HAW HAMBURG

BACHELORTHESIS
Nils Eggebrecht

Autonome Zurückführung nach Verlassen der Fahrbahn für autonome Modellfahrzeuge

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Nils Eggebrecht

Autonome Zurückführung nach Verlassen der Fahrbahn für autonome Modellfahrzeuge

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Technische Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stephan Pareigis
Zweitgutachter: Prof. Dr. Tim Tiedemann

Eingereicht am: 28. Mai 2020

Nils Eggebrecht

Thema der Arbeit

Autonome Zurückführung nach Verlassen der Fahrbahn für autonome Modellfahrzeuge

Stichworte

Autonomes Fahren, Straßenerkennung, Fahrbahn Rückführung, Modellfahrzeug, Kamera, Carolo-Basic-Cup

Kurzzusammenfassung

Diese Arbeit behandelt die Entwicklung eines Systems zur autonomen Zurückführung eines Modellfahrzeugs zur Straße. Dabei wird ein Modellfahrzeug im Maßstab 1:10, das autonom einer Straße folgt, nach einem fehlerhaften Verlassen der Fahrbahn zur rechten Fahrspur zurückgeführt. Das entwickelte System entstand als Lösungsansatz für den Carolo-Basic-Cup, dessen Regeln die Rahmenbedingungen für das autonome Fahren dieser Arbeit bilden. Die verwendete Fahrzeugplattform und Software-Basis, auf der das System zur autonomen Zurückführung aufbaut, werden analysiert. Anschließend werden als Hauptbestandteil die autonome Straßenerkennung und das Zurückfahren zur Straße beschrieben. Das Gesamtsystem wird in einem Live-Test auf einer Teststrecke evaluiert, wofür ein Modul entwickelt wurde, das ein Verlassen der Straße erzwingt.

Nils Eggebrecht

Title of Thesis

Autonomous return after leaving the lane for autonomous model vehicles

Keywords

Autonomous driving, Road detection, Road return, Model vehicle, camera, Carolo-Basic-Cup

Abstract

This work deals with the development of a system for the autonomous return of a model vehicle to the road. A 1:10 scale model vehicle that autonomously follows a road is

guided back to the right lane after incorrectly leaving the lane. The developed system was built as a solution for the Carolo-Basic-Cup, whose rules build the framework for the autonomous driving for this work. The vehicle platform and software basis used to build the autonomous return system are analyzed. Then, autonomous road detection and road retrun are described as the main components. The entire system is evaluated in a live-test, for which a module has been developed that forces the vehicle to leave the road.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

BTR BackToRoad.

FRP Fehlerrouten-Punkt.

FSM Finite State Machine.

HAW Hochschule für Angewandte Wissenschaften.

IMU Inertial Measurement Unit.

LA linke Außenlinie.

LS linke Fahrspur.

M Mittellinie.

MS Mittellinien-Segment.

RA rechte Außenlinie.

RP Routen-Punkt.

RS rechte Fahrspur.

ToF Time of Flight Sensor.

VESC Vedder Electronic Speed Controller.

VP Vergleichspunkt.

Codebeispielverzeichnis

1 Einleitung

Diese Arbeit entstand im Zusammenhang mit einer Teilnahme am Carolo-Basic-Cup 2020. Der Carolo-Basic-Cup ist ein studentischer Wettbewerb, der eine Plattform bietet, sich mit der Konzeption und Implementierung automatisierter Modellfahrzeuge zu befassen. Die Herausforderung besteht darin, das leistungsstärkste Fahrzeugleitsystem im Maßstab 1:10 für verschiedene Szenarien zu realisieren. Die Anforderungen werden in Anlehnung an realistische Straßenumgebungen und Situationen gesetzt. Im jährlichen Wettbewerb haben die teilnehmenden Studierenden die Möglichkeit, ihre Kompetenzen vor einer Jury, bestehend aus Mitgliedern der Bereiche Industrie und Wissenschaft, zu präsentieren[1].

Einen Einblick in die Herausforderungen des autonomen Fahrens wurde durch eine erstmalige Teilnahme des Autors mit dem „TeamWorstCase“ am Carolo-Basic-Cup im Jahr 2018 erlangt. Im Wettkampf musste wiederholt mit der Fernsteuerung eingegriffen werden, da das Fahrzeug die Fahrbahn verlassen hat. Ein Eingriff in den Betrieb des Fahrzeugs mittels einer Fernsteuerung darf dabei laut Regelwerk erst dann erfolgen, wenn das Fahrzeug für eine Sekunde still stand. Der manuelle Eingriff und die damit verbundene limitierte Bewegungsgeschwindigkeit von $0{,}3\,m/s$, die im Carolo-Basic-Cup Regelwerk festgelegt ist, haben jedoch eine Auswirkung auf den Faktor Zeit. Durch langsame Bewegungen konnte das Fahrzeug nicht schnell zurück auf die Straße fahren, dies führte dementsprechend zu hohen Zeitverlusten in der Bewertung.

Um diese Geschwindigkeitsbegrenzung zu umgehen, wird im folgenden Dokument eine Softwarelösung beschrieben, die es ermöglicht, dass das Fahrzeug in die Lage versetzt, autonom zurück zur Fahrbahn zu finden. Dies ist vorteilhaft, da es im autonomen Betrieb keine Geschwindigkeitsbegrenzung gibt.

Diese Arbeit widmet sich dem Ziel, durch eine geeignete Software eine autonome Zurückführung des Modellfahrzeugs auf die Fahrbahn zu ermöglichen und daraus resultierend Zeit zu sparen, um wichtige Punkte zu sammeln.

Das System bietet eine Lösung für die zwei Fälle:

1. Das Fahrzeug verlässt mit kontrollierter Geschwindigkeit, sowie sensorisch korrekt erfasster Position und Ausrichtung, also ohne Rutschen und Durchdrehen der Reifen, die Fahrbahn.

2. Das Fahrzeug verlässt mit unkontrollierter Geschwindigkeit, sowie fehlerhaft erfasster Position und Ausrichtung durch zum Beispiel Rutschen und oder Durchdrehen der Reifen, die Fahrbahn.

Ziel ist es, dass das Fahrzeug nach Auftreten der genannten Fälle autonom zurück auf die Fahrbahn fährt und sich auf der rechten Fahrspur positioniert.

Als Abgrenzung wird im Rahmen dieser Arbeit keine Erkennung implementiert und verwendet, die das Verlassen der Fahrbahn detektiert. Eine solche Erkennung könnte wie in der Masterarbeit „Systemidentifikation eines autonomen Fahrzeugs mit einer robusten, kamerabasierten Fahrspurerkennung in Echtzeit“ von Eike Jenning erreicht werden [4]. In dieser wird „[...] die Fahrspur durch ein kubisches Polynom approximiert und anschließend die Fahrzeuglage in der Fahrspur, der tangentiale Steuerkurswinkel des Fahrzeugs zur Fahrspur und der Kurvenradius der Fahrspur identifiziert“ [4, vgl. iii], wodurch ein bevorstehendes Verlassen der Fahrbahn signalisiert werden könnte [4, vgl. S. 33-36].

Grundlage und Voraussetzung für diese Arbeit ist, dass das Erkennen des Verlassens der Fahrbahn durch den Bediener erfolgt und über die Fernsteuerung dem System mitgeteilt wird. Die autonome Zurückführung zur Straße wird durch den manuellen Eingriff eingeleitet, aber es soll erreicht werden, dass das Fahrzeug nicht manuell mit der Fernsteuerung gesteuert werden muss. Durch autonomes Zurückfahren soll, im Vergleich zum Fahren per Fernsteuerung, welches nur mit niedriger Geschwindigkeit erlaubt ist, Zeit eingespart werden. Da diese Arbeit im Zusammenhang mit einer Teilnahme am Carolo-Basic-Cup 2020 entwickelt wurde, werden als Basis für den Aufbau der Modellstrecken die Regeln aus dem Carolo-Basic-Cup verwendet. Das Modellfahrzeug im Maßstab 1:10 welches vom Team „TeamWorstCase“ für den Carolo-Basic-Cup 2020 entwickelt wurde, dient als Basis für diese Arbeit. Die Software dieser Arbeit, die ausschließlich in C++ entwickelt wurde, ist ein Teil der Software, die für den Wettbewerb verwendet wurde. Die Bilderkennung und -verarbeitung wird mit Hilfe der Softwarebibliothek OpenCV (Open Source Computer Vision Library) umgesetzt. OpenCV stellt eine Infrastruktur für Computer Vision-Anwendungen bereit und verfügt über optimierte Algorithmen [11]. Beim

Carolo-Cup wird OpenCV auch von weiteren Teams verwendet, z.B. vom Team „KITcar“, das für das Karlsruher Institut für Technologie antritt [10].

Die Arbeit ist wie folgt gegliedert und strukturiert: Zu Beginn (Kapitel 2) werden die Grundlagen eines zulässigen Modellfahrzeugs, sowie die Straßensituation nach dem Regelwerk des Carolo-Basic-Cups beschrieben. Kapitel 3 befasst sich daraufhin mit der Systemanalyse. In diesem Kontext wird die Fahrzeugplattform mit der Hardware und der vorhandenen Rechenleistung vorgestellt. Weiterhin werden hier die Grundlagen der Software erklärt, auf denen diese Arbeit basiert. Kapitel 4 behandelt die autonome Straßenerkennung mit den Merkmalen und der eindeutigen Erkennung von Straßen. In Kapitel 5 wird dann eine Evaluation der Funktionalität thematisiert. Der Abschluss (Kapitel 6) beinhaltet ein Fazit inklusive einer Bewertung der erreichten Funktionalität, sowie möglichen Verbesserungen.

Das ganze System ist stark an das Regelwerk des Carolo-Basic-Cups angelehnt und ist damit schwer auf andere Einsatzmöglichkeiten portierbar. In einer realen Straßensituation ist es beispielsweise undenkbar, dass ein System, das keine Informationen über die Situation hinter dem Fahrzeug hat, rückwärts fährt. Außerdem kann im realen Straßenverkehr nicht davon ausgegangen werden, dass die Situation im Laufe der Zeit unverändert bleibt.

2 Grundlagen des Carolo-Basic-Cup-Regelwerks

In diesem Kapitel werden die Grundlagen des Carolo-Basic-Cup-Regelwerks, die als Rahmenbedingungen der Umsetzung dienen, erläutert.

2.1 Modellfahrzeuge

Bei dem Fahrzeug muss es sich laut Regelwerk um ein „zwei Achsen 1:10 Modell“ mit einem Elektromotor handeln. Die einzuhaltenden Maße des Fahrzeugs sind in Tabelle 2.1 aufgelistet. Dabei ist zu beachten, dass die Spurbreite von der Mitte der Räder aus gemessen wird [1, vgl. S. 9].

Das Fahrzeug muss, wie im realen Straßenverkehr, unterschiedliche Fahrmanöver signalisieren können. Dazu gehört das aktive Bremsen, für das am Heck des Fahrzeugs drei gut sicht- und unterscheidbare Bremslichter installiert sein müssen. Außerdem muss das Fahrzeug beim Überholen, Wenden oder Parken mit Hilfe eines gelben / orangefarbenen Lichtes auf der richtigen Seite mit einer Frequenz von maximal $2\,Hz$, blinken. Zusätzlich zu der Fahrzeugbeleuchtung in Anlehnung an den realen Straßenverkehr muss es eine weitere Leuchte geben, welche am höchsten Punkt des Fahrzeugs positioniert ist. Die

Maß	Wert in mm
max. Breite	300
max. Höhe	300
min. Radstand	200
min. Spurbreite	160

Tabelle 2.1: Abmessungen des Fahrzeugs

Leuchte blinkt mit einem deutlich sichtbaren blauen Licht, sobald das Fahrzeug mit der Fernsteuerung kontrolliert wird [1, vgl. S. 11].

Der relevanteste Vorteil, der durch das Verwenden von Modellfahrzeugen entsteht, ist die Größe. Damit ist zum Beispiel eine Teststrecke in einem kleineren Raum umsetzbar. Eine Teststrecke mit einer Länge von ca. $30\,m$, wie sie für diese Arbeit zur Verfügung steht, müsste für ein normales Fahrzeug $300\,m$ lang sein. Außerdem ist das Verletzungsrisiko bei einem 1:10 Modellfahrzeug stark reduziert, da die Kraft bei einem Modellfahrzeug mit einem Gewicht von etwa $2{,}5\,kg$ deutlich geringer ist und kein Fahrer im Fahrzeug benötigt wird.

2.2 Eingriff per Fernsteuerung

Ein Eingriff mit einer Fernsteuerung ist laut Regelwerk nach einem eindeutigen Fehlverhalten des Fahrzeugs erlaubt. Ein Beispiel dafür ist das vollständige Verlassen der Fahrbahn. In diesem Fall darf ein Teammitglied die Fernsteuerung dafür verwenden, das Fahrzeug mit einer Geschwindigkeit von $0{,}3\,m/s$ zurück auf die Fahrspur zu steuern. Bevor das Fahrzeug per Fernsteuerung kontrolliert werden darf, muss es eine Sekunde im Stillstand stehen. Es dürfen keine weiteren Funktionen über die Fernsteuerung umgesetzt werden [1, vgl. S. 10].

2.3 Straßensituation

In diesem Kapitel werden die verschiedenen Straßensituationen anhand des Carolo-Basic-Cup Regelwerks beschrieben [1, vgl. S. 15-17]. Das Softwaresystem des Modellfahrzeugs soll demnach so angepasst werden, dass es in den folgenden Situationen die Straße sowohl erkennen, als ihr auch folgen kann.

Nach dem Regelwerk ist die Komplexität der Straßensituationen begrenzt und definiert. Die Straßensituationen bestehen aus einer Straße mit je zwei parallelen Fahrspuren, eine für jede Fahrtrichtung. Diese vorgegebene Straßensituation soll eine ländliche Straßenumgebung imitieren, die aus geraden Abschnitten, Kurven, Kreuzungen und einer Parkzone mit mehreren Parkplätzen besteht. Alle Straßenmarkierungen sind, sofern nicht anders angegeben, weiß und zwischen $18\,mm$ und $20\,mm$ breit. Es gibt weiterhin eine Startlinie (karierte Linie von ca. $50\,mm$ Breite), die den Beginn der Strecke markiert und die $10\,m$

lange Parkzone einleitet [1, vgl. S. 15-17]. Stopp-Linien können den Eintritt in Kreuzungen anzeigen und sind 36 bis 40 mm breit [1, vgl. S. 20]. Start- und Stopplinien, sowie Hindernisse werden unter dem Begriff Features zusammengefasst. An beliebigen Stellen können bis zu zwei Fahrspurmarkierungen gleichzeitig für maximal 1000 mm fehlen. Ausnahme bilden dabei Kreuzungen und die Parkzone [1, vgl. S. 16,17]. Hindernisse bestehen aus weißer Pappe und sind rechteckig. Die minimale Länge beträgt 10 cm, die Breite des Hindernisses liegt zwischen 10 cm und 40 cm und die Höhe zwischen 10 cm und 24 cm [1, vgl. S. 29].

Eine mögliche Straßensituation ist in Abbildung 2.1 zu sehen.

Für die Zurückführung zur Straße sind Parklücken, Start- und Stopplinien insofern relevant, als dass die Linien nicht versehentlich für die Straßenerkennung verwendet werden sollen.

2.3.1 Fahrspurmarkierungen

Mit Hilfe der Straßenmarkierungen werden die Fahrspuren dargestellt, die für die Straßenerkennung relevant sind, da sich die Parameter der Software aus den Größen der folgenden Maße ergeben. Jede Fahrspur hat eine Breite von 350 mm bis 450 mm, gemessen von den Innenseiten der Außen- und Mittellinie. Beide Fahrspuren sind durch eine gestrichelte Mittellinie getrennt, die nach 200 mm Länge für weitere 200 mm unterbrochen ist. Diese Form setzt sich fort, bis eine Kreuzung oder die Startlinie erreicht ist, sodass die Mittellinie an diesen Punkten möglicherweise mit einer Lücke endet. Die linken und rechten Spurgrenzen sind mittels durchgezogener weißer Linien markiert. Benachbarte Streckenabschnitte haben einen Abstand von mindestens 50 mm, gemessen an den Außenkanten der Fahrbahnmarkierungen. Der minimale Abstand der Strecke zum Ende des definierten Bereiches beträgt 300 mm. Die Kurve mit dem kleinsten Radius hat einen Innenradius von 1000 mm. Es muss immer mindestens eine Straßenmarkierung vorhanden sein, dies betrifft die Außenlinien und die Mittellinie. Die Gestaltung des Bereichs außerhalb der Straße bleibt undefiniert. Darüber hinaus können sich Artefakte in Form von Objekten oder Resten von Fahrspurmarkierungen außerhalb des Straßenbereichs befinden. Der minimale Abstand zwischen Artefakten und gültigen Fahrspurmarkierungen beträgt 100 mm [1, S. 16, 17].

