HOCHSCHULE COBURG

Hochschule für angewandte Wissenschaften Coburg

Fakultät Elektrotechnik und Informatik

Studiengang: Informatik

Bachelorarbeit

Detektion der Bewegung von Verkehrsteilnehmern aus Positionsdaten

Recep Furgan Torlak

Abgabe der Arbeit: 25. März 2022

Betreut durch:

Prof. Dr. Thomas Wieland, Hochschule Coburg

Zusammenfassung

Das Ziel dieser Forschung ist es, die Detektion der Bewegung von Verkehrsteilnehmern aus Positionsdaten zu ermitteln. Dazu wird die folgende Forschungsfrage gestellt: "Kann aus den Positionsdaten von Verkehrsteilnehmern die Bewegung ermittelt und der Typ mit Hilfe eines Klassifizierers bestimmt werden?"

Um die Forschungsfrage zu beantworten, wird eine Schnittstelle implementiert, welche mit der Verkehrssimulation CARLA die Bewegung der verschiedenen Personen und Fahrzeugen errechnet. Die berechneten Daten werden im Anschluss mit Hilfe eines Klassifizierers ausgewertet und die Typen der Verkehrsteilnehmer bestimmt.

Die Ergebnisse zeigen, dass die Detektion der Bewegung anhand von Positionsdaten machbar und eine hohe Erfolgsrate in der Bestimmung der Art des Verkehrsteilnehmers erreicht wird.

Dies zeigt, dass der Einsatz dieses Detektionsverfahrens auch in der realen Welt gut funktionieren und dazu beitragen könnte, andere Teilnehmer im Verkehr besser zu detektieren.

Abstract

The objective of this research is to implement the detection of the movement of road users by position data. For that the following research issue is set: "Is it possible to detect the movement of road users by means of their position data and determine the type of the road users with the help of a classifier?"

To answer the research issue an interface was implemented, which calculates the movement of different people and vehicles with the traffic simulator CARLA. The calculated data is then used to determine the types of the traffic users by their calculated data by means of a classifier.

The results of the calculations and classifications demonstrate that the detection of the movement with the help of position data is possible and achieves a high accuracy of the classification. This shows, that the application of this detection process is also viable for the real world and could contribute to a better detection of road users.

Inhaltsverzeichnis

Abbildungsverzeichnis

Codebeispielverzeichnis

Symbolverzeichnis

Symbol	Bedeutung	[phys. Einheit]
$\bar{a}$	Durchschnittliche Beschleunigung	[m²/s]
$\Delta\theta$	Winkeldifferenz	[°]
Δs	Streckendifferenz	[m]
Δt	Zeitdifferenz	[s]
Δv	Geschwindigkeitsdifferenz	[m/s]
$\vec{v}$	Vektor	
ω	Winkelgeschwindigkeit	[°/sec]

Abkürzungsverzeichnis

API	Application Programming Interface
CARLA	Car Learning to Act
CSS	Cascading Style Sheets
CSV	Comma-separated values
EGNOS	European Geostationary Navigation Overlay Service
GLONASS	Global Orbiting Navigation Satellite System
GNSS	Global Navigation Satellite System
HTML	Hyper Text Markup Language
JSON	JavaScript Object Notation
km/h	Kilometres per hour
LED	Light-emitting diode
Lidar	Light Detection and Ranging
Lkw	Lastkraftwagen
PDOP	Position Dilution of Precision
Radar	Radio Detection and Ranging

1 Einleitung

Immer wieder kommen Menschen ums Leben, die in einem Unfall mit Kraftfahrzeugen verwickelt sind. Einer der Hauptgründe hierfür ist das Fehlverhalten der Autofahrer. In der Automobilbranche wird immer mehr Zeit und Geld in die Forschung investiert, um Verkehrsteilnehmer besser zu detektieren, Unfälle zu vermeiden und den Verkehr sicherer zu machen. Im Hauptfokus für dieses Ziele befindet sich das autonome Fahren. Vom Spurhalteassistenten bis hin zum vollautomatischen Fahrzeug, das autonome Fahren hat verschiedene Stufen in denen all dies möglich ist. Die Detektion von Verkehrsteilnehmern ist bereits im Großteil aller neuen Fahrzeuge mit eingebaut und hilft den Autofahrern in täglichen Szenarien. Laut einer Umfrage im Jahr 2021 sind nur 33% aller Befragten davon überzeugt, dass selbstfahrende Auto gut sind und dadurch Unfälle und Verkehrsverstöße reduziert werden. [VISION+2021] Um genau diese Erwartung zu erfüllen und mehr Menschen zu überzeugen ist es notwendig, die Detektion von Fußgängern und Radfahrern so sehr zu verbessern, dass ein Fehlverhalten weitgehend unmöglich ist. Ein Fehler in der Detektion könnte schwerwiegende Folgen haben bis hin zum Tod von Verkehrsteilnehmern.

1.1 Motivation

Beim autonomen Fahren spielt die Detektion der Verkehrsteilnehmer eine wichtige Rolle. In modernen Lösungen werden Technologien wie Radar, LiDAR oder Stereokameras verwendet, um Verkehrsteilnehmer zu detektieren, die entsprechende Aktionen durchzuführen und ein unfallfreies Fahrverhalten zu bewerkstelligen.

Jede dieser Technologien hat ihre Vor- und Nachteile, weshalb das alleinige Nutzen einer der Varianten nicht ausreichend ist. Bei Implementierungen mit einer oder mehreren Kameras ist eine der größten Schwachstellen das Bild, da bei schlechter Beleuchtung eine schwächere Performanz bei Stereokameras (Zwei spezielle Kameras, die das menschliche Sehen nachstellen und mit Hilfe von Strahlen die Tiefe der beobachteten Stellen bestimmen) zu beobachten ist. Anders als bei Stereokameras haben Radar und LiDAR keine Probleme mit der Beleuchtung des Umfelds. Eines der Hauptprobleme von Radar (Detektion von Objekten mit Hilfe von elektromagnetischen Wellen) ist die Erkennung von mehreren Objekten, da nicht unterschieden werden kann, ob es sich um ein einzelnes oder mehrere Objekte handelt. [RADAR] LiDAR (Berührungsloses Messen mit Lasern) hingegen funktioniert sehr schlecht bis gar nicht

bei bestimmten Wetterkonditionen (z.B. Schnee, Regen, Nebel oder bewölkter Himmel). [LiDAR] Wie genau die genannten und weiteren Technologien funktionieren und welche Vor- und Nachteile diese haben, wird detailliert im 2. Kapitel wiedergegeben. Durch die Detektion von Verkehrsteilnehmern anhand von GNSS Daten soll die bisherige Verkehrsteilnehmererkennung unterstützt oder sogar ersetzt und dadurch verbessert werden.

1.2 Aufgabenstellung

Das Ziel dieser Bachelorarbeit ist es, GNSS Daten von Verkehrsteilnehmern in der Nähe anzufragen und zu verarbeiten. Anhand der Position und dem jeweiligen Zeitstempel werden die Daten der Verkehrsteilnehmer errechnet. Nachdem die Daten errechnet wurden, kann dann anhand eines Klassifizierer bestimmt werden, um welche Art von Verkehrsteilnehmer es sich handelt, was nicht Teil dieser Arbeit ist. Anhand der ermittelten Art und den anderen Daten kann eine entsprechende Reaktion, sofern nötig, ausgeführt werden, um einen eventuellen Unfall zu vermeiden und zu einem sicheren Verkehr beizutragen. Stand 2021 gibt es etwa 62,6 Millionen Smartphone-Nutzer in Deutschland. [Tenzer+2022] Laut dem Statistischen Bundesamt lebten in Deutschland im Jahr 2021 ca. 83,2 Millionen Menschen. [Population] Umgerechnet haben somit rund 75% oder 3 von 4 aller Menschen in Deutschland ein Smartphone in der Tasche wobei fast jedes Smartphone von Haus aus GNSS-Fähig ist. Diese Besonderheit trägt dazu bei, dass mit Hilfe der Smartphones ein Großteil aller Verkehrsteilnehmer erkannt werden kann, sofern das Smartphone mitgenommen wird. Die meisten modernen Fahrzeuge haben ebenfalls ein GNSS-fähiges Steuergerät eingebaut, wodurch die GNSS Daten davon auch als Ersatz dienen können, falls der Autofahrer selbst kein GNSS-fähiges Gerät mit sich trägt.

2 Allgemeine Grundlagen

Im folgenden Kapitel werden die Frameworks Angular und Ionic zur Erstellung von mobilen Anwendungen beschrieben. Des Weiteren werden einige Techniken wie Stereosehen, Radar und LiDAR im Detail erklärt, die beim autonomen Fahren verwendet werden. Unter anderem wird die Verkehrssimulation CARLA und die von ihr zur Verfügung gestellten Features erläutert. Die zum Erfassen der Positionsdaten verwendete Technik GNSS wird ebenfalls genauer beschrieben, um einen besseren Einblick zu verschaffen. Zum Ende wird das mathematische Verfahren der Interpolation und Extrapolation klargelegt.

2.1 Ionic

Das Open-Source Toolkit Ionic eignet sich zum Erstellen von performanten und qualitativen Mobiltelefon- und Computeranwendungen. Die Anwendungen werden anhand von Webtechnologien wie HTML, CSS und JavaScript entwickelt, wobei auch beliebte Frameworks wie Angular, React und Vue dabei unterstützt werden. [Ionic]

Ionic setzt den Schwerpunkt auf die Entwicklung der Oberfläche der Anwendung. Die Entwicklung kann anhand eines einfachen Skriptes oder mit Hilfe von Bibliotheken oder Frameworks erfolgen. Die entwickelte Codebasis ist vielseitig verwendbar und kann für verschiedene Plattformen verwendet werden. [Ionic]

2.2 Angular

Angular ist eine auf die Programmiersprache TypeScript basierende Entwicklungsplattform. Um skalierbare Webanwendungen zu entwickeln, wird ein Komponenten-basiertes Framework angeboten. Des Weiteren bietet Angular eine Sammlung von Bibliotheken mit einer Vielfalt an Features. Zu den Features gehören unter anderem Routing, Formularmanagement, Client-Server Kommunikation und viele mehr. Es werden auch Entwicklerwerkzeuge zur Verfügung gestellt, um die Entwicklung aufzubauen, zu testen und den Code zu aktualisieren. [Angular]

2.3 Stereosehen

Stereosehen ist eine Technik des Maschinellen Sehens bei welcher zwei oder mehre Stereokameras verwendet werden, um eine vollständige 3D Ansicht bieten zu können. Das Fundament des Stereosehen ähnelt der 3D Ansicht des Menschlichen Sehens und basiert auf Triangulierung von Strahlen auf mehrere Sichtpunkte. [Stereo Vision]

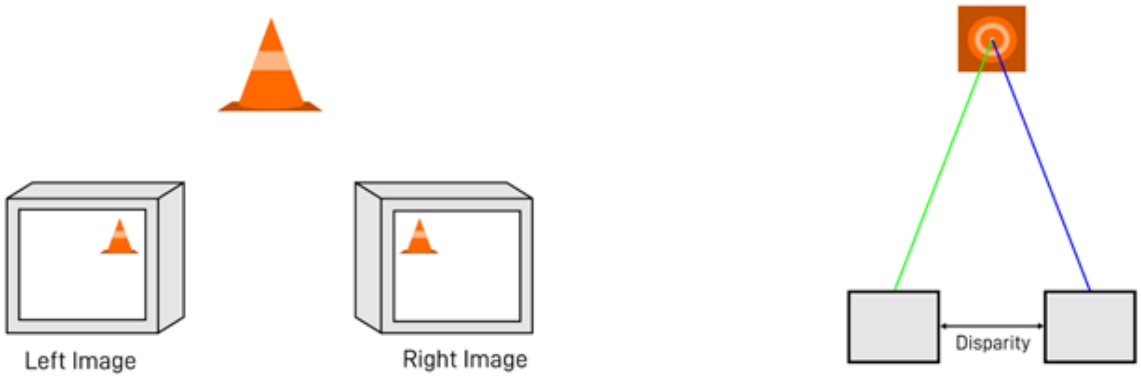

Abbildung 1: Stereokamera Konzept

Quelle: [Stereo Vision]

Eine Stereokamera kopiert das Verhalten der menschlichen Augen sehr genau, um eine akkurate Tiefenwahrnehmung in Echtzeit zu ermöglichen. Die festgelegte Distanz, in der die beiden Sensoren voneinander aufgestellt sind, ermöglicht es ähnliche Pixel von beiden 2D Ebenen zu Triangulieren (Abbildung 1 rechts). [Stereo Vision]

Jedes Pixel in einem Bild einer Digitalkamera sammelt Licht, welches die Kamera über einen 3D Strahl erreicht. Wenn ein Merkmal in der Welt als eine Pixelstelle in einem Bild identifiziert werden kann, kann festgestellt werden, dass dieses Merkmal auf dem 3D Strahl liegt, der mit dem Pixel assoziiert ist. Wenn mehrere Kameras verwendet werden, erhöht sich die Anzahl der zu erhaltenden Strahlen. Die Stelle, an der sich die Strahlen überschneiden, sagt den 3D Standort eines Objektes und dessen Merkmale voraus. [Stereo Vision]

Der Hauptvorteil der Stereokamera besteht darin, dass keine Laser oder LEDs für die Detektion benötigt werden. Außerdem ist eine Stereokamera deutlich günstiger im Vergleich zu anderen Technologien des Maschinellen Sehens in 3D. Die Detektion von weit entfernten und

bewegenden Objekten ist sehr gut. Die Tiefenerkennung funktioniert nach einer Kalibrierung sogar in Echtzeit. [Stereo Vision]

Die Nachteile der Stereokamera zeichnen sich dadurch aus, dass das Ganze sehr rechenaufwendig sein kann. Die Abtastung von texturlosen oder schlecht beleuchteten Oberflächen führt zu schlechteren Ergebnissen. Die Beleuchtung kann unter anderem durch bestimmte Wetterverhältnisse beeinträchtigt werden. [Stereo Vision]

