HOCHSCHULE COBURG

Hochschule für angewandte Wissenschaften Coburg

Fakultät Elektrotechnik und Informatik

Studiengang: Informatik

Bachelorarbeit

Klassifizierung der Relevanz von Verkehrsteilnehmern zur Erweiterten Umfeldwahrnehmung Autonomer Fahrzeuge

Patrick Opitz

Abgabe der Arbeit: 3. Oktober 2022

Betreut durch:

Prof. Dr. Thomas Wieland, Hochschule Coburg

Kurzzusammenfassung

In einer Bachelorthesis aus dem letzten Semester wurde ein Konzept erarbeitet, wie aus den Ortsdaten von Mobilfunkendgeräten, Art und Bewegungsinformationen von Verkehrsteilnehmern wie Fußgängern, Autos und Fahrradfahrern, bestimmt werden können. In einer anderen Bachelorarbeit aus dem vorangehenden Semester, wurden diese Bewegungsdaten über verschiedene Klassifikationsmethoden bewertet und verglichen, um die Verkehrsteilnehmer nur anhand ihrer Daten zuzuordnen. Als nächsten Schritt soll mit dieser Arbeit untersucht werden, wie aus diesen Ergebnissen Abschätzungen zur Relevanz dieser Teilnehmer für den Fahrweg eines einzelnen Fahrzeugs abgeleitet werden können. Um dies zu untersuchen, kann durch die Ortsdaten mit Bewegung der anderen Autos abgeschätzt werden, ob diese den eigenen Weg kreuzen oder nicht. Diese Möglichkeit soll in der Simulationsumgebung namens CARLA mit der Programmiersprache Python realisiert werden. Hierzu wird in CARLA ein kleines Straßennetz simuliert, in welchem diverse virtuelle Verkehrsteilnehmer wie Auto, Radfahrer, Lastwagen und Fußgänger vorhanden sind, die sich standardmäßig wie ein perfekter Straßenverkehr ohne Unfälle oder besondere Situationen verhalten. Die Aufgabe dieser Bachelorarbeit ist es nun, diese Simulation mit Python-Methoden zu erweitern, um die Idee der Relevanz zu testen und zu bewerten. Dies wird gegebenenfalls zu einem späteren Zeitpunkt in der Realität getestet und umgesetzt.

Abstract

As part of a bachelor’s thesis from the last semester, a concept has been worked upon to determine location data from mobile terminals and movement information of road users such as pedestrians, cars and cyclists. A different bachelor’s thesis from the previous semester evaluated and compared these motion data using different classification methods in order to assign road users based only on their data. The next step is to investigate how these results can be used to make estimations about the relevance of the participants for the route of a single vehicle, which is the goal of this bachelor’s thesis. To investigate this, one can use the location data with movement of the other cars to estimate whether they cross one's own path or not. This possibility is to be realized in the simulation environment named CARLA with the programming language Python. For this purpose, a small road network is simulated in CARLA with several virtual road users of cars, bicycles, trucks and pedestrians. By default, the road traffic functions perfectly without any accidents or special situations. The task of this Bachelor thesis is to extend the simulation with Python methods in order to test and evaluate the idea regarding relevance. At a later point in time, this simulation may be tested and realized in the real world.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Codeverzeichnis

1. Einleitung

Laut Destatis kommt es innerhalb des Straßenverkehres immer wieder zu Unfällen, bei denen mittlerweile mehr Menschen dabei sterben als verletzt zu werden. Laut Messungen liegen ungefähr 88 % aller Unfälle an fehlerhaftem Fahrverhalten der Fahrzeugführer. Das heißt, der Ausgangspunkt hierbei ist immer die mangelnde Aufmerksamkeit, Betrunkenheit, falsche Geschwindigkeit, zu dichtes Auffahren, Verschätzen beim Abbiegen oder Wenden, Nichtbeachtung der Vorfahrt oder falsche Überholvorgänge und sonstiges Fehlverhalten wie auch falsche Straßenbenutzung. [1, p. 50]

Zusätzlich merkt Destatis an, dass der Großteil aller Unfälle sich innerorts ereignen. Außerorts auf Landstraßen oder Kreisstraßen geschehen weniger Unfälle und nur ein relativ kleiner Anteil wird auf Autobahnen gemeldet. Die meisten Menschen sterben an Unfällen außerorts. Im Gegensatz dazu gab es innerorts die meisten leichten Verletzten und mehr Schwerverletzte als außerorts. [1, p. 44]

Die schwachen Verkehrsteilnehmer (Motorradfahrer, Fahrradfahrer und Fußgänger) haben besonders unter den Unfällen zu leiden, da diese bei einem Zusammenstoß immer die größeren Folgen zu tragen haben. Somit sollten diese besonders geschützt werden. Da Fehler von Autofahrern die wahrscheinlichste Ursache von Unfällen sind, sollten diese mehr auf ihre Fehler aufmerksam gemacht werden.

Das Ziel dieser Bachelorarbeit ist es, von Verkehrsteilnehmern die Positionsdaten und dessen Richtungsinformationen zu verarbeiten und anhand dessen eine Relevanz dieser zueinander aufzustellen. Durch die Position und Richtung kann zusätzlich die Geschwindigkeit der Verkehrsteilnehmer errechnet werden. Mit diesen Informationen wird eine mögliche Behinderung des Straßenverkehrs berechnet, die im schlimmsten Fall zu einem Unfall führen könnte. Durch diese Vorhersagen lassen sich die Verkehrsteilnehmer auf mögliche Gefahren hinweisen, wodurch Unfälle mit angemessen Handeln verhindert werden können.

Im folgenden Kapitel werden zunächst die allgemeinen Grundlagen und die Methodiken vorgestellt, welche die Wissensbasis für die Arbeit darstellen. Hier wird der Titel der Arbeit genauer beschrieben sowie das Wort Simulation und andere häufig aufkommende Begriffe, die außerhalb des Kontexts anders verstanden werden könnten, als in dieser Arbeit definiert. Zusätzlich werden in diesem Kapitel die mathematischen Grundlagen beschrieben.

Im nächsten Teil der Arbeit wird auf die spezifischen Grundlagen eingegangen. Die einerseits gebraucht werden, um die Arbeit zu verstehen, andererseits was genau für diese Arbeit verwendet wird, sodass diese überhaupt Funktionen kann. In diesem Kapitel wird erstmalig beschrieben, auf welche Informationen und Erkenntnisse vorangegangener Bachelorarbeiten diese Arbeit aufbaut. Weiterhin wird die Technologie sowie die Programme und deren genutzte Version, die für diese Arbeit verwendet werden, genauer vorgestellt. Im ersten Teil des Hauptteiles geht es um die Anforderungen und das Gesamtkonzept. In diesem Kapitel handelt es sich um die Straßennetze von CARLA, mit welchen der Straßenverkehr in CARLA gesteuert werden kann und für welche zusätzliche Einstellungsmöglichkeiten zur Verfügung stehen. Im darauffolgenden Teil wird die Programmstruktur sowie die Implementierung der einzelnen Methoden nacheinander beschrieben. Der letzte Teil des Hauptteils handelt um die Evaluierung der zuvor beschriebene Programmmethoden mit vereinzelten Tests und Screenshots. Am Ende wird die Arbeit noch mit einer kleinen Zusammenfassung abgerundet und im Nachgang folgt noch ein kurzer Zukunftsausblick.

2. Allgemeine Grundlagen und Methodiken

In diesem Kapitel befinden sich die theoretischen Grundlagen, um die Arbeit verständlicher zu gestalten.

2.1 Klassifikation

Unter Klassifikation versteht man die Einteilung einer Menge anhand seiner Merkmale in verschiedenen Gruppen oder Klassen. In dieser Arbeit sollen die Verkehrsteilnehmer untereinander klassifiziert werden. Dies erfolgt anhand der Auswirkung für den jeweiligen anderen Teilnehmer. Diese Auswirkung wird anhand der Relevanz gemessen.

2.2 Relevanz

Das Wort Relevanz steht für den allgemeinen Zusammenhang von zwei Objekten in der Bedeutsamkeit und Wichtigkeit. In dieser Arbeit geht es um die Relevanz von 2 Verkehrsteilnehmern zueinander. In Hinsicht auf Zeit und Sicherheit hat ein Verkehrsteilnehmer auf einen anderen keinen Einfluss auf die Fahrzeit und gefährdet ihn auch nicht. Somit sind diese beiden Teilnehmer zueinander irrelevant. Im Vergleich zur Relevanz wird die Sicherheit höher gemessen als die Zeitbeeinflussung.

2.3 Simulation

Eine Simulation ist ein Modell, das die Funktionsweise eines bestehenden oder geplanten Systems nachahmt und durch die Möglichkeit, verschiedene Szenarien oder Prozessänderungen zu testen, Entscheidungsgrundlagen liefert.

Simulationen können eingesetzt werden, um die Leistung zu verbessern, einen Prozess zu optimieren, die Sicherheit zu erhöhen und Theorien zu testen. Die wissenschaftliche Modellierung von Systemen ermöglicht es dem Benutzer, einen Einblick in die Auswirkungen verschiedener Bedingungen und Vorgehensweisen zu gewinnen.

Die Simulation funktioniert durch den Einsatz einer intuitiven Simulationssoftware zur Erstellung eines visuellen Modells eines Prozesses. Diese visuelle Simulation sollte Details zu Zeitabläufen, Regeln, Ressourcen und Beschränkungen enthalten, um den realen Prozess genau wiederzugeben. [2]

2.4 Actor

Das Wort Actor kommt aus dem Englischen und bedeutet Schauspieler. Im Zusammenhang mit dieser Arbeit wird der Begriff Actor stellvertretend für ein simuliertes Objekt von verschiedenen Verkehrsteilnehmern verwendet. Diese werden von der Software CARLA simuliert, und können aus sowohl Fahrzeugen wie Autos, LKWs, Motorrädern sowie Fahrradfahrer oder Fußgänger bestehen. Die hierbei generierten Objekte werden an verschiedenen Orten innerhalb der Simulationsumgebung gesetzt und arbeiten im Anschluss ihre vorgegebenen Regeln und Bedingungen ohne weiteren Einfluss ab. Ausführlichere Beschreibungen zu diesem Vorgehen finden innerhalb des Kapitels 4.3 statt. Zusätzlich bietet CARLA die Möglichkeit verschiedene Informationen, wie die aktuelle Position, den Namen des Objekts, die Geschwindigkeit, die Richtung der Bewegung und vereinzelten Sensoren aller beteiligten Actors abzufragen. Die Informationen und Daten werden in dieser Arbeit zur Vereinfachung der Analyse verwendet, um die Vorteile der CARLA Simulation anzuwenden.

2.5 Hero

Das englische Wort für Held.

Innerhalb dieser Arbeit wird ein spezieller Actor als Hero bezeichnet, der wiederum wie alle anderen Actors vollkommen automatisch im Straßenverkehr navigiert. Der Hero ist in diesem Zusammenhang jedoch eine Ausnahme, da hier wahlweise auch manuell innerhalb der Simulation interagiert werden kann. Diese Möglichkeit wird hier gezielt eingesetzt, um manche Situationen herbeizuführen und besser nachvollziehbar zu machen. Dieses Objekt wird in allen Methoden als sogenannter Hauptdarsteller betrachtet. Damit ist gemeint, dass ein User mit der Simulation interagiert und die Steuerung eines Objekts übernimmt. Hierbei werden alle weiteren Actors als weitere Verkehrsteilnehmer gesehen. Somit hat der Hero in Abhängigkeit seiner Interaktion direkten Einfluss auf alle Beteiligten. Letztendlich kann jeder Actor als Hero gesehen werden. In dieser Arbeit werden alle Actors aus der Sicht des Heros betrachtet.

2.6 Euklidischer Abstand (Euclidean Distance)

Der Euklidische Abstand, auch Euklidische Distanz, **euklidische Metrik** oder **euklidische Norm** genannt, ist ein Mittel der euklidischen Geometrie (anschauliche Geometrie des Zwei-

oder Dreidimensionalen), um den Abstand zwischen zwei Punkten zu berechnen, indem die Länge der Strecke, durch einen Zwischenvektor ausgerechnet wird.

Im Zweidimensionalen:

$$d = \sqrt{(Bx_2 - Ax_1)^2 + (By_2 - Ay_1)^2}$$

- (Ax_1, Ay_1) sind die Koordinaten des ersten Punkts A.
- (Bx_2, By_2) sind die Koordinaten des anderen Punkts B.
- d ist die Distanz zwischen den Punkten A und B.

Im Dreidimensionalen:

$$d = \sqrt{(Bx_2 - Ax_1)^2 + (By_2 - Ay_1)^2 + (Bz_2 - Az_1)^2}$$

- (Ax_1, Ay_1, Az_1) sind die Koordinaten des ersten Punkts A.
- (Bx_2, By_2, Bz_2) sind die Koordinaten des zweiten Punkts B.
- d ist die Distanz zwischen Punkt A und Punkt B.

2.7 Bremswegfaustregel

„Bremsweg = Tacho durch 10 zum Quadrat."
[3]

3. Spezifische Grundlagen

3.1 Bachelorvorarbeit

Im Rahmen dieser Bachelorarbeit gibt es zwei vorangehende Bachelorarbeiten, die sich mit ähnlichen Themen auseinandergesetzt hatten. Die Ergebnisse der beiden Vorarbeiten werden in dieser Arbeit als Grundstein verwendet, um weiter auf diesen aufzubauen.

3.1.1 Detektion der Bewegung von Verkehrsteilnehmern aus Positionsdaten

Inhalt:

> Maschinelle Lernverfahren gewannen in den vergangenen Jahren zunehmend an Relevanz in den entsprechenden Fachbereichen. Die vorliegende Bachelorarbeit beschäftigt sich mit der Anwendung von maschinellen Lernverfahren zur Vorhersage eines Klassifikationsproblems. Es soll im Rahmen der erweiterten Umfeldwahrnehmung autonomer Fahrzeuge erforscht werden, inwieweit sich Positionsdaten von bewegten Mobilfunkteilnehmern im Straßenverkehr dazu eignen, Vorhersagen über die Art des Verkehrsteilnehmers zu treffen. Dafür werden zunächst verschiedene Verkehrsszenarien mit der Simulationssoftware CARLA simuliert, aus denen die erzeugten Positionsdaten entnommen und im Nachgang zu relevanten Informationen (Geschwindigkeit, Beschleunigung etc.) umgewandelt werden. Diese werden in verschiedenen Datensätzen aufgezeichnet und später für das Training / die Evaluation der Algorithmen verwendet. Um eine möglichst umfassende Perspektive in Bezug auf den Forschungsgegenstand zu generieren, werden drei verschiedene Klassifikationsmethoden (K-Nearest-Neighbor, Decision Tree, Support Vector Machine) in Betracht gezogen, deren Ergebnisse darauffolgend analysiert und für die Bewertung der einzelnen Verfahren verglichen werden. Zudem werden, je Lernverfahren, verschiedene Aspekte (Klassenanzahl, Umgebungskomplexität, Genauigkeit der GNSS-Daten) betrachtet und deren Auswirkung auf die Genauigkeit der jeweiligen Erkennungsraten untersucht. Die Resultate der Untersuchungen ergeben, dass sich die Informationen, welche sich durch die GNSS-Daten gewinnen lassen - je nach Szenario - gut dazu eignen, unbekannte Objekte durch Klassifikationsverfahren korrekt vorherzusagen. [4]

3.1.2 Klassifizierung der Bewegungsmuster von Mobilfunkteilnehmern zur erweiterten Umfeldwahrnehmung autonomer Fahrzeuge

Inhalt:

Seit einigen Jahren durchdringt das Thema Automatisierung alle Bereiche der Gesellschaft, so auch die Arbeitswelt. Immer mehr Prozesse aus allen Arbeitsbereichen werden automatisiert. Auch Software und Code-Generierung bilden hierbei keine Ausnahmen, insbesondere die Generierung hardwarenahen Codes, der beispielsweise dazu dient, die zeitintensive Entwicklung zu beschleunigen.