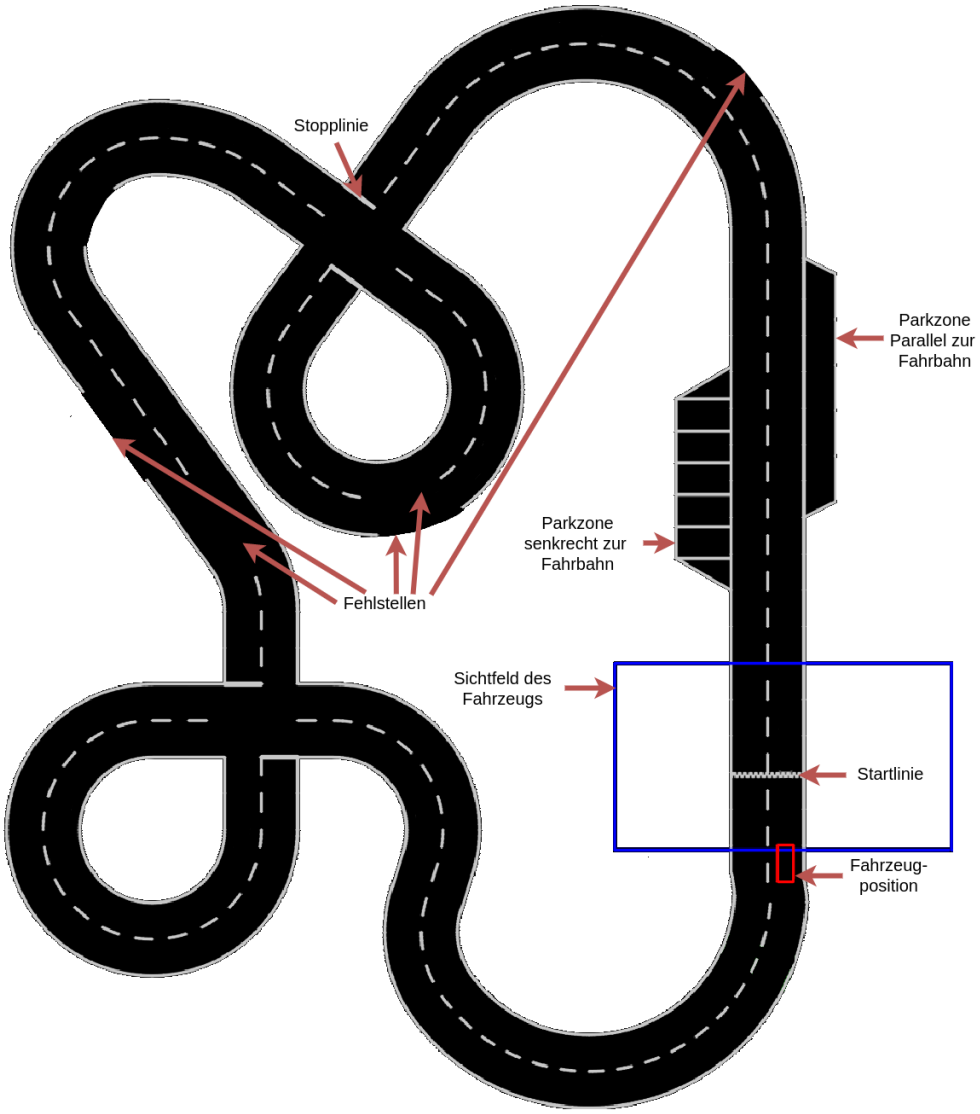

Abbildung 2.1: Mögliche Straßensituation

Die autonome Straßenerkennung (Kapitel 4) basiert auf den beschriebenen Straßensituationen. Die Parameter für die Software wiederum berechnen sich aus den oben genannten Maßangaben der Straßensituationen.

3 Analyse

In der Analyse wird das bereits vorhandene System, bestehend aus Hardware und Software, das als Grundlage für diese Arbeit verwendet wird, beschrieben. Das verwendete Fahrzeug und die Software-Basis werden erläutert.

3.1 Fahrzeugplattform

In den folgenden Kapiteln wird auf die Fahrzeugplattform eingegangen, auf der diese Arbeit aufbaut. Dabei werden die Hardware-Basis, die Rechenleistung, sowie die verbauten Sensoren und Aktoren beschrieben.

3.1.1 Hardware-Basis

Es wird das Modellfahrzeug „Flatmobil“ verwendet, das für den Carolo-Basic-Cup 2020 vom „TeamWorstCase“ entwickelt wurde. Das Chassis stammt vom Hobby Modellfahrzeug „RS4 Sport 3 Drift SUBARU BRZ“ und wurde vom teilnehmenden Team erweitert [5]. Es handelt sich dabei um ein 1:10 Elektro-Modellauto mit Allradantrieb, der mit einem Kardan Antriebsstrang funktioniert. Der Aufbau des „RS4 Sport 3“ wurde durch ein eigens entworfenes Design ersetzt und ist in Abbildung 3.1 zu sehen.

3.1.2 Rechenleistung

Damit auf dem Fahrzeug eine autonome Erkennung berechnet werden kann, steht dem Fahrzeug die Rechenleistung eines leistungsstarken Computers, dem Intel NUC I5 der 8. Generation mit $8\,GB$ Arbeitsspeicher zur Verfügung [2]. Zur Ansteuerung der Sensorik, des Fernsteuerungsempfängers, sowie der Beleuchtung wird ein Mikrocontroller STM32 [8] verwendet, der mit dem Intel NUC kommuniziert.

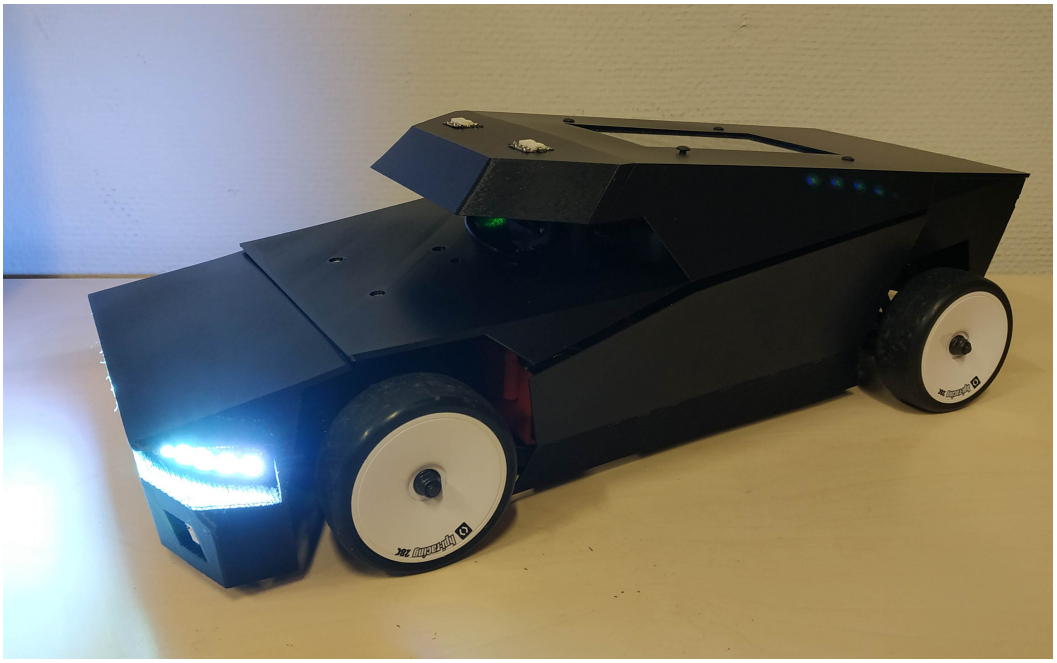

Abbildung 3.1: Modellfahrzeug „Flatmobil“

3.1.3 Sensoren

Für die autonome Erkennung der Straße wird Sensorik benötigt, mit der die Umgebung des Fahrzeugs wahrgenommen werden kann. Für die Bilderkennung wird eine monochrome IDS Industrial USB 3.0 Kamera mit einem Weitwinkelobjektiv verwendet [9]. Zusätzlich wird die zurückgelegte Strecke mittels eines Odometers bestimmt. Auf dem Fahrzeug ist außerdem ein optischer Time of Flight Sensor (ToF)-Sensor verbaut, der auf der rechten Seite, senkrecht zur Fahrtrichtung die Entfernung zwischen Fahrzeug und dem nächsten Objekt misst. Bezüglich der Zurückführung zur Straße gibt es jedoch keinen Verwendungszweck für den ToF-Sensor. Mit der Inertial Measurement Unit (IMU) MPU 9025 wird die Ausrichtung des Fahrzeugs festgestellt [3].

Die Kamera ist direkt an den Intel NUC angeschlossen. Die anderen genannten Sensoren werden über den Mikrocontroller angesprochen.

3.1.4 Aktoren

Die angesteuerten Aktoren für den Motor sind ein digitaler Servo und ein Vedder Electronic Speed Controller (VESC) für die Ansteuerung des Motors. Weitere Aktoren sind die LED-Scheinwerfer für eine bessere Hinderniserkennung, sowie die LEDs für die Bremslichter und Blinker. Ein Display auf dem Fahrzeug zeigt Statusinformationen vom laufenden Betrieb des Systems, mit denen das Verhalten des Fahrzeugs nachvollzogen werden kann. Der Touchscreen des Displays wird auch zur Eingabe der Modi-Auswahl verwendet und fungiert somit als Aktor und Sensor.

In Abbildung 3.2 sind die einzelnen Komponenten des Fahrzeugs zu erkennen. Der Akku ist wechselbar und wird auf der linken Seite des Fahrzeugs auf Höhe des Motors montiert. Der Spannungswandler befindet sich hinter dem Fernsteuerungsempfänger(9).

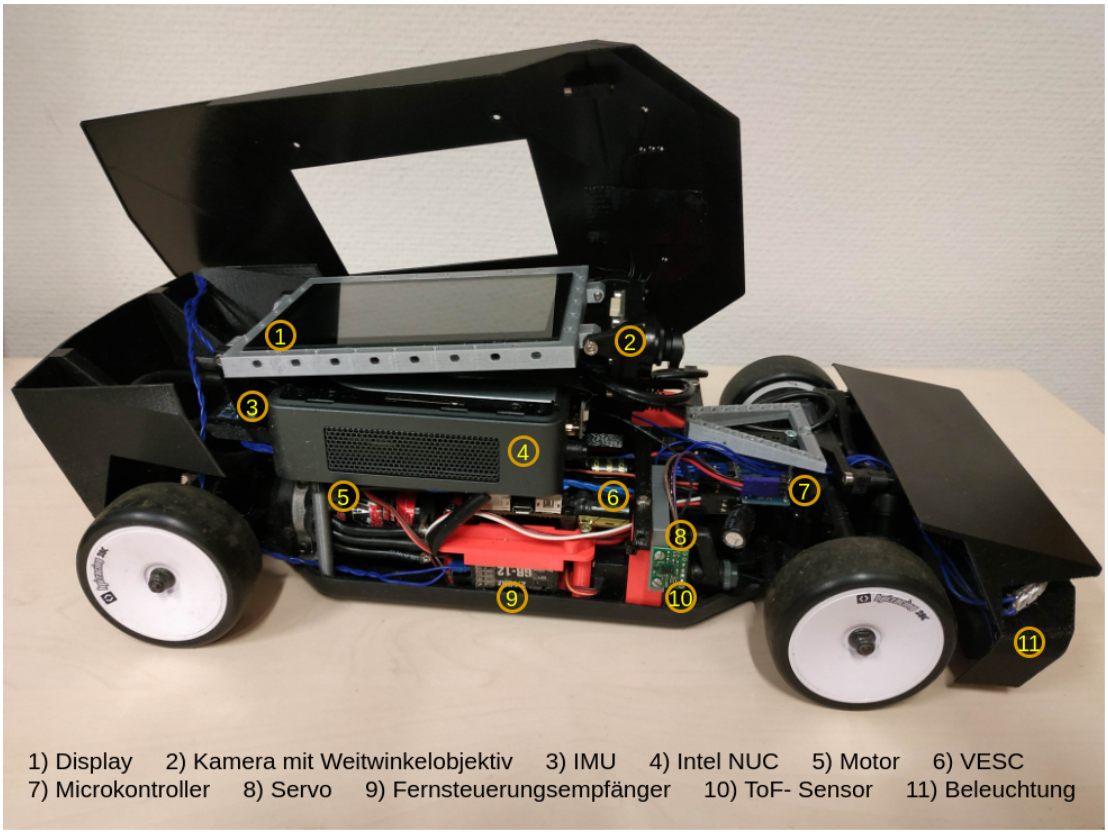

Abbildung 3.2: Übersicht der Komponenten

3.2 Software-Basis

Damit aus der vorhandenen Hardware ein autonomes System werden kann, wird eine Software benötigt, welche die Sensorik auswertet, die Informationen verarbeitet und die Aktoren ansteuert.

Das Projekt basiert auf einer Software für das autonome Fahren, die zeitgleich vom Team „TeamWorstCase“ zu dieser Arbeit für den Carolo-Basic-Cup 2020 entwickelt wurde. Die Software für das autonome Fahren liest die Bilder der Kamera ein und führt eine Bildvorverarbeitung durch, die auch für diese Arbeit mitverwendet wird. In Kapitel 4 wird auf die Auswahl des Bildes eingegangen, auf dem die Straßenerkennung für das Zurückfinden zur Straße entwickelt wird. Die Software für das autonome Fahren bietet ebenfalls das Interface zur Ansteuerung der Sensoren und Aktoren, sowie das Abfahren einer geplanten Route.

3.2.1 Struktur der Software für das autonome Fahren

In diesem Kapitel geht es um die Software, die das autonome Fahren des Fahrzeugs ermöglicht und als Basis dieser Arbeit dient. Die Software für die Zurückführung zur Straße wird gesondert in Kapitel 4 beschrieben.

Die einzelnen Aufgaben, wie z.B. freies Fahren, Suchen eines Parkplatzes, Einparken, Überholen oder an einer Kreuzung Vorfahrt gewähren, sind als Zustände in einer Finite State Machine (FSM) umgesetzt. Für das Zurückfahren zur Straße wird darüber hinaus in der FSM ein Zustand „BacktoRoad“ hinzugefügt.

Die Bildverarbeitung und die Erkennung der Straße, sowie der Features, zu denen die Hindernisse, sowie Start- und Stopp-Linien gehören, funktioniert über eine Pipeline-Struktur, in der die Bilder in verschiedenen Schritten verarbeitet und weitergereicht werden.

In der Abbildung 3.3 wird die Struktur für das autonome Fahren in einem Datenflussdiagramm dargestellt. Der erste Schritt der Pipeline ist das Einlesen des Bildes mit der Kamera. Danach wird in dem Modul Transformation das Bild in die Vogelperspektive transformiert und mit verschiedenen Filtern zu einem binärisierten Bild verarbeitet. Der dritte Schritt der Pipeline ist die Straßen- und Featureerkennung, welche jedoch nicht für die Erkennung der autonomen Zurückfindung zur Straße verwendet wird. Die gefundene Straße, sowie die einzelnen Features, wie Hindernisse und Start- und Stopp-Linien,

werden in eine Worldmap geschrieben. Die FSM läuft parallel zur beschriebenen Pipeline und arbeitet auf den Daten der Worldmap und übergibt dem Driver die Route. Der Driver berechnet mit der Route die Steuerbefehle und reicht sie an den Mikrocontroller zur Steuerung der Hardware weiter.

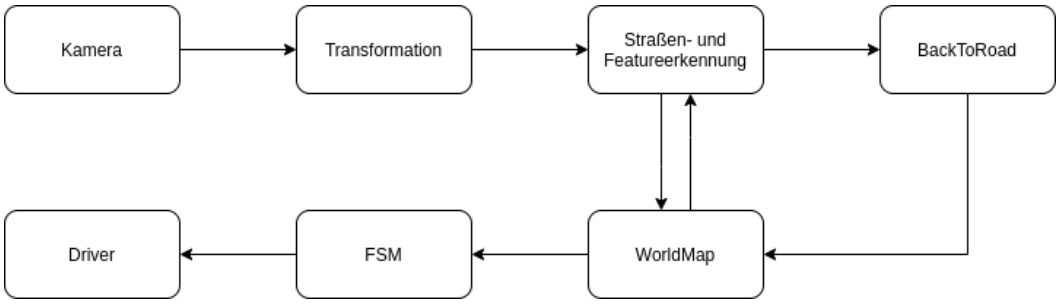

Abbildung 3.3: Struktur der Software für das autonome Fahren

In der Worldmap wird die im Bild erkannte Straße mit Hilfe der gefahrenen Strecke und der Ausrichtung des Fahrzeugs mit der bereits bekannten Straße aus der Worldmap zusammengeführt. Die gefahrene Strecke wird mit dem Odometer gemessen und die Ausrichtung des Fahrzeugs durch die IMU erfasst. Das Zusammenführen der Strecke baut eine Historie der zurückgelegten Stecke auf. Durch das Speichern der Informationen von jeder aktuellen Aufnahme und dem Zusammenfügen dieser, wird erreicht, dass Features wie z.B. Hindernisse gespeichert werden und deren Positionen bekannt bleiben. Wenn beispielsweise ein statisches Hindernis überholt wird, kann jederzeit in der Worldmap nachgeschaut werden, welche Position es hat und es muss nicht mehr vom Routenplaner zwischengespeichert werden. Würde die Information weder vom Routenplaner, noch in der Worldmap festgehalten werden, würde der Routenplaner den Überholvorgang abbrechen, sobald das Fahrzeug auf der linken Spur neben dem Hindernis fährt, da es damit nicht mehr im Sichtbereich des Bildes wäre. Falsch erkannte Elemente, die einmal zu einem Fehlverhalten geführt haben, sollen nicht in einer weiteren Runde wieder zum gleichen Fehler führen. Deshalb hat die Worldmap eine maximale Länge von $10\,m$. Die Daten der Worldmap bleiben $10\,m$ gefahrene Strecke bestehen, danach werden die ältesten Daten wieder verworfen und durch neue ersetzt.

Auf Basis der Informationen aus der Worldmap wird eine Route geplant, die Hindernisse und andere Features berücksichtigt. Die geplante Route wird dann vom Driver abgefahren. Der Driver ist ein Softwaremodul, welches die Geschwindigkeit und den Lenkwinkel berechnet und diese Informationen weiter an den Mikrocontroller gibt. Dafür wird die geplante Route aus der Worldmap genommen und mit höherer Geschwindigkeit verarbei-

tet, als die Bildrate der Kamera. Dadurch ist es möglich, auch ohne neue Informationen aus einem Bild verschiedene Steuersignale wie Lenkwinkel und Geschwindigkeit an den Mikrocontroller zu senden, der diese weiter an die Hardware sendet. Sollte die Kamera im ungünstigsten Fall keine neuen Bilder liefern, bleibt das Fahrzeug am Ende der berechneten Route stehen. Der Driver berechnet dabei in jedem Durchgang für die geplante Route eine maximale Geschwindigkeit. Durch häufiges aktualisieren der maximalen Geschwindigkeit, beim Abfahren der Route, wird erreicht, dass ein Stillstand des Fahrzeugs immer am jeweils letzten Punkt der Route möglich ist.

Die Funktionalität der autonomen Zurückführung zur Straße durch das BackToRoad (BTR)-Modul wird in der Pipeline hinter der Straßen- und Featureerkennung eingehängt. Das BTR-Modul gibt die Informationen weiter an die Worldmap.

