Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Nils Kruse

Kameragestützte Fahrspurerkennung für autonome Modellfahrzeuge

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Nils Kruse

Kameragestützte Fahrspurerkennung für autonome Modellfahrzeuge

Bachelorarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Technik und Informatik
Studienrichtung Technische Informatik
am Fachbereich Elektrotechnik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Stephan Pareigis
Zweitgutachter : Prof. Dr.-Ing. Andreas Meisel

Abgegeben am 8. September 2008

Nils Kruse

Thema der Diplomarbeit

Kameragestützte Fahrspurerkennung für autonome Modellfahrzeuge

Stichworte

TFALDA, Photonfocus, ARM7, CAN, LPC, intelliTruck, Fahrspurerkennung, PWM, Servomotor, Sobel-Operator

Kurzzusammenfassung

Diese Arbeit beschäftigt sich mit der Entwicklung und Implementierung eines Fahrspurerkennungssystems für ein sich autonom bewegendes Modellfahrzeug. Mittels einer Kamera werden in Fahrtrichtung liegende Bilder auf einen an Board befindlichen PC übertragen, welcher mit einer Software eine ausgelegte Fahrspur erkennt. Die ermittelten Daten werden ausgewertet und ein daraus berechnete Lenkwinkel per CAN-Bus an einen Microcontroller übertragen, welcher die Ansteuerung des Lenkservos übernimmt.

Nils Kruse

Title of the paper

Camera based lane detection system for autonomous model vehicles

Keywords

TFALDA, Photonfocus, ARM7, CAN, LPC, intelliTruck, lane detection, PWM, servomotor, sobel operator

Abstract

This thesis is about the development and the implementation of a lane detection system for autonmous model vehicles. With a mounted camera pictures in direction of motion are transfered on a pc on board of the vehicle that detects an outlaid lane. The data will be evaluated and a calculated steering-angle is send via CAN to a microcontroller that assumes the control of the steering.

Danksagung

An dieser Stelle möchte ich all diejenigen erwähnen, ohne dessen Unterstützung diese Arbeit nicht möglich gewesen wäre. Ohne eine Wertung durch die Reihenfolge der Nennung zu bezwecken, möchte ich mich bei folgenden Menschen herzlich bedanken.

Ein Dank gilt meinen beiden Betreuern. Sie ermöglichten mir meine Bachelorarbeit als Abschluss meines Studiums an der HAW-Hamburg anzufertigen und begleiteten mich in dieser letzten Phase. Ein Dank an:

Prof. Dr. rer. nat. Stephan Pareigis
Prof. Dr.-Ing. Andreas Meisel

Für eine unvergessliche Zeit während des Studiums möchte ich mich bei Martin Arvidsson, Stefan Gahr, Felix Kolbe, Filip Nowacki, Andrej Rull und Manuel Trittel bedanken.

Dem gesamten „intelliTruck“-Team gilt ein Dank für die Unterstützung während der Umsetzung meiner Arbeit, sowie allgemein für eine lustige Zeit.

Herrn Lohmann danke ich für seine tatkräftige Unterstützung bei der Inbetriebnahme des Microcontrollers.

Nicht zu vergessen sind alle meine Freunde, die mir durch ihre Motivation stets eine sehr große Hilfe waren.

Abschließend geht ein besonderer Dank an meine Familie. Ohne euch wäre all dies nicht möglich gewesen. Ich danke euch für alles!

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1. Motivation

Die EU-Kommission treibt mit ihrem e-Safety Programm durch die Forderung nach der Halbierung der Unfallzahlen bis zum Jahr 2010 die Einführung aktiver elektronischer Assistenzsysteme stark voran (vgl. Kirchner u. a., 2005, S.718). Mit den Assistenzsystemen soll der Häufigkeit und Schwere von heutigen Verkehrsunfällen entgegengewirkt werden. Nicht zuletzt möchte man somit die Zahl der Verkehrstoten eindämmen.