2.4 Radar

Radar (Radio Detection and Ranging) ist ein Verfahren zur Entdeckung und Positionsbestimmung von festen und bewegten Objekten mit Hilfe elektromagnetischer Wellen. [Spektrum]

Die elektromagnetischen Wellen werden von einem Radargerät gebündelt und als Primärsignal ausgesendet. Anschließend werden die von Objekten reflektierten „Echos“ als Sekundärsignal empfangen und nach verschiedenen Kriterien ausgewertet. Aus den Echos können verschiedene Informationen gewonnen werden wie z.B. der Winkel bzw. die Richtung zum Objekt. Auch die Entfernung zum Objekt lässt ich aus der Zeitverschiebung zwischen dem Senden und Empfangen des Signals ermitteln. Des Weiteren kann die Relativbewegung zwischen Sender und Objekt durch den Doppler-Effekt aus der Verschiebung der Frequenz des reflektierten Signals berechnet werden. Auch die Wegstrecke kann aus der Aneinanderreihung einzelner Messungen (Pulsen) und der Absolutgeschwindigkeit des Objektes bestimmt werden. Bei guter Auflösung des Radars können Konturen des Objektes erkannt werden (z.B. der Flugzeugtyp) oder sogar Bilder gewonnen werden (Erd- und Planetenerkennung). [Fraunhofer]

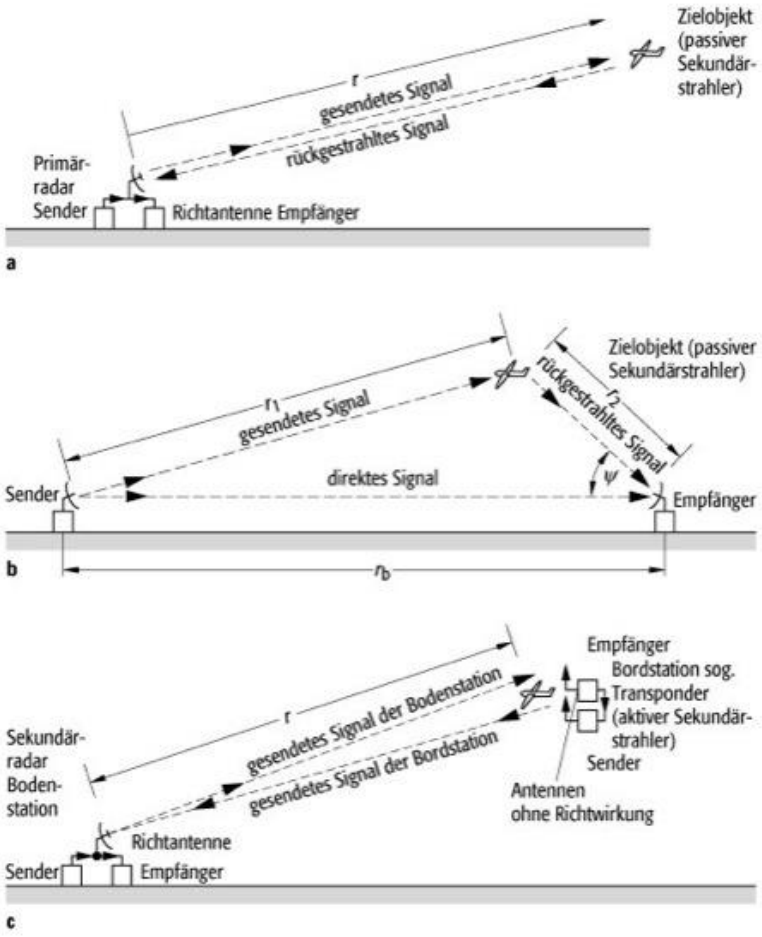

Abbildung 2: Radararten und Konzepte

Quelle: [Spektrum]

Die Funktionalität des Primärradars basiert auf das Aussenden von Sendeimpulsen, welche vom Ziel reflektiert und dann vom Radargerät empfangen werden. Die reflektierten Sendeimpulse werden vom Ziel selbst erzeugt und haben dieselbe Frequenz wie die empfangenen Sendeimpulse. Die zurückgesendeten Sendeimpulse werden als passive Echos bezeichnet und enthalten Informationen über die Richtung, Höhe und Entfernung der Ziele. [Wolff SSR PSR]

Das bistatische Radarverfahren ist eine Variante des Primärradar. Im Vergleich zum Primärradar, bei welchem die gleiche Antenne zum Senden und Empfangen genutzt wird, besteht das bistatische Radargerät aus einer Senderstellung und einer oder mehreren Empfängerstellungen. [Wolff Bistatisch] Der Sender sendet ein Signal aus, welches vom (Haupt-)Empfänger direkt empfangen wird. Zudem wird das Signal auch von einem oder mehreren

(Hilfs-)Empfängern reflektiert, wovon die reflektierten Strahlen dann wiederum vom (Haupt-)Empfänger erfasst werden.

Anders als beim Primärradar, arbeiten Sekundärradare mit aktiven Antwortsignalen. Nachdem die Sendeimpulse als Abfragen ausgesendet werden, werden diese von einem Antwortgerät empfangen und verarbeitet. Die Antwort enthält nebst den Informationen über die Richtung, Höhe und Entfernung der Ziele auch noch die Kennung und Identifizierung der Ziele. [Wolff SSR PSR]

Die Vorteile in der Radar Technologie bestehen unter anderem darin, die Position, Entfernung und Geschwindigkeit eines Objektes genaustens ermitteln zu können. Des Weiteren können mehrere einzelne Ziele gleichzeitig anvisiert und erkannt werden. Isolatoren wie Gummi oder Plastik werden ohne Probleme durchdrungen und können Objekte auch hinter Materien dieser Art detektieren. Die 3D Abbildung von Objekten durch die Detektion aus verschiedenen Winkeln ist ebenfalls möglich.[RADAR]

Einer der Nachteile bei Radaren ist das nicht erkennen von Objekten, die sich hinter leitenden Materialien befinden. Auch sehr langsames Bremsen (~1,61 km/h) wird kaum bis gar nicht erkannt was bei der Detektion von Verkehrsteilnehmern eine wichtige Rolle spielt. Wenn sich mehrere Objekte an einem Punkt befinden, können diese nicht als einzelne Objekte erkannt werden. Signale von anderen Radaren können das eigene Signal unterbrechen, sofern diese nicht anständig ausgerichtet werden.[RADAR]

2.5 LiDAR

LiDAR (Light Detection and Ranging) ist ein Verfahren zur berührungslosen Abstandsmessung durch Laser. Es gibt 3 Varianten der LiDAR Sensoren, die eingeteilt sind in die 1D-, 2D- und 3D Variante. [Weber+2018]

Für die direkte Abstandsmessung werden LiDAR-Sensoren auf ein natürliches Ziel oder einen Reflektor gerichtet. Wie in der Abbildung 3 gezeigt wird, gibt es eine Lichtquelle die als Laser dient und durch eine Linse auf ein Objekt strahlt. Das angestrahlte Objekt reflektiert das Licht, welches dann von einem Lichtsensor empfangen und verarbeitet wird. Sensoren, die auf dieser Basis in einer Dimension (Distanz) arbeiten, sind sogenannte eindimensionale, also 1D-Sensoren. [Weber+2018]

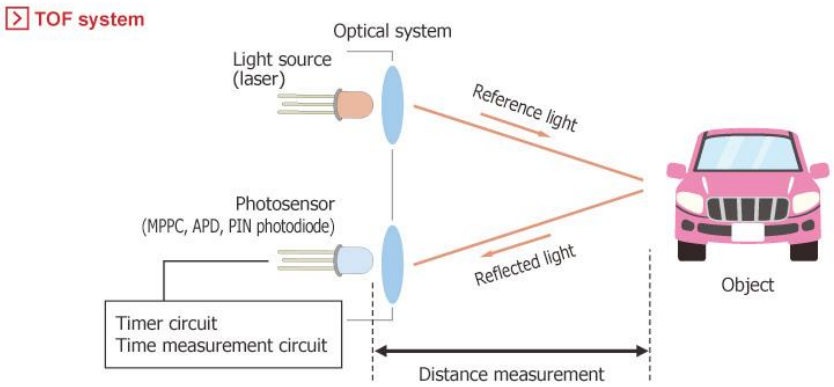

Abbildung 3: LiDAR Konzept

Quelle: [HAMAMATSU]

Bewegt man einen Messstrahl bzw. rotiert diesen in einer Ebene, so erhält man Informationen zum Abstand und zum Winkel, folglich in zwei Dimensionen. Für eine solche Messung eingesetzte Sensoren werden üblicherweise als 2D-Laserscanner oder 2D-LiDAR-Sensoren bezeichnet. Sie ermitteln Messwerte in einer sequenziellen Reihenfolge mit üblicherweise gleichem zeitlichen Abstand der Messungen. [Weber+2018]

Geschwenkte LiDAR-Sensoren arbeiten in der dritten Dimension. Durch das Schwenken werden der Abstand und die Position in x-, y- und z-Richtung ermittelt. Die gleiche Information über die unterschiedlichen Raumparameter erhält man, wenn mehrere Sende- oder Empfangssysteme in unterschiedlichen horizontalen Winkeln in einem Sensor scannend bewegt werden. Diese bezeichnet man heute als Mehrlagenscanner. [Weber+2018]

Der größte Vorteil eines LiDAR Sensor besteht darin, dass dieser in unmittelbarer Zeit sehr große Mengen an Messungen durchführen kann, die bis auf einen Zentimeter genau sind. Die generierten Daten können einfach in 3D Abbildungen umgewandelt werden, um die Umgebung zu interpretieren. LiDAR wird außerdem durch externe Lichtquellen nicht beeinträchtigt und leistet gute Arbeit bei schlechten Lichtverhältnissen. Die ermittelten Daten sind zudem direkt als Messdaten verfügbar und müssen nicht extra entziffert oder interpretiert werden. [LeddarTech]

Die großen Mengen an generierten Daten können auch ein Nachteil sein, wenn die verwendete Hardware mit der Verarbeitung nicht hinterherkommt. Des Weiteren hat LiDAR Probleme mit dem Erkennen von dicht bewachsenen Orten. [Ivankov+2020] Eines der größten Probleme bei LiDAR jedoch ist, dass nur angestrahlten Objekte erkannt werden. Ein Objekt, dass sich hinter einem anderen befindet kann somit nicht detektiert werden, wenn das Objekt davor die ausgestrahlten Lichtstrahlen abfängt.

2.6 CARLA

CARLA (Car Learning to Act) ist ein Open-Source Verkehrssimulator und wurde von Grund auf entwickelt um Entwicklung, Training und Validierung autonomer Fahrsysteme zu unterstützen. Zusätzlich zum Open-Source Code und Protokollen bietet Carla offene digitales Assets (Städtische Layouts, Gebäude und Fahrzeuge), welche für diesen Zweck entwickelt wurden und frei genutzt werden können. Die Simulationsplattform unterstützt flexible Spezifikationen der Sensor Suites, Umweltbedingungen, volle Kontrolle aller statischen und dynamischen Akteure, Kartengenerierung und vieles mehr.[CARLA]

Zur Simulation des autonomen Fahrens werden verschiedene Sensoren eingesetzt, welche ebenfalls in CARLA verfügbar sind. Zu den verfügbaren Sensoren gehören Kollisionsdetektor, Stereokamera, GNSS Sensor, Spurenüberschreitungsdetektion, LiDAR Sensor, Hinderniserkennung, Radar Sensor und viele mehr.

Anhand der von CARLA angebotenen Python API kann während der Laufzeit mit der generierten Welt interagiert werden und die von den Sensoren generierten Daten abgerufen werden. In der Vielfalt der möglichen Aktionen können unter anderem alle Akteure in Laufzeit manipuliert werden. Zu den Akteuren gelten nicht nur Verkehrsteilnehmer, sondern auch andere Elemente der Welt (u.a. Verkehrsschilder, Ampeln und Zuschauer). Die Akteure können über die API abgefragt und manipuliert werden. Die möglichen Aktionen hängen von dem Typen des Akteurs ab und können in der Dokumentation nachgelesen werden.

Mit Hilfe der API können verschiedene Szenarien generiert und das Verhalten von Verkehrsteilnehmern genaustens beobachtet werden. Das fast unbegrenzte Laden von Verkehrsteilnehmern und die freie Bestimmung wie schnell sich diese bewegen und welchen Weg sie gehen macht dies möglich. Durch das Laden eines Zuschauers ist es auch möglich, sich in der

Welt frei zu bewegen, um verschiedene Stellen der Welt einzeln oder von der Vogelperspektive aus alles auf einmal zu beobachten.