Die moderne Industrie ist ohne automatisierte Prozesse nicht mehr denkbar. Vor allem bei der Herstellung von Standardprodukten ist die Automatisierung von Prozessen essenziell, um die Produktionskosten gering zu halten. Maschinen- und Großanlagenhersteller haben in der Regel für jede ihrer Anlagen ein fertiges Steuerungsprogramm. Abhängig von den gebuchten Optionen und Zusatzfunktionen der Kunden soll das allgemeine Steuerungsprogramm angepasst werden. Erste Inbetriebnahme und Anpassungsarbeiten an der Steuerungssoftware werden meistens von Monteuren vor Ort beim Kunden vorgenommen, die wenig Programmiererfahrung mitbringen, weshalb Fehler verursacht werden können. Die Automatisierung dieser Anpassungsarbeiten spart Zeit, Ressourcen sowie Kosten und verschafft einen langfristigen Vorteil, um wettbewerbsfähig zu bleiben.

Diese Arbeit befasst sich damit, eine Anwendung in C# zu entwickeln, die automatisch ein passendes Steuerungsprogramm für Betriebsanlagen generiert, indem sie die spezifischen Anforderungen eines Kundenauftrags berücksichtigt. Die Anwendung soll Gesamtprojekte anhand vorgefertigter Projektteile erstellen. Sie trägt den Namen TIA-Generator, kurz TIAG. [5]

3.2 CARLA

CARLA ist ein Open-Source-Simulator für die Forschung zum autonomen Fahren. Es dient zur Unterstützung der Entwicklung, dem Training und der Validierung von autonomen städtischen Fahrsystemen. Zusätzlich zum quelloffenen Programmcode bietet CARLA offene digitale Assets (Stadtgrundrisse, Gebäude und Fahrzeuge), die zu diesem Zweck erstellt wurden und frei verwendet werden können. Die Simulationsplattform unterstützt die flexible Spezifikation von Sensoren und Umgebungsbedingungen. [6]

3.3 Python 3.7

Für diese Arbeit wird bevorzugt mit Python in Version 3.7 programmiert, anstelle einer neueren Programmversion. CARLA in Version 0.9.13 funktioniert mit der genannten Python-Version am besten. Dazu kommen diverse, herunterladbare Bibliotheken zu Python 3.7.

4. Anforderungen und Gesamtkonzept

Das folgende Kapitel beschreibt die Anforderungen und das Gesamtkonzept, die für diese Bachelorarbeit zu erbringen sind. Als Simulationsumgebung wird das Open-Source-Projekt CARLA verwendet, in welchem städtischer Verkehr simuliert werden kann. Dieser besteht aus Fahrzeugen, Motorrädern, Fahrrädern sowie Fußgängern. Die hier beschriebenen Verkehrsteilnehmer werden dazu verwendet, um ihre jeweiligen Positionen sowie Geschwindigkeiten abzufragen und um diese anschließend weiterzuverarbeiten. Anschließend wird einer der simulierten Fahrzeuge als sogenanntes Hauptfahrzeug bestimmt, von welchem die Bewegungsdaten der anderen Verkehrsteilnehmer relativ zum ausgewählten Hauptfahrzeug berechnet werden. Dadurch errechnet sich der Abstand und die relative Position der anderen Modelle, wodurch die beschriebene Relevanz schlussgefolgert werden kann.

4.1 Verschiedene Städte in CARLA

In CARLA steht eine breite Auswahl von verschiedenen Städten zum Simulieren zur Verfügung. Dabei sind viele der einzelnen Städte in Hinblick auf Komplexität sehr ähnlich aufgebaut. Nur hinsichtlich der Größe lassen sie sich diese im Wesentlichen voneinander differenziert betrachten. Die Simulation dieser Arbeit wurde hauptsächlich in den Städten „Town10“ und „Town07“ durchgeführt. Auf dessen Unterschiede und Eigenschaften wird nachfolgend kurz eingegangen.

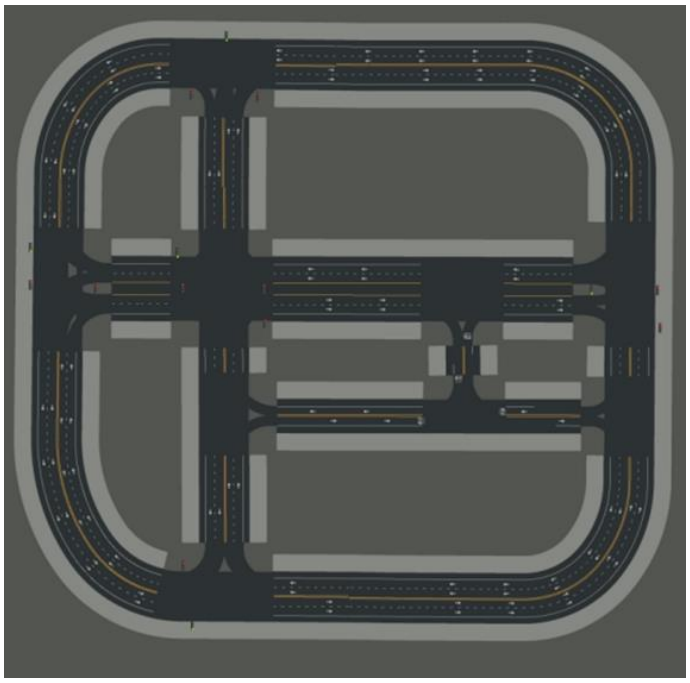


Abbildung 1: Town10 Layout
Quelle: [7]

„Town10“ verkörpert ein kleines amerikanisches Stadtbild mit simplem Straßennetz. Die meisten Städte, die in CARLA standardmäßig zur Verfügung stehen, unterscheiden sich nicht wesentlichen von diesem Stadtbild. Hier sind lediglich geraden Straßen, 90-Grad-Kurven und Straßenmündungen inklusive Ampeln beinhaltet. Nur die Anzahl der Straßen ändert sich jeweils in den meisten anderen Städten, mit vereinzelten kleineren Änderungen. Da Town10 die neueste Stadt sowie die Standardstadt der aktuellen CARLA-Version ist, wurde sie zusammenfassend zu den anderen Städten als Basis ausgewählt.

Abbildung 2: Town07 Layout
Quelle: [7]

„Town07“ hat ein anderes Umfeld, dass nun keine Stadt mehr ist, sondern eher eine ländliche Farmlandschaft simuliert. Zum einen ist Town07 nicht nur größer, sondern hat ebenfalls auch andere Kreuzungen als nur 90-Grad-Kurven. Zum anderen ist einer der größten Unterschiede dieser Stadt im Gegensatz zu anderen, dass zwei Straßen in der Stadt verbaut sind und nicht nur gerade von einer Kreuzung zu anderen führt. Die verschiedenen Straßen haben in diesem Fall schon eine kurvige Straßenführung. Um noch weitere zusätzliche Ereignisse zu simulieren, beinhaltet Town07 noch einen Kreisverkehr und einen zusätzlichen Parkplatz, welche nicht in Town10 vorhanden sind. Eine Übersicht aller genutzten Town-Karten von 10 und 07 sowie die nicht betrachteten, jedoch im Sortiment enthaltenen Town-Karten, sind im Anhang A1 ersichtlich. In dieser Übersicht lassen sich die Größenunterschiede und Gemeinsamkeiten genauer erkennen, insbesondere warum nur diese 2 Städte zum Testen verwendet wurden. Würde dieser Test in allen Städten durchgeführt werden, so führe das Resultat in ähnlichen Städten zu gleichen Ergebnissen. Eine Umsetzung davon würde zeitlich und auch vom Umfang her den Rahmen dieser Bachelorarbeit sprengen.

4.2 Straßenverkehr in CARLA

Der Straßenverkehr in CARLA wird standardmäßig ohne zusätzliche Änderungen perfekt simuliert. Damit ist gemeint, dass sich alle Verkehrsteilnehmer, ob Auto, Motorrad, Radfahrer und sogar Fußgänger, sich exakt an die Verkehrsregeln halten. Alle Fahrzeuge fahren die gleiche Geschwindigkeit, halten den Mindestabstand zueinander ein und machen keine Fehler bei Ampeln oder Verkehrsschildern. Da die beschriebenen Regeln für die Bachelorarbeit zu realitätsfremd, zu unfallfrei und beim Vergleich des Abstands meistens gleiche Ergebnisse liefert, muss im Vorfeld für diese Bachelorarbeit das Regelwerk abgeändert werden. Dieses Vorgehen ist in diesem Fall notwendig, um nicht unendlich viele verschiedene Straßenverkehrssituationen zu simulieren, wurde in dieser Arbeit eine einmalige und einheitliche Änderung der Standard-CARLA-Regeln vorgenommen

1. Der Standard-Sicherheitsabstand von 5 Metern aller Fahrzeuge wird bei der Erstellung einmalig zufällig zwischen 0,00 und 5,00 Metern, mit auch möglichen Änderungen im Kommabereich festgelegt. Dieser ändert sich nicht mehr, außer das Fahrzeug wird gelöscht oder neu erstellt. Falls genau 0,00 Meter bei einem Fahrzeug ausgewählt wird, fährt dieses ungehindert und sofern es plausibel ist, in das andere Fahrzeug hinein und verursacht damit einen Unfall.

2. Alle Verkehrsteilnehmer ignorieren nun zu 50 % (Münzwurf) statt normal 0 % (nie) die Rotschaltung jeder Ampel. Somit fahren sie einfach über die Kreuzung, obwohl sie an dieser anhalten müssten. Dies machen sie auch unabhängig davon, ob andere Fahrzeuge sich auf der Kreuzung befinden. Nur der Sicherheitsabstand aus der Regeländerung 1 würde in diesem Fall einen Unfall verhindern, indem das später eintreffende Auto sich an dessen vorbestimmten Abstand hält.

3. Zusätzlich bekommen alle Verkehrsteilnehmer bei der Erstellung die Anweisung nur zwischen 10 % und 200 % Standard-Geschwindigkeit zu fahren. Die Standard-Geschwindigkeit wird von der CARLA-Karte bestimmt, aber ist in dieser Arbeit auf 60 km/h festgelegt. Bekommt ein Teilnehmer eine Zahl unter 100 %, fährt er dementsprechend langsamer bis minimal 1/10 der Geschwindigkeit. Im anderen Fall fährt das Fahrzeug maximal doppelter Geschwindigkeit auf genau 200 %. Da ein Fahrzeug diesen Wert immer beibehält, darf ein Wert von 0% nicht vergeben werden, da es sonst für immer stehen bleiben würde.

Diese Änderungen wurden absichtlich extremer als für die Realität üblich gewählt, damit schneller Unfälle oder andere besondere Situationen im Rahmen der Testvorgänge in dieser Bachelorarbeit entstehen können.

4.3 Verkehrsteilnehmer Relevanz

4.3.1 Relevanz vom Abstand

Die erste Möglichkeit der Relevanzbestimmung bietet der Abstand. Dieser wird einfach aus den jeweiligen Koordinaten der beiden Teilnehmer berechnet. Möchte man die Relevanz nur allein anhand des Abstands festlegen, gilt die einfache Regel, dass der Verkehrsteilnehmer (Actor) mit dem geringsten Abstand zu einem selbst (Hero) die größte Relevanz hat. Im Umkehrschluss hat der Actor mit dem größten Abstand die kleinste Relevanz zum Hero.

Nimmt man zum Abstand noch die Geschwindigkeit dazu, kann mithilfe der Bremswegfaustregel (Kapitel 2.7) der Mindestabstand zwischen zwei Verkehrsteilnehmern bestimmt werden, der für ein sicheres Fahren vorgesehen ist. Alle Fahrzeuge unter diesem Abstand haben eine kritische Relevanz zueinander. Teilnehmer, die leicht darüber sind, haben eine moderate Relevanz, die mit steigendem Abstand weiter ins Unbedeutende sinkt.

4.3.2 Relevanz von Sicherheitsparameter TTC-α

Der nächste Schritt Richtung aussagekräftiger Relevanz ist der Sicherheitsparameter TTC-α.

Die Rechnung vom Parameter ist:

$$\text{TTC-}\alpha = \left|\frac{AS}{A.v} - \frac{BS}{B.v}\right|$$

AS: Strecke von Fahrzeug A zum Treffpunkt S in Meter

BS: Strecke von Fahrzeug B zum Treffpunkt S in Meter

A.v: Gesamtgeschwindigkeit von Fahrzeug A in Sekunden

B.v: Gesamtgeschwindigkeit von Fahrzeug B in Sekunden

Der α Wert drück aus, um wie viele Sekunden sich beide Verkehrsteilnehmer bei einem möglichen Treffpunkt verpassen. Solle hier kein Treffpunkt resultierten oder es unmöglich sein, sich an diesem zu treffen, ist TTC-α unendlich groß. Resultiert hier jedoch ein konkreter Zusammenstoß, ist der Wert genau null. Daher folgt, desto kleiner der α-Wert ist, umso wahrscheinlicher kommt es zu einem Unfall.

4.3.3 Relevanz von TTC (time-to-collision)

Die TTC-Rechnung berechnet, wie lange es dauert, bis ein Fahrzeug mit einem anderen mit konstanter Geschwindigkeit und Richtung kollidieren wird.

Grundsätzlich wird der einfache TTC-Wert mit der folgenden Formel berechnet:

$$\mathrm{TTC} = \frac{D}{V_r}$$

[8, p. 3]

D steht für die Strecke zwischen den beiden Fahrzeugen.

V_r steht für die relative Geschwindigkeit der beiden Fahrzeuge.

In dieser Arbeit wird anstatt des simplen TTC-Wertes der erweitere TTC-Wert verwendet.

Im Gegensatz zum simplen TTC wird dieser Situation abhängig berechnet. Es wird unter 32 verschiedenen Annäherungsmöglichkeiten zweier Fahrzeuge unterschieden. Jede Ecke jedes Fahrzeugs (4 Ecken) kann auf jeder der 4 Seiten des anderen Fahrzeugs aufprallen. Diese Berechnung muss mit beiden Fahrzeugen wiederholt werden. Es sind jedoch nur 10 Unfallkonfigurationen möglich. [9, p. 3]

Aus diesen kann jeweils ein aussagekräftiges TTC gebildet werden, mit welchen man genau sagen wann, wo und wie die beiden Fahrzeuge kollidieren werden.

Mit dem TTC-Wert kann man das Risiko von zwei Fahrzeugen bestimmten, dazu gibt es einen Sicherheitskäfig Abb.4. Der TTC mit einer prozeduralen Bremsung vergleicht und bestimmt, wie gefährlich die Situation ist.

