Fahrzeugtechnik & -antriebe

EVOMOTIV

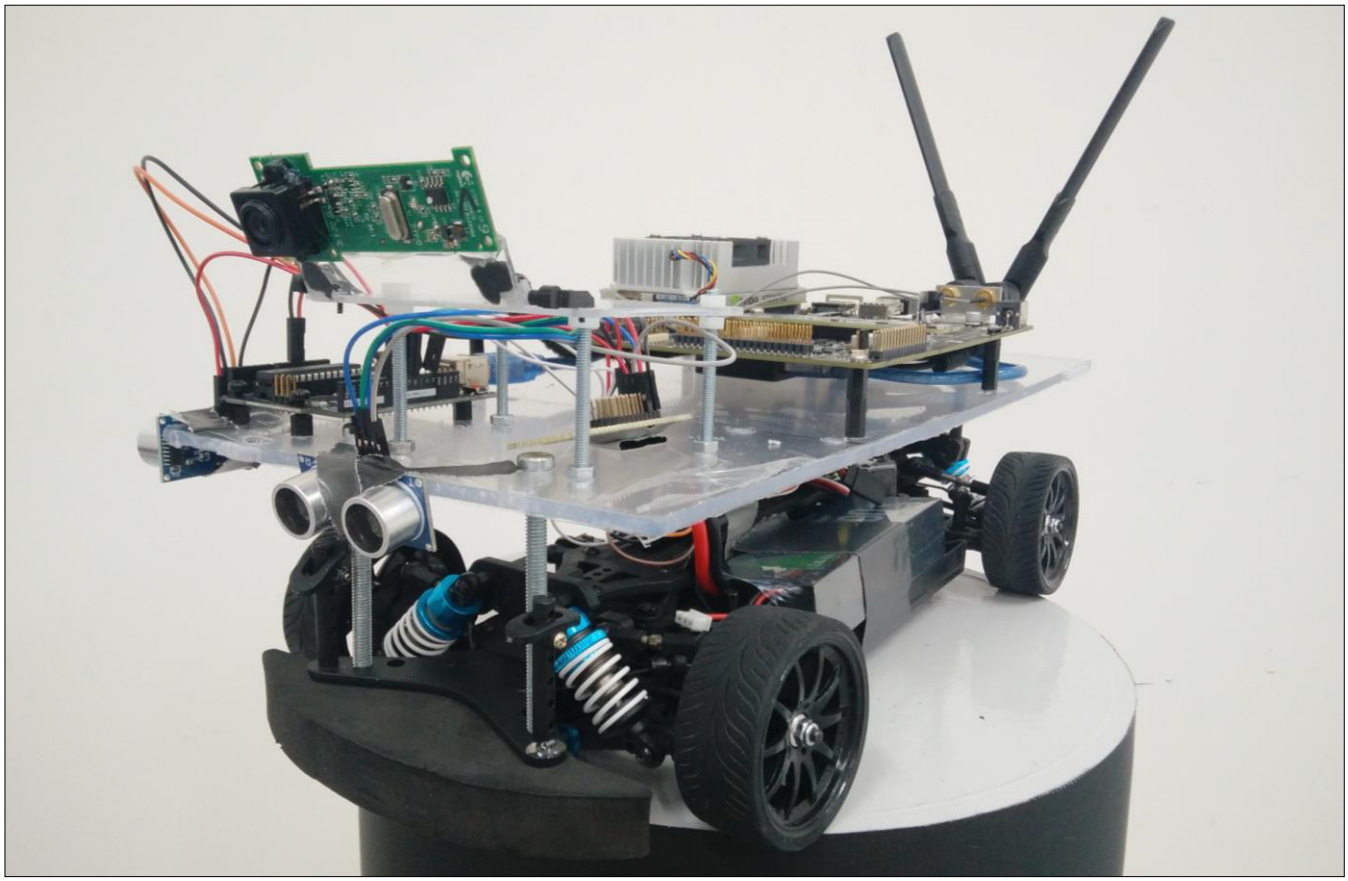
Brandenburgische Technische Universität Cottbus

Masterarbeit MA042

Lehrstuhl
Fahrzeugtechnik und –antriebe

Thema

Entwicklung eines Demonstrators für Fahrerassistenzsysteme und Implementierung einer Querführung



Name, Vorname: Fitzer, Fabian
Matrikel-Nr.: 3215017
Zeitraum: 01.10.2017 – 31.03.2018
Exemplar: 1 / 3

Prof. Dr.-Ing. Heinz Peter Berg
M.Sc. David Bromberger
Dipl.-Ing. (FH) Arthur Kessler

BTU Cottbus-Senftenberg
LS Fahrzeugtechnik & -antriebe
Siemens-Halske-Ring 14
03046 Cottbus

Brandenburgische Technische Universität Cottbus - Senftenberg

Masterarbeit-Nr.:	042	**Betreuer 1:**	Prof. Dr.-Ing. Heinz Peter Berg
Name, Vorname:	Fitzer, Fabian	**Betreuer 2:**	M.Sc. David Bromberger
Matrikel-Nr.:	3215017		
Beginn:	01. Oktober 2017	**Extern:**	Dipl.-Ing. (FH) Arthur Kessler
Ende:	31. März 2018		
Verlängerung:			
Note:		**Exemplare:**	BTU-FTA: 3

Sperrvermerk: ~~Arbeit darf in Teilen veröffentlicht werden~~.
Arbeit darf nicht veröffentlicht werden, die Beteiligten erhalten registrierte Exemplare.

Thema: **Entwicklung eines Demonstrators für Fahrerassistenzsysteme und Implementierung einer Querführung**
Development of a demonstrator for driver assistance systems and implementation of transverse guidance

Automatisiertes Fahren stellt einen wachsenden Markt dar, der hinsichtlich Sicherheit und Mobilität viel Potenzial bietet. Im Bereich Fahrerassistenzsysteme hat EVOMOTIV mittlerweile ein breites Know-How aufgebaut. Dieses soll weiter ausgebaut und im Bereich automatisiertes Fahren ergänzt werden. Durch eigene Lösungen in diesen Bereichen können Alleinstellungsmerkmale und ggfs. eigene Produkte entstehen. Zu diesem Zweck soll ein Demonstrator entstehen, der verschiedene Funktionen des automatisierten Fahrens abbilden kann. Dazu sind von Herrn Fitzer folgende Aufgaben zu erfüllen:

- Recherchearbeiten:
  - zu aktuellen Fahrzeugassistenzsysteme
  - Akzeptanz und Relevanz im Hinblick auf das automatisierte Fahren für den Kunden
- Aufbau eines Fahrzeuges zur Demonstration von hochautomatisierten Fahrfunktionen
- Integration von Sensorik und Rechnerhardware
- Entwicklung einer erweiterungsfähigen Softwareumgebung
- Implementierung einer Spurfolgefunktion und Hinderniserkennung
- Analyse und Ausblick verwendeter Ablaufstrukturen der implementierten Funktionen in der Hardware des Demonstrators

FTA-Bedingungen:
Alle Daten sind mit geeigneter Software abzulegen und zu dokumentieren. Die Arbeit wird schriftlich und digital dem LS FTA übergeben. Die digitalen Daten unterliegen ebenso wie konkrete Messergebnisse der Geheimhaltungspflicht des LS FTA. Auf Wunsch wird die Arbeit für maximal fünf Jahre geheimgehalten, siehe optionale Geheimhaltungserklärung.
Den Abschluss der Arbeit bilden eine 20minütige Präsentation am LS FTA; bei der der Betreuer der Fremdfirma anwesend sein muss; und ein vom Studenten zu erstellendes Plakat im Format DIN A0 Hochformat (digital in Power Point, Corel Draw, Adobe Acrobat oder PhotoShop, zusätzlich ausgedruckt), welches einen Überblick über die Arbeit und die Ergebnisse bietet. Das Plakat muss zur öffentlichen Darstellung am Lehrstuhl geeignet sein. Der Betreuer der Fremdfirma unterbreitet einen detailliert begründeten Notenvorschlag.
Die Arbeit ist in drei Exemplaren in einem verschlossenen Umschlag/Karton dem Studierenden-Sekretariat vorzulegen, dort registrieren zu lassen und anschließend dem Betreuer des Lehrstuhls persönlich oder dem Sekretariat des Lehrstuhls ungeöffnet auszuhändigen. Der Meldebogen ist vom Studierendensekretariat ausfüllen zu lassen.

- Prof. Dr.-Ing. Heinz Peter Berg - *- M.Sc. David Bromberger -* *- Dipl.-Ing. (FH) Arthur Kessler-*

Eidesstattliche Erklärung

Der Verfasser erklärt, dass er die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt hat. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

---------- ----------

Ort/Datum Unterschrift

Kurzzusammenfassung

Fahrerassistenzsysteme bilden einen zunehmend wichtigen Teil der Automobilentwicklung und sind somit auch Gegenstand aktueller Forschung. Das Ziel dieser Arbeit besteht in der Entwicklung eines erweiterbaren Demonstrators für die Ausbildung von Ingenieuren in diesem Bereich. Hierzu wird ein Konzept auf Basis eines ferngesteuerten Fahrzeuges erstellt. Die Entscheidungen zu Hardware- und Softwarekomponenten werden dargelegt und ein Fahrzeug als Versuchsgrundlage aufgebaut. Durch die Implementierung einer Softwarebasis mithilfe des „Roboter Operating Systems“ können Sensordaten systemweit beliebig verwendet werden. Anschließend werden verschiedene Methoden der digitalen Bildverarbeitung mit „OpenCV“ zur Spur- und Schildererkennung skizziert und bewertet. Den Schluss dieser Arbeit bildet die Ableitung eines Algorithmus aus den gewonnen Daten zur Querregelung des Fahrzeuges. Nachfolgend werden verschiedene Ansatzpunkte der Erweiterung des Datensatzes und der Funktionalität aus den bestehenden Daten dargelegt.

Abstract

Driver assistance systems form an increasingly important part of automotive development and are therefore also the subject of current research. The goal of this thesis is in the development of an expandable demonstrator for the education of engineers in this field. For this purpose, a concept is created based on a radio-controlled car. The decisions on hardware and software components are shown and a vehicle is built as an experimental basis. By implementing a software environment using the „Robot Operating System“, sensor data can be used anywhere in the system. Then different methods of digital image processing are outlined and evaluated with „OpenCV“ for track and sign recognition. The conclusion of this work is the derivation of an algorithm from the data obtained for transverse guidance of the vehicle. Finally, various starting points for extending the data set and functionality from the existing data are set out.

Inhaltsverzeichnis

Formelzeichen und Abkürzungen

Formelzeichen

Zeichen	Einheit	Benennung
λ	m	Wellenlänge
$\dot{r}$	m/s	Relativgeschwindigkeit
a_y	m/s^2	Querbeschleunigung
b	-	Y-Achsensverschiebung
C_0	m/s	Lichtgeschwindigkeit
d	m	Entfernung
EG	s^2/m	Eigenlenkgradient
$f_{Doppler}$	Hz	Doppler-Frequenz
l	m	Radstand
M	kg/mol	Molare Masse
m	-	Steigung
n_{Rad}	U/s	Raddrehzahl
R	$(kg*m^2)/(s^2*mol*K)$	Universelle Gaskonstante
R	m	Kurvenradius
t	s	Zeit
T	s	Umlaufzeit
T	K	Temperatur
U	m	Radumfang
$v_{Fahrzeug}$	m/s	Fahrzeuggeschwindigkeit
v_{Schall}	m/s	Schallgeschwindigkeit
δ_H	-	Gierverstärkung

θ	°	Gradientenwinkel
ϰ	-	Adiabatenexponent
μ	-	Gauss Peak
ρ	-	Abstand vom Ursprung
σ	-	Varianz

Abkürzungen

ACC	Adaptive Cruise Control
API	Application Programming Interface
ESP	Elektronische Stabilitätskontrolle
FAS	Fahrerassistenzsysteme
FSR	Full Speed Range
GPIO	general purpose input/output
IMU	inertial measurement unit (Inertiale Messeinheit)
KOS	Koordinatensystem
PWM	Pulsweitenmodulation
RC	Radio Control
RGB	Rot, Grün, Blau
ROI	Region of Interest
ROS	Robot Operating System
SAE	Society of Automotive Engineers
VDA	Verband Deutscher Automobilindustrie
WHO	World Health Organization
IPM	Inverse Perspective Mapping

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Alleine im Jahr 2017 gab es in Deutschland 3177 Verkehrstote zu beklagen. Das sind 38 Getötete auf eine Million Einwohner [1]. Weltweit betrachtet sind die Zahlen noch weitaus erschreckender. Die neusten Berichte der WHO aus dem Jahr 2015 beschreiben insgesamt 1,25 Millionen Tote im Jahr 2013 und die Zahlen bleiben seit Jahren auf einem anhaltend hohen Niveau. Somit sind Verkehrsunfälle im internationalen Vergleich die neunt häufigste Todesursache. Bei jungen Menschen zwischen 15 und 29 Jahren führen Verkehrsunfälle diese Statistik sogar an [2].

Neben schlecht ausgebauter Infrastruktur und ungenügenden Sicherheitsvorschriften sind diese hohen Zahlen vor allem auf menschliches Versagen zurückzuführen. Trotz einer erheblichen Zunahme des Verkehrsaufkommens in den letzten Jahrzehnten in Deutschland und einer ebenso stark gestiegenen Anzahl von Unfällen, nahm die Zahl der Getöteten merklich ab. Jedoch ist dieser Trend in den letzten Jahren stagnierend. Eine weitere Reduzierung scheint somit nicht durch eine Anpassung der Gesetzeslage oder Verbesserung der passiven Sicherheit von Fahrzeugen möglich [3].

An dieser Stelle können die Hersteller und Zulieferer der Automobilbranche mit Innovationen im Bereich Fahrerassistenz und autonomes Fahren ansetzen. So hat sich zum Beispiel die Firma Volvo das Unternehmensziel gesetzt bis zum Jahr 2020 die Zahl der getöteten Personen in neuen Fahrzeugen der Marke auf null zu reduzieren [4].

Um dieses Ziel zu erreichen und insgesamt die Zahl der Unfälle drastisch zu reduzieren, bedarf es einer Vielzahl verschiedener Sensoren und Aktoren im Automobil. Die Entwicklung von Funktionen zur Entlastung des Fahrers stellt forschende Unternehmen jedoch vor besondere Herausforderungen. Übergibt der Fahrer die Verantwortung an ein technisches System, muss dessen Funktionsfähigkeit immer gewährleistet werden können. Auch die Umfeld Erkennung, Streckenplanung und Regelung muss nicht nur unter Sicherheitsaspekten funktionieren, sondern auch Komfortansprüchen gerecht werden. Ingenieursgesellschaften wie die EVOMOTIV GmbH können mit ihrer Arbeit hier einen entscheidenden Beitrag leisten. Diese Thesis zeigt auf, wie mit frei verfügbaren Mitteln ein autonomes Fahrzeug als Demonstrator aufgebaut werden kann. Dabei wird ein Entwurf erstellt, der, unter der Verwendung von Standardtools der Automobilindustrie, die Möglichkeit der einfachen Erweiterung bietet. So kann in Zukunft das Fahrzeug nicht nur als Demonstrator verwendet werden, sondern dient auch als Experimentier- und Entwicklungsumgebung für Studierende und Jungingenieure.

Ein besonderes Augenmerk bei der anschließenden Umsetzung des Konzeptes liegt auf der Verwendung frei verfügbarer Software und Hardware. Dies wird durch die Open Source Lizensierung von Betriebssystem, Middleware und Bildverarbeitung möglich. Eine Objekt- und Spurerkennung wird so implementiert, dass der Demonstrator selbstständig einer Spur folgen kann und dabei Hindernisse und Stoppschilder beachtet. Des Weiteren wird die Sensorik zur Parklückenvermessung für autonome Längsparkmanöver verwendet.

Zu Beginn dieser Thesis wird ein Konzept für ein autonomes Fahrzeug im Maßstab 1:10 erstellt. Dabei werden die Entscheidungen zur Verwendung der Sensorik und Softwarearchitektur begründet und in den Kontext zu aktuell verfügbaren Seriensystemen in der Fahrzeugbranche gestellt. Im Anschluss erfolgt eine Beschreibung der Umsetzung von einzelnen Fahrfunktionen. Nachfolgend werden die gewählten Algorithmen der Bildverarbeitung und Fahrzeugsteuerung veranschaulicht und die Ergebnisse von Tests unter Realbedingungen bewertet. Schließlich folgen Erläuterungen zu den möglichen Erweiterungen und Verbesserungen sowohl auf Software-, als auch auf Hardwareebene.

2 Problemstellung und Zielsetzung

Diese Masterthesis bildet den Start für das Projekt „Demonstrator für autonomes Fahren und Car2Car-Kommunikation“ der EVOMOTIV GmbH. Als Ingenieursgesellschaft der Bereiche Maschinenbau, Elektrotechnik, sowie Luft- und Raumfahrttechnik liegt ein Fokus der Tätigkeiten und Leistungen auf den Bereichen Fahrerassistenzsysteme und autonomes Fahren. Diese Fachbereiche sind in sogenannten Kompetenzfeldern zusammengefasst. Ein grundsätzliches Ziel besteht dabei immer in dem Aufbau und der Weitergabe von Wissen und Erfahrung, um eine ganzheitliche Entwicklungskompetenz, bis hin zur serienreifen Gesamtfahrzeugentwicklung anbieten zu können [5].

In der Vergangenheit hat die EVOMOTIV GmbH bereits eine Vielzahl von Projekten in den genannten Zukunftsbereichen erfolgreich bearbeitet. Dabei konnte ein Aufbau von eigenem Wissen, durch den Einsatz bei Kunden vor Ort, nur begrenzt stattfinden. Die bestehenden Kompetenzen aus dem Bereich „Fahrerassistenzsysteme“ sollen nun erweitert und mit Wissen aus dem Gebiet des autonomen Fahrens ergänzt werden. Dies soll die Möglichkeit eröffnen, eigene Produkte und Lösungen zu diesen Themenbereichen anbieten zu können. Als Plattform soll hierzu ein Fahrzeug im Miniaturmaßstab verwendet, um hochautomatisierte oder autonome Fahrfunktionen vorführen zu können. Dieses Fahrzeug soll als Aushängeschild der Firma agieren und kann bei der Akquise von Kundenprojekten unterstützend eingesetzt werden. Durch ein maßgeschneidertes internes Weiterbildungssystem und die gezielte Ausbildung von Praktikanten und Studierenden im Rahmen von Abschlussarbeiten soll das gesammelte Wissen weitergegeben werden. Ein Baustein dieses Prozesses kann in Zukunft das Projekt „Demonstrator Autonomes Fahren und Car2Car-Kommunikation“ darstellen.

Das Ziel der vorliegenden Arbeit besteht in der Entwicklung und dem Aufbau eines Demonstrators. Dabei wurde ein Konzept entwickelt, welches eine spätere Erweiterung einfach möglich macht. Ein Fokus liegt auf der Auswahl und Vernetzung passender Hardware und Software. Diese gibt dem Fahrzeug die grundsätzliche Möglichkeit autonome Fahrfunktionen darzustellen. Auf dieser Basis werden zwei Programme entwickelt die in bestimmten Szenarien solche Funktionen demonstrieren. Dabei besteht nicht der Anspruch ein vollständig autonom fahrendes Fahrzeug zu schaffen, welches selbstständig sämtliche Verkehrssituationen bewältigen kann. Es sollen lediglich einzelne Funktionen in einem gewählten Szenario zu Demonstrationszwecken dargestellt werden können.

.

3 Überblick Fahrerassistenzsysteme

Eine zunehmende Automatisierung des Güter- und Individualverkehrs wird weithin als Revolution der Mobilität betrachtet. Nicht der Mensch steht in der Verantwortung des unfallfreien Fahrens, sondern ein technisches System. Dieser Umstand bringt vor allem rechtliche und ethische Fragen mit sich. Daneben wirft die technische Umsetzung einerseits eine Vielzahl an Fragen und Problemen auf und birgt aber andererseits auch neue Chancen und Möglichkeiten für Unternehmen und den einzelnen Menschen. Das folgende Kapitel gibt einen Überblick über am Markt befindliche Systeme und erläutert die Stufen des autonomen Fahrens und eine Begriffsdefinition. Dabei erfolgt eine Beschränkung auf den Individualverkehr im PKW-Bereich.

3.1 Autonomes Fahren

Der Begriff „autonomes Fahren“ stammt aus dem Altgriechischen. Die Komposition aus den Worten „autòs“ (selbst, persönlich) und „nómos“ (menschliche Ordnung, vom Menschen gesetztes Recht) hat in der deutschen Sprache eine Reihe von Bedeutungen in unterschiedlichen gesellschaftlichen Feldern [6]. Politik, Soziologie, Philosophie und Ethik spielen neben der Technik eine Rolle. Im Bereich der Fahrzeugentwicklung ist vor allem die Bedeutung der Eigenständigkeit hervorzuheben und schafft somit eine klare Abgrenzung vom Begriff des Automobils. Hierbei steht die Fähigkeit sich zu bewegen („mobilis“; lat.: beweglich [7]) im Gegensatz zur Eigenbestimmung des autonomen Fahrens im Vordergrund. Somit findet eine Verlagerung der Autonomie vom Fahrer hin zum Fahrzeug statt.

Neben dem Begriff „autonom“ findet in der Literatur auch „automatisiert“ häufig Verwendung. Dabei gilt es zu bedenken, dass diese Begrifflichkeiten ein relativ junges Feld der Forschung abbilden und somit viele Rahmenbedingungen noch nicht abgesteckt sind. Hinzu kommt die Verwendung des Wortes in mehreren Technikbereichen, wie der Luftfahrt und des Güterverkehrs. Um Verwechslungen und Überschneidungen im Bereich der Automobilbranche zu vermeiden wurde von der SAE, eine ab Januar 2014, gültige Begriffsdefinition eingeführt, die autonomes Fahren in sechs Stufen unterteilt. Die SAE International ist eine internationale Gesellschafft mit Sitz in den USA, die sich zum Ziel gesetzt hat, weltweit gültige Standards zu schaffen, um die Entwicklung von Mobilitätslösungen zu vereinfachen und zu verbessern [8].

In Tabelle 3-1 sind die einzelnen Stufen aufgeführt. Bei dieser Darstellung handelt es sich um eine Übersetzung aus dem Englischen. Die Stufen reichen von „keinerlei Automation“ bis zu einem „vollständig autonomen und selbstfahrenden Fahrzeug“. Eine ähnliche Definition veröffentlichte 2012 auch die „Bundesanstalt für Straßenwesen“. Hierbei wird der Grad der Automation in 5 Stufen unterteilt, die von „keiner Automation“ bis einer „hohen Automation“ reichen. Ein vollständig selbstfahrendes, alle Szenarien abdeckendes System wird in diesem Kontext nicht genannt [9]. Die rechtlichen Konsequenzen der Unterteilungen und der aktuelle Stand der Serientechnik und Forschung werden in den nachfolgenden Kapiteln beschrieben.

Tabelle 3-1: Stufen des autonomen Fahrens nach SAE [10]

Stufe	Bezeichnung	Beschreibung
0	No Automation	Der Fahrer hat jederzeit die Kontrolle über alle Fahrfunktionen und überwacht die Umgebung
1	Driver Assistance	Übernahme von Beschleunigung/Bremse oder Lenkung in bestimmten Fahrsituationen
2	Partial Automation	Übernahme von Beschleunigung/Bremse und Lenkung in bestimmten Fahrsituationen
3	Conditional Automation	Das System übernimmt in bestimmten Situationen die Kontrolle, der Fahrer muss jederzeit eingreifen können
4	High Automation	Das System übernimmt in bestimmten Situationen die Kontrolle, ohne auf einen Fahrer angewiesen zu sein
5	Full Automation	Das System vollführt alle Fahraufgaben, die ein menschlicher Fahrer sonst übernehmen würde

Die Zunahme der Automatisierung zwischen den einzelnen Stufen basiert auf der Ausweitung der Assistenzfunktionen und deren Vernetzung. Dazu muss die Umfelderkennung von Stufe zu Stufe genauer werden und mehr Aufgaben des Fortbewegungsvorganges vom Menschen an das Fahrzeug übergeben werden.

Um Doppeldeutigkeiten und Fehlinterpretationen zu vermeiden wird in dieser Arbeit grundsätzlich der Begriff „autonomes Fahren“ verwendet. In Hinblick auf den Demonstrator dieser Arbeit ist von einem automatisierten Fahrzeug die Rede, dass sämtliche Fahraufgaben in gewählten Szenarien ohne menschliche Eingriffe bewältigen kann. Laut Definition der SAE entspricht dieser Anwendungsfall einer Automation der Stufe vier, da nicht sämtliche Fahraufgaben des Straßenverkehrs abgedeckt werden. Allerdings ist eine Übertragung der allgemeingültigen Definition für Personenkraftwagen auf ein Fahrzeug für Demonstrationszwecke nicht sinnvoll, da weder eine Anwendung im Straßenverkehr in Planung ist, noch ein Eingriff durch einen Fahrzeugführer im herkömmlichen Sinn möglich ist. Trotzdem kann ein Demonstrator einzelne Funktionen, die zur Erreichung der einzelnen Stufen nötig sind, aufzeigen und deren Umsetzung und Anwendung visualisieren.

3.2 Aktuelle Systeme

Aktuelle Populärliteratur und Artikel mit einem nicht wissenschaftlichen Schwerpunkt unterscheiden meist zwischen autonomen und teilautonomen Fahrzeugen. Hierbei ist zum einen ein Fahrzeug gemeint, dass ohne Eingriff des Fahrers sämtliche Fahraufgaben übernehmen kann und zum anderen eine Vernetzung von Fahrerassistenzfunktionen, um lediglich in bestimmten Situationen die Kontrolle an das Fahrzeug abgeben zu können. Eine genauere Unterscheidung findet zum Beispiel in der Online

Ausgabe des Magazins „Der Spiegel“ nicht statt [11]. Die Hersteller von Fahrzeugen beschränken sich heutzutage meist auf eine Aufzählung aller verfügbaren Funktionen bei der Einführung einer neuen Baureihe, verzichten aber auf die Einteilung in die jeweiligen Stufen nach SAE [12]. Dabei werden Marketingnamen für neue Technologien eingeführt und die Funktionen beschrieben, jedoch selten die technische Umsetzung in den Vordergrund gestellt.

Im folgenden Kapitel wird der aktuelle Stand der Technik im Bereich Fahrerassistenzsysteme aufgezeigt und zukünftige Möglichkeiten werden erläutert. Zur besseren Orientierung wurde eine Einordnung in die Klassifizierung nach den Stufen des autonomen Fahrens der SAE gewählt. Diese sind in Tabelle 3-1 aufgeführt. Dabei ist zu beachten, dass aufgrund der Aktualität des Themas und der fortlaufenden Forschung keine allumfassende Nennung gewährleistet werden kann. Der Begriff „Fahrerassistenzsysteme“ (im Folgenden FAS) bezeichnet im Allgemeinen alle Systeme, die den Fahrer unterstützen. Somit auch z.B. Entertainment, Navigationsfunktionen oder Fahrwerksregelsysteme wie ESP. Die Verwendung des Begriffes in der vorliegenden Arbeit bezieht sich lediglich auf die Kategorie der prädiktiven Assistenzsysteme.

Stufe 0 bezeichnet ein herkömmliches Fahrzeug, bei dem der Fahrer jederzeit die Kontrolle über sämtliche Funktionen behält. Hierbei besteht keine technische Unterstützung durch FAS. Jedoch können Warnsysteme wie der „Totwinkelwarner“ oder eine „Spurverlassenswarnung“ verbaut sein. Ein Totwinkelwarner setzt auf rückwärtig ausgerichtete Sensoren (i.d.R. Radar) zur Erfassung von Hindernissen im Bereich des toten Winkels des Fahrzeuges. Bei einer positiven Erkennung erfolgt eine akustische und/oder optische Warnung für den Fahrer. Eine „Spurverlassenswarnung“ setzt auf die optische Erkennung der Fahrspur (meist durch Kameradaten). Diese Warnung erfolgt bei einem unbeabsichtigten Verlassen der Spur, wenn kein Blinker gesetzt ist. Es kann akustisch ausgeführt sein, durch einen Warnton, optisch im Kombiinstrument und/oder durch ein taktiles Feedback, wie eine Vibration im Lenkrad oder Sitz.

Stufe 1 beschreibt ein Fahrzeug mit FAS zur Quer- oder Längsregelung. Querregelung bezeichnet die Kontrolle über die Querführung des Vehikels. Quer ist in diesem Fall als orthogonal zur Fahrrichtung definiert und somit in Spurrichtung. Eine Querregelung übernimmt die Kontrolle über die Lenkung. Im Gegensatz dazu regelt eine Längsführung, längs zur Fahrrichtung und hat somit die Kontrolle über die Vorgänge Beschleunigung und Bremsen [13]. Viele aktuelle Fahrzeuge fallen in diese Kategorie, da die Technologie bereits verfügbar ist. Ein erstes Längsregelsystem wird bereits seit 1958 von der Firma „Chrysler“ unter der Bezeichnung „Cruise Control“ angeboten. Diese Geschwindigkeitsregelanlage regelt das Fahrzeug auf eine vom Fahrer eingestellte Geschwindigkeit, durch elektronische Motoreingriffe (früher mechanisch). Moderne Systeme erlauben über Steuergerätezugriff auch einen Bremseingriff, um beispielsweise bei Bergabfahrt die Geschwindigkeit konstant zu halten [14]. Eine Erweiterung dieses Systems stellt die adaptive Geschwindigkeitsregelanlage dar. Die Fahrzeuggeschwindigkeit wird neben der Wunschgeschwindigkeit des Fahrers auch auf vorausfahrende Fahrzeuge angepasst und ein Sicherheitsabstand eingestellt. Zur Erfassung von Hindernissen wird meist auf Daten verschiedener Radarsensoren für Nah- und Fernbereichserkennung zurückgegriffen. In seiner aktuellen Ausprägung als

ACC FSR („Full Speed Range“) ermöglicht dieses System eine Abbremsung bis zum Stillstand und das selbstständige Anfahren in einem bestimmten Zeitintervall. Erste Systeme dieser Art finden bereits seit Anfang der 2000er Jahre in der Serie Verwendung [15]. Eine Möglichkeit der Querregelung ist durch Spurhalteassistenten gegeben. Es erfolgt eine Erkennung der Fahrspur durch eine im Frontbereich verbaute Kamera. Wird die aktuelle Fahrzeugposition relativ zur Fahrspur mit einbezogen, kann die momentane Abweichung berechnet werden. In einem ersten Schritt wird der Fahrer über das Verlassen der Spur lediglich gewarnt (siehe Stufe 0, Spurverlassenswarnung). Um eine Querführung aus den Daten der Spurerkennung zu realisieren bedarf es vor allem der Möglichkeit in die Lenkung einzugreifen. Dies wird in Form von elektromechanischen Lenksystemen realisiert, die die herkömmlichen elektrohydraulischen Systeme ersetzen. Eine exakte Beschreibung der möglichen Regelalgorithmen einer Querführung erfolgt in Kapitel 7.3.

Eine gleichzeitige Kombination von Quer- und Längsführung ermöglicht ein Fahrzeug der **Stufe 2.** Dieser Stand ist in vielen aktuellen Fahrzeugen durch die Kombination und Erweiterung der oben genannten Systeme im Serieneinsatz möglich. Um dies zu erreichen und eine umfassende Umfelderkennung zu ermöglichen, ist eine Vielzahl von Sensoren notwendig. Die verschiedenen Hersteller setzen hierbei auf eine unterschiedliche Kombination aus Radar, Lidar, Kameras und Ultraschallsensoren in abwechselnder Stückzahl. Auf eine Aufzählung aller Sensorkombinationen aller Hersteller wird in dieser Arbeit verzichtet.

Werden Quer- und Längsführungssysteme kombiniert kann ein Fahrzeug, Fahraufgaben eigenständig bewältigen. Allerdings sind hier einige Einschränkungen zu beachten, die ein Fahrzeug der Stufe 5 und damit ein vollständig autonomes Fahrzeug charakterisieren würden. Die größte Einschränkung bezieht sich auf den Anwendungsfall. Aktuelle Systeme können lediglich in bestimmten Szenarien Fahraufgaben übernehmen. Beispiele dafür sind der Stau- und der Einparkassistent. Ein Stau auf der Autobahn stellt ein relativ überschaubares Szenario dar. Aufgrund der baulichen Trennung der Fahrspuren und des Fehlens von Fußgängern und Lichtzeichen ist die Umfeldüberwachung auf die Fahrzeuge der näheren Umgebung und die Fahrspur beschränkt. Hinzu kommen niedrige Geschwindigkeitsbereiche. Auch beim automatisierten Parken ist die Geschwindigkeit niedrig. Das Fahrzeug vermisst eine Parklücke und übernimmt, nach der Bestätigung durch den Fahrzeugführer, Quer- und Längsführung, um das Fahrzeug in die Lücke zu steuern. Diese Systeme gibt es in unterschiedlicher Ausprägung. In einem ersten Schritt übernimmt das Fahrzeug nur die Lenkung und der Fahrer muss selbstständig bremsen und beschleunigen und die Gangwechsel übernehmen. Aktuelle Systeme übernehmen alle Aspekte der Fahrsituation und stellen somit ein System der Stufe 2 dar [16]. Der wichtigste Aspekt, der die oben genannten Systeme beschränkt, ist die notwendige Überwachung durch den Fahrer. Die Verantwortung liegt zu jeder Zeit und somit auch bei aktivierten Systemen beim Fahrzeugführer. Außerdem ist dieser verpflichtet, jederzeit die Kontrolle übernehmen zu können und muss daher seine volle Aufmerksamkeit gewährleisten. In der Praxis wird dies durch zeitliche Beschränkungen erreicht. Durch kapazitive Sensoren oder Kraftsensoren am Lenkrad wird überprüft, ob der Fahrer seine Hände am Lenkrad hat. Ist dies nicht der Fall wird der Fahrer nach einer bestimmten Zeit („hands off time“) akustisch und/oder

optisch gewarnt. Darauf folgen mehrere Stufen der Deaktivierung, die bis zu einem Nothalt und Verständigung einer Notfallbehörde führen können.