3.2.2 Telemetrie-Anwendung

Damit das Verhalten des Fahrzeugs vom Anwender nachvollzogen werden kann wird eine Telemetrie-Anwendung verwendet, in der die vom Fahrzeug aufgenommenen und verarbeiteten Bilder über einen Livestream angezeigt werden. Das Fahrzeug überträgt sämtliche Informationen an einen außen stehenden Computer auf dem die Telemetrie-Anwendung läuft. Die Parameter des Systems auf dem Fahrzeug können während der Fahrt über die Telemetrie-Anwendung gelesen und angepasst werden. Diese Anwendung ist essentiell für das Testen und Optimieren der Funktionalität, sowie der Nachvollziehbarkeit der Entscheidungsfindung des Systems. Die Telemetrie-Anwendung wurde ursprünglich von Nils Schönherr für eine „Kamera-basierte Minimalautonomie“ entwickelt [6, vgl. S. 18].

4 Autonome Zurückführung zur Straße

In diesem Kapitel wird beschrieben, wie die Straße erkannt und verifiziert wird. Außerdem geht es um die Positionierung auf der rechten Spur.

4.1 Erkennungsmerkmale Straße

Für ein autonomes Zurückkehren zur Straße ist es wichtig, diese wiederzuerkennen, sobald sie sich im Sichtfeld befindet. Eine vollständige Straße ohne Sondersituation besteht aus drei Linien: Der linken Außenlinie (LA), der rechten Außenlinie (RA), sowie der Mittellinie (M). Ein Szenario, welches ohne zusätzliche Elemente auskommt, darf laut Carolo-Basic-Cup-Regelwerk nur streckenweise auftreten und ist somit in diesem Kontext nicht realistisch. Die Parklücken sind auf der linken Straßenseite senkrecht und auf der rechten Seite parallel zur Fahrbahn angeordnet. Außerdem gibt es Kreuzungen, vor denen jeweils die M unterbrochen und danach fortgesetzt wird. Es können Stopplinien vor dem Beginn einer Kreuzung auftreten, welche von der RA zur M gehen und quer zur Fahrtrichtung angeordnet sind.

Durchgezogene Linien in Fahrtrichtung können die LA und RA, aber auch die Randlinien der Parklücken sein. Es ist möglich, dass $100\,mm$ neben der Straße weitere Außenlinien einer anderen Fahrbahn im Sichtfeld sind.

Das deutlichste Element zum Erkennen einer Straße in dem Szenario ist ein weißes Mittellinien-Segment (MS). Es ist mit 18 - $20\,mm \times 200\,mm$ genau definiert. Der Abstand zwischen zwei MS ist zusätzlich mit $200\,mm$ festgelegt. Eine M besteht dabei aus mehreren MS. Ein MS bildet mit den beiden Außenlinien LA und RA die parallel zu dem MS sind die Basis auf dem die Straßenerkennung basiert.

Die Abbildung 4.1 zeigt eine mögliche Parkzone, die nach einer Startlinie beginnt.

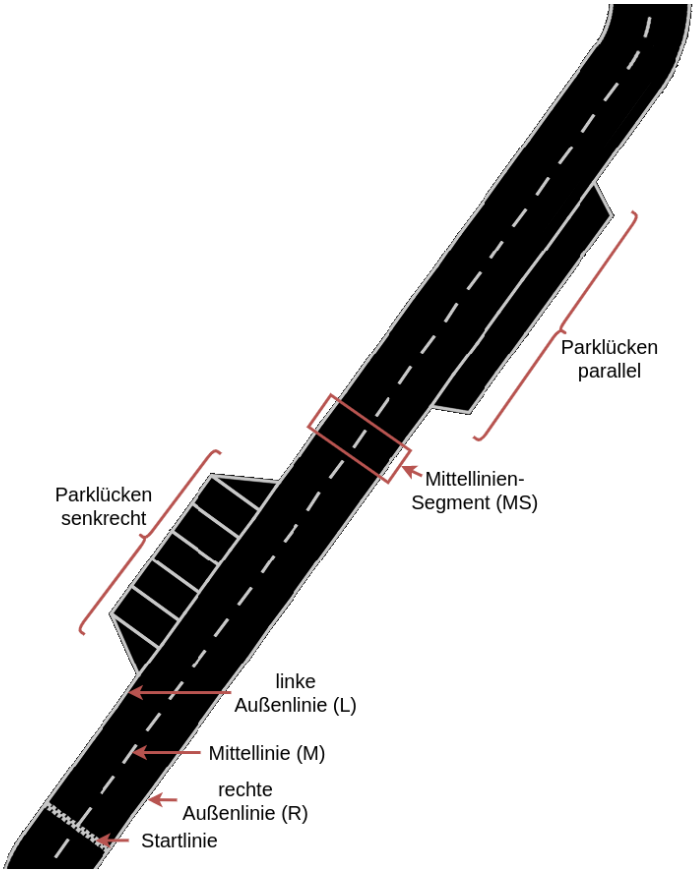

Abbildung 4.1: Parkzone

4.2 Aktivierung des BackToRoad-Moduls

Es wird unterschieden zwischen dem Basissystem und dem BackToRoad-Modul (BTR-Modul). Das Basissystem beinhaltet die Straßenerkennung und Routen-Planung, mit dem das Fahrzeug im autonomen Normalfall seine Strecken fährt. Dazu gehört ebenfalls die Software, die in Kapitel 3.2.1 „Struktur der Software für das autonome Fahren“ beschrieben wurde. Für die autonome Rückführung, die Straßenerkennung zum erneuten Finden der Straße und dem Positionieren auf der rechten Fahrbahn ist das BTR-Modul zuständig. Es arbeitet dabei nicht auf den Daten der Worldmap, jedoch wird die geplante Route von dem BTR-Modul in die Worldmap geschrieben. Das Basissystem nutzt dann die Route aus der Worldmap und fährt diese ab.

Das Starten des BTR-Moduls wird durch die Fernsteuerung ausgelöst. Da keine weitere Funktionalität per Fernsteuerung umgesetzt werden darf (siehe Abschnitt 2.2), entscheidet ein Software-Modul, ob das Basissystem oder das BackToRoad-Modul nach dem Einsatz der Fernsteuerung aktiviert wird. Dafür wird überprüft, ob das Fahrzeug sich in der Zeit während die Fernsteuerung aktiv war, bewegt hat. Hat sich das Fahrzeug nach dem es die vorgeschriebene Sekunde still Stand bewegt, wird das Basissystem aktiviert, da davon ausgegangen wird, dass das Fahrzeug manuell positioniert wurde. Hat sich das Fahrzeug nach dem Stillstand nicht bewegt, wird das BTR-Modul gestartet.

4.3 Softwarekonzept für das Zurückfahren zur Straße

Es wird angenommen, dass das Fahrzeug im autonomen Normalbetrieb einen Fehler macht, wodurch es die Straße verlässt. Im Folgenden werden die Schritte beschrieben, die nötig sind, damit das Fahrzeug nach dem manuellen aktivieren des BTR-Moduls zurück zur Fahrbahn fährt und sich neu auf der Straße positioniert.

Zunächst fährt das Fahrzeug die gleiche Strecke, die es vorwärts gefahren ist, rückwärts zurück (siehe Kapitel 4.3.1). Mittels Straßenerkennung wird der Straßenverlauf erkannt, sobald sich das Fahrzeug wieder auf der Straße befindet. Daraufhin positioniert es sich in Fahrtrichtung auf der rechten Fahrspur, bevor es die Kontrolle an das Basissystem wieder abgibt.

4.3.1 Zurückführung zur Straße

In diesem Unterkapitel wird beschrieben, was notwendig ist, damit die Strecke, die bereits gefahren wurde, rückwärts abgefahren werden kann.

Die Prämisse für den Einsatz des BTR-Moduls ist, dass das Fahrzeug autonom der Straße folgt und diese dann außerhalb des Regelfalls verlässt. Sollte das Fahrzeug zu diesem Zeitpunkt noch nicht gefahren sein und das BTR-Modul wird aktiviert, bleibt es an seiner Position stehen, da keine gefahrene Strecke gespeichert wurde. Bei der gespeicherten Strecke handelt es sich nicht um Informationen aus der Worldmap. Während des Betriebs des Basissystems ist das BTR-Modul nicht deaktiviert. Es läuft ein Datenlogger mit, der für das Abspeichern der gefahrenen Strecke in einer Liste zuständig ist. Wenn die Entfernung zwischen der aktuellen und zuletzt gespeicherten Position größer als $200\,mm$

ist, was der Größe eines MS entspricht, fügt der Datenlogger die aktuelle Position der Liste hinzu. Um den Prozessor nicht unnötig zu belasten, läuft die Bildverarbeitung des BTR-Moduls im Normalbetrieb nicht mit.

Für das Verlassen der Straße außerhalb des Regelfalls werden zwei verschiedene Fälle untersucht.

Beschreibung und Untersuchung des ersten Falls:

Das Fahrzeug verlässt mit kontrollierter Geschwindigkeit, sowie sensorisch korrekt erfasster Position und Ausrichtung, also ohne Rutschen und Durchdrehen der Reifen, die Fahrbahn.

Es kann in diesem Fall davon ausgegangen werden, dass das rückwärts Zurückfahren der zuvor gefahrenen Strecke das Fahrzeug wieder auf die Straße führt. Um der Strecke rückwärts zu folgen, muss die zuvor gefahrene Strecke gespeichert werden. Um eine Vermischung der Speicherstände zu unterbinden, wird die rückwärts gefahrene Strecke nicht gespeichert. Die vom BTR-Modul rückwärts gefahrene Strecke soll kein zweites Mal abgefahren werden, was eine zusätzliche Haltung dieser Daten obsolet macht. Wenn das BTR-Modul startet, werden zu der gespeicherten Strecke keine weiteren Wegpunkte hinzugefügt.
Beim Rückwärtsfahren werden alle zuvor abgefahrenen Wegpunkte aus der gespeicherten Strecke entfernt. Nach der Verifizierung positioniert sich das Fahrzeug dann auf der rechten Fahrspur. Sobald die Positionierung abgeschlossen ist, wird das sogenannte „BTR-Ende“ erreicht und die Kontrolle wird wieder an das Basissystem übergeben.

In Abbildung 4.2 ist die gefahrene Strecke bei einem Fehlerfall zu sehen. Das Fahrzeug fährt vorwärts der grünen Route entlang, macht bei der Fehlstelle an der RA einen Fehler und fährt von der Straße. Bei dem Punkt „BTR-Start“ wurde das BTR-Modul aktiviert. Es übernimmt die Kontrolle des Fahrzeugs und fährt rückwärts die Koordinaten auf der schwarz gepunkteten Route an, die zuvor abgespeichert wurden. An dem Punkt „BTR-Straße verifiziert“ ändert sich die Fahrtrichtung und das Fahrzeug positioniert sich auf der rechten Fahrbahn (schwarz durchgehende Route). An dem Punkt „BTR-Ende“ übergibt das Fahrzeug die Steuerung wieder an das Basissystem.

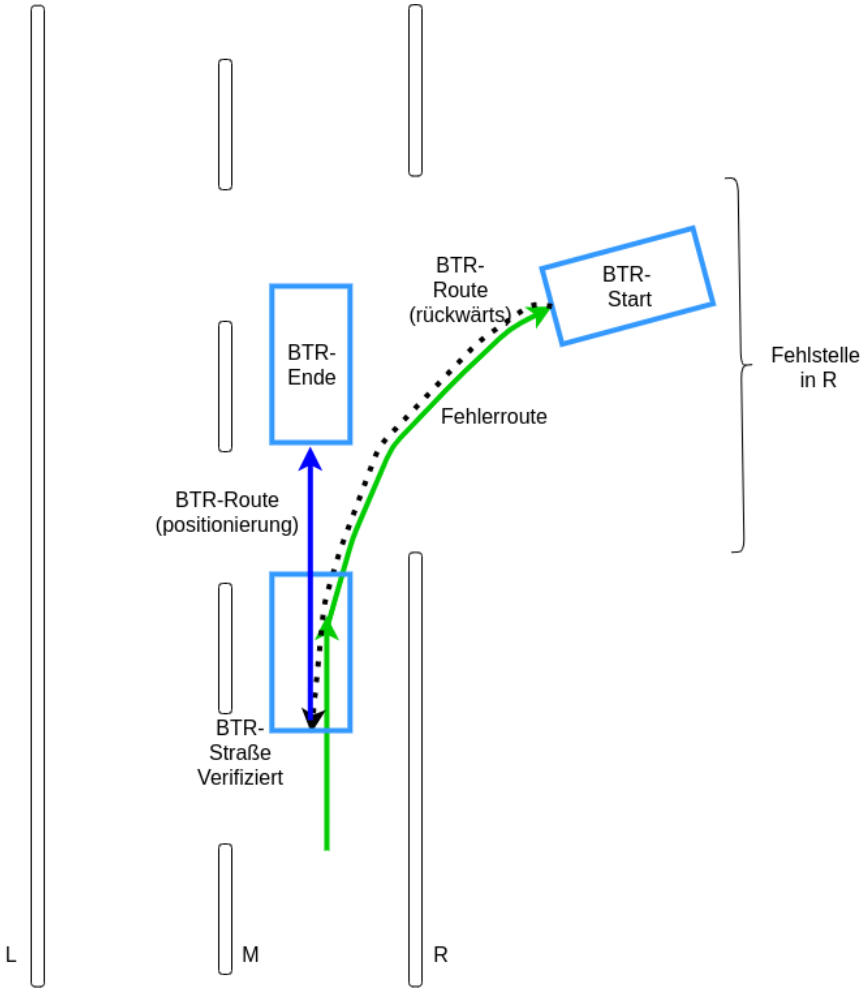

Abbildung 4.2: BacktoRoad-Modul

Beschreibung und Untersuchung des zweiten Falls:

Das Fahrzeug verlässt mit unkontrollierter Geschwindigkeit, sowie fehlerhaft erfasster Position und Ausrichtung durch zum Beispiel Rutschen und oder Durchdrehen der Reifen, die Fahrbahn.

Es kann in diesem Fall nicht davon ausgegangen werden, dass durch das rückwärts Abfahren der zuvor gefahrenen Strecke das Fahrzeug wieder auf die Straße fährt. Wenn die Abweichung der fehlerhaft erfassten Daten nur klein ist, also das Fahrzeug nur wenig unter- oder übersteuert hat, wird das Fahrzeug die Straße trotzdem erreichen. Eine stär-

kere Abweichung führt dazu, dass das Fahrzeug nicht oder nur teilweise auf die Fahrbahn zurückgeführt wird.

Wenn das Fahrzeug beispielsweise in einer Linkskurve stark nach rechts außen rutscht und dabei die Fahrbahn verlässt, führt dies dazu, dass es rückwärts parallel neben der Fahrbahn zurückfährt.

Wenn das Fahrzeug eine Strecke von $2\,m$ neben oder auf der Fahrbahn gefahren ist und keine Verifizierung stattgefunden hat, wird angenommen, dass sich das Fahrzeug neben der Straße befindet, auf die es zurückfahren soll.

Daraus leiten sich zwei Szenarien bezüglich des zweiten Falls ab:

Erstens: Die gespeicherte Strecke wird beim rückwärts Abfahren um den Versatz zwischen Fahrzeug und Straße korrigiert. Dadurch soll erreicht werden, dass das Fahrzeug beim Rückwärtsfahren auf die gewünschte Strecke fährt.

Zweitens: Das Fahrzeug positioniert sich direkt auf der rechten Fahrspur, sobald eine Strecke von $2\,m$ neben der Fahrbahn zurückgefahren wurde.

Die Korrektur der gespeicherten Strecke kostet mehr Zeit, da mehr Strecke rückwärts abgefahren wird als bei der direkten Positionierung.

Die gespeicherte Strecke wird in jedem Fehlerfall rückwärts abgefahren, bei ungenauen Messdaten wird keine Korrektur der gespeicherten Strecke durchgeführt: Die entstehenden Probleme der falschen Positionierung werden mittels Straßenerkennung abgefangen.

4.3.2 Straßenerkennung vom BackToRoad-Modul

Das Fahrzeug befindet sich im Normalfall nicht auf der Straße, wenn die Erkennung der Straße gestartet wird. Die Straßenerkennung soll unabhängig davon funktionieren, ob das Fahrzeug sich auf oder neben der Straße befindet. Daraus ergibt sich die Anforderung, dass die Straßenerkennung des BTR-Moduls unabhängig von alten Zuständen funktioniert. Die Straßenerkennung ist die größte Aufgabe des BTR-Moduls. Wenn die Straßenerkennung eine Straße fehlerhaft erkennt und verifiziert, führt dies dazu, dass das Basissystem nicht auf der Straße startet und keine Weiterfahrt stattfinden kann, weil keine Straße durch das Basissystem erkannt wird.

Entfernung zum MS (in mm)	Größe unbearbeitete Aufnahme (in mm)	Größe transformiertes Bild (in mm)
200	12,2	12,2
600	4,2	13,3
1000	2,1	14,2

Tabelle 4.1: Mittellinien-Segmentgrößen im Verhältnis zur Entfernung

Es ergibt sich daher die Notwendigkeit, dass die Straßenerkennung des BTR-Moduls mit einer hohen Genauigkeit arbeitet.

Auswahl des Bildes für die Straßenerkennung

Damit die Straßenerkennung eine hohe Genauigkeit erreicht, ist es wichtig, eine gute Bild-Basis auszuwählen, auf der die Straßenerkennung basiert.

Alle eingelesenen Bilder werden vom Basissystem an das BTR-Modul übergeben. Die folgenden Versionen des gleichen Eingangsbildes stehen dabei für die Bildverarbeitung des BTR-Moduls zur Auswahl.

Eine der Bildversionen ist ein nicht transformiertes Bild. In einem nicht transformierten Bild ist ein Objekt fester Größe im vorderen Bereich des Bildes deutlich größer dargestellt als im Hinteren. Wie aus Tabelle 4.1 von Mittellinien-Segmentgrößen im Verhältnis zur Entfernung zu entnehmen ist, ist ein Objekt in 1000 mm Entfernung circa sechsmal kleiner, als wenn es direkt vor dem Fahrzeug positioniert wäre. Beim Vergleichen von zwei Objekten in unterschiedlichen Entfernungen zum Fahrzeug ist jeweils eine Umrechnung der Entfernungen nötig. Deswegen wird beispielsweise das mit dem Weitwinkelobjektiv aufgenommene Bild (siehe Abbildung 4.3), welches nur auf den für die Straßenerkennung relevanten Bereich zurechtgeschnitten wurde, nicht verwendet. Der genannte „relevante Bereich“ beinhaltet alle Straßenmarkierungen die im Sichtfeld zu sehen sind und einen kleinen Bereich über dem Horizont, der die Erkennung von Hindernissen ermöglicht.