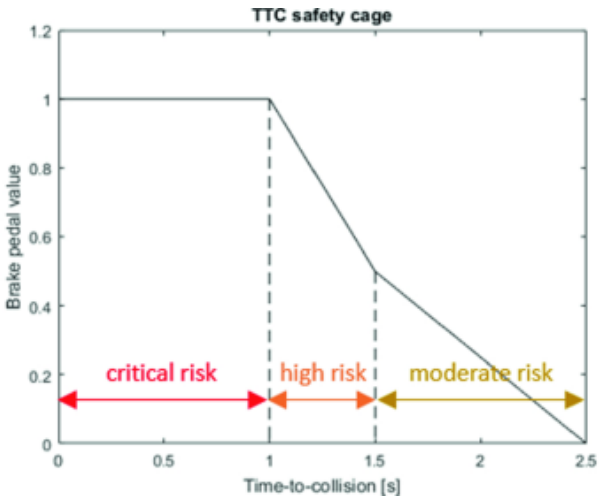


Abbildung 3: TTC-Sicherheitskäfig
Quelle [10]

4.3.4 THW (Time-headway)

Der THW-Wert sagt aus, wie viel Zeitvorsprung das eine Fahrzeug zum anderen hat. Die Rechnung zum THW-Wert sieht dem TTC-Wert ähnlich, mit dem Unterschied, dass nicht die relative Geschwindigkeit von beiden Fahrzeugen verwendet wird, sondern die subjektive Geschwindigkeit eines Fahrzeugs zum anderen. Daher muss man beachten, von welchem Fahrzeug aus man den THW-Wert haben möchte, da dieser selbst bei 2 Fahrzeugen von einem zum anderen unterschiedlich ist.

$$\mathrm{THW} = \frac{D}{V_s},$$

[8, p. 3]

Die untenstehende Abbildung ist eine Grafik für einen THW-Sicherheitskäfig. Dieser ist funktionell identisch zu dem vom TTC-Sicherheitskäfig.

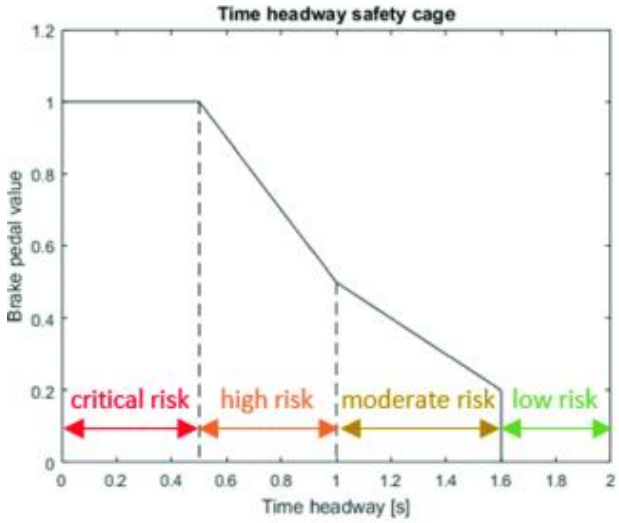


Abbildung 4: THW-Sicherheitskäfig
Quelle [10]

4.3.5 Relevanzvergleich

Die Abstandsrelevanz allein ist im Vergleich zu den anderen Werten TTC-α, TTC und THW redundant, da der Abstand in allen Methoden berücksichtigt wird. Wenn man TTC direkt mit THW mithilfe der beiden Sicherheitskäfige vergleicht, sieht man, dass die Sicherheit von TTC bei einem höheren Wert als von THW schneller abnimmt. Daher kann man sagen, dass TTC mehr Auswirkungen auf die Gesamtrelevanz hat als THW. Aber die beste Aussage der Relevanz bekommt man, wenn man beide Werte in Relation zueinander stellt. Wie im Graphen von Abb.6.

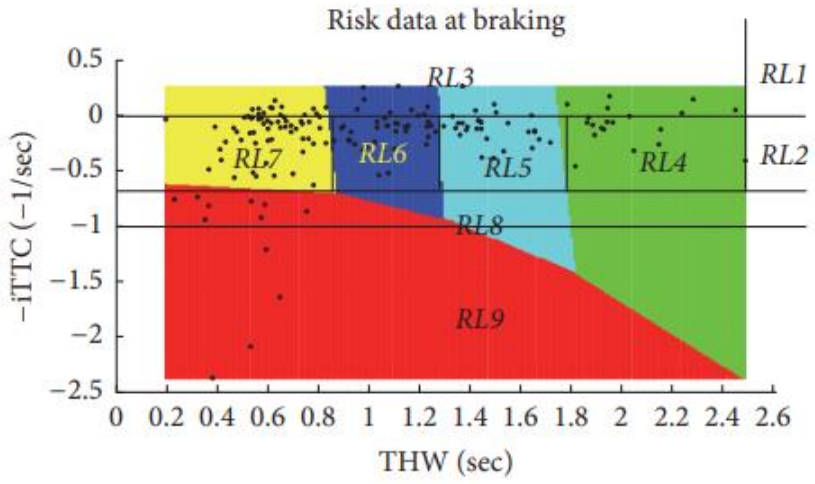


Abbildung 5: Risiko Einteilung mit THW und TTC
Quelle: [8]

4.3 Programm Aufbau

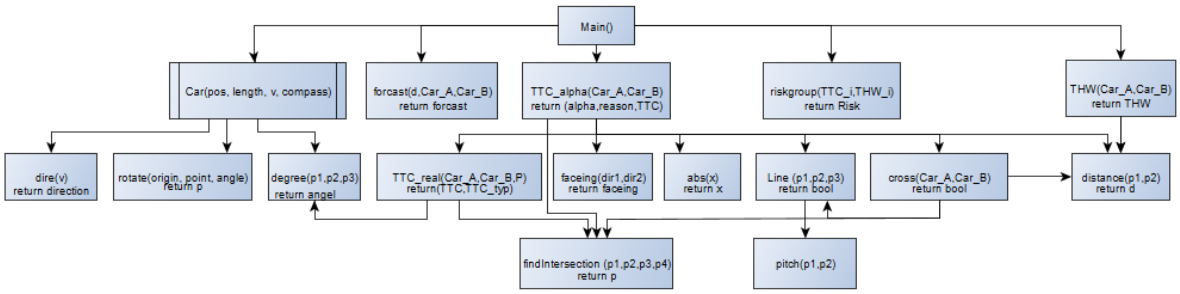


Abbildung 6: Methoden Verlauf
Quelle: [selbst erstellt]

In oben aufgeführter Abbildung sieht man sämtliche Methoden des Programms auf einen Blick. In den Kästen stehen die Namen der Methoden mit deren Übergabeparametern und anschließenden Rückgabewerten. Die Pfeile zeigen den Methodenfluss an. Die Methode am Fuß des Pfeils ruft irgendwann während ihrer Bearbeitung die Methode an der Spitze des Pfeils auf und wartet auf dessen Rückgabewert. Erst, wenn sie diesen erhält, arbeitet diese weiter. Der Kasten mit zwei senkrechten Strichen steht als einziger für einer der beiden Klassen. Die andere Klasse wurde weggelassen, da sie von nahezu allen Methoden aufgerufen wird.

5. Implementierung

Auf Grundlage der beiden vorherigen Bachelorarbeiten werden in dieser zur Vereinfachung die Positionen der einzelnen Verkehrsteilnehmer und dessen Typ als schon bestimmt gewertet.

5.1 CARLA Abfragen

Hierzu wird einfach im Client Standardfunktionen von CARLA verwendet. Um die Positionskoordinaten der einzelnen Fahrzeuge zu erhalten, wird jeweils folgendes Attribut der Vehicle-Klasse von CARLA mit den folgenden 3 Befehlen abgefragt:

```
vehicle.get_location().x
vehicle.get_location().y
vehicle.get_location().z
```

Um die Bezeichnung des Fahrzeugs zu bestimmten, wird der Befehl…

```
get_actor_display_name(vehicle, truncate=22)
```

… verwendet, mit welchen man den Namen des Actors in eine temporäre Variable abspeichert. Außerhalb müssen die Ergebnisse von CARLA von der vorherigen Bachelorarbeiten verwendet werden. Um nun die Werte der anderen Autos im Client vom Hero zu erhalten, wird in dessen Quellcode nach der Anzeige der allgemeinen Information vom Helden wie folgt angepasst:

```
1  vehicles = world.world.get_actors().filter('vehicle.*')
2  if len(vehicles) > 1:
3        self._info_text += ['Nearby vehicles:']
4        for vehicle in vehicles:
5              vehicle_type = get_actor_display_name(vehicle, truncate=22)
               self._info_text += ['Nearby vehicles:']
6              self._info_text.append('%s %s' % (vehicle_type,
7        vehicle.get_location))
```

Code 1: Beispiel Anzeige des Clienten

In diesem Codeausschnitt wird zuallererst überprüft, ob es überhaupt mehr als nur ein Fahrzeug in der Simulation gibt. Dies wird erreicht, in dem alle Actors vom Typ Vehicle in eine Variable gespeichert werden, welche anschließend auf Länge geprüft wird. Anschließend wird aus der vorherigen Variable eine For-Schleife erstellt, die nun die Liste von anderen Fahrzeugen durchläuft und jeweils die passenden Informationen von diesen an die Hero-Anzeige des Clients anhängt. Das Resultat ist nun die folgende Anzeige, mit der die Anzahl aller Fahrzeuge, die in einer Liste mit Typ und Position stehen, ausgibt.

5.2 Abstände

Für den Anfang werden die anderen Actors nach ihrer Distanz von nah nach fern vom Hero sortiert und im Anschluss ausgegeben. Hierfür wird der vorherige Code um ein paar Zeilen erweitert.

```
1  vehicles = world.world.get_actors().filter('vehicle.*')
2  if len(vehicles) > 1:
3              self._info_text += ['Nearby vehicles:']
4              distance = lambda l: math.sqrt((l.x - t.location.x)**2 + (l.y –
               t.location.y)**2 + (l.z - t.location.z)**2)
6              vehicles = [(distance(x.get_location()), x) for x in vehicles
               if x.id != world.player.id]
7              for d, vehicle in sorted(vehicles, key=lambda
8              vehicles: vehicles[0]):
9                  if d > 200.0:
                       break
10                 vehicle_type = get_actor_display_name(vehicle, truncate=22)
11                 self._info_text += ['Nearby vehicles:']
12                 self._info_text.append('%    4dm    %s    %s'    %    (d,
13                 vehicle_type, vehicle.get_location))
```

Code 2: Erweitere Klienten Anzeige

In Zeile 4 vom Quellcode wird eine Distanz mit einem Lambda-Ausdruck definiert, um diesen später besser nutzen zu können. Die Berechnung davon ist der Euklidische Abstand (Grundlagen 2.6) vom Hero zum jeweils anderen Actor. In der darauffolgenden Zeile wird der Abstand konkret ausgerechnet, indem die ganze Fahrzeugliste durchlaufen wird. Mithilfe der If-Anweisung wird ausgeschlossen, dass die Distanz zum Hero ausgerechnet wird. Nun wird

die For-Schleife mit den Abständen erweitert und anschließend sortiert. Zeile 7 und 8 dienen dazu, dass nur Fahrzeuge, die sich näher als 200 Meter zu unserem Hero befnden, weiter beachtet werden. Abschließend wird noch die Anzeige in Zeile 11 mit den dazugewonnenen Informationen erweitert, welche den Abstand der Actoren zum Hero anzeigt.

5.3 Geschwindigkeit

Der nächste Schritt in der Simulation ist die Berechnung der jeweiligen Geschwindigkeit der Fahrzeuge durch eigene Positionsänderungen. Dafür könnte man das Velocity-Attribut von CARLA-Fahrzeugen verwenden. Diese Werte werden aber erst in späteren Funktionen zur Vereinfachung genutzt. Zuallererst wird die Geschwindigkeit ausgerechnet, damit die Rechnung sowie das Ergebnis außerhalb von CARLA verwendet werden kann. Nun werden von dem gewünschten Fahrzeug die Koordinaten, getrennt nach Achsen, in temporären Variablen gespeichert. Im Anschluss wird eine bestimmte Zeit abgewartet, die in diesem Fall bei einer Sekunde liegt. Die Skalierung der Geschwindigkeit ist in diesem Fall Meter pro Sekunde. Diese Zeit erhalten wir, indem innerhalb der CARLA Simulation die Ticks von einer Sekunde abgewartet werden. Im Anschluss wird die aktuelle Position mit der gespeicherten verglichen. Dies wird erreicht, indem wir die neue Position einer Achse vom vorherigen abziehen, womit wir die Positionsänderung einer Sekunde erhalten. Da die Koordinaten auch negativ sein können, müssen die Differenzen noch einmal im „Betrag“ genommen werden, weil ansonsten eine negative Geschwindigkeit rechnerisch herauskommen könnte. Das Koordinatensystem von CARLA ist bereits metergenau. Somit ist das Resultat der Betrag der Geschwindigkeit der jeweiligen Achse. Innerhalb des Straßenverkehrs sprecht man nicht von Achsengeschwindigkeiten, sondern von der allgemeinen Geschwindigkeit eines Fahrzeugs. Diese kann mit zwei verschiedenen Methoden berechnet werden, die im Grunde gleiche Ergebnisse liefert.

Die erste Methode mathematisch ausgedrückt:

$$v = \left| \sqrt{(0 - Ax)^2 + (0 - Ay)^2 + (0 - Az)^2} - \sqrt{(0 - Bx)^2 + (0 - By)^2 + (0 - Bz)^2} \right|$$

v = Allgemeine Geschwindigkeit des Fahrzeugs in $\frac{m}{s}$

$A_{x,y,z}$ = Alte Position mit jeweiliger Achsen Koordinate

$B_{x,y,z}$ = Neue Position mit jeweiliger Achsen Koordinate

In der zweiten Methode fasst man die zuvor ausgerechneten Achsengeschwindigkeiten zu einer neuen, allgemeinen Geschwindigkeit zusammen, indem man wieder den Euklidischen Abstand zu 0 nimmt, was in diesem Fall aber nicht der Ursprung ist, sondern der Ursprung der Geschwindigkeiten ist.

$$v = \sqrt{(vx - 0)^2 + (vy - 0)^2 + (vz - 0)^2}$$

v = Allgemeine Geschwindigkeit des Fahrzeugs in $\frac{m}{s}$

v_x= Geschwindigkeit (x-Achse)

v_y= Geschwindigkeit (y-Achse)

v_z= Geschwindigkeit (z-Achse)

Resultierend haben wir nun von jedem Fahrzeug die Geschwindigkeit der einzelnen Achsen und die allgemeine Geschwindigkeit, die wir in zukünftigen Methoden nutzen können. Allein gibt diese keinen Aufschluss auf die Relevanz der Autos zueinander.

