R O B O T I K P R A K T I K U M

Wintersemester 2017/ 2018

Dokumentation - Gruppe 1

tinf101030 - Julian Finck

tinf101364 - René P. Keller

inf101301 - Lea Morschel

tinf101313 - Adrian Sorge

tinf101087 - Julian Weihe

Inhaltsverzeichnis

1 Einführung

Das Robotik-Praktikum, dessen Ergebnis diese Dokumentation festhält, findet parallel und in Kombination mit der Bachelor-Vorlesung 'Einführung in die Robotik' statt, welche von Herrn Prof. Dr. Ulrich Hoffmann im Wintersemester 2017 gehalten wird.

Ziel ist die Vertiefung und Anwendung der in der Vorlesung vermittelten Inhalte, wobei der diessemestrige Schwerpunkt auf autonomen Robotern liegt. Im Praktikum, unter der Leitung von Hermann Höhne, wird mit einem AADC[1] 2017 Audi-Modell im Maßstab 1:8 gearbeitet. Es soll zu Beginn eine nicht näher eingeschränkte Zielsetzung ausgearbeitet werden, welche auf der grundlegenden Anforderung der kollisionsfreien Fahrt aufbaut.

1.1 Zielsetzung

Im Kontext der gegebenen Grundanforderung der Kollisionsvermeidung werden eigene Ziele definiert. Das Auto soll später in der Lage sein, unter verschiedenen Lichtverhältnissen, einer zentralen blauen Fahrbahnmarkierung zu folgen. Bei Näherkommen eines von Ultraschallsensoren detektierbaren Hindernisses soll das Modell langsamer werden und vor ihm zum Stehen kommen sowie die Fahrt erst wieder aufnehmen, wenn die Sensoren signalisieren, dass die Fahrbahn wieder frei ist.

Die zu Beginn des Praktikums ausgearbeitete Roadmap findet sich in Abbildung 1.

[1] Audi Autonomous Driving Cup: https://www.audi-autonomous-driving-cup.com/wettbewerb/ueberblick/

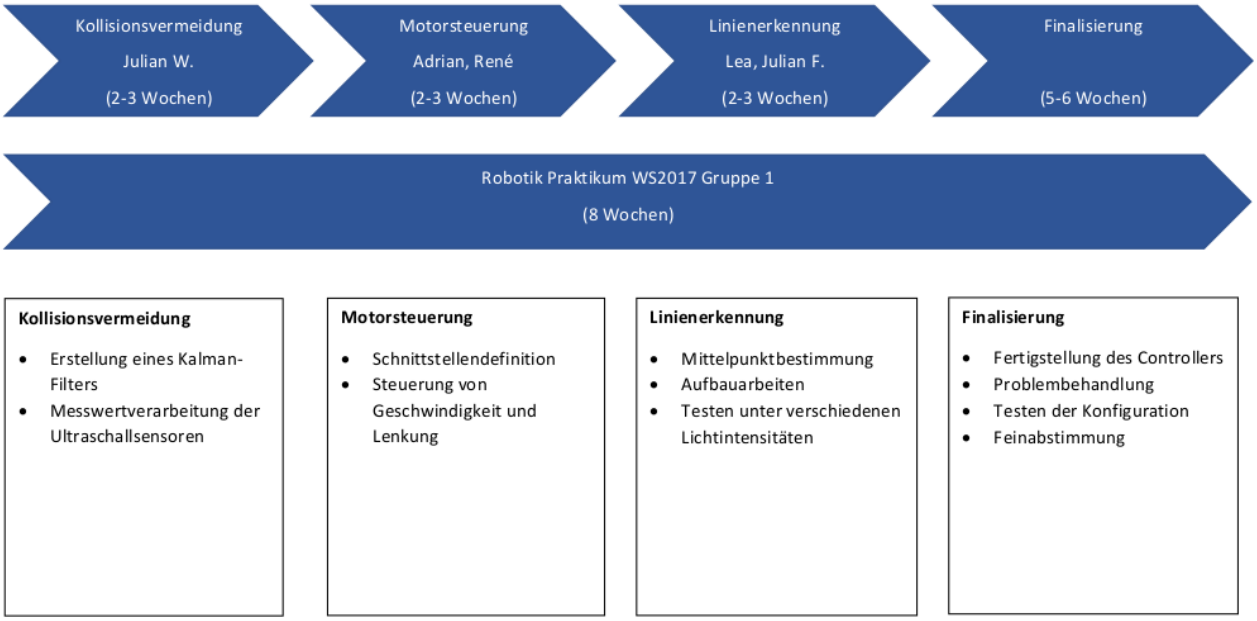


Abbildung 1: Zu Projektbeginn erstellte Roadmap

Die Umsetzung des Projektes wird zu Anfang in vier Bereiche unterteilt:

1. Linienerkennung und -verwertung
    a) Mittelpunktbestimmung
    b) Testen unter verschiedenen Lichtintensitäten
2. Abstandserkennung und Kollisionsvermeidung
    a) Messwertverarbeitung der Ultraschallsensoren
    b) Messwertglättung durch Filterung (Kalman)
3. Motorsteuerung
    a) Schnittstellendefinition zur Ansteuerung
    b) Steuerung von Geschwindigkeit und Lenkung
4. Zusammenbringung/ Fusionsschnittstelle
    a) Definition und Ausarbeitung des Controllers (Verhaltens-Logik)
    b) Problembehandlung, Feinabstimmung, Testung

2 Grundlagen

Im Folgenden werden die Grundlagen, bestehend aus Aufbau des genutzten Modellautos, der darauf laufenden Software sowie das genutzte Framework erläutert, welche für das Verständnis des Projektes essenziell sind.

2.1 Hardwareplattform

Das Audimodell bietet verschiedene Hardwarekomponenten, um das autonome Fahren zu verwirklichen. Es ist in Abbildung 2 zu sehen.