2.7 GNSS

GNSS (Global Navigation Satellite System) ist der Überbegriff aller Satelliten-Navigationssysteme. Weit verbreitet ist die Bezeichnung GPS (Global Positioning System) welche der Name des Systems ist, das vom US-amerikanischen Militär entwickelt wurde. Nebst dem US-amerikanischen System gibt es auch GLONASS (Global Orbiting Navigation Satellite System), welches vom russischen Militär genutzt wird. Galileo ist das System der europäischen Union und BeiDou-2 ist das System, welches von China entwickelt wurde. [GNSS+2022]

An einem GNSS-System sind Satelliten im Weltraum, Bodenstationen als Kontroll-Segment, geostationäre Satelliten mit Korrektursignalen und das GPS-Gerät des Benutzers beteiligt. Die Ortung erfolgt durch das laufende Senden der Position und Uhrzeit der Satelliten als codierte Radiosignale zur Erde. Durch die empfangenen Signale ermittelt der Empfänger dann die Entfernung zu allen Satelliten und daraus seine Position auf der Erde. Für die Berechnung wird die Zeit bestimmt, die die Radiowellen vom Satelliten bis zum Empfänger brauchen. Für den gesamten Ablauf benötigt der Empfänger eine Antenne, eine Quarzuhr, etwas Speicher und einen Prozessor zum Rechnen. Für eine eindeutige Standortbestimmung reicht es Signale von mindestens einem von vier Satelliten zu empfangen. Je mehr Satellitensignale empfangen werden, desto zuverlässiger und auch meist genauer wird die Position des Benutzer bestimmt. [GPS]

Heutzutage sind GPS-Empfänger in nahezu jedem mobilen Gerät und vielen modernen Fahrzeugen eingebaut. Diese Standardempfänger liefern Standortgenauigkeiten von ca. 5-15m, was für gängige Anwendungen wie z.B. Navigation ausreichend ist. [GPS]

Alle beteiligten Elemente – die Satelliten, die Atmosphäre, die Erde und die GNSS-Empfänger – sind ständig in Bewegung, was die technische Umsetzung hochkomplex macht. Fehler sind in der Standortbestimmung unausweichlich und werden größtenteils durch Geostationäre Satelliten wie z.B. EGNOS oder von Bodenstationen aus korrigiert und in die Standortbestimmung mit einberechnet. [GPS]

EGNOS (European Geostationary Navigation Overlay Service) ist das erste gesamteuropäische Navigationssystem. Es erweitert das US-amerikanische GPS-Satelliten Navigationssystem und macht es geeignet für sicherheitskritische Anwendungen wie das Fliegen von Flugzeugen oder das Navigieren von Schiffen durch enge Kanäle. [EGNOS]

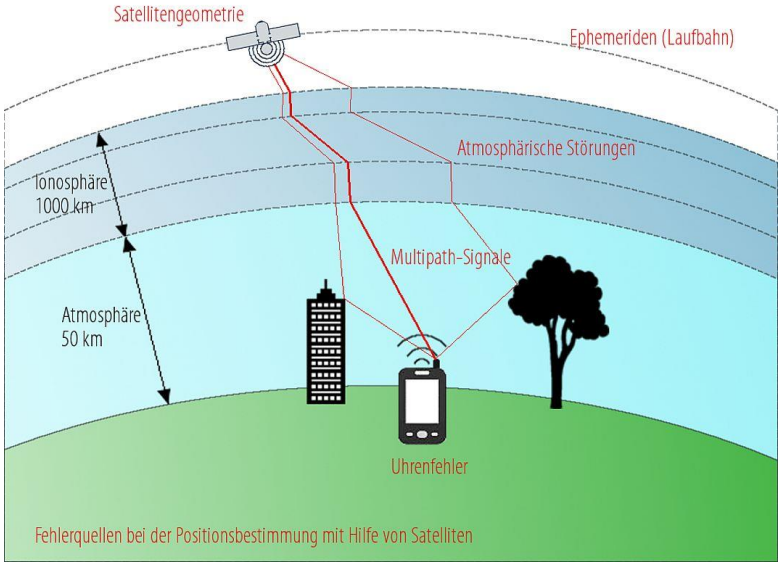

Abbildung 4: GNSS Fehlerquellen

Quelle: [GPS]

Auf dem Weg zum GNSS-Empfänger werden die Positionsdaten durch eine Vielzahl an Fehlerquellen verfälscht wovon eine die Satellitenfehler sind. Die Abbildung 4 hilft dabei die Fehlerquellen zu verbildlichen. Zu Satellitenfehlern gehören Gravitationseffekte (Einfluss auf die hochgenauen Atomuhren, Abweichungen 10-100m), Satellitengeometrie, d.h. die Position der Satelliten zueinander (PDOP, Abweichungen 5-10m) und Ephemeridenfehler (Abweichungen 5-10m – Abweichungen zwischen berechneter und tatsächlicher Satellitenlaufbahn). Eine weitere Fehlerart sind Atmosphärische Fehler (Laufzeitfehler in der Tropo- und Ionosphäre, Abweichungen bis zu 150m). [GPS]

2.8 Interpolation

In den naturwissenschaftlich-technischen Anwendungen stellt sich häufig das Problem, dass von einer unbekannten Funktion $n+1$ Kurvenpunkte bekannt sind. Die Abszissenwerte $x_0, x_1, x_2, \dots, x_n$ werden in diesem Zusammenhang als Stützstellen, ihre zugehörigen Ordina-

tenwerte $y_0, y_1, y_2, \dots, y_n$ als Stützwerte bezeichnet. Gesucht wird dann eine möglichst einfache Ersatz- oder Näherungsfunktion

$$y = f(x) \tag{2.1}$$

die mit der unbekannten Funktion in den $n + 1$ Stützstellen übereinstimmt.[Papula+2001]

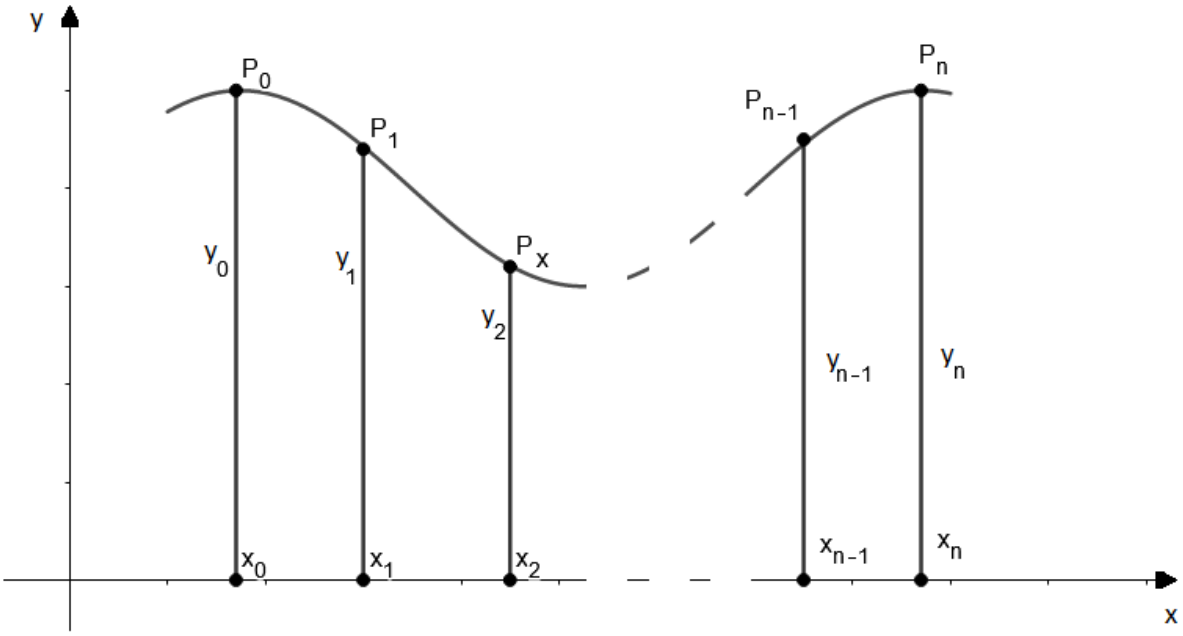

Abbildung 5: Näherungspolynom für eine unbekannte Funktion durch $n + 1$ vorgegebene „Stützpunkte“

Quelle: [Papula+2001]

Anhand des Polynomsatzes lässt sich die Funktion (2.2) leicht gewinnen, welche auch als Interpolationspolynom n-ten Grades bezeichnet wird:

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{2.2}$$

Ein Ansatz zum Bestimmen der Polynomkoeffizienten ist es, der Reihe nach die Koordinaten der $n + 1$ Stützpunkte $P_0, P_1, P_2, \dots, P_n$ in den Polynomansatz einzusetzen. Durch das Einsetzen wird ein lineares Gleichungssystem mit $n + 1$ Gleichungen und $n + 1$ Unbekannten $a_0,\ a_1, a_2, \dots, a_n$ erhalten:

$$a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n = y_0$$

$$a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n = y_1$$

$$a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_n x_2^n = y_2 \quad (2.3)$$

...

$$a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n = y_n$$

Das Gleichungssystem besitzt genau eine Lösung, wenn sämtliche Stützstellen $x_0, x_1, x_2, \dots, x_n$ voneinander verschieden sind. Da der Rechenaufwand beim Lösen aber sehr aufwendig ist, kann auch das Interpolationspolynom von Newton hergenommen werden.[Papula+2001]

Der von Newton stammende Ansatz für ein Interpolationspolynom n-ten Grades lautet:

$$y = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2) + \cdots \quad (2.4)$$
$$\dots + a_n(x - x_0)(x - x_1)(x - x_2) \dots (x - x_{n-1})$$

$x_0, x_1, x_2, \dots, x_n$ sind die Stützstellen der n+1 vorgegebenen Kurvenpunkte (Stützpunkte). Die Koeffizienten werden dann mit der Steigungs- oder dem Differenzenschema berechnet werden.[Papula+2001]

Für dividierte Differenzen 1. Ordnung aus zwei aufeinanderfolgenden Stützpunkten werden folgende Formeln gebildet:

$$[x_0, x_1] = \frac{y_0 - y_1}{x_0 - x_1}$$

$$[x_1, x_2] = \frac{y_1 - y_2}{x_1 - x_2} \quad (2.5)$$

...

Für dividierte Differenzen 2. Ordnung aus drei aufeinanderfolgenden Stützpunkten werden folgende Formeln gebildet:

$$[x_0,\ x_1,\ x_2] = \frac{[x_0,\ x_1] - [x_1,\ x_2]}{x_0 - x_2}$$

$$[x_1,\ x_2,\ x_3] = \frac{[x_1, x_2] - [x_2, x_3]}{x_1 - x_3} \quad (2.6)$$

...

Für dividierte Differenzen 3. Ordnung aus drei aufeinanderfolgenden Stützpunkten werden folgende Formeln gebildet:

$$[x_0, x_1, x_2, x_3] = \frac{[x_0, x_1, x_2] - [x_1, x_2, x_3]}{x_0 - x_3}$$

$$[x_1, x_2, x_3, x_4] = \frac{[x_1, x_2, x_3] - [x_2, x_3, x_4]}{x_1 - x_4} \tag{2.7}$$

...

Entsprechend werden die dividierten Differenzen höherer Ordnung gebildet.[Papula+2001]

3 Anforderungen und Gesamtkonzept

Das folgende Kapitel beschreibt die Anforderungen die für diese Bachelorarbeit und das Gesamtkonzept der Arbeit.

Als Simulationsumgebung wird das Open-Source Projekt CARLA verwendet, worin verschiedene Szenarien generiert werden, die städtische Umgebungen mit Verkehr darstellen. Die als Verkehrsteilnehmer benutzten Modelle bestehen aus Fahrzeugen, Motorrädern, Fahrrädern, Fußgängern und Lastkraftfahrzeugen, die den Verkehr beleben. Zur Ermittlung der Position wird die von CARLA angebotene GNSS Implementierung genutzt. Mit Hilfe der ermittelten Positionsdaten werden zu Beginn Bewegungsdaten sowie Geschwindigkeit und Ausrichtung berechnet. Die Bewegungsdaten werden einem externen Klassifizierer zur Verfügung gestellt, sodass dieser anhand der Daten bestimmt, um welche Art von Verkehrsteilnehmer es sich handelt. Zur Verbesserung der Klassifizierung werden im Verlauf der Arbeit auch weitere Bewegungsdaten wie die Winkelgeschwindigkeit und Beschleunigung hinzugefügt. Einer der Verkehrsteilnehmer wird als Held zugewiesen, um Bewegungsdaten anderer Verkehrsteilnehmer relativ zum Helden zu berechnen. Dazu gehören der Abstand und die relative Position zum Helden. Die Positionsdaten werden im späteren Verlauf mit einer realistischen Abweichung verfälscht, um die GNSS Implementierung realitätsnah zu entwickeln.

4 Design und Architektur

Im folgenden Kapitel werden die einzelnen Etappen beschrieben, wie vorgegangen wurde und welche Erkenntnisse in den entsprechenden Schritten gewonnen wurden. Zu jedem Unterkapitel gibt es eine Beschreibung gefolgt von Diagrammen, die die Implementierung veranschaulichen. Zum Ende wird dann eine kurze Evaluation zu den Etappen gemacht, um die Erkenntnisse zusammenzufassen.

4.1 Mobile Applikation zur Erfassung von GNSS Daten

Eine mobile Anwendung für das Android Betriebssystem ermöglicht es, GNSS Daten für ein bestimmtes Gerät zu sammeln. Die Anwendung frägt GNSS Daten an und schreibt diese auf eine lokale Datei des Geräts, auf dem die Anwendung ausgeführt wird. Nach dem Befahren einer festgelegten Route kann die geschriebene Datei entnommen und analysiert werden, um festzustellen, wie stark die GNSS Daten in einer realen Anwendung abweichen.

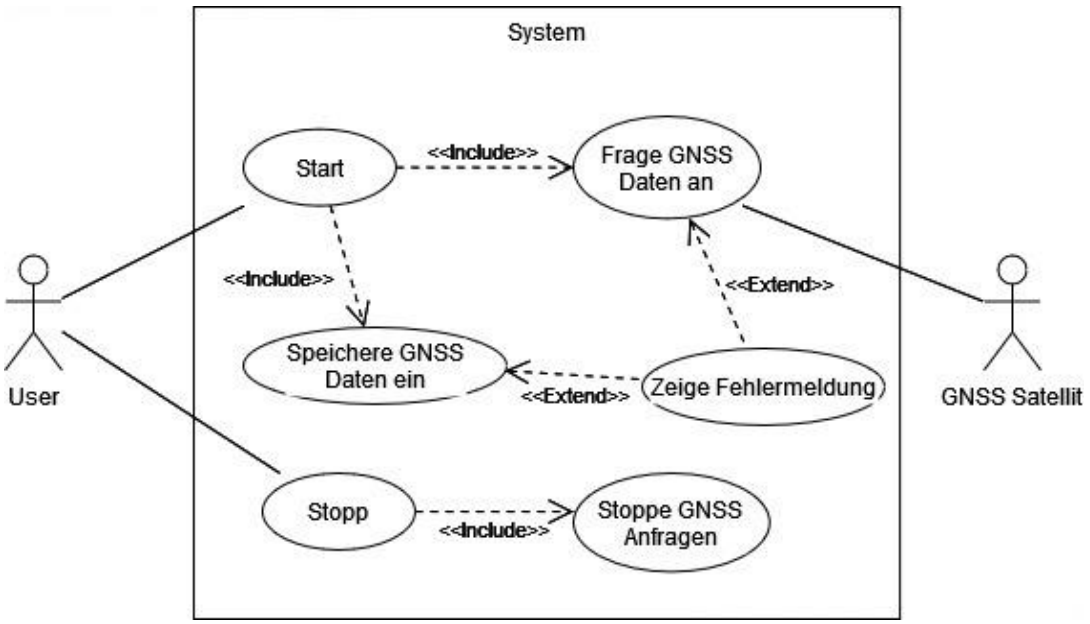

Abbildung 6: Use-Case Diagramm GNSS Applikation

Quelle: [Eigene Darstellung]

Wie aus der Abbildung 6 zu erkennen ist, hat der Nutzer die Möglichkeit das Programm zu starten, um die Positionsdaten anzufragen. Die Anfragen für die GNSS Daten können auch gestoppt werden, um keine weiteren Anfragen mehr auszusenden. Im Falle eines Fehler wäh-

rend der Positionsanfrage oder während des Schreibens auf die Datei wird eine Fehlermeldung angezeigt. Die Fehlermeldung ermöglicht es dem User zu signalisieren, dass ein Problem aufgetreten ist.

