THM TECHNISCHE HOCHSCHULE MITTELHESSEN | CAMPUS GIESSEN | MNI Mathematik, Naturwissenschaften und Informatik

Projektbericht

Implementierung eines Proof of Concepts zur Visualisierung von
„vehicle-to-infrastructure“ Kommunikation
im Kontext von Ampelinformationen

vorgelegt dem

Fachbereich Mathematik, Naturwissenschaften und Informatik

der Technischen Hochschule Mittelhessen

Samir Faycal Tahar M’Sallem

im Dezember 2022

Projektpartner: Valtech Mobility GmbH

Referent: Prof. Dr.-Ing. Seyed Eghbal Ghobadi

Inhalt

1 Einleitung

1.1 Projektpartner

Die 12-wöchige Projektphase wurde bei der Valtech Mobility GmbH abgehalten, die ein Softwareunternehmen ist. Sie selbst beschreibt sich als „Automotive Software Company für digitale Lösungen rund um das vernetzte Fahrzeug. Joint Venture der Volkswagen Gruppe und der Digitalagentur Valtech“. Hauptschwerpunkte sind die Entwicklung von Konzepten und deren Realisierung rund um das vernetzte Fahrzeug. Dabei ist sie für die Implementierung von Frontend und Backend Lösungen für diverse OEMs im Automobilbereich verantwortlich.

Die Valtech Mobility GmbH verfügt über sechs Standorte in Deutschland, wobei ich am Standort Neu-Isenburg tätig war. Zuvor hatte ich bereits etwa ein Jahr bei der Valtech Mobility als Werkstudent in einem anderen Projekt gearbeitet, weshalb mir die Wahl des Projektpartners für die Projektphase denkbar leichtfiel.

1.2 Projektvorstellung

Im Rahmen der Projekttätigkeit bei der Valtech Mobility GmbH habe ich ein Proof of Concept für eine App mit Backend entwickelt. Das Ziel war es eine Komfortfunktion im vernetzten Fahrzeug zu demonstrieren. Dazu sollte aus Ampelinformationen, die in das Fahrzeug transportiert werden können, ein Mehrwert generiert werden, um Fahrten im Fahrzeug komfortabler zu machen und Kraftstoff und CO2-Emissionen einzusparen.

Dem Benutzer werden dafür die Schaltzyklen der Ampeln visuell zur Verfügung gestellt, sodass dieser auf die Ampelphasen reagieren kann. Weiterhin wird eine Geschwindigkeitsempfehlung ausgesprochen, an der sich der Fahrer orientieren kann, um die Grüne-Welle zu befahren und konsequenterweise ohne einen Stopp, eine Kreuzung zu passieren.

Das Frontend wurde mittels einer Flutter App nativ für IOS und Android Geräte entwickelt. Das Backend bildet ein Spring Boot Microservice, der die Daten der Infrastruktur durch einen Content-Provider bezieht.

Der Content-Provider TTS, hat es sich zur Aufgabe gemacht, Daten der Infrastruktur zu sammeln, um dann V2I-Systeme zu realisieren. Diese Daten werden via API-Schnittstelle von unserem Backend abgefragt und dann aufbereitet.

Die daraus gewonnenen Daten werden dann in regelmäßigen Abständen zwischen Frontend und Backend ausgetauscht.

Hintergrund für die Entstehung dieser Projektidee ist das Referenzsystem „Traffic Information Online“ der Audi AG [1], dessen Backend ebenfalls im Jahre 2016 von der Valtech Mobility GmbH programmiert wurde. Um Kundenakquise für Neukunden zu betreiben, wurde daher die Idee des hier vorgestellten Proof of Concepts erstellt.

2 Vorgehensweise

Das Proof of Concept wurde im Rahmen von agiler Prozessentwicklung umgesetzt. Dazu wurde Scrum als Projektmanagement-Verfahren verwendet, welches das kleine Team bestehend aus mir und zwei weiteren Werkstudenten sehr gut unterstützte.

Im Anforderungsprozess haben wir uns mit unserem Product Owner Manuel Hirschauer zusammengesetzt und wichtige Key-Features auf einem Miro Board diskutiert und visualisiert. Eingestuft haben wir diese dann nach Machbarkeit, Nutzen und Unsicherheit. Gemeint ist die Einschätzung, welches Spezialwissen für ein Feature benötigt wird und welche Unsicherheiten noch zu Anfang bestehen. Dies war insofern wichtig, da wir zwar Features vom Referenzsystem von Audi kannten, aber aufgrund von Geheimhaltung keinen Zugriff auf Konzepte oder konkrete Umsetzungen erhielten.

Nachdem dann grundlegende Features festgelegt wurden, begannen wir mit dem Planen der jeweils 2-wöchigen Sprints. Dies erfolgte in regelmäßigen Abständen vor jedem neuen Sprint. Begleitet wurde dies mit der Evaluation der vorherigen Sprints. Dazu haben wir Retrospektiven geführt, in denen wir reflektierten, was gut lief und wo es Verbesserungsbedarf gibt. Jeder Sprint erhielt ein individuelles „sprint goal“, welches stellvertretend für das Entwicklungsziel nach jeder zweiwöchigen Iteration stand.

In Hinblick auf dieses Projektmanagement ist anzumerken, dass wir unsere Sprints und unseren Backlog in Atlassian Jira geplant haben. Weiterhin wurden unsere Repositories in Atlassian BitBucket gehostet, da diese Cloud-Anwendung geeignet ist, um Git-Repositories zu verwalten, wenn Jira verwendet wird.

3 Entwicklungsprozess

Nachdem die grundlegenden Disziplinen des Projektmanagements umgesetzt wurden, konnte mit der Entwicklung des Proof of Concepts begonnen werden. Die ersten Sprints bestanden überwiegend aus der Einrichtung der Infrastruktur, Erstellung der Projekte und Einfindung in die Programmiersprache Dart und das Flutter Framework, da diese für mich neue Technologien waren.

Im Anschluss habe ich mich in die Dokumentation der API-Schnittstelle unseres Content-Providers TTS („Traffic Technology Services Inc.“) eingearbeitet, um im Anschluss mit der Implementierung unseres eigenen Backends zu beginnen.

3.1 Der Content-Providers TTS

Über die Schnittstelle erhielten wir Zugriff auf Kreuzungsinformationen und Ampelschaltzyklen. Die Daten beschränkten sich auf die Stadt Ingolstadt, da dort die meisten vernetzten Ampelsysteme existieren.