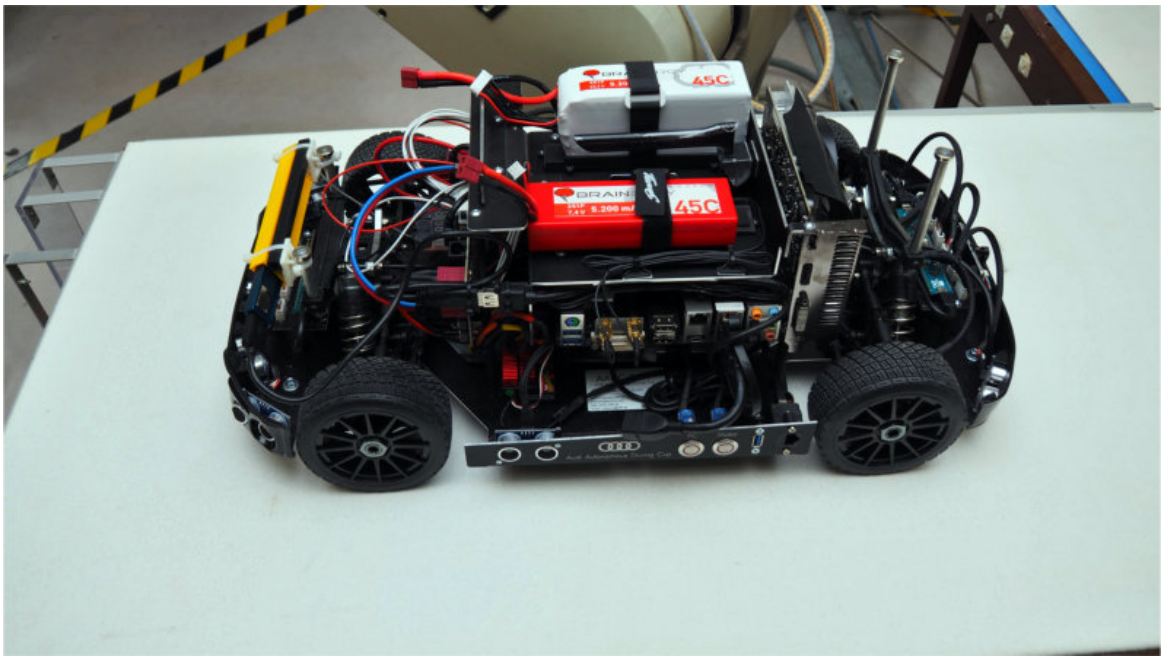


Abbildung 2: Seitliche Ansicht des im Projekt verwendeten Audi-Modells

2.1.1 Grundkomponenten

Die Stromversorgung der Steuerung und der Motoren während der Fahrt wird jeweils von einem separaten Akku gewährleistet.

2.1.2 Rechner

Herzstück des Audi ist das *GIGABYTE GA-Z170N-WIFI miniITX* Mainboard mit einem *Intel Core i3-6100T* und 8 GB DDR4 RAM. Unterstützt wird es von einer *NVIDIA GeForce GTX*

1050Ti.

Das Betriebssystem wird von einem USB-Stick gebootet, um das Auto mit mehreren Gruppen nutzen zu können.

2.1.3 Aktoren

Angetrieben wird das Modell von einem *Hacker SKALAR 10 21.5 Brushless Motor 1/10*. Die Lenkung funktioniert über einen *Absima 'ACS1615SG' Combat Series* Servo.

2.1.4 Sensoren

Eine visuelle Erfassung der Umwelt erfolgt durch eine frontal angebrachte Tiefen-Kamera und eine rückwärtige RGB-Kamera. Zur Abstandsdetektion sind 10, in verschiedene Richtungen schauende Ultraschallsensoren angebracht (siehe Abbildung 3). Beschleunigung, Ausrichtung und Lage im Raum werden vom einem Motion-Tracking Sensor übernommen. Aufschluss über die Radstellung wird mittels Drehgeber gegeben. Das Auslesen der Sensordaten erfolgt über Arduinos, welche per USB Protokoll die Daten zum Mainboard senden.

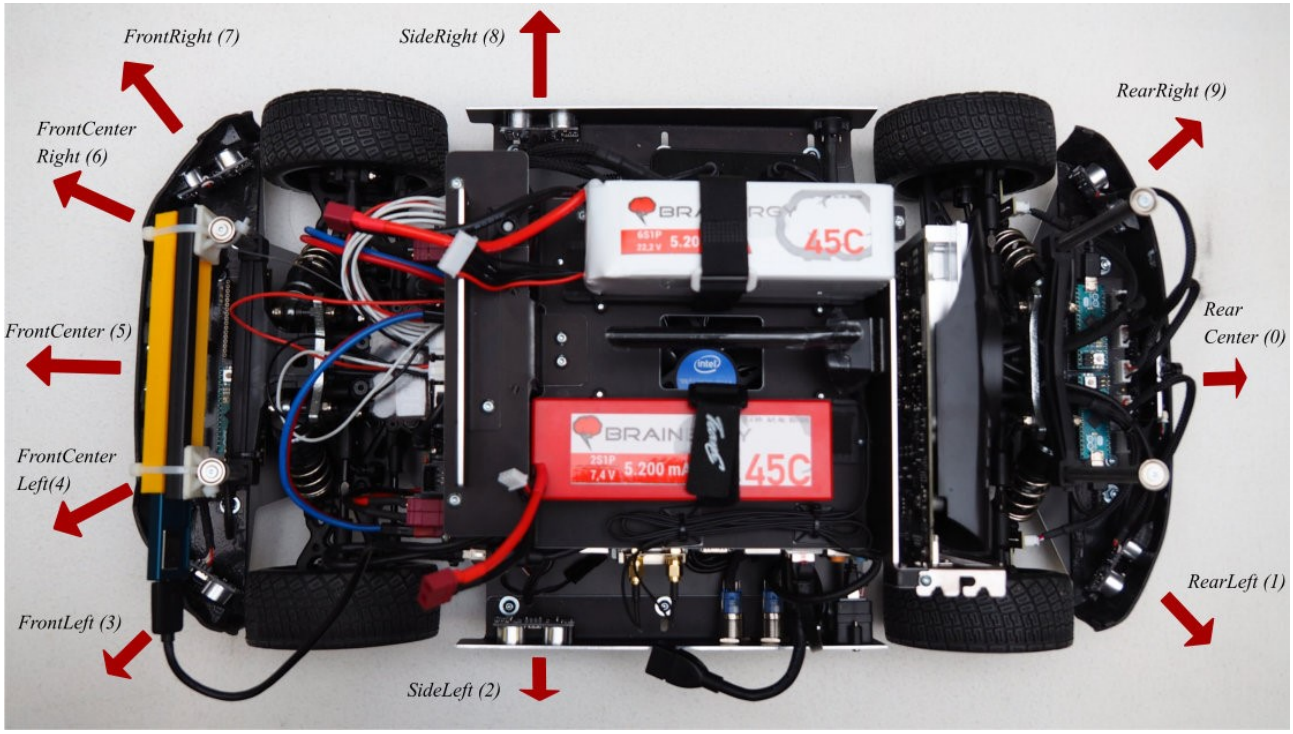


Abbildung 3: Platzierung der Ultraschallsensoren am Audi-Modell

2.2 Softwareplattform

Das Modellauto verwendet die Ubuntu Version 16.04 und zur Bildschirmübertragung ist der x11vcn Serverdienst eingerichtet. Die Software EB Assist ADTF 2.14 von Elektrobit ist installiert, um die damit bereitgestellte Programmoberfläche und das ADTF-Framework[2] zu verwenden.