5.4 Fahrtrichtung

Mit zwei kurzen If-Anweisungen kann die Fahrtrichtung der Fahrzeuge jeweils bestimmt werden. Zuerst wird Nord oder Süd bestimmt, was bedeutet, dass Fahrzeuge gerundet auf der y-Achse mehr in positive Richtung oder in die negative Richtung fahren. Die x-Achse drückt die Fahrtrichtungen Westen und Osten aus. Wenn der Wert der Achse gegen null geht, fährt das Auto in keiner dieser Richtungen. Die Fahrtrichtung steht fest, sobald die Ergebnisse beider Anweisungen kombiniert wurden.

```
    if round(vehicle.get_velocity().y,1) > 0:
        richtung = "S"
    elif round(vehicle.get_velocity().y,1) < 0:
        richtung = "N"
    else:
        richtung = ""

    if round(vehicle.get_velocity().x,1) > 0:
        richtung += "E"
    elif round(vehicle.get_velocity().x,1) < 0:
        richtung += "W"
    else:
        richtung += ""
```

Code 3: Richtung Methode

5.5 Abstandsänderungsvorhersage

Mit der Geschwindigkeit und dem Abstand aus den beiden vorherigen Kapiteln, kann nun vorhergesagt werden, wie wahrscheinlich sich die betrachteten Fahrzeuge in der Zukunft relativ zum Hero bewegen werden. Hier lässt sich zwischen zwei Konzepten entscheiden. einmal nämlich entweder über die Distanzänderungen in der Zukunft oder über die relative Beschleunigung zwischen den beiden ausgewählten Fahrzeugen.

Für das Konzept der Distanzänderung werden im ersten Schritt die Achsenkoordinaten der aktuellen Position des Actors mit dessen jeweiligen Achsengeschwindigkeit verrechnet.

$$Bx = Ax + \frac{vx}{t}$$

$$By = Ay + \frac{vy}{t}$$

$$Bz = Az + \frac{vz}{t}$$

$A_{y,x,z}$ = Achsenkoordinate der alten Position

$B_{x,y,z}$ = Achsenkoordinate der neuen Position

$V_{x,y,z}$ = Achsengeschwindigkeit

t = Einheitlich ausgewählte Zeitspanne

Somit ist das Resultat nach der Berechnung der aktuellen Geschwindigkeit die Koordinaten eines zukünftigen Punktes, der genau t Sekunden von der jetzigen Position des Actors entfernt ist. Im zweiten Schritt, nachdem die neuen Koordinaten beider Autos (Actor und Hero) berechnet wurden, werden mit diesen eine Distanz zwischen den neuen Punkten mit dem sogenannten Euklidischen Abstand berechnet. Im Anschluss wird diese von der realen Distanz abgezogen. Dadurch resultiert eine Distanzänderung während der Zeitspanne t.

Mathematische Distanzänderung ausführlich:

$$d_\alpha = d_{old} - \sqrt{\begin{matrix} ((Bx2_{old} + \frac{Bvx}{t}) - (Ax1_{old} + \frac{Avx}{t})^2 \\ +((By2_{old} + \frac{Bvy}{t}) - (Ay1_{old} + \frac{Avy}{t})^2 \\ +((Bz2_{old} + \frac{Bvz}{t}) - (Az1_{old} + \frac{Avz}{t})^2 \end{matrix}}$$

$$d_\alpha = d_{old} - \sqrt{(Bx2_{new} - Ax1_{new})^2 + (By2_{new} - Ay1_{new})^2 + (Bz2_{new} - Az1_{new})^2}$$

A = Actor Wert

B = Hero Wert

$v_{x,y,z}$ = Achsengeschwindigkeit

x_i, y_i, z_i = Achsenkoordinaten

$_{old}$ = Alte Punkte

$_{new}$ = Neue Punkte

d_{old} = Alte Distanz zwischen Hero und Actor

d_α = Distanzänderung

t = Einheitlich ausgewählte Zeitspanne

In dieser Arbeit wird für t eine Sekunde gewählt, da dieser Wert die Rechnungen damit vereinfacht. Wird dieser Wert jedoch zu groß gewählt, verliert dieser an Wichtigkeit. Je größer die Zeitspanne, desto unwahrscheinlicher ist es, dass die Geschwindigkeiten der Fahrzeuge in gleicher Zeitspanne identisch bleiben. Somit weicht das Ergebnis der Hervorsage umso mehr von der Realität ab, je größer t ist. Wird im Gegensatz dazu t zu klein gewählt, so kommt das

Fahrzeug zwar näher an die Realität, jedoch verliert das Ergebnis an Aussagekraft. Der Mensch kann sich unter einer Abstandsänderung von einer Millisekunde nichts vorstellen. Anschließend wird d_α in eine Methode mit Switch if else Anweisungen gesetzt, die Anhand der Größe von d_α mehrfach Minus oder Plus anzeigt. Die Methode befindet sich im Anhang A1.

5.6 TTC (Time To Collision)

Um die TTC-Methoden zu implementieren, wurde zur Hilfe eine zusätzliche eine Fahrzeugklasse erstellt.

5.6.1 Car Klasse

```
1   class Car:
2       def __init__(self, pos, length, v, compass):
3           self.pos = pos
4
5           self.length = length
6
7           self.v = v
8
9           self.new = Cord(pos.x + v.x, pos.y + v.y)
10
11          self.v_dis = distance(pos, self.new)
12
13          if(self.v.x == 0 and self.v.y == 0):
14              angle = compass
15          else:
16
17          self.angle = angle
18          self.corner_A = rotate(pos, Cord(pos.x - length.x / 2, pos.y + length.y
            / 2),  math.radians(angle))
19
20          self.corner_B = rotate(pos, Cord(pos.x - length.x / 2, pos.y - length.y
            / 2),  math.radians(angle))
21
22          self.corner_C = rotate(pos, Cord(pos.x + length.x / 2, pos.y - length.y
23          / 2),  math.radians(angle))
24
25          self.corner_D = rotate(pos, Cord(pos.x + length.x / 2, pos.y + length.y
            / 2),  math.radians(angle))
26
27          self.direction = dire(v)
```

Code 4: Car Klasse

Diese Klasse enthält folgende Attribute eines Fahrzeugs:

pos_x,y,z = Die x,y,z Position des Fahrzeugs mit der CARLA-Funktion auf Grundlage von Kapitel 5.0

length_x,y = Die Länge und Breite des Fahrzeugs aus CARLA.

v_x,y = Die Geschwindigkeit auf den beiden Achsen aus Kapitel 5.2

new_x,y = Ein temporärer Punkt in der Zukunft des Fahrzeugs, der in einer Sekunde ohne Geschwindigkeitsänderung erreicht wird.

v_dis = Distanz zwischen dem neuen und alten Punkt

angle = Der Winkel, in welche Richtung sich das Objekt bewegt

corner_A,B,C,D_x,y = Sind die 4 Zweidimensionalen-Ecken des Fahrzeugs, die jeweils mit dem zuvor ausgerechneten Winkel um das Zentrum gedreht wird.

Direction = Die Fahrrichtung als Wort, die mit der Methode von 5.3 bestimmt wird

5.6.2 TTC-α-Implementierung

Nun zu den eigentlichen Methoden von TTC. Aufgrund der Länge befindet sich diese im Anhang dieser Arbeit. Als Erstes gibt die Methode TTC-α, die als Aufrufparameter 2 Car-Objekte von der zuvor angelegten Klasse übernimmt. Das erste Objekt ist immer der Hero und das zweite Objekt der jeweilige Actor, der mit dem Hero verglichen wird. Mit dieser Methode wird ein möglicher Schnittpunkt von der aktuellen Richtung sowie Geschwindigkeit der beiden Verkehrsteilnehmer bestimmt und überprüft, ob beide Fahrzeuge an diesem Punkt überhaupt aufeinandertreffen können. Das Ergebnis dieser Methode ist der TTC-α, welcher angibt, wie groß der Zeitunterschied von beiden Fahrzeugen zum existierenden Kollisionspunkt ist. Ist der Wert nicht bestimmbar, werden beide Fahrzeuge mit aktueller Fahrweise niemals kollidieren. Ist der Wert andererseits genau 0, werden beide Fahrzeuge genau gleichzeitig an dem Schnittpunkt ankommen, was zu einem Unfall führt. Ansonsten resultiert ein Sekundenwert, wie oft sich beide Autos in der Einheit: Zeit verpassen. Um diesen Wert zu bestimmen, muss zuallererst herausgefunden werden, wie sich beide Fahrzeugen zueinander verhalten und wie dies Auswirkungen auf den Treffpunkt haben könnte. Das Verhalten der Fahrzeuge kann in 3 Hauptkategorien mit eigenen Spezialfällen eingeteilt werden. In der ersten Hauptkategorie bewegen sich beide Fahrzeuge nicht. Das bedeutet, sie stehen aktuell im Stau oder parken auf einem Parkplatz. Damit ist die Rechnung komplett hinfällig, denn wenn sich beide Fahrzeuge nicht bewegen, werden sie auch niemals aufeinanderprallen. Das kann höchstens der Fall sein,

wenn sie sich schon gegenseitig angestoßen haben und aktuell am exakt gleichen Punkt stehen. In diesem Fall muss wiederum kein Zeitpunkt bis zum Zusammenstoß ausgerechnet werden, da dieser schon geschehen ist. Somit kann der Wert, wenn beide Fahrzeuge sich nicht bewegen, ohne viel Rechnung auf endlos gesetzt werden, sprich der Platzhalter für Endlos wurde im Programm das Char x verwendet. Dies wird überprüft, indem dass beide Objekte keine Achsengeschwindigkeit besitzen. Somit ist zugleich die Kategorie auch schon abgeschlossen.

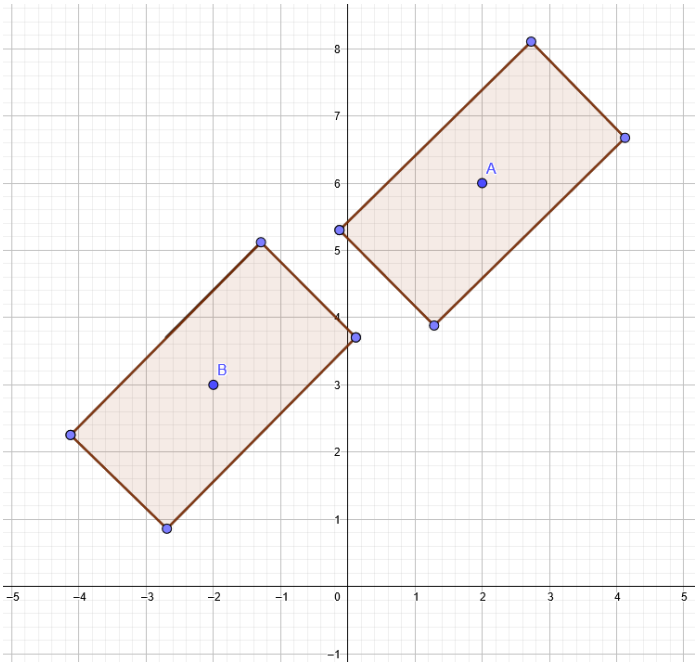


Abbildung 7: Beispiel von zwei stehenden Fahrzeugen
Quelle: [Selbst erstellt]

Die zweite Kategorie befasst sich damit, dass eines der beiden Fahrzeuge sich bewegt und das andere nicht. In diesem Fall muss untersucht werden, ob das fahrende Fahrzeug sich auf einer Linie in der Richtung des stehen Fahrzeugs befindet. Sollte das der Fall sein, wird das Resultat ohne Richtungsänderung auf jeden Fall zu einem Unfall führen. Fährt das fahrende Auto jedoch nicht direkt zum stehenden, werden beide Fahrzeuge niemals einen Aufprall provozieren.

Dies wird mit der Cross-Methode getestet: siehe Anhang A3

Am Anfang dieser Methode werden die Seiten von dem fahrenden Fahrzeug mithilfe einer Gerade in Bewegungsrichtung verlängert, bis sie auf eine der beiden diagonalen Geraden des stehenden Fahrzeugs treffen. Somit resultieren zwei Schnittpunkte. Im Anschluss wird die

Methode überprüft, ob mindesten einer der beiden berechneten Punkte, zwischen den beiden Eckpunkten des stehenden Fahrzeugs liegen. Trifft dieser Fall ein, wird das fahrende Fahrzeug irgendwann in der Zukunft in das stehende Fahrzeug hineinfahren.

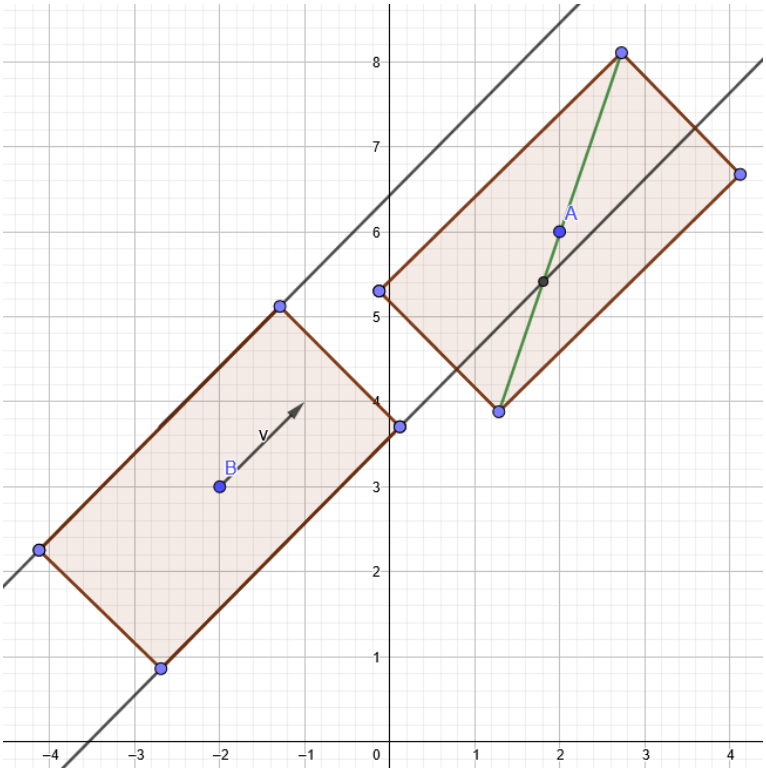


Abbildung 8: Beispiel eines fahrenden Fahrzeugs
Quelle: [selbst erstellt]

In der letzten Kategorie bewegen sich beide Fahrzeuge. Am Anfang wird der benötigte Schwerpunkt in dieser Kategorie berechnet. Hierfür wird in der Bewegungsrichtung wieder eine Gerade, ausgehend vom Zentrum, gezogen. Nun werden die beiden Geraden miteinander verglichen. Die beiden Geraden können entweder genau gleich sein, parallel zueinander liegen oder sich schneiden. Für den Fall, dass nur eine Gerade existiert, muss überprüft werden, ob beide Fahrzeuge auf einer Linie fahren, was bedeutet, dass sie auf einer geraden Straße hintereinanderfahren. Dies wird überprüft, indem geschaut wird, ob die Mittelpunkte der Fahrzeuge sowie beide zukünftigen Punkte sich auf einer Geraden befinden. Dies erreicht man mit der Linienmethode, welche die drei Ortspunkte erhält. Die Methode ermittelt von jedem Punkt zu den jeweils beiden anderen Punkten, die Steigung aus. Sind alle drei Steigungen gleich, sind alle drei Punkte auf einer Geraden. Da dies aber nur den Mittelpunkt der Autos

abdeckt und Autos in der Realität nicht genau mittig zueinander fahren, muss man zur Sicherheit die Cross-Methode von der vorigen Kategorie verwenden. Sollte einer der drei Bedingungen eintreffen, fahren Hero und Actor auf einer Linie. Das allein reicht jedoch noch nicht aus, um auszusagen, ob die beiden Fahrzeuge zusammenstoßen werden oder nicht. Nun muss die Richtung von den beiden Fahrzeugen beachtet werden. Fahren beide Fahrzeuge jeweils in die gleiche Richtung oder jeweils in zwei verschieden Richtungen. Falls beide Fahrzeuge in dieselbe Richtung fahren, muss bestimmt werden, ob das hinterherfahrende Fahrzeug schneller als das andere ist. Sollte das der Fall sein, werden beide Fahrzeuge irgendwann miteinander kollidieren. Wenn jedoch das hintere Fahrzeug langsamer ist, wird im besten Fall niemals eine Kollision stattfinden. Wenn beide Fahrzeuge in unterschiedliche Richtungen fahren, muss überprüft werden, ob die beiden Fahrzeuge aufeinander zu- oder voneinander wegfahren. Fahren sie aufeinander zu, ist das resultierende Ergebnis ein möglicher Zusammenstoß und im anderen Fall hat das eine Auto auf das andere keinen Einfluss.