Die Schnittstelle von TTS liefert eine Zuordnung („mapping“) der Fahrzeugposition zu einer bestimmten Kreuzung in der Umgebung. Daraufhin werden die aktuelle, sowie die nächsten zwei Signalphasen mit Zeitstempel zurückgeliefert. Für die Zukunft ist aber geplant, dass dies durch unser Backend erfolgt und TTS alle Schaltzyklen aller Kreuzungen und deren Anfahrten an uns liefert und das Backend dann selbstständig berechnet, welche Kreuzung passend zur Fahrzeugposition ist.

Die Eingabeparameter sind die Position des Fahrzeugs und die Kompassrichtung, in der das Fahrzeug ausgerichtet ist. Das ist notwendig, um die Fahrtrichtung identifizieren zu können. Liefert man diese Daten aus, so wird eine Kreuzung in der näheren Umgebung zurückgeliefert, auf die das Fahrzeug zufährt.

Eine Kreuzung („intersection“) im Sinne ihres Aufbaus, lässt sich dadurch definieren, dass sie mindestens zwei Anfahrten („approaches“) enthält. Jede Anfahrt hat mitunter mehrere Fahrspuren („lanes“), die in der Regel nicht nur geradeaus führen, sondern auch nach links oder rechts. Für jede Anfahrt existiert mindestens ein Ampelkopf („signal head“), der die Ampelschaltung signalisiert. Insofern die Schaltzyklen für verschiedene Richtungen variieren, existieren unter Umständen auch mehrere Ampelköpfe für eine Anfahrt. Jeder Anfahrt kann eine Kompassrichtung („bearing“) zugeordnet werden, die die Richtung zur Kreuzungsmitte widerspiegelt.

Diese Information wird benötigt, da Anfahrten nicht immer in perfekter Ausrichtung zu den Himmelsrichtungen stehen und somit ein Zahlenwert – hier die Kompassrichtung – herangezogen wird. Da das Fahrzeug ebenfalls in eine Kompassrichtung ausgerichtet ist, kann mittels dieser Informationen die Zuordnung zur korrekten Kreuzungsanfahrt erfolgen. Damit ist es möglich, die Fahrzeugposition zur richtigen Kreuzungsanfahrt („intersection approach“) zuzuordnen. Nur durch diesen Umstand können korrekte Ampelschaltzyklen im Fahrzeug angezeigt werden, da sonst keine Gewissheit bestünde, dass das Fahrzeug tatsächlich in Richtung der Kreuzung bewegt wird.

Nebst dem haben wir von TTS eine Liste von allen Kreuzungen mit deren Position („latitude“ / „longitude“) erhalten, sowie die jeweiligen Anfahrten. Dabei sind die Anfahrten durch eine Menge von Punkten angegeben, die jeweils einen Umriss („approach outline“) um die Kreuzungsanfahrt bilden.

3.2 Key Features

Im Rahmen des Anforderungsprozesses haben wir die Key Features auf verschiedenen Ebenen definiert.

Ziel ist es vordergründig, die große Datenmenge des Content-Providers prägnant auf einer Übersichtskarte zu visualisieren. Weiterhin sollen potenzielle Kunden die Möglichkeit erhalten, zu verstehen, welche Vorteile und Bereicherungen für den individuellen Fahrer und das gesamte Verkehrsgeschehen, durch die Nutzung der Ampelinformationen entstehen können.

Schlussendlich soll die Rolle der von mir generierten Geschwindigkeitsempfehlung zu tragen kommen, indem gezeigt wird, dass sofern ein Fahrzeug diese Empfehlung befolgt, es stets der Grünen Welle folgt.

Aus diesen Ausführungen haben wir dann konkrete Funktionalitäten für unsere App und das Backend geformt:

1. Die Übersichtskarte („overview map“):

   Auf der Karte werden alle uns bekannten Kreuzungen und deren Anfahrten angezeigt. Indem man auf eine Anfahrt klickt, sollen dann die entsprechenden Ampelschaltungen sichtbar werden. Im Wesentlichen liegt der Nutzen darin, zu verstehen, welche unterschiedlichen Ampeltypen existieren und welche unentdeckten Probleme im Anforderungsschritt unbeachtet blieben.

2. Die Geschwindigkeitsempfehlung

   Diese soll basierend auf der momentanen Fahrzeugposition und den Ampelschaltphasen eine Empfehlung aussprechen, die den Fahrer befähigt, im richtigen Moment an der Ampel anzukommen, sodass dieser ohne einen Stopp die grüne Ampel passieren kann. Andockpunkte und Erweiterungsmöglichkeiten dieses Verfahrens haben sich erst im Implementierungsprozess ergeben, weshalb dieser Punkt zu einem späteren Zeitpunkt intensiver diskutiert wird.

3. Die simulierte Fahrtroute („Demo-Modus“)

   Um den reellen Nutzen des Wissens über die Ampelinformationen demonstrieren zu können, sollte die App die Möglichkeit erlauben eine Route innerhalb von Ingolstadt zu definieren, die dann - ähnlich zur Navigation über eine Navigationsapp - abgefahren wird. Dabei bewegt sich ein simuliertes Fahrzeug entlang der Route und reagiert auf die Ampelphasen, indem es die von uns generierte Geschwindigkeitsempfehlung beachtet und seine Geschwindigkeit entsprechend anpasst. Ziel dabei ist es, zu beweisen, dass bei Beachtung der Geschwindigkeitsempfehlung die Grüne Welle optimal befahren wird und das Fahrzeug zu keinem Zeitpunkt zum Stehen kommen muss.

4. Der Bezug zur Realität („Live-Modus“)

   Dieser Modus, der die echte GPS-Position des Handys abruft und dann die Ampelschaltungen der Ampel anzeigt, auf die man sich zubewegt, soll den Benutzer befähigen, die App in der Realität zu testen. Vorausgesetzt wird dabei natürlich, dass man sich in Ingolstadt befindet und eine Kreuzung in der Nähe ist. Da dies zu Demonstrationszwecken nicht optimal ist, wurde der vorher erwähnte „Demo-Modus“ als Hauptaugenmerk identifiziert. Dennoch haben wir dieses Feature mitberücksichtigt, da wir uns erhofften, dies selbst erproben zu können.

3.3 Implementierung

Das Backend bildet eine REST-API, die über die Flutter App angesprochen wird. Dieses ist in der Lage Daten des Content-Providers TTS anzufragen und aufzuarbeiten. Erwähnenswert ist, dass es nur einen Endpunkt gibt, mit dem das Backend kommuniziert. Dieser Endpunkt liefert lediglich die Ampelinformationen für die Ampel in Fahrtrichtung der übergebenen Positionsdaten.