2.2.1 ADTF-Framework: Grundlegende Strukturen

Das Framework stellt Funktionen zur Kommunikation mit den im Abschnitt 'Hardwareplattform' erläuterten Sensoren und Kameras bereit. Außerdem ermöglicht es eine Programmaufteilung in Funktionsblöcke, *Filter* genannt, welche parallel auf mehreren Prozessorkernen ausgeführt werden können. Die Kommunikation zwischen diesen Blöcken wird über Nachrichtenqueues und Priorisierung vom Framework sichergestellt. Die Ausführung der Funktionsblöcke wird einstellbar, periodisch und eventgesteuert ausgelöst und beginnt meist mit dem Anstoß zum Auslesen der Sensoren, welche dann das Event über die Queues an andere Funktionsblöcke weitergeben.

Von der Benutzeroberfläche aus lassen sich die Funktionsblöcke grafisch miteinander verbinden, welches die Kommunikation zwischen den Blöcken repräsentiert, und die Ausführung der Funktionen starten. Des Weiteren gewährt die Oberfläche Zugriff auf fertige Funktionsblöcke, etwa zum Anzeigen und Abspeichern von Kamerabildern und Sensordaten oder zum Abspielen selbiger. Durch Referenzen aus dem Programm heraus können über die Benutzeroberfläche bestimmte Parameter zur Initialisierung und zur Laufzeit gesetzt werden.

[2]Automotive Data and Time-Triggered Framework: https://www.elektrobit.com/products/eb-assist/adtf/

3 Implementierung

3.1 Umgebung

Die Zielsetzung sieht vor, das Auto eine blaue Linie verfolgen zu lassen. Zu Testzwecken wird daher auf dem hell grauen Boden eine große, ovalförmige Linienstruktur mit mattblauem Klebeband befestigt, deren Maße etwa 1,5 m x 3 m beträgt. Zur Vermeidung von störender Spiegelung wird zudem das Fenster zugestellt, um die Sonneneinstrahlung zu minimieren.

3.2 Projektbestandteile

Das Projekt wird zu Beginn in drei verschiedene Aufgabenbereiche aufgeteilt, die separat bearbeitet werden (siehe Unterabschnitt 1.1). Diese werden im Folgenden näher beschrieben.

1. Motorsteuerung
2. Abstandserkennung und -verwertung
3. Linienerkennung und -verfolgung

3.2.1 Motoransteuerung

Die zentrale Aufgabe der Motoransteuerung ist die Aufbereitung und Weitersendung der Geschwindigkeit und des Lenkwinkels an die Aktoren sowie die Initialisierung des Speed Controllers.

Da die interne Kommunikation der Motordaten auf Werte zwischen -100 bis 100 normiert sind, ist eine Umrechnung dieser in reale Steuerdaten notwendig. Hierbei lässt sich auch die Maximalgeschwindigkeit festlegen bzw. drosseln. Bedingt durch Reaktionszeiten der Sensoren hat sich eine Drosselung auf 12 Prozent der tatsächlichen Maximalgeschwindigkeit bewährt.

Um zeitkritisch auf ein von den Ultraschallsensoren detektiertes Hindernis mit einem Nothalt reagieren zu können, empfängt die Motorensteuerung Flags direkt von der Ultraschallauswertung und umgeht damit weitere Programmlogik.

3.2.2 Abstandserkennung und -verwertung

Die Abstandserskennung ist eine der grundlegenden Anforderungen an einen autonomen Roboter zur Ermöglichung von gefahrloser Navigation. In diesem Projekt wird sie auf Basis von Messwerten der zehn im Auto verbauten Ultraschallsensoren (siehe Abbildung 3) implementiert.

Die eingehenden, mediangefilterten Abstandswerte liegen im Bereich von 0 bis 400 cm. Diese werden unterschiedlich gewichtet, dann der geringste Wert in eine Geschwindigkeit umgerechnet. Die Übersetzung eines Abstands- in einen Geschwindigkeitswert wird anhand einer linearen Funktion realisiert, die einen minimalen Input-Grenzwert zum Anstoßen der Bewegung und gleichsam einen maximalen Input-Wert beinhaltet, ab dem die volle Geschwindigkeit zurück gegeben wird (siehe Abbildung 4). Dies dient dazu, einen gewissen Mindestabstand beim Fahren nicht zu unterschreiten, sowie zum Ignorieren von Hindernissen, die weiter als ein gewisser Maximalwert von den Sensoren entfernt aufgefasst werden. Ersterer wird durch Testergebnisse statisch auf 20 cm gesetzt, letzterer auf 100 cm. Im Abstands-Wertebereich von 20 cm bis 100 cm gilt die lineare Funktion $y = x$.