1.2. Stand der Technik

Die Entwicklung von Assistenzsystemen spielt im Automobilbau schon seit langer Zeit eine entscheidende Rolle. Mit Hilfe solcher Systeme soll dem Fahrer das führen eines Fahrzeuges erleichtert werden. So wurde schon in der frühen Automobilgeschichte der Bremskraftverstärker entwickelt, welcher dem Fahrer das Bremsen und somit das kontrollieren eines Fahrzeugs in schwierigen Situationen erleichtern soll.

Zu beginn der 70er Jahre wurde zunehmend Wert auf den Einzug elektronisch unterstützter Assistenzsysteme gelegt. So wurde im Jahr 1978 von der Firma Bosch ein elektronisches Antiblockiersystem auf den Markt gebracht. Über die Jahre folgten Systeme wie ACC (Adaptive Cruise Control), ESP (Elektronisches Stabilitäts Programm) oder AFIL (Alarm bei Fahrspurabweichung durch Infrarot Linienerkennung). Weiterreichende Informationen zu diesen Systemen finden sich bei Bosch (2008) und Citroën (2008)

Sehr weit fortgeschritten bei der Entwicklung von Assistenzsystemen ist unter anderem die Volkswagen AG. Diese hat im Sommer diesen Jahres ein System mit dem Namen „Lane Assist“ herausgebracht. Es handelt sich dabei um einen kameragestützten Assistenten, der den Farhzeugführer beim Halten seiner Spur unterstützt. Weitere Informationen dazu finden sich bei Volkswagen (2008).

1.3. Aufgabenstellung

Ziel dieser Arbeit ist die Entwicklung eines dem „Lane Assist“ ähnlichen Systems, welches jedoch noch weiter über den lediglich korrigierenden Eingriff dieses Systems hinaus geht. Es soll eine vollständig autonome Steuerung der Fahrzeuglenkung ermöglicht werden.

Diese Arbeit setzt an den Erkenntnissen aus der Arbeit „Bildverarbeitungsmodul zur Fahrspurerkennung für ein autonomes Fahrzeug“ (Kant (2007)) an. Durch die verwendete Hough-Transformation ist die Fahrspurerkennung zu Anfällig gegenüber anderen sich in dem Kamerabild befindlichen Kanten. Aus diesem Grund soll für die Fahrspurerkennung ein anderer Algorithmus implementiert werden, der weniger Anfällig gegenüber eines solchen Problems ist, aber dennoch performant genug ist um bei hohen Fahrzeuggeschwindigkeiten eingesetzt werden zu können. Angeboten hat sich dafür der in der Studienarbeit „Fahrspurerkennung mit Three Feature Based Lane Detection Algorithm (TFALDA)“ (Berger (2008)) verwendete Algorithmus. Dieser hat sich dort als sehr Robust erwiesen.

Die Aufgabe dieser Arbeit ist daher den TFALDA in eine Fahrzeugarchitektur zu implementieren, die dafür bestimmt ist sich autonom entlang einer einzelnen ausgelegten Fahrspur zu bewegen.

1.4. Kapitelübersicht

Das Kapitel 2 beinhaltet die Grundlagen zum verwendeten Alorithmus zur Fahrspurerkennung und zur Bildverarbeitung auf denen diese Arbeit aufbaut.

Kapitel 3 fasst zusammen, welche Anforderungen an die Fahrspurerkennung in Hinblick auf die Entwicklung ihrer gestellt werden.

Das 4. Kapitel beschreibt, welche Hardware zur Implementierung der Fahrspurerkennung verwendet wurde und wie das gesamte System aufgebaut ist. Weiterhin wird aufgeführt, warum sich für ein bestimmtes Bauteil entschieden wurde, wenn mehrere ähnliche zur Verfügung standen.