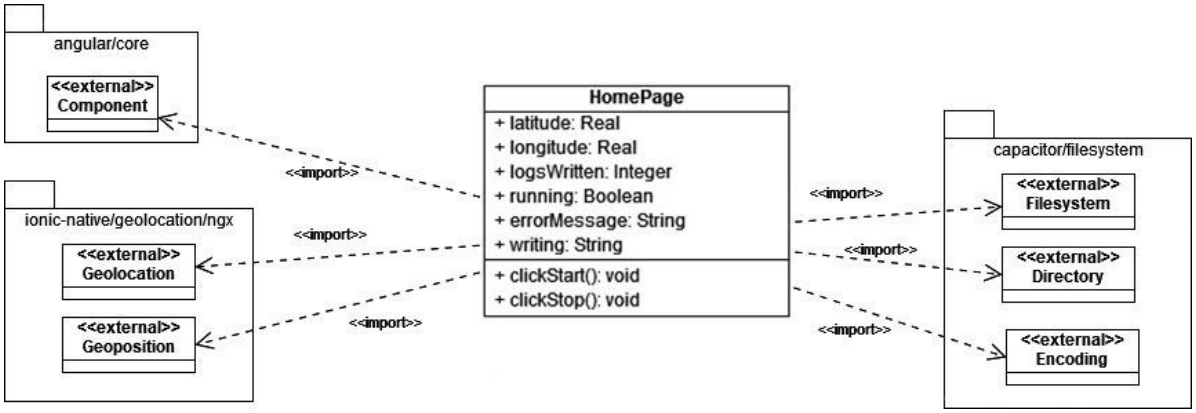

Abbildung 7: Klassendiagramm GNSS Applikation

Quelle: [Eigene Darstellung]

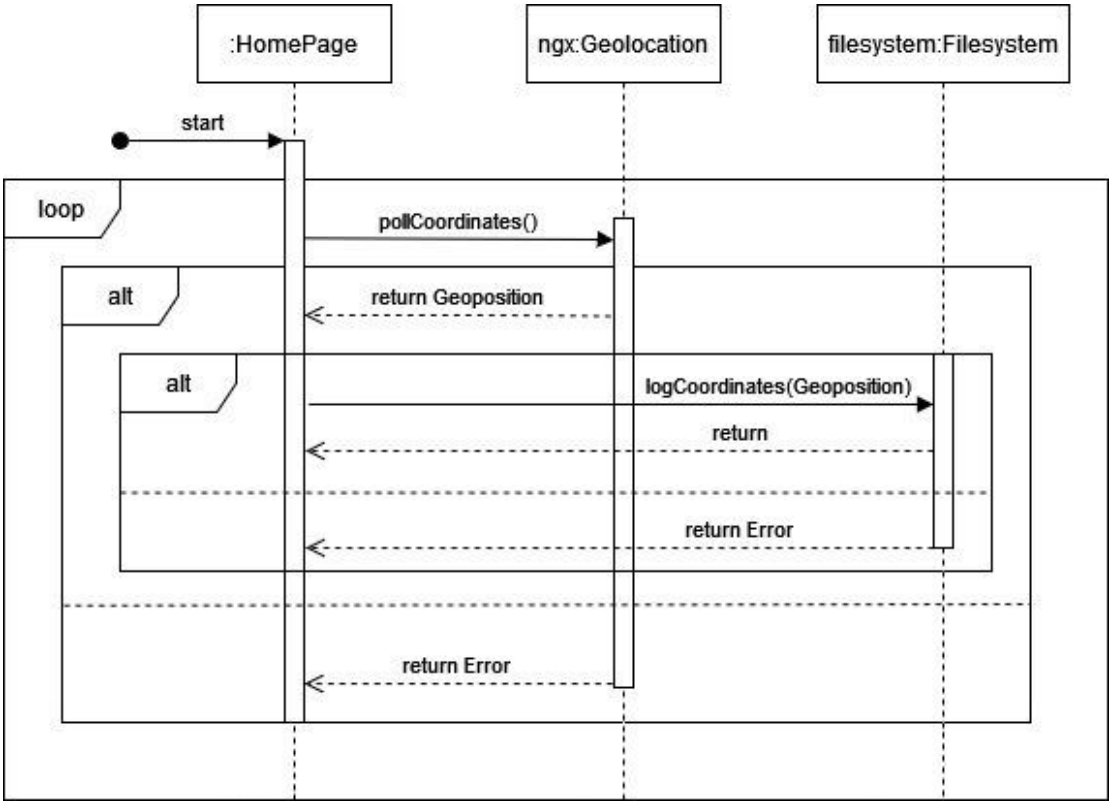

Abbildung 8: Sequenzdiagramm GNSS Applikation

Quelle: [Eigene Darstellung]

Durch das Starten der des Prozesses werden die Koordinaten von einer externen Quelle so lange angefragt, bist die Schleife unterbrochen wird. Die Anfragen werden asynchron ausgesendet, weshalb eine neue Anfrage immer erst dann ausgesendet wird, wenn die vorherige Anfrage durch Daten oder eine Fehlermeldung beantwortet worden ist.

4.2 Schnittstelle

Mit Hilfe von CARLA wird eine Welt generiert, in welcher verschiedene Arten von Verkehrsteilnehmern geladen werden. Für die Interaktion mit der Welt wird eine separate Schnittstelle entwickelt, welche erst einmal die Grundfunktionen bietet, um die aus der Simulation erarbeiteten Daten zu exportieren.

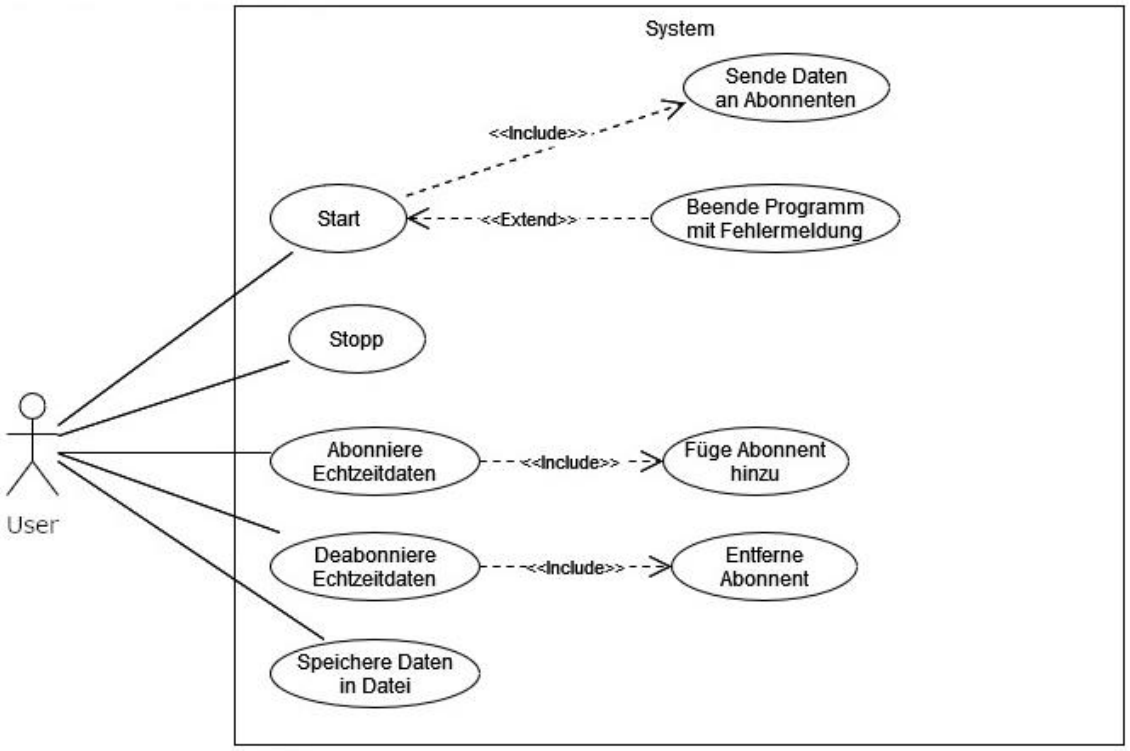

Abbildung 9: Use-Case Diagramm Schnittstelle

Quelle: [Eigene Darstellung]

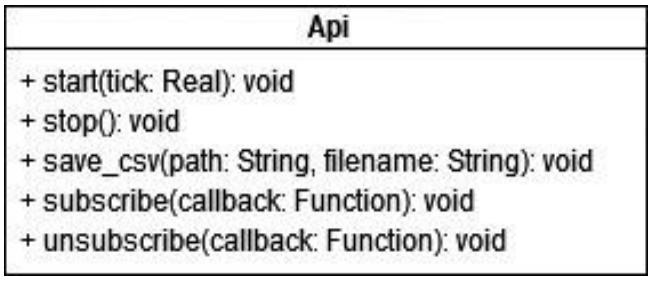

Abbildung 10: Klassendiagramm Schnittstelle

Quelle: [Eigene Darstellung]

Das in Abbildung 9 dargestellte Use-Case Diagramm stellt die möglichen Interaktionen des Nutzers mit dem System dar. In der Abbildung 10 wird durch ein Klassendiagramm verdeutlicht, wie die Klasse aussehen soll. Der Nutzer hat die Möglichkeit das Programm zu starten und somit den Prozess der Positionsabfrage und Datengenerierung anzustoßen. Im Falle eines Fehlers bei der Verbindung mit der CARLA Instanz wird das Programm mit einer Fehlermeldung beendet. Das Programm kann ebenfalls zu beliebiger Zeit gestoppt werden und somit auch die Datenabfrage stoppen. Durch ein Abonnement der Echtzeitdaten können die Daten, die aus der Positionsabfrage errechnet wurden, während der Laufzeit empfangen werden. Eine Abbestellung der Echtzeitdaten ist jederzeit möglich und stoppt den Datenfluss für den jeweiligen Abonnenten. Des Weiteren besteht auch die Möglichkeit die generierten Daten während oder nach Ablauf des Programms in eine CSV Datei zu speichern.

4.3 Positionsermittlung mit GNSS Empfänger

Die Positionen der Verkehrsteilnehmer können anhand von GNSS Sensoren ermittelt werden, welche von CARLA zur Verfügung gestellt werden. Für jeden Verkehrsteilnehmer, der sich in der Simulation befindet, wird ein GNSS Sensor erstellt und an jeweils einen Verkehrsteilnehmer gebunden. Einer der Verkehrsteilnehmer übernimmt die Rolle des Helden, auf welchem die Anwendung ausgeführt wird. Die Positionsdaten werden in bestimmten Zeitabständen ermittelt und an den Helden weitergeleitet, sodass dieser die Daten verarbeiten kann.

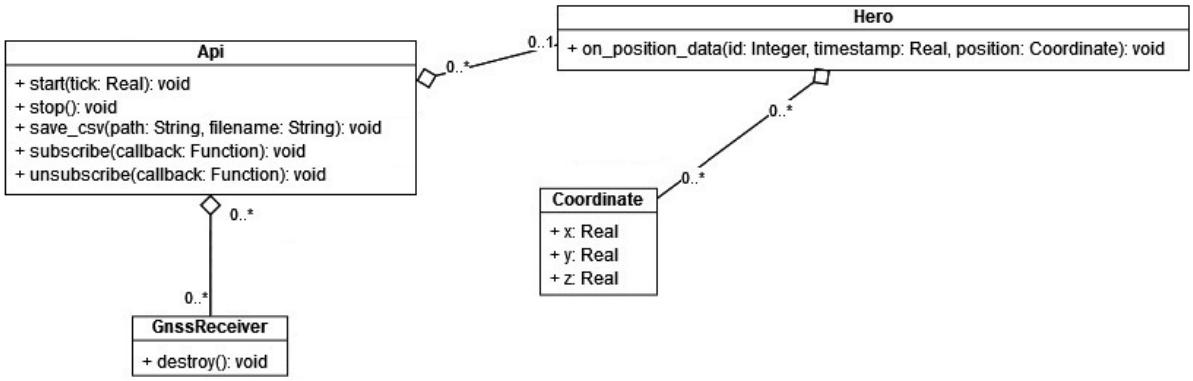

Abbildung 11: Klassendiagramm mit GNSS Empfänger

Quelle: [Eigene Darstellung]

Die Schnittstelle bietet die Möglichkeit, die Positionsanfragen in einem bestimmten Zeitintervall zu starten und bei Bedarf zu stoppen. Des Weiteren werden GNSS Empfänger an alle in der Simulation vorhandenen Verkehrsteilnehmer gebunden. Mit der vorgegebenen Rückruffunktion werden die Daten asynchron an den Helden weitergesendet, sobald die vorgegebene Zeit erreicht ist. Die empfangenen Positionsdaten werden als Geographische Koordinaten empfangen und in 3D Koordinaten umgewandelt.