Des Weiteren kommuniziert das Backend mit Schnittstellen von Google, um Geschwindigkeitsbegrenzungen und Verkehrsinformationen abzufragen. Diese sind besonders für die spätere Berechnung der Geschwindigkeitsempfehlung entscheidend, da diese Daten ebenfalls berücksichtigt werden.

Um auf der Karte in der App alle Kreuzungen und deren Anfahrten anzuzeigen, führt das Frontend einen Aufruf („request“) an das Backend aus, um die entsprechenden Informationen abzufragen. Im Anschluss werden diese dann mittels Marker in der App angezeigt. Die Anfahrten werden durch Polygone dargestellt, da die Straßenverläufe in der Regel nicht gradlinig sind. Mit einem Klick auf eines dieser Polygone wird dann die entsprechende Ampelinformation zurückgegeben. Dies geschieht, indem zwei Punkte aus dem Polygon betrachtet werden. Da Punkte in unserem Fall durch Längen- und Breitengrad („latitude /longitude“) definiert sind, kann eine Kompassrichtung vom ersten zum zweiten Punkt ermittelt werden.

In Kombination mit dem ersten Punkt, kann dann die entsprechende Kreuzungsanfahrt mit ihren Ampelschaltungen aus den Daten des Content-Providers ausgelesen werden. Diese umständliche Herangehensweise ist erforderlich, da wir keine reinen Datenfeeds zur Verfügung haben, sondern nur einen Endpunkt, den man mittels einer Positionsinformation anfragen kann. Würde man den Kreuzungsmittelpunkt ohne eine Kompassrichtung als Position übergeben, so würde man Daten für die falsche Anfahrt bekommen, da eine Anfahrt stets durch eine Kompassrichtung definiert ist.

Hinsichtlich des Demo-Modus ist zu sagen, dass dieser in regelmäßigen Abständen Aufrufe an das Backend ausführt und dann das Fahrzeug entsprechend der Ampelinformationen fahren lässt. Wenn die Navigation beginnt, wird ein Aufruf ausgeführt, der erfragt, ob das Fahrzeug sich in Ingolstadt befindet. Sofern das Fahrzeug in Ingolstadt ist, wird eine Distanz zurückgegeben, in der die nächste Kreuzung liegt.

Nachdem das Fahrzeug diese Distanz zurückgelegt hat, wird erneut geprüft, ob eine Kreuzung in der Nähe ist. Sollten Ampelschaltungen für diese Ampel vorhanden sein, so werden diese angezeigt.

Mit der späteren Entwicklung von GLOSA ist eine weitere Funktionalität hinzugekommen, die es ermöglicht, dass das Fahrzeug mit einer Geschwindigkeit bewegt wird, die entsprechend der Geschwindigkeitsempfehlung angepasst werden kann.

Konkret bedeutet das, dass das Fahrzeug vor roten Ampeln stehen bleibt und sobald die nächste Grünphase beginnt wieder losfährt und auf die Höchstgeschwindigkeit beschleunigt.

Um die Vorteile des Wissens über die nächsten Ampelphasen zu verdeutlichen, beachtet das Fahrzeug die vom Backend berechneten Geschwindigkeitsempfehlungen, so dass es zu keinem Zeitpunkt aktiv zum Stillstand kommt. Dies wird erreicht, in dem unser Backend neben der Fahrzeugposition und Ausrichtung auch die momentane Fahrzeuggeschwindigkeit, die Distanz zur Kreuzung und die Geschwindigkeitsbegrenzung berücksichtigt. Diese Daten fließen in die Berechnung der Geschwindigkeitsempfehlung ein und resultieren in einer Geschwindigkeit, die das Fahrzeug umsetzt. Dabei existiert für jede Geschwindigkeitsempfehlung eine Ober- und Untergrenze. Diese Grenzen stehen dabei für die minimale und maximale Geschwindigkeit, die das Fahrzeug fahren sollte, um die grüne Welle zu befahren.

Im ersten Beispiel wurde die Berechnung, ohne hinzuziehen von Verkehrsinformationen durchgeführt. Das bedeutet, dass nur die Abstände und die Zeiten gemäß des Weg-Zeit-Gesetzes herangezogen wurden. Es wird geprüft, ob das Fahrzeug unter Beachtung der Höchstgeschwindigkeit - der Geschwindigkeitsbegrenzung auf der Straße - in der Lage ist, die Ampel zu überqueren.

Nachfolgend werden Szenarien aufgelistet, die als Grundlage für die Entscheidungsfindung des Algorithmus der Geschwindigkeitsempfehlung dienen. Als Prämisse dient die Annahme, dass lediglich zwei Phasen (grün/rot) vom Content-Provider geliefert werden, die in abwechselnder Reihenfolge auftreten.

Folgende Szenarien sind dabei zu identifizieren:

1. Die Ampel ist grün und das Fahrzeug ist in der Lage, die Kreuzung zu passieren, bevor die Rotphase beginnt
2. Die Ampel ist grün und das Fahrzeug ist nicht in der Lage, die Kreuzung zu passieren, bevor die Rotphase beginnt
3. Die Ampel ist rot und das Fahrzeug kann während der nächsten Grünphase, die Kreuzung passieren

Im ersten Szenario würde geprüft werden, ob die Geschwindigkeit, die erforderlich ist, damit das Fahrzeug passiert, geringer als die Höchstgeschwindigkeit ist. Sofern dies der Fall ist, wird die Geschwindigkeitsempfehlung entsprechend dieses Szenarios berechnet.

Das zweite Szenario stellt das Gegenbeispiel zum ersten Szenario dar. In diesem Fall würde die Zeit betrachtet werden, bis die übernächste Phase - die neue Grünphase - beginnt. Aus dieser Zeit wird dann eine Geschwindigkeit berechnet, die als Höchstgeschwindigkeit angesehen wird. Dies ist insofern wichtig, da das Fahrzeug sonst vor dem Beginn der neuen Grünphase an der Ampel ankommen würde und folglich zum Stillstand kommen müsste.

Für das letzte Szenario bedient sich das Backend am Zeitpunkt des Starts der neuen grünen Phase und dessen Dauer. Die Minimalgeschwindigkeit ist die Geschwindigkeit bis zum Ende der neuen Grünphase, wogegen die Maximalgeschwindigkeit die Geschwindigkeit bis zum Beginn der Grünphase ist. Das Fahrzeug muss sich in diesem Geschwindigkeitsbereich bewegen, um sicher zu stellen, während der Grünphase die Kreuzung zu passieren.