In Kapitel 5 wird beschrieben, wie die Fahrspurerkennung umgesetzt wurde. Es wird beschrieben, wie die Bilder der Kamera aufgearbeitet werden, wie der TFALDA implementiert wurde, wie die Fahrspur aus den Bildern berechnet wird und wie anhand der gewonnenen Daten der anzusteuernde Lenkwinkel bestimmt wird.

Danach werden in Kapitel 6 die durchgeführten Test aufgeführt und die Ergebnisse der Tests dargestellt.

Im abschließenden Kapitel wird ein kurzes Fazit zur implementierten Fahrspurerkennung, sowie eine Übersicht über mögliche Verbesserungen an dem System gegeben.

2. Grundlagen

2.1. TFALDA

Der „Three-Feature Based Automatic Lane Detection Algorithm“ (im Folgendenen nur noch kurz als TFALDA bezeichnet) ist ein simpler, robuster und efizienter Algorithmus welcher dafür ausgelegt wurde in Echtzeit in einem aufgenommenen Bild eine Fahrspur zu erkennen. Er besteht aus drei Komponenten welche in Yim und Oh (2003) wie folgt beschrieben werden:

2.1.1. Pre-Processing

Bevor man ein Bild auf eine mögliche Fahrspur untersucht ist es sinnvoll dieses nach den eigenen Bedürfnissen zu verändern. Ein erster Ansatz ist es, dass man sich auf einen oder mehrere Bereiche beschränkt, die für die Fahrspurermittlung von Interesse sind, im folgenden kurz als ROI (Region of interrest) bezeichnet.

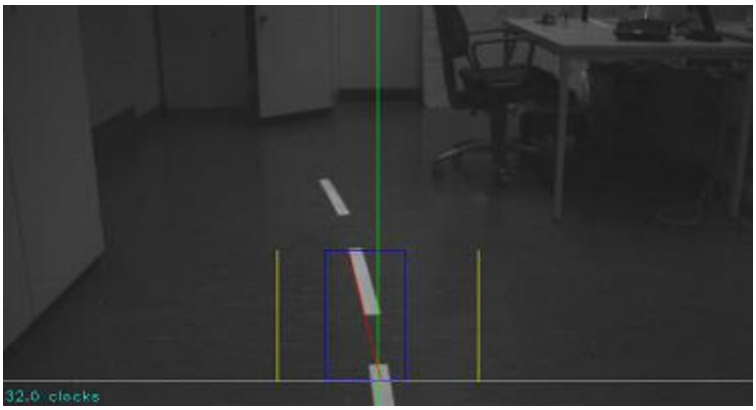

Abbildung 2.1.: Pre-Processing: Ganzes Bild

Abbildung 2.2.: Pre-Processing: Bildausschnitt

Da heutige Kameras üblicherweise eine recht hohe Auflösung haben und eine hohe Anzahl von Bildpunkten eine verhältnismäßig lange Bearbeitungsdauer bedeuten empfiehlt es sich das ROI um einen geeigneten Faktor zu verkleinern. Dadurch verringert sich die Anzahl der zu bearbeitetenden Bildpunkte und somit auch die Verarbeitungszeit eines jeden Bildes. Als

Abbildung 2.3.: Pre-Processing: Resampling

letzter Schritt wird das ROI nun gefiltert um die darin enthaltenen Kanten hervorzuheben. Diese werden sehr hell eingezeichnet. Überflüssige Informationen werden ausgeblendet.

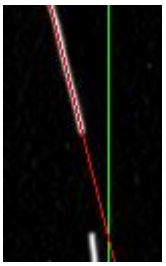

Abbildung 2.4.: Pre-Processing: Kantenverstaerkung

2.1.2. Automatic Lane Detector

Diese Komponente untersucht das bis hierhin vorverarbeitete Bild nun auf alle möglichen Fahrspuren, im folgenden als lane-candidate bezeichnet. Dabei wird jeder mögliche lane-candidate in einem dreidimensionalen Merkmalsraum dargestellt. Die drei Dimensionen des Merkmalsraums sind dabei wie folgt definiert: Richtung *D*, Intensität *I* und Startpunkt *P*. Ein ROI der Breite *n* Pixel hat entsprechend *n* lane-candidates. Der lane-candidate *Ci* ist dabei durch folgenden Vektor (vgl. 2.1) beschrieben:

$$C_i = \begin{pmatrix} P(C_i) \\ I(C_i) \\ D(C_i) \end{pmatrix}; \qquad \forall i \in \mathbf{A} \tag{2.1}$$