Um das nächst höhere Level der **Stufe 3** zu erreichen sind die nötigen Systeme, in Hinblick auf Sensorik und Rechenleistung, heute schon in der Serie verfügbar. Als erstes Serienfahrzeug bezeichnet Audi den A8 der Modellreihe D5 als Fahrzeug, welches in bestimmten Situationen vollständig automatisiert ist. Der Funktionsumfang entspricht dabei den oben genannten Stauassistenten („Staupilot“) der Stufe 2. In einem Geschwindigkeitsbereich bis 60 km/h kann das System auf Straßen mit einer baulichen Trennung Quer- und Längsführung eigenständig ausführen. Die wichtigste Änderung liegt in der Verantwortung, die hierbei an das System und somit an die Audi AG übergeht. Das bedeutet für den Fahrer, dass er durch die Entbindung von seinen Fahraufgaben, während der Fahrt abgelenkt sein darf. Zur Erreichung dieser Automatisierung ist von technischer Seite vor allem eine doppelte Absicherung der einzelnen Systeme umzusetzen. Im Gegensatz zu einem Fahrzeug der Stufe 2 muss für die Stufe 3 bei einem Ausfall, zum Beispiel des primären Lenksystems ein Ersatzsystem zur Verfügung stehen („Fallback-System“). Auch die Software muss die Funktion im Falle einer Fehlerkennung weiterhin gewährleisten können. Eine Redundanz der Systeme ist jederzeit zu gewährleisten. Per Definition muss jedoch der Fahrer im Falle einer Notsituation oder wenn die Systemgrenzen erreicht werden (z.B. eine Geschwindigkeit > 60 km/h beim Staupiloten) die Kontrolle wieder übernehmen können. Aktuell wird dazu eine Übergabezeit in Höhe von ca. 10 Sekunden angestrebt. Durch entsprechende Warnung muss das System den Fahrer zur Übernahme der Fahraufgaben mit einer Vorwarnzeit auffordern. Aktuell ist diese Funktionalität grundsätzlich in den Fahrzeugen auf Wunsch verbaut, jedoch nicht aktiviert. Die Gesetzeslage lässt zurzeit noch keine Fahrzeuge der Stufe 3 zu, wobei mit einer Änderung dieses Umstandes innerhalb der nächsten zwei Jahre gerechnet wird [17].

Die **Stufe 4** des autonomen Fahrens bedingt eine hohe Automatisierung des Systems, das nicht auf einen Fahrer als „Fallback-Strategie“ angewiesen ist. Der VDA (Verband Deutscher Automobilindustrie) sieht als Anwendungsfälle dieser Stufe fahrerloses Parken und eine städtische Fahrsituation vor [18]. Das fahrerlose Parken, auch Valet Parking genannt wird bereits seit mehreren Jahren erprobt. Dabei übergibt der Fahrer bei der Einfahrt eines Parkhauses das Fahrzeug und dieses sucht fahrerlos eine Parkposition und kehrt später auf Anforderung wieder zurück. Technisch wird zwischen strukturierten und unstrukturierten Umgebungen unterschieden. Strukturierte Umgebungen basieren meist auf einer genauen Umgebungskarte. Ein Beispiel für ein Valet Parking System in einer unstrukturierten Umgebung wurde in einem größeren Umfang erstmals auf der „Consumer Electronics Show“ 2015 in Las Vegas gezeigt. Das System der BMW Group orientiert sich dabei an Daten einer Karte und speziell entwickelter Lasersensoren, um in einem Parkhaus automatisiert einzuparken [19]. Einen anderen Ansatz verwendet die Daimler AG in einem Pilotprojekt mit der Robert Bosch GmbH in Stuttgart. Dabei wird ein Serienfahrzeug verwendet und lediglich das Parkhaus durch Sensoren aufgerüstet. Die Fahrspurplanung und Hindernisvermeidung erfolgt extern und wird dann kabellos an das Fahrzeug übertragen [20].

Einen weiteren großen technischen Entwicklungsschritt bildet die **Stufe 5**. Diese sieht ein fahrerloses Fahrzeug vor, dass alle Fahrsituationen automatisiert bewältigen kann. Die Verantwortung liegt dabei jederzeit beim Fahrzeug, bzw. dessen Hersteller. Per definitionem sind für diese Stufe Fahrzeugkonzepte vorstellbar, die vom heutigen Verständnis deutlich abweichen. So sind zum Beispiel keine Steuerelemente für den Fahrer, wie Lenkrad und Fahrpedale, für den Fahrbetrieb notwendig. Viele Hersteller haben bereits Konzepte oder Entwicklungsfahrzeuge mit den entsprechenden Fähigkeiten präsentiert. Die größte Aufmerksamkeit erreichte dabei die Firma „Waymo“. Als Tochterunternehmen des Technikunternehmens Google, wurde bereits 2015 mit dem selbstentwickelten Fahrzeug „Firefly“ ein Konzept ohne Pedale und Lenkrad auf öffentlichen Straßen getestet [21].

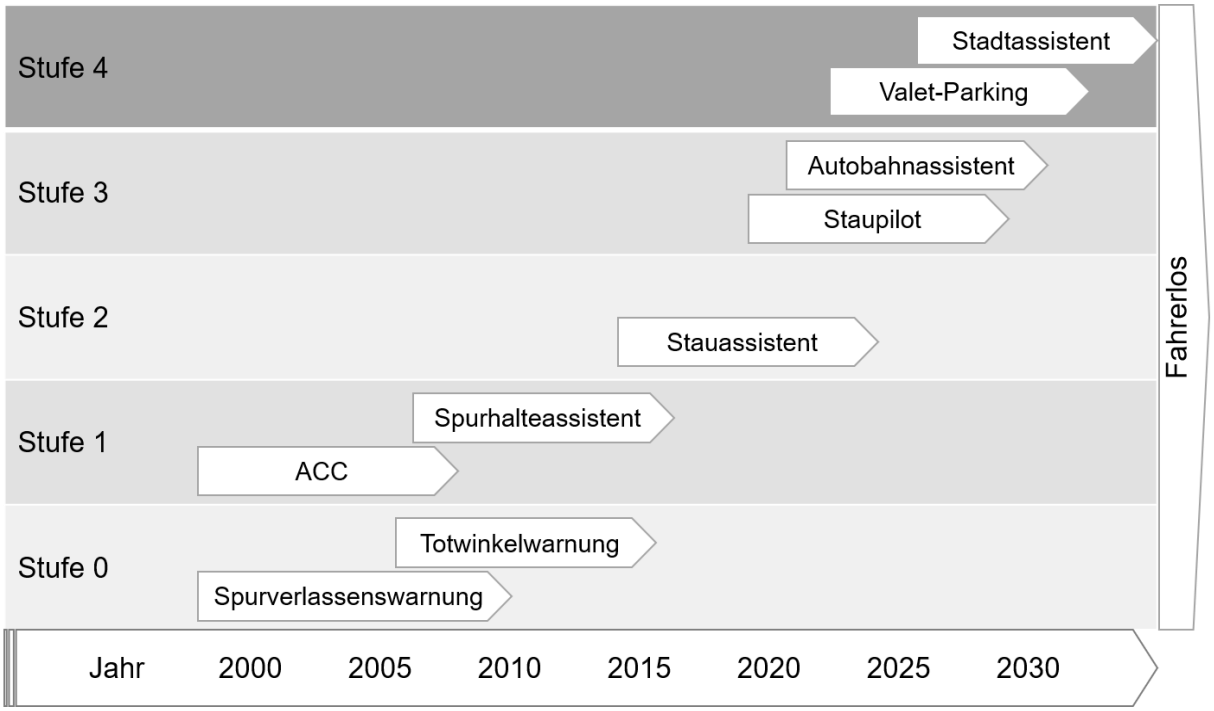


Abbildung 3-1: Übersicht FAS nach VDA [18]

Abbildung 3-1 zeigt eine Übersicht der am Markt befindlichen Systeme. Diese orientiert sich an der Übersicht des VDA und klassifiziert diese nach den Stufen des autonomen Fahrens. Es lässt sich erkennen, dass ein Serieneinsatz von Systemen der Stufe 2 heute bereits Realität und auch Stufe 3 technisch möglich ist.

3.3 Gesellschaftliche Relevanz

Neben der technischen Umsetzung von autonomen Fahrfunktionen sind für eine Gesellschaft vor allem zwei Aspekte wichtig. Zum einen müssen rechtliche Grundlagen geschaffen werden, um Funktionen von Herstellerseite zu testen und diese später als Kunde nutzen zu dürfen. Zum anderen besteht die Frage nach der Kundenakzeptanz für automatisierte Fahrfunktionen. Dazu ist auch die Marktverfügbarkeit und die Ausrüstungsquote der aktuellen Fahrzeugflotte mit FAS zu betrachten.

In Deutschland wird der Straßenverkehr durch die Straßenverkehrsordnung (StVO) geregelt und dieses bildet somit auch den rechtlichen Rahmen für das autonome Fahren. Die aktuelle Version dieses Gesetzes wurde im April 2017 verabschiedet und soll die Lücken zwischen dem, was erlaubt und dem was technisch möglich ist, zu schließen. So wird nun explizit in §1a der Fahrer als Fahrzeugführer bestimmt, auch wenn dieser ein System zum automatisierten Fahren aktiviert [22]. Diese Regelung stellt einerseits eine Neuerung dar, andererseits sind damit nur Systeme bis zu einem Grad der Stufe 2 abgedeckt und damit der aktuelle Serienstand. Dies erklärt die Notwendigkeit einer gesetzlichen Neuregelung, um beispielsweise das in Kapitel 3.2 genannte System der Firma Audi auf öffentlichen Straßen zu erlauben. Daneben gibt es zahlreiche Beispiele für Sondergenehmigungen der einzelnen Bundesländer und Kommunen zum Testen hochautomatisierter Fahrfunktionen auf öffentlichen Straßen. Diese Erlaubnis wird zusehends von den Regionen als ein Faktor für wirtschaftlichen Erfolg gesehen. Dabei wird die Veränderung der Gesetzgebung, vor allem um sich im internationalen Vergleich einen Vorteil zu verschaffen, forciert. Bei diesem Vorgehen ist es wichtig, länderübergreifende Standards zu finden, um einerseits die Entwicklung für die Hersteller zu vereinfachen und andererseits den Endverbrauchern eine gewisse Sicherheit zu geben. Hierbei gilt es Normen zur technischen Abnahme von Systemen einzuführen und Standards für die Überprüfung festzulegen.

Aktuelle Untersuchungen des Deutschen Verkehrssicherheitsrates (DVR) aus dem Jahr 2016 zeigen, dass nur ca. 40% aller Neuwagen mit mindestens einem Fahrerassistenzsystem ausgerüstet sind. Dabei wird vom Verbraucher ein zu hoher Preis, im Durchschnitt als dritthäufigste Ursache für eine Entscheidung gegen ein FAS angegeben. Der erwartete Nutzen spielt ebenfalls eine untergeordnete Rolle. Hingegen ist die Verfügbarkeit für das jeweilige Fahrzeugmodell ausschlaggebend. Hinzu kommt, dass die Interessenten oft keine Kenntnis über die Funktion und Verfügbarkeit diverser Systeme besitzen [23]. Eine Datenbank des DVR unter der Aktion „Bester Beifahrer“ listet dazu die Verfügbarkeit verschiedener Assistenzsysteme nach Fahrzeugklassen auf [24]. Abbildung 3-2 aus dem Jahr 2017 weist auf deutlich eine zunehmende Verfügbarkeit aller Assistenzsysteme in Abhängigkeit der Fahrzeugkategorie hin. Dies zeigt, dass die neuen Technologien zuerst in den hochpreisigen Fahrzeugen der Hersteller verbaut werden und nach und nach eine breitere Käuferschicht durch die Verfügbarkeit in Massenprodukten erreicht wird. Eine Studie der Firma Robert Bosch GmbH aus dem Jahr 2012 unterstützt die These, dass eine Ausrüstung mit FAS häufig an einem Unwissen der Käufer über Funktion und Nutzen scheitert. Weiterhin stimmen 18% aller befragten deutschen Probanden der Behauptung voll oder teilweise zu, dass FAS unnötige Technik sei [25]. Dahingegen belegen Untersuchungsergebnisse des amerikanischen Marktforschungsunternehmes „J.D. Power“, dass nur ein geringer Teil der Käufer von Automobilen mit FAS überhaupt die verbaute Technik in ihren Fahrzeugen nutzt. So sagten ca. 35% der Befragten aus, dass sie den Einparkassistenten noch nie verwendet hätten. Dabei wurden im Jahr 2015 amerikanische Fahrer befragt, die seit mindestens 90 Tagen ein neues Fahrzeug mit FAS nutzten [26].

Ein Aspekt der das Kauf- und Nutzungsverhalten von FAS maßgeblich beeinflusst ist das subjektive Komfortempfinden. Aktivierung und der Aufwand zur Überwachung eines Systems müssen für den Konsumenten in einem positiven Verhältnis zum tatsächlichen Nutzen stehen und somit für eine Entlastung

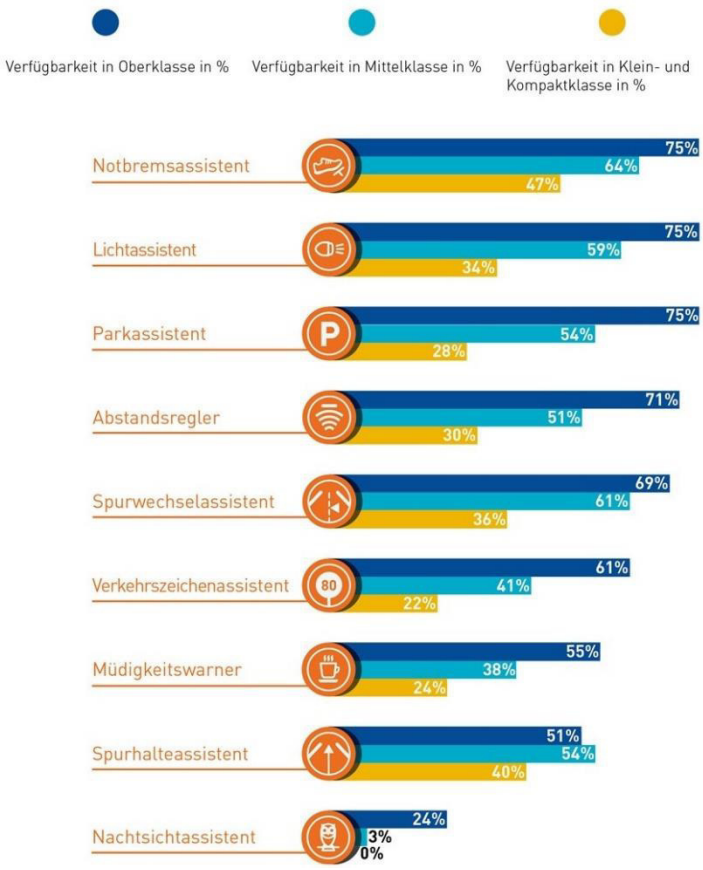


Abbildung 3-2: Verfügbarkeit von FAS nach Fahrzeugklassen [24]

sorgen [27]. Ähnliche Faktoren spielen auch bei der Akzeptanz von autonomen Fahrzeugen, bzw. automatisierten Fahrfunktionen eine Rolle. Grundlegend muss festgehalten werden, dass es zu diesem Thema bisher nur empirische Untersuchungsdaten gibt und auch nur diese geben kann. Die vorgestellten Studien basieren auf Fragen zu fiktiven Zukunftsszenarien, da eine Markteinführung für die untersuchten Gegenstände zum jetzigen Zeitpunkt noch nicht absehbar und planbar ist.

Eine Untersuchung der Technologieberatungsfirma „Detecon“ aus dem Jahr 2016 zeigt eine eher positive Wahrnehmung von autonomen Fahrzeugen durch die 643 befragten Personen. Es können sich generationenübergreifend etwas mehr als die Hälfte der Probanden die Nutzung eines solchen Fahrzeuges vorstellen. Demgegenüber steht nur ca. ein Drittel ablehnend. Allerdings lässt sich eine starke Abhängigkeit der Meinung vom Alter feststellen. Abbildung 3-3 zeigt diesen Zusammenhang in Diagrammform. Jüngere Menschen zwischen 18 und 29 Jahren weisen eine deutlich größere Zustimmung der Nutzungsabsicht auf als andere Altersgruppen [28]. Diese Generation wird auch als „Digital Natives“ bezeichnet und ist demnach mit Computertechnik aufgewachsen. Zu einem ähnlichen Ergebnis kommt der „TÜV Rheinland in einer Umfrage aus dem Jahr 2017 mit 1408 Befragten. Bei der Frage nach der Nutzungsabsicht gaben 86% der 18-29-jährigen Probanden eine positive Antwort, jedoch nur 62% bei den 50-60-jährigen. Diese Untersuchung stellt zudem eine Korrelation zwischen der allgemeinen Nutzungshäufigkeit und der Nutzungsabsicht von autonomen Fahrzeugen dar.

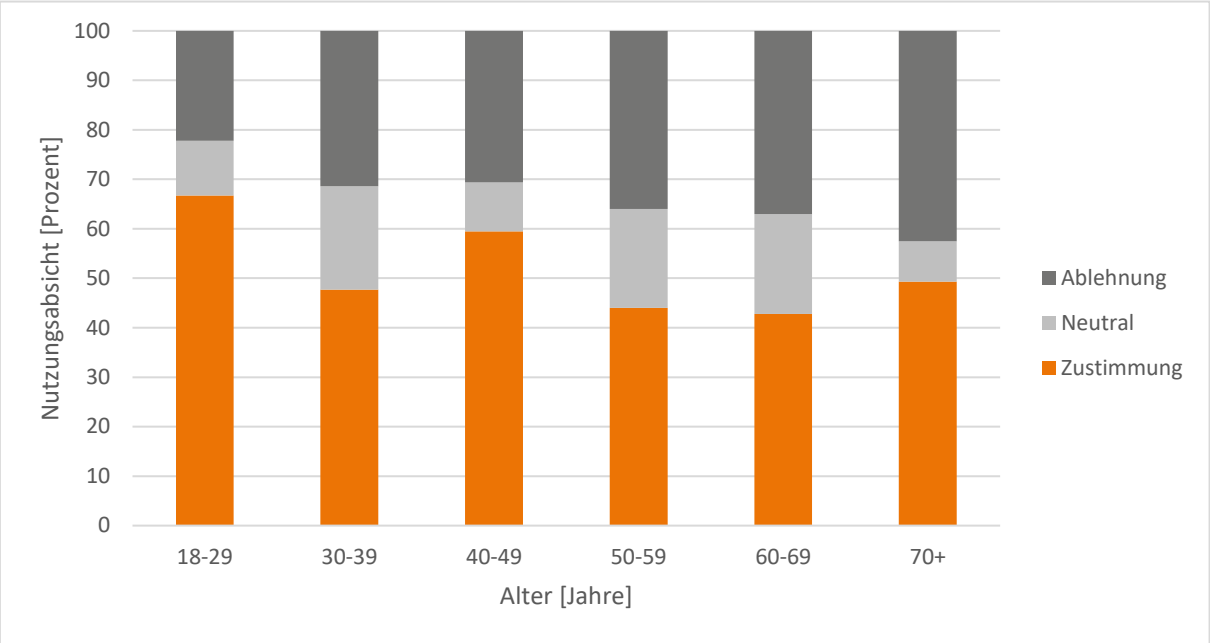


Abbildung 3-3: Nutzungsabsicht autonomes Fahren nach Alter [28]

Bei einer jährlichen Fahrleistung von mehr als 30.000km gaben ca. 80% eine positive Antwort auf die Frage nach der Nutzungsabsicht (unabhängig vom Alter). Bei einer jährlichen Fahrleistung von unter 10.000km lag diese Rate lediglich bei 66% [29]. Somit lässt sich feststellen, dass Menschen, die bisher besonders viel Zeit im Fahrzeug verbringen, eher autonomes Fahren nutzen würden. Diese Feststellung deckt sich mit der These, dass der Komfortzugewinn eine wichtige Rolle bei der Akzeptanz autonomer Fahrzeuge spielt. Hinzu kommt die Frage nach dem möglichen Nutzungsgebiet. Hier zeigt sich eine deutliche größere Zustimmung für den privaten Bereich. 50,9% der Befragten sehen eine private Nutzung positiv, aber nur 41,7% können sich autonome Fahrzeuge in der geschäftlichen Nutzung vorstellen. Gleichzeitig lehnt eine deutlich höhere Anzahl an Befragten die geschäftliche Nutzung grundsätzlich ab (42,5% gegen 32,2% bei der privaten Nutzung) [28]. Diese Zahlen illustrieren, dass autonomes Fahren als Entlastung und damit als Zeitgewinn für die Freizeit gesehen wird.

Der „TÜV Rheinland“ hat lediglich Personen befragt, die einen eigenen PKW besitzen. Die „Detecon“-Studie unterscheidet an dieser Stelle zwischen Probanden mit und ohne Automobil und kann hierbei auch einen deutlichen Unterschied feststellen. Personen ohne eigenen PKW haben eine höhere Zustimmungsrate als Personen, die bereits ein Auto fahren. In diesem Zusammenhang wird der Reiz des eigenständigen Fahrens, ohne Assistenzsysteme als Spaßfaktor der PKW-Nutzung beschrieben und erklärt die negative Einstellung zu FAS.

Ein wichtiger Aspekt für die Befragten stellt die Sicherheit der autonomen Systeme dar. Hierzu wurde sowohl nach der Ausfallwahrscheinlichkeit, als auch nach der Gefahr von Hackerangriffen gefragt. Insgesamt werden diese Themen vom Nutzer tendenziell kritisch betrachtet. 65% der Befragten der „Detecon“-Studie schätzen das Risiko von autonomen Fahrzeugen zurzeit als zu hoch ein. Ebenso befürchten 62% einen möglichen Kontrollverlust durch Hackerangriffe. Ein positives Zeugnis stellen die Probanden des „TÜV Rheinland“ der Deutschen Automobilindustrie aus. 19,5% trauen am ehesten der

„Daimler AG“ zu, ein funktionierendes autonomes Fahrzeug zu entwickeln. Nur 10% sehen hier „Tesla“ vorne. Insgesamt wird Herstellern von höherpreisigen Fahrzeugen die größte technische Kompetenz im Bereich autonomes Fahren zugetraut Dieses Ergebnis ist in Abbildung 3-4 dargestellt.

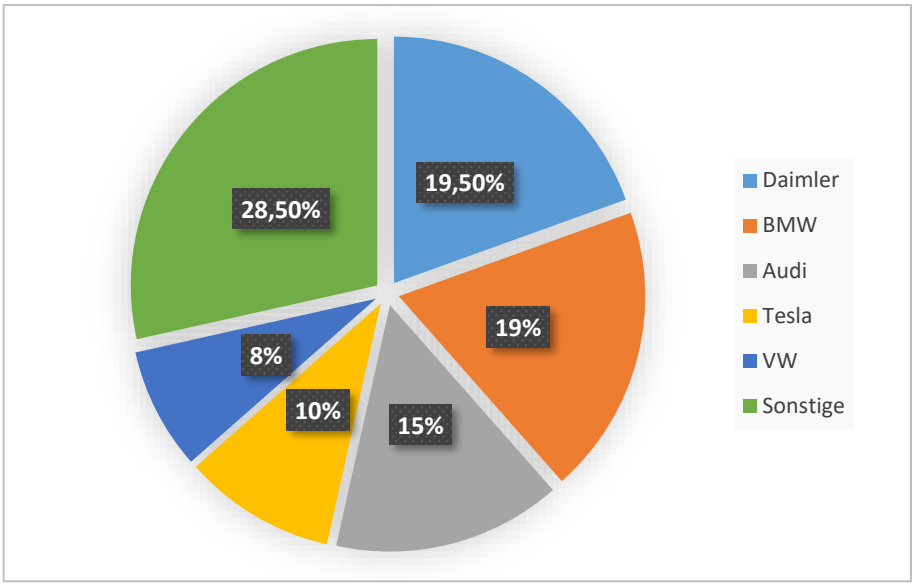


Abbildung 3-4: Welchem Hersteller werden der Bau autonomer Fahrzeuge zugetraut [29]

Der wichtigste Schritt bis zur Verbreitung und Akzeptanz von autonomen Fahrzeugen, oder solchen mit hochautomatisierten Fahrfunktionen ist eine eindeutige Rechtsgrundlage. Hier ist die Politik gefragt, rechtliche und ethische Fragen eindeutig zu klären, um dem Nutzer ein sicheres Gefühl zu vermitteln. Das Vertrauen in die technischen Neuerungen scheint gegeben zu sein, sobald ein zusätzlicher Komfortgewinn offensichtlich ist.

Weitere Untersuchungen zeigen ein sehr heterogenes Bild der Ergebnisse. Eine Studie der Firma „Goodyear“ zeigt, dass Kunden in Deutschland dem autonomen Fahren mehrheitlich skeptisch gegenüberstehen und führt dies auf den Wunsch nach Fahrspaß zurück. Dieser überwiegt gegenüber dem Zugewinn an Zeit [30]. Somit wird deutlich, dass ein aktuelles Thema, dessen Serieneinführung in der Zukunft liegt, schlecht durch Befragungen in Form von erdachten Szenarien erfasst werden können. Bedingt durch die geringen Kenntnisse zur Thematik „autonomes Fahren“ und eine uneindeutige Rechtslage schwanken die Ergebnisse der Befragungen stark, wenn auch eine allgemeine Tendenz für eine Akzeptanz von autonomen Fahrzeugen erkennbar ist.

4 Konzepterstellung

Das Wort Konzept stammt aus dem Lateinischen und bedeutet „Entwurf; erste Fassung; grober Plan“ [31]. Ein Konzept stellt somit als Entwurf die Grundsäule eines Projektes dar. Das nachfolgende Kapitel beschreibt die Konzepterstellung für das Projekt „Demonstrator für autonomes Fahren“. Dabei wird auf die zeitliche Planung eingegangen und eine grobe Einteilung der Projektphasen, anhand von Meilensteinen, vorgenommen. Anschließend wird die Recherche zu den Hauptkomponenten eines realen Demonstrators offengelegt und der Entscheidungsprozess für einzelne Hardware- und Softwarebausteine erläutert. Im Mittelpunkt stehen hierbei die Auswahl eines geeigneten Basisfahrzeuges und die Wahl der Sensoren, da diese Entscheidungen reale Investitionen nach sich ziehen. Die verwendete Software ist im Gegensatz hierzu vollständig frei verfügbar, da die Auswahl sich auf Produkte mit Open Source Lizensierung beschränkt.

4.1 Projektplan

Der Erfolg eines Projektes wird zu einem wesentlichen Teil von einer gründlichen Planung bestimmt. Neben einer Budgetplanung (diese wird in Kapitel 8 erläutert) ist vor allem die zeitliche Planung von entscheidender Bedeutung. Die vorliegende Thesis bildet insgesamt betrachtet lediglich den Start für das Projekt „Demonstrator für autonomes Fahren“. Da die Rahmenbedingungen einer Masterthesis mit sechs Monaten im Vorfeld gegeben sind, bezieht sich die hier gezeigte Planung auf die Zeit von Start der Thesis bis zum Abgabezeitpunkt.

Die Software „MS Project“ bietet als Teil der Office-Suite von Microsoft die Möglichkeit, ein Projekt mit allen notwendigen Zwischenschritten im Detail zu planen. Abbildung 4-1 zeigt einen Ausschnitt des angefertigten Projektplanes in Form eines Gantt-Diagrammes. Es sind die einzelnen Prozessschritte zu sehen, beginnend mit der Hardwaredefinition und die dazugehörigen Zeitabschnitte. Im unteren Teil des Bildes sind die sogenannten Meilensteine aufgeführt. Die wichtigsten Punkte sind hierbei, die Lieferung der wichtigsten Hardwarekomponenten und deren Inbetriebnahme.

Auf der Grundlage dieses Diagrammes und der detaillierteren Unterpunkte kann stets der Überblick über einzelne Projektbausteine und offene Punkte behauptet werden. Die Unterpunkte, auch Vorgänge genannt, sind außerdem der Leitfaden für den Aufbau dieser schriftlichen Arbeit.

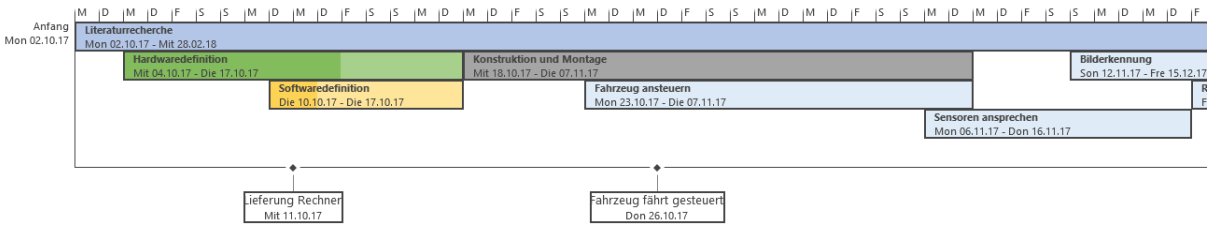


Abbildung 4-1: Gantt-Diagramm Projekt Masterthesis

4.2 Fahrzeug

Die Auswahl eines geeigneten Basisfahrzeuges ist für die Ausrichtung des Projektes von zentraler Bedeutung. Für einen Demonstrator im Bereich „autonomes Fahren“ sind eine Vielzahl unterschiedlicher Fahrzeuge als Basis vorstellbar.

Zu diesen zählen:

- RC-Autos in verschiedenen Maßstäben
- Golfmobile
- Seniorenmobile
- Gokarts

Bei der Entscheidungsfindung sind mehrere Kriterien bedeutend. Zum einen spielt die Größe des Fahrzeuges eine Rolle. Große Abmessungen machen eine Lagerung schwierig, wohingegen sehr kleine Abmessungen eine Unterbringung der weiteren Hardware erschweren. Des Weiteren muss die Automatisierbarkeit der einzelnen Steuerungskomponenten möglich sein. Golfkurs, Seniorenmobile und Gokarts besitzen zumeist eine mechanische Lenkung. Dies zieht eine aufwendige Umrüstung nach sich, die den Rahmen dieser Arbeit übersteigen würde. Gleichzeitig sind die Kosten für RC-Autos um ein Vielfaches geringer als für die anderen Fahrzeuge und sind zudem einfach verfügbar. Die Vorgaben für dieses Projekt, in Form eines Projektsteckbriefes, sahen keine Personenbeförderung und keine Straßenzulassung vor. Daher und aufgrund der oben genannten Punkte wurde ein RC-Auto als Basis gewählt.

RC steht hierbei für „radio control“. Dies bezeichnet im Englischen eine Funksteuerung [32]. Am Markt gibt es ferngesteuerte Autos mit unterschiedlichen Antriebskonzepten. Es finden sich sowohl klassische Verbrenner als auch elektromotorische angetriebene. Für dieses Projekt überwiegen die Vorteile der Elektromotoren. So kommen eine einfache Ansteuerung durch einen Motorregler und ein geringerer Wartungsaufwand zu den äußeren Vorteilen, wie die geringere Lärm- und Geruchsbelastung, hinzu.

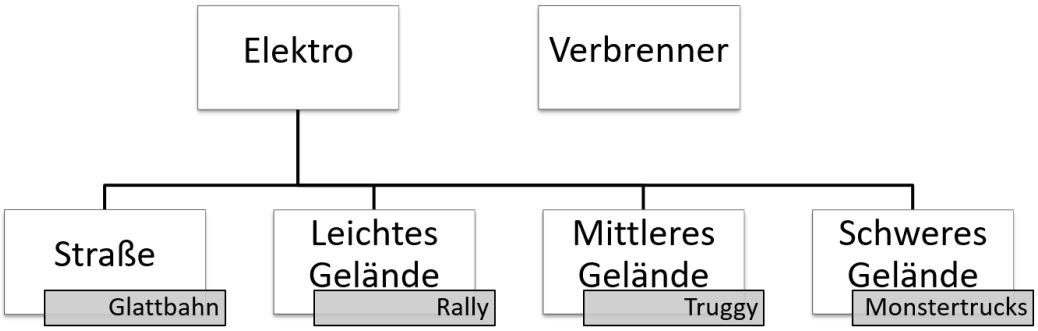


Abbildung 4-2: Übersicht RC-Autos

Abbildung 4-2 zeigt eine Übersicht der Klassen, der am Markt verfügbaren RC-Autos. Aufgrund des geplanten Einsatzes des Demonstrators in einer überdachten und ebenerdigen Umgebung bietet sich ein Glattbahn-Fahrzeug als Basis an. Die Rad-Reifenkombination dieser Klasse erlaubt einen Indoor-Einsatz und ist zudem von den Proportionen am ehesten mit einem realen PKW zu vergleichen.