Aus den Ausführungen kann man erkennen, dass stets ein Geschwindigkeitsbereich zurückgegeben wird. Die schlussendliche Geschwindigkeitsempfehlung orientiert sich an der momentanen Geschwindigkeit des Fahrzeugs. Es wird überprüft, ob das Fahrzeug in der Lage ist auf die Geschwindigkeit zu beschleunigen, indem angenommen wird, dass das Fahrzeug eine Beschleunigungskraft von 3,5 m/s^2 hat. Ist eine höhere Beschleunigung erforderlich, so kann keine Geschwindigkeitsempfehlung erfolgen.

Andernfalls wird die Momentangeschwindigkeit des Fahrzeugs betrachtet und das Fahrzeug erhält eine Geschwindigkeitsempfehlung.

Um einerseits eine hohe Energieeffizienz und Einsparung von CO2 Emissionen zu bewirken, aber andererseits die Komfortfunktion der Ampelschaltungen nicht zu vernachlässigen, gibt es zwei wählbare Modi, die als Fahrerprofile zu verstehen sind:

1. Der so genannte „fuel-save“-Modus gibt stets die untere Grenze des Geschwindigkeitsbereichs zurück. Daraus folgt, dass das Fahrzeug eine sehr niedrige Geschwindigkeitsempfehlung erhält, folglich aber auch die größte Einsparung von Kraftstoff und CO2 Emissionen aufweist. Auf der anderen Seite bedingt dies einen starken Eingriff in den Verkehr da das Fahrzeug übermäßig bremsen und damit seine Geschwindigkeit drastisch reduzieren muss.
2. Da in vielen Fällen Fahrer nicht bereit sind, ihre Geschwindigkeit an diese Empfehlung anzupassen, existiert der so genannte „comfort“-Modus.

Dieser versucht zu jedem Zeitpunkt die momentane Geschwindigkeit des Fahrzeugs beizubehalten, sofern diese innerhalb des Geschwindigkeitsbereichs liegt, welcher die grüne Welle befähigt. Andernfalls ist der Fahrer dazu angewiesen, seine Geschwindigkeit an die Ober- bzw. Untergrenze anzupassen. In diesem Verfahren liegt das Hauptaugenmerk nicht auf der Energieeffizienz, sondern auf der reinen Komfortfunktion gegenüber dem Fahrer, da dieser weniger beschleunigen und bremsen muss.

Diese Modi haben sich aus verschiedenen Meinungen von Fahrern und auch Studien ergeben, die im Teil „Verwandte Arbeiten“ aufgegriffen werden.

Um die Entscheidungsfindung noch verständlicher zu vermitteln, liegt in der Sektion „Anhang“ ein Diagramm bei.

3.4 Testing

Ein Softwareprodukt dieser Art lässt sich am besten in der Realität testen. Es war geplant, dass wir die Möglichkeit erhalten in Ingolstadt mit einem Fahrzeug unsere App zu testen. Dazu sollte ein Fahrzeug von Audi, welches über den Dienst „Traffic Information Online“ [1] verfügt, als Referenz genutzt werden. Geplant war demnach, dass sowohl das Fahrzeug als auch die App unter gleichen Bedingungen getestet werden.

Dazu wurden folgende Kriterien notiert:

- Die Fahrzeuggeschwindigkeit wird sowohl vom Fahrzeug als auch von der App korrekt angezeigt
- Der Zeitpunkt der Erfassung ist identisch
- Die GPS-Position des Handys und des Fahrzeugs ist identisch
    - Erweitert folgt daraus, dass beide Backend’s die Position zur richtigen Kreuzungsanfahrt zuordnen
- Sowohl Fahrzeug als auch Handy rechnen mit dem gleichen Routenverlauf

In Hinblick auf den Test wurden folgende Metriken notiert, die es zu untersuchen gibt:

- Performanz (im Sinne von Schnelligkeit und Verfügbarkeit der Resultate)
- Zuverlässigkeit (Umsetzbarkeit der Ergebnisse, Konsistenz)
- Sicherheit (des Fahrers)
- Design (Ablenkungsgefahr, generelles SW-Design Konzept)

Gerne würde ich diese Aspekte in der Zukunft untersuchen und ein Fazit hinsichtlich unserer Implementierung und der bestehenden ziehen. Leider war es mir innerhalb der Projektphase nicht möglich, dieses Fazit zu vollziehen.

Um dennoch ein Fazit hinsichtlich der Zuverlässigkeit und Konsistenz der Daten ziehen zu können, habe ich unsere Geschwindigkeitsempfehlung näher untersucht. Dabei habe ich geprüft, inwiefern die Geschwindigkeitsempfehlung in Abhängigkeit der Ampelphasen variiert und ob diese insgesamt konsistent sind. Nach den grundsätzlichen Entwicklungskonzepten von Software ist zu erwarten, dass bei gleicher Eingabe stets dasselbe Ergebnis folgt.

Übertragen auf den Kontext von Ampelinformationen bedeutet dies, dass bei gleicher Ampelschaltung, gleicher Fahrzeuggeschwindigkeit und gleichem Abstand stets dieselbe Geschwindigkeitsempfehlung erfolgen soll. Diese Annahme erfolgt mit dem Vorbehalt, dass sonstige Verkehrsereignisse und Verkehrsteilnehmer außer Acht gelassen werden.

Um dieses Vorhaben umzusetzen, habe ich einen Versuchsaufbau mit der Testmenge = 101 implementiert. Konkret bedeutet das, dass alle zwei Sekunden ein Aufruf (insgesamt 101-mal) an das Backend erfolgt, der stets dieselben Positionsdaten und die Geschwindigkeit des Fahrzeugs enthält.

Zu erwarten ist, dass die Geschwindigkeitsempfehlung so lange ansteigt, bis sie das Maximum (Höchstgeschwindigkeit auf der Straße) erreicht. Daraufhin wird die Geschwindigkeit auf ein Minimum abfallen, bis sie schließlich wieder ansteigt. Prägnant sollten dabei die Wendepunkte sein, an denen dieser Wechsel zwischen minimaler und maximaler Geschwindigkeit erscheint. Auch soll die vorherige Prämisse des Gesetzes der „gleichen Ampelphase resultierend in der stets gleichen Geschwindigkeit“ erkennbar sein.

Um dies zu visualisieren, bediene ich mich an der Zeit, bis die nächste Grünphase Eintritt. Die sogenannt TTG („time to green“) ist ein numerischer Wert in Sekunden, welcher angibt, wann die nächste Grünphase beginnt oder wie lang die momentane Grünphase noch andauert.

Die TTG wird interpretiert als:

- Positive Ganzzahl, die repräsentiert, wann die nächste Grünphase eintritt

- Negative Ganzzahl, die angibt wann die Grünphase endet, sofern sie momentan aktiv ist

Für den Versuchsaufbau wird nun eine Ampel gewählt, die zum Zeitpunkt X des Experiments ein gleichbleibendes Schaltverhalten aufweist. Dies ist wichtig, da sonst keine Aussagekraft hinsichtlich der Vergleichbarkeit getroffen werden kann.