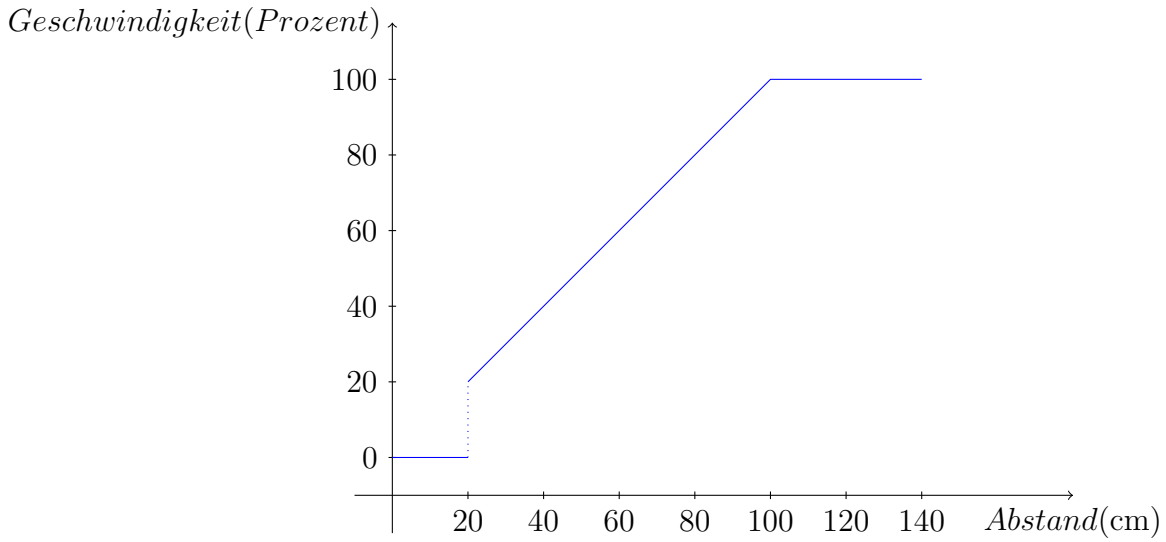


Abbildung 4: Übersetzung von Abstandswert in prozentuale Geschwindigkeit

Die Geschwindigkeit wird mithilfe des kleinsten von allen Ultraschallsensoren gemessenen Abstandswertes berechnet. Da den zur Seite oder gar nach hinten gerichteten Sensoren bei geradliniger Fahrt nicht die selbe Priorität beigemessen werden sollte, wie den nach vorne ausgerichteten, wo der Abstand eher kollisionsrelevant ist, wird zunächst jeder Wert nach Relevanz

gewichtet und entsprechend zu einem neuen Wert verrechnet. Mit jedem Sensor wird ein Winkel assoziiert, welcher auf den Bereich zwischen -1 und 1 abgebildet wird, wobei 0 die frontale Ausrichtung beschreibt. Der Bereich von 0 bis -1 beschreibt im Modell den gegen den Uhrzeigersinn gerichteten Halbkreis bis zum nach hinten gerichteten Sensor (Abbildung 3; der Reihe nach die Sensoren (5), (4), (3), (2), (1), (0)) und äquivalent der von 0 bis 1 gehende Bereich den Halbkreis im Uhrzeigersinn zum nach hinten gerichteten Sensor (Abbildung 3; der Reihe nach die Sensoren (5), (6), (7), (8), (9), (0)). Alle rückgerichteten Sensoren werden der Einfachheit halber auf 1 respektive -1 abgebildet, da sie bei der normalen Fahrt nicht weiter von Interesse sind. Die Zuordnung der Winkel ist veranschaulicht in Abbildung 5.

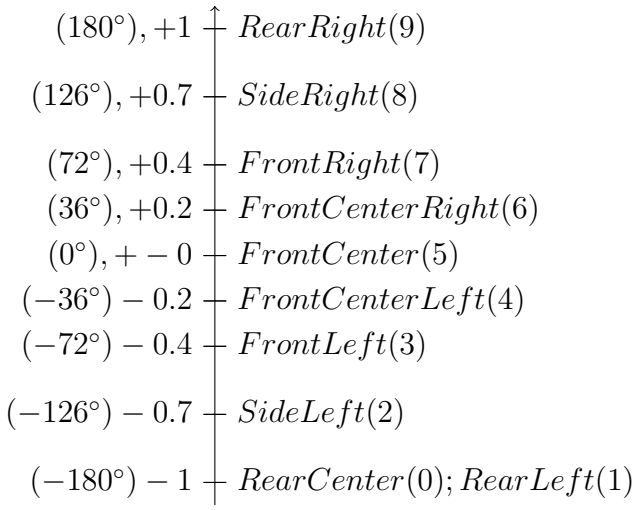


Abbildung 5: Ultraschallsensoren und ihr assoziierter Winkel, projiziert auf den Bereich zwischen -1 und 1

Basierend auf diesen Winkeln wird gemäß einer Gauss-Kurve der Gewichtungsfaktor für den jeweiligen Sensor bestimmt, mit dem dessen Messwert multipliziert wird (dargestellt in Abbildung 6). Dieser Faktor ist stets größer oder gleich 1, was bedeutet, dass ein gemessener Abstand nur zunehmen und damit an Relevanz verlieren kann, wenn es darum geht, im Anschluss den kleinsten zur Geschwindigkeits-Berechnung zu wählen.

Solche statisch festgelegten Gewichtungsfaktoren sind aber vor allem für ein Geradeaus-Fahren mit gewisser Geschwindigkeit optimiert. Das ist beispielsweise in Kurven problematisch, in welchen das Hauptaugenmerk nicht mehr geradeaus liegen sollte, da die Geschwindigkeit durch links oder rechts vorbeifahrend frontal erkannte Hindernisse unnötig gedrosselt werden würde. Daher wird eine dynamische Anpassung des relevanten Sensor-Blickkegels realisiert.