4.4 Positionsermittlung ohne GNSS Empfänger

Die Positionsermittlung mit den von CARLA zur Verfügung gestellten GNSS Sensoren hat während den Testdurchläufen viele fehlerhafte Daten aufgewiesen. Die Testdurchläufe verliefen auf ebenen Straßen, auf welchen sich die Verkehrsteilnehmer in der xy-Ebene bewegen. Die Daten von den GNSS Sensoren wiesen Änderung auf den x- und y-Koordinaten auf, jedoch gab es auch Änderungen in den z-Koordinaten. Nach genauerem Überprüfen der Koordinaten haben diese viele Fehler aufgewiesen, auch wenn keine Abweichung für die Koordinaten eingestellt wurde. Aufgrund dessen wurde ein alternativer Ansatz entwickelt, mit dem die GNSS Koordinaten der Verkehrsteilnehmer von CARLA über die Python API direkt abgefragt werden. Wie in Abbildung 12 im Klassendiagramm zu erkennen ist, fällt bei diesem Ansatz die `GnssReceiver` Klasse weg, da diese dazu dient, die GNSS Sensoren zu initialisieren und die Positionsdaten weiterzuleiten. Weitere Änderungen werden im Sequenzdiagramm in Abbildung 13 beschrieben.

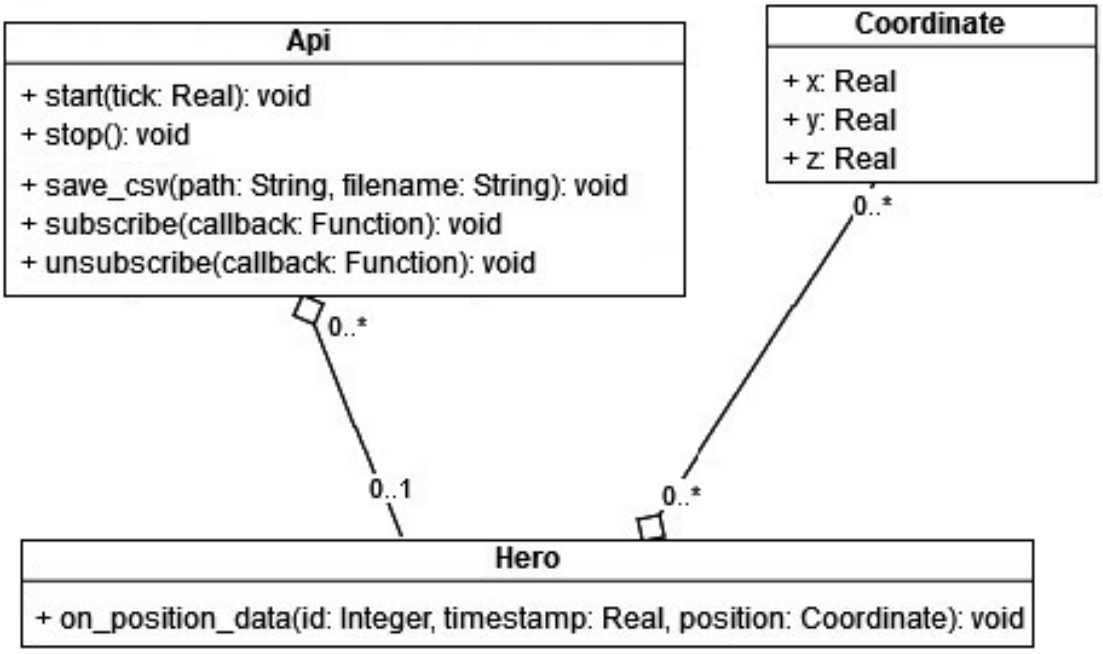

Abbildung 12: Klassendiagramm ohne GNSS Empfänger

Quelle: [Eigene Darstellung]

Da der GNSS Sensor komplett wegfällt, wird in der Abbildung 13 beschrieben, wie die Positionsdaten stattdessen ermittelt werden. Anders als mit dem GNSS Sensor, geschieht die Positionsanfrage in der Api. Die Position wird mit Hilfe der Python Schnittstelle von den jeweiligen Vekehrsteilnehmern direkt angefragt. In einer Schleife wird dabei durch die Liste aller in der Simulation vorhandenen Verkehrsteilnehmer iteriert und nacheinander die Position abgefragt. Da kein Zeitstempel für die Position mitgeliefert wird, wird der Zeitstempel hergenommen, zu dem die Positionsdaten ankommen. Um das Zeitintervall für die Positionsanfragen einzuhalten, wird die Zeit am Anfang der Schleife festgehalten und gestoppt, sobald die Positionen aller Akteuere erhalten wurden. Falls die für die Anfragen benötigte Zeit das vorgegebene Zeitintervall nicht erreicht hat, wird die restliche Zeit noch abgewartet, bevor die Schleife erneut durchlaufen wird. Die Positionsabfrage erfolgt in einem parallelen Prozess weshalb die die Variable der Abbruchbedinugng der Schleife jederzeit geändert und die Positionsabfrage gestoppt werden kann.

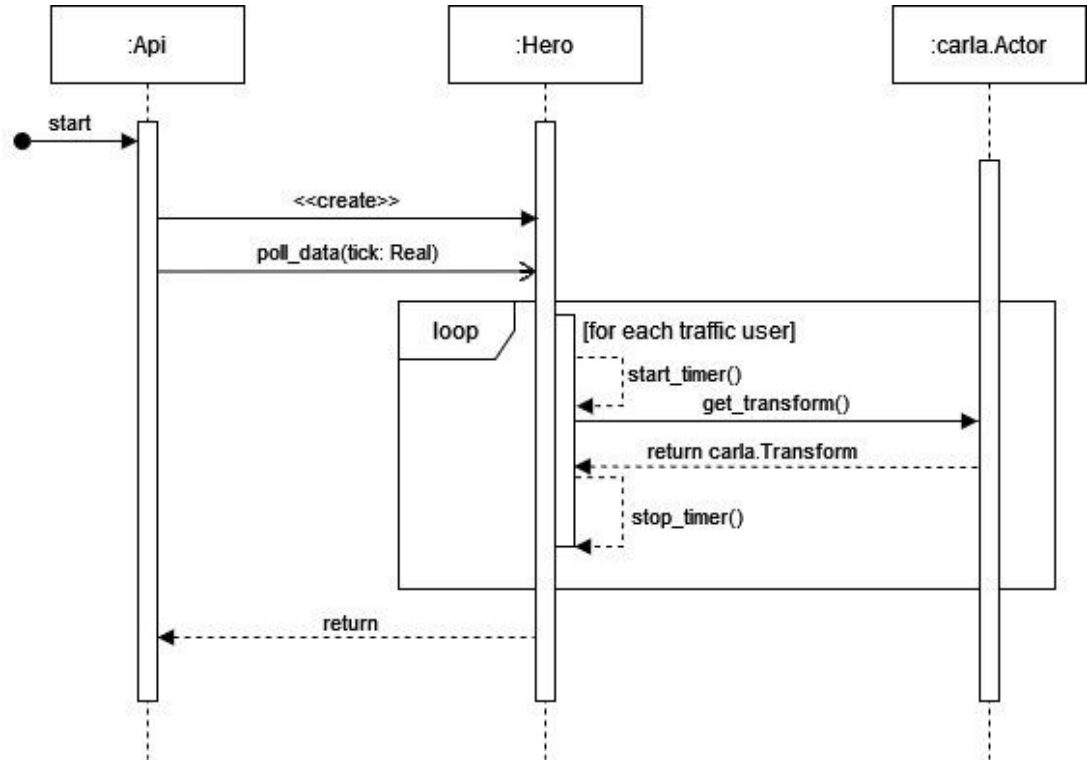

Abbildung 13: Sequenzdiagramm ohne GNSS Empfänger

Quelle: [Eigene Darstellung]

4.5 Allgemeine Bewegungsdaten

Für die allgemeinen Bewegungsdaten der Verkehrsteilnehmer wird die Geschwindigkeit und Ausrichtung ermittelt. Anhand der Positionsdaten an zu zwei Zeitpunkten, die sehr nah aneinander liegen, wird die Ausrichtung und Geschwindigkeit ausgerechnet.

```
velocity(new_position, new_timestamp) {
      save_current_data_as_previous();
      save_new_data_as_current();
      Δs = Vector(current_position, previous_position).length();
      Δt = current_timestamp – previous_timestamp;
      return Δs / Δt;
}
```

Code 1: Pseudocode zur Berechnung der Geschwindigkeit

Der Pseudocode in Code 1 zeigt, wie die Geschwindigkeit ermittelt wird. Zur Vereinfachung wird von einer geradlinig gleichförmigen Bewegung ausgegangen. Die zuvor als aktuell gespeicherten Daten (Position und Zeitstempel) werden dafür als vorherige und die neu erfassten als aktuelle Daten gesichert. Nachdem ein Vektor zwischen den zwei Positionen aufgespannt wurde, ergibt sich aus der Länge dieses Vektors die gefahrene Strecke. Der Quotient der gefahrenen Strecke und der Differenz der Zeitstempel ergibt somit die durchschnittliche Geschwindigkeit.

```
orientation(new_position) {
     save_current_data_as_previous();
     save_new_data_as_current();
     direction = Vector(current_position, previous_position);
     return angle_between_vectors(direction, y_axis_vector);
}
```

Code 2: Pseudocode zur Berechnung der Ausrichtung

Ähnlich wie bei der Geschwindigkeit, wird der Vektor zwischen der vorherigen und aktuellen Position aufgespannt. Der aufgespannte Vektor wird dann zusammen mit einem Vektor entlang der y-Achse verwendet, um den dazwischen liegenden Winkel zu berechnen und daraus eine Aussage über die Ausrichtung zu treffen.

4.6 Daten relativ zum Helden

Ein wichtiger Punkt bei der Forschung ist es, das Programm auf einem der Verkehrsteilnehmer auszuführen und nicht nur Bewegungsdaten über die anderen sich im Verkehr befindenden zu sammeln. Das Fahrzeug, welches die Anwendung ausführt, wird als Held bezeichnet und benötigt weitere Informationen, um über Kriterien wie z.B. der Relevanz für sich zu bestimmen. Die zwei wichtigsten Daten hierfür sind die Entfernung zum Helden und die relative Position zum Helden.

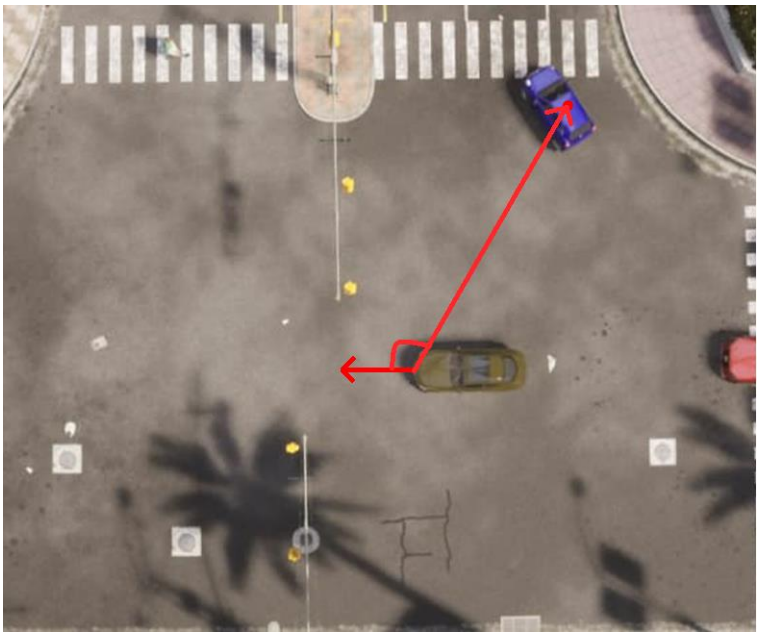

Abbildung 14: Relative Position zum Helden eingezeichnet auf einem Bildschirmfoto der CARLA Simulation

Quelle: [Eigene Darstellung]

Die Entfernung zum Helden lässt sich durch die aktuelle Position des Helden und des anderen Verkehrsteilnehmers bestimmen. Dazu wird ein Vektor zwischen den zwei Punkten aufgespannt und im Anschluss die Länge des Vektors berechnet. Die relative Position zum Helden hingegen erfordert etwas mehr Aufwand. Wie in der Abbildung 14 dargestellt wird, wird die relative Position anhand von drei Punkten bestimmt, wovon zwei dem Helden angehören und einer dem Verkehrsteilnehmer, dessen relative Position berechnet werden soll. Einer der zwei Punkte ist die vorherige Position des Helden und der zweite die aktuelle Position. Anhand dieser zwei Punkte kann die aktuelle Ausrichtung des Helden bestimmt werden. Die aktuelle Position des anderen ist der dritte Punkt, der für die Berechnung benötigt wird. Nebst der Ausrichtung wird ein weiterer Vektor zwischen der aktuellen Position des Helden und der aktuellen Position des anderen Verkehrsteilnehmers aufgespannt. Anhand des Vektors der Ausrichtung und des Vektors, der den Abstand beschreibt, wird der Winkel zwischen diesen beiden berechnet. Da der ausgerechnete Winkel mit der vorherigen Position berechnet wurde, muss der Winkel angepasst werden, was in der Implementation genauer beschrieben wird.

4.7 Lückenhafte Datensätze

Der Abstand zum Helden und die Position relativ zum Helden sind Informationen, für die die nötigen Daten vom selben Zeitpunkt sein müssen. Es ist so gut wie unmöglich, dass die Positionsdaten für genau den Zeitpunkt zu erhalten sind, wie sie beim Helden zur Verfügung stehen. Um die Position des jeweiligen Verkehrsteilnehmers zu ermitteln, müssen daher die Positionen für die bestimmten Zeitpunkte erraten werden. Für diesen Zweck stehen die mathematischen Operationen der Interpolation und Extrapolation zur Verfügung. Durch das Sammeln der Positionen aller Verkehrsteilnehmer, können diese gesammelten Daten für die Inter-/Extrapolation genutzt werden. Wenn sich der Zeitpunkt für die gesuchte Position innerhalb des Zeitraum der gespeicherten Daten befindet, wird die Interpolation angewendet. Sollte sich der Zeitpunkt außerhalb des Zeitraums der gespeicherten Daten befinden, wir die Extrapolation angewendet.

```
get_position(timestamp) {
     if (timestamp < data.smallest_timestamp or
     timestamp > biggest_timestamp) {
          return interpolate(timestamp, data);
     else {
          return extrapolate(timestamp, data);
     }
}
```

Code 3: Pseudocode zur Inter-/Extrapolation für fehlende Datensätze

4.8 Datensatz erweitern: Winkelgeschwindigkeit

Eines der wichtigsten Merkmale, um Verkehrsteilnehmer zu klassifizieren, ist die Winkelgeschwindigkeit. Die Winkelgeschwindigkeit beschreibt, wie schnell die Ausrichtung in einer bestimmten Zeit geändert wurde. Menschen, Fahrräder, Lkws und Autos wechseln ihre Rich-

tung alle unterschiedlich schnell. Diese Information kann eine große Hilfe dabei sein, verschiedene Arten von Verkehrsteilnehmern zu unterscheiden.