Ein weiteres Unterscheidungsmerkmal bei RC-Autos liegt in der Art der Elektromotoren. Es wird zwischen „brushed"- und „brushless"-Motoren unterschieden. Modelle mit einem „brushed"-Antrieb besitzen eine Gleichstromversorgung und einen Permanentmagneten. Zur Umpolung wird ein Kommutator mit Kohlebürsten verwendet. Dahingegen kommt ein „brushless"-Motor ohne Kohlebürsten aus. Die Umpolung wird hierbei durch ein elektrisches Drehfeld erzeugt, welches durch einen Regler gesteuert wird. Meist wird im Modellbau ein dreipoliger Aufbau verwendet. Dabei werden jeweils zwei Spulenwicklungen mit positiver Gleichspannung belegt und erzeugen ein Magnetfeld. Durch einen Permanentmagneten wird nun eine Rotationsbewegung erzeugt. Es gibt sowohl einen innen- als auch einen außendrehenden Aufbau [33].

In Tabelle 4-1 werden die Vor- und Nachteile der beiden Konzepte dargestellt. Daraus geht hervor, dass den Vorteilen des „brushless"-Antriebes lediglich die, im Vergleich, höheren Kosten und die komplizierte Regelung entgegenstehen. Da RC-Auto Motoren in der Regel mit einem integrierten Regler ausgeliefert werden, entfällt dieser Nachteil im vorliegenden Fall. Somit wurde insbesondere aufgrund der konzeptionellen Vorteile der „brushless"-Antrieb gewählt. Ohne den mechanischen Kontakt der Kohlebürsten zur Umpolung arbeitet dieser beinahe verschleißfrei und somit auch wartungsfrei. Dies sorgt im längerfristigen Betrieb für ein problemloseres Verhalten.

Tabelle 4-1: Vergleich Elektromotoren

	Brushed	**Brushless**
+	• Günstig • Einfache Wartung • Flüssige Schalteffekte • Einfache Steuerung	• Geringer Wartungsaufwand • Geringe Wärmentwicklung • Hohe Effizienz • Meist vollständig abgedichtet
-	• Mechanische Abnutzung • Wärmeentwicklung • Anfällig gegenüber Nässe	• Hoher Regelaufwand • höhere Anschaffungskosten

Aus den genannten Gründen wurde als Basisfahrzeug ein 1:10 Glattbahn Modell der Marke „Carson" angeschafft. Mit seinem „brushless"-Motor und Allradantrieb erreicht dieses Fahrzeug hohe Leistungswerte und lässt über seine Motorregelung („ESC") eine genaue Antriebskalibrierung zu [35]. Außerdem lassen sich der „ESC" und der Lenkservo über PWM-Signale ansprechen (siehe Kapitel 5.2). Abbildung 4-3 zeigt das Fahrgestell des Modells. An der Front und im Heckbereich lassen sich Aufnahmen für eine

Kunststoffkarosserie erkennen. Diese können für die Unterbringung der Sensorik und weiteren Hardwarekomponenten genutzt werden.

Abbildung 4-3: Basisfahrzeug Carson X10E [34]

4.3 Sensoren

Die Sensorik stellt das Fundament für alle Funktionen dar. Aufgrund der aufgezeichneten Daten werden Entscheidungen gefällt und Fahrwege geplant. Im Idealfall wird eine 360° Rundumsicht geboten, durch die alle möglichen Hindernisse, wie zum Beispiel andere Verkehrsteilnehmer, und Vorgaben, wie Geschwindigkeitsbegrenzungen im Umfeld des Fahrzeuges erkannt werden. Diese „Sicht“ muss so angepasst sein, dass für alle Geschwindigkeitsbereiche jederzeit eine ausreichende Erkennung gewährleistet werden kann, um Kollisionen vermeiden zu können.

Es gibt zurzeit keinen Sensor, der für alle denkbaren Szenarien ausreichend Daten zur Verfügung stellen kann. Vor allem die Reichweite der Erkennung unterscheidet die am Markt verfügbaren Sensoren. So werden bei höheren Geschwindigkeiten hauptsächlich Daten aus einer größeren Entfernung vor dem Fahrzeug benötigt, um mögliche Hindernisse rechtzeitig zu erkennen und darauf reagieren zu können. Für Parkmanöver oder in einer Fahrsituation in einer städtischen Umgebung ist, aufgrund der Vielzahl an Hindernissen, eine genaue Kenntnis des Nahfeldes rund um das Fahrzeug notwendig. Bei der Auswahl der Sensoren für die vorliegende Arbeit wurden mehrere Kriterien beachtet: Zum einen muss die Sensorik auf die geplanten Fahrszenarien und die Größe des Fahrzeuges angepasst sein. Das heißt, dass die Reichweite der Umfelderkennung nicht von großer Bedeutung ist, da das Fahrzeug im Maßstab 1:10 nur einen geringen Geschwindigkeitsbereich abdecken kann. Dieser Bereich ist zusätzlich durch das gewählte Fahrszenario auf niedrige Geschwindigkeiten begrenzt. Insbesondere eine gute Erkennung des näheren Umfeldes ist hierbei jedoch entscheidend. Für die vorgesehenen Manöver ist weiterhin eine Bilderkennung vorzusehen, um die Aufgabe der Spur- und Objekterkennung zu bewältigen. Für einen Einsatz auf einem Demonstrator ist außerdem die Größe der Sensoren zu beachten, damit keine Überladung des Basisfahrzeuges erfolgt. Außerdem muss, in Hinblick auf ein geringes Gesamtgewicht, eine Energieversorgung mit leichten Akkumulatoren sichergestellt werden können.

Zum anderen muss bei der Auswahl eine Kosten-Nutzen-Rechnung aufgestellt werden, da ein vorgegebenes Projektbudget eingehalten werden soll.

Die nachfolgenden Unterkapitel geben einen Überblick über die in Betracht gezogenen Sensoren zur Bilderkennung und der Abstandsmessung/ Umfelderkennung. Dabei wird kein Anspruch auf eine vollständige Auflistung aller möglichen technischen Systeme erhoben. Außerdem werden gesetzliche Vorgaben und Umweltvorschriften nicht weiter beachtet, da diese nur für einen Serieneinsatz maßgeblich sind.

4.3.1 Bildverarbeitung

Die verarbeitenden Algorithmen der Objekterkennung stellen einen wesentlichen Anteil dieser Arbeit dar. Dazu muss ein Bild der Fahrzeugumgebung in Form eines Videos aufgezeichnet und weitergeleitet werden. Am Markt existieren zwei unterschiedliche Systeme, die auf der gleichen technischen Basis stehen. Es werden Mono- und Stereokameras unterschieden.

Beide Systeme basieren auf demselben technischen Prinzip. Über ein Linsensystem wird einfallendes Licht der Umgebung auf einem Halbleiterchip in elektrische Impulse umgewandelt, die dann in Farbinformationen der einzelnen Bildpunkte umgerechnet werden [36]. Monokameras setzen auf einen Chip, wohingegen Stereokameras zwei Bildsensoren besitzen. Diese sind nebeneinander in einem Abstand angeordnet, der mit den menschlicheren Augen vergleichbar ist, und liefern über diese Konfiguration zusätzlich Disparitätsinformationen. In Verbindung mit der zeitlichen Veränderung des Bildes (optischer Fluss) können die Entfernung und die Größe von Objekten im Bildausschnitt bestimmt werden [37]. Diese Berechnung von Zusatzinformationen stellt den größten Vorteil einer Stereokamera dar. Demgegenüber stehen der deutlich höhere Platzbedarf und höhere Anschaffungskosten im Vergleich zu Monokameras. Dazu kommt ein erhöhter Rechenaufwand aufgrund der doppelten Datenmenge von Stereobildern. Des Weiteren gibt es aktuelle Forschungen, die Möglichkeiten der Entfernungsmessung durch den optischen Fluss auch bei Einzelbildkameras aufzeigen [38].

Für das vorliegende Anwendungsszenario ist daher eine Monokamera die geeignete Wahl. So kann testweise eine einfache USB-Webcam (Logitech Webcam Pro 9000) verwendet werden. Diese zeichnet sich durch ihr geringes Gewicht bei relativ kompakten Abmessungen aus. Außerdem kann sie per Standard-USB Verbindung in ein Rechnersystem eingebunden werden.

4.3.2 Abstandsmessung/ Umfelderkennung

Ein weiterer wichtiger Faktor zur Umsetzung des Konzeptes ist die korrekte Umfelderkennung und Abstandsmessung zu potentiellen Hindernissen. Hierbei sollen einerseits Kollisionen vermieden und andererseits die Umwelt vermessen werden, um beispielsweise Parklücken zu erkennen. Am Markt haben sich verschiedene Systeme auf elektromagnetischer, optischer oder akustischer Basis etabliert. Bei der Auswahl sind besonders die Reichweite, die Messgenauigkeit, sowie die Verfügbarkeit bzw. der Preis

ausschlaggebend. Nachfolgend werden die gängigsten Sensorarten in ihren Besonderheiten aufgezählt und anschließend verglichen.

Radar (Radio Detection and Ranging) ist ein System von gebündelten Wellen in einem festgelegten Frequenzbereich von 24-77 GHz. Es gibt eine Sender- und Empfängereinheit und über die Laufzeitdifferenz der gesendeten Signale kann die Entfernung zu einem Objekt bestimmt werden. Zusätzlich kann nach Christian Doppler ein mathematischer Zusammenhang zwischen Frequenzänderung Relativgeschwindigkeit hergestellt werden:

$$f_{Doppler} = \frac{-2\dot{r}}{\lambda} \quad (4.1)$$

Somit macht Radar neben einer Entfernungsmessung auch eine Messung von Geschwindigkeiten erkannter Objekte möglich [16].

Lidar (Light Detection and Ranging) ist eine optische Methode zur Distanzbestimmung und Umfelderkennung, die ebenso mit der Laufzeitmessung eines ausgesendeten Impulses arbeitet. Diese ist proportional zur Entfernung und kann über folgende Formel bestimmt werden:

$$d = \frac{c_0 * t}{2} \quad (4.2)$$

Systeme auf Lidarbasis können zur Abstandsmessung wie Radar eingesetzt werden, besitzen jedoch zusätzlich die Fähigkeit mehrere Objekte gleichzeitig durch entsprechende Algorithmen zu erkennen [16]. Weiterhin ist es möglich eine komplette Umgebungskarte durch 360° Lidarsensoren zu erstellen. Dafür werden hauptsächlich mechanisch drehbare Systeme verwendet, die eine Rundumsicht simulieren. Dies wird durch die hohe Geschwindigkeit der Lichtimpulse ermöglicht. Durch Auswertung der Lichtintensität ist es sogar möglich beliebige Objekte zu klassifizieren [39].

Ultraschall ist eine akustische Methode zur Abstandsmessung. Im Frequenzbereich von 40 bis 50kHz wird ein Impulssignal von einem Piezokristall erzeugt und mittels Laufzeitmessung dieses Signals die Entfernung zu einem Objekt bestimmt. Dabei erfolgt eine Berechnung der Distanz nach der gleichen Formel wie bei Lidargestützten Systemen, jedoch unter Verwendung der Schallgeschwindigkeit [40]:

$$d = \frac{v_{Schall} * t}{2} \quad (4.3)$$

Tabelle 4-2 bietet eine Übersicht der Vor- und Nachteile der einzelnen Technologien. Dabei wird deutlich, dass für die Anwendung in einem Projekt im Maßstab 1:10 und bei niedrigen Geschwindigkeiten, die Reichweite von Ultraschallsensoren ausreichend ist. Dazu kommen niedrige Kosten und eine problemlose Integrierbarkeit in eine Softwareumgebung. Aus diesem Grund werden im Demonstrator Ultraschallsensoren verwendet. Die zusätzlichen Möglichkeiten von Radar und Lidar, wie erhöhte Reichweite

und Rundumsicht, werden für die angestrebten Ziele nicht benötigt, ließen sich aber bei Bedarf auf dem Fahrzeug nachrüsten.

Tabelle 4-2: Vergleich Sensoren

	Lidar	Radar	Ultraschall
+	• Hohe Reichweite (über 100m) • Hohe Auflösung • 3d Scan möglich	• Hohe Reichweite (250m) • Direkte Geschwindigkeitsmessung	• Günstig • Leicht verfügbar • Rundumsicht im Nahbereich möglich
-	• Sehr hohe Kosten • Verfügbarkeit • Mechanische Anfälligkeit	• Hohe Kosten • Verfügbarkeit • Interferenzen möglich	• Interferenzen möglich • Geringe Reichweite (ca. 3m – 10m)

4.4 Weitere Hardwarekomponenten

Zur Verarbeitung aller Sensorsignale und zur Berechnung der benötigten Daten ist ein zentrales Element ein leistungsfähiger Entwicklungsrechner. Bei der Auswahl einer geeigneten Hardware sind mehrere Kriterien zu beachten, die aufgrund der besonderen Projekteigenschaften entstehen. Zum einen müssen die Abmessungen möglichst kompakt und zum anderen darf die Leistungsanforderung an die Stromversorgung nicht zu hoch sein. Beides ist den eingeschränkten Platzverhältnissen auf einem Modellfahrzeug geschuldet. Im Konflikt dazu steht die Anforderung an eine hohe Rechenleistung, die besonders von der Bildverarbeitung gestellt wird. Diese muss Bilddaten mit einer hohen Wiederholungsrate verarbeiten können, um eine schnelle Erkennung auch bei Bewegung des Fahrzeuges zuzulassen.

Die Anforderungen an die Größe und die Energieaufnahme können nur von einem Einplatinencomputer erfüllt werden. Diese Rechner, wie der „Raspberry Pi“, sind sehr kompakt, besitzen jedoch keine große Rechenleistung. Eine Ausnahme bildet der „NVIDIA Jetson TX2“ als Entwicklerausführung. Mit insgesamt sechs Prozessorkernen und acht Gigabyte Arbeitsspeicher eignet er sich sowohl für die parallele Signalverarbeitung und komplexe Berechnungen, als auch als Entwicklungsumgebung zur Programmierung der geplanten Funktionen. Dazu kommt die Ähnlichkeit der Rechnerarchitektur zu der, die heutzutage schon in vielen Automobilen der großen OEMs verbaut wird. Dort werden NVIDIAs Lösungen zum maschinellen Lernen und zur Bildverarbeitung im Bereich des autonomen Fahrens genutzt [41]. Mit einer maximalen Leistungsaufnahme von ca. 15 Watt und einer möglichen Versorgungsspannung von 9-15 Volt eignet sich der Rechner somit hervorragend für den mobilen Einsatz [42].

Als weitere Hardware werden ein Akkumulator mit einer Spannung von 11,1V zur Energieversorgung und ein „Arduino UNO“-Microcontrollerboard verbaut. Dieses Board dient als Kommunikationsschnittstelle zwischen Entwicklungsrechner und Fahrzeugbasis. Außerdem eignet es sich zur Einbindung der

Ultraschallsensoren. Ein „Arduino“ zeichnet sich durch seine geringe Energieaufnahme und niedrige Anschaffungskosten aus.

Für eine Orientierung des Fahrzeuges ist nicht nur das Wissen über die aktuelle Umgebung notwendig, sondern auch die Position des Fahrzeuges in dieser Umgebung. Zu diesem Zweck wurde eine „Inertiale Messeinheit“ der Firma Robert Bosch GmbH (BNO55) verbaut. Diese „IMU“ ist auf einem Breakoutboard der Firma Adafruit Industries verbaut und lässt sich damit einfach über die GPIO-Headerpins mit dem Jetson-Rechner verbinden. Der Sensor liefert die lineare Beschleunigung, Winkelgeschwindigkeit und magnetische Orientierung in alle drei Raumrichtungen als Rohdaten. Somit sind keine weiteren Umrechnungen notwendig [43]. Zur Aufzeichnung von Geschwindigkeitsdaten steht ein einfacher Hallsensor zur Verfügung. Nach dem Halleffekt induziert ein Magnetfeld an einem stromdurchflossenen Leiter eine Hallspannung. In Abbildung 4-4 ist diese Spannung als „U20“ eingezeichnet [44].

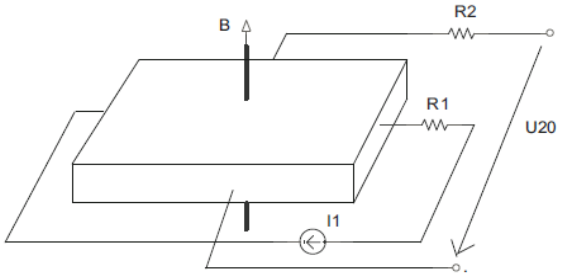


Abbildung 4-4: Prinzip Hall-Effekt [43]

Dieses Phänomen kann genutzt werden, um die Anwesenheit eines Magneten zu detektieren. Wird nun ein Magnet an einem rotierenden Bauteil angebracht und den Hallsensor im Gegensatz dazu bewegungsfest, kann durch den Ausschlag der Hallspannung eine Umdrehung festgestellt werden. Für diese Arbeit wurde ein Hallsensor fest am hinteren Radträger montiert und ein Magnet am Felgeninnenbett. Über den bekannten Umfang des Rades kann nun leicht die Geschwindigkeit des Rades in Abhängigkeit der Umdrehungen pro Minute „U“ bzw. mit der Umlaufzeit „T“ über folgende Formel berechnet werden:

$$v_{Fahrzeug} = U * n_{Rad} = \frac{U}{T} \qquad (4.4)$$

Eine hochgenaue Kenntnis der Geschwindigkeit ist für den vorgesehenen Anwendungszweck nicht erforderlich, da auch eine genaue Positionsbestimmung nicht vorgesehen ist. Daher sind die gezeigte Abschätzung der Geschwindigkeit und eine daraus resultierende Berechnung des zurückgelegten Weges ausreichend. Der Fehler durch Reifenschlupf ist in den geplanten niedrigen Geschwindigkeitsbereichen zu vernachlässigen.

4.5 Software

Die verwendete Software bildet das eigentliche Kernstück des Projektes. Sie soll das Auslesen der Sensordaten, das Verarbeiten von Bildinformationen und das Steuern aller Funktionen ermöglichen.

Um aus den rohen Sensordaten verwertbare Informationen für einen Entscheidungsalgorithmus zu erhalten, ist die Kombination verschiedener Softwarekomponenten notwendig. Diese sollen im Folgenden kurz vorgestellt und ihr Anwendungszweck erläutert werden. Die selbst programmierten Skripte werden in den Kapiteln 0 und 7 beschrieben.

4.5.1 Middleware

Als Middleware wird eine Softwarearchitekturkomponente bezeichnet, die die Verbindung zwischen Hardware- und Softwareebene herstellt. Dies ermöglicht die Basiskommunikation zwischen Betriebssystem und Anwendersoftware [45]. Für ein selbstfahrendes Fahrzeug eignet sich besonders „ROS“. Das „Robot Operating System“ hat sich inzwischen als De-facto-Standard in der Entwicklung autonomer Vehikel zu Lande, zu Wasser und in der Luft etabliert. So verwendet zum Beispiel auch die Firma Robert Bosch GmbH diese Open Source-Software zur Entwicklung vieler ihrer Robotik Anwendungen [46].

Diese Middleware bietet mehrere Vorteile, die für die vorliegende Arbeit ausschlaggebend sind. Auf der ersten Stufe gelingt durch eine breite Treiberunterstützung das Einbinden von Hardware. Weiterhin ermöglicht es als Kommunikationsebene den einfachen Austausch von Daten zwischen einzelnen Skripten. Diese können parallel laufen und garantieren so eine Echtzeitfähigkeit [47].

4.5.2 Bildverarbeitung

Das zentrale Element dieser Arbeit stellt die Bildverarbeitung dar. Die Bilddaten der Kamera müssen eingelesen und so weiterverarbeitet werden, dass Objekte und Fahrspuren erkannt werden. Zu diesem Zweck bedarf es der Anwendung einer Reihe von Algorithmen und mathematischen Operationen. Ein umfangreiches Tool um maschinelle Sehen umzusetzen ist das Open Source-Framework „OpenCV“. Es bietet auf unterschiedlichen Plattformen die Möglichkeit Bilddaten einzulesen und in Echtzeit zu verarbeiten. Im herkömmlichen Sinn ist „OpenCV“ kein Anwenderprogramm mit einer Benutzeroberfläche. Es ermöglicht lediglich die Nutzung von spezifischen Befehlen und Algorithmen der Bildverarbeitung in Skripten der Programmiersprachen „Java“, „C++“ oder „Python“ [48]. Die Fahrspurerkennung und Stoppschilderkennung dieses Projektes wurde, aufgrund der guten Lesbarkeit der Sprache, in Python implementiert.

5 Konstruktion und Vermessung

Konstruieren bedeutet der Wortherkunft nach etwas zusammenbauen [7]. Die technische Konstruktion geht weit darüber hinaus und bezeichnet im heutigen Kontext die Planung eines technischen Gegenstandes bis zu einem fertigungsfähigen Modell und aller dazugehörigen Fertigungsunterlagen [49]. Das folgende Kapitel beschreibt die technische Umsetzung des Konzeptes für einen „Demonstrator für autonomes Fahren“. Dieses beginnt mit der Planung der Platzverhältnisse und der Unterbringung aller notwendigen Komponenten auf einem gegebenen Basisfahrzeug. Darüber hinaus wird ein CAD-Modell für eine genaue Übersicht erstellt. Aus dieser Planung heraus folgt die Umsetzung von Teilschritten des Projektes in einer Testumgebung. Dieses beginnt bei der Ansteuerung der Servomotoren des Basisfahrzeuges und endet mit der Vermessung von Sensoren, um deren Funktionalität in einem realen Umfeld in Hinblick auf die Messgenauigkeit zu überprüfen. Schließlich wird der Aufbau eines funktionsfähigen Testfahrzeuges auf Basis der Konstruktion und der Einzelkomponenten beschrieben. Dieses dient der Anpassung aller Parameter an die gegebenen realen Umstände.

5.1 Packaging

Für eine genaue Abschätzung ist eine Kenntnis der Abmesungen alle Komponenten unverzichtbar. Aufgrund fehlender Daten seitens der Hersteller wurde, vor der Bestellung der Elemente, eine erste Größenabschätzung auf Basis der vorhandenen Maße getroffen. Für die Befestigung der benötigten Hardware auf dem Basisfahrzeug bietet es sich an eine zweite Ebene in Form einer Grundplatte oberhalb der Fahrzeugkomponenten anzubringen. Abbildung 5-1 zeigt in orange gefärbt die maximal möglichen Abmessungen ohne über das Fahrgestell hinaus zu reichen.

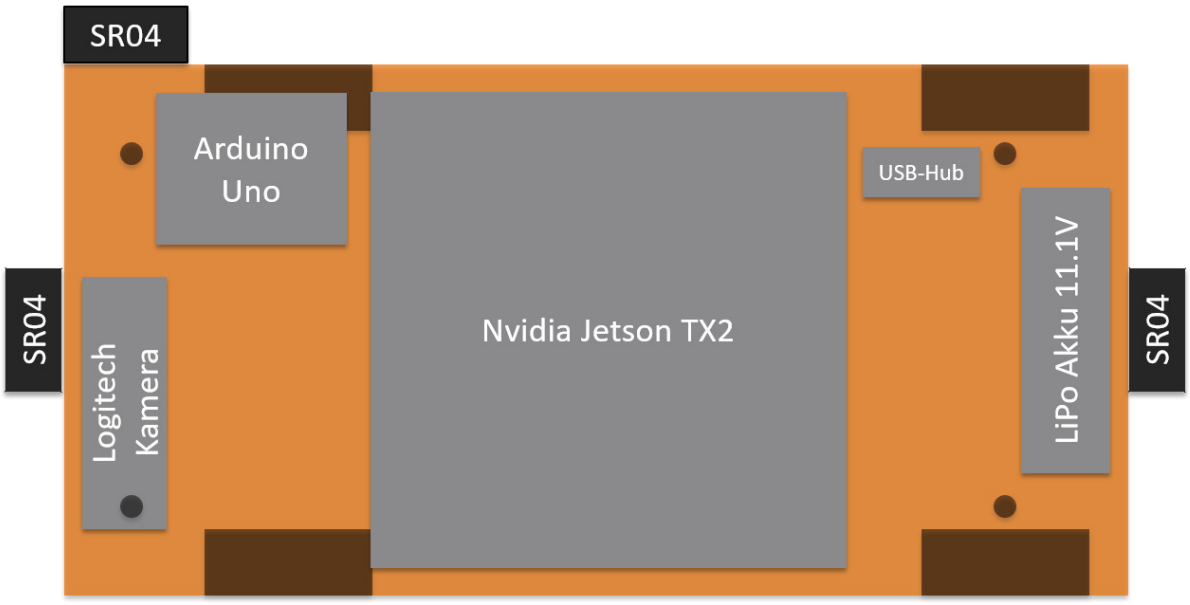


Abbildung 5-1: 2D-Planung Platzverhältnisse

Um alle Komponenten unterbringen zu können wurde bei der Auswahl dieser auf kompakte Abmaße und ein geringes Gewicht geachtet. Es zeigt sich, dass alle Komponenten Platz finden und auch genug

Raum für die Verkabelung zur Verfügung steht. Bei der Verteilung wurde weiterhin auf eine gleichmäßige Gewichtsverteilung geachtet. Die Kamera wird für eine bessere Übersicht auf einer separaten Plattform etwas erhöht montiert und schafft so weiteren Platz auf der Grundplatte.

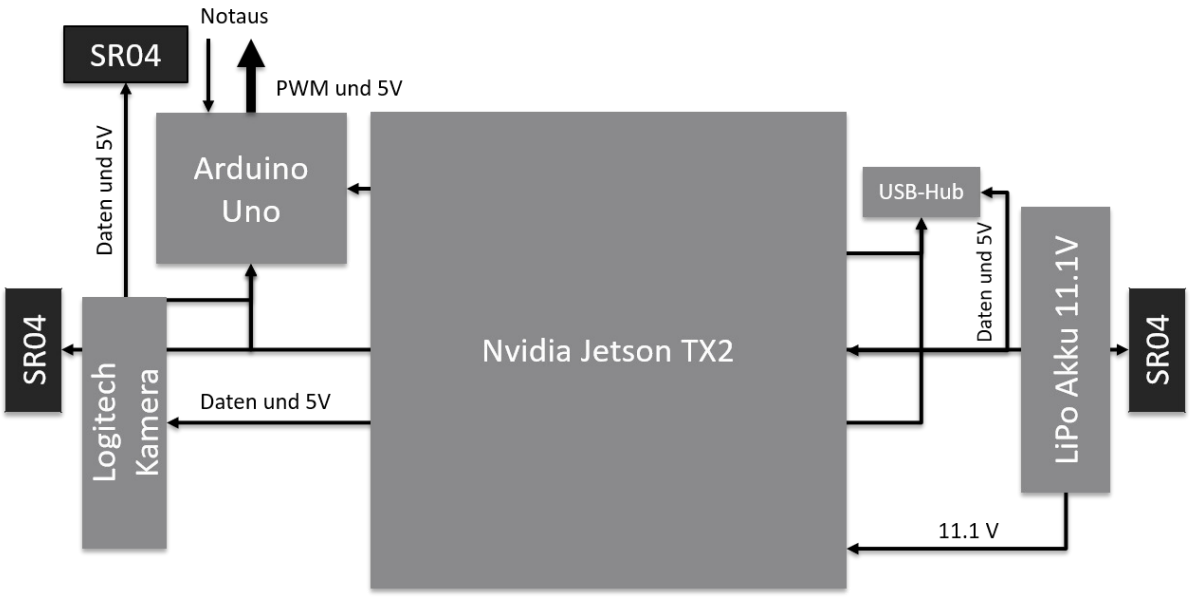


Abbildung 5-2: Verkabelung der Hardware

Auf Grundlage der Größenplanung wurde mit Abbildung 5-2 ein Stromlauf- und Datenplan erstellt. Die Stromversorgung der Hardwareebene basiert auf dem Lithium Polymer Akkumulator, der den Jetson-Rechner mit einer Spannung von 11,1 Volt versorgt. Über den USB-Hub werden allen weiteren Komponenten mit Energie versorgt. Dieser dient außerdem zum Datenaustausch mit den Sensoren. Die Ultraschallsensoren (hier als SR04 bezeichnet) sind über vier Pins mit dem Arduino verbunden. So senden sie Daten und werden mit Spannung versorgt. Eine Besonderheit stellt der Ausgang des Arduinos zum Fahrgestell dar. Hier liegen 5 Volt für den Lenkservo an und es werden die PWM Signale zur Steuerung weitergeleitet. Außerdem können Notaussignale empfangen werden, die von der Originalfunkfernbedienung ausgesendet werden können.

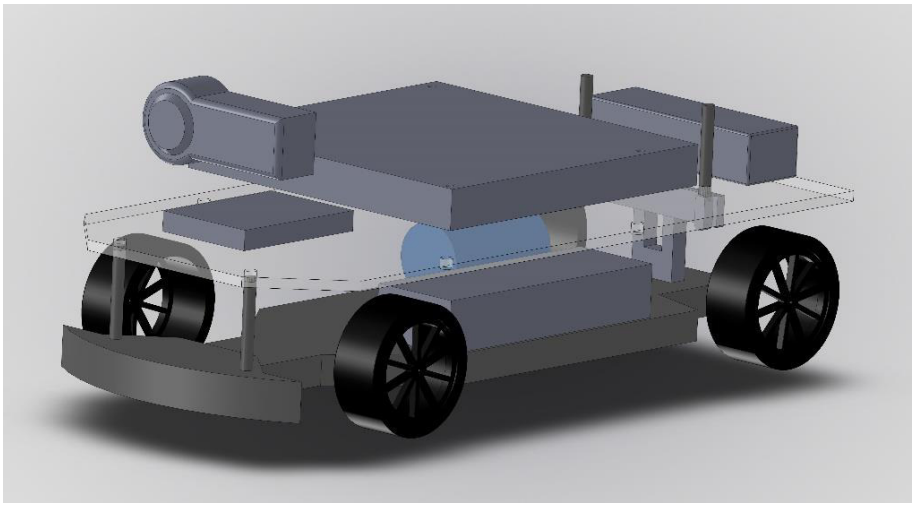


Abbildung 5-3: CAD-Modell des Demonstrators

Nach der Beschaffung aller Komponenten erfolgte die Vermessung, vornehmlich des Basisfahrzeuges, um eine genaue Planung vornehmen zu können. Auf Basis dieser Daten entstand ein dreidimensionales CAD-Modell des geplanten Demonstrators (siehe Abbildung 5-3). Für die Konstruktion wurde die Software „SolidWorks“ verwendet. Dabei wurde bewusst auf Details verzichtet, da hierbei keine Grundlage für eine Fertigung entstehen soll, sondern lediglich eine 3D-Visualisierung der Planung. Die 3D Ansicht bestätigt hierbei die vorangegangene Planung im Zweidimensionalen. Eine Unterbringung aller Komponenten ist realisierbar.

5.2 Ansteuerung RC-Auto

Um ein RC-Auto eigenständig fahren zu lassen, ist es in einem ersten Schritt notwendig, die Kontrolle über die Steuerung des Fahrzeuges zu erlangen. Dies beinhaltet die Lenkung und das Motorsteuergerät. Eine mechanische Bremsanlage ist nicht vorhanden. Das Basisfahrzeug empfängt in seiner Originalkonfiguration Steuerungsbefehle von einer Funkfernbedienung, mit einer Frequenz von 2,4 GHz. Diese werden von einem Funkempfänger in Signale zur Ansteuerung des Servomotors der Lenkung und des „ESC“ verwendet. An dieser Stelle bietet sich die Möglichkeit einzugreifen. Die Funksignale werden durch selbstgenerierte Steuerungssignale ersetzt. Da die Datenblätter des Fahrzeuges und des Motors weder Informationen über die Art noch die Form der Signale liefern mussten diese vermessen werden. Zu diesem Zweck wurde ein digitales Oszilloskop mit den Ausgängen des Funkempfängers verbunden und per USB an einem Computer angeschlossen. Über die Software „Pico Scope 6“ der Firma Pico Technology können nun die Signale in digitaler Form ausgelesen und gespeichert werden.

Abbildung 5-4 zeigt das empfangene Signale bei einer neutralen Gaspedalstellung. Somit liegt das Steuersignal für das „ESC“ vor.