Dabei soll folgendes gelten:

- $A = 0..n; \qquad \forall \mathbf{A} \in Z^+$
  A sei definiert als die Menge aller ganzer positiver Zahlen von 0 bis n mit n als maximaler Breite des ROI
- $P(C_i) = i$
  $P(C_i)$ sei definiert als der Startpunkt *i* des lane-candidates aus der Menge **A**
- $I(C_i) = \sum \overline{P_i Q_i}$
  $I(C_i)$ sei definiert als die aufsummierte Pixelintensität auf der Strecke $\overline{P_i Q_i}$
- $D(C_i) = Q_{j,x} - P_{i,x}$
  $D(C_i)$ sei definiert als die Subtraktion der X-Werte der Punkte p_i und q_j. Dies sind die Start- und Endpunkte der Strecke mit der höchsten Pixelintensität.

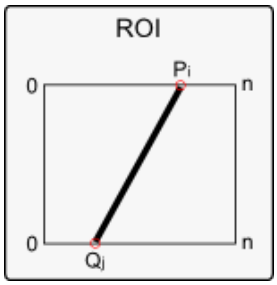

Abbildung 2.5.: Automatic Lane Detection: Aufbau des ROI

2.1.3. Lane Interference System

Diese dritte und letzte Komponente hat zwei Aufgaben. Zum einen fügt es bei der verwendung von zwei oder mehr ROIs alle gefunden Informationen über die Fahrspur zusammen, um dieser dann entsprechend folgen zu können. Zum anderen dient es als Fangnetz um sicherzustellen, dass nur plausible Fahrspuren verfolgt werden. So ist es beispielsweise möglich große Sprünge zwischen einer aktuell und einer zuvor erkannten Fahrspur als ungültig zu deklarieren. Weitere Plausibilitätsprüfungen sind möglich.

2.1.4. Die I-Operation

Als I-Operation wird jener Programm-Abschnitt bezeichnet, welcher die Pixelintensität eines lane-canditates errechnet. Es wird von jedem Punkt P_i zu jedem Punkt Q_j ein Vektor aufgespannt und die Intensitäten der Grauwerte addiert. Erkennbar ist dies in folgendem Code-Ausschnitt:

Listing 2.1: I-Operation

```
1  for(i=roi[0]; i<roi[1]; i++) {
2    sumGreyValues = 0;
3
4    for(j=roi[0]; j<roi[1]; j++) {
5      // Copy line out of image
6      cvSampleLine(image, pt1, pt2, buffer, 8);
7
8      sumGreyValues_tmp = 0;
9      for(k=0;k<500;k++)
10       // Sum gray values
11       sumGreyValues_tmp += buffer[k];
12
13     if(sumGreyValues_tmp > sumGreyValues) {
14       sumGreyValues = sumGreyValues_tmp;
15       j_max = j;
16     }
17   }
18 }
```

2.1.5. Komplexität

Betrachtet man die I-Operation des TFALDA, so fallen einem die zwei ineinander verschachtelten *for*-Schleifen ins Auge. Diese beiden Schleifen laufen jeweils von 0 bis n. Daraus lässt sich eine Komplexität von $O(N^2)$ erkennen. Da die übrigen Teilfunktionen des Algorithmusses ein konstantes Komplexitätsverhalten aufweisen bleibt es im gesamten bei einem quadratischen Komplexitätsverhalten.