Die Ergebnisse werden in eine statische JSON-Datei exportiert und später graphisch aufgearbeitet. Verwendete Werte sind dabei die TTG, die Geschwindigkeitsempfehlung, sowie die Minimal- und Maximalgeschwindigkeit.

Nachdem diese Daten aufgezeichnet wurden, können sie graphisch als „min-max-average“ Diagramm dargestellt werden. Dabei entspricht:

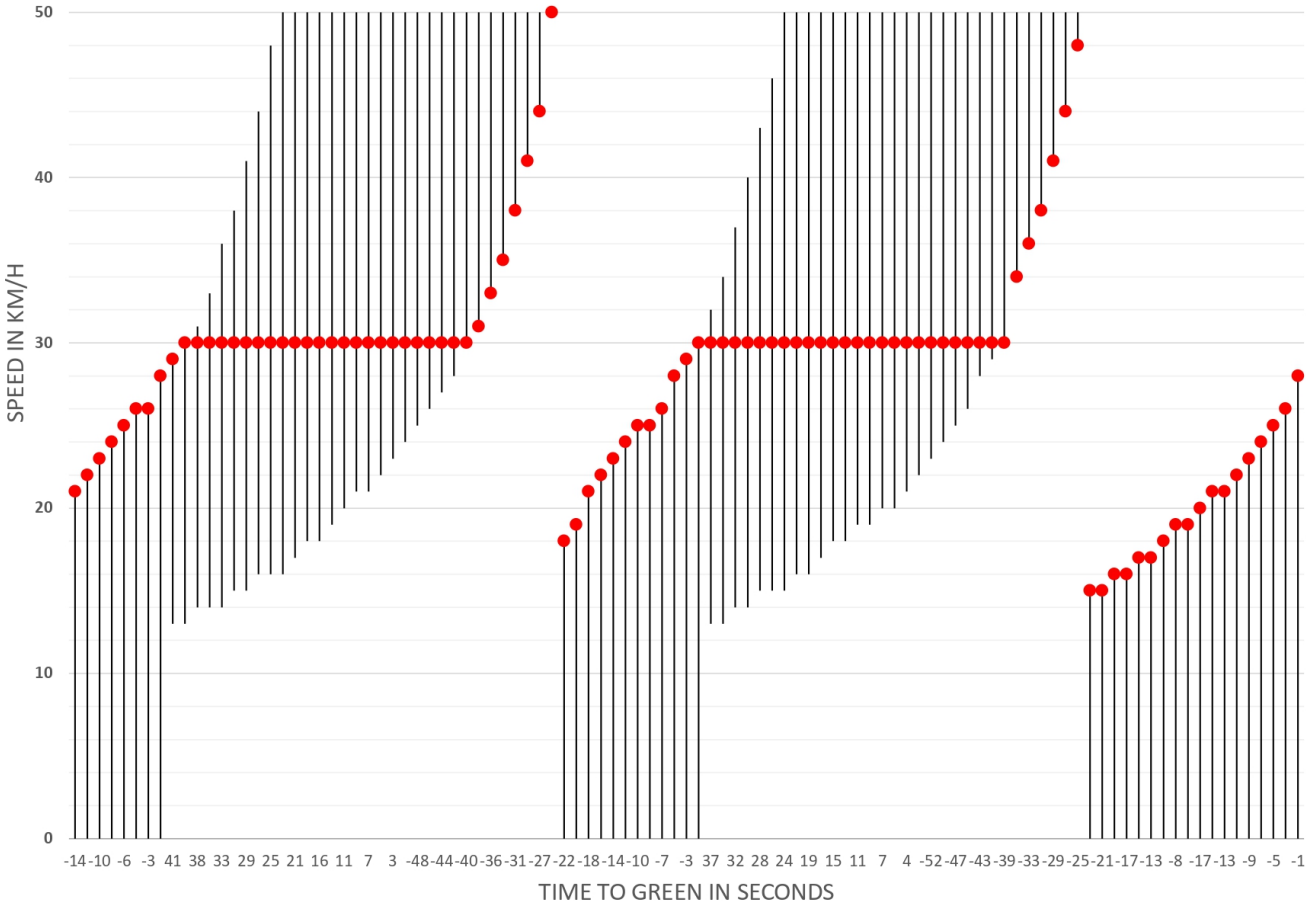
- X-Achse: TTG
- Y-Achse:
    - Min: Minimalgeschwindigkeit
    - Max: Maximalgeschwindigkeit
    - Average: Geschwindigkeitsempfehlung

Die Geschwindigkeitsempfehlung ist aber nicht als Durchschnittswert („average“) zu verstehen, sondern als Wert, der zwischen den beiden Grenzen liegt und wie zuvor erwähnt, berechnet wurde.

Für den Versuch wurden folgende Eingabewerte gewählt:
latitude: 48.726405691398355,
longitude: 11.441749699789504,
bearing: 67,
vehicleSpeed: 30km/h

Die daraus resultierenden Ergebnisse hinsichtlich der Kreuzung sind:
distanceToStopLine: 335m,
speedlimit: 50km/h

Diese Werte verbleiben während dem gesamten Versuch als Konstanten. Die einzige wandelbare Variable ist die TTG, welche abhängig von den Ampelschaltungen ist. Führt man den folgenden Versuch aus erhält man nachfolgende Grafik:



Die roten Punkte betiteln die Geschwindigkeitsempfehlung, während die schwarzen Striche für den Bereich zwischen minimaler und maximaler Geschwindigkeit stehen.

Wie man erkennen kann, versucht der Algorithmus stets die Geschwindigkeit an die Geschwindigkeit des Fahrzeugs anzupassen. Besonders gut erkennt man dies auf der X-Achse im Bereich von 41 bis -48. Sofern dann aber aufgrund der Ampelschaltungen erforderlich ist, dass das Fahrzeug schneller als die konstant angegebene 30km/h fahren muss, wird die minimal benötigte Geschwindigkeit zurückgegeben.

Ebenso verhält sich das Verfahren im Bereich der Höchstgeschwindigkeit. Am Wendepunkt (auf der X-Achse -25), Kann man erkennen, dass das Fahrzeug zuvor eine Empfehlung von 50 KMH erhalten hat, was bereits der Höchstgeschwindigkeit entspricht. Da nun zwei weitere Sekunden vergangen sind, müsste das Fahrzeug schneller als die Höchstgeschwindigkeit fahren.