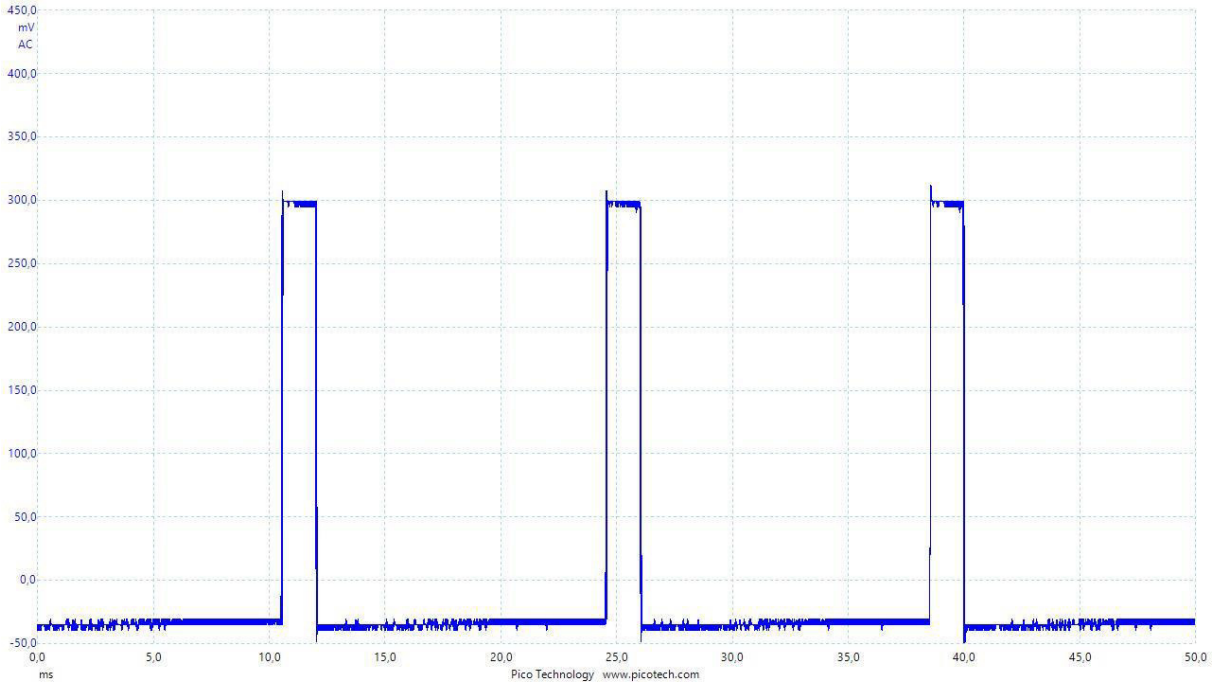


Abbildung 5-4: PWM-Signal, neutrale Gaspedalstellung

Der Signalverlauf entspricht einem PWM-Signal (Pulsweitenmodulationssignal). PWM ist ein weit verbreitetes System zur Steuerung von technischen Anwendungen. Dabei wird ein Rechtecksignal moduliert, das lediglich einen „High"- und einen „Low"-Wert annehmen kann. Durch eine elektronische Schaltung werden Signale so überlagert, dass ein Ausgangsignal in konstanter Frequenz ausgegeben wird. Dabei kann die Dauer des Highpegels variiert werden und dadurch, bzw. durch das Verhältnis der Dauer des Highpegels zu der Gesamtperiodendauer, Informationen übermittelt werden [50].

Das gemessene Signal der Funksteuerung hat bei einer Periodendauer von ca. 15 ms ungefähr 1,5 ms einen Highpegel. Dies entspricht der Neutralstellung der Fernbedienung. Weitere Messungen haben folgende Werte für die unterschiedlichen Gaspedalstellungen ergeben:

Tabelle 5-1: Messungen ESC-Signale

Stellung der Fernbedienung	Dauer Highpegel
Neutral	1500 µs
Voll rückwärts	1000 µs
Voll vorwärts	2000 µs

Für die Steuerung eines Servomotors existiert eine Standard-Arduinobibliothek der Arduino LLC. Dabei wird als Eingabe ein gewünschter Winkel verwendet und die Bibliothek wandelt diesen in ein PWM-Signal um. Diese Funktionalität basiert auf der Annahme, dass alle Servomotoren ein ähnliches PWM-Signal verwenden. Durch Versuche mit verschiedenen Winkeln wurde der Arbeitsbereich des Servomotors des Basisfahrzeugs bestimmt. Dieser operiert in einem Bereich von 70° bis 110°. Somit entsprechen 90° einer Geradeausstellung. Dieses wird als PWM-Signal mit einer Pulsweite von 1500 µs moduliert. Demnach entspricht das Signal für das „ESC" den Standardsignalen für einen Servomotor.

5.3 Vermessung Sensoren

Die theoretische Betrachtung der Ultraschallmesstechnik lässt auf eine ausreichend hohe Genauigkeit für die geforderte Abstandsmessung schließen. Als Sensoren wurde das Modell „HC SR04" von KT-Electronic gewählt. Das Datenblatt gibt einen Messbereich von 2 cm bis 3 m an, bei einer Messgenauigkeit von +/- 3 mm. Der Abstrahlwinkel beträgt hierbei 15° [51].

Um diese Angaben und die Anwendbarkeit auf einem 1:10 Fahrzeug zu überprüfen, wurde ein Testszenario aufgebaut. Dabei gilt es einerseits die Ansteuerung und das Auslesen mithilfe eines Arduino und andererseits die Genauigkeit zu überprüfen. Insbesondere die Abstrahlcharakteristik bei einer bodennahen Montage ist auf eventuell auftretende Interferenzen zu untersuchen. Zusätzlich dazu wird die

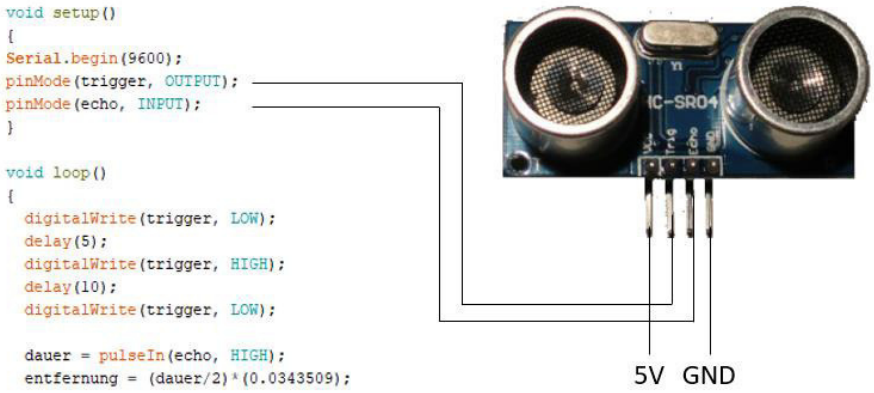


Abbildung 5-5: Ultraschall Pinbelegung und Programmierung

Reproduzierbarkeit der Ergebnisse evaluiert. Für die Tests wurde ein Ultraschallsensor auf dem Frontprallschutz des Basisfahrzeuges montiert und der Abstand zu einer schallharten Wand aus lackierten Beton gemessen. Der Boden der Messumgebung besteht aus demselben Material.

Der Ultraschallsensor besitzt vier Anschlusspins. Die Belegung ist in Abbildung 5-5 zu sehen. Ebenso zeigt diese Abbildung die Umsetzung der Ansteuerung und des Auslesens in einem Programmcode der Arduino IDE. Um ein Signal zu erhalten wird der Trigger erst auf „Low“ und anschließend für 10 ms auf „High“ geschaltet. Im Anschluss kann über ein „High“-Signal des Echo-Pins die Länge der Signallaufzeit ausgelesen werden. Diese Echo Zeit muss halbiert werden, da die Strecke auf dem Hin- und Rückweg jeweils einmal durchlaufen wird. Durch eine Multiplikation mit der Schallgeschwindigkeit wird der Abstand berechnet. Die Schallgeschwindigkeit berechnet sich nach folgender Formel:

$$v_{Schall} = \sqrt{\frac{\kappa RT}{M}}\,[52] \qquad (5.1)$$

Da der Adiabatenexponent „κ“, die molare Masse „M“ und die universelle Gaskonstante „R“ für Luft als konstant zu betrachten sind, ist die Schallgeschwindigkeit nur von der Umgebungstemperatur abhängig. Diese Abhängigkeit wird in der hier angewendeten Berechnung nicht beachtet. Im Gegensatz dazu wird ein fester Wert von 343,5 m/s verwendet. Dies entspricht einer Umgebungstemperatur von 20° C. Bei einer Temperatur von 10° C beträgt dieser Wert ca. 337,5 m/s und hat somit eine Abweichung von ca. 1,9%. Aufgrund der zu erwartenden geringen Temperaturschwankungen in der Testumgebung kann diese Temperaturabhängigkeit in vorliegender Berechnung vernachlässigt werden.

Bei der Durchführung der Messung wurde auf eine exakt orthogonale Ausrichtung zur Wand geachtet. Der Abstand wurde sowohl analog mit einem Maßband gemessen, als auch digital mit einem Laserentfernungsmesser, überprüft (siehe Abbildung 5-6).

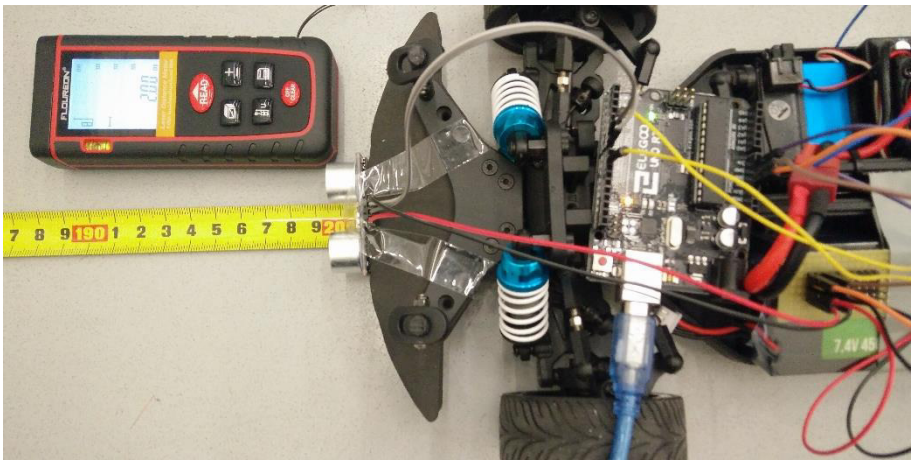


Abbildung 5-6: Testumgebung Ultraschallvermessung

Die Messungen wurden in den verschiedenen Abständen jeweils 15-mal wiederholt. Dabei haben sich folgende Mittelwerte und Standardabweichungen ergeben:

Tabelle 5-2: Messwerte Ultraschallvermessung

Entfernung	Mittelwert [cm]	Standardabweichung [cm]
5 cm	4,9093	0,0561
10 cm	9,51	0,1491
100 cm	98,2173	0,2792
200 cm	198,0287	0,5015
300 cm	303,1727	10,7451

Insgesamt ergibt sich eine Erhöhung der Standardabweichung mit zunehmender Distanz. Dabei ist die Entfernung von 300 cm zu vernachlässigen. Hierbei handelt es sich laut Datenblatt um die Maximaldistanz des Sensors. Die Messgenauigkeit ist insgesamt als ausreichend zu bezeichnen und auch die Reproduzierbarkeit ist gut. Abbildung 5-7 zeigt beispielhaft die Auswertung für einen Abstand von 200 cm in einem Balkendiagramm. Dabei wird für jede Einzelmessung der ermittelte Abstand dargestellt, mit dem Mittelwert als horizontalen Balken. Es lässt sich eine geringe Streuung erkennen.

Für die Anwendung der Ultraschallsensoren zur Hinderniserkennung und Kollisionsvermeidung sind diese Ergebnisse als ausreichend genau zu bezeichnen. Die maximale Geschwindigkeit des Fahrzeuges wird im autonomen Modus auf ca. 1m/s begrenzt. Bei Höchstgeschwindigkeit kann ein Hindernis somit in etwas unter 3s Zeitabstand vor einer Kollision erkannt werden. Diese Zeit ist ausreichend um eine Kollision durch einen Bremseingriff zu vermeiden. Auch die Reproduzierbarkeit ist gut genug, um im Betrieb keine großen Schwankungen erwarten zu müssen. Folglich ist der genannte Sensortyp für das vorliegende Projekt nutzbar.

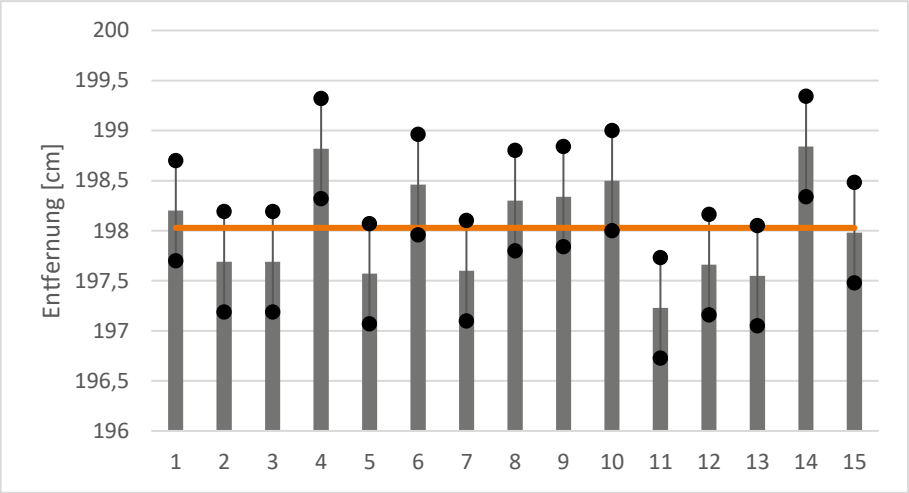


Abbildung 5-7: Messergebnis Ultraschall 200 cm

6 Bildverarbeitung

Für die Entwicklung eines Demonstrators für autonomes Fahren ist die zugrundeliegende Software die wichtigste Systemkomponente. Sie legt die Funktionen fest und damit auch die Fähigkeiten eines Fahrzeuges. Für die vorliegende Arbeit wurde ein Anwendungsszenario definiert, dass sich als „Spurfolgen“ beschreiben lässt. Es soll ein Parcours aus einer Fahrspur, den das Fahrzeug ohne äußere Eingriffe abfahren können soll aufgebaut werden. Dabei sollen Hindernisse erkannt und rechtzeitig vor diesen gestoppt werden. Außerdem soll das Fahrzeug Verkehrszeichen, wie ein Stoppschild, detektieren und entsprechend darauf reagieren.

Um dieses Ziel zu erreichen sind mehrere Schritte und einzelne Softwarekomponenten der Bilderkennung erforderlich. Diese teilen sich in die Spurerkennung und die Schildererkennung auf. Im folgenden Kapitel werden mehrere Ansätze der beiden Bereiche vorgestellt und verglichen.

Die digitale Bildverarbeitung basiert größtenteils auf einer bestimmten Form der Mustererkennung (engl.: „pattern recognition“). Der Mensch erkennt Gegenstände, weil er gelernt hat wie diese, anhand von Umrissen und Farbgebung aussehen. Dieses Verhalten gilt es auf der Maschinenebene anzuwenden. Der Computer muss lernen wie Dinge aussehen, um diese wiederzuerkennen und die Informationen dazu verarbeiten zu können. Dafür wird die Software „OpenCV“ verwendet. Dieses Open Source Softwarepaket beinhaltet bereits eine Reihe von Standardalgorithmen zur Bildverarbeitung und eignet sich damit zum Testen verschiedener Ansätze. Die Umsetzung erfolgt hierbei in Form von einzelnen Skripten in der Programmiersprache „Python“. Diese Einzelskripte können nachfolgend ohne großen Aufwand in eine gemeinsame Umgebung mit der Middleware „ROS“ zusammengefasst und zu einem Fahralgorithmus kombiniert werden.

6.1 Spurerkennung

Die Spurerkennung ist das bedeutendste Element dieser Arbeit und auch essentiell für eine funktionsfähige Querführung der FAS der Automobilhersteller und Zulieferer. Nur bei einer erfolgreichen Erkennung kann auch die Regelung funktionieren. Dabei wird hier aufgrund der vorhandenen Kameradaten ein Ansatz der Bilderkennung gewählt. Dieser Ansatz wird auch in der Serie am häufigsten verwendet. Daneben existieren Systeme auf Basis von Infrarotsensoren in der Fahrzeugfrontstoßstange. Dieses eignet sich jedoch nur für eine Warnung beim Fahrspurwechsel und nicht für eine Querführung [16].

Ziel der zu erstellenden Spurerkennung ist unter anderen die Detektion der Fahrbahnbegrenzungen. Es soll die Information zu der Position des rechten und linken Seitenstreifens der jeweiligen Fahrspur gewonnen werden. Außerdem muss die Position des Fahrzeuges relativ zur erkannten Fahrspur berechnet werden. Ein weiteres Ziel ist die Erkennung von Kurven. Die Information über den Kurvenradius kann genutzt werden, um die Fahrzeuggeschwindigkeit entsprechend anzupassen. Auch die Kenntnis der Kurvenrichtung kann für eine Vorregelung der Querführung verwendet werden.

Um diese Ziele zu erreichen, wurden aus der Literatur zwei Ansätze der Spurerkennung herausgearbeitet und programmtechnisch umgesetzt. Diese Ansätze werden hinsichtlich ihrer Robustheit, also ihrer Erkennungsrate unter verschiedenen Bedingungen und der benötigten Rechenleistung verglichen. Die Rechenleistung spielt bei der Umsetzung eine entscheidende Rolle. Nur wenn die Erkennung schnell und stabil abläuft kann eine Querregelung in verschiedenen Geschwindigkeitsbereichen realisiert werden. Daneben bedeutet ein geringerer Rechenbedarf auch eine geringere Energieaufnahme. Eine gute Energieeffizienz ist für die mobile Anwendung auf einem Demonstrator wichtig, um ausreichende Laufzeiten des Systems, trotz geringer Aufnahmekapazität für einen Energiespeicher zu gewährleisten.

6.1.1 HoughLine Linienerkennung

Die nachfolgend vorgestellte Methode zur Spurerkennung geht auf einen, erstmals von Paul Hough entwickelten, Algorithmus zur Linienerkennung in Bildern zurück. Dabei ist anzumerken, dass der Algorithmus von Hough nur einer von vielen Schritten zur Spurerkennung ist. Jedoch ist dieser Schritt maßgeblich und damit namensgebend für dieses Kapitel. Das gesamte Vorgehen basiert auf der Arbeit von Yue Wang, Eam Khwang Teoh und Dinggang Shen [53], die ihren Ansatz erstmalig im Jahr 2004 veröffentlicht haben. Die Erkennung basiert auf der Annahme, dass Fahrspuren als Linien im Bild detektiert werden können.

Das Vorgehen mit seinen Einzelschritten, vom Einlesen des Kamerabildes bis zur Extraktion der geforderten Spurinformationen, ist Abbildung 6-1 gezeigt. Nachfolgend werden die einzelnen Schritte erläutert und deren Umsetzung anhand von Quelltext in Python aufgezeigt.

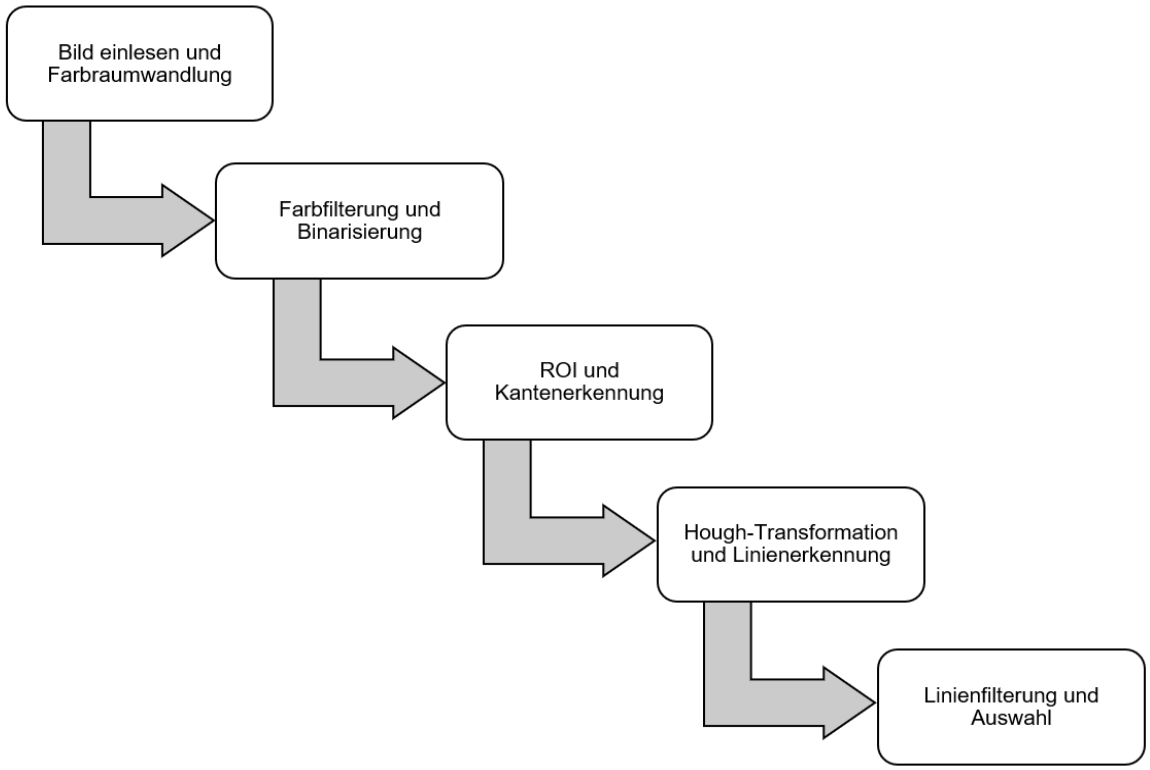


Abbildung 6-1: Ablauf HoughLine Linienerkennung

In einem ersten Schritt wird das **Bild der Kamera eingelesen**. „OpenCV“ nutzt hier die Linux-Programmierschnittstelle (API) „V4L2“ um die Rohdaten der Kamera per USB-Schnittstelle auszulesen und anschließend eine Pipeline des Multimedia-Frameworks „GStreamer“. Die Daten liegen als RGB-Bild mit einer Auflösung von 640x480 Pixeln vor. Somit besitzt jeder Pixel neben seinen Koordinaten, drei 8-Bit Farbinformationen. Für die Weiterverarbeitung ist die Extraktion von Pixeln in bestimmten Farbräumen notwendig.

Zu diesem Zweck wird das Bild in **Graustufen umgewandelt** und zusätzlich in den HSV-Farbraum transformiert. Dieser Farbraum beinhaltet, wie RGB, drei 8-Bit Informationen je Pixel. Jedoch setzt sich ein Pixel hierbei aus einem Farbwert (**H**ue), der Farbsättigung (**S**aturation) und einem Hellwert (**V**alue) zusammen. Damit ist dieser Farbraum besonders gut geeignet um Farben zu extrahieren, da sich auch Mischfarben leicht durch verschiedene Bereiche angeben lassen. Für den vorliegenden Algorithmus wird die Farbe „weiß“ aus dem Graustufenbild extrahiert. Dazu werden mithilfe der Funktion „inRange“ alle Pixel mit einem Grauwert von 0 bis 100 wie folgt gefiltert:

$$mask\_white = cv2.inRange(gray\_image, 0, 100) \quad \textbf{(6.1)}$$

Zusätzlich werden gelbe Pixel aus dem folgenden Spektrum mit der selben Funktion aus HSV-Bild gefiltert:

$$lower\_yellow = np.array([0, 0, 0], dtype = "uint8") \quad \textbf{(6.2)}$$

$$upper\_yellow = np.array([180, 255, 30], dtype = "uint8") \quad \textbf{(6.3)}$$

Der letzte Wert des erstellten Arrays mit dem Bereich von 0-30 symbolisiert hierbei die Farbe Gelb. Diese beiden Farben wurden gewählt um reale Fahrspuren zu extrahieren. Diese sind in den vielen Ländern weltweit weiß oder gelb ausgeführt. Somit lässt sich die vorgeschlagene Filterung auf das meiste Beispielbildmaterial anwenden. Nach der Filterung werden die zwei entstandenen Informationen durch die Funktion „bitwise_and“ zu einem Bild kombiniert.

Dabei entsteht ein **binäres Bild**. Ein Binärbild enthält pro Pixel keine Farbinformation mehr, sondern lediglich eine Null oder eine Eins. Eins steht in diesem Fall für einen weißen oder gelben Pixel und null für alle anderen Farbwerte. Abbildung 6-2 zeigt den Erfolg dieser Filterung und Umwandlung anhand eines Beispielbildes einer Straße.

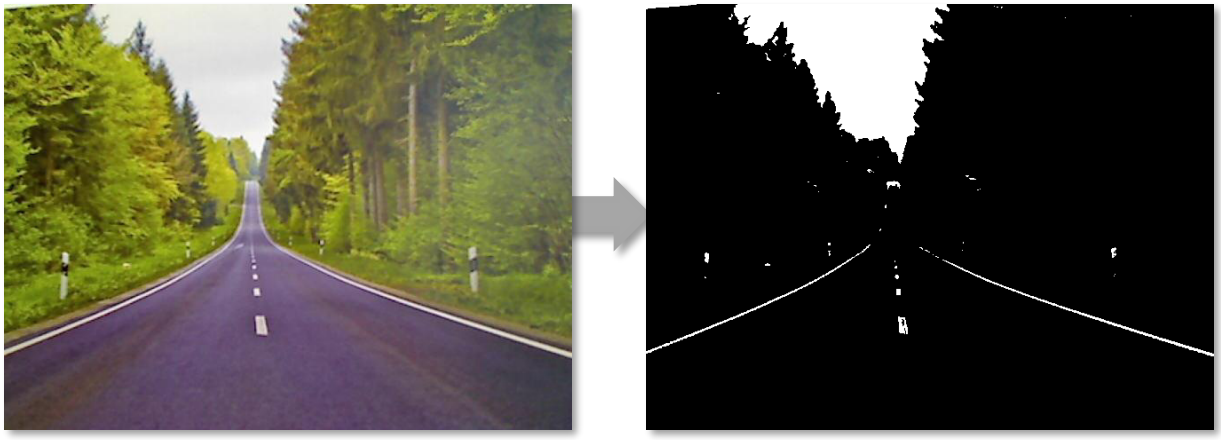


Abbildung 6-2: Filterung und Wandlung in Binärbild Hough

Um dieses Bild für die Linienerkennung vorzubereiten ist es notwendig die Zahl der erkannten Pixel weiter zu reduzieren. Hierzu wird der „**Canny Edge Detector**“ verwendet. Es wird angenommen, dass Fahrspuren als Linien im Bild symbolisiert werden und einen kontrastreichen Übergang zu der restlichen Bildumgebung bilden. Der Algorithmus zur Kantenerkennung erkennt also die Ränder der Fahrspur und reduziert somit die Anzahl der „positiven“ Pixel in dem vorhandenen Binärbild. Der Ansatz zur automatisierten Kantenerkennung von John Canny basiert im Wesentlichen auf einem Gradienten-basierten Bilderkennungsalgorithmus [54]. Das Vorgehen kann auf Graustufenbilder angewendet werden und erkennt durch die Helligkeitsschwankungen mögliche Kanten. In der Praxis wird dies durch die Berechnung des Helligkeitsgradienten sowohl in horizontaler, als auch in vertikaler Richtung erreicht. Zu diesem Zweck verwendet der Algorithmus einen „Sobel-Operator“, um, durch eine mathematische Faltungsoperation, den Gradienten zu bestimmen. Aus dem Originalbild entstehen mithilfe der Operatoren:

$$K_{GX} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (6.4)$$

$$K_{GY} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (6.5)$$

die Gradienten G_X und G_Y für jeden einzelnen Bildpunkt. Mithilfe des Satzes des Pythagoras kann aus den Richtungsgradienten die Gradientenstärke oder der Betrag des Gradienten eines jeden Bildpunktes bestimmt werden. Diese wird auch als „edge strength“, also Kantenstärke, bezeichnet. Folgende Operation wird angewendet:

$$|G| = \sqrt{(G_X)^2 + (G_Y)^2} \quad (6.6)$$

Eine Kante im Bild muss orthogonal zu der Richtung des einzelnen Pixelgradienten liegen. Diese wird im Zweidimensionalen, durch Anwendung des Tangens, folgendermaßen bestimmt:

$$\theta = arctan\left(\frac{G_Y}{G_X}\right) \quad \textbf{(6.7)}$$

Nun liegt ein Bild mit Informationen zu der Stärke und Richtung des Helligkeitsgradienten vor. Um daraus relevante Kanten zu filtern, ist ein mehrstufiger Algorithmus notwendig. In einem ersten Schritt werden die Gradientenwinkel auf die nächsten 45° gerundet. Nun werden die Gradientenstärken benachbarter Pixel in Richtung und Gegenrichtung des Gradientenwinkels verglichen und so eine Zusammengehörigkeit erkannt. Repräsentiert der aktuelle Wert das lokale Maximum, wird sein Wert gespeichert und die benachbarten Werte gelöscht. Das Vorgehen ist in Abbildung 6-3 dargestellt. Jeder Bildpunkt wird durch einen Gradientenwert und dessen Richtung repräsentiert. Die lokalen Maxima sind weiß hinterlegt und stellen eine mögliche Kante dar [55].

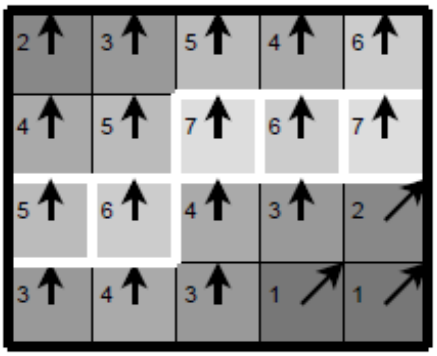


Abbildung 6-3: Gradientenbild Canny Edge Detection [53]

Anschließend wird ein Verfahren der doppelten Schwellwertbegrenzung angewendet. Alle Pixel unterhalb eines definierten Schwellwertes werden aussortiert. Bildpunkte in einem Bereich zwischen Schwellwert eins und zwei werden als schwach und Punkte oberhalb des höheren Schwellwertes zwei als stark markiert. Die als stark markierten Pixel werden anschließend sofort in das endgültige Bild übernommen. Schwache Pixel werden mithilfe eines sogenannten Hystereseverfahrens weitergehend untersucht. Dabei werden die benachbarten Pixel untersucht und lediglich in das endgültige Bild übernommen, wenn eine Verbindung zu einem starken Bildpunkt besteht. Dieses Vorgehen basiert auf der Annahme, dass Nicht-Kanten und Bildstörungen nicht als starke Pixel übernommen werden. Aus diesem Grund ist die sorgfältige Auswahl der Schwellwertgrößen für eine erfolgreiche Filterung entscheidend. Der gesamte „Canny-Edge-Detection“ Algorithmus ist in „OpenCV“ als Funktion implementiert und wird folgendermaßen angewendet:

$$cannyedges = cv2.Canny(gaussgray, lowthreshold, highthreshold) \quad \textbf{(6.8)}$$

Als Eingangsparameter werden das Graustufenbild („gaussgray“) und der untere und oberer Schwellwert übergeben. In dieser Arbeit werden Schwellwerte von 50 und 150 verwendet, da diese bei den vorhandenen Licht- und Kontrastverhältnissen die besten Ergebnisse liefern.

Eine Anfälligkeit des vorgestellten Algorithmus besteht in der Missinterpretation des Bildrauschens. Durch diesen Bildfehler können Helligkeitsschwankungen zu Fehlerkennungen führen. Aus diesem

Grund wird ein vorgeschalteter Glättungsalgorithmus auf das gefilterte Graustufenbild angewendet, der das Bild weichzeichnet. Dazu wird ein Filter verwendet, der auf Basis eines Gaus´schen Kerns den Detailgrad verringert und so Störungen reduzieren kann. Diese zweidimensionale Glockenkurve wird mathematisch durch folgende Funktion repräsentiert:

$$G(x,y) = Ae\frac{-(x-\mu_x)^2}{2\sigma_x^2} - \frac{(y-\mu_y)^2}{2\sigma_y^2} \quad \textbf{(6.9)}$$

Dabei stehen x und y für die Koordinaten des Pixels im Bild. σ stellt die Varianz dar und μ steht für den Maximalwert der Glockenkurve. Wird diese mathematische Faltungsoperation auf ein Graustufenbild angewendet kann eine Gewichtung der Helligkeitswerte gewonnen werden. Dies hat den Effekt, dass einzelne Bildpunkte die extrem von benachbarten Pixeln in Bezug auf ihre Helligkeitswerte abweichen, an diese angeglichen werden und somit ein gleichmäßigeres Bild entsteht. Dies trifft zum Beispiel auf Bildfehler durch Rauschen zu [56]. Abbildung 6-4 zeigt links die Weichzeichnung mittels des vorgestellten Filters und rechts die darauf angewendete Kantenerkennung.