2.2. Faltung von Bildern

Um Bilder zu glätten, zu schärfen oder um enthaltene Kanten zu verstärken werden diese gefaltet. Zur Berechnung des Grauwerts eines Zielbildpunktes werden die Grauwerte des gleichen Bildpunkts im Quellbild sowie um diesen herumliegende Bildpunkte herangezogen. Eine Maske (auch als Faltungskern bezeichnet) legt fest, welche Bildpunkte dabei herangezogen werden und mit welchem Faktor jeder multipliziert werden muss um den Zielgrauwert zu berechnen.

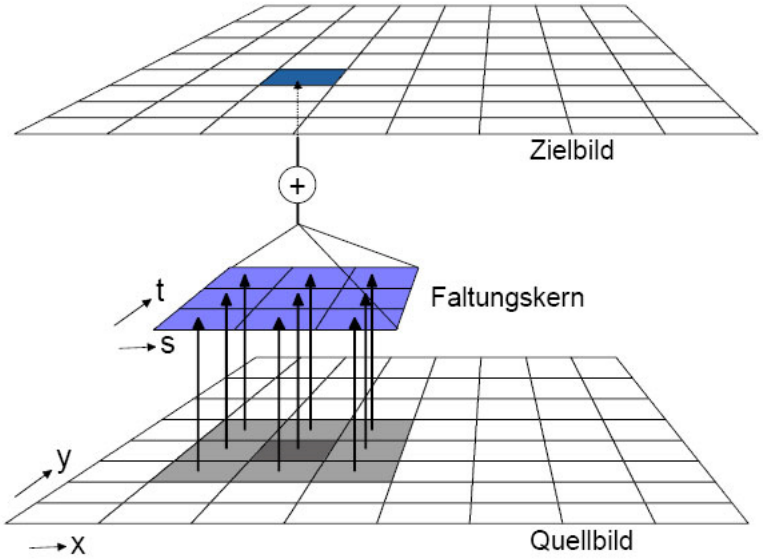

Abbildung 2.6.: Faltung mit einem Faltungskern [1)]

In der Abbildung 2.6 ist der Vorgang der Faltung visualisiert. Als Beispiel dient hier eine 3x3-Faltungkern. Es wird spalten- und zeilenweise über das Zielbild iteriert. Für jeden Zielbildpunkt werden nun die Bildpunkte des Quellbilds, die von der Maske abgedeckt werden mit den Werten in der Maske multipliziert und deren Produkte anschließend addiert. Diese Summe bildet dann den neuen Grauwert des Zielbildpunkts.

[1] Quelle: http://www.informatik.haw-hamburg.de/fileadmin/Homepages/ProfMeisel/Vorlesungen/WP_RobotVision/V/RV03.pdf; S.1

Die Felder in einem Faltungskern sind wie in Tabelle 2.1 bezeichnet:

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Tabelle 2.1.: Bezeichnung innerhalb eines 3x3-Faltungskerns

Zur Berechnung eines Zielbildpunktes mit Hilfe eines 3x3-Faltungskerns wird die folgende Formel verwendet:

$$\widetilde{g}(x,y) = \begin{pmatrix} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 & w_9 \end{pmatrix} * \begin{pmatrix} g(x-1,y+1) \\ g(x,y+1) \\ g(x+1,y+1) \\ g(x-1,y) \\ g(x,y) \\ g(x+1,y) \\ g(x-1,y-1) \\ g(x,y-1) \\ g(x+1,y-1) \end{pmatrix} \tag{2.2}$$

Dabei gilt:

- $\widetilde{g}$: Zu berechnender Grauwert des Zielbildes
- g: Grauwert im Quellbild
- w_n: Wert in der Faltungsmatrix (siehe Tabelle 3.1)
- x,y: Koordinaten im Quell- und Zielbild

Für weitere Informationen zu Faltungsfiltern empfiehlt sich ein Blick in Dössel (2000) auf Seite 96 und folgende.