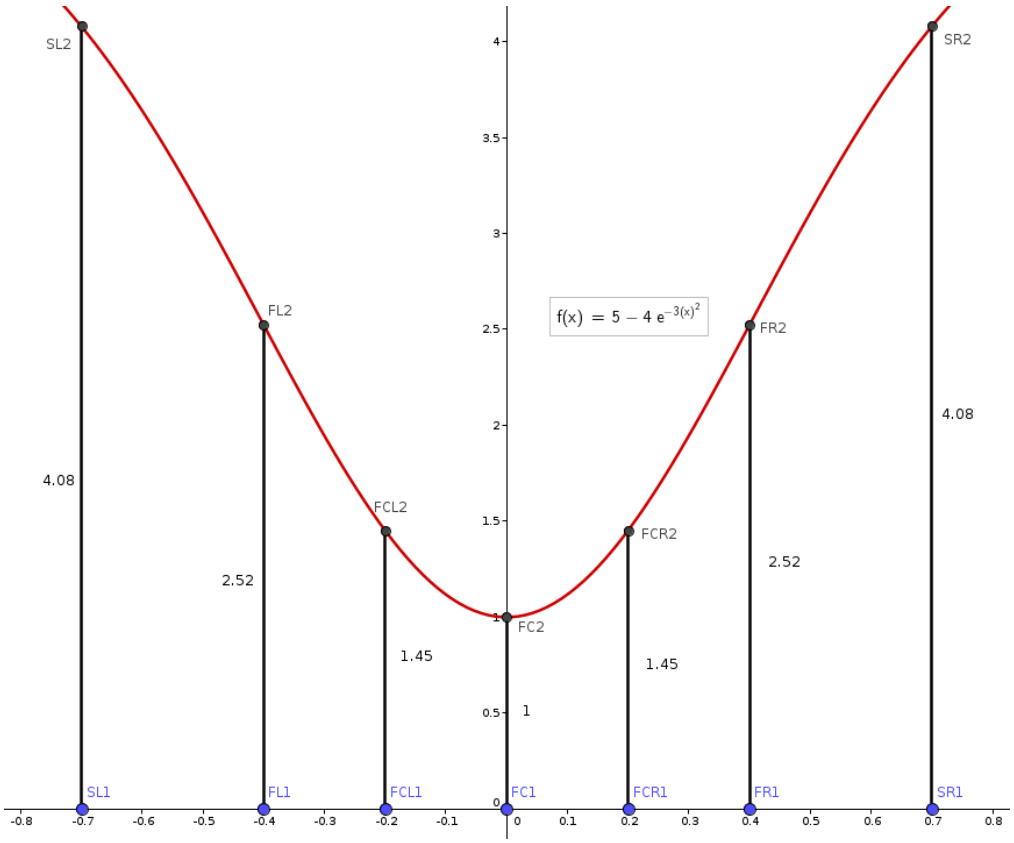


Abbildung 6: Gewichtung der Abstands-Messwerte beim Geradeaus-Fahren

Dazu wird gemäß des eingeschlagenen Lenkwinkels, der dazu auf den Bereich von -1 bis 1 normiert einfließt, die Gauss-Kurve auf der x-Achse verschoben, wie dargestellt in Abbildung 7.

Konkret sind in Abbildung 6 und 7 die fixen Ablesepunkte der einzelnen Sensoren auf der x-Achse abgebildet, welche an den Sensor-Winkeln in Blickrichtung des Autos angelehnt werden. Der entsprechende Punkt auf der Gauss-Kurve gibt den jeweiligen Gewichtungs-Faktor an, wobei ein größerer Wert eine größere Unempfindlichkeit bedeutet. Gemäß Lenkwinkel wird nun die Kurve auf der x-Achse verschoben, wodurch die Gewichtung sich auf die entsprechend seitlich liegenden Sensoren verschiebt.

Die Gauss-Kurve entspricht folgender Formel,

$$sensorWeight = 5 - 4 * \exp\Big(-3 * (sensorAngle - steeringAngle)^2\Big) \tag{1}$$

wobei *sensorAngle* der auf den Bereich zwischen -1 und 1 normierte Sensorwinkel ist, für den eine Gewichtung berechnet werden soll, und *steeringAngle* der eingeschlagene Lenkwinkel, ebenfalls im Bereich zwischen -1 und 1 liegend. Die Höhe der Kurve, bestimmt durch die

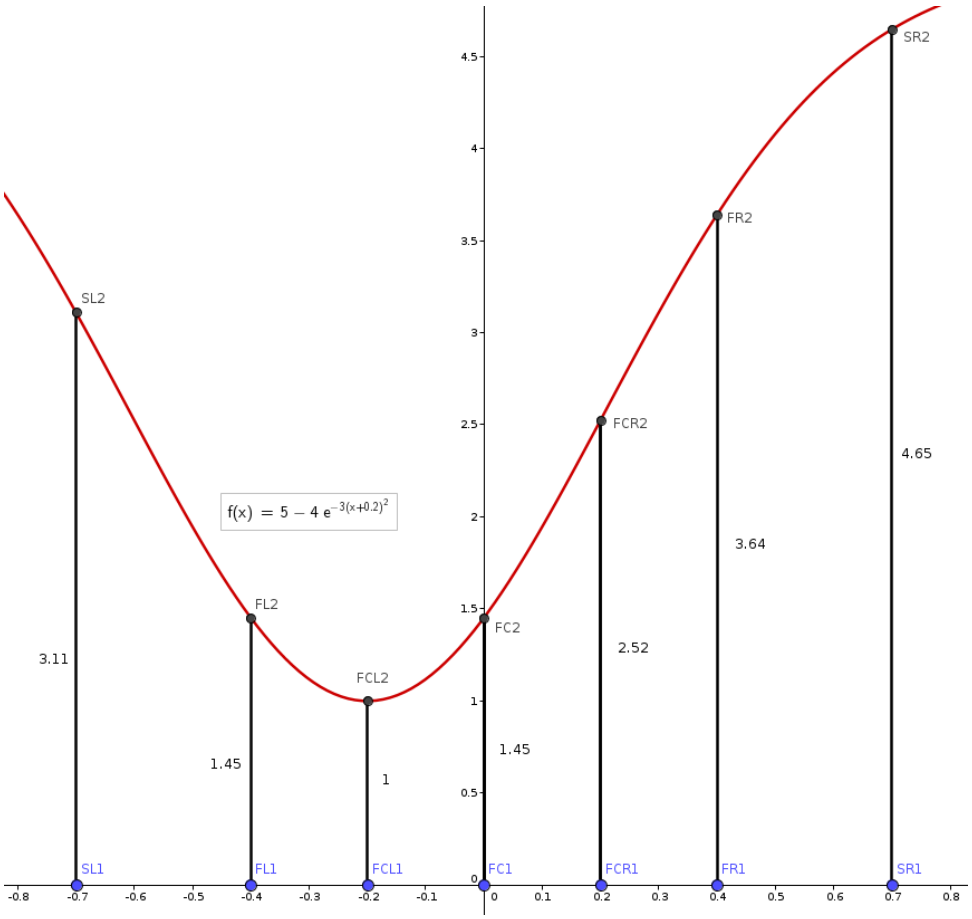


Abbildung 7: Gewichtung der Abstands-Messwerte bei Lenkung nach links (maximale Verschiebung)

addierte 5 sowie die multiplizierte -4, wurden durch Versuche als angemessen herausgestellt und festgelegt.