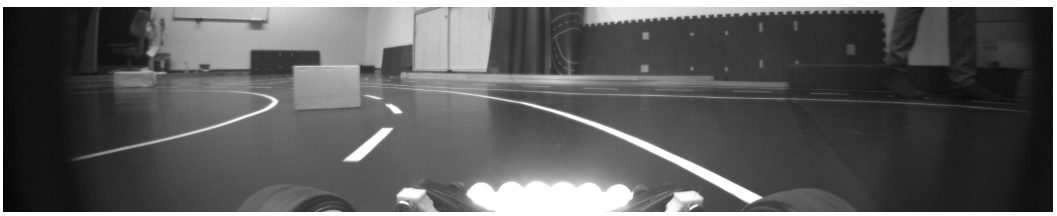

Abbildung 4.3: Unbearbeitete Aufnahme des Bildes

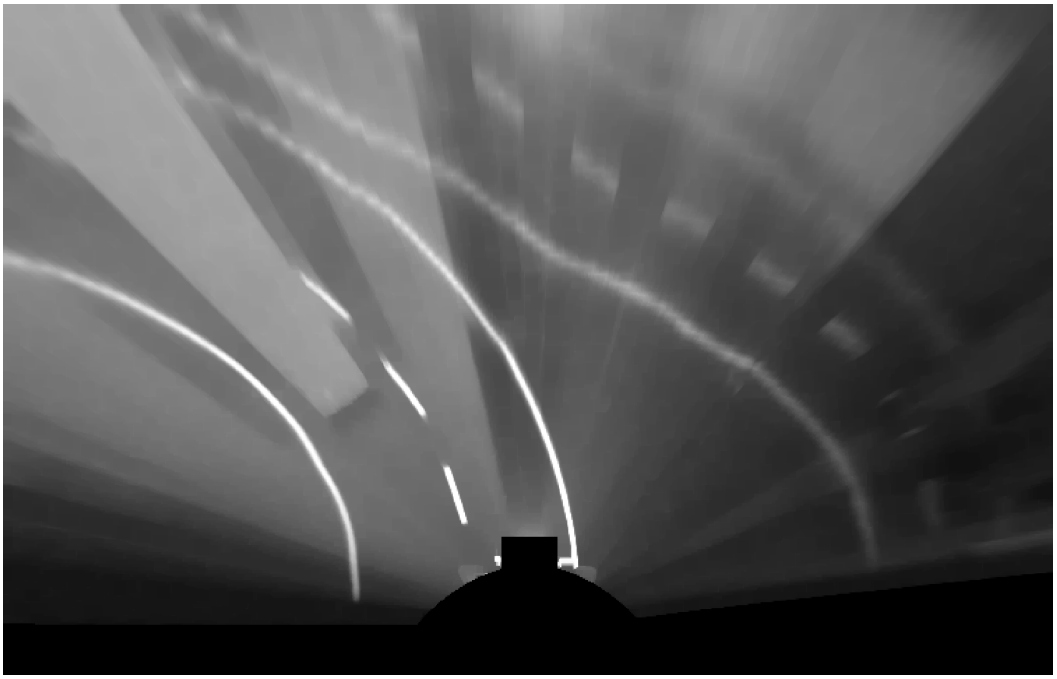

Abbildung 4.4: In Vogelperspektive transformiertes Bild

Die zweite Bildversion ist ein transformiertes Bild (siehe Abbildung 4.4). Das Bild wurde in eine Vogelperspektive transformiert. Dies hat den Vorteil, dass die Straßenbreiten vorne, sowie hinten im Bild jeweils die gleichen Dimensionen aufweisen.

In der dritten Bildversion wurde ein Median-Filter (siehe Abbildung 4.5) angewandt. Durch diesen wird das Bild optisch geglättet und die Straßenmarkierungen aus dem Bild herausgefiltert. Hindernisse, welche groß im Bild dargestellt sind, bleiben weiterhin gut erkennbar. Es ist jedoch nicht zielführend die Straßenerkennung auf Grundlage von Bildern, auf denen ein Median-Filter angewendet wurde, zu entwickeln, da Straßenmarkierungen nicht ausreichend dargestellt werden können.

Eine vierte Bildversion ergibt sich aus der Kombination der zuvor beschriebenen Bildversionen: Durch Subtraktion des transformierten Bildes mit dem Median-Bild entsteht ein subtrahiertes Bild (siehe Abbildung 4.6) auf dem die Straßenmarkierungen deutlich zu sehen sind. Die Hindernisse, welche im Median-Bild vordergründig dargestellt sind, werden durch die Subtraktion größtenteils entfernt. Auf dem subtrahierten Bild sind bis auf kleine Artefakte nur Straßenmarkierungen zu sehen, welche in verschiedenen Grautönen dargestellt werden. In Abbildung 4.6 sind in der Mitte von der RA helle und an den Enden von der LA dunklere Grautöne zu sehen.

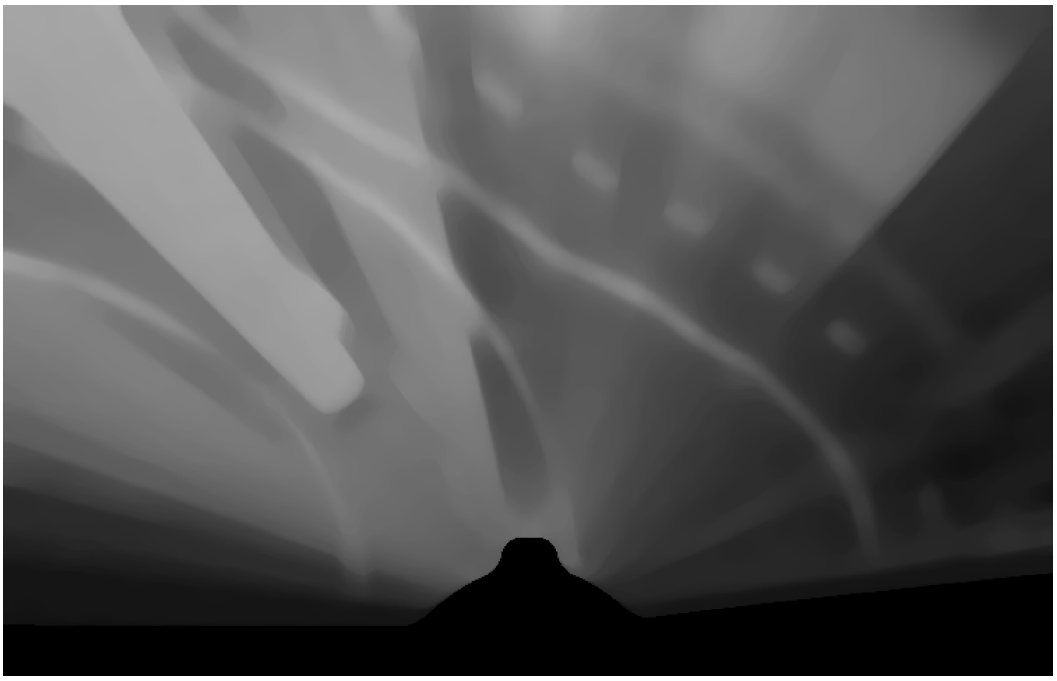

Abbildung 4.5: Glättung des Bildes mit einem Medianfilter

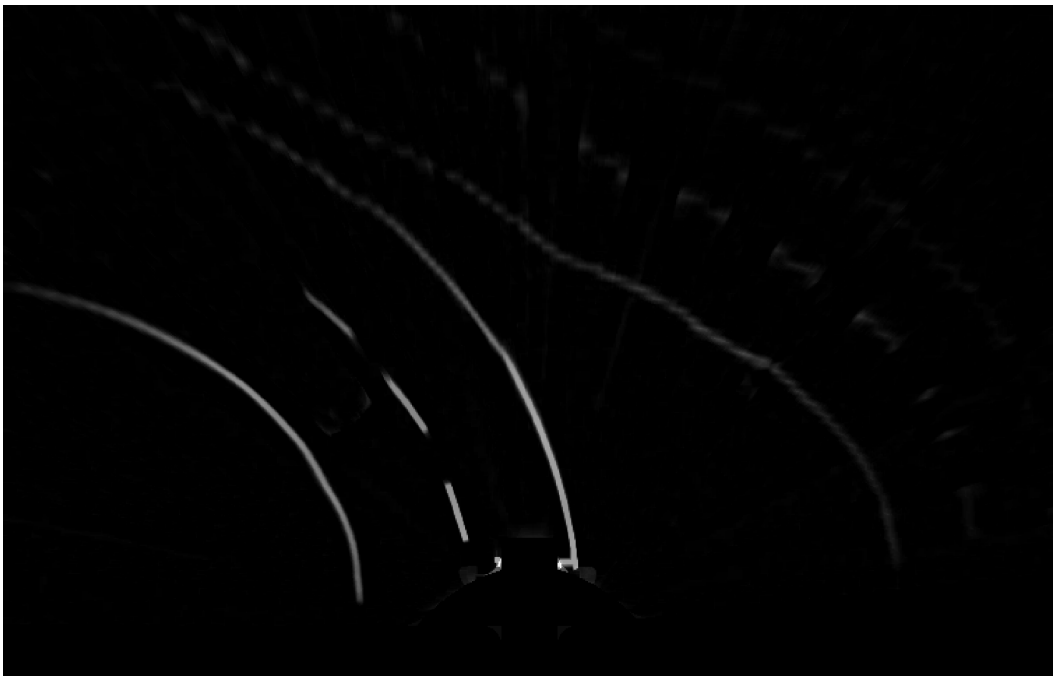

Abbildung 4.6: Subtraktion aus transformiertem und geglättetem Bild

Abbildung 4.7: Binärisiertes Bild

In Bildversion fünf wurde ein weiterer Schritt umgesetzt: In Abbildung 4.7 ist das Ergebnis zu sehen, wenn alle Grautöne über einem bestimmten Schwellenwert weiß, sowie unter diesem zu schwarz binärisiert werden. Dadurch ist es möglich, eine binäre Abfrage im Bild zu verwenden. Die Erkennung von Hindernissen ist für die Straßenerkennung nicht relevant, diese spielen nur eine Rolle beim Positionieren auf der rechten Fahrspur.

Aus Sicht des Entwicklers ist es für die Straßenerkennung, am sinnvollsten nur mit Bildern zu arbeiten, die ausschließlich auf binären Werten basieren. Auf dem binären Bild ist direkt ersichtlich, ob der Wert eines Pixel als weiß oder schwarz ausgewertet wird. Dies ist im Vergleich zu einem Grauwert nicht der Fall und ermöglicht somit eine einfachere Implementation. Aus diesem Grund wird für die Straßenerkennung die fünfte Bildversion verwendet.

Erkennung der Straße mit Hilfe von Mittellinien-Segmenten

Die Erkennung der Mittellinien basiert auf der Annahme, dass MS als eindeutige Konturen stets erkennbar sind. Um die Straße zu erkennen, werden einzelne MS erkannt und

verknüpft. Die Erkennung soll dabei jedoch unabhängig von der jeweiligen Ausrichtung zur Straße funktionieren.

Die Straßenerkennung arbeitet mit den Konturen die in einem Bild zu sehen sind. Für jedes zusammenhängende weiße Objekt in einem Bild wird eine Kontur angelegt, die in einer Liste aller Konturen gespeichert wird. Eine Kontur beschreibt ein Objekt mit Hilfe einer Kurve, die mit Punkten entlang der Außenkante des Objektes beschrieben wird. Beim Binärisieren des Bildes entstehen jeweils im grauen Bereich um den Schwellenwert weiße Konturen, zwischen denen Lücken erkennbar sind. Ein Beispiel für die Lücken ist in Abbildung A.3 zu sehen. Um die Lücken zu überbrücken, wird von jeder Kontur der Mittelpunkt bestimmt und diese werden jeweils auf den Abstand untereinander verglichen. Ist der Abstand der Mittelpunkte kleiner als $30\,mm$, werden die Konturen zu einer zusammengefügt. Dadurch entstehen im Randbereich des Schwellenwertes mehr Konturen der gewünschten Größe. Die zusammengefügten Konturen sind in Abbildung A.4 rot dargestellt. Konturen aus wenigen Pixeln helfen bei der Erkennung nicht weiter und werden im nächsten Schritt aussortiert.

Um Konturen mit Form einer Mittellinienmarkierung (siehe Definition in der Abbildung 4.8) zu erkennen, werden die einzelnen Konturen untersucht. Alle Konturen, welche nicht der Form einer Mittellinienmarkierung entsprechen, werden herausgefiltert. Dies ist der Fall, wenn die Bogenlänge oder der Umfang zu klein oder zu groß sind. In Abbildung A.5 sind die Konturen, die nicht herausgefiltert wurden, blau markiert. Die Bogenlänge ist die Summe der Längen zwischen den einzelnen Punkten, die eine Kontur bilden.

Zur Verifikation einer Mittellinienmarkierung werden die passenden Außenlinien gesucht. Dazu werden parallel zur Mittellinienmarkierung zwischen einer minimalen und maximalen Entfernung jeweils die Punkte Left und Right gesucht (siehe Abbildung 4.8), die auf den Außenlinien liegen.

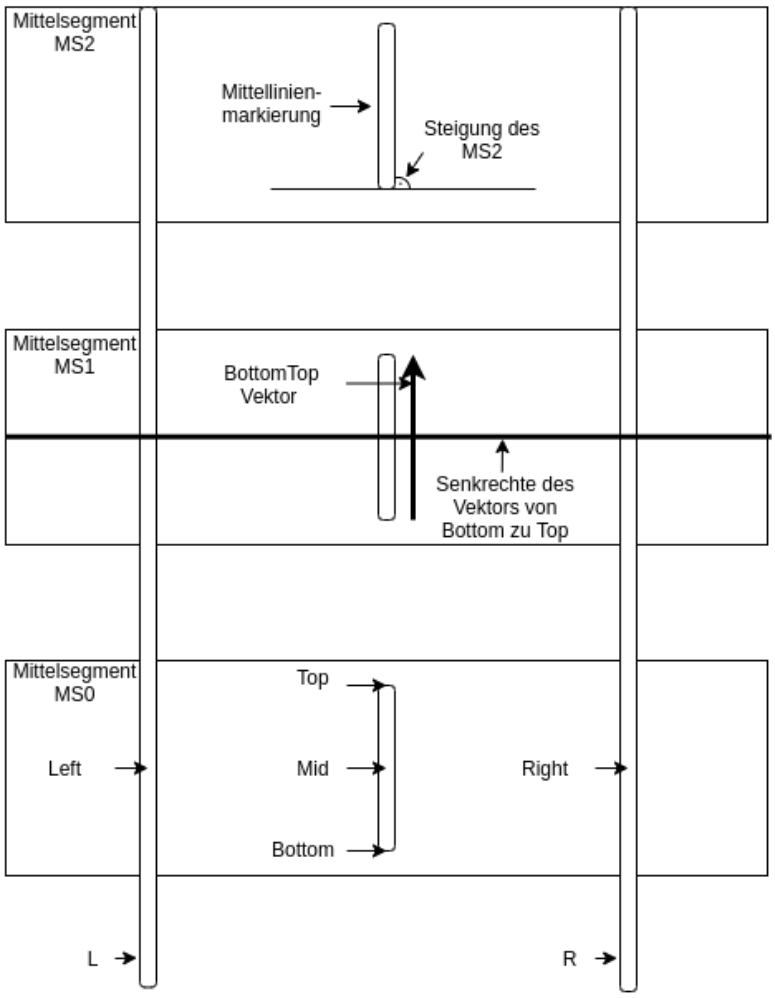

Abbildung 4.8: Mittellinien-Segment

Die Abstände ergeben sich aus der kleinsten Breite der Fahrspur mit 350 mm und der größten Breite mit 450 mm. Mit einer Abweichung von bis zu 30% wird der minimale Abstand von 245 mm und der maximale Abstand mit 585 mm erhalten. Die Abweichung ist notwendig, da Aufnahmen mit Bewegungsunschärfe während der Fahrt zu Rauschen in den Bildern führen kann. Außerdem wird die Kalibrierung für die Transformation in die Vogelperspektive mit einem Bild gemacht, während das Fahrzeug still steht. Im direkten Vergleich ist zu sehen, dass in Abbildung 4.9 weniger Bewegungsunschärfe vorhanden ist als in Abbildung 4.10. In Abbildung 4.10 sind mit sehr hoher Bewegungsunschärfe jeweils MS, mit der Länge l, 0,7l und 1,3l zu finden. Alle drei MS sollen akzeptiert werden, deshalb wird eine Unschärfe von 30% im BTR-Modul verwendet.

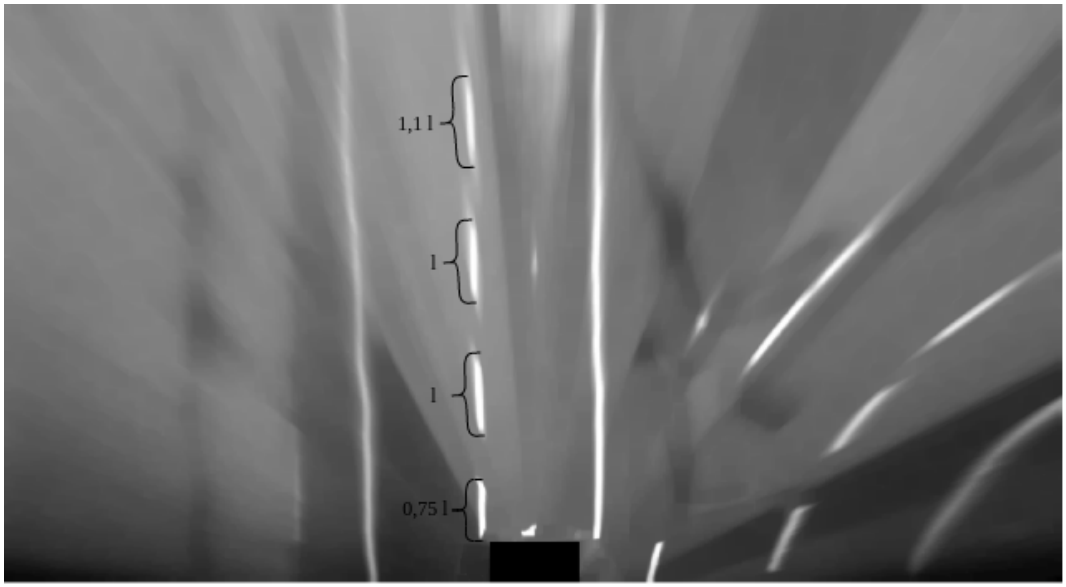

Abbildung 4.9: Transformiertes Bild während des Fahrzeugstillstands.

