,35	1
11,1		10,7		14,5												11,1	17,5	11,3			17,7					9,7
7	6,38	9	4,84	1	3,54	9,16	6,49	8,46	4,20	5,12	6,48	9,02	6,01	6,55	8,93	9	7	6	6,48	9,54	6	6,54	8,93	9,26	8,53	0
11,1		10,8														11,1		10,8			16,9			12,0		5,1
8	6,81	0	4,84	9,18	6,55	9,15	6,48	8,47	4,22	5,12	6,48	9,00	9,57	6,56	8,94	5	5,15	0	6,62	9,71	3	9,16	6,49	0	4,23	4
11,1		10,8	11,2						15,0				11,9			11,1		10,8			16,6			12,8		5,1
7	7,19	1	4	9,53	6,54	9,15	6,47	1	4,21	5,77	6,48	6,52	0	9,26	8,95	6	5,29	0	6,62	9,72	1	9,16	6,48	8	4,37	4
11,1		10,8	12,5						15,1					10,4		11,1		10,8			16,3			12,8	12,9	5,1
6	5,39	0	6	9,72	4,08	9,15	6,48	2	4,21	5,10	6,48	7,01	8,99	6	8,94	6	5,49	0	9,22	9,72	9	9,15	6,47	9	2	5
11,1	10,1	10,8							14,9				17,5	11,1		11,1		10,8			15,9			13,0		5,1
9	0	0	6,56	9,78	6,05	9,15	6,48	8,71	3	5,13	6,48	6,51	3	9	8,95	6	5,81	0	6,53	9,73	8	9,15	6,48	1	6,59	4
11,1	11,4	10,8											17,2	12,2		11,1	14,7	10,8			14,9			13,0		5,1
7	4	0	6,58	9,79	6,04	9,17	6,48	8,67	6,86	5,12	6,47	6,53	2	4	8,95	6	6	0	6,60	9,73	6	9,15	6,48	6	6,76	5
11,1	10,5	10,8	10,1										17,3	12,6		11,1	14,5	10,8			14,7					5,1
6	6	0	8	9,73	8,26	9,15	6,48	6,55	6,85	5,14	6,48	6,52	5	1	8,94	6	3	0	6,62	9,76	1	9,15	6,48	6,82	6,95	4
11,1		10,8	10,1										17,7			11,1		10,8		10,4	14,3			17,6		12,9	5,1
6	8,95	1	5	9,72	8,31	9,16	6,48	4,37	6,84	5,14	6,47	6,51	2	7,66	8,94	8	8,01	0	6,70	0	8	9,15	0	6,80	4	4
11,1	12,9	10,7	10,2		13,8			14,3	12,7							11,1		10,8			10,9			17,8		15,6	5,1
7	9	9	6	9,72	7	9,16	6,49	8	7	5,15	6,48	6,51	9,94	7,92	8,94	9	8,39	0	6,94	9,74	3	9,16	3	6,81	3	4
11,1		10,8			16,8			15,7				12,5				11,1		10,7						18,2		12,6	5,1
7	5,00	1	9,56	9,71	5	9,17	6,49	5	4,20	5,15	6,48	3	4,89	8,14	8,94	8	7,48	9	7,47	6,72	8,66	9,15	0	6,81	7	5
11,1		10,8			16,8								13,7			11,1		10,8						18,4			5,1
6	5,20	0	9,25	9,72	1	9,15	6,47	7,11	4,17	5,15	6,48	9,97	9	8,52	8,94	8	9,93	1	7,89	9,75	9,55	9,15	0	6,80	9,81	5
						13,5	13,3		10,0							11,1		10,8						18,5		15,5	5,1
8,94	9,72	9,09	8,94	8,95	6,93	8	8	9,78	4	7,95	8,69	5,40	5,94	9,76	9,65	9	9,79	1	8,41	9,78	8,26	9,16	8	6,81	9	4
						10,5	13,6	10,0		14,5					11,3	11,1		10,8		10,4	17,2		20,0		12,9	5,1
6,75	9,11	9,09	8,96	8,94	8,74	8	1	0	9,93	4	8,71	5,39	5,55	7,79	3	6	7,66	0	9,14	2	0	9,16	8	6,80	4	5
							12,6	10,5		14,4												12,0	12,4			6,4
6,85	9,10	9,71	8,95	8,93	9,00	5,27	7	2	9,67	8	8,69	5,50	5,51	7,73	8,56	5,88	6,29	8,67	9,76	6,49	7,59	1	6	6,47	6,47	8

								10,6		14,5						12,0	12,2	12,2					12,4	
8,95	9,10	9,11	3,81	8,94	9,60	9,11	8,38	8	9,69	3	8,62	5,43	5,36	7,62	8,82	1	9	8	4,05	6,48	5,90	6,48	3	6,47	6,48
					16,5			12,1		14,6	10,1				10,6	12,0	12,2	10,1							
8,94	9,72	9,11	3,81	8,94	9	9,10	8,17	0	9,69	5	7	5,37	8,57	9,71	8	0	0	1	7,57	6,47	9,70	6,49	6,48	6,48	6,48
			21,8		18,5			12,3		14,7	17,3		17,5				12,1	10,1							
8,94	9,08	9,12	4	8,93	5	9,71	8,30	5	9,72	2	7	5,53	8	9,72	8,75	7,18	1	2	6,47	6,48	9,05	6,48	6,48	6,47	6,48
			17,6		22,1			12,4		14,9							12,0	10,0							
8,94	9,09	9,70	1	6,87	9	9,11	8,22	4	5,84	4	8,62	5,43	5,52	9,73	7,92	6,92	7	8	6,48	6,48	9,59	6,48	6,49	6,48	6,48
			14,0	10,3				13,2		15,7				12,3			12,0	10,0							
8,96	8,94	9,11	3	4	4,97	9,11	8,23	4	5,73	2	5,39	5,35	5,25	7	9,81	8,50	3	8	6,48	6,48	9,05	9,01	6,48	6,49	6,48
				10,0										12,3			12,0	10,0							
8,95	8,94	8,93	9,10	9	4,98	9,10	8,19	5,92	5,79	5,93	9,60	5,38	5,47	8	9,75	8,04	1	8	6,48	6,48	6,48	9,00	6,48	6,48	6,48

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Günther

Vorname: Alexander

dass ich die vorliegende Bachelorarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Konzeption eines Messsystems für die kombinierte Vermessung des Fahrzeugumfelds und der Mobilfunknetze

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ist erfolgt durch:

21244 Buchhol i.d.N.	19.09.2023	[redacted]
Ort	Datum	Unterschrift im Original