2.3. Pulsweitenmodulation

Unter einem PWM-Signal ist ein Signal zu verstehen, welches zyklisch stets einen „high“- und einen „low“-Pegel ausgibt. Die Periodendauer der Pegelwechsel sowie die Pulsweite können dabei individuell eingestellt werden.

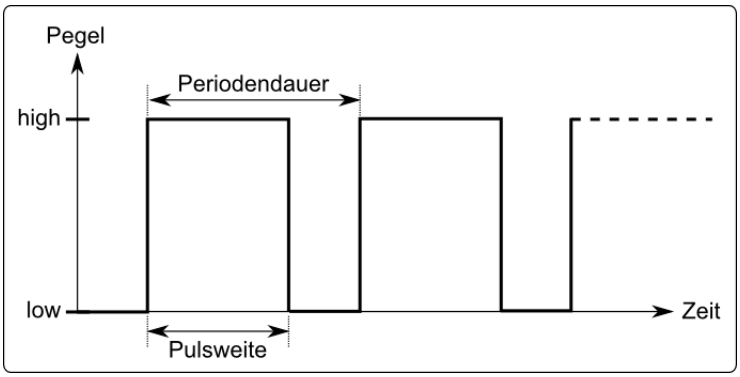

Abbildung 2.7.: Beispiel eines PWM-Signals

Maßgebend für die Periodendauer (vgl. Abb. 2.7) ist die Bitauflösung der PWM-Stufe im Microcontroller. Mit einer 8-Bit PWM-Stufe kann die Periodendauer maximal auf $2^8 = 256$ Werte aufgelöst werden. Das bedeutet, dass mindestens ein Takt und höchstens 256 Takte zur Auflösung der Periodendauer verwendet werden können.

Der Takt der PWM-Stufe leitet sich aus der Taktfrequenz des Prozessors ab. Daher kann der PWM-Takt höchstens dem Prozessortakt entsprechen. Niedrigere Taktungen sind über das einschalten eines entsprechenden Vorteilers möglich.

Die Pulsdauer bestimmt für welche Dauer ein „high“-Pegel anliegt. Diese auch als Einschaltzeit bezeichnete Zeitspanne wird im folgenden mit t_{ein} abgekürzt.

Weitere Detailinformation finden sich in (Wallentowitz, 2006, S.542f).

2.4. Modellfahrzeug „intelliTruck“

Abbildung 2.8.: Der „intelliTruck“

Bei dem Fahrzeug handelt es sich um ein handelsübliches Modellfahrzeug im Maßstab 1:5. Es ist mit einem 27ccm großen und 2,6PS starkem Verbrennungsmotor ausgestattet und schafft unter optimalen Bedingungen eine Höchstgeschwindigkeit von 65km/h. Für die Umsetzung dieser Arbeit wurde das Fahrzeug mit einem kleinen aber leistungsstarken PC sowie mit einer auf dem Dach montierten und in Fahrtrichtung ausgerichteten Kamera ausgerüstet. Weitere Informationen zum Fahrzeug finden sich auf der Homepage des FAUST-Projekts der HAW-Hamburg (FAUST (2008)).

3. Anforderungen an die Fahrspurerkennung

Im folgen sollen kurz die Randbedingungen definiert werden, unter denen die Fahrspurerkennung entwickelt werden soll.

3.1. Witterungseinflüsse

Bei dem Einsatz der Fahrspurerkennung innerhalb von Gebäuden ist es möglich das System auf bestimmte Boden- und Lichtverhältnisse einzurichten. Da das Fahrzeug aber im Outdoor-Bereich eingesetzt werden soll hat man selbst kein Einfluss auf diese Verhältnisse. Man ist abhängig von den äußeren Witterungseinflüssen. Daher muss das System in der Lage sein eine Fahrspur trotz einfallender Lichter und Schatten, sowie bei wechselhafter Bodennässe korrekt zu erkennen.