Da dies nach den Konventionen des Algorithmus nicht möglich ist, wird die übernächste Grünphase empfohlen. Die damit verbundene Geschwindigkeitsempfehlung ist drastisch niedriger, weshalb der Wendepunkt zustande kommt.

Wie auch bereits vorher im Graph ersichtlich, wiederholt sich die Geschwindigkeitsempfehlung nun periodisch. Daran wird einerseits deutlich, dass die Ampel eine feste Schaltreihenfolge hat, zum anderen aber auch, dass der Algorithmus stets die gleiche Geschwindigkeit bei gleichen Gegebenheiten liefert.

Somit ist bewiesen, dass der Algorithmus hinsichtlich der Zuverlässigkeit und Konsistenz korrekt arbeitet.

Ebenso konnte durch den Demo-Modus bewiesen werden, dass sofern ein Fahrzeug sich an die Geschwindigkeitsempfehlung hält mit Sicherheit die Kreuzung passieren kann. Gleichermaßen liefert der Aufruf der empfohlenen Geschwindigkeit die gleiche Geschwindigkeit erneut zurück.

4 Fazit

4.1 Bewertung der Arbeitsergebnisse

Insgesamt kann die Arbeit, die innerhalb der letzten drei Monate erfolgt ist, als sehr positiv bewertet werden. Wir haben es geschafft aus einer Idee ein Konzept zu entwickeln, welches funktional und für mögliche interessierte Kunden vorstellbar ist. Durch den Einsatz von UI/UX Designern und deren Input ist es möglich gewesen, das Konzept als vollständigen Verkaufsprototypen darzustellen. Es ist geplant, dass die App auch auf der CES am 5. Januar 2023 präsentiert werden soll. Dies ist für uns als Entwickler ein großer Meilenstein.

Hervorzuheben ist, dass die App auf Android und iOS lauffähig ist, und das Backend, aus dem Internet heraus aufrufbar ist, so dass es auch auf echten Geräten verwendet werden kann.

Hinsichtlich Schwächen der Umsetzung ist zu sagen, dass wir innerhalb der Projektphase lediglich die bereits interpretierten Daten von TTS (unter Nutzung deren „mapping“) verwenden und nicht unsere eigene Interpretierung der Positionsdaten ausführen.

Es ist geplant, dass dieser Schritt aber noch in der Zukunft erfolgt. Für den ersten Schritt ist es dennoch ein sehr gelungenes Ergebnis.

4.2 Bewertung der Stelle im Unternehmen

Aufgrund der Tatsache, dass ich bereits vorher im Unternehmen als Werkstudent angestellt war, war es für mich sehr angenehm, die Projektphase auszuführen.

Anzumerken ist, dass das Projekt und die Idee, durch meinen Vorschlag entstanden ist und das Unternehmen mir sehr viel Freiraum und Flexibilität hinsichtlich der Umsetzung überlassen hat. Der große Vorteil war, dass das Unternehmen selbst Interesse an der Idee gefunden hat und es dadurch Unterstützung von meinem Betreuer und weiteren Werkstudenten gab. Ich war ebenfalls sehr frei in der Bestimmung meiner Arbeitszeit und meines Arbeitsortes. Ich konnte im Überfluss im Home-Office arbeiten und trotzdem täglich in Meetings die Meinungen, Anregungen und Hilfestellungen von meinem Betreuer aufnehmen.

5 Ausblick auf die Bachelorarbeit

Die Bachelorarbeit beginnt nun unmittelbar nach dem Ende der Projektphase.

Vertiefend möchte ich mich auf das Thema der Geschwindigkeitsempfehlung konzentrieren. Innerhalb der momentanen Realisation wird auf Verkehrsinformationen einer Google API verwiesen. Da diese Daten nicht zu vollständig verwertbar sind, soll dieser Ansatz weiterentwickelt werden.

5.1 Verknüpfung der Projektphase

Bereits innerhalb der Projektphase habe ich mich bei der Entwicklung der Geschwindigkeitsempfehlung auf Sekundärliteratur bezogen. Dabei ist besonders eine Studie der „City of Ottawa“ zum Tragen gekommen:

Mittels Probanden wurde ein solches System getestet und erprobt und Rückschlüsse hinsichtlich der Einsparung von Kraftstoff und CO2 Emissionen gezogen.

Zunächst werden Annahmen hinsichtlich der Implementierung erläutert. Diese Annahmen umfassen etwa die Dauer, die ein Fahrzeug wartet, ihr es über eine Ampel fahren kann. Da dies bloß eine Schätzung ist, offeriert dieser Ansatz ein Verbesserungspotential. Eine Verbesserung erfolgte in unserem Beispiel bereits durch das Nutzen der Google API. Trotzdem ist es möglich das Verfahren noch besser und robuster zu gestalten. Dazu möchte ich „vehicle-to-vehicle“ Kommunikation verwenden. Das Ziel ist es, eine noch bessere Geschwindigkeitsempfehlung zu erhalten, die Verkehrsteilnehmer auf der Straße berücksichtigt.

Um dies bestmöglich zu realisieren, möchte ich die Simulationsumgebung SUMO verwenden und dort die bereits vorhandenen Ampelschaltungen aus unserem Backend visualisieren. Darauf aufbauend möchte ich ein Netzwerk implementieren, welches Fahrzeuge befehligt, untereinander Informationen auszutauschen. Dabei soll es ein Fahrzeug geben, welches in Richtung der Kreuzung fährt, wo gegen andere Fahrzeuge bereits an einer roten Ampel warten. Die bereits wartenden Fahrzeuge sollen, dass sich annähernde Fahrzeug warnen, in dem sie ihre Position und Geschwindigkeit offenbaren. Das Fahrzeug erhält von der API-Informationen über die Ampelphasen und bezieht die Informationen der Fahrzeuge mit in die Berechnung der Geschwindigkeitsempfehlung ein. Der daraus generierte Mehrwert soll sich dahingehend bemerkbar machen, dass das Fahrzeug, ohne zu stoppen die Kreuzung passieren kann. Besonders das Heranziehen von Informationen über andere Verkehrsteilnehmer ist dabei ein

erfolgskritischer Faktor, da dies weder in der bisherigen Implementierung von mir noch innerhalb der Studie oder der Referenzimplementierung von Audi erfolgt.

Dieser Aspekt würde eine innovative Lösung darstellen und ich erhoffe mir daraus, Ergebnisse und Vorschläge für weitere Implementierungen formulieren zu können.

5.2 Verwandte Arbeiten

Das Referenzprodukt von Audi [1] wurde innerhalb der Projektphase hinsichtlich seiner Features als Leitbild verwendet. Innerhalb der Bachelorarbeit möchte ich dieses nicht erwähnen.