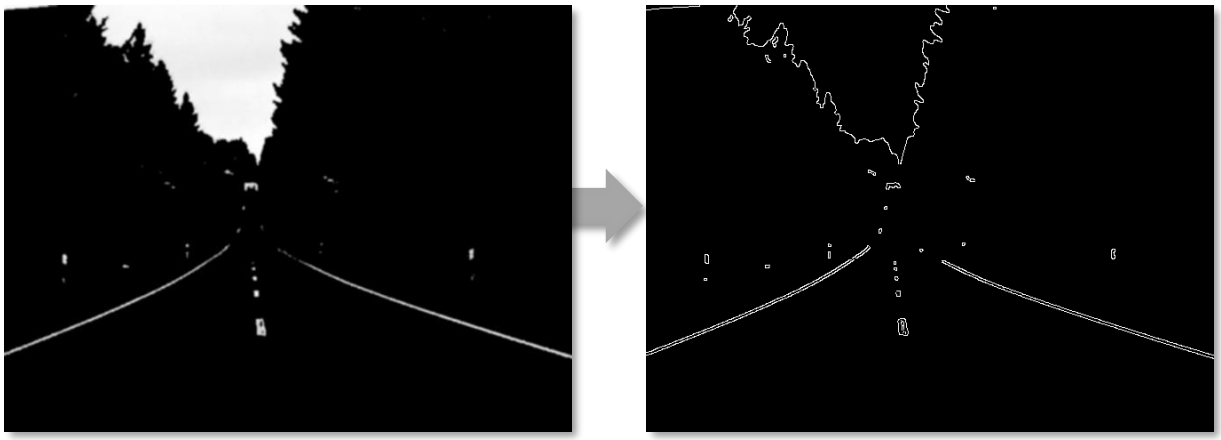


Abbildung 6-4: Weichzeichnung und Kantenerkennung Hough

Die genannten Bildoperationen dienen der Vorbereitung des Bildes für die eigentliche Kantenerkennung. Zunächst wird jedoch eine „**Region of Interest**“ aus dem Gesamtbild extrahiert. Auf Bildern von Straßen liegen die Fahrspuren im Regelfall im unteren Bildbereich und sind, aufgrund der Entfernung, in der Bildmitte zusammenlaufend. Daher kann ein trapezförmiger Bereich des Originalbildes ausgeschnitten werden, der den Bereich abdeckt, in dem eine Fahrspur vermutet werden kann. Durch diese Filterung kann der Rechenaufwand deutlich reduziert werden.

Die eigentliche Kantenerkennung basiert auf der **Hough-Linienerkennung** oder auch Hough-Transformation. Dieser Algorithmus basiert auf der Annahme, dass geometrische Formen durch einfach mathematisch parametrierbare Formeln darstellbar sind. Eine Linie oder Gerade kann dementsprechend in der Form:

$$y = mx + b \quad \textbf{(6.10)}$$

dargestellt werden. Hierbei sind die Parameter „m“ die Steigung und „b“ die Y-Achsenverschiebung. Diese Parameter sollen im Folgenden für mögliche Geraden gefunden werden.

Ein einzelner Punkt in einem zweidimensionalen, kartesischen Koordinatensystem („KOS“) kann durch die Angabe seiner Position (x- und y-Parameter) hinreichend beschrieben werden. Eine Transformation in Polarkoordinaten repräsentiert gleichzeitig eine Punkt zu Linientransformation. In diesem Koordinatensystem wird eine Gerade durch den Winkel θ und den Abstand vom Ursprung ρ wie folgt parametrisiert:

$$\rho = y \sin\theta + x \cos\theta \qquad (6.11)$$

Ein einzelner Punkt kann durch viele Winkel und Abstände beschrieben werden und ergibt somit in einem Polarkoordinatensystem eine Linie in Form einer Sinoide. Dieses Vorgehen wird in Abbildung 6-5 gezeigt. Die Punkte A, B und C des linken kartesischen-KOS werden in das rechte KOS mit Polarkoordinaten (auch Hough-Raum genannt) transformiert. Dort stellen diese drei Punkte drei Sinoiden dar. Der Schnittpunkt $\bar{l}$ hat die Koordinaten ρ und θ. Diese stellen in kartesischen Koordinaten eine Gerade l dar. Wird dieses Verfahren auf eine Vielzahl von Punkten angewendet, ergibt sich ein erweiterter Hough-Raum mit einer Vielzahl von Schnittpunkten. Dieser Hough-Raum wird nun auf lokale Maxima untersucht. Die Koordinatenbereiche, die die höchste Anzahl an Linien aufweisen, stellen mit der größten Wahrscheinlichkeit Linien im Ursprungsbild dar [57].

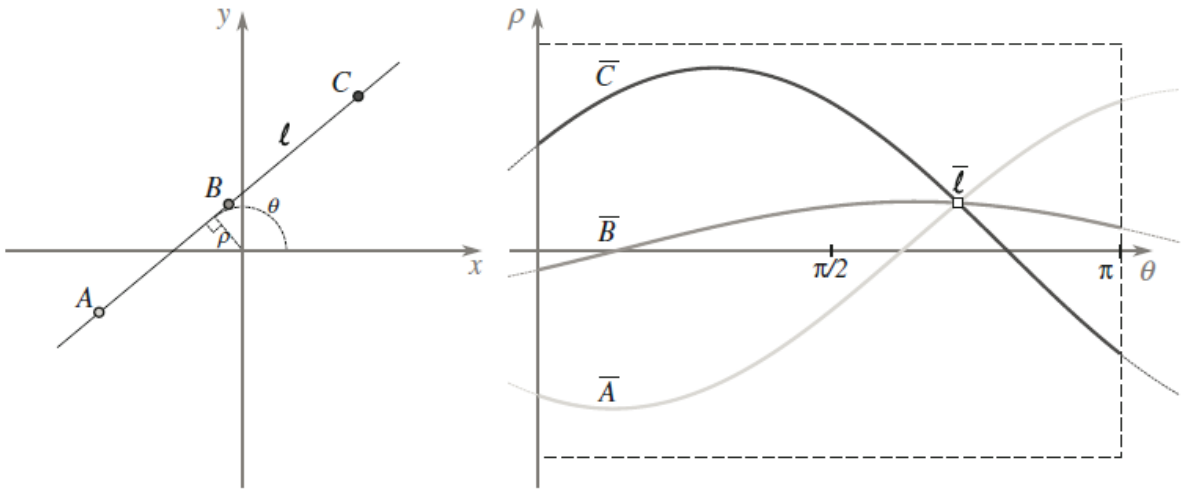


Abbildung 6-5: Transformation kartesische in Polarkoordinaten [55]

Dieses Vorgehen wird in „OpenCV“ als probabilistisches Verfahren umgesetzt. Somit werden nur zufällig ausgewählte Bildpunkte verglichen. Diese Umsetzung erreicht eine ausreichende Erkennungsrate, bei gleichzeitig deutlich verringerten Rechenaufwand.

Die erkannten Linien dieses Algorithmus werden in Abbildung 6-6 als blaue Linien dargestellt. Zum einen werden diese im Binärbild angezeigt (links) und zum anderen auf dem Ausgangsbild (rechts). Die

Bilder zeigen eine Reihe von Fehlerkennungen in Form von Linien, die nicht auf den realen Fahrspuren liegen. Ein Grund dafür ist die Parameterauswahl der einzelnen Ablaufschritte. Durch eine genaue Anpassung an die gegebenen Lichtverhältnisse kann eine deutliche Reduzierung der Fehlerkennungen erzielt werden. Eine weitere Verbesserung wird durch die Sortierung der Linien erreicht.

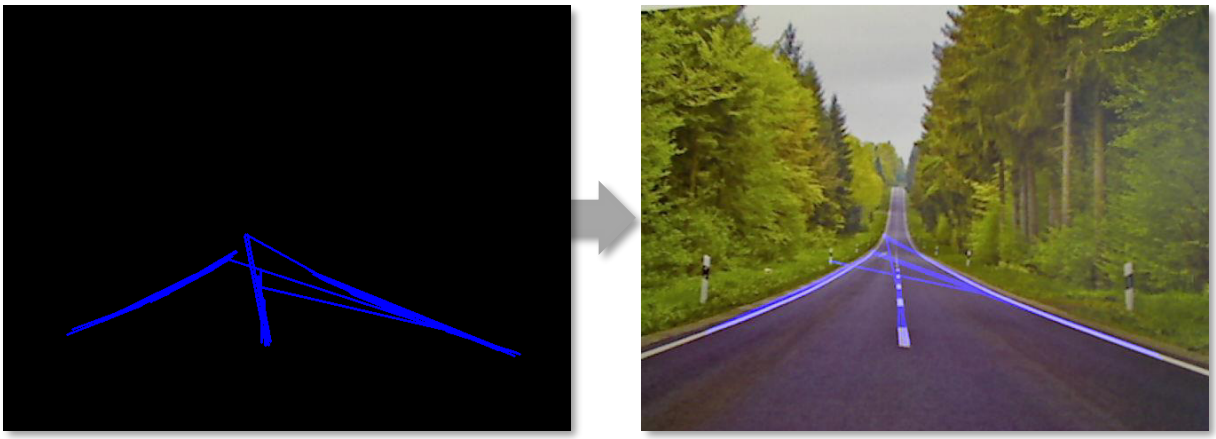


Abbildung 6-6: Visualisierung erkannte Linien Hough

Ein erster Schritt teilt die erkannten Linien in Bereiche auf, die den Fahrspuren „links“, „mitte“ und „rechts“ entsprechen. Durch eine Berechnung des Linienwinkels aus den Startpunkten (*x1, y1*) und Endpunkten (*x2, y2*) nach folgender Formel:

$$angle = np.arctan2(y2 - y1, x2 - x1) * 180.0/np.pi \quad (6.12)$$

erfolgt eine Klassifizierung. Eine anschließende Mittelung der geclusterten Linien führt zu einem Ergebnis von drei möglichen Linienkandidaten, die sich wiederum als Linien auf dem Ausgangsbild darstellen lassen (siehe Abbildung 6-7).

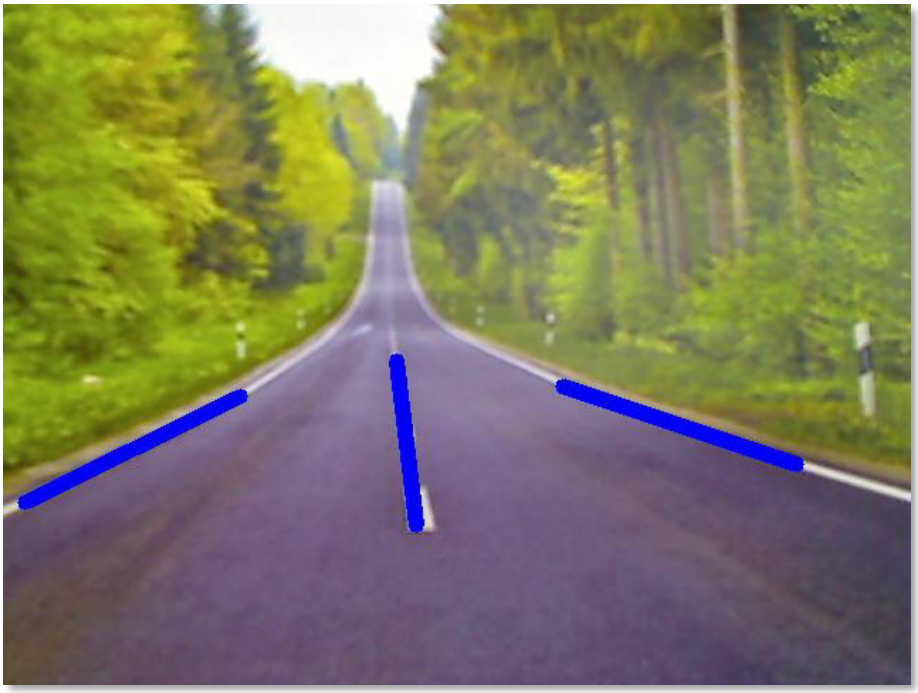


Abbildung 6-7: visualisiertes Endergebnis der HoughLine Linienerkennung

Diese weitere Filterung führt zu akzeptablen Ergebnissen. Die verfügbaren Daten der erkannten Linien bestehen aus den Start- und Endpunkten. Im Testbetrieb mit Videomaterial zeigt sich eine zweckmäßige Performance. Dabei ist jedoch eine hohe Anzahl von Fehlerkennungen zu vermerken. Diese sind auf die schwierige Schwellwertbestimmung der Hough-Transformation zurückzuführen. Eine gesetzte Parameterkombination führt nur in Abstimmung auf die Licht- und Kontrastverhältnisse eines Bildes zu der gewünschten Erkennungsrate.

Ein offensichtlicher Nachteil dieser vorgeschlagenen „HoughLine-Linienerkennung“ liegt in der fehlenden Fähigkeit Kurven zu erkennen. Eine Problemumgehung wird in der Literatur in Form einer gestuften Erkennung vorgeschlagen. Dazu wird das eingelesene Bild, in horizontaler Richtung, in Bereiche unterteilt und die Linienerkennung auf die einzelnen Bereiche angewendet. Das Ergebnis besteht aus erkannten Linien der Einzelsegmente. Aus diesen kann nun ein Kurvenradius abgeschätzt werden. Eine Schwierigkeit dieser Vorgehensweise liegt in der Anpassung der Bereiche. Es muss für jeden Bereich die optimale Filterung gewählt werden, um eine homogene Fahrspurerkennung zu gewährleisten. Ein weiterer Nachteil liegt in der Zunahme der Rechenoperationen, da für jeden Einzelbereich eine Hough-Transformation und eine Filterung durchgeführt werden muss.

Tabelle 6-1: Bewertung HoughLine-Linienerkennung

Kriterium	**Bewertung**
Rechenaufwand	- 1000 Wiederholungen: 82,74s := 12,09 Hz
Erkennungsrate	- Aufwändige Filterung der Fahrspurkandidaten
Robustheit	- Anfälligkeit für Schwankungen der Licht- und Kontrastverhältnisse
Fähigkeiten	- Nur Geraden können erkannt werden, Kurven nur mit großem Aufwand

Tabelle 6-1 zeigt eine Zusammenfassung der Bewertung des vorgestellten Algorithmus. Dabei werden vier relevante Kriterien unterschieden. Die Berechnung des Rechenaufwandes bezieht sich auf die Messung der Zeit von 1000 Schleifendurchgängen und damit 1000 verarbeiteten Bildern. Diese Messung wurde fünfmal wiederholt und das Ergebnis gemittelt. Während der Messung wurde das Originalbild mit den visualisierten Linien auf dem Bildschirm ausgegeben (siehe Abbildung 6-7).

6.1.2 Bird-View Spurerkennung

Eine weitere Methode der Spurerkennung wird nachfolgend beschrieben und bewertet. Dieser Ansatz basiert in großen Teilen auf der Arbeit von Yong Ding, Zhang Xu, Yubin Zhang und Ke Sun aus dem

Jahr 2016 [58]. Dabei wird eine Bildtransformation, auch „Inverse Perspective Mapping“ genannt, verwendet. Diese verändert das Ursprungsbild ausschnittsweise dahingehend, dass eine Art Draufsicht gewonnen wird. Diese Ansicht wird auch „Bird View“ genannt und ist daher namensgebend für das folgende Kapitel. Ein Vorteil dieses Transformationsansatzes liegt in der konstanten Spurbreite. Eine Filterung der erkannten Linien, anhand ihres Winkels, ist nicht notwendig. Weiterhin wird ein Ansatz verwendet, der auch die Detektion von Kurven und die Berechnung des entsprechenden Kurvenradius zulässt. Nachfolgend werden die einzelnen Programmschritte vom Einlesen des Bildes bis zur Berechnung der geforderten Fahrspurparameter erläutert. Eine Übersicht ist in Abbildung 6-8 gezeigt. Da einzelne Schritte den vorgestellten Methoden aus Kapitel 6.1.1 entsprechen, wird auf eine genaue Beschreibung an dieser Stelle verzichtet und stattdessen auf das entsprechende Kapitel verwiesen.

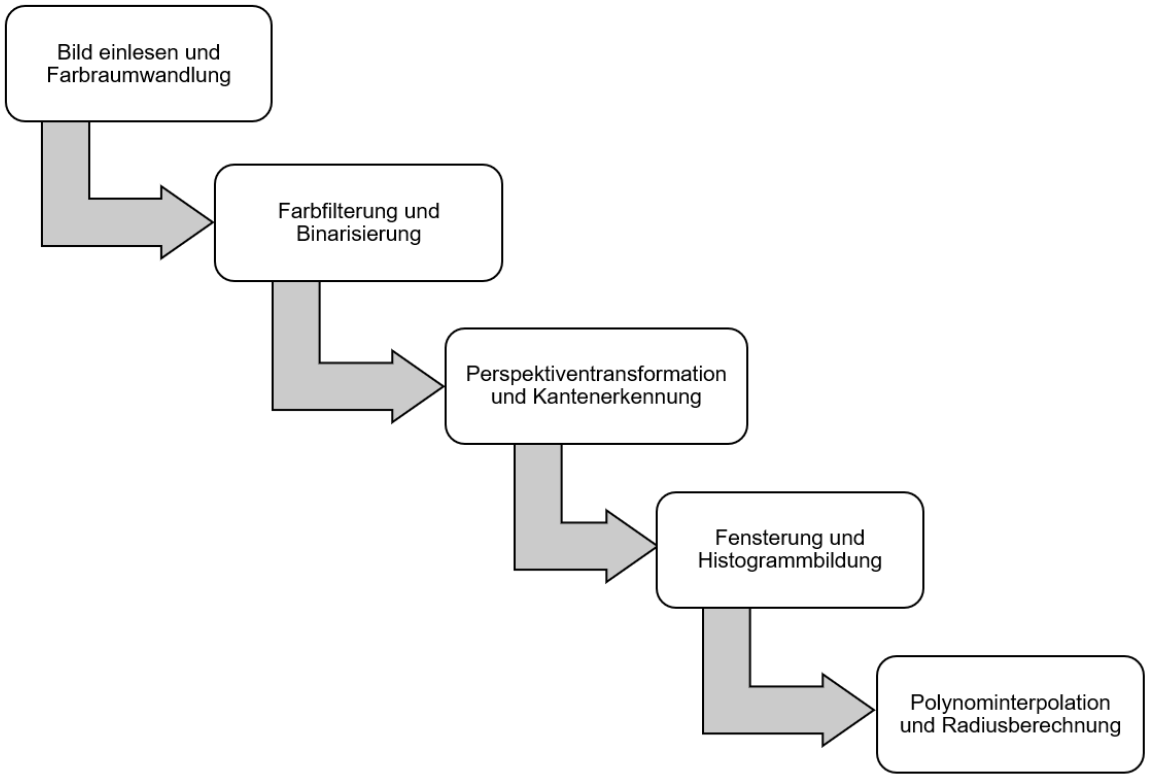


Abbildung 6-8: Vorgehen Bird View Spurerkennung

Für die Fahrspurerkennung ist ebenfalls eine Filterung des eingelesenen Bildes notwendig. Diese besitzt denselben Algorithmus wie die im vorangegangenen Kapitel vorgestellte Methode. Es wird eine **Farbfilterung** gemäß der erwarteten Fahrspur vorgenommen und anschließend das Bild **binarisiert** und mithilfe des „**Canny Edge Detection**“ Algorithmus auf mögliche Kanten untersucht. Die Anwendung dieses Vorgehens ist in Abbildung 6-9 dargestellt. Als Ausgangsbild wurde dabei eine Straßenszene mit gelben und weißen unterbrochenen Spurmarkierungen und einem leichten Kurvenradius gewählt, um alle Features der vorgestellten Erkennung bestmöglich zu testen. An dieser Stelle liest die Kamera ein Breitbild im Verhältnis 16:9 ein. Diese Einstellung beschneidet lediglich den Bildausschnitt oben und unten und hat keinerlei Auswirkung auf das Sichtfeld der Kamera. Da hierbei der Rechenaufwand, aufgrund der reduzierten Anzahl an Bildpunkten, verringert ist, wird das Einlesen im weiteren Verlauf wieder

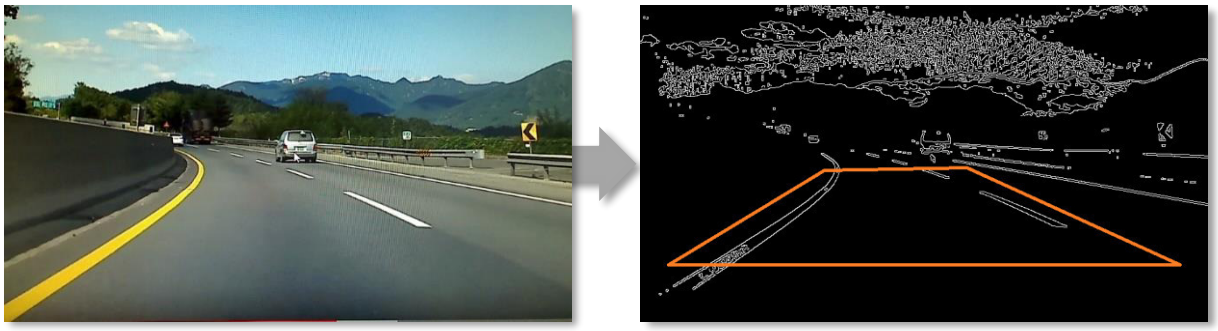


Abbildung 6-9: Filterung und Kantenerkennung Ursprungsbild - Bird

auf ein Bildverhältnis von 4:3 umgestellt. Dies sichert eine Vergleichbarkeit mit der in Kapitel 6.1.1 vorgestellten Methode.

In dem rechten Bild der Abbildung 6-9 ist ein orangefarbenes Trapez erkennbar. Dieses dient als Basis für das anschließende „Inverse Perspective Mapping“. IPM beschreibt eine Methode der digitalen Bildverarbeitung, die es ermöglicht, aus einem zweidimensionalen Bild durch eine Transformation ein zweites Bild zu erzeugen. Die Abbildung zeigt dieselbe Bildszene des Ausgangsbildes, jedoch aus einer anderen Perspektive [59]. In dem Beispiel einer aufgenommenen Straßenszene liegt der Bildfokus in Richtung des Horizonts. Die Fahrspuren sind zulaufend und treffen sich in einem virtuellen Fluchtpunkt. Das Ziel ist es eine Draufsicht zu erhalten („Bird View“), in der die Fahrspuren nicht zulaufend sind und somit eine konstante Spurbreite besitzen. Dieses wird durch Anwendung einer Transformationsmatrix erreicht. In „OpenCV“ erfüllt die Funktion:

$$M = cv2.getPerspectiveTransform(transform\_src, transform\_dst) \quad (6.13)$$

den genannten Zweck. „M“ ist hierbei die Variable für die Transformationsmatrix. Die Funktion „*getPerspectiveTrannsform*“ berechnet diese aus der Quelle („*transform_src*“) und dem Ziel („*transform_dst*“). Dazu werden die Variablen als „floating point“ Matrizen vorher in Abhängigkeit des Eingangsbildes und der erwarteten Spurposition definiert. Die Transformationsmatrix wird als 3x3 Matrix berechnet, so dass folgende Gleichung erfüllt ist [60]:

$$\begin{bmatrix} t_i x_i' \\ t_i y_i' \\ t_i \end{bmatrix} = M * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (6.14)$$

Dabei repräsentieren x_i und y_i die Ursprungswerte. Hier ist zu beachten, dass in „OpenCV“ der Ursprung des Koordinatensystems in der linken oberen Bildecke platziert ist. Ein Beispiel für eine Transformation ist in Abbildung 6-10 gezeigt. Die roten Punkte stellen die Quelle dar. Wobei die grünen Punkte das Ziel illustrieren. Für das vorliegende Beispiel lauten die entsprechenden Transformationsparameter:

$$transform\_src = np.float32([[240,240],[400,240],[600,400],[40,400]]) \quad (6.15)$$

$$transform\_dst = np.float32([[0,0],[w-1,0],[w-1,h-1],[0,h-1]]) \quad (6.16)$$

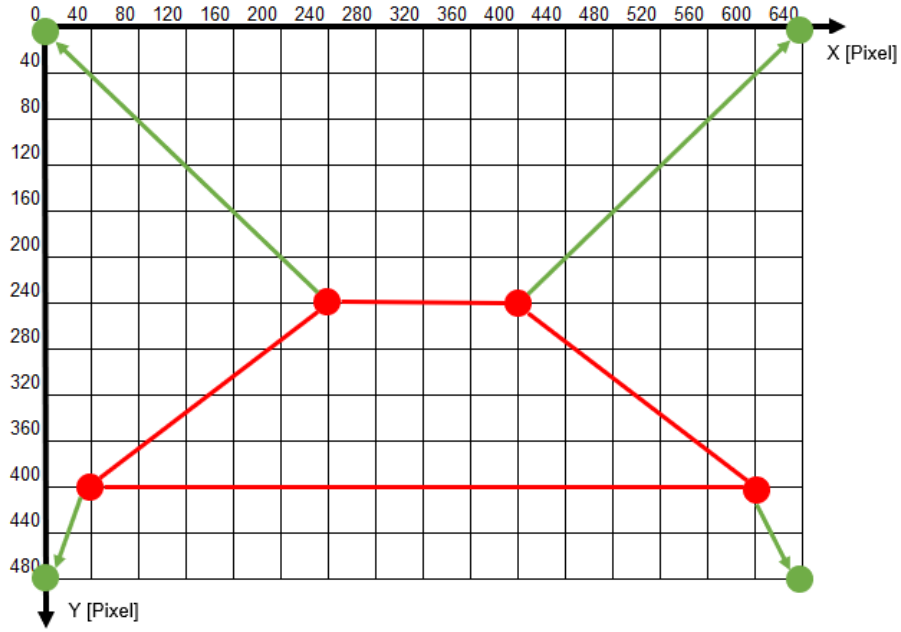


Abbildung 6-10: Beispieltransformation mit KOS

Die berechnete 3x3 Transformationsmatrix wird nun verwendet, um die eigentliche Bildtransformation durchzuführen. Nach folgender Formel:

$$dst(x,y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right) \quad (6.17)$$

wird aus der Quelle („src“) das Ziel („dst“) berechnet. Diese Transformation wird auf das, mithilfe des „Canny Edge Detection“ Algorithmus bearbeite, Binärbild angewendet. Das Ergebnis ist eine Draufsicht der vorliegenden Straße. Dabei ist die Auswahl der Quellenkoordinaten von entscheidender Bedeutung. Nur wenn die Punkte die gesamte Fahrbahnmarkierung abdecken, kann die Spurerkennung erfolgreich sein. Die Koordinaten des Ziels sind nicht von funktionaler Bedeutung. Eine geeignete Auswahl erleichtert lediglich die Darstellung des transformierten Bildes. Die linke Illustration der Abbildung 6-11 zeigt die angewendete Transformation auf das im Hochformat dargestellte Testbild. Dabei sind eine konstante Spurbreite und die leichte Krümmung der Straße zu erkennen.

Die rechte Bildseite zeigt die gleiche Szene mit einer grünen Fensterung. Diese grünen Rechtecke visualisieren die Aufteilung des Bildes in einzelne Erkennungsbereiche und bilden die Grundlage für weitere Berechnungen.

Die eigentliche Detektion der Fahrspuren basiert auf der Idee, dass eine Spur im gefilterten Bild dort liegen muss, wo sich die größte Anzahl an erkannten Pixeln befindet. Diese Methode kann auch als

Suche nach den lokalen Pixelmaxima beschrieben werden. Ausgehend von der Annahme, dass im transformierten und zuvor gefilterten Bild nur relevante Pixel als positiv (binär = 1) gekennzeichnet sind, wird ein Histogramm des transformierten Bildausschnittes erstellt.

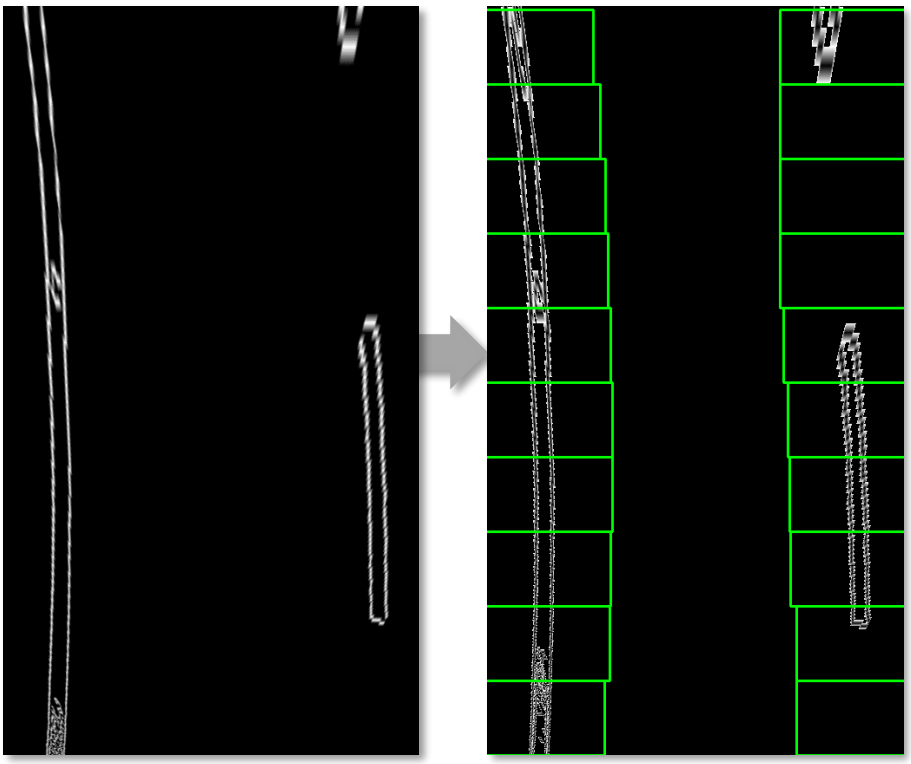


Abbildung 6-11: transformiertes Bild und Fensterung - Bird

Die Histogrammfunktion von OpenCV erstellt ein X-Y-Diagramm, welches die Pixelanzahl über der Bildbreite darstellt. Das erstellte Diagramm für das vorliegende Beispiel ist in Abbildung 6-12 zu sehen. Dabei lassen sich zwei Maxima erkennen. Am linken Bildrand die linke Fahrbahnmarkierung und am rechten Bildrand die rechte Fahrbahnmarkierung.

Beide Erhebungen besitzen zwei lokale Maxima. Dies ist auf die Kantenerkennung zurückzuführen, die die Fahrbahnmarkierung in ihre rechte und linke Kante, also den Übergang zur Straße, aufteilt. Des Weiteren ist das rechte Maximum deutlich schwächer ausgeprägt. Dies ist mit der unterbrochenen Fahrbahnmarkierung dieser Seite zu erklären.

Die Pixelmaxima können rechnerisch bestimmt werden und dienen als Ausgangspunkt für die gezeigte Fensterung. In einer Schleife wird nun, für die gewählte Anzahl der Fenster, ein Algorithmus durchlaufen, der für jedes Fenster das Pixelmaximum bestimmt. Dabei wird der jeweilige Wert an den nächsten Schleifendurchlauf übergeben und dient diesem als Ursprung. Durch eine Aufteilung des Bildes in der Bildmitte, wird das Vorgehen jeweils für die rechte und linke Fahrbahnmarkierung durchgeführt. Dadurch wird auch eine Kurvenabbildung möglich. Dieses Vorgehen ist durch die festgelegte Fensterbreite beschränkt. Ein breites Fenster ermöglicht einen großen darstellbaren Kurvenradius, wohingegen ein schmales Fenster die Erkennungsgenauigkeit erhöht. Aufgrund der mechanischen Beschränkungen des Basisfahrzeuges ist nur ein relativ großer Kurvenradius technisch darstellbar. Daher wurde eine

Fensterbreite von 80 Pixeln gewählt. Dies stellt einen guten Kompromiss für den beschriebenen Zielkonflikt dar.

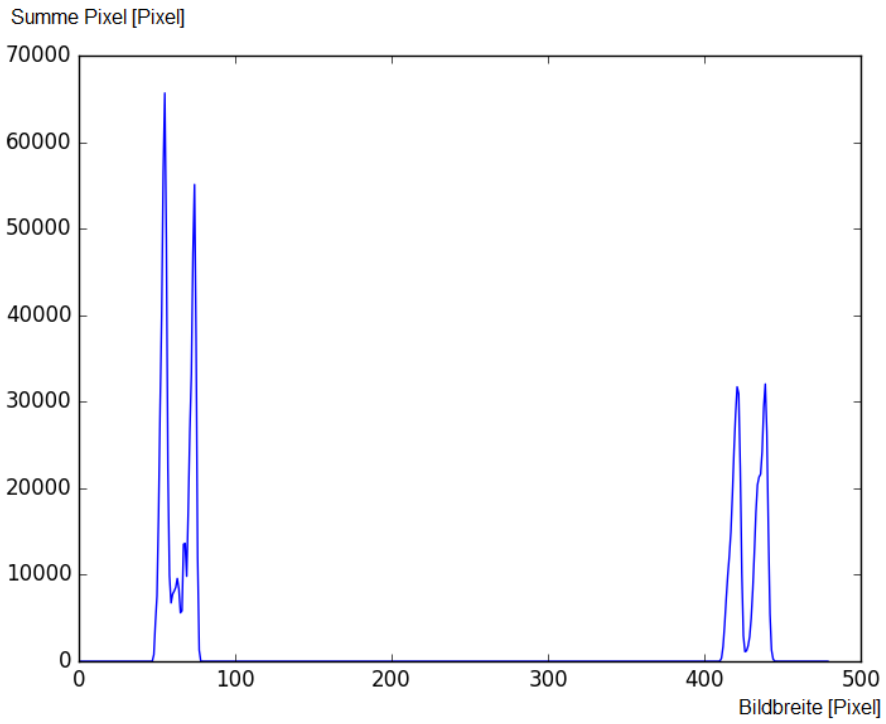


Abbildung 6-12: Histogramm des transformierten Beispielbildes - Bird

Die erkannten Maxima der Fenster werden in einer „list"-Variable gespeichert und dienen nun als Stützstellen für eine Polynominterpolation. Dabei steht die Anzahl der Stützstellen in Abhängigkeit von der Anzahl der gewählten Fenster. Eine Reduzierung der Fenster führt zu einem verminderten Rechenaufwand, allerdings auch zu einer geringeren Genauigkeit der Spurabbildung. Die Funktion:

$$left\_fit = np.polyfit(lefty, leftx, 2) \quad (6.18)$$

berechnet die Parameter für ein Polynom zweiter Ordnung. Die Interpolation nutzt hierzu die Methode der kleinsten Quadrate. Eine quadratische Funktion wurde an dieser Stelle gewählt, da diese eine Straßenkurve am besten abbilden kann und zusätzlich den Rechenaufwand begrenzt. Abbildung 6-13 zeigt auf der linken Seite die Darstellung der berechneten Fahrbahnpolynome der rechten und linken Begrenzung. Zusätzlich ist die Fläche dazwischen grün markiert, um die erkannte Fahrbahn zu visualisieren. Es zeigt sich eine sehr genaue Abbildung der gezeigten Fahrbahnmarkierung. Sowohl die durchgehende linke, als auch die unterbrochene rechte Markierung werden bis auf einen kleinen Restfehler gut dargestellt. Diese geringe Abweichung ist durch die gewählte Polynomart begründet und kann für das vorliegende Problem vernachlässigt werden.