3.2.3 Linienerkennung und -verfolgung

Damit das Auto der blauen Fahrbahnlinie folgen kann, wird die Linie im Kamerabild detektiert. Anschließend wird die Position der Linie ausgewertet, um daraus Steuersignale für die Lenkung zu gewinnen. Diese Aufgaben werden von zwei Filtern gelöst:

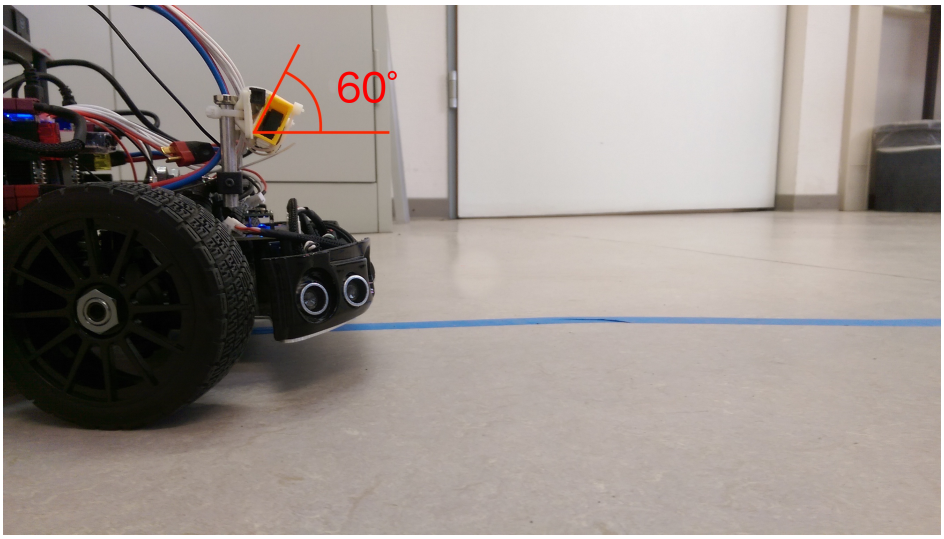


Abbildung 8: Winkel der Frontkamera am Modell

BlueImgFilter Dieser Filter markiert die Fahrbahnlinie und greift dafür auf die OpenCV Bibliothek zurück. Die Kamera ist in einem 60 Grad Winkel montiert (siehe Abbildung 8). Es ist davon auszugehen, dass die Fahrbahnlinie der einzig blaue Bildinhalt im relevanten Bildausschnitt ist. Zunächst wird das Eingangsbild mit OpenCV vom RGB- in den HSV-Farbraum transformiert, um die Festlegung des Farbbereiches für Blauwerte zu vereinfachen. Mittels der OpenCV-Funktion *inRange()* werden anschließend alle nicht blauen Bildinhalte heraus gefiltert und ein Binärbild zurückgegeben, welches alle blauen Flächen markiert. Die endgültigen Farbwertgrenzen sind das Ergebnis von folgenden Versuchen mit variierenden Lichtverhältnissen:

Versuch 1 - Szenario mit geringster Lichtintensität Lichtverhältnisse: Jalousie runter gelassen, Beleuchtung aus, Tageslicht von den Oberlichtern, bewölkt (mittags)

Versuch 2 - Szenario mit höchster Lichtintensität Lichtverhältnisse: Jalousie oben, alle Lichter an, keine Wolken (mittags)

Auswertung:

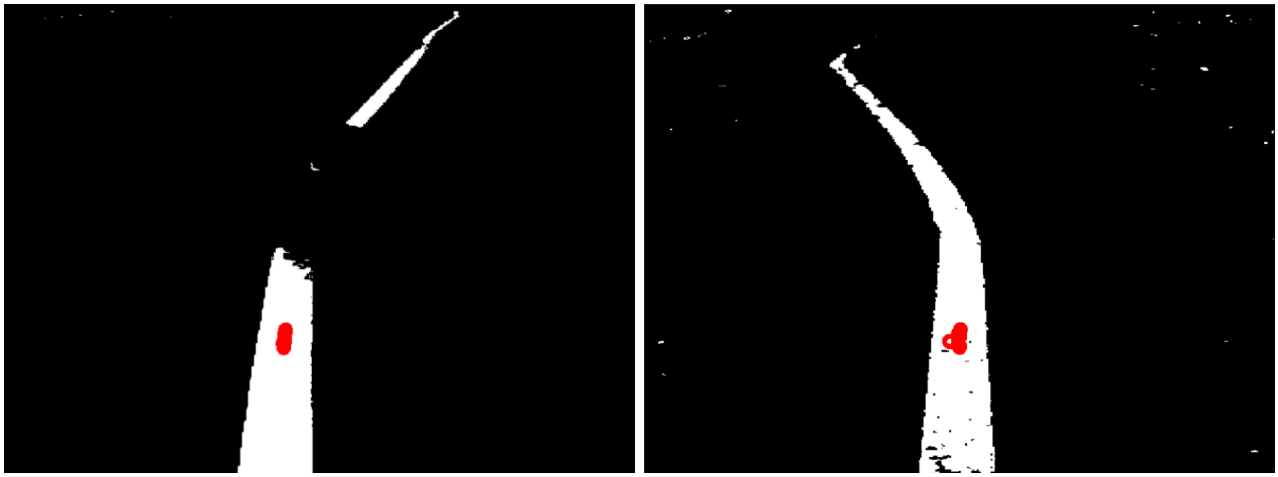
Als Ergebnis dieser Versuche sind die in Tabelle 1 folgenden Unter- bzw. Obergrenzen der Funktion *inRange()* eruiert worden, welche in beiden Extremfällen eine Positionsberechnung der Linie ermöglichen.

Tabelle 1: Ergebnisse der Tests bei verschiedener Lichtintensität

	Untergrenze	Obergrenze
Hue	95	130
Saturation	100	255
Value	70	255

Im dunklen Szenario beeinträchtigt der geringe Rauschabstand die Erkennung der Linie leicht, wie in Abbildung 9 (b) zu erkennen ist. Die Ergebnisse lassen dennoch eine ausreichend robuste Positionsberechnung zu.

Im hellen Szenario ist der Rauschabstand so groß, dass *inRange()* die Fahrbahnmarkierung nahezu optimal ausschneidet (Abbildung 11 (a)). Reflexionen der Sonne können zu Clipping führen und verhindern eine durchgehende Erkennung der Markierung (siehe Abbildung 9 (b)).



(a) Viel Licht und Reflektion (b) Wenig Licht

Abbildung 9: Gefilterte Linienerkennung unter extremen Lichtverhältnissen

Daher ist es wichtig, direkte Sonneneinstrahlung auf der Fahrbahn zu vermeiden.

OneLineDetect Dieser Filter erhält das Binärbild und ermittelt die Position der blauen Linie relativ zur Position des Autos. Dafür wird in einer Bildzeile kurz unterhalb der Bildmitte (kor-

reliert mit dem Bereich 20 cm vor dem Auto) nach der größten Anzahl weißer, zusammenhängender Pixel gesucht. Der Mittelpunkt wird berechnet und als Zentrum der dort verlaufenden Linie interpretiert. Der Abstand zwischen ermittelten Linienmittelpunkt und Bildmitte wird auf einen Wert zwischen -100 und 100 normiert und ausgegeben (siehe Abbildung 10).