Zur Berechnung der Winkelgeschwindigkeit wird folgende Formel verwendet:

$$\omega = \frac{\Delta\theta}{\Delta t} \tag{4.1}$$

Das $\Delta\theta$ beschreibt in dieser Formel die Differenz der beiden Ausrichtungen als Winkel. Der Ansatz, die Winkelgeschwindigkeit programmiertechnisch umzusetzen, lässt sich in folgendem Pseudocode darstellen:

```
angular_velocity(new_orientation, new_timestamp) {
    save_current_data_as_previous();
    save_new_data_as_current();
    Δθ = current_orientation – previous_orientation;
    Δt = current_timestamp – previous_timestamp;
    return Δθ / Δt;
}
```

Code 4: Pseudocode zur Berechnung der Winkelgeschwindigkeit

4.9 Verfälschte Daten

Die mit GNSS ermittelten Positionen sind in der Praxis nicht so Fehlerfrei, wie sie von der Simulation angegeben werden. Um die Positionsdaten der Realität näher zu bringen, werden die Positionen mit einer zufällig gewählten Abweichung verfälscht. Wie groß die Abweichung maximal werden kann, hängt von vielen Faktoren ab und ist daher frei wählbar.

Der erste Ansatz die Daten zu verfälschen war, jede einzelne Komponente der Koordinaten mit der vorgegebenen Abweichung zu verrechnen. Bei einem einfachen Vektor $\vec{v} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ und einer Abweichung von 10, wäre der am stärksten verfälschte Vektor $\vec{v}' = \begin{pmatrix} 11 \\ 10 \\ 10 \end{pmatrix}$. Nach Berechnung der Längen der beiden Vektoren anhand der Formel

$$|\vec{v}| = \sqrt{x^2 + y^2 + z^2} \tag{4.2}$$

erschließt sich daraus, dass $|\vec{v}| = 1$ und $|\vec{v}'| = \sim 17.92$ ist. In den Dokumentationen zur Beschreibung von GNSS wird immer von einer Abweichung der Position geschrieben, aber nicht speziell von einer Abweichung der einzelnen Komponenten der Position. Daraus erschließt sich, dass der erste Ansatz zur Verfälschung der Koordinaten im 3-dimensionalen Raum ungeeignet ist, da die verfälschte Koordinate weiter entfernt sein kann als das vorgegebene Maß.

Da Straßen in den betrachteten Szenarien relativ eben sind, werden zur Vereinfachung nur die x- und y-Koordinaten verfälscht und die z-Koordinaten in ursprünglicher Form gelassen. Die Länge des verfälschten Vektors $\vec{v}' = \begin{pmatrix} 11 \\ 10 \\ 0 \end{pmatrix}$ beträgt somit $|\vec{v}'| = \sim 14.87$ und ist weiterhin weit über dem vorgegebenen Maß.

Der zweite Ansatz zur Berechnung für das Verfälschen der Koordinaten ist es, weiterhin im 2-dimensionalen zu bleiben, diesmal jedoch einen Kreis zur Berechnung zu verwenden. Der Kreis hilft dabei, die möglichen Positionen der verfälschen Koordinaten abzubilden.

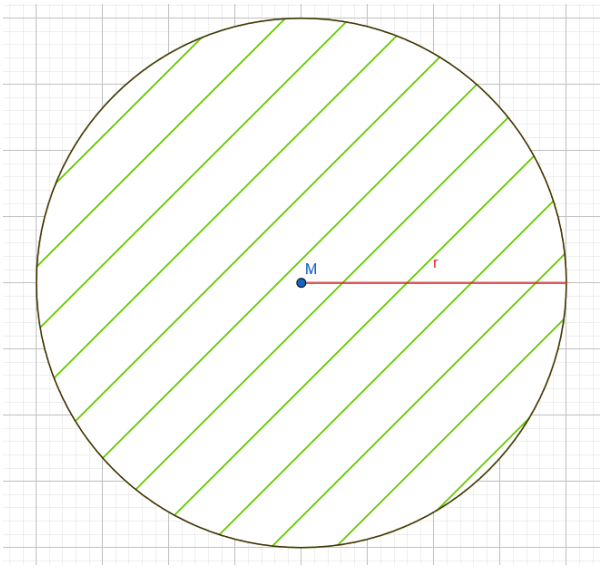

Abbildung 15: Kreis mit möglichen Abweichungswerten

Quelle: [Eigene Darstellung]

Mit Hilfe der Hauptform der Kreisgleichung können Werte eingesetzt und ein Kreis aufgezeichnet werden:

$$r^2 = (x - x_0)^2 + (y - y_0)^2 \tag{4.3}$$

y_0 = y-Koordinate des zu verfälschenden Punktes

x_0 = x-Koordinate des zu verfälschenden Punktes

r = Maximale Abweichung

Die verfälschten Punkte können nur im schraffierten grünen Bereich oder auf dem Rand des Kreises liegen. Der Mittelpunkt *M* des Kreises stellt die zu verfälschenden Koordinaten dar und der Radius *r* die maximale Abweichung. Um eine verfälschte Koordinate zu erhalten, wird aus der gleichverteilten Menge 0 bis *r* eine zufällige Zahl entnommen, welche als die x-Komponente der verfälschten Koordinate verwendet wird.

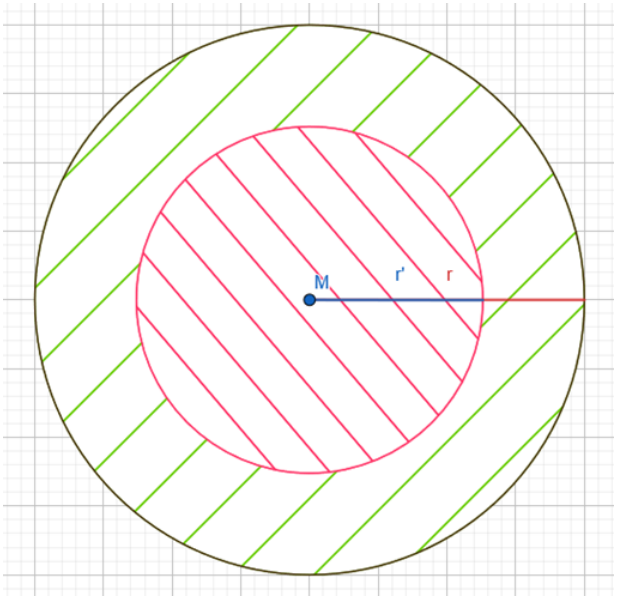

Abbildung 16: Kreis mit eingeschränkten möglichen Abweichungswerten

Quelle: [Eigene Darstellung]

Der Radius *r'* des neu definierte Kreises beträgt *x*, für den gilt: $0 \leq r' \leq r$. Durch das Gleichsetzen der Gleichung werden ein bzw. zwei Punkte auf der Kante des Kreises ermittelt. Die Gleichung lautet nach dem Gleichsetzen:

$$y_{1,2} = \pm\sqrt{r'^2 - (x - x_0)^2} + y_0 \tag{4.4}$$

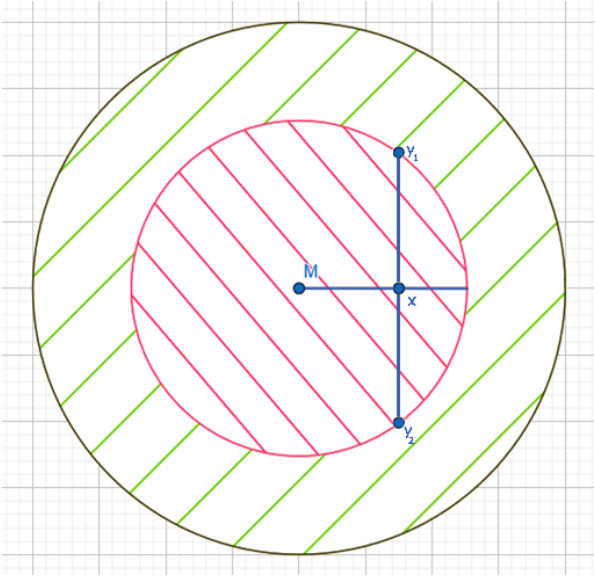

Abbildung 17: Kreis mit zufällig gewählten Abweichungspunkten

Quelle: [Eigene Darstellung]

Durch eine zufällige Wahl kann eines der beiden errechneten *y* als der verfälschte Punkt ermittelt und somit den realen Punkt ersetzen.

4.10 Datensatz erweitern: Beschleunigung

Um die Klassifizierung weiter zu verbessern, wird auch die Beschleunigung in den Datensatz mit aufgenommen. Die Beschleunigung verschiedener Arten von Verkehrsteilnehmern unterscheidet sich stark, weshalb diese eine große Rolle bei der Klassifizierung spielt.

```
accelaration(new_velocity, new_timestamp) {
      save_current_data_as_previous();
      save_new_data_as_current();
      Δv = current_velocity – previous_velocity;
      Δt = current_timestamp – previous_timestamp;
      return Δv / Δt;
}
```

Code 5: Pseudocode zur Berechnung der Beschleunigung

Die Beschleunigung wird anhand der neu ermittelten Geschwindigkeit und dem Zeitpunkt berechnet, zu dem der Verkehrsteilnehmer sich mit der Geschwindigkeit bewegt. Die zuvor gespeicherten Daten werden hierfür überschrieben und aktualisiert. Die Beschleunigung wird im Anschluss mit Hilfe der Formel

$$\bar{a} = \frac{v - v_0}{t} = \frac{\Delta v}{\Delta t} \tag{4.5}$$

berechnet, wodurch man die Durchschnittliche Beschleunigung erhält.

5 Implementierung

5.1 Mobile Applikation zur Erfassung von GNSS Daten

Die mobile Anwendung wurde mit Hilfe des Frameworks Ionic für Smartphones mit dem Betriebssystem Android entwickelt. Ionic bietet verschiedene Module an, um z.B. GNSS Daten zu ermitteln (`ionic-native/geolocation/ngx`) oder Dateien auf dem Gerät zu öffnen und zu bearbeiten (`capacitor/filesystem`). Das Framework Angular wurde zusammen mit den Auszeichnungssprachen HTML und CSS und der Programmiersprache TypeScript dafür verwendet, die Benutzeroberfläche zu gestalten und das Verhalten zu implementieren.

Abbildung 18: Bildschirmfoto der mobilen Applikation

Quelle: [Eigene Darstellung]

Die Klasse `HomePage` beinhaltet zwei `public` Methoden `_ng_clickStart` und `_ng_clickStop`. Die Methode `_ng_clickStart` ist mit dem Klicken des START-Button verbunden und frägt rekursiv die GNSS Daten mit Hilfe des eingebundenen Moduls

an und speichert diese in einen Zwischenspeicher. Sobald der STOP-Button betätigt wird, wird die `_ng_clickStop` Methode ausgeführt, wodurch die Anfragen für die GNSS Daten gestoppt werden. Im Anschluss werden die zwischengespeicherten Daten Zeile für Zeile in eine Datei auf dem Gerät gespeichert. Die START- und STOP-Buttons deaktivieren sich bei der Betätigung selbst und aktvieren den anderen Knopf. Im Bildschirm werden während der Laufzeit die Anzahl der erfassten GNSS Daten angezeigt mit den aktuellen Positionsdaten darunter. Im Falle eines Fehlers wird die entsprechende Fehlermeldung über der Anzahl der bisher erfassten GNSS Daten eingeblendet, um über den entsprechenden Fehler zu warnen. Fehler bei der Positionsermittlung werden angezeigt, unterbrechen jedoch nicht den Programmlauf.

5.2 Schnittstelle

Die `Api` Klasse hat als Übergabeparameter die Host-Adresse und den Port, über den die CARLA Simulation erreicht wird. Bei der Initialisierung der Instanz wird versucht, dass der Client eine Verbindung zur Simulation herstellt. Wenn die Verbindung aus irgendeinem Grund nicht erfolgreich sein sollte, wird das Programm mit einer Fehlermeldung beendet. Die Methode `start` ermöglicht es das Hauptprogramm zu starten, sowie die Abfrage der Positionsdaten und die Berechnung der Bewegungsdaten anhand dieser. Mit `stop` wird das Hauptprogramm gestoppt und keine weiteren Anfragen für die GNSS Daten ausgesendet. Um die errechneten Daten zu exportieren, wird die Methode `save_csv` angeboten, welche einen Pfad und einen Dateinamen als Argumente fordert, um die Daten in eine CSV Datei zu schreiben und zu speichern. Die Daten können auch in Echtzeit erhalten und analysiert werden. Dazu kann die `subscribe` Methode verwendet werden, welche eine Funktion als Argument erwartet und diese mit den errechneten Daten aufruft, sobald diese verfügbar sind. Falls keine Daten mehr empfangen werden sollen, kann mit der `unsubscribe` Methode die entsprechende Funktion übergeben und der Datenfluss gestoppt werden.