Es existiert eine Vielzahl von Artikeln und Abhandlungen, die Vorschläge und Ideen zur Verbesserung von GLOSA basierten Systemen liefern. Folgende sind dafür relevant: [2] [3] [4] [5] [6] [7] [8].

Weiterhin möchte ich die Studie der „City of Ottawa“ [9] als Motivationsfaktor nutzen, weshalb die Bachelorarbeit in diesem Bereich verortet ist. Sie weist einfache Annahmen auf, welche durch „vehicle-to-vehicle“ Kommunikation verbessert werden könnten.

Dazu ist spannend zu sehen, dass Versuche existieren in denen eine sogenannte „vehicle-to-vehicle-to-infrastructure“ [10] Kommunikation in ihrer Architektur beschrieben wurden. Auf diese Arbeit möchte ich mich auch beziehen, da ich gegebenenfalls parallelen in Hinsicht auf meine Umsetzung ziehen kann.

Da mit der Bachelorarbeit das Verkehrsproblem innerhalb von Metropolen angesprochen wird, möchte ich bezugnehmend mittels der Simulationslösung [11] ein Konzept zur Fahrzeugkommunikation entwerfen. Dieses Konzept bedient sich an einem sogenannten „shared information space“, welcher in [12] [13] diskutiert wird. Weiterhin soll durch das Zusammenspiel des „Super“-Fahrzeugs und den anderen Verkehrsteilnehmern das übergeordnete Problem des Verkehrs beseitigt bzw. verbessert werden. Da mich dies stark an das Verhalten eines Kollektivs (etwa Fischschwärme oder Bienenschwärme) erinnert hat, möchte ich auf kollektive Intelligenz („Schwarmintelligenz“) Bezug nehmen und klären, inwiefern die vernetzten Fahrzeuge als Kollektiv agieren und somit ein Problem gemeinsam lösen, wobei jede Partei einen Teil beiträgt und gleichermaßen profitiert [14] [15] [16].

Abkürzungsverzeichnis

PoC	Proof of Concept
V2I	Vehicle to infrastructure
OEM	Original equipment manufacturer
GLOSA	Green light optimal speed advisory
SPAT	Signal phase and timing
TTG	Time to green
CP	Content provider
V2V	Vehicle to vehicle

Literaturverzeichnis

[1] Audi AG, Hrsg., „Ampelinformationen online | Audi connect“, Zugegriffen: 4. Dezember 2022. [Online]. Verfügbar unter: https://www.audi.de/de/brand/de/service-zubehoer/connect/ampelinformation-online.html

[2] V. Nguyen, O. T. T. Kim, T. N. Dang, S. I. Moon, und C. S. Hong, „An efficient and reliable Green Light Optimal Speed Advisory system for autonomous cars“, in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Okt. 2016, S. 1–4. doi: 10.1109/APNOMS.2016.7737260.

[3] R. Bodenheimer, A. Brauer, D. Eckhoff, und R. German, „Enabling GLOSA for adaptive traffic lights“, in *2014 IEEE Vehicular Networking Conference (VNC)*, Dez. 2014, S. 167–174. doi: 10.1109/VNC.2014.7013336.

[4] R. Bodenheimer, D. Eckhoff, und R. German, „GLOSA for adaptive traffic lights: Methods and evaluation“, in *2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM)*, Okt. 2015, S. 320–328. doi: 10.1109/RNDM.2015.7325247.

[5] D. Eckhoff, B. Halmos, und R. German, „Potentials and limitations of Green Light Optimal Speed Advisory systems“, in *2013 IEEE Vehicular Networking Conference*, Dez. 2013, S. 103–110. doi: 10.1109/VNC.2013.6737596.

[6] R. Stahlmann, M. Möller, A. Brauer, R. German, und D. Eckhoff, „Technical evaluation of GLOSA systems and results from the field“, in *2016 IEEE Vehicular Networking Conference (VNC)*, Dez. 2016, S. 1–8. doi: 10.1109/VNC.2016.7835967.

[7] T. Tielert, M. Killat, H. Hartenstein, R. Luz, S. Hausberger, und T. Benz, „The impact of traffic-light-to-vehicle communication on fuel consumption and emissions“, in *2010 Internet of Things (IOT)*, Nov. 2010, S. 1–8. doi: 10.1109/IOT.2010.5678454.

[8] K. Reif, Hrsg., *Fahrstabilisierungssysteme und Fahrerassistenzsysteme*. Wiesbaden: Vieweg+Teubner, 2010. doi: 10.1007/978-3-8348-9717-6.

[9] „Final Report to Tranport Canada: I2V Connected Vehicle Pilot Project - City Fleet - Signalized Intersection Approach and Departure Optimization Application“, Jan. 2020. Zugegriffen: 4. Dezember 2022. [Online]. Verfügbar unter: https://tcdocs.ingeniumcanada.org/sites/default/files/2020-05/City%20of%20Ottawa%20-%20I2V%20Connected%20Vehicle%20Pilot%20Project%20-%20City%20Fleet%20-%20Signalized%20Intersection%20Approach%20and%20Departure%20Optimization%20Application.pdf

[10] J. Miller, „Vehicle-to-vehicle-to-infrastructure (V2V2I) intelligent transportation system architecture“, in *2008 IEEE Intelligent Vehicles Symposium*, Juni 2008, S. 715–720. doi: 10.1109/IVS.2008.4621301.

[11] P. Alvarez Lopez *u. a.*, „Microscopic Traffic Simulation using SUMO“, in *2005 IEEE Intelligent Transportation Systems Conference (ITSC)*, Maui, USA, Nov. 2018. Zugegriffen: 6. Dezember 2022. [Online]. Verfügbar unter: https://elib.dlr.de/124092/

[12] U. M. Borghoff und J. H. Schlichter, „Communication Systems and Shared Information Spaces“, in *Computer-Supported Cooperative Work: Introduction to Distributed Applications*, U. M. Borghoff und J. H. Schlichter, Hrsg. Berlin, Heidelberg: Springer, 2000, S. 285–325. doi: 10.1007/978-3-662-04232-8_6.