3.2. Geschwindigkeit

Da direkt von der Fahrzeuggeschwindigkeit abhängig ist, wie weit sich das Fahzeug innerhalb eines Berechnungszyklusses bewegt ist ein performanter Algorithmus erforderlich.

Die maximale Geschwindigkeit des serienmäßigen Fahrzeugs ist Herstellerseitig mit 65km/h angegeben. Bedingt durch die zusätzliche Elektronik, liegt die tatsächliche Höchstgeschwindigkeit zwar unter diesem Wert, allerdings wird weiterhin mit ihm gerechnet, da der tatsächliche Wert durch einen sich ständing ändernden Fahrzeugaufbau bisher nicht ermittelt wurde.

Der Umrechnungsfaktor von *km/h* auf *m/s* ist *1/3,6*.

$$\Rightarrow v_{max} = \frac{65 km/h}{3,6} \approx 18 m/s \tag{3.1}$$

Wird davon ausgegangen, dass das Fahrzeug sich mit voller Geschwindigkeit bewegt, so legt es in jeder Sekunde eine Strecke von maximal *18m* zurück. Zwar wird das System unter diesem Aspekt entwickelt, jedoch wird es zum Zeitpunkt der Entwicklung nur mit Schrittgeschwindigkeit bewegt um bei einem Fehlverhalten schnell eingreifen zu können.

3.3. Unmpfindlichkeit

Da das Fahrzeug im Outdoor-Bereich eingesetzt wird ist es erforderlich das komplette System vor Erschütterungen, Feuchtigkeit und Staub zu Schützen. Insbesonders gilt dies für die Kamera, da diese auf dem Fahrzeug montiert wird. Auch einem Speichermedium, welches unempfindlich gegenüber Erschütterungen durch den Motor und das Terrain ist, bedarf es.

3.4. Fahrstreifenabstand

Bei der Auslegung einer gestrichelten Fahrsur ist sich an den Schmalstrich-Leitlinien an Knotenpunkten im normalen Straßenverkehr zu orientieren. „Am Knotenpunkt werden die ankommenden Verkehrsströme aufgegliedert und neu gebündelt 'gesammelt' oder 'verteilt'; am Knotenpunkt ist auch deren Richtungsänderung möglich (Siehe dazu: (Korda, 2005, S.265))“. Die Leitlinien werden dort wie folgt markiert:

Abbildung 3.1.: Markierung von Schmalstrich-Leitlinen

Streifenbreite	15cm
Streifenlänge	100cm
Streifenabstand	100cm

Tabelle 3.1.: Markierungsverhältnis bei Schmalstrich-Leitlinen

Die Bemaßungen (vgl. Tab. 3.1) sind der Vorgabe für Fahrbahnmarkierungen aus (Zeppelin, 2006, S.181) entnommen.

Das Fahrzeug hat wie im Grundlagenkapitel beschrieben einen Maßstab von 1:5. Dieser wird ebenfalls für die Fahrstreifen übernommen. Es ergeben sich daher folgende Parameter für die Fahrstreifen:

Streifenbreite	5cm
Streifenlänge	20cm
Streifenabstand	20cm

Tabelle 3.2.: Markierungsverhältnis bei Schmalstrich-Leitlinen im Maßstab

4. Technische Ausgangslage

In diesem Kapitel wird ein kleiner Einblick auf die verwendete Hardware, deren Aufbau, sowie den Abarbeitungsvorgang im System vermittelt.