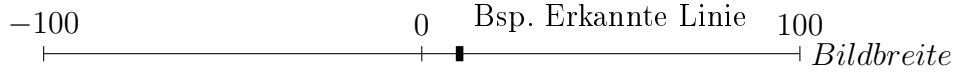


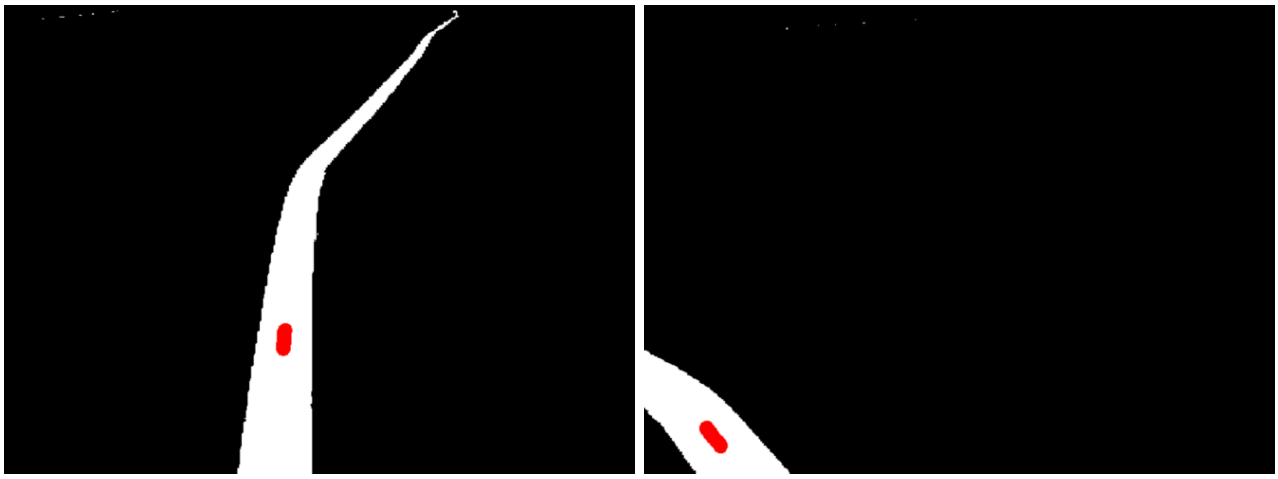
Abbildung 10: Beispiel eines errechneten Fahrstreifen-Mittelpunktes (bei 10) innerhalb einer Bildzeile, normiert auf den Bereich -100 bis 100

Um die Berechnung der Linienmitte robuster zu gestalten, was besonders bei verrauschten Kamerabildern in dunklen Lichtverhältnissen nützlich ist, wird die Linienmitte auch in benachbarten Bildzeilen berechnet. Auf die errechneten Linienmittelpunkte wird der Median-Filter angewendet. Außerdem gibt es eine Mindestanzahl an zusammenhängenden weißen Pixeln, die erreicht werden muss, um als Fahrbahnmarkierung zu gelten.

Wird beispielsweise in einer Kurve in dem Bereich unter der Bildmitte keine Linie detektiert, so wird in einem zweiten Suchbereich nahe des unteren Bildrandes gesucht (siehe Abbildung 11 (b)). Dieser Bereich korreliert mit dem Abstand von 5 cm vor dem Auto.

Wird auch hier keine Linie detektiert, wird der Wert -101 ausgegeben.

Auf den Abbildungen 9 und 11 sind in Rot jeweils diejenigen Punkte markiert, die zur Linienpositionierung erkannt wurden.



(a) Wenig gekrümmte Fahrstrecke

(b) Enge Kurve

Abbildung 11: Gefilterte Linienerkennung unter normalen Lichtverhältnissen - unterschiedliche Behandlung gemäß Fahrbahnkrümmung

3.3 Interprozesskommunikation

Da die Programmierung in verschiedenen Funktionsblöcken erfolgte, musste eine einheitliche Kommunikation sichergestellt werden. Hierzu wurde eine separate Funktionssammlung erstellt, die von allen Funktionsblöcken inkludiert wurde. Sie beinhaltet Definitionen über den Nachrichtentyp und die Zusammensetzung strukturierter Datentypen. Da der vom ADTF gestellte Ablauf zum Senden oder Empfangen eines bestimmten Nachrichtentyps zur Codeverdopplung in den Funktionsblöcken führen würde, wurden die Funktionen zusammengefasst und ausgelagert. Damit ließ sich auch einheitlich sicherstellen, dass Funktionsblöcke beim Empfangen die Nachricht erst Kopierten, bevor deren Inhalt verändert wurde.

3.4 Projektstruktur

Die drei zuvor beschriebenen Aufgabenbereiche werden über ADTF in Funktionsblöcke aufgeteilt.