Die rechte Bildseite der Abbildung 6-13 zeigt eine Rücktransformation der erkannten Fahrspur auf das Originalbildformat. Dies wird durch eine Umkehrung der bereits beschriebenen Transformationstechnik erreicht. Eine Anwendung der Rücktransformation auf das Ausgangsbild ist in Abbildung 6-14 zu sehen.

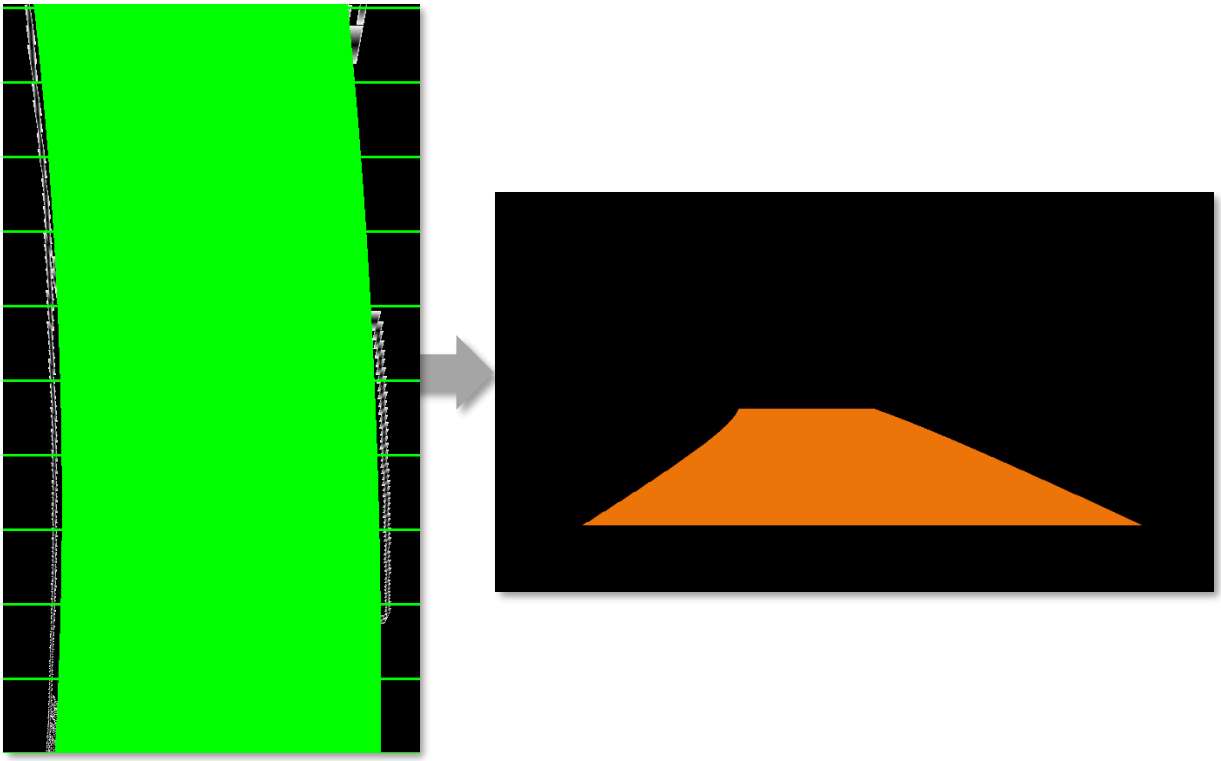


Abbildung 6-13: Polynomdarstellung und Rücktransformation - Bird

Für eine Verbesserung der Erkennung bei Bewegtbildern wurde zusätzlich ein Algorithmus zur Gewichtung der letzten erkannten Polynome implementiert. Dazu werden die letzten zehn Polynome der rechten und linken Seite in einer Variable gespeichert und das jeweils aktuelle Polynom doppelt gewichtet. Aus dieser Variable wird der Durchschnitt errechnet, der entsprechend im Bild dargestellt wird. Diese Gewichtung kann Fehlerkennungen effektiv ausgleichen und verbessert somit die Gleichmäßigkeit der Erkennung.

Im linken oberen Bildabschnitt der Abbildung 6-14 ist zusätzlich der berechnete Kurvenradius dargestellt. Diese Berechnung stellt einen großen Vorteil der präsentierten Erkennungsmethode dar.

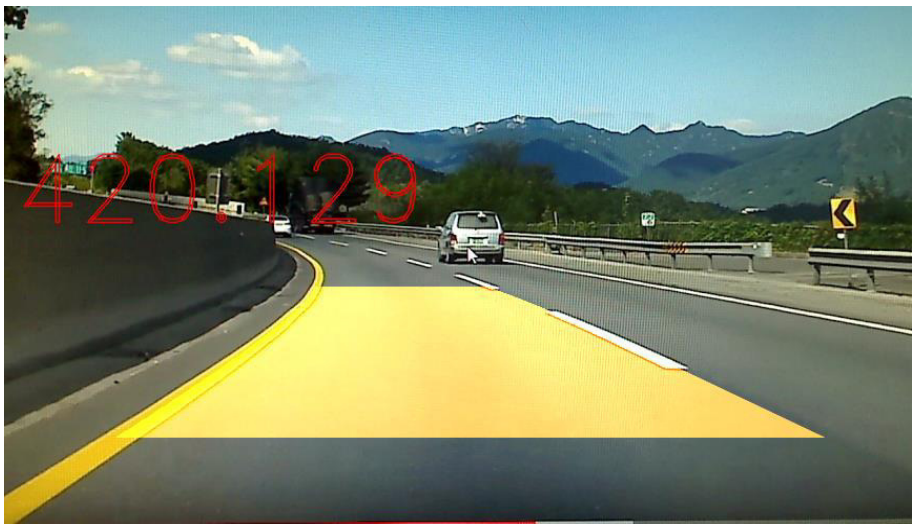


Abbildung 6-14: visualisiertes Endergebnis der Bird View Spurerkennung

Durch die Repräsentation der Fahrbahnmarkierungen als Polynom kann dieser Radius einfach aus den Bildparametern bestimmt werden. Es findet eine Rückrechnung der erkannten Polynome in die Koordinaten des Ursprungsbildes statt. Anschließend wird durch das Annähern eines virtuellen Kreises an die erkannte Kurve der Radius bestimmt. Der dargestellte Kurvenradius ist der Mittelwert aus dem linken und rechten Markierungsradius. Der Wert wird in Metern ausgegeben. Diese Berechnung basiert auf einer statischen Umrechnung von Anzahl der Pixel zu Metern. Diese Umrechnungsfaktoren müssen für jede Kameraposition erneut vermessen und angepasst werden. Auf dem gezeigten Testbild basiert der errechnete Wert auf einer Abschätzung der realen Maße.

Tabelle 6-2: Bewertung Bird View Spurerkennung

Kriterium	Bewertung
Rechenaufwand	- 1000 Wiederholungen: 113,39s := 8,82 Hz
Erkennungsrate	- Gute Erkennung auch bei bewegten Bildern
Robustheit	- Gute Erkennung auch in wechselnden Bedingungen und unterbrochenen Spuren
Fähigkeiten	- Berechnung des Kurvenradius aus Polynom leicht möglich

Tabelle 6-2 zeigt eine Bewertung der vorgestellten Methode der Spurerkennung. Bei der Anwendung auf bewegte Bilder ist eine durchweg gute Erkennung festzustellen, die eine geringe Anfälligkeit für Schwankungen der Licht- und Kontrastverhältnisse aufweist. Die Möglichkeit den Radius einer erkannten Kurve abzuschätzen, bedeutet weiterhin, für eine zukünftige Querregelung auf Basis der erzeugten Daten, einen enormen Vorteil. Diese Daten können beispielsweise in eine Vorregelung einfließen oder für eine Anpassung der Geschwindigkeit an den Streckenverlauf dienen. Die Messung des Rechenaufwandes zeigt eine deutliche Verschlechterung der Erkennungsfrequenz gegenüber der in Kapitel 6.1.1 vorgestellten Methode. Dies ist hauptsächlich in der komplizierteren Visualisierung der Fahrspur begründet und lässt sich in einer späteren Anwendung verbessern.

Für einen weiteren Vergleich ist in Abbildung 6-15 die Anwendung der in Kapitel 6.1.1 gezeigten Spurerkennung auf das neue Ausgangsbild dargestellt. Deutlich ist hierbei die Fehlerkennung der rechten Markierung zu erkennen. Weiterhin zeigt sich die Beschränkung der Erkennung auf Geraden und der Kurvenverlauf kann nicht abgebildet werden. In Summe überwiegen somit die Vorteile der „Bird-View Spurerkennung“. Trotz des erhöhten Rechenaufwands kann dauerhaft eine stabile Erkennung der Fahrbahnmarkierungen sichergestellt werden. Weiterhin ist die Anfälligkeit für Fehler durch Lichtschwankungen, aufgrund der Methode der Pixelmaxima, stark reduziert. Für eine Anwendung als Spurerkennung in einer Querregelung ist die „Bird-View Spurerkennung“ somit nach einer Optimierung der Laufzeit am besten geeignet.

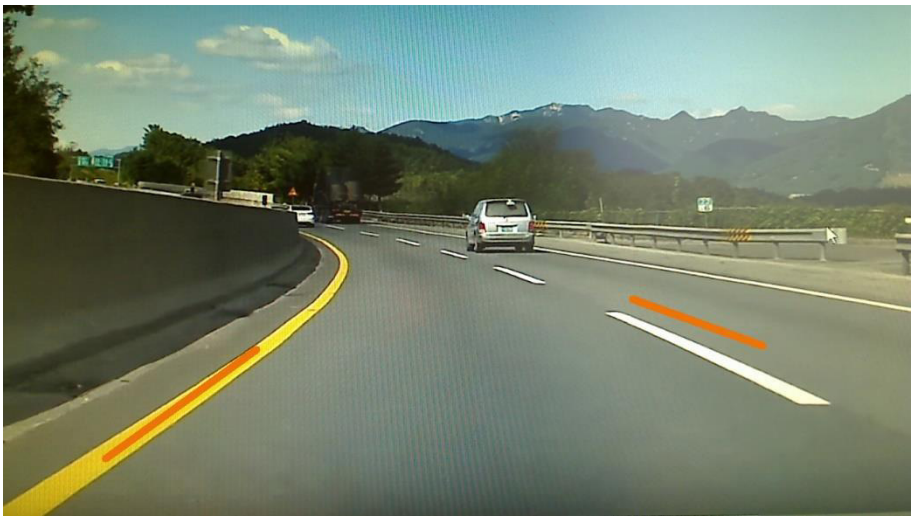


Abbildung 6-15: Anwendung HoughLine Linienerkennung auf Beispielbild

6.2 Schildererkennung

Zur Bewältigung des selbstgewählten Parcours ist eine Schildererkennung vorgesehen. Anhand eines Stoppschildes werden im folgenden Unterkapitel drei verschiedene Methoden der Schildererkennung vorgestellt, miteinander verglichen und bewertet. Ein Stoppschild wurde hierbei als Beispiel gewählt, da sich seine Konsequenz leicht auch in einem 1:10 Fahrzeug demonstrieren lässt. Ein erkanntes Stoppschild muss, als Element der Längsregelung, ein Anhalten des Fahrzeuges bewirken. Dazu ist eine robuste Erkennung auch unter wechselnden Lichtverhältnissen und Blickwinkeln gefordert.

6.2.1 Template Matching

Die Methode des „Template Matching“ ist ein einfacher Vorlagenabgleich. Es wird ein Bild eingelesen und mit einer vorher definierten Vorlage durch maschinelle Suchmethoden abgeglichen. In „OpenCV“ ist die Funktion dazu folgendermaßen aufrufbar:

$$res = cv2.matchTemplate(gray, template, cv2.TM\_CCOEFF) \quad (6.19)$$

Die Parameter der Funktion sind das eingelesene Bild („*gray*“), die Vorlage („*template*“) und eine mathematische Methode, um die Bilder zu vergleichen („*cv2.TM_CCOEFF*“). Der Algorithmus der Funktion vergleicht nun Bildpunkt für Bildpunkt die Vorlage mit dem, in Graustufen umgewandelten, eingelesenen Bild. Das Ergebnis ist eine Matrix *res*, die die Bewertung der Übereinstimmung enthält. Es kann aus insgesamt sechs verschiedenen Varianten des Abgleichens ausgewählt werden. Die oben Gezeigte verwendet die Gewichtungsformel:

$$R(x,y) = \sum_{x'y'} (T'(x',y') * I(x+x', y+y')) \quad (6.20)$$

Dabei steht *T* für die Vorlage und *I* für das eingelesene Bild. *R* ist die resultierende Matrix, die für jede Koordinate einen Wert der möglichen Übereinstimmung enthält. Im Anschluss an die Erkennung kann die Position des Maximums der Kongruenz bestimmt werden. Diese Position ist der wahrscheinlichste Ort für ein „match". Auch mehrere lokale Maxima lassen sich bestimmen. Durch Anwendung eines Schwellwertes können die Ergebnisse sortiert werden [61].

Abbildung 6-16: Schildererkennung Template und Beispiele

In einer Testreihe wurden alle Gewichtungsmethoden miteinander verglichen und wiesen, bei dem angewendeten Testmaterial, keine signifikanten Unterschiede auf. Dazu wurde das linke Bild in der Abbildung 6-16 als Vorlage eingelesen. Anschließend wird dieses mit der vorgestellten Methode mit den drei rechten Bildelementen derselben Abbildung verglichen. Die gewählten Beispielbilder enthalten das Originalbild und zwei Szenen aus der realen Welt. Dabei weisen die Stoppschilder unterschiedliche Hintergründe und Winkel auf. Die Erkennungsrate der vorgeschlagenen Methode ist nicht zufriedenstellend. Erst ab einem Schwellwert von unter 0,3 können Stoppschilder erkannt werden. Dieser Wert entspricht einer Sicherheit von 30%. Weiterhin ist die Erkennung der beiden „realen" Stoppschilder nur rudimentär möglich.

6.2.2 Mustererkennung

Die folgende Methode basiert auf einem selbstentwickelten Algorithmus, der wiederkehrende Muster von Stoppschildern erkennen kann. Diese Muster sind zum einen die rote Farbe in und zum anderen die charakteristische Form eines Achtecks.

Das eingelesene Bild wird in einem ersten Schritt nach Farben gefiltert. Dazu wird die bereits in Kapitel 6.1.1 vorgestellte Funktion *cv2.inRange* verwendet. Die Schwellwerte für die Farbe rot im HSV-Farbraum sind die folgenden:

$$upper = np.array([180, 255, 255], dtype = np.uint8) \quad (6.21)$$

$$lower = np.array([170, 70, 50], dtype = np.uint8) \quad (6.22)$$

Das gefilterte Binärbild muss anschließend mithilfe der Funktion *cv2.erode* bearbeitet werden. Diese Funktion löscht die Umrisse von Pixelstrukturen. Bei Anwendung mit einem großen Kern können kleinere Pixelansammlungen gelöscht und somit Erkennungsfehler beseitigt werden. Für eine weitere Reduzierung wird die Funktion *cv2.morphologyEx* verwendet. Diese Funktion wurde speziell entwickelt,

um Bildfehler zu beseitigen und kann alleinstehende Pixel erkennen und diese löschen. Auf das so gefilterte Bild wird die Funktion *cv2.findContours* angewendet:

$$contours = cv2.findContours(morp, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE) \quad \textbf{(6.23)}$$

Diese Funktion kann in Binärbildern die Konturen von Objekten erkennen und speichert diese als Vektorinformationen in der Variable *contours*. Die Option $cv2.RETR\_EXTERNAL$ sorgt dafür, dass nur äußere Konturen gespeichert werden und verhindert somit, dass die inneren Übergänge zu der weißen Schrift des Stoppschildes erkannt werden. $cv2.CHAIN\_APPROX\_SIMPLE$ bewirkt, dass lediglich die Endpunkte der erkannten Konturen gespeichert werden. Dadurch reduziert sich der Rechenaufwand und die Erkennung von Formen wird vereinfacht. Die Funktion *cv2.approxPolyDP* wird anschließend angewendet, um aus den gespeicherten Punkten Formen zu generieren. Entspricht die Anzahl der erkannten Formen genau acht, liegt ein Achteck vor und die erkannte Form wird als Stoppschild bewertet.

Abbildung 6-17: Mustererkennung visualisiertes Stoppschild

Abbildung 6-17 zeigt die Visualisierung eines erkannten Stoppschildes. Dieses wurde mit der Kamera von einem Ausdruck abgefilmt. Insgesamt zeigt sich diese Methode der Erkennung als ausreichend genau für eine Längsregelung. Stoppschilder werden mit einer zufriedenstellenden Sicherheit erkannt und die Anzahl der Fehlerkennungen ist als gering zu bezeichnen. Jedoch ist die Robustheit nicht hinlänglich. Eine Erkennung von verzerrten oder verfärbten Stoppschildern kann nicht gewährleistet werden, da der Algorithmus diese Fälle nicht abdeckt.

6.2.3 Haar-Cascade Classifier

Eine dritte Variante basiert auf einer Methode des maschinellen Lernens („machine learning“) zur Musterextraktion und Wiedererkennung. Der vorgestellte Algorithmus wurde erstmalig im Jahr 2001 von

Viola und Jones vorgestellt und ermöglicht die Erkennung von beliebigen Objekten. Der Name der Methode setzt sich aus mehreren Begriffen zusammen. „Haar“-Wavelets beschreiben die ersten entdeckten Wavelet-Funktionen. Diese Rechteckfunktionen werden allgemein zur Bildanalyse genutzt und dienen als Basisfunktion der Frequenzanalyse. Diese Funktionen haben folgende Gestalt:

$$f(x) = \begin{cases} 1 & 0 \leq x \leq 1/2 \\ -1 & 1/2 \leq x \leq 1 \\ 0 & sonst \end{cases} \qquad (6.24)$$

Die von Viola und Jonas vorgeschlagene Methode nutzt diese Basisfunktionen, um Muster in Bildern zu erkennen. Dazu wird eine Cascade an Mustern so zusammengestellt, dass ein Klassifizierer („classifier“) zur Objekterkennung entsteht. In „OpenCV“ ist der vorgeschlagene Algorithmus als Funktion implementiert. Aus einer Reihe von Positivbeispielen (Bilder die das zu klassifizierende Objekt enthalten) und Negativbeispielen (Bilder die das zu klassifizierende Objekt nicht enthalten) werden mithilfe einer überwachten Lernmethode alle relevanten Muster herausgefiltert. Theoretisch besitzt ein 24x24 Pixel großes Bild 160000 haarähnliche Muster. Diese setzen sich aus möglichen Kontrastunterschieden zwischen einzelnen Bildpunkten zusammen. Mithilfe der mathematischen Funktion des „AdaBoost“ („Adaptive Boosting“) kann der hinterlegte Algorithmus jedes Beispiel auf alle vorhandenen Muster untersuchen und ihre Relevanz gewichten. Diese Gewichtung wird umso genauer, je mehr Lernbeispiele verwendet werden. In der Literatur wird meist eine Mindestanzahl von 40 Positiv-, sowie 40 Negativbeispielen vorgeschlagen, um eine ausreichende Sicherheit der Erkennung zu gewährleisten. Durch die Anwendung der „AdaBoost“ Funktionalität kann die Performance des Anlernens deutlich verbessert werden und daher ist auch eine große Beispielzahl möglich. Die Clusterung oder auch Klassifizierung kann in einer „xml“-Datei gespeichert werden und dient als Basis für die Erkennung von Objekten in beliebigen Bildern [62].

Als Classifier für das vorliegende Projekt wird eine vorgefertigte Datei von der Internetplattform „Github“ verwendet. Diese unter BSD-Lizenz veröffentlichte Version entstand im Rahmen des Nanodegree „Autonomous Driving“ der Udacity Onlineakademie und wurde unter Verwendung von insgesamt 40 positiv und 40 Negativbeispielen generiert. Dadurch zeichnet sich diese Version durch eine hohe Erkennungsrate und Sicherheit aus [63].

Zu Beginn einer Erkennung wird die „xml“-Datei des Classifiers in das Programm geladen und dient nun als Basis für die Funktion zur Objekterkennung:

$$stops = stop\_detect.detectMultiScale(gray, 1.4, 5) \qquad (6.25)$$

„*Stop_detect*“ ist das Classifier-Objekt auf welches die Methode *„detectMultiScale“* angewendet wird. Als Eingangsparameter wird ein Graustufenbild („*gray*“) und weiterhin ein Skalierungsfaktor an die Funktion übergeben. Diese Methode untersucht das eingelesene Bild stufenweise auf Objekte unterschiedlicher Größe. Dabei werden die hinterlegten Muster auf eine sehr effiziente Weise zusammengefasst,

was eine extrem schnelle Detektion ermöglicht. Erkannte Objekte werden in der Variable „*stops*“ gespeichert. Dabei werden Informationen sowohl zur Position, als auch zur Größe des erkannten Objektes abgespeichert und lassen sich einfach als Rechteck im Bild darstellen. Auf eine Visualisierung der Erkennung wird an dieser Stelle verzichtet, da diese keinen Mehrwert bietet.

Die vorgestellte Stoppschilderkennung ist einerseits sehr schnell und andererseits äußerst robust. Auch Schilder in wechselnden Lichtverhältnissen und unter extremen Winkeln werden sicher detektiert. Diese Eigenschaft ist für eine Anwendung während der Fahrt besonders wichtig und daher wird die vorgestellte Methode für einen Fahralgorithmus verwendet. Das „Template Matching“ und die „Mustererkennung“ sind sehr anfällig für wechselnde Lichtverhältnisse und veränderte Blickwinkel. Daher sind diese Varianten für das gewählte Fahrszenario nicht geeignet.

7 Fahralgorithmus

In Kapitel 0 wurden die Möglichkeiten Spuren und Schilder mittels digitaler Bildverarbeitung in einem Kamerabild zu erkennen beschrieben. Um das gewählte Szenario „Spurfolgen“ zu bewältigen gilt es die Daten aus der Bilderkennung mit anderen Daten zusammenzuführen und daraus eine Quer- und Längsregelung abzuleiten. Dazu muss ein Algorithmus entwickelt werden, der Entscheidungen zur Fahrgeschwindigkeit und Richtung trifft. Gleichzeitig ist in Hinblick auf eine spätere Erweiterbarkeit darauf zu achten alle Daten möglichst universell verfügbar zu halten. Nur so können diese für weitere Anwendungsfälle genutzt werden.

7.1 Softwareumgebung

Für die Entwicklung einer Softwareumgebung steht die Middleware „ROS“ zur Verfügung. „ROS“ unterteilt eine Softwareumgebung in mehrere Ebenen (siehe Abbildung 7-1):

Abbildung 7-1: Aufbau der ROS-Struktur

Die Workspace-Ebene beinhaltet als Ordnerstruktur alle weiteren Elemente eines „ROS“-Projektes. Eine zugrundeliegende Struktur wird durch die Linux-Infrastruktur „Catkin“ gebildet. Dies sorgt dafür, dass erstellte Ordner und ausführbare Dateien in dem Workspace-Ordner durch spezielle „ROS“-Befehle sichtbar und auch ausführbar sind. Die Packages-Ebene liegt unterhalb der Workspace-Ebene und dient als Struktur für einzelne Funktionen innerhalb eines Projektes. Für das vorliegende Beispiel wäre es denkbar, ein Package mit der Bezeichnung „Spurfolgen“ und entsprechend weitere Packages, wie „Einparken“ etc. einzuführen. Innerhalb eines Packages sind durch die „Catkin“ Routine eine Vielzahl von Unterordnern angelegt worden, die zum Beispiel Startroutinen und personalisierte Nachrichtenstrukturen definieren. Die wichtigste Funktionalität liegt mit den Nodes in dem Unterordner „scripts“. Diese Knoten bilden als Einzelskripte die Funktionalität eines Packages ab.

Für das vorliegende Projekt sind vorwiegend Knoten einzuführen, die Rohdaten verarbeiten und weiterleiten. Dazu stehen die Sensordaten der Kamera, der Ultraschallsensoren und der IMU bereit. Weiterhin müssen Knoten zur Verfügung stehen, die diese Daten zu einer Quer- und Längsführung verarbeiten und als Steuersignal an das Basisfahrzeug senden. Für die Kommunikation der Knoten untereinander müssen sogenannte Topics definiert oder verfügbare Standardtopics angepasst werden. Diese Datenthemen legen das Format der verschickten Daten fest und stellen mit ihrem Namen eine eindeutige ID, über die andere Knoten auf die Daten zugreifen können.

Aus diesen Anforderungen wurde ein Package entwickelt, dessen Knotenstruktur in Abbildung 7-2 zu sehen ist. Dieser Strukturplan zeigt in den ovalen Feldern die Bezeichnung der einzelnen Knoten und auf den Verbindungslinien die Benennung der verschickten Nachricht. Aufgrund der Komplexität des

dargestellten Sachverhaltes, wurde diese automatisch generierte Darstellung auf zwei Zeilen aufgetrennt.

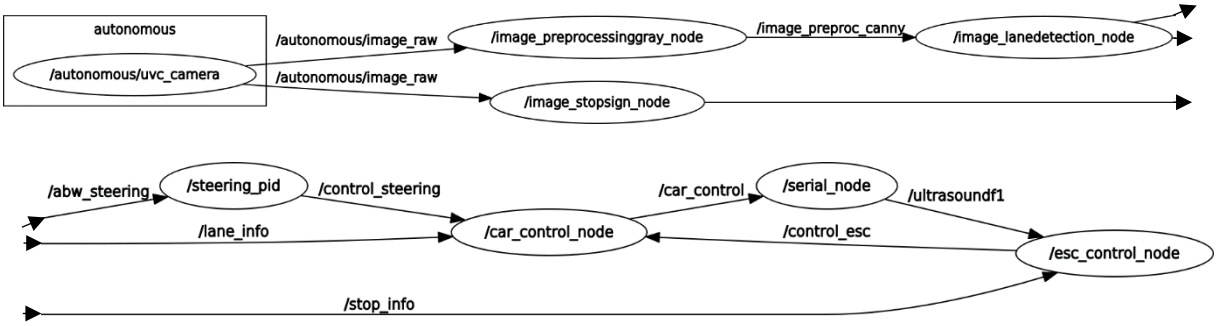


Abbildung 7-2: ROS-Knotenstruktur

Links oben ist der zentrale Kameraknoten (**„/autonomous/uvc_camera“**) zu sehen. Er bindet, über die bereits erwähnte GStreamer Pipeline, die Kameradaten in die ROS Umgebung ein. Die Bilddaten werden hierbei, bei einer Auflösung von 640x480 Pixeln mit 25 Hz im Format „MJPEG“ ausgelesen. Diese geringe Auflösung ist für eine Objekterkennung und die Menge an Daten ausreichend und kann auch im mobilen Einsatz auf einem „Nvidia Jetson“ leicht verarbeitet werden. Zwei Knoten nutzen die zur Verfügung gestellten Kameradaten. Zum einen nutzt der Knoten **„/image_preprocessinggray_node“** eine Schnittstelle zur Software „OpenCV“ für Filterung des Kamerabildes. Es findet eine Umwandlung in Graustufen, eine Farbfilterung, eine Binarisierung und die Kantenerkennung statt. Diese Vorgänge werden in Kapitel 0 im Detail beschrieben. Zum anderen verwendet der Knoten **„/image_stopsign_node“** die Kameradaten, um mithilfe der in Kapitel 6.2.3 beschriebenen Classifier-Technik nach Stoppschildern zu suchen. Die Information über eine Erkennung wird durch die Topic „/stop_info“ weitergeleitet. Für die Verwendung der Bilddaten mit „OpenCV“ ist die Umwandlung der „ROS“-Daten durch eine Bildbibliothek notwendig. Dies wird durch folgenden Befehl erreicht:

$$cv\_image = self.bridge.imgmsg\_to\_cv2(data, desired\_encoding = "passthrough") \quad (7.1)$$

Die umgewandelten Bilddaten der Kantenerkennung werden durch den Knoten **„/image_lanedetection_node“** zur Erkennung der Fahrspur genutzt. Das Thema „/lane_info“ enthält Daten zur Spurposition im Bild, dem Kurvenradius und ob eine Erkennung erfolgreich ist. Die genaue Verarbeitung der Spurdaten wird in Kapitel 7.3 beschrieben. Ebenso erfolgt in dem genannten Kapitel eine Beschreibung des Reglerknotens **„/steering_pid“.**

Auf der rechten Seite der Abbildung 7-2 ist der Knoten **„/esc_control_node“** dargestellt. Dieser ist für die Längsregelung des Demonstrators verantwortlich. In dem beschriebenen Anwendungsszenario wird, zur Vereinfachung an dieser Stelle lediglich zwischen Fahren und Halten unterschieden. Die Fahrzeuggeschwindigkeit wird nicht adaptiv geregelt. Als Eingangssignale erhält der Knoten Daten des Ultraschallsensors und der Stoppschilderkennung. Die Entscheidung, ob das Fahrzeug Fahren oder Halten soll, wird durch „if“-Abfragen der einzelnen Daten und durch das Setzen von Flags realisiert. Werden

Objekte in einer Entfernung kleiner als 50cm vor dem Fahrzeug erkannt wird eine Flag auf den Status True gesetzt. Diese Entfernung ist als Sicherheitsabstand ausreichend, um eine Vermeidung von Kollisionen immer zu gewährleisten. Ebenso wird eine Flag auf den Status True gesetzt, wenn eine Stoppschilderkennung erfolgt ist. Ist eine der beiden Flags aktiv, sendet der beschriebene Knoten das Signal Halten. Hierbei wird der PWM-Wert der Neutralstellung des Motors verwendet (1500). Die Unterscheidung erfolgt durch eine „if"-Abfrage mit „und-nicht" Unterscheidung wie folgt:

$$if\ not\ self.stopee1\ and\ not\ self.stopee2: \quad (7.2)$$

Das Signal für Fahren lautet „1550". Dieser PWM-Wert bedeutet für den Motorregler langsames Vorwärtsfahren und stellt die kleinstmögliche Geschwindigkeit ein.

Die Daten des „/esc_control_node" und des „/steering_pid" werden im **„/car_control"** Knoten zusammengefasst und als gebündelte Nachricht („/car_control") an das Fahrzeug gesendet. Diese Nachricht enthält den geforderten Winkel des Servomotors und die PWM-Vorgabe für den Motorregler. Zusätzlich findet hier eine Umrechnung des Lenkwinkels aus Werten des PID-Reglers in real verfügbare Werte statt.

Ein zentrales Element ist der Knoten **„/serial_node"**. Dieser repräsentiert den Kommunikationszweig zu dem Arduino und damit die Hardwareverbindung des Systems. Der Name wurde gewählt, da der Arduino über eine serielle USB-Schnittstelle Daten verschickt und somit durch eine Zusatzerweiterung für Python eingebunden werden kann. Die Software auf dem Arduino liest Ultraschalldaten ein, rechnet diese in eine Entfernung um und versendet sie als „ROS"-Nachricht („/ultrasoundf1"). An dieser Stelle werden Daten zur Steuerung des Fahrzeuges empfangen. Der gesamte Quelltext des Arduino Knotens ist im Anhang unter Anhang 1 zu finden.

Alle gezeigten Nachrichtenthemen lassen sich von einer Vielzahl an Knoten gleichzeitig nutzen. Bei der Generierung der Nachrichtentypen wurde auf eine allgemeingültige Konzeption geachtet. Somit lassen sich mit der vorgestellten „ROS"-Umgebung leicht weitere Funktionen, durch eine Wiederverwendung der gezeigten Knoten und Erweiterung durch eigene, implementieren. Eine Einbindung der Daten der IMU findet für das vorliegende Konzept nicht statt. Die Daten der Beschleunigung werden in diesem einfachen Ansatz der Quer- und Längsregelung nicht berücksichtigt. Allerdings ließen sie sich leicht über einen weiteren Knoten einbinden.

Die Skripte der Knoten liegen als ausführbare Python-Programme in dem Unterordner „scripts". Durch die Einbindung des „ROS"-Befehls „roslaunch" wird ein komfortabler Start der gesamten Anwendung erreicht. Durch Aufrufen des Befehls „roslaunch autonomous lanefollowergray.launch" wird die Anwendung des Spurfolgens gestartet und alle benötigten Skripte automatisch aufgerufen. Der genaue Quelltext ist im Anhang unter Anhang 2 zu finden.