5.3 Positionsermittlung mit GNSS Empfänger

Der GNSS Empfänger wird in der `GNSS Receiver` Klasse als `carla.GnssSensor` eingebunden. Zur Initialisierung wird ein neuer Akteur in die CARLA Simulation geladen, welcher an den gewünschten Verkehrsteilnehmer (Akteur) gebunden wird. Zum Schluss wird ei-

ne Methode von der `Hero` Klasse an den GNSS Empfänger gebunden, welche die GNSS Daten bei der Ankunft verarbeitet. Die ankommenden Daten enthalten die Längen- und Breitengrade, sowie die Höhe und den Zeitstempel. Die Positionsdaten werden in 3D-Koordinaten umgewandelt und zusammen mit dem Zeitstempel und der ID des Akteurs an den Helden weitergeleitet. Dies erfolgt mittels der öffentlichen `on_position_data` Methode der Hero Klasse. Das Empfangen der Positionsdaten erfolgt asynchron und kann jederzeit durch das Entfernen des Empfängers mit Hilfe der `destroy` Methode beendet werden.

Der Held empfängt die Daten und speichert diese zum einen ab, zum anderen wandelt er die Daten ins JSON Format um und reicht sie an alle Methoden weiter, die sich als Abonnenten in der `Api` Klasse mittels der `subscribe` Methode eingeschrieben haben.

5.4 Positionsermittlung ohne GNSS Empfänger

Die GNSS Daten werden in einem neuen Thread in der `Api` Klasse ermittelt, sodass die Koordinaten weiterhin asynchron angefragt werden. Das Starten der Anfragen der GNSS Daten erfolgt über die `start` Methode, welche einen Thread erstellt, in dem die private Methode `_poll_data` mit dem Argument `tick` eingebunden wird. Das Argument `tick` beschreibt, wie viel Zeit mindestens vergehen soll, bevor die Koordinaten aller bekannten Verkehrsteilnehmer erneut angefragt werden. Falls die Abfrage der Position der Akteure vor der vorgegebenen Zeit fertig ist, wird die restliche Zeit über ein aktives Warten mit Hilfe der `sleep` Methode aus dem `time` Modul verstrichen. Die Abfrage der Positionsdaten erfolgt hierfür in einer Schleife, die so lange läuft, bis die private Variable `_stop` wahr wird. Da die Schleife in einem separaten Prozess läuft, kann die `_stop` Variable jederzeit überschrieben und die Schleife gestoppt werden. Ähnlich wie bei der Positionsermittlung mit dem GNSS Empfänger, wird nach der Ermittlung der Position die private Methode `_on_position_data` des Helden aufgerufen, der dann die Daten verarbeitet und weiterleitet.

5.5 Allgemeine Bewegungsdaten

Ein Verkehrsteilnehmer wird in die Simulierte Welt geladen und seine Position in einem vorgegebenen Zeitintervall ermittelt. Mit Hilfe des von CARLA zur Verfügung gestellten GNSS Sensoren kann ein GNSS Sensor an den Verkehrsteilnehmer gebunden werden. Durch eine hinzugefügte Funktion werden die ermittelten Daten an diese weitergeleitet und können ver-

arbeitet werden. Die übermittelten Daten beinhalten einen Zeitstempel, die Höhe und den Längen- und Breitengrad. Der Breiten- und Längengrad sowie die Höhe werden 1:1 in x, y und z Koordinaten als Position übernommen und zusammen mit dem Zeitstempel gespeichert.

Die zuletzt ermittelte Position wird zusammen mit dem entsprechenden Zeitstempel nochmals separat gespeichert und mit jeder neu ermittelten Position ersetzt, sodass die Positionen von dem vorherigen Zeitpunkt und dem aktuellen Zeitpunkt immer parat sind.

Bei einzelnen Verkehrsteilnehmern wird die Ausrichtung errechnet, welche der Winkel zwischen dem aktuellen Richtungsvektor und einem Vektor ist, der die y-Achse repräsentiert. Die Position zum aktuellen Zeitpunkt wird als die Spitze s und der vorherigen Position als die Basis b für den aktuellen Richtungsvektor $\vec{v}$ verwendet. Der Vektor, der die y-Achse repräsentiert, beträgt $\vec{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$.

Der Winkel zwischen dem Vektor $\vec{v}$ und der y-Achse kann mit der Formel

$$\cos\theta = \frac{\vec{a} \odot \vec{b}}{|\vec{a}|\,|\vec{b}|} \tag{5.1}$$

oder

$$\sin\theta = \frac{|\vec{a} \times \vec{b}|}{|\vec{a}|\,|\vec{b}|} \tag{5.2}$$

berechnet werden. Der daraus resultierende Winkel liegt im Wertebereich $[0; 180]$, weshalb keine Aussage darüber getroffen werden kann, ob sich der Vektor $\vec{v}$ im 1. oder 2. Quadranten befindet. Durch das Zusammenführen der beiden oben genannten Formeln kann auch der Tangens des Winkels berechnet werden.

$$\tan\theta = \frac{sin\theta}{cos\,\theta} = \frac{|\vec{a} \times \vec{b}|}{\vec{a} \odot \vec{b}} \tag{5.3}$$

Für das Berechnen des Winkels bieten die meisten Programmiersprachen die Funktionen `atan` und `atan2`, wovon `atan` den Winkel zwischen $[-90; 90]$ und `atan2` den Winkel zwischen $[-180; 180]$ liefert. Durch die Verwendung von `atan2` und dem optionalen Aufaddieren von 360 bei negativen Winkeln ist es möglich genaue Aussagen über die Ausrichtung eines Verkehrsteilnehmers zu treffen.

Des Weiteren kann auch die Geschwindigkeit eines Verkehrsteilnehmers anhand der zuletzt gespeicherten und aktuellen Position und der jeweiligen Zeitstempel ermittelt werden. Für die Geschwindigkeit wird die Formel

$$v = \frac{\Delta s}{\Delta t} \tag{5.4}$$

verwendet. Da die für die Formel nötigen Elemente alle parat sind, kann durch das Einsetzen die Geschwindigkeit ausgerechnet werden.

5.6 Daten relativ zum Helden

Zur Berechnung der Entfernung des Verkehrsteilnehmers zum Helden wird ein Vektor zwischen der aktuellen Position des Helden p_h und der Position des Verkehrsteilnehmers p_{tu} zu dem Zeitpunkt mit folgender Formel aufgespannt:

$$\vec{v} = p_h - p_{tu} \tag{5.5}$$

Zum Berechnen des Vektors wird die statische Methode `vector` aus der Klasse `MathOperations` verwendet, die die Differenz der einzelnen Komponenten der Koordinaten berechnet und anschließend als eine `Vector` Klasse zurückgibt. Um die Entfernung aus dem Vektor zu berechnen, wird im Anschluss die Länge des Vektors berechnet:

$$d = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{5.6}$$

Diese Berechnung erfolgt in der statischen Methode `vector_length`, welche ebenfalls Teil der `MathOperations` Klasse ist. Für die relative Position zum Helden werden die vorherige Position, die aktuelle Position des Helden und die aktuelle Position des jeweiligen Verkehrsteilnehmers benötigt. Ein Vektor wird zwischen der aktuellen Position des Helden und der aktuellen Position des jeweiligen Verkehrsteilnehmers aufgespannt. Ein weiterer Vektor zwischen der aktuellen Position und der vorherigen Position des Helden.

Wie auch bei der Berechnung der Ausrichtung, wird bei der Berechnung der relativen Position der Winkel zwischen den zwei Vektoren berechnet:

$$\tan\theta = \frac{|\vec{a}\times\vec{b}|}{\vec{a}\odot\vec{b}} \tag{5.7}$$

5.7 Lückenhafte Datensätze

Fehlende Daten werden anhand der privaten Methode `_predict_position` in der `Hero` Klasse ausgerechnet. Mit Hilfe der eingegebenen ID wird der entsprechende Datensatz der bisherigen Positionen mit Zeitstempeln ermittelt. Anhand des Datensatzes wird jeweils ein Array für die Zeitstempel, die x-Komponente, die y-Komponente und die z-Komponente der Positionen erstellt. Nachdem die Zeitstempel geordnet wurden, wird untersucht, ob der gesuchte Zeitstempel innerhalb des Wertebereich liegt. Falls der Zeitstempel im Wertebereich liegt, wird interpoliert. Wenn nicht, wird extrapoliert. Mit Hilfe des `scipy` Moduls wird die Methode `interpolate` verwendet, um Inter-/Extrapolation durchzuführen. Die Zeitstempel werden mit den einzelnen Arrays zusammen in die Methode `interpolate` eingefügt und damit die einzelnen Komponenten der gewünschten Koordinate ermittelt.

5.8 Datensatz erweitern: Winkelgeschwindigkeit

Die Winkelgeschwindigkeit wird berechnet, sobald neue Positionsdaten ankommen. Mit der `update` Methode der `RecentData` Klasse werden die Bewegungsdaten mit Hilfe der neuen Positionsdaten und Zeitstempeln berechnet und zurückgegeben. Zur Berechnung der Winkelgeschwindigkeit werden die neu berechnete und die vorher gespeicherte Ausrichtung sowie die dazu gehörigen Zeitstempel in der privaten `_get_angular_velocity` Methode verwendet. Falls die Orientierung eines Verkehrsteilnehmers nicht ermittelt werden konnte, weil sich dieser seit Beginn der Detektion noch nicht bewegt hat, wird auch keine Ausrichtungsgeschwindigkeit ausgegeben. Die eigentliche Berechnung der Winkelgeschwindigkeit erfolgt mit Hilfe der statischen `angular_speed` Methode, welche Teil der `MathOperations` Klasse ist.

5.9 Verfälschte Daten

Sobald die GNSS Daten in der `_poll_data` Methode der `Api` Klasse ankommen, werden die echten Koordinaten als `Location` zusammen mit der maximalen Verfälschung als `float` an die statische `distorted_coordinate` Methode der `MathOperations` Klasse übergeben. Die Koordinaten werden mit ihrer jeweiligen Achse und `_real` zwischengespeichert. Nach der Berechnung der verfälschten Koordinaten, werden diese mit ihrer

jeweiligen Achsenbeschriftung und `_fake` zwischengespeichert und als Instanz der `Coordinate` Klasse zurückgegeben.

5.10 Datensatz erweitern: Beschleunigung

Sobald neue Positionsdaten ankommen, wird nach Berechnung der aktuellen Geschwindigkeit auch die Beschleunigung in der `RecentData` Klasse berechnet. Mit Hilfe der privaten `_get_accelaration` Methode wird die vorherige Geschwindigkeit und die aktuelle Geschwindigkeit aus der privaten `Recent` Instanz `_recentVelocity` ermittelt. Nach dem Berechnen der Geschwindigkeitsdifferenz und der Differenz der Zeitstempel wird geprüft, ob die Zeitdifferenz 0 ist, um eine Division durch 0 zu vermeiden. Falls dies nicht der Fall ist, wird der Quotient aus den beiden Differenzen gebildet, daraus die durchschnittliche Beschleunigung errechnet und an die `update` Methode der `RecentData` Klasse zurückgegeben. Die `update` Methode wiederum wird von der `Hero` Klasse aufgerufen und speichert die ermittelten Werte in der `Actor` Instanz des jeweiligen Verkehrsteilnehmers ab.

6 Evaluierung

Beim Test der GNSS Genauigkeit von mobilen Endgeräten wurde eine Route festgelegt und mit dem Fahrrad gefahren. Anhand der entwickelten mobilen Applikation wurden die GNSS Daten in bestimmten Zeitintervallen angefragt und aufgeschrieben. In Abbildung 18 wird auf der rechten Seite die geplante Route und auf der linken Seite die detektierte Route durch die protokollierten Positionen dargestellt.

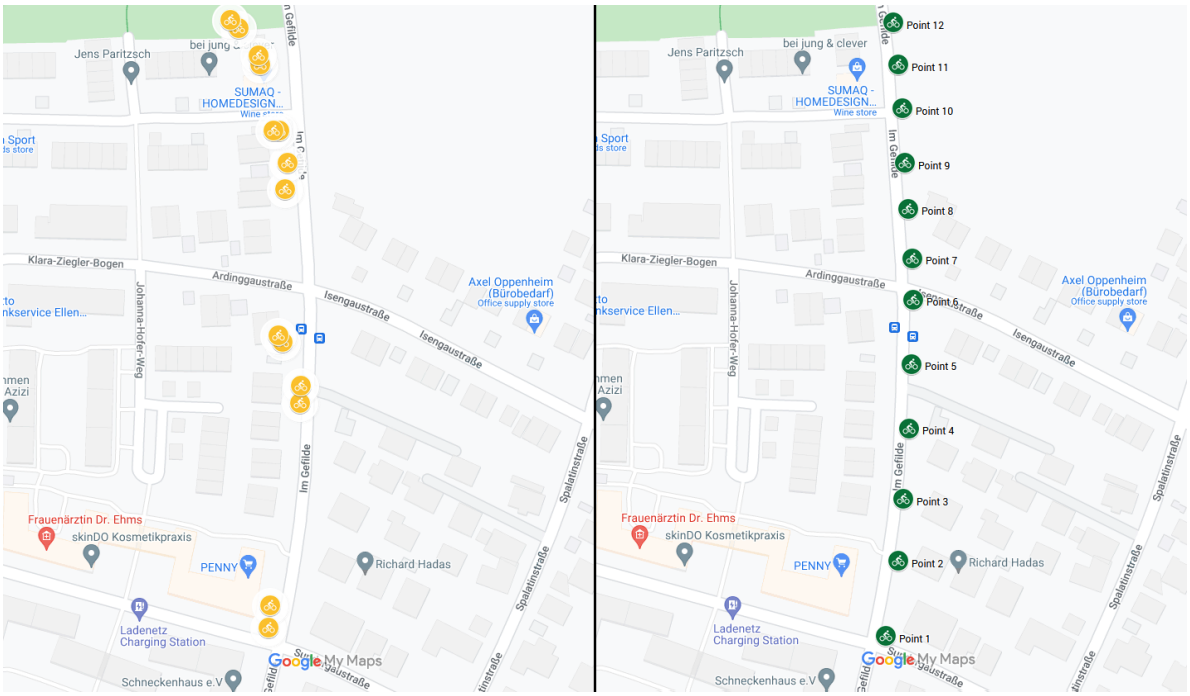

Abbildung 19: Vergleich durch Applikation erfasster und realer GNSS Koordinaten

Quelle: [Maps]

Die ermittelten Positionsdaten weichen sehr stark von den realen Koordinaten ab und haben teilweise sehr große Zeitabstände. Anhand dieser Daten könnte es schwer werden, jegliche Bewegungsdaten zu berechnen, da entweder zu viele Daten fehlen oder die ermittelten Daten zu stark abweichen. Im linken Teil der Abbildung 18 sind die Koordinaten am Ende der Straße am stärksten abgewichen wodurch die Punkte mitten auf der Wiese liegen. Was auch auffällt ist, dass keiner der Punkte auf der rechten Straßenseite detektiert wurde, obwohl die gesamte Strecke auf der rechten Straßenseite befahren wurde.