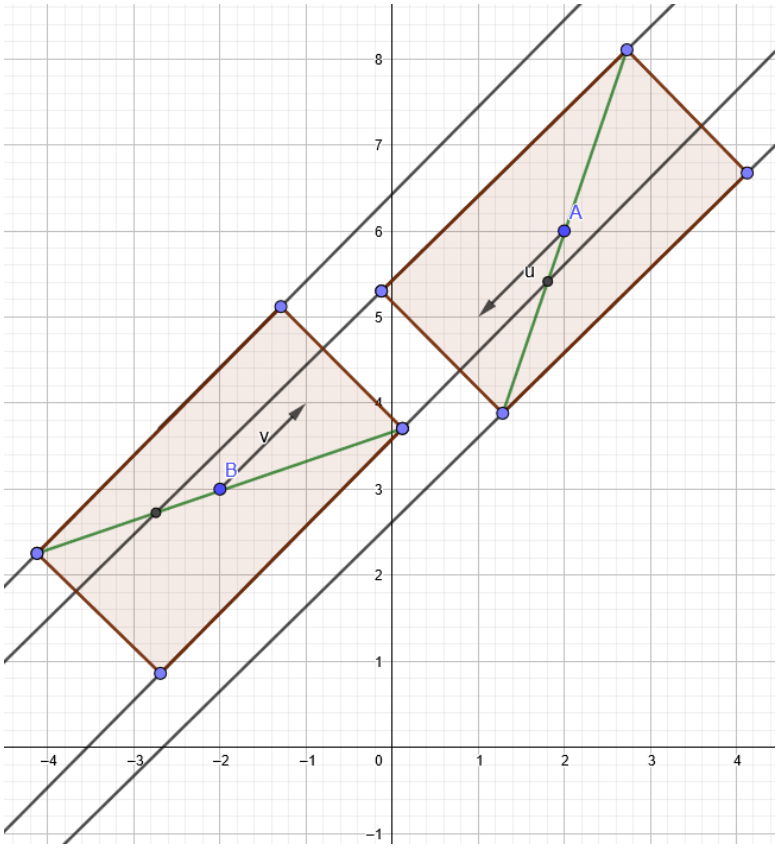


Abbildung 9: Beispiel zwei fahrende Fahrzeuge
Quelle: [selbst erstellt]

Fahren beide Fahrzeuge nicht auf einer Linie kann mit den Mittelpunkten und der Bewegungsrichtungen der jeweiligen Autos eine Gerade berechnet werden. Schneiden sich die beiden Geraden an einem Punkt ist dies der gesuchte Schnittpunkt.

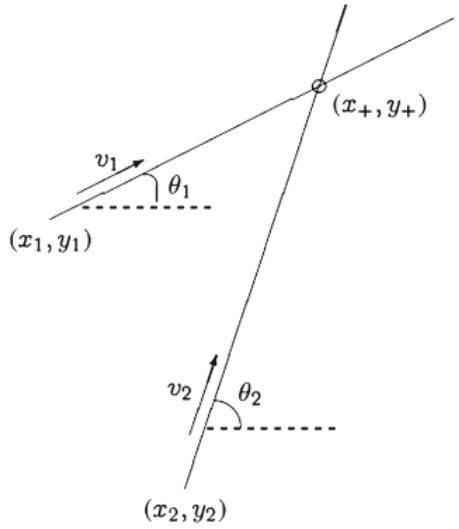


Abbildung 10: Mögliche Schnittpunktberechnung
Quelle: [9]

Sollte dieser Schnittpunkt nicht berechnet werden können, dann bewegen sich die Autos auf einer Geraden, was mit den vorherigen Fällen behandelt wurde oder die beiden Geraden verlaufen parallel, wodurch sich die Autos auch in den meisten Fällen nicht begegnen. Die möglichen Begegnungen werden ebenfalls oben mit der Cross-Methode ausgefiltert. Sollte die Schnittpunktberechnung nicht möglich sein, kann davon ausgegangen werden, dass die Autos nicht auseinanderfahren werden. Wird der Schnittpunkt richtig berechnet, kann nun der eigentliche TTC-α Wert berechnet werden. Davor muss jedoch noch einmal überprüft werden, ob beide Fahrzeuge in die Richtung des entsprechenden Punktes fahren. Trifft dies zu, so werden die Einzelzeiten der beiden Fahrzeuge zum Treffpunkt hin berechnet. Der gesuchte TTC-α-Wert ergibt sich aus dem Zeitunterschied. Die Relevanz des Zeitunterschieds α wird im dazugehörigen Kapitel: Evaluierung weiter beschrieben. Nachdem der α Wert bestimmt wurde, kann nun die TTC-Berechnung über die nächste Methode vom folgenden Kapitel erfolgen. Diese Methode gibt am Ende entweder das Ergebnis der TTC-Methode als Zeit zurück oder abhängig von den Spezialfällen die Zeit 0, wenn die Autos sicher in dem Fall aufeinander fahren oder das Char x für eine unendliche Zeit, dass mit den Fahrzeugen nichts passieren wird.

5.6.3 TTC-Implementierung

Um den realen TTC-Wert zu bekommen, reicht die Berechnung des α Werts aus der vorigen Methode nicht aus. Hierzu benötigt man eine zusätzliche Methode, nur TTC genannt, welche sich ebenfalls im Anhang dieser Arbeit befindet. Bei dieser werden in Bewegungsrichtung die einzelnen Eckpunkte der Autos geschnitten. Hierzu gibt es zwei Hauptfälle zu beachten. Einmal dass der Schnittwinkel der beiden Halbgeraden von der vorherigen Methode einmal zu anderen Ergebnissen führt, wenn der Winkel kleiner oder größer als 90° ist.

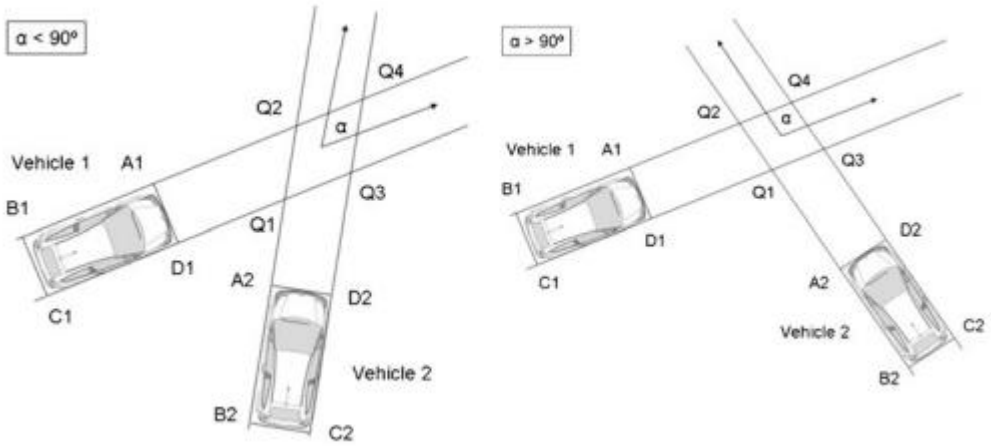


Abbildung 11: TTC-Beispiele
Quelle: [9]

Nun werden die Zeiten von jedem Eckpunkt beider Fahrzeuge zu den jeweils dazugehörigen Q-Schnittpunkten aus der Abbildung ausgerechnet. Sollte der Winkel kleiner als 90° sein, können nur 6 verschiedene Zusammenstöße geschehen, wie in der unteren Abbildung zu sehen ist.

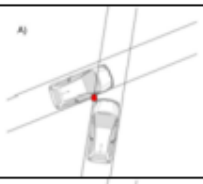
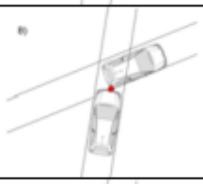
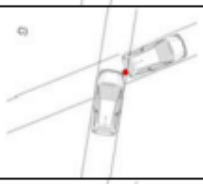
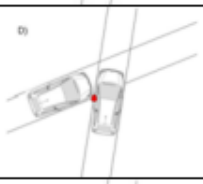
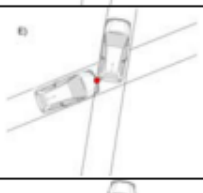
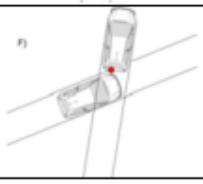
	Situation	Diagram	Time conditions (Situation in the event the condition is not satisfied)	TTC
A)	Corner Vehicle 2 hits side Vehicle 1		TC11 > TA21 > TD11 B D	TA21
B)	Front part Vehicle 2 hits corner Vehicle 1		TC13 > TD23 > TA21 > TC11 C * A	(TA21, TD23)
C)	Corner Vehicle 2 hits rear part Vehicle 1		TB14 > TD24 > TD23 > TC13 X * B	(TD23, TD24)
D)	Corner Vehicle 1 hits side Vehicle 1		TB21 > TD11 > TA21 E A	TD11
E)	Front part Vehicle 1 hits corner Vehicle 2		TB22 > TA12 > TD11 > TB21 F * D	(TD11, TA12)
F)	Corner Vehicle 1 hits rear part Vehicle 2		TC24 > TA14 > TA12 > TB22 X * E	(TA12, TA14)

Tabelle 1: TTC-Fälle bei Winkel größer 90°

Quelle: [9]

Sollte der Winkel größer als 90° sein, können nur die folgenden 4 Zusammenstöße passieren.

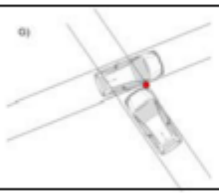
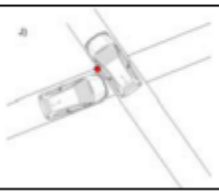
	Situation	Diagram	Time conditions (Situation in the event the condition is not satisfied)	TTC
G)	Corner Vehicle 2 hits side Vehicle 1		TC13 > TD23 > TD13 X H	TD23
H)	Corner Vehicle 1 hits front part Vehicle 2		TD13 > TD23 G; TA21 > TD11 I TA21 > TD23 * TD13 > TD11 *	(max(TD11,TD23), min(TA21, TD13)
I)	Corner Vehicle 2 hits front part Vehicle 1		TD11 > TA21 H; TA22 > TA12 J TD11 > TA12 * TA22 > TA21 *	(max(TA21,TA12), min(TD11, TA22)
J)	Corner Vehicle 1 hits side Vehicle 2		TB22 > TA12 > TA22 X I	TA12

Tabelle 2: TTC-Fälle bei Winkel kleiner 90°

Quelle: [9]

Diese 10 Situationen werden nun entsprechend mit mehreren if und elif Bedingungen nacheinander abgeprüft und anschließend, falls ein TTC-Wert bestimmt wurde, wird der kleinste davon in die TTC-α Methode zurückgegeben. Sollte es mit dieser Methode nicht möglich sein ein TTC-Wert zu bestimmen, so werden die Beiden Objekte mit dem jetzigen Stand nicht kollidieren und ein Char „X“ wird zurückgeben., welche dann in beiden Fällen auch dem Gesamtergebnis von beiden Methoden entspricht.

5.6.4 THW-Implementierung

Die THW-Methode bekommt nur 2 Cars als Übergabeparameter. Somit kann sie unabhängig von den anderen Methoden verwendet werden. Für die Methode muss nur beachtet werden, dass die Gesamtgeschwindigkeit von dem ausgehenden Fahrzeug nicht null ist. Sollte die Geschwindigkeit null sein, kann kein THW berechnetet werden und anstelle des Rechnungsergebnisses gibt die Methode dann ein Char „X“ zurück.

```
1  def THW(Car_A,Car_B):
2      if Car_A.v_dis!=0:
3          return round(distance(Car_A.pos, Car_B.pos) / Car_A.v_dis,2)
4      else:
5          return "X"
```

Code 5: THW-Methode

5.6.5 Risikogruppen-Methode

Diese Methode im Anhang A2 braucht die Ergebnisse von den vorigen zwei Methoden, nämlich einmal den TTC-Wert und einmal den THW. Am Anfang der Methode werden beide Übergabe-Parameter nacheinander überprüft, ob ein gültiger Float-Wert vorhanden ist. Sollte ein Wert nicht gültig sein, wird dieser auf einen hohen Wert gesetzt. Zusätzlich wird vor der Risikoberechnung von TTC über die Umkehrfunktion das inverse iTTC berechnet. Da iTTC, THW und einige andere Werte im Vergleich eine Gleitkommazahl darstellen, müssen diese vorher alle einheitlich 10^x genommen werden, damit aus allen eine ganze Zahl wird. Grund hierfür ist Python 3.7 bei welchem eine nicht mit dem mathematischen „größer (>)“ oder „kleiner (<)“ verglichen werden kann. Der Vergleich im Anschluss richtet sich beim Zuordnen des Risikos nach der Vorlage von:

Risk group	Risk level	Thresholds
High risk	RL9	$1.0 \leq$ iTTC
	RL8	$0.67 \leq$ iTTC < 1.0
	RL7	$0 \leq$ iTTC < 0.67 & THW < 0.9
⇓	RL6	$0 \leq$ iTTC < 0.67 & $0.9 \leq$ THW < 1.3
	RL5	$0 \leq$ iTTC < 0.67 & $1.3 \leq$ THW < 1.8
	RL4	$0 \leq$ iTTC < 0.67 & $1.8 \leq$ THW < 2.5
Low risk	RL3	iTTC < 0 & THW < 2.5
	RL2	$0 \leq$ iTTC < 0.67 & $2.5 \leq$ THW
	RL1	iTTC < 0 & $2.5 \leq$ THW

Abbildung 12: Risiko Grenzen

Quelle: [8]

6. Evaluierung

In diesem Kapitel werden die Ergebnisse der einzelnen Methoden präsentiert.

6.1 Abstand

Für die Überprüfung, ob der Abstand richtig berechnetet wird, wurde die Position aller Fahrzeuge in ein Koordinatensystem übergeben. Nun wird geschaut, welche Informationen der Hero erhält und vergleicht diese mit dem Stadtbild.

Abbildung 13: Client Ansicht Abstand

Abbildung 14: Abstandsbeispiels Koordinatensystem mit Fahrzeugen

Tabelle 3: Abstandsvergleich zwischen Client-Ansicht und Koordinaten-System
Quellen: [selbst erstellt]

Um den Vergleich übersichtlicher zu gestalten, wurden die Fahrzeuge in diesem Beispiel jeweils mit der gleichen roten ID in der sortierten Liste sowie im Koordinatensystem gekennzeichnet. Bei der Gesamtübersicht ist es schwierig zu erkennen, wo sich die Harley (Nummer 1) mit 4 m Abstand befindet. Wenn die Stelle vergrößert angeschaut wird, erkennt man genau, dass die Harley östlich (in Abb. links) vom Hero ist.