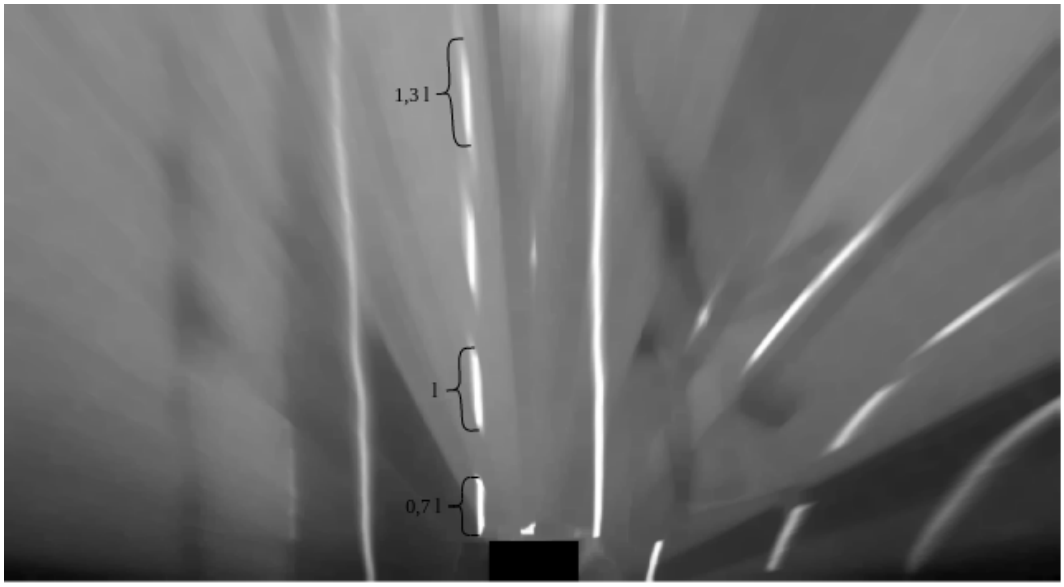

Abbildung 4.10: Transformiertes Bild während das Fahrzeug in Bewegung ist.

Für das Suchen der Punkte Left and Right werden die Außenpunkte Top und Bottom (siehe Abbildung 4.8) der Mittellinienmarkierung ermittelt. Entlang der Senkrechten des Vektors von Bottom zu Top wird nach den Punkten Left und Right gesucht.

Die Senkrechte wird für die beiden Außenlinien, jeweils einmal durch Mid, Bottom und Top angelegt. Entlang der Senkrechten wird im Abstand zwischen 245 mm und 585 mm nach den Außenlinien LA und RA gesucht.

Der Bereich, in dem auf der Senkrechten gesucht wird, ist grün in der Abbildung A.6 dargestellt.

Sobald entlang der Senkrechten ein weißer Pixel gefunden wurde, wird angenommen, dass dieser zu der Außenlinie gehört, da er sich im richtigen Abstand zur Mittellinienmarkierung befindet. Der gefundene Punkt wird auf die Senkrechte vom Mittelpunkt gerechnet und als Außenlinie zu einem MS gespeichert.

In Abbildung 4.8 ist ein MS zu sehen, welches sich aus fünf Punkten (Bottom, Mid, Top, Left und Right), der Steigung und der Nummer von der Straße dem dieses Segment zugeordnet ist, zusammensetzt. Die Punkte Mid, Top und Bottom, beschreiben den Mittelpunkt, sowie den obersten und untersten Punkt des MS. Left und Right sind die dazugehörigen Punkte auf LA und RA. Die Steigung wird beschrieben durch:

$$slope = arctan2((bottom - top).y, (bottom - top).x) \cdot 180/\pi;$$

Die Nummer der Straße wird beim Zusammenfügen der MS zu einer Straße zugeordnet, um eine Verbindung der MS zu schaffen.

Wenn beispielsweise zur linken Seite entlang der Senkrechten, die durch den Punkt Bottom führt, ein weißer Pixel gefunden wurde, wird der Offset von Bottom zu Mid auf den gefundenen Pixel gerechnet und als Left abgespeichert. Damit wird erreicht, dass Left and Right immer auf der Senkrechten durch den Punkt Mid liegen. Sobald ein passender Punkt für Left and Right gefunden wurde, wird die Suche beendet, da für die weitere Suche nur ein Punkt von LA und RA für jedes MS benötigt wird. In Codebeispiel 4.1 ist der Pseudocode für das Suchen entlang der Senkrechten vom Punkt Bottom zur linken Seite.

```
for (unsigned long i = 0; i < MS.size(); i++) {
  // perpendicular counter clockwise
  cv::Point2f perpVec;
  getPerpendicularCounterClockwise(perpVec, MS[i].top, MS[i].bottom);
  for (int j = -disOuterLineMin; j > -disOuterLineMax; --j) {
    // search bottom left
    cv::Point2f p;
    p = j * perpVec;
    p = p + MS[i].bottom;
    if (p.x > 0 && p.x < binaryPic.cols - 1 && p.y > 0 &&
        p.y < binaryPic.rows - 1) {
      cv::Scalar intensity = binaryPic.at<uchar>(p.y,p.x)
      // if pixelValue != black
      if (intensity[0] != 0) {
        MS[i].leftLine = p + (MS[i].mid - MS[i].bottom);
        break;
      }
    }
  }
}
```

Codebeispiel 4.1: Suche der linken Außenlinie von einem Mittellinien-Segment

Sollte entweder LA oder RA nicht gefunden werden, wird die Kontur aus der Liste der MS entfernt. Es bleiben nur noch Elemente übrig, zu denen beide Außenlinien zugeordnet werden können. In Abbildung A.7 haben alle rot markierten MS beide Außenlinien. Die blau markierten werden aussortiert, da sie maximal eine der beiden Außenlinien haben.

Um aus der Liste der MS Straßen zu bilden, werden die einzelnen MS miteinander verglichen und den jeweiligen Straßen zugeordnet. Dabei wird jedes MS mit allen anderen verglichen. Eine Straße setzt sich somit aus einer Liste von MS zusammen. Es werden nur MS der gleichen Straße hinzugefügt, wenn folgende Bedingungen zutreffen:

1) Die Distanz der Mittelpunkte zwischen MS A und MS B ist größer 140 mm und kleiner 260 mm. Die Abstandsgrenzen bilden sich aus dem Abstand zwischen zwei Mittellinienmarkierungen von 200 mm plus 30% Abweichung.

2) Die Steigung des MS A und des MS B darf sich nicht um mehr als 22 $^\circ$ unterscheiden. Der Winkel ergibt sich aus dem minimalen inneren Radius von 1000 mm, den eine Kurve haben darf und aus der kleinsten Breite der Fahrspur von 350 mm. Der Radius r der Mittelspur beträgt minimal 1350 mm. Zwischen Beginn eines MS und

Beginn des nächsten MS liegen 400 mm. Die maximale Winkeländerung wird für diese Strecke berechnet. Die Formel für den Umfang eines Kreises lautet:

$$U = 2 \cdot \pi \cdot r$$

Die Winkeländerung $\Delta\phi$ auf einer Strecke s berechnet sich mit:

$$\Delta\phi = \frac{360 \cdot s}{U}$$

Durch Zusammenführen der beiden Formeln ergibt sich für die Winkeländerung auf einer Strecke:

$$\Delta\phi = \frac{360 \cdot s}{2 \cdot \pi \cdot r}$$

Durch Einsetzen vom Radius $r = 1{,}35\,m$ und der Strecke $s = 0{,}4\,m$ ergibt sich eine Winkeländerung von $\Delta\phi = 16{,}9765\,^\circ$ pro $0{,}4\,m$. Mit einer Abweichung von 30% ergibt sich ein maximaler Unterschied zwischen den beiden Steigungen von 22,06945 °. Dies ergibt gerundet 22,07 °.

3) Der Winkel zwischen dem Vektor A.Bot zu A.Top und dem Punkt B.Mid darf nicht größer sein als 16,55 °. Der Abstand zwischen den Punkten A.Top und B.Mid beträgt $0{,}3\,m$ und bildet in diesem Fall die Strecke s. Der kleinste Radius r einer Kurve beträgt weiterhin $1{,}35\,m$. Setzt man diese Werte in die Formel für die Winkeländerung ein, ergibt sich eine Winkeländerung von $\Delta\phi = 12{,}7324\,^\circ$ pro $0{,}3\,m$. Durch die Abweichung von 30% ergibt sich ein maximaler Winkel von dem Vektor A.Bot zu A.Top und dem Punkt B.Mid von 16,5521 ° und gerundet von 16,55 °.

Folgende Beispiele dienen zur Verdeutlichung der Bedingungen.

Erstes Bsp.: Vor und nach einer Fehlstelle werden die MS jeweils nicht miteinander verknüpft, da der Abstand zwischen den beiden MS größer als 260 mm ist.

Zweites Bsp.: Eine Haltelinie an einer Kreuzung wird daher auch nicht zu einer falschen Straße zugeordnet werden. Dies liegt daran, dass der Abstand und Winkel zum vorherigen Element zwar stimmt, aber die Steigung der einzelnen Elemente stark abweicht, da diese fast einen 90 ° Winkel aufweisen.

Nachdem jedes MS mit den anderen MS verglichen wurden, ist jedes MS einer Straße zugeordnet. Im nächsten Schritt werden die Straßen, ohne die Entfernung zur nächsten Straße zu berücksichtigen, miteinander verknüpft. Dadurch ist es möglich, Fehlstellen und Kreuzungen zu überbrücken. Die Anzahl der Straßen wird minimiert, wodurch die

Auswahl der Straße, auf die das Fahrzeug zurückfahren soll, vereinfacht wird. Im nächsten Schritt werden alle Straßen untereinander verglichen, dabei werden immer zwei MS mit den dichtesten Mittelpunkten der beiden zu vergleichenden Straßen bestimmt. Zwischen diesen wird, wie beim ersten beschriebenen „Zusammenbau“ der Straßen, jeweils die Steigung und der Winkel abgeglichen.

Straßen, die aus nur einem MS bestehen, werden nicht als Straße akzeptiert und entfernt. Ein einzelnes MS ist nicht aussagekräftig, weshalb das Fehlerpotential einer falschen Straße zu folgen groß ist. Damit die Reihenfolge der zur selben Straße gehörenden MS stimmt, werden diese neu sortiert. Für die Sortierung wird als erstes ein Endstück-MS, welches den Anfang oder das Ende der Straße bildet, ermittelt (siehe Codebeispiel 4.2).

```
1  startMS = road.MS[0];
2  idxNearestMS = 0;
3  for (long i = 0; i < road.MS.size(); ++i) {
4    if (road.MS[i].mid != startMS.mid) {
5      float disOfBM = cv::abs(startMS.top - road.MS[i].mid);
6      float disOfTM = cv::abs(startMS.bottom - road.MS[i].mid);
7      if (disOfTM - disOfBM < 0) {
8        startMS = road.MS[i];
9        idxNearestMS = i;
10     }
11   }
12 }
```

Codebeispiel 4.2: Finden des Endstück-Mittellinien-Segmentes

Auf Grund des Sichtwinkels besteht eine Straße jeweils aus bis zu sieben MS, im Mittel sind es drei. Da es sich um eine sehr geringe Anzahl zu vergleichender Elemente handelt, wird der Sortieralgorithmus Selectionsort verwendet.

Die sortierte MS Liste enthält initial das Endstück-MS. Danach wird iterativ das verbleibende MS mit der geringsten Entfernung zum letzten einsortierten MS eingefügt, bis alle MS sortiert sind. Die Sortierung sorgt für eine Reihenfolge, bei der immer das MS mit dem kleinsten Abstand folgt. Damit das erste Element der Straße das dichtere Ende von der Straße zur Fahrzeugposition ist, werden die Entfernungen vom ersten und letzten MS zur Fahrzeugposition verglichen. Ist das erste MS weiter entfernt, als das Letzte, wird der Vektor invertiert.

Bei der Auswahl der Straße an einer Kreuzungssituation gibt es oftmals die Problematik, dass die zum Fahrzeug senkrechte Straße dichter ist, als die, die hinter der Kreuzung

weiter führt. Die Entfernung der Straße wird dabei von dem nächstem MS (MS0) der jeweiligen Straße gemessen. Die Auswahl der Straße an einer Kreuzungssituation ist in Abbildung 4.11 aufgezeigt.

Eine Drehung des Fahrzeuges um 90° beim Verlassen setzt ein sehr starkes Rutschen auf der Fahrbahn voraus. Deshalb wird angenommen, dass eine Straße die senkrecht zur Fahrtrichtung verläuft, nicht die Gesuchte ist, sondern die Straße mit der Steigung, die ähnlich derer der Fahrtrichtung ist, aber weiter entfernt hinter der Kreuzung beginnt. Die fehlerfreie Auswahl der zu folgenden Straße wird also über einen Vergleich der Steigung erreicht. Um diesen Fall zu berücksichtigen, wird die Entfernung zwischen Straße und Fahrzeugposition auf der x-Achse betrachtet. Die y-Achse verläuft entlang der Fahrtrichtung und die x-Achse senkrecht dazu. Die Straße ist nur zulässig, wenn die Entfernung auf der x-Achse nicht größer als $500\,mm$ ist. Die Entfernung auf der y-Achse spielt dabei keine Rolle, da sonst Straßen, die weiter entfernt sind, aber die richtige Ausrichtung haben, nicht zulässig wären. Dies geschieht folgendermaßen: Es wird die Straße mit der jeweils kleinsten Steigung im Vergleich zur Fahrtrichtung gesucht, die auf der x-Achse nicht weiter entfernt ist als $500\,mm$ zwischen dem Vergleichspunkt (VP) und der Mitte von $MS0.mid$ und $MS0.right$. Die ausgewählte Straße befindet sich dabei stets im Sichtfeld der Kamera. Die Fahrzeugposition liegt hinter dem Sichtfeld, deshalb wird die Straße verlängert und ein VP, der vor dem Fahrzeug liegt, für die Auswahl der Straße und der Verifizierung verwendet.

4.3.3 Verifizierung der Straße

Nachdem eine Straße ausgewählt wurde, wird eine Verifizierung durchgeführt. Die Folge einer erfolgreichen Verifikation ist die Positionierung des Fahrzeugs auf der rechten Fahrspur (RS) der verifizierten Straße. Für die Verifizierung wird überprüft, ob sich das Fahrzeug auf der RS befindet. Wenn dies nicht der Fall ist, wird in einem zweiten Schritt überprüft, ob es sich auf der linken Fahrspur (LS) befindet.

Die ersten beiden Elemente der Straße werden zum Vergleich verwendet, um herauszufinden, ob sich das Fahrzeug auf der RS befindet. Dazu werden jeweils Hilfspunkte zwischen und vor den beiden MS berechnet. Die Hilfspunkte $P1$ und $P2$ liegen jeweils mittig zwischen „MS.mid“ und „MS.right“. Dabei wird für $P1$ das Erste und für $P2$ das zweite MS verwendet. Zwischen den Punkten $P1$ und $P2$ wird eine Gerade angelegt, auf der jeweils im Abstand von einem Viertel der Spurbreite die Distanz zum VP berechnet wird. Wenn

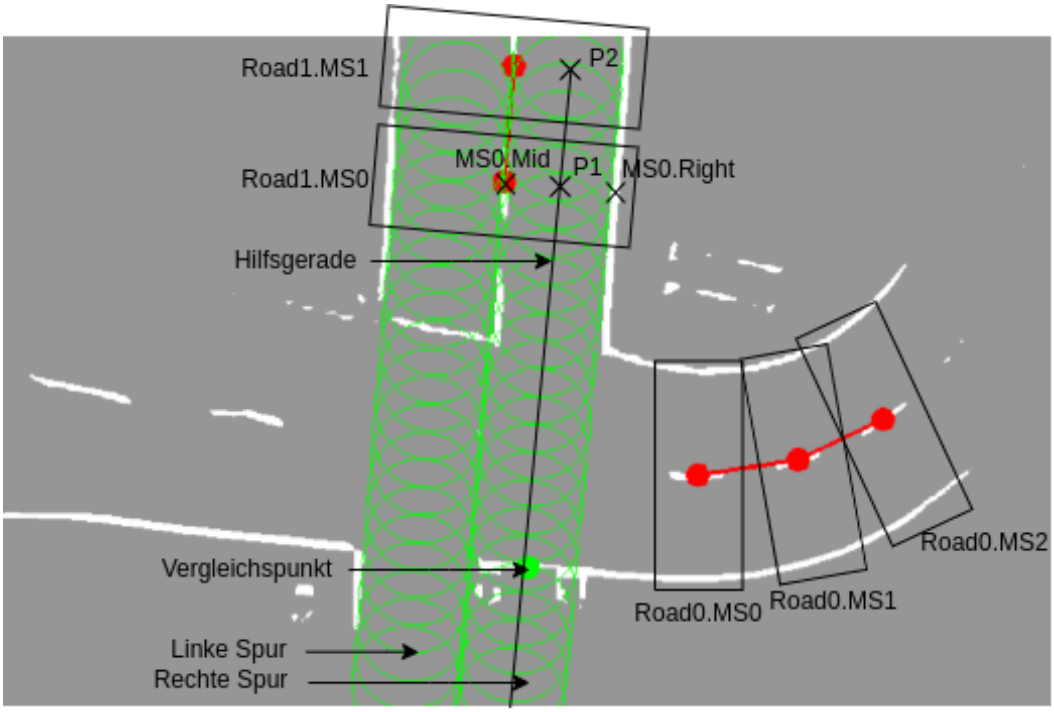

Abbildung 4.11: Auswahl der Straße und Verifizierung

die Distanz kleiner ist als die Hälfte der Spurbreite, dann befindet sich das Fahrzeug auf der RS.