7.2 Aufbau des Demonstrators

Für die Anwendung der vorgestellten Softwarekomponenten und für die Entwicklung einer Querregelung ist der Aufbau eines realen Demonstrators unerlässlich. Zu diesem Zweck wurde, auf Grundlage des Fahrzeugkonzeptes, eine Plattform aus Acrylglas präpariert, die alle weiteren Hardwarekomponenten aufnimmt. Der Entwicklungsrechner, der Arduino, der USB-Hub und der Akkumulator sind direkt auf der Plattform befestigt. Zudem wurde eine zweite Ebene als Basis für die Befestigung der Kamera im vorderen Bereich der Platte montiert. Die Position ist auf den Blickwinkel der Kamera zur Straße angepasst. Der Winkel um die Y-Achse (bezogen auf ein Standardfahrzeug-KOS) lässt sich nachträglich justieren. Dabei ist die Ausrichtung ein Kompromiss zwischen der Erfassung der Spur im Nahbereich und der Möglichkeit, Schilder in weiterer Entfernung zu detektieren. Die Plattform besitzt vier Aussparungen, die der Befestigung auf dem Basisfahrzeug dienen. Eine Verbindung der benötigten Anschlüsse ist in einem Kabel mit Steckerverbindung zusammengefasst. Dadurch ist eine einfache Demontage, für Revisions- und Wartungszwecke, möglich. Das Ergebnis des Aufbaus ist in Abbildung 7-3 zu sehen.

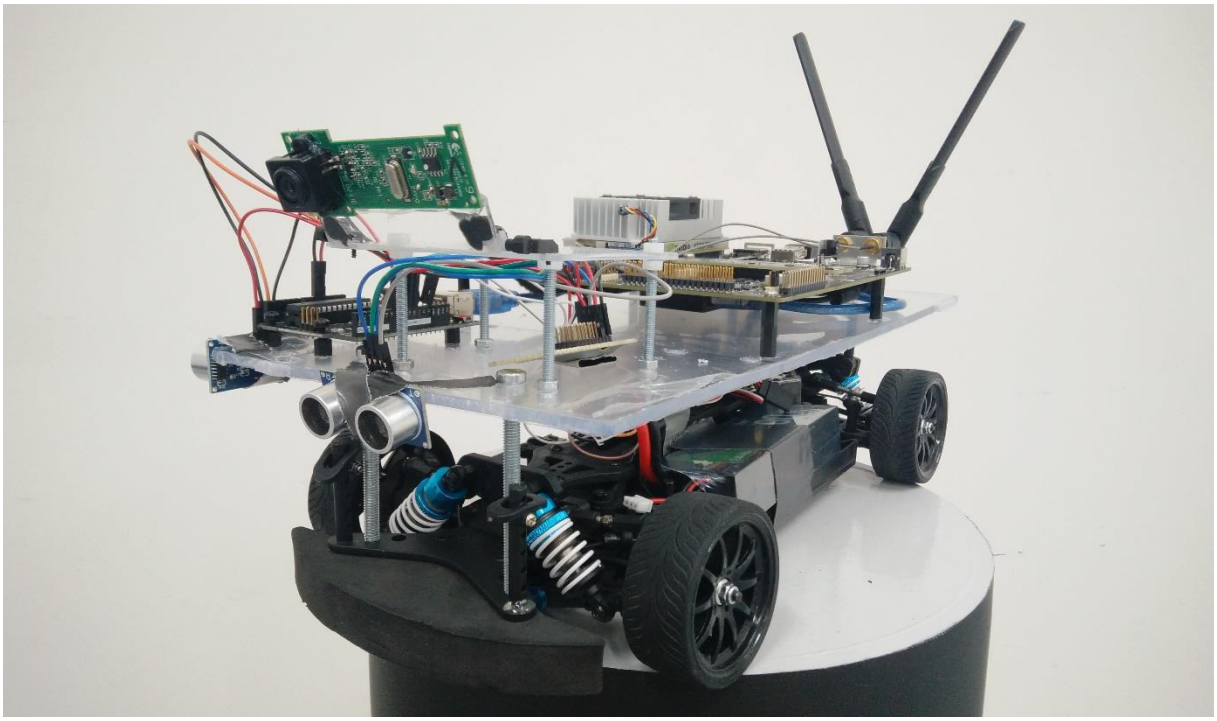


Abbildung 7-3: Demonstrator Probeaufbau

Weiterhin wurde das Fahrwerk, im Rahmen der vorhandenen Möglichkeiten, auf das höhere Gewicht eingestellt. Durch eine Erhöhung der Federsteifigkeit kann das Fahrzeugniveau, auch bei montierter Plattform, konstant gehalten werden.

7.3 Querregelung

Um einen Parcours autonom zu bewältigen, ist die Kontrolle der Lenkung und die Umsetzung in eine Querregelung durch einen Computer entscheidend. Dafür müssen die Daten der Spurerkennung interpretiert und vom Computer in Lenkbefehle umgesetzt werden. Für eine Querführung muss zum einen die Lage der Spur und zum anderen die Lage des Fahrzeuges relativ zur Spur bekannt sein.

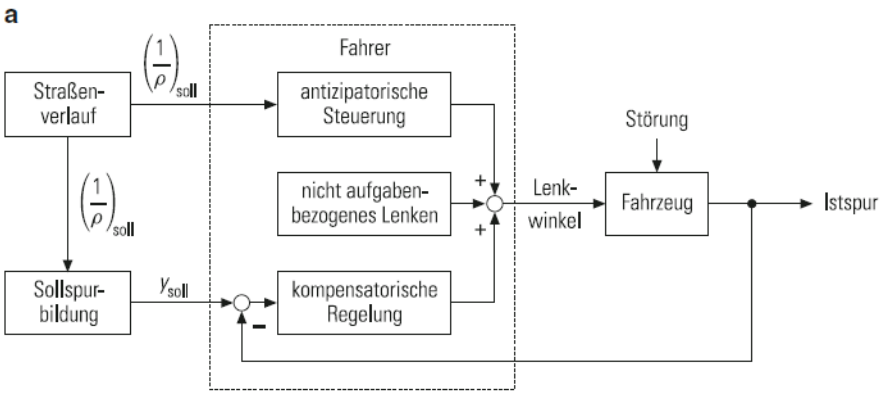


Abbildung 7-5: Modell der Istspurfindung [63]

Abbildung 7-4 zeigt das Modell der Findung einer Istspur durch den Fahrer [64].Der Straßenverlauf gibt einen bestimmten Krümmungsradius vor und der Fahrer reagiert durch Veränderung des Lenkwinkels antizipatorisch. Dazu kommen kompensatorische Einflüsse der angepassten Geschwindigkeit bei Kurvenfahrt und ergeben einen Solllenkwinkel. Störungen auf das Fahrzeug, wie Seitenwind und unebener Straßenbelag, haben einen Einfluss auf die Istspur und müssen ausgeglichen werden. In dieser Arbeit wird eine vereinfachte Betrachtungsweise angenommen, da kein Fahrer Einfluss auf das Modell nehmen kann. Weiterhin kann der Regelkreis als linear betrachtet werden, da eine Kursänderung direkt aus einer Änderung des Lenkwinkels resultiert. Alle Regelaufgaben werden durch ein PID-Modul in der „ROS“-Umgebung übernommen.

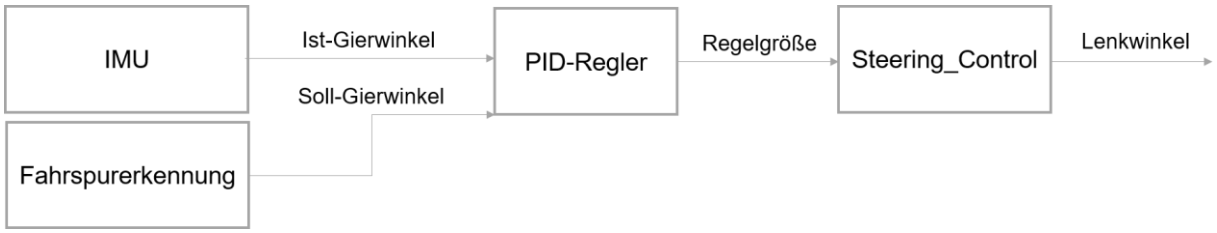


Abbildung 7-4: Querregelung Variante 1

Für eine automatisierte Querregelung sind vereinfacht zwei unterschiedliche Regelvarianten zu unterscheiden. Variante 1 ist von dem aktuellen Vorgehen in Serienfahrzeugen abgeleitet. Die Fahrspurerkennung berechnet durch den Krümmungsradius der Straße und der aktuellen Geschwindigkeit einen Soll-Gierwinkel des Fahrzeuges (siehe Abbildung 7-5). Aus den aktuellen Fahrzeugdaten wird der momentane Gierwinkel ausgelesen und ein Regler versucht die Differenz durch das Stellen des Lenkwinkels auszugleichen. Eine weitere Version dieser Variante besteht in der Berechnung einer Sollquerbeschleunigung und einer Regelung mithilfe der aktuellen Querbeschleunigung des Fahrzeuges [16]. Das beschriebene Vorgehen setzt zum einen ein genaues Fahrzeugmodell (meist als Einspurmodell vereinfacht) und zum anderen genaue Beschleunigungs- und Winkeldaten voraus. Die Daten der Querbeschleunigung und des Gierwinkels aus den Rohdaten der IMU stehen grundsätzlich im gezeigten Demonstrator zur Verfügung. Jedoch ist an dieser Stelle eine aufwendige Filterung der Rohdaten notwendig, um verlässliche Werte zu erhalten. Zusätzlich ist die Modellbildung sehr aufwendig und nicht im Umfang dieser Arbeit vorgesehen. Aus den genannten Gründen wurde eine zweite Variante aus den vorhandenen Daten entwickelt.

Abbildung 7-6: Querregelung Variante 2

Abbildung 7-6 zeigt einen Entwurf für eine einfach aufgebaute Regelstrecke. Die Fahrspurerkennung liefert die Daten der aktuellen Fahrspur. Somit sind die Position der rechten und linken Fahrbahnmarkierung sind als Ganzzahlwerte in Pixeln bekannt. Aus diesen beiden Werten lässt sich die Spurbreite und die aktuelle Spurmitte berechnen. Unter der Annahme, dass die Kamera in der Fahrzeugmitte montiert ist, lässt sich nach:

$$spur.abw = (w/2 - (spur.right + spur.left)/2) \quad (7.3)$$

die aktuelle Abweichung von der Fahrspurmitte berechnen. Diese Abweichung wird als aktueller Status für einen einfachen PID-Regler verwendet. Dieser Reglertyp kann durch Variation der einzelnen Reglerantеile leicht angepasst und auf die aktuellen Gegebenheiten abgestimmt werden. Des Weiteren ist für „ROS“ eine einfach konfigurierbare PID-Bibliothek im Wiki der ROS Foundatio“ verfügbar.

Die Sollabweichung wird ebenso durch die Fahrspurerkennung berechnet und in einem ersten Schritt dauerhaft als Null ausgegeben. Der Regler kann seine Ausgabe in einem Bereich von -20 bis +20 variieren. Dies entspricht den möglichen Lenkwinkel des Basisfahrzeuges von 70° bis 110° und wird in dem Knoten „/steering_control“ durch die Addition der Zahl 90 umgerechnet. Um einem möglichen nervösen Regelverhalten vorzubeugen wurde eine Funktion in der Fahrspurerkennung integriert, die die aktuelle Abweichung in einem Bereich von -10 bis +10 Pixel auf den Wert null setzt. So wird dem Fahrzeug ein Fahrschlauch vorgegeben, der dem mittleren Spurbereich entspricht.

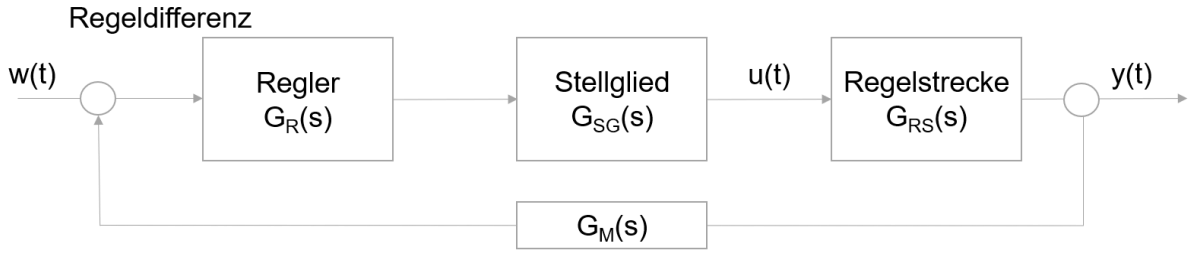


Abbildung 7-7: Linearer Regelkreis

Abbildung 7-7 zeigt die Darstellung eines linearen Regelkreises. $W(t)$ bezeichnet die Führungsgröße. Diese wird im vorliegenden Fall auf eine Abweichung von null festgesetzt und besitzt somit keine Zeitabhängigkeit. Die Regeldifferenz berechnet sich aus der Abweichung des Sollwertes der Führungsgröße und dem aktuellen Wert der Messübertragungsfunktion $G_M(s)$ [65]. Hier kann ein Einfluss der Messung auf die Messgröße vernachlässigt werden, da die Bildaufzeichnung die Abweichung nicht beeinflusst und somit die Abweichung durch Formel 7.3 zu bestimmen ist. Das Stellglied wird durch das Lenkungssystem repräsentiert. Eine Regleränderung wird in einen Lenkwinkel umgerechnet und kann linear in eine Richtungsänderung übersetzt werden. Der volle Lenkeinschlag (+/- 20° Lenkwinkel von

der Ausgangsstellung) ergibt einen Fahrradius von 1,75m. Eine Geradeausfahrt wird durch eine Lenkwinkelstellung von 90° erreicht. Die Änderung der Abweichung steht somit in Abhängigkeit der Gierwinkeländerung (Giergeschwindigkeit). Dieses Verhältnis kann auch als Gierverstärkung bezeichnet werden und berechnet sich für eine direkte Lenkübersetzung vereinfacht nach folgender Formel:

$$\delta_H = \frac{l}{R} + EG * a_y \qquad (7.4)$$

Der Ackermannlenkwinkel (*l/R*) bildet das Verhältnis von Radstand zu Kurvenradius (Abstand des Fahrzeugschwerpunktes zu Momentanpol des Fahrzeuges). Der Eigenlenkgradient „*EG*“ kann für ein neutrales Fahrverhalten als null angenommen werden, womit die Querbeschleunigung „a_y“ wegfällt [64]. Bei einem Radstand von 257mm ergibt sich somit ein maximaler Ackermannwinkel von 0,1469 (ca. 8°). Die Regelstrecke ergibt sich aus der Spurerkennung. Diese ist von vielen Faktoren, wie der aktuellen Lichtsituation und der Filterung der Erkennung durch verschiedene Algorithmen, abhängig. Eine genaue Bestimmung der Übertragungsfunktion bedarf einer Bestimmung des Einspurmodells des Fahrzeuges und die Kenntnis der Lenkgeometrie. Für eine erste Auslegung der Regelstrecke kann für das vorliegende Modell auf diese Berechnung verzichtet werden und die Reglerparameter heuristisch bestimmt werden.

Für die Abstimmung des Reglers und zur Fehleranalyse wurde ein Testfeld in einer überdachten Halle aufgebaut. In Abbildung 7-8 ist die 2,5m lange Gerade der Teststrecke zu erkennen. Die Spur hat eine ungefähre Breite von 25 cm. Dies ist ausreichend breit für das Fahrzeug und gleichzeitig schmal genug, um eine dauerhafte Spurerkennung zu gewährleisten. Als Fahrbahnbegrenzung wird dunkelgraues Klebeband verwendet. Dies bedingt eine Anpassung der Farbfilterung der Spurerkennung. Ebenso wurden der Kamerawinkel und die Parameter der „Bird View“-Transformation an die veränderte Position angepasst.

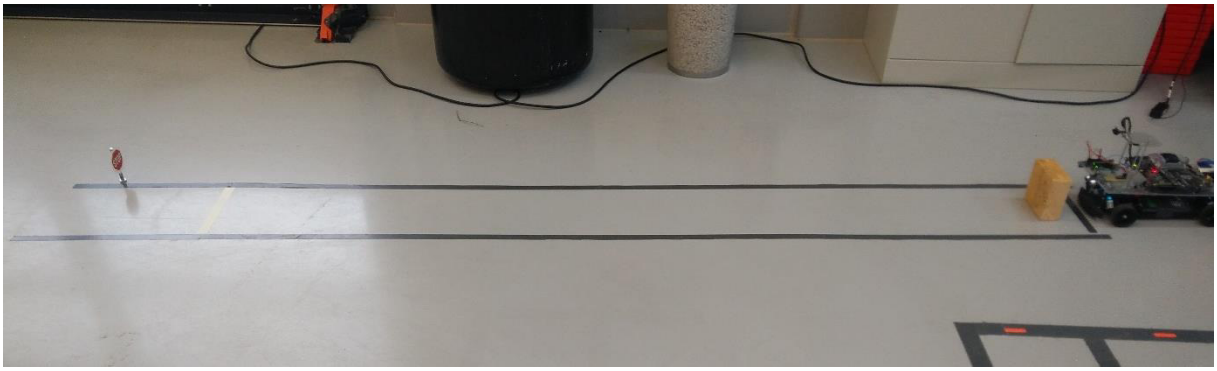


Abbildung 7-8: Teststrecke Querregelung 1

Für die Abstimmung des Reglers wurde mit einer reinen P-Konfiguration begonnen und der I- und D-Anteil auf null gesetzt. Zur Datenaufzeichnung der „ROS“-Daten wurde der Befehl „rosbag“ verwendet. Als Trigger für das Ende der Teststrecke dient das Stoppschild (siehe Abbildung 7-8 links). Trotz des recht einfachen Testszenarios konnte die Strecke in nur ca. 10% der Fälle eigenständig durch Fahrzeug bewältigt werden. Dazu zeigte die Regelung eine ungenügende Kompensation der Abweichung und

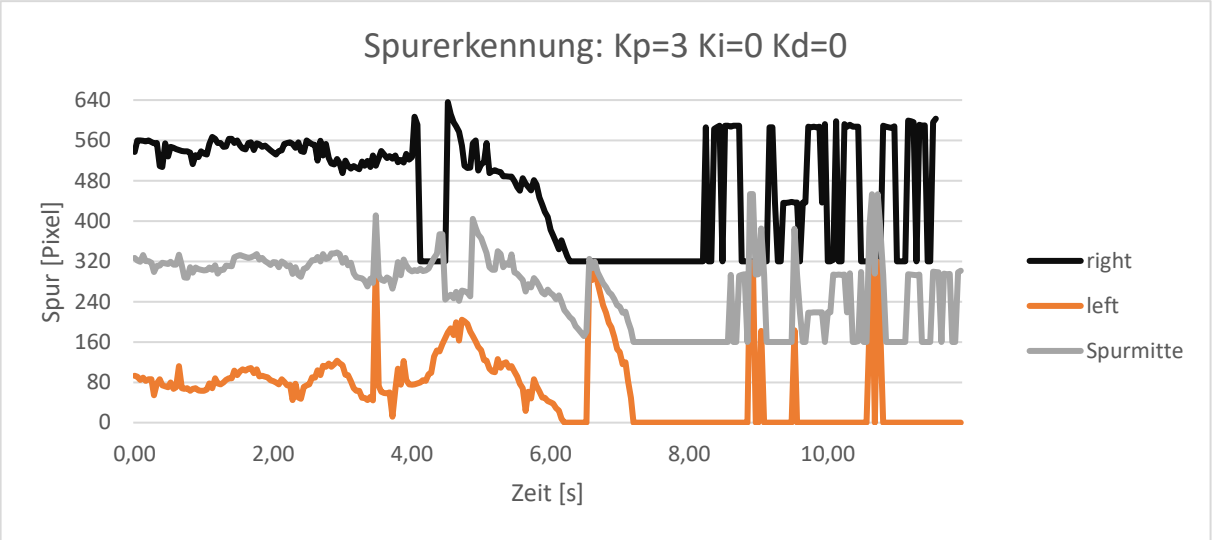


Abbildung 7-9: Daten der Spurerkennung - Fehleranalyse

war durch stetige Überschwinger geprägt. Abbildung 7-9 zeigt die Aufzeichnung der Spurerkennung mit der Position der linken und rechten Spur im Bild der Kamera und der daraus berechneten Spurmitte. Nach ca. sechs Sekunden verlässt das Fahrzeug die Spur. Auffällig zeigt sich die Erkennung der linken Spur, die bei diesem Zeitstempel einen Ausschlag in die entgegengesetzte Richtung aufweist. Ebenso sind eine Reihe von Fehlerfassungen der Spur zu erkennen. Diese werden durch starke Ausschläge der Linien deutlich.

Grundsätzlich zeigt Abbildung 7-10 ein gewünschtes Reglerverhalten. Hierbei ist die Abweichung von der Spurmitte als rote Linie dargestellt und der daraus resultierende Lenkwinkel in grün. Es ist zu erkennen, dass der Regler grundsätzlich das richtige Verhalten aufweist und einer Abweichung entgegenwirkt. Trotz dessen ist das Fahrverhalten als ungenügend zu beschreiben und daher wurde eine zweite Version der Reglerstruktur, mit verbesserten Algorithmen, entwickelt.

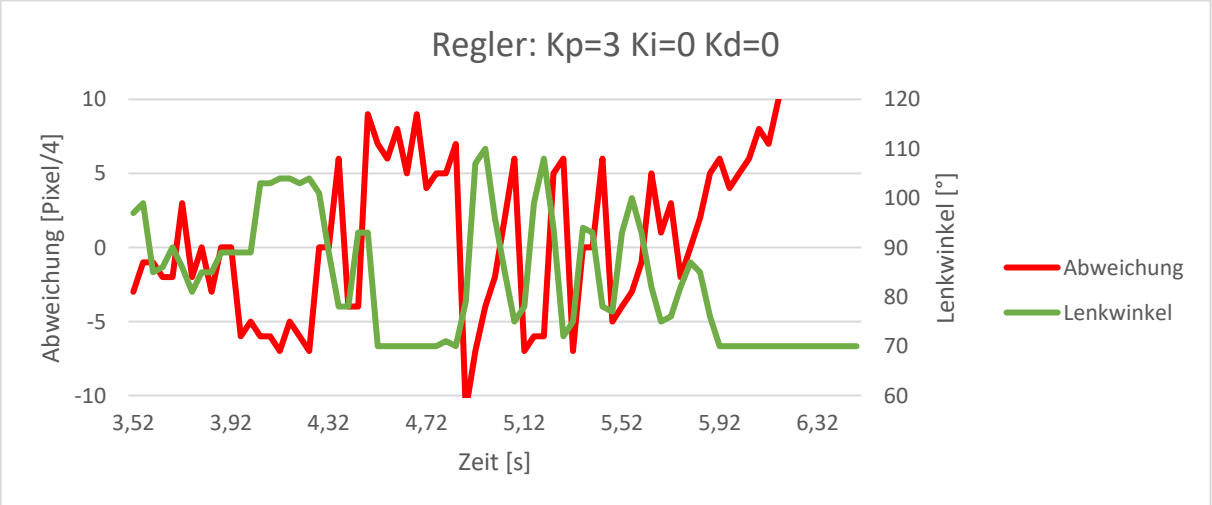


Abbildung 7-10: Abweichung und resultierender Lenkwinkel

Eine Analyse der Daten ergab einen Fehler in der Bildtransformation, der zu Fehlerkennungen am äußersten rechten und linken Bildrand geführt das. Diese Fehlerkennung war für das häufige Abkommen des Fahrzeuges von der Spur verantwortlich.

Um das Problem der Fehlerkennungen zu beheben wurde die Fahrbahnmarkierung auf ein grünes Klebeband umgestellt (siehe Abbildung 7-11). Diese Farbe bietet eine bessere Differenzierung vom Hallenboden und erlaubt eine genauere Erkennung. Durch diese Umstellung konnte die Quote der Fehlerkennung erheblich reduziert werden und das Fahrverhalten zeigt sich insgesamt stabiler.

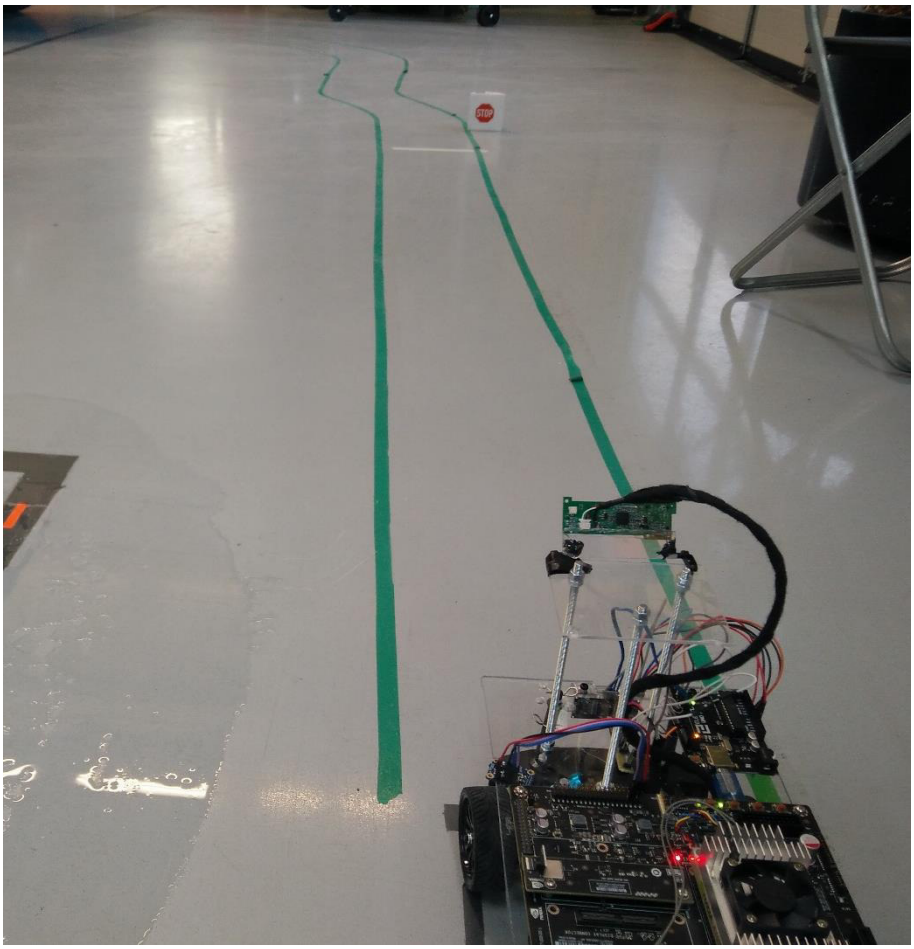


Abbildung 7-11: Überarbeitete Teststrecke Querregelung

Eine adaptive Regelung des maximalen Lenkwinkels in Abhängigkeit des Kurvenradius erlaubt ein stabiles Verhalten des Fahrzeuges auf geraden Streckenabschnitten bei gleichzeitiger Beibehaltung des maximalen Kurvenradius. Bei einem Kurvenradius von >= 5m wird der Lenkwinkel auf die Hälfte und damit einen Bereich von 80° bis 100° beschränkt. Somit ist der Regelbereich deutlich eingeschränkt und Überschwinger können so effektiv verhindert werden.

Eine Erkennung des Abkommens von der Spur ermöglicht ein Gegensteuern. Somit ist es dem Fahrzeug möglich eigenständig die Spur wieder zu finden. Ein Nichterkennen der Spur zeichnet sich durch den Pixelwert null auf der linken und einen Wert von 320 auf der rechten Seite aus. Wird einer dieser Werte erkannt und weicht er zusätzlich mehr als 50 Pixel vom vorigen Wert ab, wird eine Flag gesetzt.

Ist außerdem die Spurbreite kleiner als 200 Pixel, ist das Fahrzeug von der Spur abgekommen. Als Reaktion wird die Sollabweichung auf 200 (Abweichung nach rechts) oder -200 (Abweichung nach links) gesetzt. Dies führt zu einem Gegenlenken und Wiederfinden der Spur. Das genannte Verfahren hilft insbesondere das Kurvenverhalten deutlich zu verbessern.

Ein weiterer Algorithmus filtert die Erkennung der rechten und linken Markierung. Durch eine Begrenzung der maximalen Abweichung auf fünf Pixel zwischen zwei Erkennungen, können einerseits eine sprunghafte Regelung verhindert und andererseits Fehlerkennungen allgemein gefiltert werden. Des Weiteren hilft dieser Algorithmus, die Regelung in Kurvenbereichen zu verbessern. Durch die Veränderung des Blickwinkels bei Kurvenfahrt ist es möglich, dass eine Fahrbahnmarkierung kurzzeitig außerhalb des Erkennungsbereiches der Kamera liegt und somit nicht erfasst wird. Dies führte zu einem sprunghaften Reglerverhalten und kann durch die Filterung verhindert werden. Zusätzlich wird ein gewichteter gleitender Durchschnitt vierter Ordnung verwendet, um Sprünge in der Erkennung zu reduzieren.

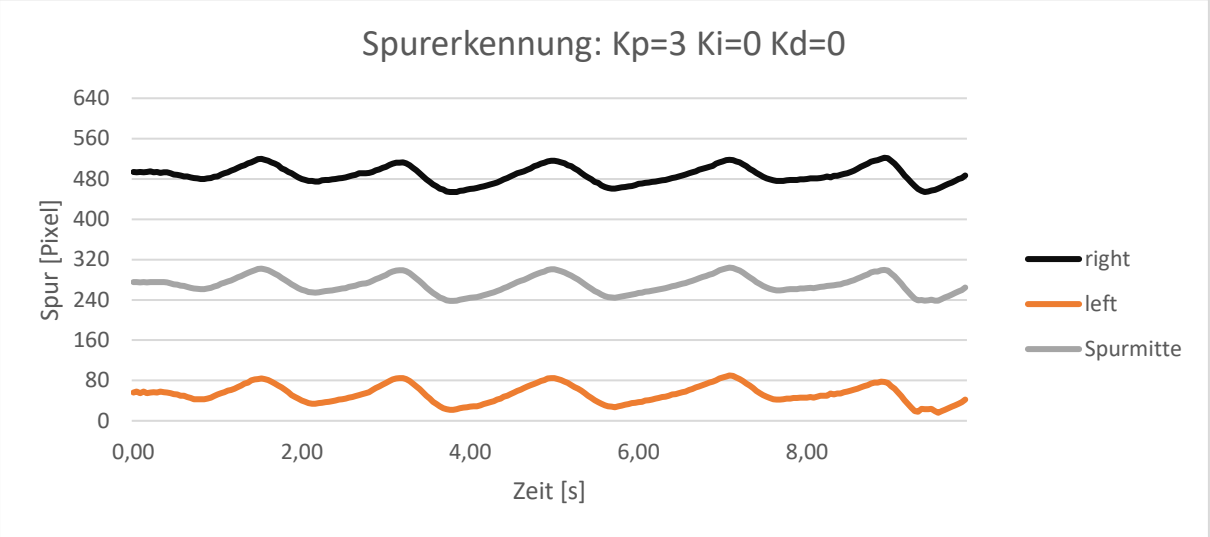


Abbildung 7-12: Daten der Spurerkennung - KP=3, KI=0, KD=0

Für die Abstimmung des Reglers wurde mit den bekannten Reglerparametern K_P=3, K_I=0 und K_D=0 als Referenzwert begonnen. Die Teststrecke besitzt wiederum eine Länge von 2,5m und das Stoppschild am Ende der Bahn wird als Trigger zur Beendigung der Messung benutzt. Abbildung 7-12 zeigt die Daten der Spurerkennung. Die erkannte Spur wird in Abhängigkeit der Zeit aufgetragen. Insgesamt zeigen sich keine Ausreißer der Erkennung der Fahrbahnmarkierung. Dies ist zum einen auf die verbesserte Detektion an sich (veränderte Spur, verbesserte Farbfilterung) und zum anderen auf die erfolgreiche Filterung und Glättung der Erkennung durch die vorgestellten Algorithmen zurückzuführen. Im Vergleich zu Version 1 der Querregelung zeigt sich eine starke Verbesserung des Fahrverhaltens. Die Fähigkeit die Spur zu halten konnte stark verbessert und außerdem der Anzahl von sprunghaften Lenkwinkeländerungen verringert werden. Die wellenförmige Positionsveränderung der Spur in der Abbildung ist auf die Einflüsse des Reglers zurückzuführen und kann durch eine weitere Abstimmung der Parameter verbessert werden. Das Verhalten des Reglers kann im Detail in Anhang 3 nachvollzogen

werden. Die Summe der Abweichung beträgt insgesamt 1271 Pixeln. Dies entspricht einer durchschnittlichen Abweichung von 4,65 Pixeln pro Bild.

Da der P-Anteil des Reglers bereits an der Grenze eines stabilen Systems liegt und außerdem die Geschwindigkeit der Regelung als gut zu bezeichnen ist, wurde im Anschluss der I-Anteil erhöht. Ziel ist es die Regelabweichung insgesamt zu verringern und das Fahrverhalten weniger ruckartig zu gestalten. Dazu wurden verschiedene Parameterkombination getestet. Ein stabiler Kompromiss aus Geschwindigkeit und Verringerung der Abweichung ist durch die Parameter K_P=3, K_I=0, K_D=0 gegeben.