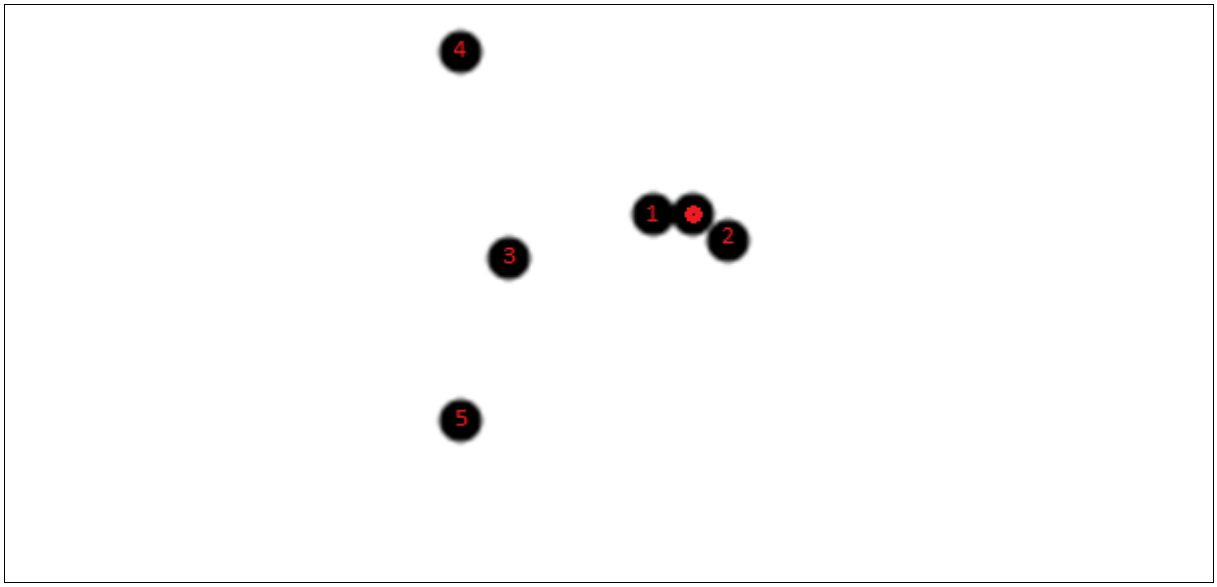


Abbildung 15: Abstands-Beispiel Zoomausschnitt
Quelle: [selbst erstellt]

Zusätzlich kann mit dem Zoom-Bild gut erkannt werden, dass die Abstandsangaben im Client richtig sind. In der Liste steht, dass der Tesla Truck (Nummer 3) 21 m vom Hero entfernt ist. Da die Harley 4 m vom Hero entfernt ist, bedeutet es, dass die Nummer 3 ca. 5-mal so weit weg sein muss, wie die Nummer 1. Das dies so sein muss, kann sehr gut auf dem Zoom erkannt werden. Diesen bildlichen Vergleich kann mit verschieden Nummer gemacht werden.

6.2 TTC-α-Evaluierung

Um zu testen, ob die TTC-α Methode sich richtig verhält, werden verschiedene Situationen aus der Simulation angeschaut, in welcher verschiedene Fälle der Methode zu erkennen sind. Es ist zusätzlich zu beachten, dass TTC-α nur mit der Position, Richtung und Geschwindigkeiten der Verkehrsteilnehmer arbeitet und nicht den Fahrbahnverlauf beachtet.

Die erste Situation wurde zufällig gewählt, um zu sehen, welche Ergebnisse die TTC-α Methode zurückgibt, ohne auf gezielte Ergebnisse zu achten. Zur besseren Übersicht werden die Ergebnisse anhand ihrer Zugehörigkeit in den nächsten Abbildungen mit unterschiedlichen Farben umrandet.

Die verschiedenen Kategorien der Zugehörigkeit sind:

Farbe	**Kategorie**	**Mögliche Ergebnisse**	
		Kein Zusammenstoß	**Zusammenstoß gefunden**
Grün	Kein gültiger Treffpunkt	X 8 -	
Organe	normaler TTC-Fall	{nicht möglich}	[Zahl] [Buchstabe] [Zahl]
Blau	nur TTC-α, kein TTC	{nicht möglich}	[Zahl] X X
Gelb	mindestens einer der beiden Verkehrsteilnehmer steht (keine Geschwindigkeit)	X [0, 1, 2] -	0 [1, 2] [Zahl]
Grau	Kein Treffpunkt bestimmbar	X 9 -	{nicht möglich}
Lila	Beide Verkehrsteilnehmer fahren auf einer Linie	X [3,4,5,6,7] -	0 [Zahl[op, sa,-sa]] [Zahl]

Ergebnis Aufbau: „TTC-α“ „Fall“ „TTC“ X=∞

Tabelle 4: TTC-α Kategorie Einteilung
Quelle: [selbst erstellt]

Abbildung 16: TTC-α Beispiel: Situation 1 Normalverkehr Town10
Quelle: [selbst erstellt]

Um die Methode besser testen zu können, werden von jedem Hero zusätzliche Information für die Überprüfung angezeigt. Dadurch wird die Client Anzeige ein bisschen unübersichtlicher. Dies ist aber aufgrund der notwendigen Informationen vonnöten, zumal in der ersten Situation noch nicht alle möglichen Kategorien vorkommen. Es werden zusätzliche Situationen erfasst, bis jede Kategorie mindestens zwei Mal vorkommt. Anschließend werden die verschiedenen Kategorien nacheinander überprüft, ob die Ergebnisse passen. In diesem Kapitel wird die orange Kategorie erstmal nicht weiter beachtet, da diese ein Sonderfall ist und nicht allein mit der TTC-α Methode ausgerechnet wird. Um Zeit zu sparen, wird bei der Erstellung der restlichen Situation nicht auf gut Glück beobachtet, bis so viele neue Ergebnisse wie möglich auftreten. Stattdessen wird eine der fehlenden Kategorien erfasst, aufgezeichnet und danach die Simulation neugestartet. Situation 2 (Anhang A4) dient für die drei lilafarbene Fälle sowie auch als Test, ob die Methode auch genauso auf der anderen Karte ohne Probleme funktioniert. Die grünen Fälle wurden schon mit der ersten Situation getestet und werden in der zweiten ignoriert. Der einzelne blaue Fall wird auch erst einmal übersprungen, da dieser in anderen Situationen häufiger als nur einmal vorkommen kann. Situation 3 (Anhang A5) dient nur um die blaue Kategorie abzuschließen. Für die Situation 4 (Anhang A6) wurde der Hero angehalten, sodass dieser für die gelbe Kategorie. Diese passiert in CARLA nicht so häufig, da die Autos in der

Simulation auch im Sillstand sich noch langsam bewegen, so als wäre wie in der Realität der Motor an. Um die Kategorie noch besser zu testen, wurde dies auf genau einer Kreuzung getestet.

6.2.1 Grüne Kategorie „Kein gültiger Treffpunkt“

In Situation 1 sieht man viele grün gekennzeichnete „X 8 -“ Fälle. Diese Kategorie kommt im normalen Straßenverkehr auch am häufigsten vor. Es geht dabei darum, dass ein möglicher Schnittpunkt zwischen dem Hero und Actor berechnet wurde, dieser aber aufgrund einer der beiden Teilnehmer nicht als Kollisionspunkt eintreten kann. Um dies besser aus der Situation 1 auslesen zu können, werden der Hero und die betroffenen Actoren *E*, *F*, *G*, *I*, *J* und *K* in GeoGebra in einem Koordinatensystem besser dargestellt.

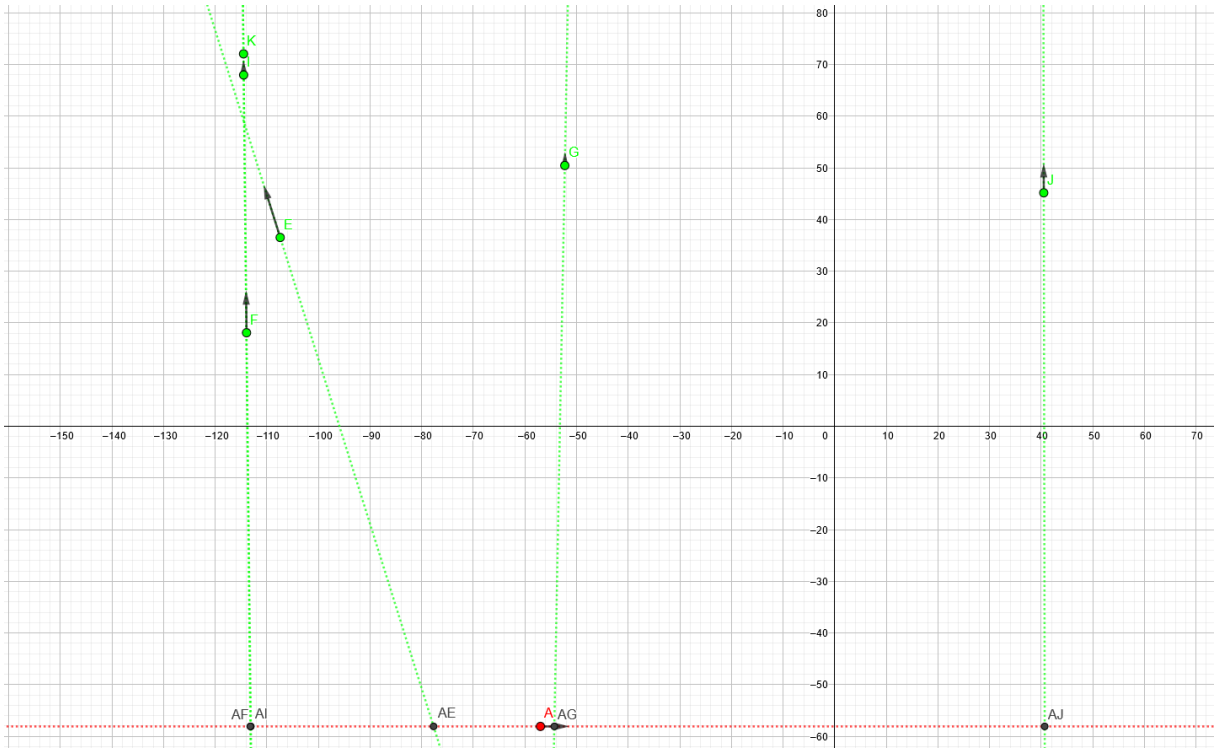


Abbildung 17: TTC-α Methode Grüne Kategorie Übersicht aus Situation 1
Quelle: [selbst erstellt]

Man erkennt auf den ersten Blick, dass der mögliche Treffpunkt von *AF*, *AI* und *AG* in der Vergangenheit vom Hero *A* liegt, wobei zusätzlich alle Actoren sich vom Treffpunkt weiter entfernen. Als nächstes ist zu erkennen, dass der mögliche Treffpunkt von *G* und *J* zwar in der

Zukunft von Hero *A* liegen, aber die beiden Actoren ebenfalls vom Treffpunkt wegfahren. Die möglichen Schnittpunkte von *AK* kann man nicht allein aus den Positions- und Geschwindigkeitsdaten im Koordinaten-System anzeigen, da die Geschwindigkeit von *K* so klein ist, dass die Anzeige technisch auf 0.0 gekürzt wurde. Um trotzdem einen möglichen Schnittpunkt zu errechnen, muss sich die Ausrichtung von K angeschaut werden, nämlich anhand der Eckpunkte.

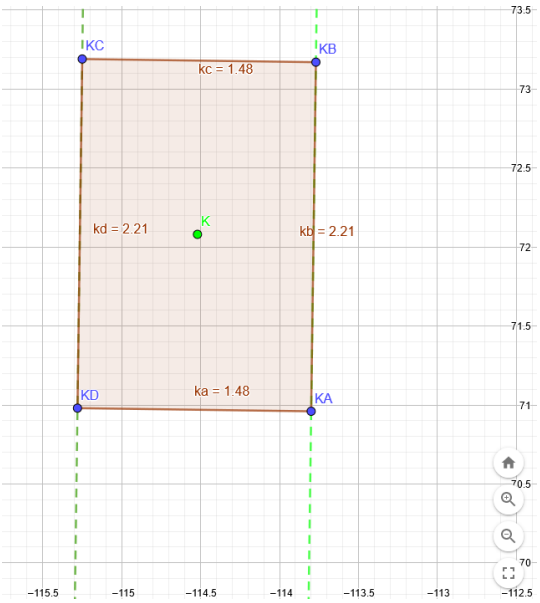


Abbildung 18: Zoom auf Auto K in Situation 1
Quelle: [selbst erstellt]

In Abb. 23 sieht man, dass die Fahrzeugfront *ka* nach unten Richtung Norden zeigt. Somit nähert sich *K* zwar dem möglichen Schnittpunkt an, wenn auch langsam. Aber Hero *A* entfernt sich von diesem. Damit sind alle Schnittpunkte ungültig, womit alle Actoren mit dem Ergebnis „X 8 -“ richtig berechnetet und zugeordnet wurden.

6.2.2 Graue Kategorie „Kein Treffpunkt bestimmbar“

In Situation 1 sieht man noch zwei Actoren *B* und *C* mit dem Ergebnis „X 9 -“. Dieses Ergebnis tritt nur ein, wenn die mögliche Schnittpunktberechnung nicht richtig durchgeführt werden kann. Im Normalfall tritt diese nur ein, wenn in der Schnittpunktberechnung durch null geteilt wird, was mathematisch nicht gemacht werden darf. Dies passiert in dem Fall, wenn die Richtung von Hero und überprüften Actor genau parallel sind. Dies kann man wieder gut im Koordinaten-System überprüfen.

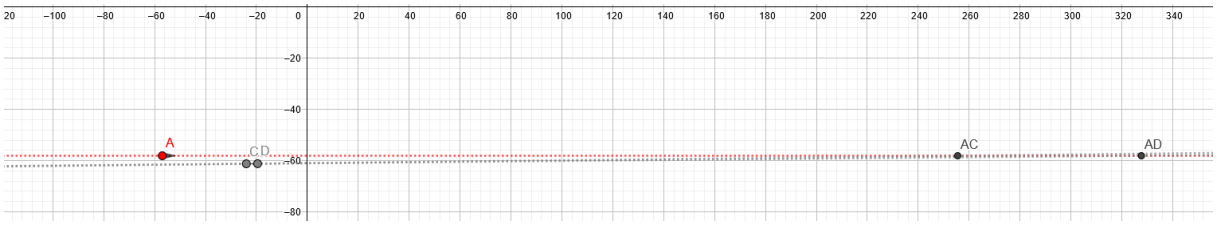


Abbildung 19: TTC-α Methode Graue Kategorie Übersicht aus Situation 1
Quelle: [selbst erstellt]

Wie sich aus obenstehender Abbildung erkennen lässt, verlaufen die Richtungen von C und D nicht parallel zur Hero-Richtung, sondern schneiden diese sogar in der Zukunft. So gesehen gehören die beiden Actoren nicht zu dem Ergebnis „X 9 -“. Sie werden trotzdem diesem zugeteilt, da die möglichen Schnittpunkte ein zu großes TTC-α ergeben und somit die TTC-Berechnung überspringen. Da es so kein Return findet, landet es in Fall 9. Dies war als Testfall absichtlich geplant, indem der passende Return vorher ausgeklammert wurde und in den folgenden Situationen, landen diese Fälle in der richtigen blauen Gruppe. Aber dieser Fehlerfall beweist, dass der Fall 9 zusätzlich als Notfallplan für unbedachte Fälle funktioniert.