Durch den Abgleich der Abstände zwischen der Hilfsgeraden und dem VP lassen sich jeweils Kreise mit Mittelpunkt auf der Hilfsgeraden einzeichnen, auf denen zu sehen ist, wie die Fahrspur erkannt wird. In der Abbildung 4.11 sind die Punkte P1 und P2 eingezeichnet, mit denen die Hilfsgerade aufgespannt wird. Der Radius wird mit der Hälfte der Spurbreite bestimmt, dies ist der Abstand zwischen der Spurmitte und den Rändern der Fahrspur. Der Abstand auf der Hilfsgeraden, von dem die Distanz jeweils überprüft wird, beträgt ein Viertel der Spurbreite. Dadurch wird erreicht, dass nur ein sehr kleiner Bereich an den Schnittstellen der Kreise nicht überprüft wird und nicht für jeden Punkt auf der Geraden ein Vergleich durchgeführt werden muss. In Abbildung 4.11 ist außerdem ersichtlich, dass der VP in einem der Kreise liegt, welche die RS markieren und sich damit auf der RS befindet.

Mittels zweier Zähler werden Bilder, mit deren Hilfe keine Straße gefunden wurde oder bei denen sich das Fahrzeug nicht auf der Fahrspur befindet, toleriert. Der Zähler „noRoadCout“ zählt die Anzahl, wie viele Bilder hintereinander das Fahrzeug nicht auf der

Straße war und der Zähler „noRightLaneCout“, die Anzahl der Bilder auf der es sich nicht auf der RS befunden hat (siehe Codebeispiel 4.3). Damit die Verifizierung als erfolgreich gilt, muss das Fahrzeug eine Strecke von $500\,mm$ am Stück rückwärts auf der Straße fahren. Dabei darf der „noRoad“ Zähler nicht größer zwei werden, anderenfalls wird die bereits rückwärts auf der Straße gefahrene Strecke zurückgesetzt.

```
roadObj road{};
// build the road
bool foundRoad = DriveBackToRoad::buildRoad(inputPic, outputPic, road);
if (foundRoad) {
    // check on right lane
    float streetwidth =
        disOfPoints(road.MS[1].left, road.MS[1].right);
    // if on right lane: reset counters
    if (DriveBackToRoad::checkOnLane(
            road.MS[0].mid, road.MS[1].mid, road.MS[0].right,
            road.MS[1].right, streetwidth, outputPic)) {
        noRoadCount = 0;
        noRightLaneCount = 0;
    } else {
        noRightLaneCount++;
        if (noRightLaneCount >= 3) {
            lastNonRightLanePos = pos;
        }
    }
    // if not on right lane: check on left lane
    // if on left lane: reset noRoadCount
    if (DriveBackToRoad::checkOnLane(
            road.MS[0].left, road.MS[1].left,
            road.MS[0].mid, road.MS[1].mid, streetwidth, outputPic)) {
        noRoadCount = 0;
    } else {
        noRoadCount++;
        if (noRoadCount >= 3) {
            lastNonRoadPos = pos;
        }
    }
    // if no road found
} else {
    if (noRoadCount <= 3) {
        noRoadCount++;
        noRightLaneCount++;
    } else {
        lastNonRightLanePos = pos;
        lastNonRoadPos = pos;
    }
}
```

Codebeispiel 4.3: Überprüfung, ob sich das Fahrzeug auf der Straße befindet

Positionierung auf der rechten zurück Fahrspur

Nachdem die Straße erkannt und verifiziert wurde, folgt der letzte Schritt. Bei diesem wird das Fahrzeug auf der RS positioniert. Die geplante Route verläuft dabei mittig auf der RS. Dafür wird der Route für jedes MS ein Punkt zwischen „MS.Mid“ und „MS.Right“ hinzugefügt. Der Route wird solange gefolgt, bis das Fahrzeug 300 mm auf der RS gefahren ist und sich damit auf dieser positioniert hat. Wenn sich das Fahrzeug auf der LS befindet, während die Verifizierung abgeschlossen wird, muss das Fahrzeug auf die RS wechseln. Bis alle 4 Reifen auf der RS sind, muss das Fahrzeug mindestens den Abstand zwischen Hinterachse zur Vorderachse (300 mm) auf der RS fahren. Die Geschwindigkeit des Fahrzeugs während des Verifizierens und Positionierens beträgt 1 m/s. Nachdem das Fahrzeug positioniert ist übernimmt das Basissystem die Kontrolle.

Die Länge der rückwärts zu fahrenden Strecke für die Verifizierung und die maximale Geschwindigkeit, die das BTR-Modul fährt, wurden durch einen Test auf der Teststrecke (Live-Test) ermittelt.

5 Evaluation

In diesem Kapitel wird die Testumgebung beschrieben und die Funktionalität des BTR-Moduls überprüft. Dafür werden die am häufigsten auftretenden Fehlersituationen (Hindernis mit Fehlstellen und Kreuzung) untersucht. Außerdem wird gezielt der zweite Fall durch simuliertes Unter- und Übersteuern getestet:

Das Fahrzeug verlässt mit unkontrollierter Geschwindigkeit, sowie fehlerhaft erfasster Position und Ausrichtung durch zum Beispiel Rutschen und oder Durchdrehen der Reifen, die Fahrbahn.

5.1 Testumgebung

Um die Funktionalität des BTR-Moduls zu testen, wird eine Testumgebung benötigt. In Kapitel 2 sind die Abmessungen der Straßenmarkierungen beschrieben, anhand derer eine ca. $30\,m$ lange Teststrecke angelegt wurde. Dazu wurden auf einem schwarzen Untergrund mittels weißem Klebeband Markierungen aufgeklebt.

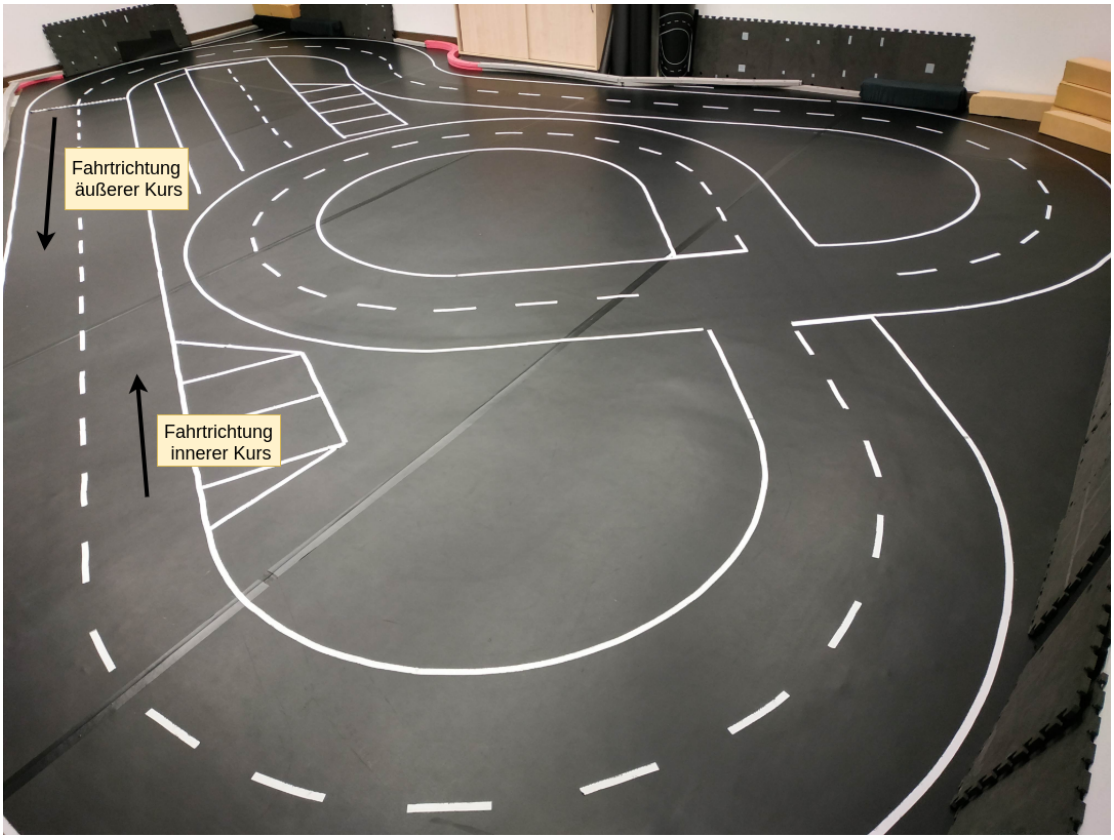

Abbildung 5.1: Testumgebung in der HAW

5.2 Straßenerkennung auf bereits aufgenommenen Bilderserien

Für eine Bewertung verschiedener Parameter der Straßenerkennung ist es vorteilhaft, wenn eine Situation exakt reproduziert werden kann. In einem Live-Test auf der Teststrecke ist die Reproduktion nur bedingt möglich, da die Situationen von äußeren Einflüssen abhängig sind. Um in einer Situation die Straßenerkennung mit verschiedenen Parametern zu testen, gibt es die Möglichkeit, die aufgenommenen Bilder und die dazugehörigen Fahrzeugzustandsinformationen, wie z.B. Position, Ausrichtung, Odometer-Werte und Geschwindigkeit, abzuspeichern. Auf der abgespeicherten Bilderserie können die Algorithmen der Straßenerkennung angewandt werden. Für eine exakte Reproduktion hat das Berechnen einer Route, sowie die Geschwindigkeit, keinen Einfluss auf den Verlauf der Bilder. Deshalb müssen die Parameter für die Länge der Strecke für die Verifizierung

und das Positionieren auf der RS in einem separaten Test auf der Teststrecke durchgeführt werden.

5.3 Herbeiführen von Fehlerfällen

Um gezielt Tests des BTR-Moduls durchzuführen, wurde ein fehler-verursachender Zustand in der FSM im Basissystem entwickelt. Dies ist notwendig, da auf der Teststrecke selten Fehler verursacht werden. Der Übergang in diesen Zustand wird mit Hilfe der Telemetrie-Anwendung manuell ausgelöst. In dieser kann angegeben werden, in welche Richtung (links oder rechts) die Fahrbahn verlassen wird. Außerdem kann die Länge der Route, auf der das Fahrzeug die Fahrbahn verlässt, verändert werden. Dies ist notwendig, da im Teststreckenraum die Entfernung zwischen Wänden und Fahrbahn unterschiedlich und nicht immer groß genug dimensioniert ist. In der Abbildung 5.2 ist eine Fehlerroute zu sehen, die ein Verlassen der Fahrbahn herbeiführt. Die Fehlerroute bildet sich dabei aus Fehlerrouten-Punkten (FRP), die einen Abstand von jeweils $200\,mm$ haben, dies entspricht der Länge eines MS. Die Winkeländerung zwischen zwei FRP beträgt dabei zwischen $5\,^\circ$ und $15\,^\circ$. Um möglichst viele verschiedene Fehlerfälle testen zu können, wird der Winkel zwischen zwei FRP jeweils per Zufall bestimmt. Es ist mit Hilfe der Telemetrie-Anwendung möglich, die Fehlerroute zu definieren, indem die Winkel genau festgelegt werden.

Der erste FRP berechnet sich aus den Routen-Punkten (RP) der geplanten Route im autonomen Normalbetrieb. Dafür werden die Routen-Punkte RP1 und RP2, welche die Fahrzeugposition einschließen, bestimmt. Dabei entspricht der RP2, der vor dem Fahrzeug liegt, dem FRP0. Um die weiteren FRP's zu bestimmen, wird ein Vektor von RP1 nach RP2 angelegt, auf eine Länge von $200\,mm$ normiert und mit Hilfe einer Rotationsmatrix um den Winkel α gedreht, anschließend wird er auf den FRP0 addiert. Das vordere Ende des Vektors zeigt dann auf den FRP1.

Die Rotationsmatrix des zufalls-generierten Winkel α lautet: $\begin{pmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{pmatrix}$

Das Vorgehen wird so lange wiederholt, bis die Länge der Fehlerroute größer ist, als die definierte Länge.

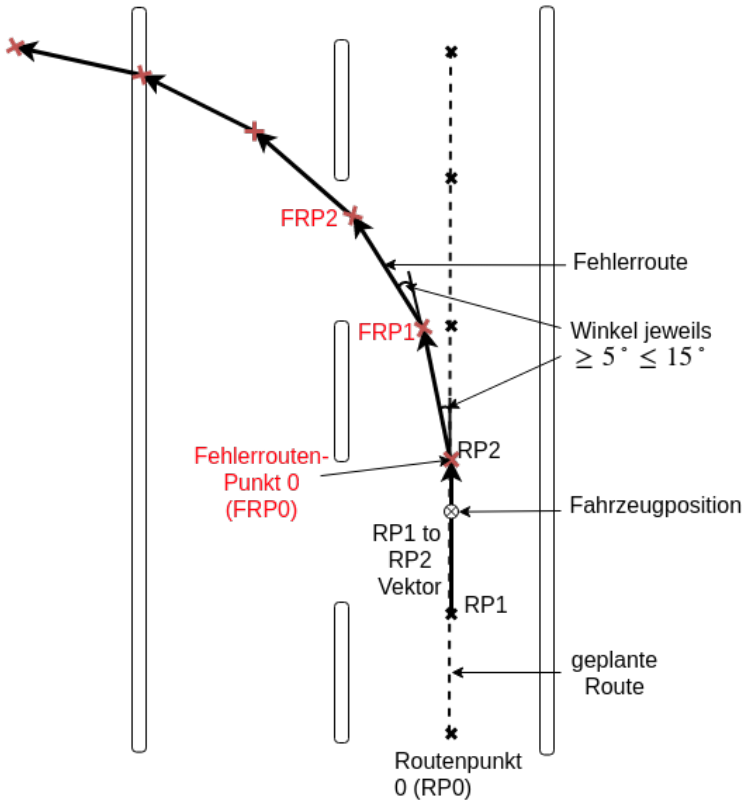

Abbildung 5.2: Fehlerroute

5.4 Parametrisierung des BackToRoad-Moduls

Mit Hilfe des Moduls zum Herbeiführen von Fehlerfällen wurde für die Parametrisierung ein regelmäßig auftretender Fehlerfall in einer Testsituation nachgestellt. In der Testsituation wurde dafür eine Gerade verwendet, da das Fahrzeug dort die höchste Geschwindigkeit aufbaut. In der Mitte der Geraden sind zwei parallele Fehlstellen mit einer maximal vorkommenden Länge von $1\,m$ vorhanden: Diese sind in der RA und der M zu finden.

In Abbildung 5.3 ist die beschriebene Testsituation zu sehen. Das Fahrzeug verlässt auf der grünen eingezeichneten Route die RS und damit die Straße. Der Ablauf ist dabei wie in Kapitel 4.3.1 beschrieben.

Zu Beginn der Testsituation hat das Fahrzeug eine Geschwindigkeit von $3\,m/s$. Beim falschen Positionieren rutscht das Fahrzeug mit dieser Geschwindigkeit nicht, oder nur

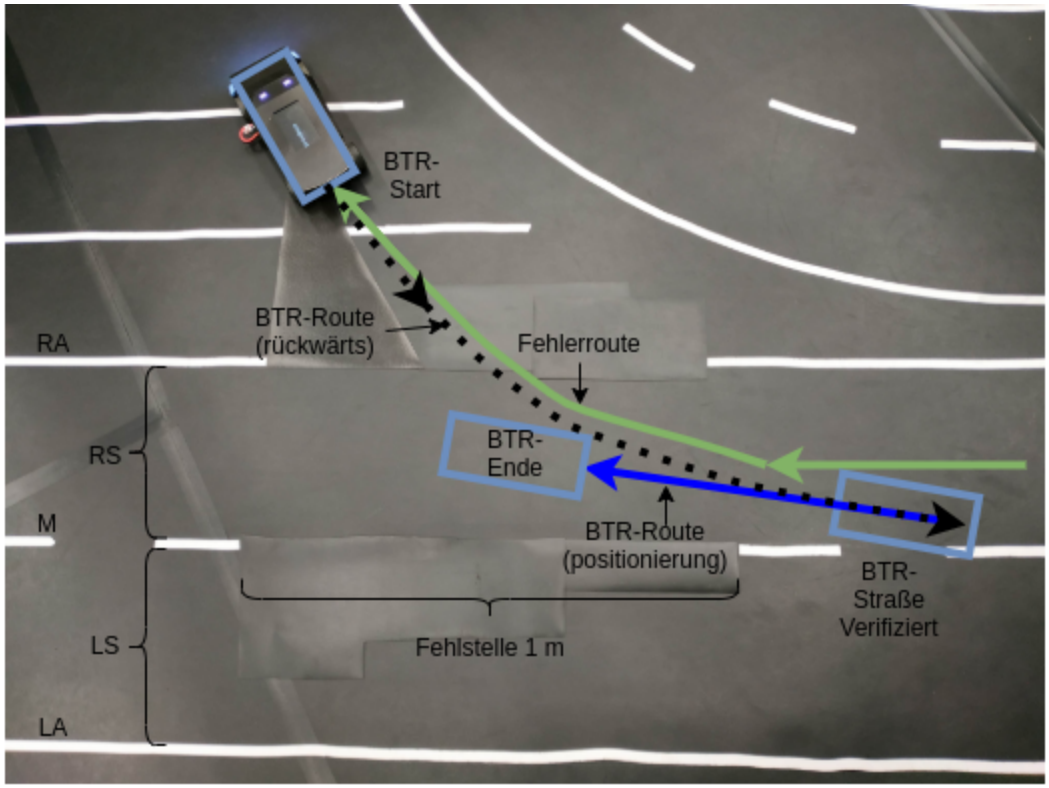

Abbildung 5.3: Fehlerfall-Situation für die Parametrisierung

minimal, da die geplante Fehlerroute abgefahren wird und der Driver das Rutschen verhindert (siehe Kapitel 3.2.1). Wenn „BTR-Straße verifiziert“ erreicht ist, wechselt die Fahrtrichtung und damit ändert sich die BTR-Geschwindigkeit von -V zu V. Durch die Richtungsänderung der Fahrtrichtung drehen die Reifen am Wendepunkt durch und das Fahrzeug kommt ins Rutschen. In der Testsituation hat das Fahrzeug ab einer Geschwindigkeit von 1,25 m/s durch das rückwärts Rutschen am Wendepunkt die MS überschritten (siehe Tabelle 5.1). Da ein Übertritt der MS aber vermieden werden soll, wurde die BTR-Geschwindigkeit auf 1 m/s festgelegt. Für die Bestimmung der rückwärts zu fahrenden Strecke wurde die gleiche Testsituation verwendet: Bei einer Strecke von 0,4 m oder weniger ist das Fahrzeug regelmäßig auf einer anderen Straße weitergefahren, da eine falsche Straße verifiziert wurde (siehe Tabelle 5.2). Bei 0,5 m und 0,7 m Strecke war die Anzahl erfolgreicher Versuche maximal, bei 0,5 m ist die BTR-Zeit kürzer, führt aber zum gleichen Ergebnis wie 0,7 m, deshalb wird 0,5 m als Standardlänge für die rückwärts zu fahrende Strecke festgelegt.