Die Erkennung der Spur ist in Abbildung 7-13 zu sehen. Im Vergleich zu Abbildung 7-12 ist auf den ersten Blick keine starke Verbesserung des Regelverhaltens zu erkennen. Jedoch konnte die Gesamtregelabweichung auf 937 Pixel oder 3,87 Pixel pro Erkennung reduziert werden. Dies entspricht einer Abnahme der Spurabweichung von ca. 26,2 %. Die Regelung der Lenkung erfolgt insgesamt in etwas gröberen Schritten, was jedoch das Fahrverhalten nicht negativ beeinflusst (siehe Anhang 3).

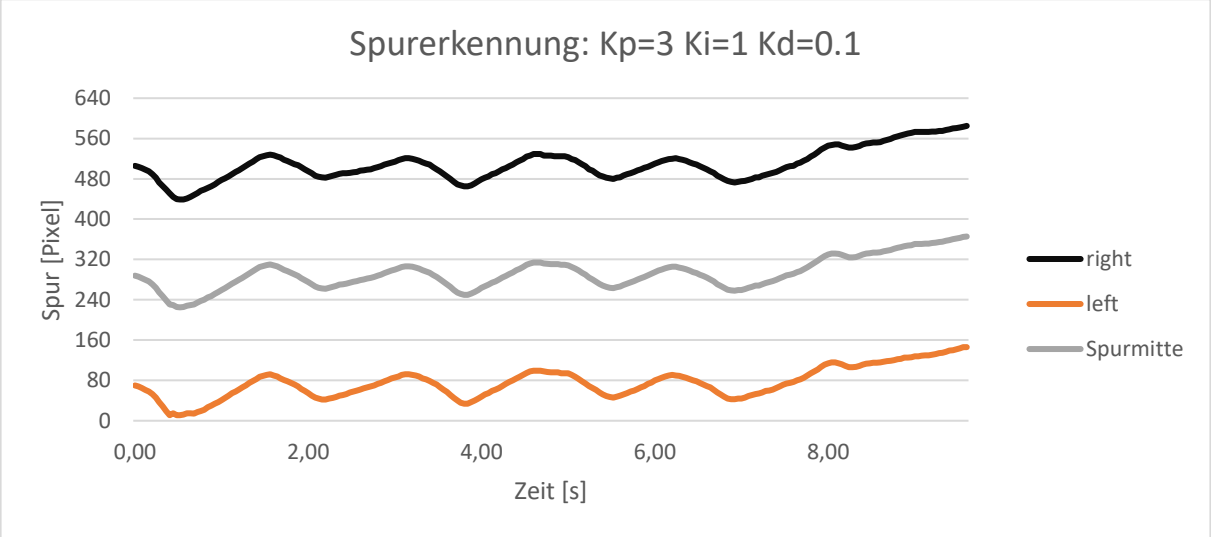


Abbildung 7-13: Daten der Spurerkennung - KP=3, KI=1, KD=0.1

8 Zusammenfassung

Das Ziel dieser Arbeit lag in der Entwicklung eines erweiterbaren Demonstrators. Dabei lag ein besonderer Fokus auf der Auswahl geeigneter Sensoren zur Umfelderkennung und der Implementierung von Software zur internen Kommunikation und zur Bildverarbeitung.

Es konnte gezeigt werden, dass durch die Verwendung von Open-Source Software ein funktionsfähiges selbstfahrendes Fahrzeug entworfen und gebaut werden kann. Diese Arbeit zeigt, wie mithilfe von günstigen Ultraschallsensoren und einer USB-Kamera eine Umfelderkennung geschaffen werden kann, die bereits die Grundlage für autonome Fahrfunktionen schafft. Die gezeigten Ansätze der Spur- und Schildererkennung erweisen sich dabei auch im realen Testbetrieb für eine Quer- und Längsregelung als ausreichend robust. Eine Spurerkennung auf Grundlage des „Inverse Perspective Mapping“ ist hierzu die geeignete Wahl. Sowohl die Erkennungsrate, als auch -geschwindigkeit sind als gut zu bewerten. Ein limitierender Faktor ist der Blickwinkel der verwendeten Kamera. Dadurch ist die Erkennung in Kurven eingeschränkt. Der vorliegende Algorithmus kann dies durch Filterung der Daten umgehen und ermöglicht somit auch in Kurven ein Halten der Spur. Für eine weitere Verbesserung ist ein Wechsel der Kamera ein denkbarer Ansatz. Der Fahralgorithmus konnte durch die Filterung der Daten und das Abstimmen des Lenkungsreglers entscheidend verbessert werden. Durch die Anpassung der Reglerparameter hat sich Abweichung des Fahrzeuges von der Spurmitte um über 26% verringert. Eine genauere Anpassung kann in Zukunft durch das Aufsetzen eines mathematischen Regler- oder kompletten Fahrzeugmodells erfolgen. Auch eine Umstellung auf die Regelung des Gierwinkels kann in diesem Zusammenhang für Verbesserung sorgen.

Tabelle 8-1: Übersicht der Projektkosten

Artikel	Preis	Bestelldatum	Lieferdatum	Link
Fahrzeug 1:10	178,94 €	11.10.2017	16.10.2017	https://www.model
Akku LiPo 7,4V	29,95 €	11.10.2017	16.10.2017	https://www.model
Ladegerät	24,95 €	11.10.2017	16.10.2017	https://www.model
Nvidia Jetson	350,00 €	10.10.2017	11.10.2017	http://www.nvidia.d
Arduino Board	11,82 €	19.10.2017	20.10.2017	https://www.amazo
USB-Hub	7,39 €	23.10.2017	26.10.2017	https://www.amazo
Akku LiPo 11,1V	23,95 €	13.11.2017	15.11.2017	https://www.amazo
Ultraschallsensoren	9,99 €	13.11.2017	15.11.2017	https://www.amazo
Bosch IMU	32,19 €	13.11.2017	20.11.2017	https://www.amazo
Ladeadapter	6,40 €	13.11.2017	16.11.2017	https://www.amazo
Spannungsadapter	6,97 €	13.11.2017	16.11.2017	https://www.amazo
Grundplatte	9,99 €	16.11.2017	16.11.2017	Hagebaumarkt
Hallsensor	1,68 €	06.02.2018	15.02.2018	https://www.amazo
Magnete	3,49 €	06.02.2018	08.02.2018	https://www.amazo
Karosseriehalter	9,19 €	06.02.2018	08.02.2018	https://tamico.de/k
Summe:	706,90 €			

Tabelle 8-1 zeigt eine Übersicht der entstandenen Kosten. Dabei sind die größten Komponenten dieser Auflistung, die Fahrzeugbasis und der Entwicklungsrechner „Nvidia Jetson“. Diese dienen als Grundlage aller weiteren Funktionen und sind damit von zentraler Bedeutung. Im Betrieb hat sich der Rechner als ausreichend leistungsstark gezeigt. Auch für weitere Funktionen der digitalen Bildverarbeitung stehen genug Reserven der Rechenleistung zur Verfügung. Die Energieeffizienz dieser Konfiguration hat sich als großer Vorteil erwiesen. Im Dauerbetrieb reicht der verwendete Akkumulator für mehr als 3 Stunden Fahrzeit. Insgesamt wurden nur ca. 36% des Gesamtbudgets ausgenutzt. Dies spricht für den effizienten Umgang mit den gegebenen Ressourcen und lässt Spielraum für weitere Verbesserungen auf der Hardwareseite.

Die implementierte Struktur in „ROS“ ermöglicht es unkompliziert weitere Fahrfunktionen zu implementieren. Dies kann einerseits auf Grundlage der bereits verfügbaren Daten geschehen oder durch das Einbinden von neuen Sensoren in die bestehende Struktur. Durch den modularen Aufbau der Software ist die Erweiterung und auch das Debugging von Funktionen problemlos möglich. Somit bietet das gezeigte Fahrzeugkonzept die besten Voraussetzungen für die Ausbildung von Studierenden und Jungingenieuren. Grundsätzliche Prinzipien der Bildverarbeitung und der Erarbeitung von Steuerungsalgorithmen können neu beigebracht oder das Wissen vertieft werden. Das Erlernte bietet die besten Voraussetzungen um die zukünftigen Herausforderungen der Automobilbranche zu meistern und die Entwicklung des autonomen Fahrens voranzutreiben.

9 Zukünftige Möglichkeiten

Die vorliegende Arbeit hat einen funktionierenden Demonstrator geschaffen und eine erste Fahrfunktion implementiert. Dabei sind im Verlauf der Entwicklung eine Reihe von Hindernissen aufgetreten, die Raum für zukünftige Verbesserungen lassen. Das folgende Kapitel soll einen Überblick über mögliche Verbesserungsansätze der vorhandenen Funktionen verschaffen, aber auch aufzeigen, was künftig mit der vorhandenen Hardware möglich sein kann.

Für den Applikationsprozess neuer Funktionen ist die Datenaufzeichnung und deren Analysewichtigste Schritt. Eine mögliche Verbesserung besteht in der Erweiterung des Entwicklungsrechners durch einen CAN-Anschluss und die Ausgabe aller in „ROS“ aufgezeichneter Daten. Dadurch entsteht zum einen die Gelegenheit live Daten einfacher zu überwachen und weiterhin diese auch vereinfacht aufzuzeichnen. Bisher ist die Aufzeichnung durch einen „ROS“-Befehl durchaus darstellbar, jedoch existiert keine Standardroutine für diesen Vorgang. Zusätzlich entsteht durch das Vorhandensein eines CAN-Anschlusses eine breitere Schnittmenge mit anderen Projekten der EVOMOTIV GmbH. Da der Umgang mit Bussystemen in der heutigen Entwicklung von Fahrzeugen von zentraler Bedeutung ist, kann die mögliche Schnittstelle als Lernplattform und zum Test weiterer Projekte dienen.

Eine andere Chance Synergieeffekte mit weiteren Projekten zu erzielen, besteht in dem Ausbau des Systems durch einen GPS-Empfänger. Diese Einbindung kann entweder direkt über eine Erweiterung des Entwicklungsrechners oder durch ein Zusatzmodul für den Arduino vorgenommen werden. Durch die Fähigkeit der Ortsbestimmung ist eine Einbindung in das Projekt „Connected Car“ möglich. Dieses hat zum Ziel durch mobile Datenkommunikation und mithilfe der Ortsbestimmung ein Frühwarnsystem für eventuelle Gefahrensituationen zu etablieren. Eine Erweiterung dieses Projektes auf den vorgestellten Demonstrator ermöglicht den Aufbau eines gefahrlosen Testumfeldes und verbessert den Testprozess der vorgeschlagenen Applikation allgemein.

Eine Verbesserung der Fähigkeiten des Fahrzeuges ist an mehreren Stellen möglich. Zum einen lassen sich Sensoren nachrüsten, um die Umfelderkennung zu verbessern. Eine Anwendung von Radar- oder Lidarsensoren ermöglicht einen größeren Erfassungsradius und erlaubt somit höhere Geschwindigkeitsbereiche. Weiterhin kann eine genaue Karte des Umfeldes für eine Trajektorieplanung erstellt werden. Diese Planung setzt ein Modell des Fahrzeuges voraus. Ein Aufbau eines solchen Modells ist zum Beispiel in der Softwareumgebung „Matlab/Simulink“ möglich. Auch „ROS“ stellt für Simulationszwecke eine Plattform zur Verfügung. Mithilfe von „Gazebo“ lassen sich dreidimensionale Fahrzeug- und Umgebungsmodelle aufbauen und alle „ROS“-Signale simulieren. Dies erlaubt neben der virtuellen Simulation von zukünftigen Funktionen, auch eine exakte Abstimmung von Reglerparametern. Eine zusätzliche Chance in der Kartenerstellung der Umgebung, liegt im Setzen eines definierten Zieles. Es ist möglich ein Ziel zu setzen und dieses durch die Streckenplanung im freien Raum ohne Kollisionen zu erreichen.

Die Komplexität des vorgestellten Parcous lässt sich leicht erweitern. So ist es vorstellbar die Objekterkennung durch weitere Fähigkeiten zu ergänzen. Nach dem Vorbild der vorgestellten Stoppschilderkennung lassen sich beliebige Straßenschilder hinzufügen. Dazu muss lediglich ein weiterer Classifier angelernt oder heruntergeladen werden. Eine Einbindung in die Struktur kann nach Vorbild der Stoppschilderkennung erfolgen. Auch weitere Objekte wie Ampeln oder Zebrastreifen sind vorstellbar. Somit ließe sich eine komplexe Straßenszene darstellen und durch den Demonstrator bewältigen.

Auch die Abbildung weiterer Funktionen ist mit der vorhandenen Hardware darstellbar. Eine automatisierte Parkfunktion bedarf lediglich dem Hinzufügen weiterer Ultraschallsensoren für den Heckbereich. Durch die Geschwindigkeitsinformation des Hallsensors lässt sich der zurückgelegte Weg berechnen. Wird diese Weginformation an die Entfernungsmessung des seitlichen Ultraschallsensors gekoppelt, können freie Parklücken erkannt werden. Ein Ablaufplan der Bewegung kann in einer ersten Version mit statischen Lenkwinkeln arbeiten und das Fahrzeug so in die Parklücke steuern. Dazu ist es weiterhin notwendig die Plattform durch einen rückwärtig ausgerichteten Ultraschallsensor zu erweitern, um gezielt Kollisionen im Heckbereich zu vermeiden. Theoretisch kann die Anzahl an Sensoren beliebig erweitert werden. Die Systemseite begrenzt, aufgrund der eingeschränkten Rechenleistung des Arduino und der eingeschränkten Datenbandbreite der USB-Verbindung, die Anzahl auf 50 Sensoren (durch die Anzahl der maximal möglichen Publisher im Arduino-Skript). Weiterhin ist die real mögliche Anzahl durch die Menge an Stecklätzen der Arduino-Datenpins begrenzt und müsste im Zweifelsfall durch weitere Arduinos ergänzt werden.

Eine Verbesserung der Spurerkennung und der Querregelung ist durch die Veränderung mehrerer Parameter möglich. Der Wechsel der Kamera hin zu einem Weitwinkelobjektiv würde den Erkennungsbereich insbesondere in Kurven erweitern und somit eine bessere Spurerkennung erlauben. Dies führt zu einem besseren Kurvenverhalten. Der Aufbau eines Fahrzeugmodells mit der Erfassung sämtlicher Parameter, wie Abmessungen, Geschwindigkeit und Kurvenverhalten, ließe eine genauere Querreglung nach Vorbild aktueller Serienansätze zu. Dazu bedarf es der Kenntnis des aktuellen Gierwinkels. Dieser kann aus den Daten der IMU ausgelesen, muss jedoch für eine Verwendung gefiltert und bearbeitet werden.

Literaturverzeichnis

[1] Statistisches Bundesamt, „destatis.de," 27 02 2018. [Online]. Available: https://www.destatis.de/DE/PresseService/Presse/Pressemitteilungen/2018/02/PD18_063_46241.html. [Zugriff am 28 02 2018].

[2] World Health Organization, „Global Status Report On Road Safety 2015," WHO Library Cataloguing-in-Publication Data, Genua, Schweiz, 2015.

[3] Statistisches Bundesamt, „Unfallentwicklung auf Deutschen Strassen," Statistisches Bundeamt, Wiesbaden, Berlin, 2016.

[4] Volvo Cars, „group.volvo.com," [Online]. Available: https://group.volvocars.com/company/vision. [Zugriff am 15 01 2018].

[5] EVOMOTIV GmbH, „evomotiv.de," juh, 2017. [Online]. Available: http://www.evomotiv.de/unser-unternehmen. [Zugriff am 20 01 2018].

[6] M. Maurer, J. C. Gerdes, B. Lenz und H. Winner, Autonomes Fahren, Berlin Heidelberg: Springer-Verlag, 2015.

[7] H. Menge, Langenscheidt Taschenwörterbuch Latein, Berlin und München: Langenscheidt KG, 2006.

[8] SAE International, „sae.org," [Online]. Available: https://www.sae.org/about/. [Zugriff am 11. Februar 2018].

[9] Bundesanstalt für Straßenwesen, „Rechtsfolgen zunehmender Fahrzeugautomatisierung," Wirtschaftsverlag NW, Bergisch Gladbach, 2012.

[10] SAE International, „SAE J3016 - Summary of SAE International's Levels of Driving Automation for On-Road Vehicles," 2014.

[11] J. Pander, „Spiegel Online," 16. Mai 2013. [Online]. Available: http://www.spiegel.de/auto/aktuell/die-neue-s-klasse-was-mercedes-audi-und-bmw-voraus-hat-a-900233.html. [Zugriff am 16. Februar 2018].

[12] Audi AG, „Presse Informationen - Der neue Audi A8," Audi MediaCenter, Ingolstadt, 2017.

[13] M. Mann, Benutzerorientierte Entwicklung und fahrergerechte Auslegung eines Querführungsassistenten, Göttingen: Cuvillier, 2008.

[14] Robert Bosch GmbH, Kraftfahrtechnisches Taschenbuch - 28. Auflage, Wiesbaden: Springer Vieweg, 2014.

[15] S. Pischinger und U. Seifert, Handbuch Kraftfahrzeugtechnik - 8. Auflage, Wiesbaden: Springer Vieweg, 2016.

[16] H. Winner, F. Lotz, S. Hakuli und C. Singer, Handbuch Fahrerassistenzsysteme 3. Auflage, Wiesbaden: Springer Vieweg, 2015.

[17] T. Grünweg, „Spiegel Online," 27. September 2017. [Online]. Available: http://www.spiegel.de/auto/aktuell/audi-a8-audi-ist-beim-autonomen-fahren-ein-level-weiter-a-1169062.html. [Zugriff am 17. Februar 2018].

[18] Verband der Automobilindustrie e.V., „Automatisierung - Von Fahrerassistenzsystemen zum automatisierten Fahren," VDA, Berlin, 2015.

[19] M. Friedl, A. Hupka und G. Tanzmeister, „Vollautomatisiertes Valet Parking: Funktions- und Planungsarchitektur," BMW Group, München, 2014.

[20] Daimler AG, „daimler.com," 2018. [Online]. Available: https://www.daimler.com/innovation/case/autonomous/fahrerlos-geparkt.html. [Zugriff am 17. Februar 2018].

[21] Waymo, „waymo.com," [Online]. Available: https://waymo.com/journey/. [Zugriff am 17. Februar 2018].

[22] Bundesregierung, *Entwurf eines Gesetzes zur Änderung des Straßenverkehrsgesetzes,* Berlin, 2017.

[23] S. Rademacher, *DVR-report,* Nr. Nr. 3/2016, p. 23, 2016.

[24] DVR, „dvr.de," 2. März 2017. [Online]. Available: https://www.dvr.de/presse/informationen/immer-mehr-fahrzeuge-mit-fahrerassistenzsystemen-erhaeltlich---auch-klein--und-kompaktwagen_id-4697.html. [Zugriff am 19. Februar 2018].

[25] Robert Bosch GmbH, „Fahrerassistenzsysteme – Wie viel Unterstützung wünschen Deutsche Autofahrer," Bosch Chassis Control Systems, Heilbronn, 2012.

[26] J.D. Power, „jdpower.com," 25. August 2015. [Online]. Available: http://www.jdpower.com/press-releases/2015-driver-interactive-vehicle-experience-drive-report. [Zugriff am 19. Februar 2018].

[27] S. Arndt, Evaluierung zur Akzeptanz von Fahrerassistenzsystemen, Wiesbaden: Springer Fachmedien, 2011.

[28] Detecon Consulting, „Autonomes Fahren: Wenn das Lenkrad zur Sonderausstattung wird," Detecon International GmbH, Köln, 2016.

[29] TÜV Rheinland, „Autonomes Fahren: Die Mehrheit vertraut dem Autopiloten," Köln, 2017.

[30] Goodyear Inc., „my-goodyear.de," [Online]. Available: https://www.my-goodyear.de/tgm/. [Zugriff am 22. Februar 2018].

[31] Duden - Die deutsche Rechtschreibung 24. Auflage, Mannheim: Dudenverlag, 2006.

[32] H. Willmann, G. Türck und H. Messinger, Langenscheidt - Taschenwörterbuch Englisch, Berlin und München: Langenscheidt KG, 2002.

[33] E. Bolte, Elektrische Maschinen, Berlin und Heidelberg: Springer Verlag, 2012.

[34] Carson Modelsport, X10E Onroad, Handbuch, Fürth.

[35] Carson Modelsport, *Dragster Sport RTR - Brushless Regler Handbuch,* Fürth, 2015.

[36] U. Häßler, F. Pfennig und D. Wüller, Digitale Fotografie, Berlin Heidelberg: Springer-Verlag, 1998.

[37] T. Führer, T. Heger und J. Heckel, „Stereovideokamera als Basis für Assistenzfunktionen," *ATZ,* pp. 22-27, 02 2014.

[38] J. Han, O. Heo, M. Park, S. Kee und M. Sunwoo, „Vehicle Distance Estimation Using a Mono-Camera For FCW/AEB Systems," *International Journal of Automotive Technology,* pp. 483-491, 2016.

[39] Z. Deng und X. Chen, „Detection of Road Obstacles Using 3D Lidar Data via Road Plane Fitting," in *Proceedings of the 2015 Chinese Intelligent Automation Conference*, Heidelberg Berlin, Springer-Verlag, 2015, pp. 441-449.

[40] G. Sorge, Faszination Ultraschall, Wiesbaden: Springer Vieweg, 2002.

[41] NVIDIA Corp., „nvidia.de," [Online]. Available: https://www.nvidia.de/self-driving-cars/. [Zugriff am 01. Februar 2018].

[42] NVIDIA Corp., „nvidia.de," [Online]. Available: http://www.nvidia.de/object/embedded-systems-dev-kits-modules-de.html. [Zugriff am 01. Februar 2018].

[43] Adafruit Industries, „adafruit.com," [Online]. Available: https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/overview. [Zugriff am 09. Februar 2018].

[44] P. Baumann, „6 Hallsensor," in *Ausgewählte Sensorschaltungen*, Wiesbaden, Springer Vieweg, 2017, pp. 133 - 147.

[45] P. Naur und B. Randell, „Software Engineering - Report on a conference sponsored by the NATO Science Commitee," Scientific Affairs Division NATO, Garmisch, 1969.

[46] Open Source Robotics Foundation, „ros.org," [Online]. Available: http://www.ros.org/testimonials/. [Zugriff am 08. Februar 2018].

[47] J. M. O'Kane, A Gentle Introduction to ROS, Columbia: CreateSpace Independent Publishing Platform, 2014.

[48] M. C. Ramon, Intel Galileo and Intel Galileo Gen 2, New York: Apress Open, 2014.

[49] G. Pahl und W. Beitz, Konstruktionslehre - Grundlagen erfolgreicher Produktentwicklung Methoden und Anwendung 5. Auflage, Berlin und Heidelberg: Springer Verlag, 2003.

[50] H.-U. Giersch, H. Harthus und V. Norbert, „Grundlagen der Leistungselektronik," in *Elektrische Maschinen*, Wiesbaden, Springer Fachmedien, 1998, pp. 51-65.

[51] KT-Electronic, „microcontroller.net," [Online]. Available: https://www.mikrocontroller.net/attachment/218122/HC-SR04_ultraschallmodul_beschreibung_3.pdf. [Zugriff am 10. Februar 2018].

[52] H. Kuhlmann, Strömungsmechanik, Hallbergmoos: Pearson, 2014.

[53] W. Yue, E. K. Teoh und S. Dinggang, „Lane detection and tracking using B-Snake," in *Image and Vision Computing 22*, Elsevier B.V., 2004, pp. 269-280.

[54] J. Canny, „A Computational Approach to Edge Detection," in *TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6*, IEEE, 1986, pp. 679-697.

[55] P. K. Kalra, „Canny Edge Detection," Department of Computer Science & Engineering, Indian Institute of Technology, Neu-Dehli, 2009.

[56] OpenCV, „opencv.org," [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html. [Zugriff am 24. Februar 2018].

[57] A. Herout, M. Dubska und J. Havel, Real-Time Detection of Lines and Grids, Brno: Springer, 2013.

[58] D. Yong, Z. Xu, Y. Zhang und K. Sun, „Fast lane detection based on bird's eye view and improved random sample consensus algorithm," Springer Science + Business Media, New York, 2016.

[59] M. Oliviera, V. Santos und A. D. Sappa, „Multimodal inverse perspective mapping," in *Information Fusion*, Aveiro, Portugal, Elsevier, 2014, pp. 108-121.

[60] OpenCV, „opencv.org," [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html. [Zugriff am 02. März 2018].

[61] OpenCV, „opencv.org," [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html. [Zugriff am 02. März 2018].

[62] P. Sharma, M. K. Gupta, A. K. Mondal und V. Kaundal, „HAAR like Feature-Based Car Key Detection using Cascade Classifier," in *Proceeding of International Conference in Intelligent Communication, Control and Devices*, Singapore, Springer Science+Business Media, 2016, pp. 689-694.

[63] M. Gaynor, „github.com," [Online]. Available: https://github.com/markgaynor/stopsigns. [Zugriff am 15. Dezember 2017].

[64] M. Mitschke und H. Wallentowitz, Dynamik der Kraftfahrzeuge 5. Auflage, Braunschweig: Springer Fachmedien, 2014.

[65] H. P. Geering, Regelungstechnik, Zürich: Springer-Verlag, 1996.

Anhang

Anhang 1

```
#include <Servo.h>
#include <ros.h>
#include <std_msgs/UInt16.h>
#include <autonomous/carcontrol.h>
#include <sensor_msgs/Range.h>

Servo esc, servo;

int triggerf1 = 10;
int echof1 = 9;
int triggers1 = 8;
int echos1 = 7;
float dauer = 0;
float entfernung = 0;
int notaus = 13;



ros::NodeHandle nh;

void control_c( const autonomous::carcontrol& cmd_msg){
  if (cmd_msg.servo > 0)
    {
    servo.write(cmd_msg.servo);
    if (cmd_msg.esc == 1500)
      {
      esc.write(1400);
      delayMicroseconds(3000);
      esc.write(1500);
      }
    else
      {
      //drive();
      esc.writeMicroseconds(1535);

      }
    }
}

sensor_msgs::Range range_msg;
ros::Subscriber<autonomous::carcontrol> sub("/car_control", control_c);
ros::Publisher pub_range1( "/ultrasoundf1", &range_msg);
ros::Publisher pub_range2( "/ultrasounds1", &range_msg);

void setup()
{
  nh.initNode();
  nh.advertise(pub_range1);
  nh.advertise(pub_range2);
  nh.subscribe(sub);

  servo.attach(11);
  esc.attach(12);

  range_msg.radiation_type = 0;
```

```
  range_msg.field_of_view = 0.15;
  range_msg.min_range = 0.0;
  range_msg.max_range = 10000.0;
}

void loop()
{
  range_msg.range = getRange_ultra(triggerf1,echof1);
  range_msg.header.stamp = nh.now();
  pub_range1.publish(&range_msg);

  nh.spinOnce();
  delay(10);
}

float getRange_ultra(int trigger, int echo) {

  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);

  digitalWrite(trigger, LOW);
  delay(5);
  digitalWrite(trigger, HIGH);
  delay(10);
  digitalWrite(trigger, LOW);

  dauer = pulseIn(echo, HIGH);
  entfernung = (dauer / 2) * (0.0343509);

  return entfernung;
}

void drive() {
  esc.writeMicroseconds(1540);
  delay(100);
  esc.writeMicroseconds(1500);
  delay(50);
}

void stope() {
  esc.write(1400);
  delayMicroseconds(3000);
  esc.write(1500);
}
```

Anhang 2

```
<launch>

      <include
            file="$(find autonomous)/launch/camera.launch"
      />


      <node
            pkg="autonomous"
            name="image_preprocessinggray_node"
            type="image_preprocessinggray_node.py"
```

```
        respawn="true"
    />


    <node
        pkg="autonomous"
        name="image_lanedetection_node"
        type="image_lanedetection_node.py"
        respawn="true"
    />


    <node
        pkg="autonomous"
        name="image_stopsign_node"
        type="image_stopsign_node.py"
        respawn="true"
    />


    <node
        pkg="autonomous"
        name="esc_control_node"
        type="esc_control_node.py"
        respawn="true"
    />


    <node
        pkg="autonomous"
        name="car_control_node"
        type="car_control_node.py"
        respawn="true"
    />




    <node
        pkg="rosserial_python"
        name="serial_node"
        type="serial_node.py"
        args="/dev/ttyACM0"
        respawn="true"
    />


    <node name="steering_pid" pkg="pid" type="controller" >
        <param name="Kp" value="3.0" />
        <param name="Ki" value="1.0" />
        <param name="Kd" value="0.0" />
        <param name="upper_limit" value="20" />
        <param name="lower_limit" value="-20" />
        <param name="topic_from_controller" value="control_steer-
ing" />
        <param name="topic_from_plant" value="abw_steering" />
        <param name="setpoint_topic" value="set_steering" />
```

```
            <param name="max_loop_frequency" value="25.0" />
            <param name="windup_limit" value="10" />
        </node>

</launch>
```

Anhang 3

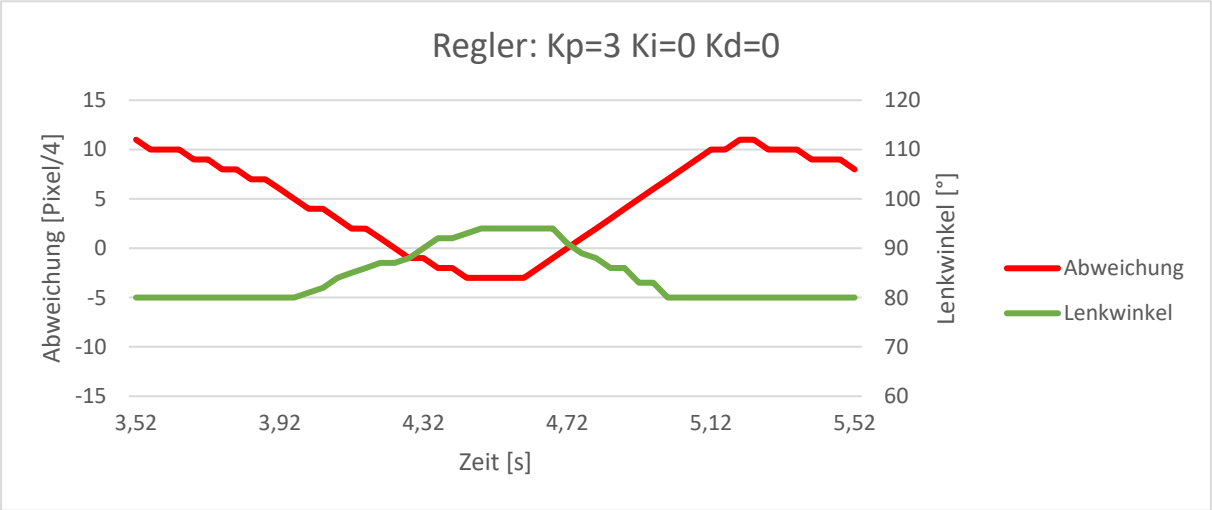


Abb. 1: Reglerverhalten KP=3, KI=0, KD=0

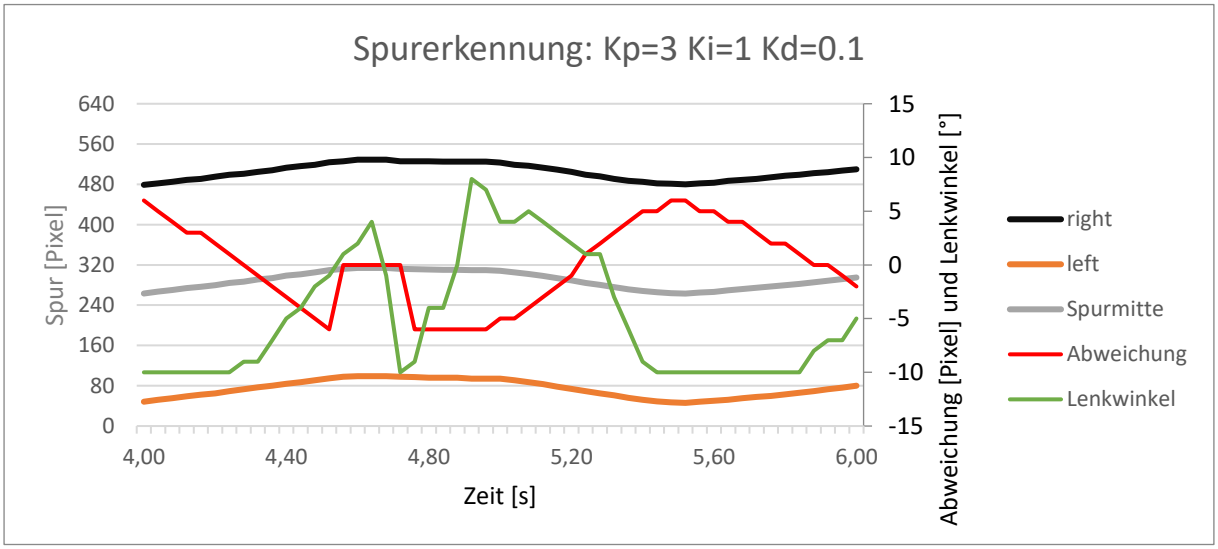


Abb. 2: Reglerverhalten in Spur KP=3, KI=1, KD=0.1