6.2.3 Lila Kategorie „eine gerade Linie“

Wenn der Hero und verglichene Actoren genau auf einer Linie fahren, kann ebenfalls kein möglicher Schnittpunkt berechnet werden. Es kommt entweder zu einem unendlichen α oder einem α mit Nullwert. Daher fangen die Ergebnisse dieser Kategorie entweder mit 0 oder X an. Danach im Grundteil geht es mit einer Zahl los, welche nur zur besseren Zuordnung der Position in dem Programm-Code dient.

Die folgenden Buchstaben sagen den wirklichen Grund aus:

Fall	Bedeutung
sa	Actor und Hero in gleiche Richtung: Hero.v > Actor.v
-sa	Actor und Hero in gleiche Richtung: Hero.v < Actor.v
=sa	Actor und Hero in gleiche Richtung: Hero.v = Actor.v
op	Actor und Hero fahren in verschiedene Richtungen

Tabelle 5: TTC-α Methode Lila Kategorie Fall Bedeutung
Quelle: [selbst erstellt]

Anschließend kommt dann abhängig vom α Wert bei 0 noch der TTC-Wert. Andernfalls steht am Ende nur ein „–“ für kein TCC.

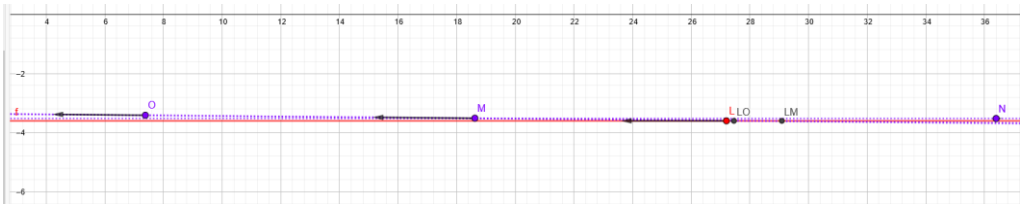


Abbildung 20: TTC-α Methode Lila Kategorie Übersicht aus Situation 2
Quelle: [selbst erstellt]

Im Koordinatensystem in oben aufgeführter Abbildung lässt sich erkennen, dass sich alle drei Actoren nicht genau auf der Richtungsgeraden des Hero befinden. Die Richtung von N verläuft sogar genau parallel, dass es keinen Schnittpunkt zwischen den Richtungen geben kann. Die drei Actoren liegen keine 0,25 m von der Bewegungsrichtung des Heros entfernt. Mit den 1,8 m der Hero-Breite liegen die Actoren bereits im Hero-Bereich. Somit wird angenommen, dass die Fahrzeuge erst mal auf einer Linie fahren werden. Zu *N* gibt es kein TTC, da es zu langsam hinter dem Hero *L* nachfährt. Bei *M* und *O* wird eine TTC-Zeit berechnet, weil aktuell *L* mehr Distanz auf der Geraden in die Richtungen der beiden zurücklegt. Dieser Wert stimmt aber auch nur, wenn die Verkehrsteilnehmer weiter auf einer Linie fahren. Es kann sein, dass nach dem berechneten TTC-Wert die Actoren mit ihrer aktuellen Richtung schon nicht mehr in Kollisionsbereich sind.

6.2.4 Blaue Kategorie „nur TTC-α“

In diese Gruppe kommen alle Actoren, die zwar für einen möglichen Treffpunkt immer einen TTC-α Wert besitzen, jedoch keinen TTC-Wert. Entweder ist α zu groß, wie in absichtlich falsch getesteten Fällen von Situation 1, da es keinen Sinn ergibt überhaupt zu prüfen, ob es einen Zusammenstoß gibt. Auf der anderen Seite kommt es in der Realität nur knapp zu keinem Zusammenstoß, da α groß genug ist, um sich nicht zu berühren.

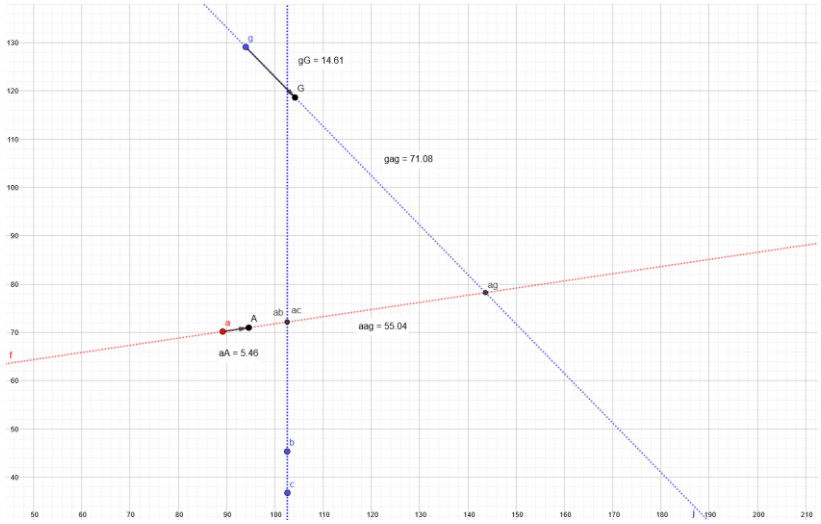


Abbildung 21: TTC-α Methode Blaue Kategorie Übersicht aus Situation 3
Quelle: [selbst erstellt]

Bei diesen drei Actoren lässt sich aus oben stehender Abbildung ableiten, warum es keine TTC-Zeit geben kann. Hero *a* ist viel schneller an dem gleichen Treffpunkt als die Actoren *b* und *c*. Bei Actor *g* sieht es so aus, als würde dieser vor Hero *a* am Schnittpunkt sein. Um das nachvollziehen zu können, sollte man das TTC-α Ergebnis einmal überprüfen. Dazu wurde zusätzlich in Abb. 21 für Actor *g* die passenden Abstände mit angegeben.

$$\text{TTC-}\alpha = \left|\frac{55{,}04m}{5{,}46\frac{m}{s}} - \frac{71{,}08m}{14{,}61\frac{m}{s}}\right| = 5{,}215\text{s}$$

Kurzfassung der Rechnung von Kapitel 4.3.2 TTC-α

Wenn man das Ergebnis der Rechnung mit Wert des Programms vergleicht, fällt auf, dass man nicht auf exakt das Richtige kommt, sondern um ca. 0,1s daneben. Das kommt daher, dass nur gerundete Werte angezeigt werden und nicht genau die exakten, die das Programm verwendet. Aber man kann mit der Nachrechnung trotzdem sehen, ob die Werte im richtigen Bereich

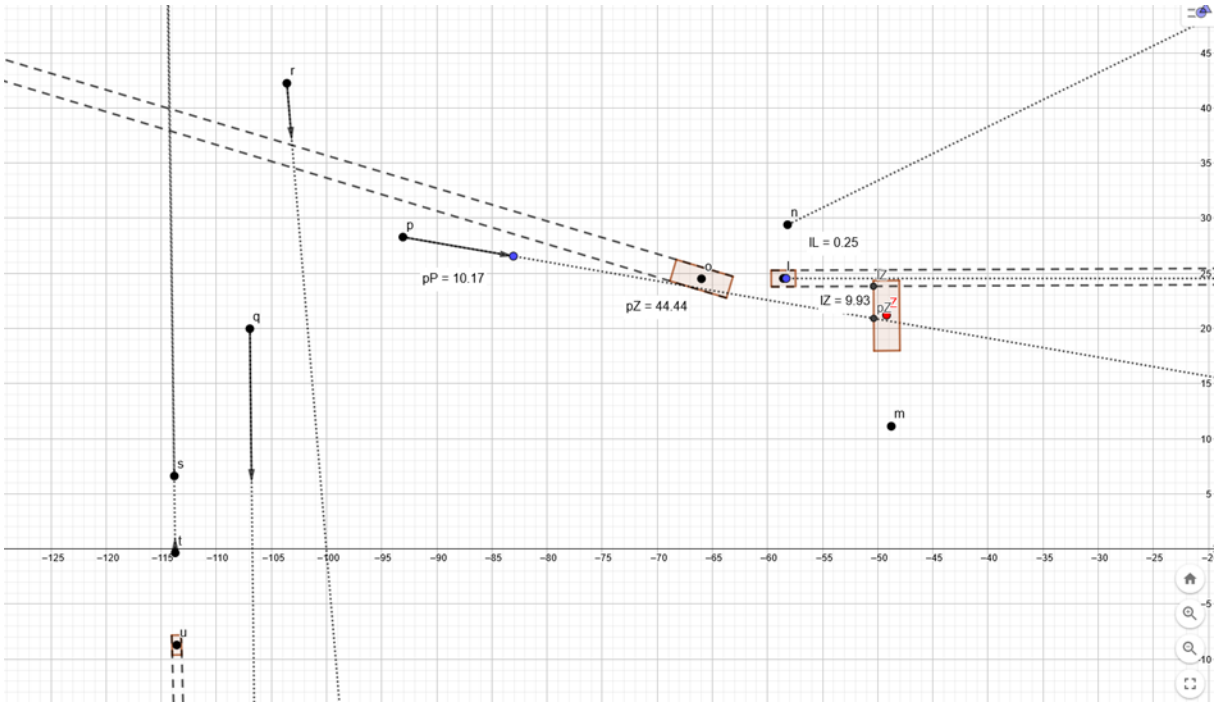
liegen. Mit einem α von 5,3s kann man davon ausgehen, dass es in dieser Situation zu keinem richtigen TTC-Fall kommt, wodurch dieser Actor in dieser Gruppe richtig ist.

6.2.5 Gelb Kategorie „Wartende oder parkende Verkehrsteilnehmer“

In der letzten normalen Kategorie kommen alle Ergebnisse mit dem Fall 0, 1 oder 2. Mit Fall 0 ist gemeint, dass sich beiderseits Hero und Actor nicht vom ihrem Standort bewegen, sprich beide haben keine Geschwindigkeit in irgendeine Richtung. Dadurch ist es unmöglich TTC-α zu berechnen, da so es niemals zu einem Unfall kommen kann. In den Fällen 1 oder 2 wird angenommen, dass einer der beiden Verkehrsteilnehmer sich nicht bewegt im Fall 1 der Hero und in Fall 2 der jeweils verglichene Actor. Sollte in beiden Fällen der jeweils fahrende Teilnehmer genau auf den stehenden fahren, kommt es zu einem hundertprozentigen Zusammenstoß, wodurch in diesen Fällen TTC-α 0 ist. Dazu wird noch berechnet, wie lange es dauert, bis eine Kollision eintritt. Sollte der fahrende Teilnehmer sich nicht auf den stillstehenden Teilnehmer zubewegen, so ist es unmöglich TTC-α zu berechnen.

	Kein Zusammenstoß	Zusammenstoß
Hero und Actor stehen	X 0 -	[nicht möglich]
Hero steht	X 1 -	0 1 TTC
Actor steht	X 2 -	0 2 TTC

Tabelle 6: TTC-α Methode Gelb Kategorie Legendentabelle
Quelle: [selbst erstellt]



(Um die Abb. besser lesen zu können wurde auf die Farbe Gelb verzichtet)

Abbildung 22: TTC-α Methode Gelb Kategorie Übersicht aus Situation 4

Quelle: [selbst erstellt]

Um die Richtung von Actor *o* und *u* zu bestimmen, muss man das reale Auto im Koordinatensystem darstellen und die Richtung mit den Seiten bestimmen, da in 4 Situation die Geschwindigkeit so klein ist, dass sie auf null gerundet ist. Es kann sein, dass wie vorhin beschrieben, der laufende Motor die Ursache ist. Somit sieht man, dass Actor *n*, *q*, *r*, *s*, *t* und *u* mit ihrer aktuellen Richtung den Hero *Z* nicht schneiden. Dies ist exakt gleiche Schlussfolgerung wie bei Ergebnis „X 1 –“. Da Actor m sich auch nicht bewegt wie Hero *Z*, passt das Ergebnis „X 0 –“ zwischen den beiden. Laut Situation 4 sollen Actor *p* und *l* auf den Hero auffahren. Bei Actor *l* lässt sich das allein mit der Richtungsgeraden nicht erkennen, weswegen zusätzlich die Seitenhalbgeraden des Fahrzeugs dazu genommen werden müssen. In dem Fall von *p* reicht die Richtungsgerade aus, um zu sehen, ob es auf *Z* landet. Nun bleibt in der Gruppe nur noch zu überprüfen, ob die TTC-Werte passen.

$$p\text{: TTC} = \left|\frac{44{,}44m}{10{,}17\frac{m}{s}}\right| = 4{,}37\text{s in Situation 4 } = 4{,}37\text{s}$$

$$l\text{ : TTC} = \left|\frac{9{,}93m}{0{,}25\frac{m}{s}}\right| = 39{,}72\text{s in Situation 4 } = 40{,}12\text{s}$$

6.3 TTC-Evaluierung

TTC-Ergebnisse erkennt man an „Zahl Buchstabe Zahl“-Werten in der Client-Ansicht. Um die TTC-Methode zu testen, bietet sich Town07 besser an, da es in dieser nicht zu viele rechtwinklige Straßen gibt, anders als in Town10. Da die TTC-Fälle nicht so häufig vorkommen, werden diese nur einzeln getestet.

6.3.1 TTC Test Fall B

Abbildung 23: TTC Test Methode Fall B
Quelle [selbst erstellt]

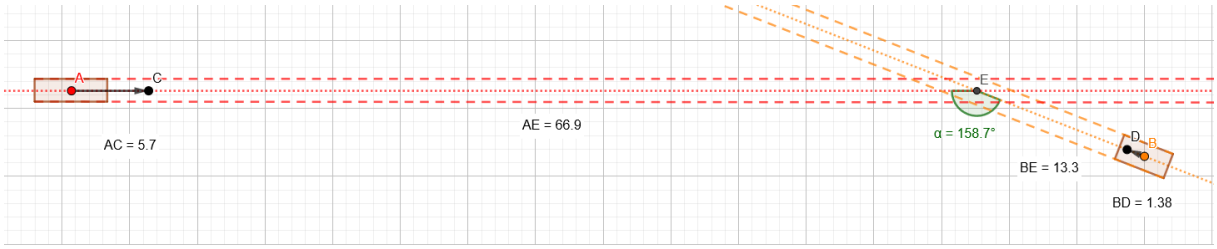


Abbildung 24: TTC Test Koordinaten System
Quelle: [selbst erstellt]

Als erstes wird wieder der TTC-α Wert überprüft.

$$\text{TTC-}\alpha = \left|\frac{66{,}9\text{m}}{5{,}7\frac{\text{m}}{\text{s}}} - \frac{13{,}3\text{m}}{1{,}38\frac{\text{m}}{\text{s}}}\right| = 2{,}10\text{s statt } 0{,}77\text{s}$$

Kurzfassung der Rechnung von Kapitel 4.3.2 TTC-α

Wir sehen, dass wir diesmal eine sehr große Abweichung verglichen mit dem Programm haben. Es stimmt etwas nicht, weshalb es so unmöglich sein wird, herauszufinden, wo und wie die

beiden Fahrzeuge zusammenstoßen werden. Erst wenn die Fahrzeuge per Vektor in Richtung ihrer Geschwindigkeit verschoben werden, lässt sich die richtige Kollision ableiten.