Versuch Nr.	t bei $V = 1\,m/s$ (in s)	t bei $V = 1{,}25\,m/s$ (in s)	t bei $V = 2\,m/s$ (in s)
1	2,82	X	X
2	2,79	2,35	$2,21$
3	2,87	2,54	$2,15$
4	2,83	2,45	X
5	2,80	X	X
ø	2,82	2,44	2,18

Tabelle 5.1: BTR-Laufzeiten bei herbeigeführtem Fehlerfall mit einem Winkel von 10° und einer Länge von $1{,}4\,m$. Die Werte wurden gewählt, da sie einem realistischen Fehlerfall, bei dem das Fahrzeug nach einem Hindernis die Fahrbahn verlässt, entsprechen. t ist die Zeit zwischen „BTR-Start“ und „BTR-Ende“. V ist die Geschwindigkeit des Fahrzeugs, wenn BTR aktiv ist. Bei einem überschreiten der Mittellinie ist der Versuch ungültig und mit (X) gekennzeichnet.

Strecke	$0{,}2\,m$	$0{,}3\,m$	$0{,}4\,m$	$0{,}5\,m$	$0{,}6\,m$	$0{,}7\,m$
Erfolgreiche Versuche	3/10	4/10	8/10	10/10	9/10	10/10

Tabelle 5.2: Testsituation: Anzahl erfolgreicher Versuche im Verhältnis zur Länge der rückwärts zu fahrenden Strecke.

5.5 Bewertung der Funktionalität des BackToRoad-Moduls

Für die Bewertung der Funktionalität des BTR-Moduls werden mehrere Tests auf der Teststrecke (Live-Test) durchgeführt. Die am häufigsten auftretenden Fehler werden durch das Herbeiführen von Fehlerfällen nachgestellt. Außerdem wird der in Kapitel 1 beschriebene zweite Fall, mit fehlerhaft erfasster Position und Ausrichtung getestet. Dafür wird das Rutschen des Fahrzeugs, durch manuelles Verschieben nachgestellt.

Der erste nachgestellte Fehlerfall behandelt die Situation, in der das Fahrzeug nach dem Überholen eines Hindernisses die Fahrspur auf der rechten Seite verlässt. Um einen realistischen Fall nachzustellen, wird die Länge l der Fehlerroute auf $1\,m$ festgelegt, d.h. ab der Aktivierung des Fehlerzustandes fährt das Fahrzeug $1\,m$, bevor es stoppt. Der Winkel α zwischen zwei FRP's beträgt jeweils zwischen $5\,°$ und $15\,°$. Der Fehler wird beim Einscheren auf die RS mit Hilfe der Telemetrie-Anwendung herbeigeführt. Das Fahrzeug fährt in der Überholsituation jeweils 20 Versuche auf dem inneren und äußeren Kurs (siehe Abbildung 5.1). Die Größe der Hindernisse variiert und sie werden dabei nach jedem zweiten Versuch neu positioniert.

	Innere Fahrbahn	Äußere Fahrbahn	Gesamt
Anzahl Versuche	20	20	40
Erfolgreiche Versuche	17	19	36
Hinderniskontakt	2	1	3
Falscher Straße gefolgt	1	0	1

Tabelle 5.3: Testsituation: Anzahl erfolgreicher und fehlerhafter Versuche beim Überholen von Hindernissen.

Aus Tabelle 5.3 ist zu entnehmen, dass 36 von 40 fehlerhaften Überholvorgängen erfolgreich waren, woraus sich eine Erfolgsquote von 90% ergibt. Ein Fehler mit Hinderniskontakt bedeutet, dass das BTR-Modul während der Zurückführung zur Straße das Hindernis berührt hat. Auffällig bei den Fehlern mit Hinderniskontakt ist, dass diese Fehler nur auftreten, wenn das Fahrzeug beim Überholen mit einem Abstand unter $50\,mm$ an dem Hindernis vorbei fährt.

Der zweite nachgestellte Fehlerfall beschreibt eine Kreuzungssituation, bei der die Kreuzung verlassen wird. Eine Kreuzung wird nicht immer auf die gleiche Art und Weise verlassen. Um das zu berücksichtigen, wird der Kreuzungs-Fehlerfall, der eine Fehlerrouten-Strecke von $l = 1{,}40\,m$ und einen α zwischen $5\,^\circ$ und $15\,^\circ$ hat, um zwei zusätzliche Varianten erweitert. Bei der Ersten wird die Länge l auf $1\,m$ verkürzt. Bei der Zweiten wird der Winkel α auf konstante $15\,^\circ$ gesetzt. In Abbildung 5.4 ist die Kreuzungssituation dargestellt, in der das Fahrzeug erfolgreich zurück auf die richtige Straße fährt.

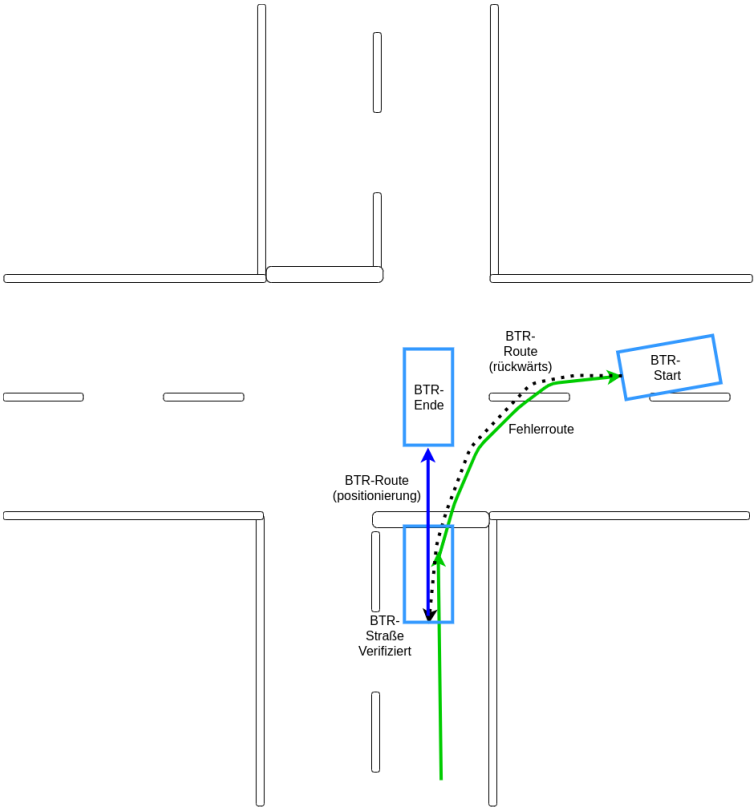

Abbildung 5.4: Kreuzungssituation: erfolgreicher Versuch

Aus Tabelle 5.4 ist zu entnehmen, dass das Zurückfinden zur Straße bei einer kurzen Strecke in 17 von 20 Fällen erfolgreich abgelaufen ist. Bei einem konstanten Winkel α von 15° ist das Fahrzeug sechsmal der falschen Straße gefolgt. Zu beobachten war, dass das Folgen der falschen Straße nur auftritt, wenn das Fahrzeug durch die Fehlerroute auf die senkrecht verlaufende Straße „abgebogen“ ist. Dieser Fall ist in Abbildung 5.5 dargestellt.

Die Erfolgsquote liegt in der Kreuzungssituation bei ca. 73 %. Die Erfolgsquote könnte durch Beheben des nicht ausreichenden Positionierens auf ca. 87 % angehoben werden.

Um den in Kapitel 1 beschriebenen zweiten Fall mit fehlerhaft erfasster Position und Ausrichtung zu testen, wird das Herausrutschen des Fahrzeugs in der Kurve nachgestellt. Um das Rutschen des Fahrzeugs zu simulieren, wird dies angehoben und neu positioniert. Dadurch wird erreicht, dass das Odometer keine Daten beim neuen Positionieren messen kann. Das Übersteuern des Fahrzeugs wird durch einen Versatz der Hinterachse herbeigeführt. Für die Fehlersituationen wird die Hinterachse jeweils 10-mal um jeweils 10 cm

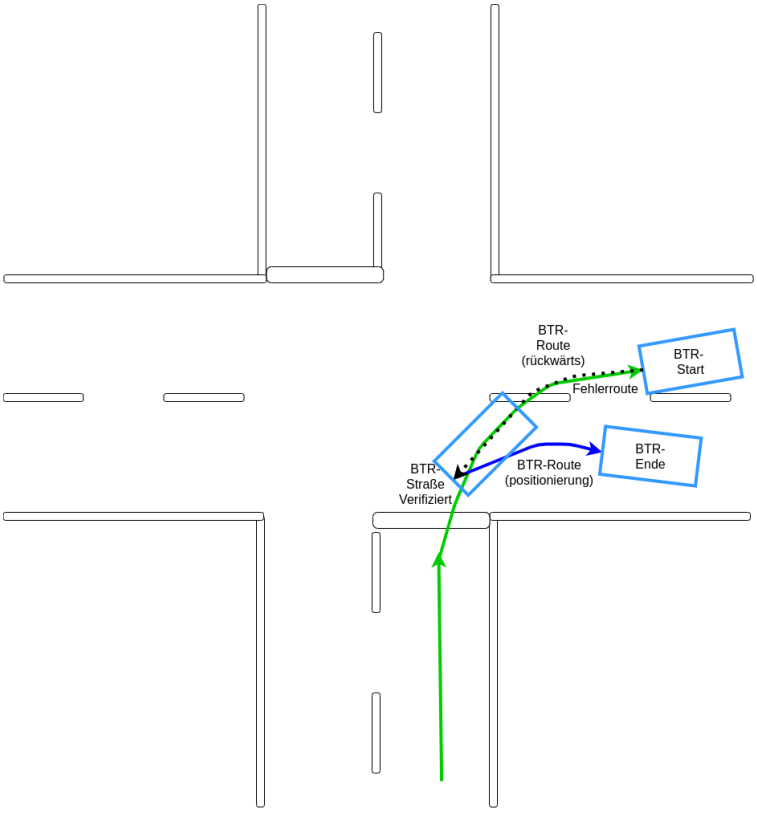

Abbildung 5.5: Kreuzungssituation: fehlerhafte Straßenauswahl, Positionierung auf der senkrecht verlaufenden Straße

und 20 cm nach außen verschoben. Außerdem wird ein Test durchgeführt, der ein Untersteuern nachstellt, in dem beide Achsen jeweils 10-mal um 10 cm und 20 cm verschoben werden.

Aus der Tabelle 5.5 ist zu entnehmen, dass nur beim Versatz beider Achsen um 20 cm, also beim Untersteuern, einmal der falschen Straße gefolgt wurde. Bei diesem fehlerhaften Versuch wurde das Fahrzeug durch die 20 cm Versatz von der richtigen Straße auf die falsche Straße versetzt. Insgesamt lag die Erfolgsquote von diesem Versuch bei 97,5 %.

Werden alle drei getesteten Situationen betrachtet, weist das BTR-Modul eine Erfolgsquote von ca. 87 % auf. Auffällig bei den Versuchen ist, dass eine falsche Straße gewählt wird, wenn das Fahrzeug sich beim Aktivieren des BTR-Moduls bereits auf einer falschen Straße befindet. Durch das Beheben des nicht ausreichenden Positionierens auf der Kreuzung, wodurch das Basissystem nicht fortfahren kann, könnte die Gesamterfolgsquote des BTR-Moduls auf 91,5 % erhöht werden.

α in °; l in m	$\alpha = 5$ - $15; l = 1{,}00$	$\alpha = 5$ - $15; l = 1{,}40$	$\alpha = 15; l = 1{,}40$
Erfolgreiche Versuche	17	14	13
Nicht ausreichend positioniert	3	4	2
Falscher Straße gefolgt	0	2	6

Tabelle 5.4: Testsituation: Anzahl erfolgreicher und fehlerhafter Versuche, die Kreuzung zu überqueren. Nicht ausreichend positioniert bedeutet, dass das Fahrzeug sich korrekt ausgerichtet auf der Kreuzung positioniert hat, aber die Erkennung der Straße durch das Basissystem nicht erfolgreich war.

Versatz in cm	$VA = 0; HA = 10;$	$VA = 0; HA = 20;$	$VA = 10; HA = 10;$	$VA = 20; HA = 20;$
Erfolgreiche Versuche	10	10	10	9
Falscher Straße gefolgt	0	0	0	1

Tabelle 5.5: Testsituation: Anzahl erfolgreicher und fehlerhafter Versuche beim Rutschen aus der Kurve. VA = Vorderachse; HA = Hinterachse;

Das BTR-Modul wurde entwickelt, um Zeit im Vergleich zum manuellen Zurückfahren zu gewinnen. Der Tabelle 5.6 ist zu entnehmen, dass eine durchschnittliche Positionierung mit dem BTR-Modul 2,82 s dauert, und eine manuelle Positionierung 7,7 s. Das BTR-Modul sorgt damit beim Zurückfahren auf die Straße für ca. 63 % Zeitersparnis gegenüber einer manuellen Positionierung.

Versuch Nr.	t bei $V = 1\,m/s$ (in s)	t per Fernsteuerung (in s)
1	2,82	7,5
2	2,79	8,5
3	2,87	9,0
4	2,83	7,0
5	2,80	6,5
ø	2,82	7,7

Tabelle 5.6: BTR-Laufzeiten beim herbeigeführtem Fehlerfall, mit einem Winkel von 10° und einer Länge von 1,4 m, die Werte entsprechen ca. der üblichen Strecke, beim Verlassen der Fahrbahn. t ist die Zeit zwischen „BTR-Start“ und „BTR-Ende“. V ist die Geschwindigkeit des Fahrzeugs, wenn BTR aktiv ist.

6 Fazit

Diese Arbeit beschäftigte sich mit der Entwicklung einer autonomen Zurückführung, die das Modellfahrzeug nach einem fehlerhaften Verlassen der Straße auf die rechte Fahrspur zurückführt.

Die Straßenerkennung, welche die Straße anhand von Mittellinien-Segmenten erkennt, hat sich bewährt. In keiner der Situationen, die im Live-Test getestet wurden, ist eine Straße fehlerhaft erkannt worden.

Die Zeitersparnis von ca. 63 % gegenüber einer manuellen Zurückführung zur Straße spricht dafür, dass es vorteilhaft ist, im Wettbewerb die autonome Zurückführung zur Straße zu verwenden. Die häufigsten Fehler, die vom BTR-Modul verursacht werden, sind das Folgen einer falschen Straße, was in den meisten Fällen nur geschah, als das Fahrzeug bereits auf einer falschen Straße stand, während das BTR-Modul gestartet wurde. Das Team-Mitglied, das während des Wettbewerbs die Kontrolle über die Fernsteuerung hat, kann trotzdem noch die Entscheidung treffen, ob das BTR-Modul eingesetzt, oder manuell zur Fahrbahn zurückgefahren wird. Sollte das Fahrzeug beim Überholen eines Hindernisses einen Fehler machen und dabei mit zu geringen Abstand am Hindernis vorbei fahren, kann manuell zur Straße gefahren werden, um einen möglichen Kontakt mit dem Hindernis während der Zurückführung durch das BTR-Modul zu vermeiden. Außerdem sollte das BTR-Modul nicht eingesetzt werden, wenn das Fahrzeug sich bereits auf einer falsch erkannten Straße befindet. In diesem Fall ist die Wahrscheinlichkeit zu groß, dass das Fahrzeug der falschen Straße weiter folgt.

Die hohe Erfolgsquote von ca. 87 % hat das Team „TeamWorstCase“ überzeugt, das BTR-Modul beim Carolo-Basic-Cup 2020 zu verwenden. Das Fahrzeug hat beim Wettbewerb jedoch keinen Fehler gemacht, bei dem die Fahrbahn verlassen, oder manuell eingegriffen werden musste. Der "Worst-Case", das Verlassen der Fahrbahn, ist nicht eingetreten und das BTR-Modul kam daher nicht zum Einsatz.

Die Erfolgsquote des BTR-Moduls könnte durch einige beschriebene Anpassungen noch weiter verbessert werden: Die nicht ausreichende Positionierung auf der Kreuzung, bei der das Basissystem nicht weiter fahren kann, könnte durch Übertragung, der vom BTR-Modul erfassten Straße an das Basissystem, behoben werden. Für den Einsatz im Wettbewerb wäre ein Detektieren des Verlassens der Fahrbahn von Vorteil, damit der manuelle Eingriff eingespart werden kann. Außerdem könnte mit der Detektion verhindert werden, dass das Fahrzeug der falschen Straße folgt, da dann die Position bekannt ist, an der die Straße verlassen wurde. Denkbar wäre es außerdem, das Fahrzeug so zu erweitern, dass eine Kamera auf die Straße hinter dem Fahrzeug ausgerichtet ist, um damit die Straßenerkennung um eine Verifizierung der Straße hinter dem Fahrzeug zu erweitern.

Beim Carolo-Basic-Cup 2020 wurde das BTR-Modul nicht benötigt, voraussichtlich wird das Team „TeamWorstCase“ aber am anspruchsvolleren Carolo-Master-Cup 2021 teilnehmen und hoffentlich, wenn es notwendig ist, mit dem BTR-Modul Zeit einsparen.