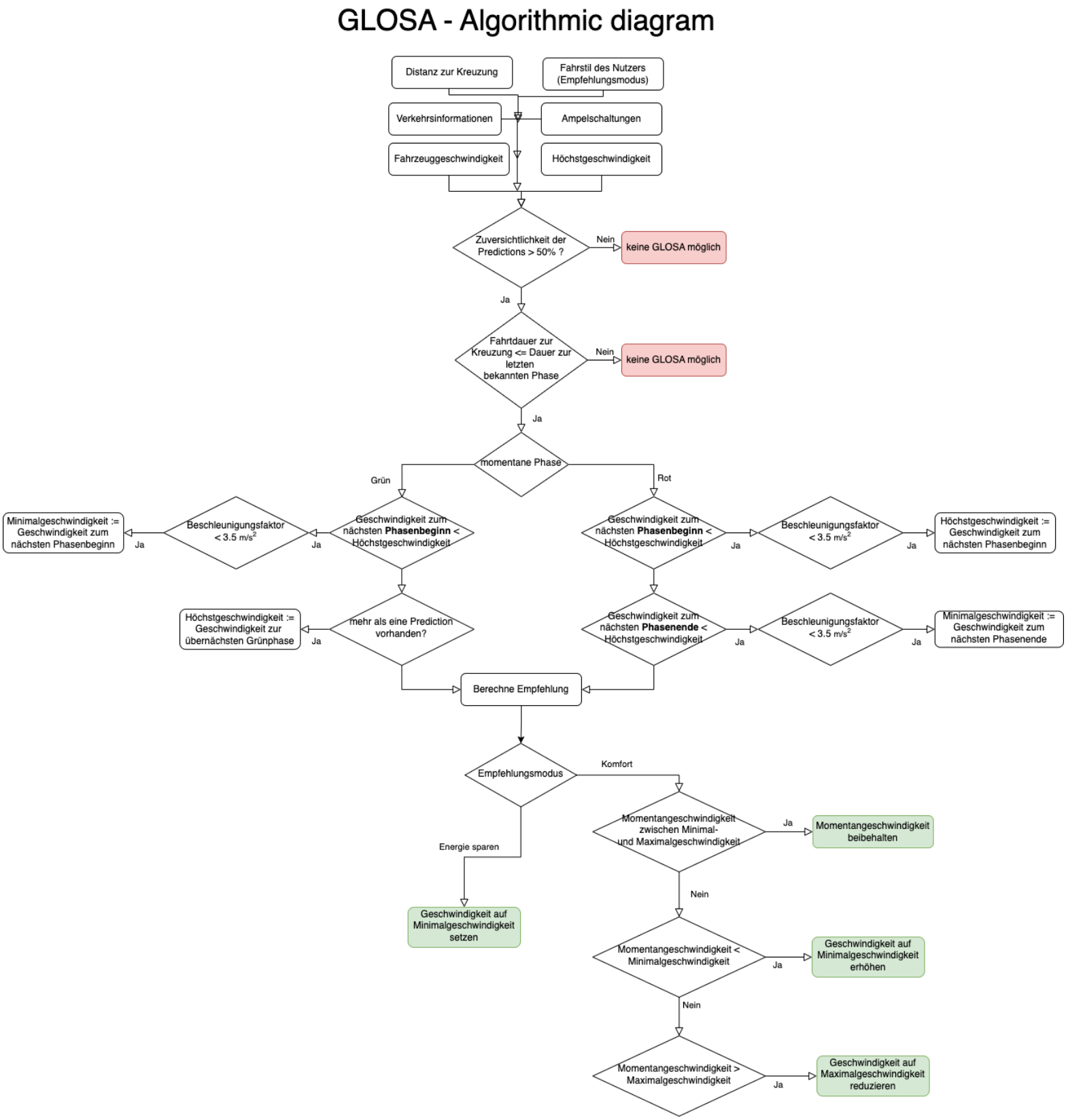
[13] J. G. Davis, E. Subrahmanian, S. Konda, H. Granger, M. Collins, und A. W. Westerberg, „Creating Shared Information Spaces to Support Collaborative Design Work“, *Inf. Syst. Front.*, Bd. 3, Nr. 3, S. 377–392, Sep. 2001, doi: 10.1023/A:1011469727367.

[14] J. M. Leimeister, „Kollektive Intelligenz“, *WIRTSCHAFTSINFORMATIK*, Bd. 52, Nr. 4, S. 239–242, Aug. 2010, doi: 10.1007/s11576-010-0234-2.

[15] H. Hamann, *Schwarmintelligenz*. Berlin, Heidelberg: Springer, 2019. doi: 10.1007/978-3-662-58961-8.

[16] Q. Li, W. Shangguan, B. Cai, und L. Chai, „Traffic Flow Guidance and Optimization of Connected Vehicles Based on Swarm Intelligence“, in *2019 Chinese Control Conference (CCC)*, Juli 2019, S. 2099–2104. doi: 10.23919/ChiCC.2019.8866595.

Anhang


GLOSA - Algorithmic diagram
Distanz zur Kreuzung
Fahrstil des Nutzers (Empfehlungsmodus)
Verkehrsinformationen
Ampelschaltungen
Fahrzeuggeschwindigkeit
Höchstgeschwindigkeit
Zuversichtlichkeit der Predictions > 50% ?
Nein
keine GLOSA möglich
Ja
Fahrtdauer zur Kreuzung <= Dauer zur letzten bekannten Phase
Nein
keine GLOSA möglich
Ja
momentane Phase
Grün
Rot
Minimalgeschwindigkeit := Geschwindigkeit zum nächsten Phasenbeginn
Ja
Beschleunigungsfaktor < 3.5 m/s²
Ja
Geschwindigkeit zum nächsten Phasenbeginn < Höchstgeschwindigkeit
Geschwindigkeit zum nächsten Phasenbeginn < Höchstgeschwindigkeit
Ja
Beschleunigungsfaktor < 3.5 m/s²
Ja
Höchstgeschwindigkeit := Geschwindigkeit zum nächsten Phasenbeginn
Höchstgeschwindigkeit := Geschwindigkeit zur übernächsten Grünphase
Ja
mehr als eine Prediction vorhanden?
Geschwindigkeit zum nächsten Phasenende < Höchstgeschwindigkeit
Ja
Beschleunigungsfaktor < 3.5 m/s²
Ja
Minimalgeschwindigkeit := Geschwindigkeit zum nächsten Phasenende
Berechne Empfehlung
Empfehlungsmodus
Komfort
Energie sparen
Momentangeschwindigkeit zwischen Minimal- und Maximalgeschwindigkeit
Ja
Momentangeschwindigkeit beibehalten
Nein
Geschwindigkeit auf Minimalgeschwindigkeit setzen
Momentangeschwindigkeit < Minimalgeschwindigkeit
Ja
Geschwindigkeit auf Minimalgeschwindigkeit erhöhen
Nein
Momentangeschwindigkeit > Maximalgeschwindigkeit
Ja
Geschwindigkeit auf Maximalgeschwindigkeit reduzieren