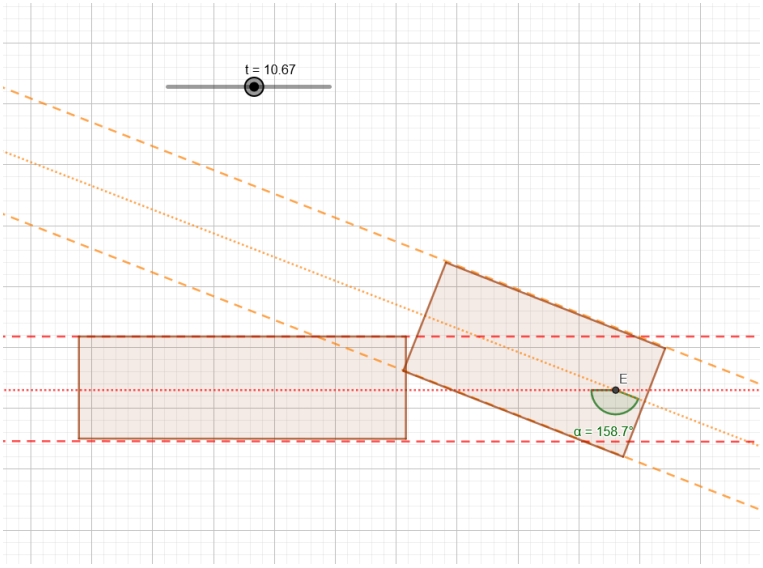


Abbildung 25: TTC Test Koordinaten Unfall
Quelle: [selbst erstellt]

Gemäß Abb. 30 wird der Zusammenstoß bei ca. t=10,67 Sekunden stattfinden. Dieser Wert entspricht dem TTC-Wert. Das Programm sagt mit den Simulationswerten von 11,05 einen sehr ähnlichen Wert aus, womit dieser sehr wahrscheinlich richtig ist, sofern wir jetzt zusätzlich den neuen Schnittpunkt der beiden Fahrzeuge als Treffpunkt annehmen.

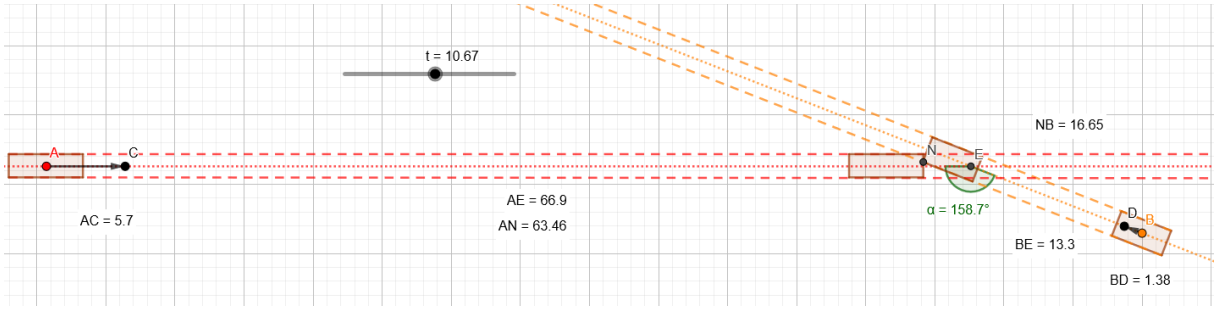


Abbildung 26: TTC Test Auflösung
Quelle: [selbst erstellt]

Mit diesen wird jetzt TTC-α neu berechnet.

$$\text{TTC-}\alpha = \left|\frac{63{,}46\text{m}}{5{,}7\frac{\text{m}}{\text{s}}} - \frac{16{,}65\text{m}}{1{,}38\frac{\text{m}}{\text{s}}}\right| = 0{,}93\text{s statt } 0{,}77\text{s}$$

Auf diese Weise kommen wir sehr viel näher an den Programmwert. Das liegt daran, dass das Programm den α Wert vom richtigen Punkt ausrechnet. Die Abweichung ist ein wenig größer, da im Beispiel viel mehr gerundete Werte verwendet wurden, als in den anderen Beispielen.

6.3.2 TTC Test Fall H und J und I

Abbildung 27: TTC Test Extra
Quelle: [selbst erstellt]

Abb. 32 ist eine Extradarstellung für einen Fall, bei welchen mehrere TTC-Fälle auf einmal auftreten. Auf die Rechnungsüberprüfung wird hierzu verzichtet

6.4 THW und Risiko Evaluierung

Abbildung 28: THW und Risiko Test
Quelle: [selbst erstellt]

In Abb. 33 sieht man, dass durch die Kombination von THW und TCC mehr Risikofälle abgedeckt werden, die sonst nur von TTC nicht beachten werden. Die ganzen Fahrzeuge mit Risiko 6, weil sie zu nah sind. Wären bei TTC als ganz sicher eingestuft.

7. Zusammenfassung

Das Ziel dieser Forschung ist es die Klassifizierung der Relevanz von Verkehrsteilnehmern zu bestimmen. Um dies zu erreichen, wurde die Detektion der Bewegung von Verkehrsteilnehmern aus Positionsdaten und die anschließenden externen Klassifizierten von vorangehenden separaten Arbeiten weiter ausgebaut. Aus den Positionsdaten und den Zeitstempeln aus diesen wird in diese Arbeit die Geschwindigkeit, Ausrichtung, Beschleunigung sowie der Abstand und dessen Änderung in Bezug von zwei Verkehrsteilnehmer berechnet. Diese Informationen werden dann in CARLA zu einer möglichen Bewegungsvorhersage zusammengefasst. Mit welcher dann eine TTC-α-, TTC- und eine THW-Zeit zwischen zwei Teilnehmern gebildet werden konnte. Anhand dieser Zeiten konnte man nun die Relevanz der Fahrzeuge zueinander anhand von einer Risikoeinschätzung gut darstellen.

8. Ausblick

Es ist festzustellen, dass durch die Anwendung von CARLA eine vielversprechende Grundlage für einen reibungslosen, automatisierten Ablauf eines virtuellen Straßenverkehrs geschaffen wurde, welches anfangs ohne Unfälle, also praktisch perfekt operierte. Um brauchbare Kollisionsvorgänge testen zu können, musste hierzu eine passende Umgebung geschaffen werden, in welcher Fahrzeuge sich nicht mehr an Verkehrsregeln halten. Mit Hilfe kleiner Änderungen der CARLA-Straßen-Regeln, wurde dies schnell erreicht. Somit konnte ohne große Probleme mit den ersten Relevanztests angefangen werden. Zu Beginn nur mit einfachen Abstandsprüfungen zu immer schwierigen Versuchen, bis schließlich zur komplexen TTC-Methode. Mit dieser konnte am Ende die Relevanz zwischen Verkehrsteilnehmer bestimmt werden. Es wurde neben denen in der Arbeit erwähnten Methoden noch weitere Versuche in CARLA durchgeführt, um mehr Rückschlüsse auf die Relevanz zu ziehen. Allerdings waren diese entweder für den Rahmen einer Bachelorarbeit mathematisch zu komplex oder haben kein brauchbares Ergebnis geliefert. Es handelte sich um Versuche mit Markov-Ketten, die eine Wahrscheinlichkeit zum Bahnwechsel ausrechnen sollten oder einen anderen Ansatz mit Clustern von Informationen, um damit möglicherweise Relevanzgruppen zu bilden. In Rahmen dieser Arbeit wurde schlussendlich mit diesen Ansätzen keine Ergebnisse erzielt. Gleichzeitig bedeutet dies nicht, dass diese Ideen in der Zukunft weiterhin ergebnislos bleiben.

Abschließend kann gesagt werden, dass trotz allem viele der geplanten Anforderungen von der CARLA-Simulation erfüllt wurden und die aktuellen Ergebnisse weiterverwendet werden können. So können die gelieferten Ergebnisse auf die Realität übertragen werden.

Literaturverzeichnis

[1] Destatis, „Verkehrsunfälle - Fachserie 8 Reihe 7 - 2021,“ 7 7 22. [Online]. Available: https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Verkehrsunfaelle/Publikationen/Downloads-Verkehrsunfaelle/verkehrsunfaelle-jahr-2080700217004.html.

[2] TWI, „WAS IST SIMULATION,“ 1 10 2022. [Online]. Available: https://www.twi-global.com/locations/deutschland/was-wir-tun/haeufig-gestellte-fragen/was-ist-simulation.

[3] H. H. Berthold Schuppar, „Bremsweg,“ in *Elementare Numerik für die Sekundarstufe*, Heidelberg, Springer Berlin, 2014, pp. 50-53.

[4] F. Torlak, „Detektion der Bewegung von Verkehrsteilnehmern aus Positionsdaten,“ 2022.

[5] M. Sohl, „Klassifizierung der Bewegungsmuster von Mobilfunkteilnehmern zur erweiterten Umfeldwahrnehmung autonomer Fahrzeuge,“ 2022.

[6] G. R. F. C. A. L. V. K. Alexey Dosovitskiy, „Carla.org,“ 1 10 2022. [Online]. Available: http://proceedings.mlr.press/v78/dosovitskiy17a/dosovitskiy17a.pdf.

[7] „Carla.docs,“ 1 10 2022. [Online]. Available: https://carla.readthedocs.io/en/latest/core_map/.

[8] L. C. J. L. Xiaoxia Xiong, „Vehicle Driving Risk Prediction Based on Markov Chain Model,“ Hindawi, Zhenjiang, 2018.

[9] F. Jiménez, Naranjo, J. E. und F. García, „An Improved Method to Calculate the Time-to-Collision of Two Vehicles.,“ *International Journal of Intelligent Transportation Systems Research,* pp. 34-42, 2013.

[10] R. B. H. J. R. d. T. S. F. Sampo Kuutti, „Safe Deep Neural Network-Driven Autonomous Vehicles Using Software Safety Cages,“ in *Intelligent Data Engineering and Automated Learning – IDEAL 2019,* Manchester, 20th International Conference, 2019, p. 150–160.

Anhang:

A1: Vorhersagen-Methode

```
1   def forcast(d,Car_A,Car_B): #Abständerung
2       newdistance = round(math.sqrt((Car_B.pos.x + Car_B.v.x - Car_A.pos.x +
        Car_A.v.x) ** 2 + (Car_B.pos.y + Car_B.v.y - Car_A.pos.y + Car_A.v.y) ** 2
        + (Car_B.pos.z + Car_B.v.z - Car_A.pos.z + Car_A.v.z) ** 2),0)
3
4       olddistance = round(d, 0)
5       if newdistance > olddistance + 4:
6           forcast = "+++++      "
7       elif newdistance > olddistance + 3:
8           forcast = " ++++      "
9       elif newdistance > olddistance + 2:
10          forcast = "  +++      "
11      elif newdistance > olddistance + 1:
12          forcast = "   ++      "
13      elif newdistance > olddistance:
14          forcast = "    +      "
15      elif newdistance < olddistance - 4:
16          forcast = "      -----"
17      elif newdistance < olddistance - 3:
18          forcast = "      ---- "
19      elif newdistance < olddistance - 2:
20          forcast = "      ---  "
21      elif newdistance < olddistance - 1:
22          forcast = "      --   "
23      elif newdistance < olddistance:
24          forcast = "      -    "
25      else:
26          forcast = "     0     "
27
28      return forcast
```

Code 6: Vorhersagen-Methode

A2: Riskiogruppen-Methode

```
1   def Riskgroup(TTC_i,THW_i):
2       Risk = "0"
3       if(THW_i == 'X'):
4           THW = float('inf')
5       if(TTC_i=='-' or TTC_i != "X"):
6           TTC = 999999 #float('inf')
7       if TTC!=0:
8           iTTC = (1/int(TTC))*100
9       else:
10          iTTC = 0;
11      if(THW_i == 'X'):
12          THW = 999999 #float('inf')
13      THW = int(THW_i) *10
14
15      if 100 <= iTTC:
16          Risk = "9"
17      elif 67<= iTTC and iTTC<100:
18          Risk = "8"
19      elif 0 <= iTTC and iTTC < 67 and THW < 9:
20          Risk = "7"
21      elif 0 <= iTTC and iTTC < 67 and 9<= THW and THW < 13:
22          Risk = "6"
23      elif 0 <= iTTC and iTTC < 67 and 13<= THW and THW < 18:
24          Risk = "5"
25      elif 0 <= iTTC and iTTC < 67 and 17<= THW and THW < 25:
26          Risk = "4"
27      elif iTTC < 0 and THW < 25:
28          Risk = "3"
29      elif 0 <= iTTC and iTTC < 67 and THW < 25:
30          Risk = "2"
31      elif iTTC < 0 and 25 <=THW:
32          Risk = "1"
33      return Risk
```

Code 7: Risikogruppen-Methode

A3: Cross Methode

```
1   def cross(Car_A, Car_B):
2       try:
3           c1 = findIntersection(Car_A.corner_B,Car_A.corner_A, Car_B.corner_B,
            Car_B.corner_D)
4           c2 = findIntersection(Car_A.corner_C,Car_A.corner_D,
            Car_B.corner_B, Car_B.corner_D)
5
6           if distance(Car_A.corner_A, c1) < distance(Car_A.corner_B, c1) and
            distance(Car_A.corner_D, c2) < distance(Car_A.corner_C, c2) and
            (Linie(Car_B.corner_B.x, Car_B.corner_D.x, c1) == True or
            Linie(Car_B.corner_B.x, Car_B.corner_D.x,c2) == True):
7               n1 = 0
8               n2 = 0
9               if Car_B.corner_B.x > Car_B.corner_D.x:
10                  n1 = Car_B.corner_B.x
11                  n2 = Car_B.corner_D.x
12              else:
13                  n1 = Car_B.corner_D.x
14                  n2 = Car_B.corner_B.x
15              if (n2 <= c1.x and c1.x <= n1) or (n2 <= c2.x and c2.x <= n1):
16                  return True
17              else:
18                  return False
19          else:
20              return False
21      except:
22          return False
```

Code 8: Cross Methode

A4: TTC-α Beispiel: Situation 2 Normalverkehr Town7

Abbildung 29: TTC-α Beispiel: Situation 2 Normalverkehr Town7
Quelle: [selbst erstellt]

A5: TTC-α Beispiel: Situation 3 Normalverkehr Town10

Abbildung 30: TTC-α Beispiel: Situation 3 Normalverkehr Town10
Quelle: [selbst erstellt]

A6: TTC-α Beispiel: Situation 4 warten an der Kreuzung Town10

Abbildung 31: TTC-α Beispiel: Situation 4 warten an der Kreuzung Town10
Quelle: [selbst erstellt]

Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich meine **Bachelorarbeit** mit dem Titel

---

Klassifizierung der Relevanz von Verkehrsteilnehmern zur Erweiterten Umfeldwahrnehmung Autonomer Fahrzeuge

---

selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie nicht an anderer Stelle als Prüfungsarbeit vorgelegt habe.

Coburg

Ort

03.10.2022

Datum

Unterschrift