Literaturverzeichnis

[1] Carolo-Cup-Regelwerk-Komitee: *Carolo-Basic-Cup Regulations 2020*. 2019. – URL https://wiki.ifr.ing.tu-bs.de/carolocup/system/files/Basic-Cup%20Regulations.pdf. – Zugriffsdatum: 04. März 2020

[2] Intel: *Intel NUC-Kit NUC8i5BEK Technische Daten*. – URL https://www.hpiracing.com/de/kit/114356

[3] InvenSense: *MPU-9250 Product Specification*. 2016. – URL https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf. – Zugriffsdatum: 17. März 2020

[4] Jenning, Eike: *Systemidentifikation eines autonomen Fahrzeugs mit einer robusten, kamerabasierten Fahrspurerkennung in Echtzeit*. 2008. – URL https://autonomesysteme.informatik.haw-hamburg.de/papers/2008Jenning.pdf. – Zugriffsdatum: 04. März 2020

[5] Racing, HPI: *RS4 Sport 3 Drift SUBARU BRZ*. – URL https://www.hpiracing.com/de/kit/114356. – Zugriffsdatum: 10. Mai 2020

[6] Schönherr, Nils: *Kamera-basierte Minimalautonomie*. 2020. – URL https://autosys.informatik.haw-hamburg.de/papers/2019Schoenherr.pdf. – Zugriffsdatum: 17. März 2020

[7] SICK: *Lichtlaufzeitmessung*. – URL https://www.sick.com/de/de/glossar/lichtlaufzeitmessung/g/p553647. – Zugriffsdatum: 23. Mai 2020

[8] STMicroelectronics: *STM32F103 Overview*. – URL https://www.st.com/en/microcontrollers-microprocessors/stm32f103.html#overview. – Zugriffsdatum: 15. Mai 2020

[9] Systems, IDS Imaging D.: *IDS UI-3271LE*. – URL https://de.ids-imaging.com/store/ui-3271le.html. – Zugriffsdatum: 15. Mai 2020

[10] Team, KITcar: *KITcar: Unsere Softwarestruktur*. – URL https://kitcar-team.de/software.php. – Zugriffsdatum: 16. Mai 2020

[11] Team, OpenCV: *About OpenCV*. – URL https://opencv.org/about/. – Zugriffsdatum: 16. Mai 2020

A Anhang

A.0.1 Bilderserie: Übersicht verschiedener Straßensituationen

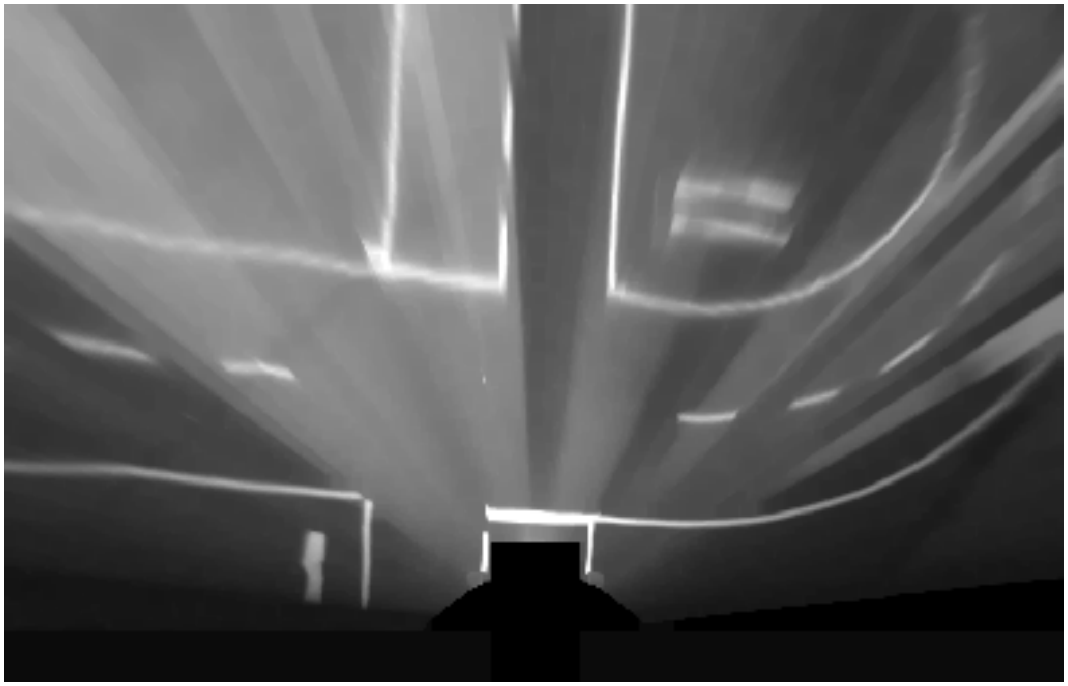

Abbildung A.1: Kreuzung mit Stopplinie in einem transformierten Bild

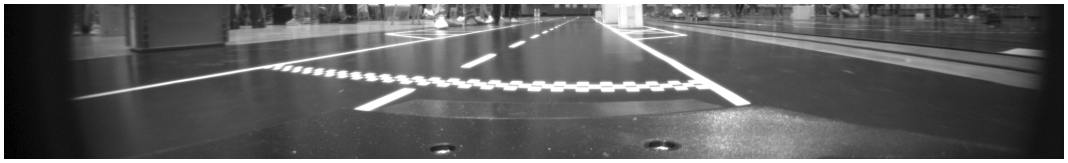

Abbildung A.2: Startlinie in einem nicht transformierten Bild

A.0.2 Bilderserie: Schritte der Erkennung von Mittellinien-Segmenten

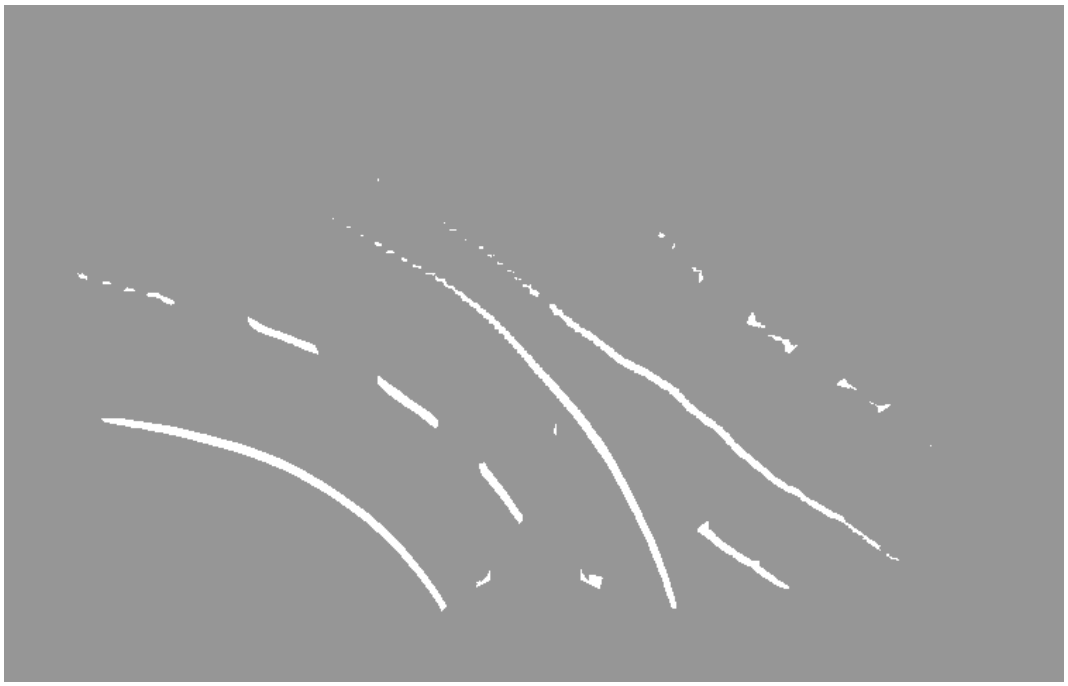

Abbildung A.3: Binarisiertes Bild mit Lücken im Grenzbereich

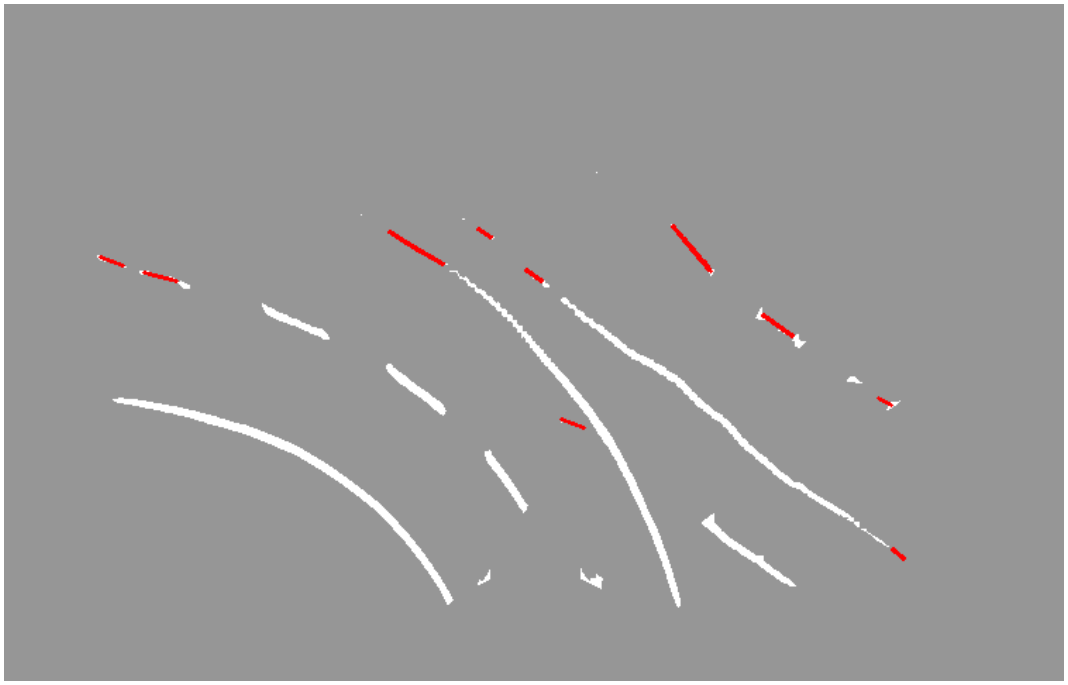

Abbildung A.4: Binarisiertes Bild mit zusammengefügten Konturen (rot)

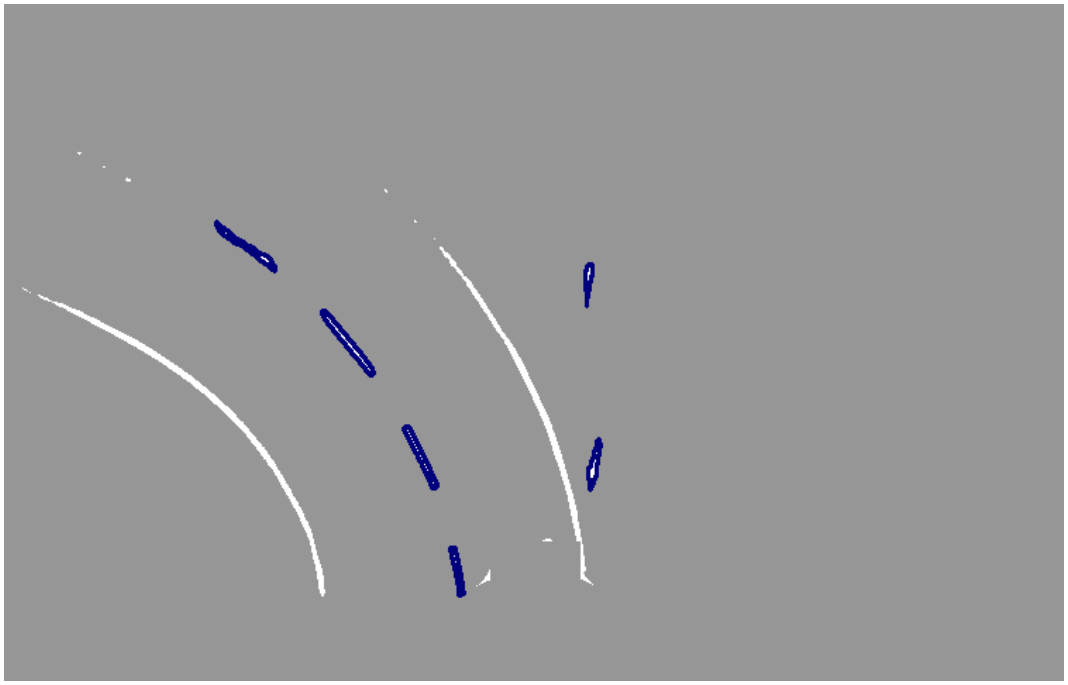

Abbildung A.5: Binarisiertes Bild mit Mittellinien-Segmenten akzeptierter Bogenlänge (blau)

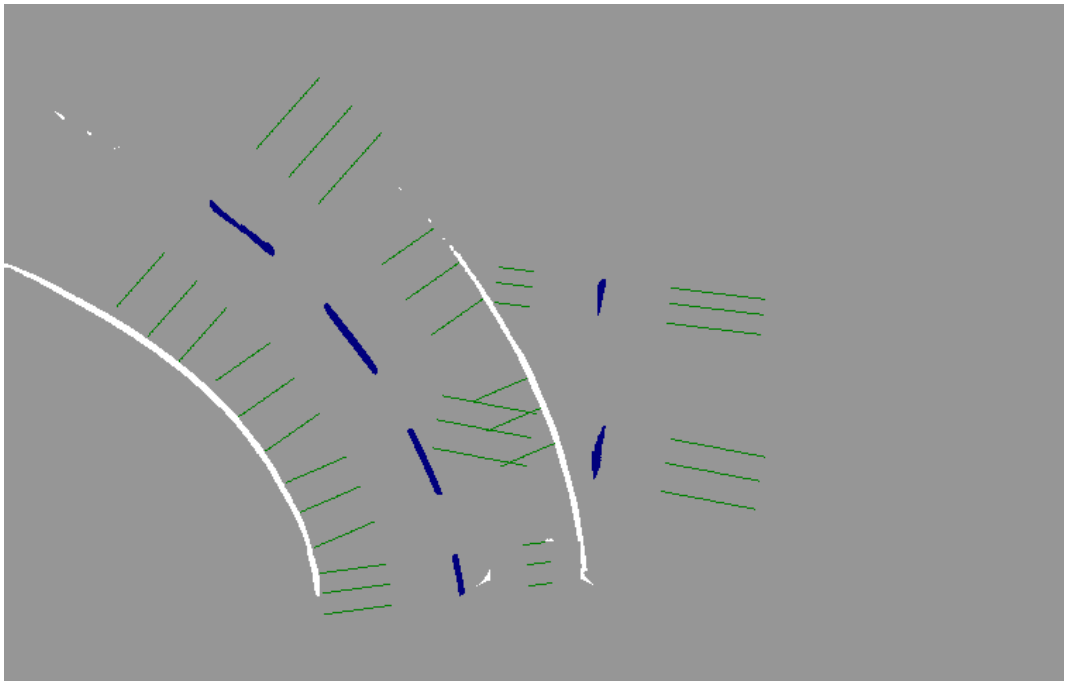

Abbildung A.6: Binarisiertes Bild mit Senkrechten zum Suchen der Außenlinien (grün)

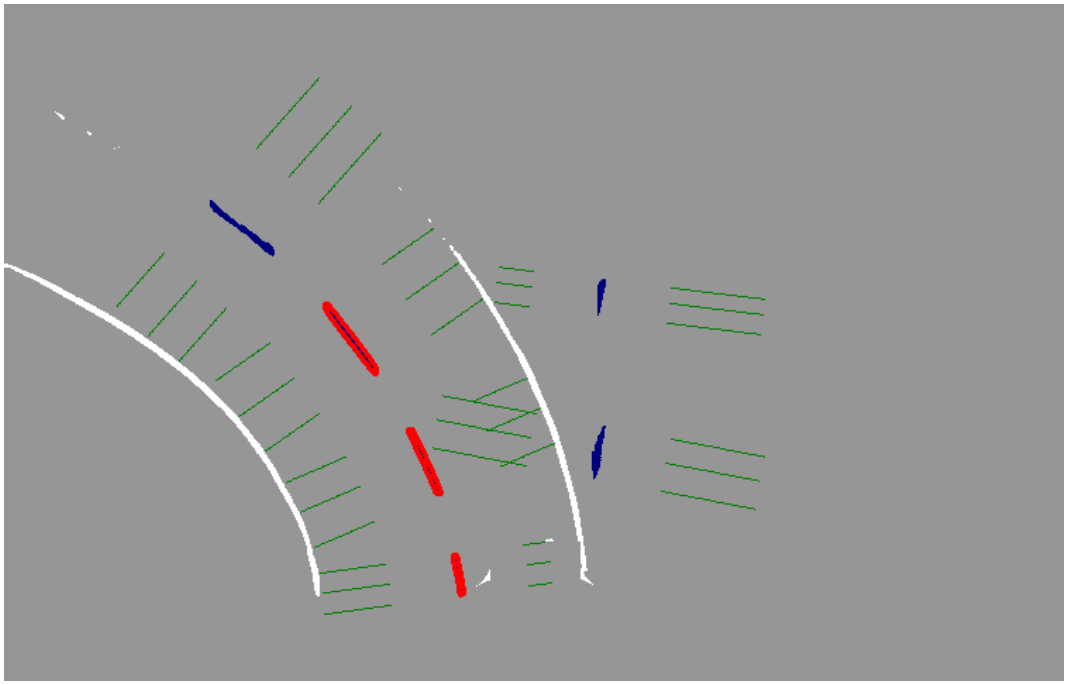

Abbildung A.7: Binarisiertes Bild mit Mittellinien-Segmenten inklusive zugeordneter Außenlinien (rot)

Glossar

Carolo-Basic-Cup Wettkampf der Basisleistungsklasse innerhalb des Carolo-Cups.

Carolo-Cup Ein Wettbewerb zum autonomen Fahren, veranstaltet von der TU Braunschweig.

Carolo-Master-Cup Wettkampf der erweiterten Leistungsklasse innerhalb des Carolo-Cups.

Fahrbahn Ein Bereich, der aus mehreren Spuren besteht und vom Fahrzeug befahren werden darf.

Fahrspur Eine Spur einer Fahrbahn.

Feature Umfasst im Bild erkannte Start- und Stopplinien, sowie Hindernisse.

OpenCV Die „Open Source Computer Vision Library“ stellt eine Infrastruktur für Computer Vision-Anwendungen bereit und verfügt über optimierte Algorithmen.

Time of Flight Sensor Sensor, der nach dem Prinzip der Lichtlaufzeitmessung arbeitet und das Zeitintervall vom Aussenden eines Lichtpulses bis zum Empfang des am Objekt reflektierten Lichtsignals misst und daraus die Distanz zwischen Sensor und Objekt ermittelt.[7].

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

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

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

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

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

Autonome Zurückführung nach Verlassen der Fahrbahn für autonome Modellfahrzeuge

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

_____	_____	_____
Ort	Datum	Unterschrift im Original