Mit Hilfe des CARLA Simulators wird eine Umgebung kreiert, in welcher verschiedene Fahrzeugarten und Fußgänger als Verkehrsteilnehmer geladen werden. Als Umgebung werden zwei verschiedene Karten verwendet, die als Town02 und Town03 bezeichnet werden.

Abbildung 20: Bildschirmfoto von Town02

Quelle: [Eigene Darstellung]

Town02 stellt einen kleinen Vorort einer Stadt mit vielen kleinen Häusern und kleineren Waldgebieten dar. Die Geschwindigkeitsbegrenzungen innerorts beschränken sich auf 30 km/h, am Stadtrand auf 60 km/h und am kleinen Waldstück 90km/h. Da die Karte sehr klein ist im Vergleich zu Town03, sind die Straßen auch entsprechend kürzer. Dadurch kann es passieren, dass die Fahrzeuge nicht genug Zeit haben die Maximalgeschwindigkeit zu erreichen, weil sie entweder abbiegen, an einer Ampel warten oder aufgrund einer niedrigeren Geschwindigkeitsbegrenzung abbremsen müssen.

Abbildung 21: Bildschirmfoto von Town03

Quelle: [Eigene Darstellung]

Town03 bietet einen sehr städtischen Anblick mit vielen Hochhäusern und weniger bis gar kein Waldgebiet. Auch die Straßen sind deutlich länger, wodurch die Fahrzeuge mehr Zeit haben die Maximalgeschwindigkeiten von 30, 60 und 90km/h zu erreichen. Anders als bei Town02, hat Town03 auch eine Stelle, an der sich die Steigung der Straße ändert und Verkehrsteilnehmer sich auf der Straße hoch oder runter bewegen. Ein weiteres wichtiges Merkmal von Town03 ist der Kreisverkehr, welcher wertvolle Daten über die Winkelgeschwindigkeit generieren könnte.

Die Höchstgeschwindigkeit in Deutschland beträgt innerorts immer 50km/h, sofern der Bereich nicht als „Tempo-30-Zone“ gekennzeichnet wurde, wodurch die Höchstgeschwindigkeit 30km/h beträgt. [Bußgeld+2022] Aufgrund dessen kann auf diesen Karten das 50km/h Szenario nicht getestet werden, welches für realistische Daten wichtig wäre. Ein weiterer Punkt bei der Simulation ist die Beschleunigung, die bei allen Fahrzeugtypen fast dieselbe ist. Autos und Motorräder fahren genauso langsam an wie Lkws, was in realistischen Verhältnissen so gut wie nie der Fall ist. Auch die von CARLA zur Verfügung gestellten Fußgänger bewegen

sich mit durchschnittlich 6km/h, wodurch Passanten, die laufen oder nicht so schnell gehen ebenfalls nicht simuliert werden.

Die GNSS Sensoren, welche von der Python Api von CARLA angeboten werden, liefern teilweise falsche Daten. Auf einer ebenen Straße ändert sich die z-Komponente der Positionsdaten, obwohl keine Abweichung eingestellt wurde. Mit der alternativen Implementation wurden die Positionsdaten von den Verkehrsteilnehmern direkt angefragt, indem mit ihrer Schnittstelle kommuniziert wurde. Mit dieser Methode wurden die Koordinaten der Verkehrsteilnehmer viel besser erkannt und fehlerfrei ermittelt. Um nicht zu viele parallele Prozesse laufen zu haben, wird nicht für jede einzelne einzelnen ein Prozess erstellt, sondern es wird mit Hilfe einer Schleife durch eine Liste aller Fahrzeuge und Passanten iteriert und alle Positionsdaten abgefragt. Dies lief problemlos und hatte keinerlei Nachteile im Vergleich zur parallelen Verarbeitung.

Die Bewegungsdaten wie die Geschwindigkeit, Ausrichtung, Winkelgeschwindigkeit und Beschleunigung lassen sich einfach berechnen. In der Simulation werden die nötigen Positionsdaten in regelmäßigen Zeitabständen ermittelt. Falls der Verkehrsteilnehmer zu Beginn des Programms an einer Ampel stehen oder sich aus irgendeinem anderen Grund nicht bewegen kann, führt dies zu Problemen bei der Berechnung der Ausrichtung. Da nicht ermittelt werden kann, in welche Richtung sich der Verkehrsteilnehmer bewegt, wird die Ausrichtung als 0 gewertet und zurückgegeben. Für die Bestimmung des Helden wird ein Fahrzeug gewählt, das sich bewegt, d.h. die Geschwindigkeit ist größer als 0. Falls keines der Fahrzeuge zu Beginn des Programms in Bewegung ist, wird das Programm mit einer Fehlermeldung beendet.

Die Daten relativ zum Helden lassen sich ebenfalls gut berechnen. Die Inter- und Extrapolation funktioniert sehr gut und liefert auch entsprechend brauchbare Ergebnisse. Auch bei der relativen Position zum Helden kam es anfangs zu Problemen, wenn sich der Held zum Zeitpunkt der Berechnung nicht bewegt. Da keine aktuelle Ausrichtung des Helden ausgerechnet werden kann, wird daher die letzte ausgerechnete Ausrichtung verwendet und überschrieben, sobald eine neue Ausrichtung berechnet wurde.

Die Verfälschung der Daten liefert vielversprechende Ergebnisse in der vereinfachten Form, sodass nur die x- und y-Komponenten der Koordinaten verfälscht werden. Dass die Fehler aus einer gleichverteilten Menge entnommen werden, hat zu keinerlei Problemen geführt und es ließ sich damit gut arbeiten.

7 Zusammenfassung

Das Ziel dieser Forschung ist es die Detektion der Bewegung von Verkehrsteilnehmern aus Positionsdaten zu implementieren. Dazu wird die folgende Forschungsfrage gestellt: "Kann aus den Positionsdaten von Verkehrsteilnehmern die Bewegung ermittelt und anhand eines Klassifizierer der Typ des Verkehrsteilnehmers ermittelt werden?" Um die Forschungsfrage zu beantworten, wurde eine Schnittstelle implementiert, welche mit der Verkehrssimulation CARLA die Bewegung von Verkehrsteilnehmern berechnet. Das Programm wird dafür auf einem Verkehrsteilnehmer ausgeführt, welchem die Rolle des Helden zugewiesen wird. Aus den Positionsdaten und den Zeitstempeln wird die Geschwindigkeit, Ausrichtung, Beschleunigung und Winkelgeschwindigkeit sowie in Abhängigkeit des Helden der Abstand und die relative Position zum Helden errechnet. Die berechneten Bewegungsdaten werden mit Hilfe eines externen Klassifizierers ausgewertet und die Typen der Verkehrsteilnehmer bestimmt. Der Klassifizierer ist nicht Teil dieser Arbeit, sondern wird in einer separaten Arbeit behandelt und implementiert. Die Ergebnisse der Berechnungen und Klassifizierung weisen bei unterschiedlichen Abweichungen von 0-10m jeweils sehr hohe Erfolgsraten auf.

8 Ausblick

Die Detektion von Verkehrsteilnehmern aus Positionsdaten wäre ein großer Fortschritt im Bereich des autonomen Fahrens. Mit Hilfe der ermittelten Daten könnte die Klassifizierung der Verkehrsteilnehmer unterstützt und dadurch verbessert werden. Visuelle Detektionsarten können auch viele Fehler enthalten, wenn es darum geht zu entscheiden, um welche Art von Verkehrsteilnehmer es sich handelt. Die ermittelten Bewegungsdaten könnten hierbei zusammen mit dem Klassifizierer deutlich bessere Ergebnisse liefern, was bei gemeinsamer Nutzung sogar zu besseren Ergebnissen führen könnte.

Als nächstes wäre es wichtig die Verkehrsteilnehmer in der Simulation realistischer zu gestalten. Fußgänger könnten z.B. stehen bleiben, um Unterhaltungen mit anderen Fußgängern zu führen, da sie derzeit ständig in Bewegung sind, sofern die Ampel nicht gerade auf Rot geschalten hat. Autos könnten nicht nur auf der Straße fahren, sondern auch einparken oder im Stau stehen, was die Geschwindigkeitsdaten stark verändern würde. Ein weiterer wichtiger Punkt ist, dass die Fahrzeuge (Autos, Fahrräder, Lkws und Motorräder) alle fast dasselbe Fahrverhalten aufweisen und sich mit derselben Beschleunigung fortbewegen und bremsen. Es wäre wichtig zu sehen, dass Motorräder größtenteils eine sehr hohe Beschleunigung haben oder Fahrräder eine teils sehr niedrige. Abgesehen von der Simulation wäre es auch von großer Bedeutung das Ganze in einem realen Umfeld zu testen, da dies viel aussagekräftigere Daten erbringen würde. In der aktuellen Lösung werden die GNSS Daten aller Verkehrsteilnehmer von einer zentralen Instanz angefragt. Für eine Implementation für das reale Umfeld wäre es auch wichtig die Positions- und andere Bewegungsdaten (z.B. Geschwindigkeit, Typ und Beschleunigung) von den mobilen Endgeräten der Verkehrsteilnehmern oder den Steuergeräten der Fahrzeuge direkt anzufragen. Dies würde die Berechnungen deutlich vereinfachen und auch genauere Werte liefern, da wegen der Abweichung der Positionsdaten sonst nur verfälschte Daten ausgerechnet werden und durch das Ablesen der Geschwindigkeit vom Steuergerät eine viel genauere Aussage gemacht wird.

Im Großen und Ganzen könnte der Aufgabenbereich in die Verbesserung der Simulation und die Entwicklung einer Lösung für das reale Umfeld geteilt werden.

Literaturverzeichnis

[VISION+2021] Umfrage: Deutschland skeptisch gegenüber autonomen Autos, https://vision-mobility.de/news/umfrage-deutschland-skeptisch-gegenueber-autonomen-autos-89374.html (Zugriff 14.02.2022).

[RADAR] Advantages and Disadvantages of RADAR systems, https://lidarradar.com/info/advantages-and-disadvantages-of-radar-systems (Zugriff 14.02.2022).

[LiDAR] Advantages and Disadvantages of LiDAR https://lidarradar.com/info/advantages-and-disadvantages-of-lidar (Zugriff 14.02.2022).

[Tenzer+2022] Tenzer, F.: Statistiken zur Smartphone-Nutzung in Deutschland, https://de.statista.com/themen/6137/smartphone-nutzung-in-deutschland/ (Zugriff 21.02.2022).

[Population] Current population of Germany, https://www.destatis.de/EN/Themes/Society-Environment/Population/Current-Population/_node.html (Zugriff 21.02.2022).

[Ionic] Introduction to Ionic, https://ionicframework.com/docs (Zugriff 02.03.2022).

[Angular] What is Angular?, https://angular.io/guide/what-is-angular (Zugriff 07.03.2022).

[Stereo Vision] Stereo Vision for 3D Machine Vision Applications, https://www.clearview-imaging.com/en/blog/stereo-vision-for-3d-machine-vision-applications (Zugriff 14.02.2022).

[Spektrum] Radar, https://www.spektrum.de/lexikon/physik/radar/11970 (Zugriff 14.04.2022).

[Fraunhofer] Was ist Radar?, https://blackvalue.de/de/radar-grundlagen/was-ist-radar.html (Zugriff 20.04.2022).

[Wolff SSR PSR] Wolff, C.: Vergleich zwischen Primärradargerät (PSR) und Sekundärradargerät (SSR), https://www.radartutorial.eu/02.basics/Vergleich%20SSR%20mit%20PSR.de.html (Zugriff 28.02.2022).

[Wolff Bistatisch] Wolff, C.: Bistatisches Radarverfahren, https://www.radartutorial.eu/05.bistatic/bs04.de.html (Zugriff 02.03.2022).

[Weber+2018] Weber, H.: FUNKTIONSWEISE UND VARIANTEN VON LiDAR-SENSOREN, S. 2ff, https://cdn.sick.com/media/docs/5/25/425/whitepaper_lidar_de_im0079425.pdf (Zugriff 02.03.2022).

[LeddarTech] Why LiDAR, https://leddartech.com/why-lidar/ (Zugriff 23.02.2022).

[Ivankov+2020] Advantages and Disadvantages of Lidar, https://www.profolus.com/topics/advantages-and-disadvantages-of-lidar/ (Zugriff 23.02.2022).

[CARLA] CARLA, https://www.carla.org (Zugriff 19.02.2022)

[GNSS+2020] What is GNSS?, https://www.oxts.com/what-is-gnss/ (Zugriff 20.02.2022).

[GPS] Wie funktioniert GPS?, https://www.magicmaps.de/gnss-wissen/wie-funktioniert-gps/?L=0 (Zugriff 20.02.2022).

[EGNOS] What is EGNOS?, https://www.esa.int/Applications/Navigation/EGNOS/What_is_EGNOS (Zugriff 23.02.2022).

[Papula+2001] Papula, L., Mathematik für Ingenieure und Naturwissenschaftler Band 1, 10., Bd. 1. Braunschweig/Wiesbaden: Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, 2001.

[Bußgeld+2022] Zulässige Höchstgeschwindigkeit innerorts: So schnell dürfen Sie fahren, https://www.bussgeldkatalog.org/hoechstgeschwindigkeit-innerorts/ (Zugriff 21.03.2022).

[Maps] MY MAPS, https://www.google.com/maps/about/mymaps/ (Zugriff 21.03.2022).

[HAMAMATSU] LiDAR, https://www.hamamatsu.com/eu/en/applications/automotive/lidar.html (Zugriff 14.02.2022).