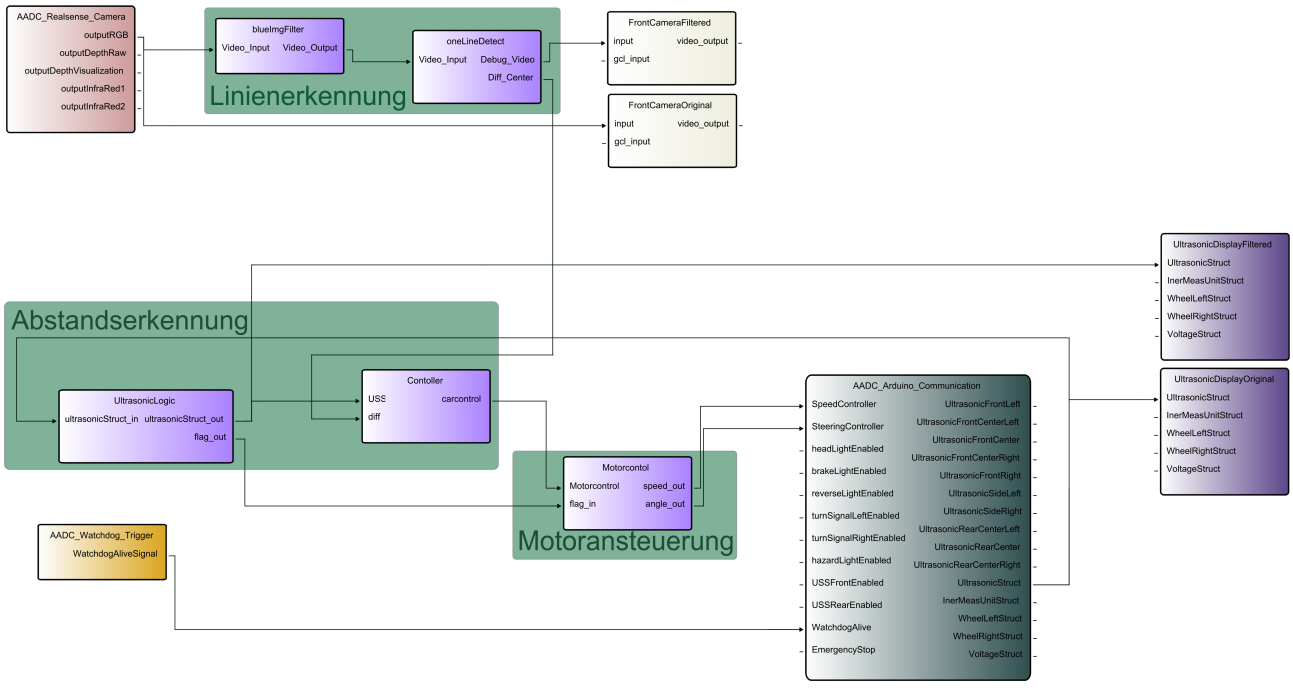


Abbildung 12: Funktionsblockplan der Test-Umgebung

3.5 Komplikationen und Fehler

Im Folgenden findet sich eine Auflistung von Fehlern und Komplikationen, die während der Entwicklung auftraten und zu Einschränkungen, der in der Roadmap gesetzten Ziele führen.

3.5.1 ADTF

Dokumentation Der Umgang mit dem ADTF gestaltet sich zu Anfang sperrig. Zu vielen vorgefertigten Filtern fehlt die Schnittstellenbeschreibung. Auch die im Projekt genutzte Kommunikation zwischen den Filtern ist nicht einheitlich und lässt sich daher nur mühsam verstehen. So ist die Kommunikation mit dem Arduino-Filter mittels einer Sequenzialisierung und der Übermittlung des Signals in Klarschrift eine gänzlich andere, als die Weitergabe von Zeigern auf Speicherbereiche, wie an anderer Stelle verwendet.

Nutzung in virtueller Maschine Die parallele Filterentwicklung setzt das verwenden von virtuellen Maschinen voraus, doch von ADTF bereitgestellte Funktionen zur grafischen Darstellung von Videosignalen funktionieren damit nicht. Hierfür ist ein weiterer im Nachhinein selbst implementierter Filter erforderlich.

3.5.2 Kalman-Filter

Zur Filterung von Messwerten wird ein Median-Filter mit jeweils parametrisierter Länge verwendet, anstelle eines ursprünglich in der Roadmap geplanten Kalman-Filters. Grund dafür ist, dass die Implementierung der in der Roadmap beschriebenen Funktionen eine einheitliche Kommunikation voraussetzt. Die dort entstandene Entwicklungszeit wird durch den weniger aufwendigeren Median-Filter kompensiert.

3.5.3 Ultraschallsensoren

Da die Ultraschallsensoren über das ADTF zu schnell abgefragt werden, kann ein Sensor das Echo eines anderen Sensors detektieren und damit eine falsche Abstandmessung ausgeben. Solche Fehlmessungen können dazu führen, dass das Auto während der Fahrt ohne augenscheinlich erkennbaren Grund seine Geschwindigkeit verringert, da vermeintliche Hindernisse erkannt werden.

3.5.4 Beschleunigungssensor

Die Verwendung des Beschleunigungssensors setzt einen angepassten Kalman-Filter voraus, da bereits Vibrationen des Motors ausreißende Messwerte von bis zu 2 G erzeugen. Deshalb ließ sich die geplante Funktion, Stoppen des Motors bei anheben des Fahrzeugs, nicht ohne weiteres realisieren.

4 Auswertung

Das Auto folgt, wie in der Zielsetzung beschreiben, einer blauen Linie und passt seine Geschwindigkeit erkannten Hindernissen bis zum Halt in gegebenem Mindestabstand an.

Der Einsatz eines Median-Filters anstatt eines Kalman-Filters führt zu keinen Einschränkungen.

Die Linienerkennung ist einfach implementiert und ist deshalb nur unter eng gefassten Rahmenbedingungen nutzbar. Zusätzliche blaue Farbbereiche im relevanten Sichtfeld werden eventuell als Fahrbahnmarkierung interpretiert.

Durch die erstmalige Verwendung des Modells zu Lehrzwecken, ist zu wenig über die genaue Funktionsweise aller gegebener Funktionen bekannt. Deshalb floss zu viel Arbeitszeit in die Einarbeitung und Auseinandersetzung mit ADTF, anstelle von eigentlicher Filterimplementierung.

5 Ausblick

Das Projekt bietet viel Potenzial für Erweiterungen.

Im Baustein *OneLineDetect* beispielsweise wurde bereits eine zweite Funktion implementiert, die es ermöglicht, die zwei breitesten Linienquerschnitte in einer Bildzeile zu ermitteln und direkt eine entsprechende Entscheidung zu treffen, welche weiter verarbeitet werden soll. Nach Anpassung der globalen Schnittstellen könnten später bei Erkennung mehrerer Linien diese weiter gereicht und vom zentralen Logik-Baustein entsprechend im Sinne einer Abbiege-Entscheidung verwertet werden.

Auch die Gewichtung der gemessenen Ultraschall-Abstandsmesswerte könnte im nächsten Schritt nicht nur, wie bislang, gemäß eingeschlagenem Lenkwinkel im Sinne einer seitlichen Relevanzverschiebung angepasst werden, sondern auch entsprechend der aktuellen Geschwindigkeit generell mehr oder weniger empfindlich gemacht werden (dynamisches Stauchen/ Strecken der Gauss-Kurve).

Es bleibt zu hoffen, dass die anfänglich investierte Zeit des Aufsetzens der und Einarbeitens in die Umgebung weitere Arbeit an weiteren Projekten am Audi-Modell problemorientierter und -fokussierter ermöglichen kann.

Abbildungsverzeichnis