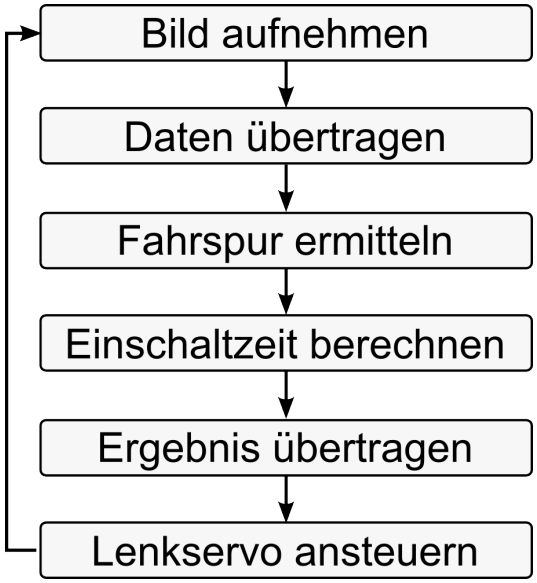

Abbildung 4.1.: Technische Ausgangslage: Abarbeitungsvorgang

Eine Kamera nimmt in Fahrtrichtung liegende Bilder auf und sendet diese an einen PC. Dieser übernimmt die Auswertung der Bilddaten und die Erkennung der Fahrspur. Ist eine Fahrspur erkannt wird der anzusteuernde Lenkwinkel mittels einer CAN-Nachricht an einen Microcontroller übertragen, der die Lenkung des Fahrzeugs ansteuert.

4.1. Verfügbare Kameras

Zur Auswahl standen bei der Umsetzung der Fahrspurerkennung zwei verschiedene Kameras. Im folgenden werden diese kurz erläutert, sowie derer Vor- und Nachteile aufgezeigt.

4.1.1. IDS uEye LE

Abbildung 4.2.: IDS uEye LE

Die erste Kamera ist die „uEye LE“ aus dem Hause IDS. Es handelt sich um eine sehr kompakte und kostengünstige Kamera die in professionellen Bereichen eingesetzt wird. Sie arbeitet mit CMOS-Sensor und liefert ein monochromes Video-Signal. Bei einer Auflösung von 752×480px schafft sie eine Bildwiederholrate von 87fps. Im Folgenden eine Auflistung der wichtigsten Daten zu der Kamera:

Technologie	CMOS-Sensor
Auflösung	752×480px
Farbformat	8-Bit Monochrom
Bildwiederholrate	87fps
Schnittstelle	USB 2.0 (full speed)
Spannungsversorgung	+5V DC
Stromverbrauch	ca. 115±15mA
Abmessungen	ca. 35×35×30mm

Tabelle 4.1.: Wichtige Daten der uEye LE im Überblick

Weitere Detailinformationen finden sich in IDS-Imaging (2008) (*S.*2).

Zwar ist die Kamera von ihren Daten her Leistungsstärker als die zweite verfügbare Kamera und profitiert ihr gegenüber von einer Spannungsversorgung über den USB-Port, jedoch

bringt sie auch zwei mehr oder weniger schwerwiegende Nachteile mit sich. Als erstes hat die Kamera (vgl. Abbildung 4.2) kein Gehäuse und ist somit nur bedingt für den Einsatz im Freien und noch schlechter für den Einsatz im Gelände nutzbar. Des Weiteren stellte sich bei Probeaufnahmen heraus, dass sie sehr Lichtempfindlich ist und die Bilder somit unter bestimmten Lichtverhältnissen zu sehr überbelichtet werden. Damit ist keine vernünftige Fahrspurerkennung realisierbar.

4.1.2. PhotonFocus MV-D750E-20-U2

Abbildung 4.3.: PhotonFocus MV-D750E-20-U2 USB-Kamera

Als zweites steht eine Kamera zur Auswahl, die normalerweise in der industriellen Bildverarbeitet verwendet wird. Sie ist nicht ganz so leistungsstark, wie das erste Modell. Jedoch reicht eine Bildwiederholrate von 60fps bei einer Auflösung von 750×400px immernoch problemlos aus, sodass die Vorzüge der Kamera ganz klar im Vordergrund stehen. Vorteilhaft ist, dass die Kamera in einem robusten Gehäuse aus Aluminium gekapselt ist. Somit ist sie unempfindlich gegenüber dem Einsatz im Freien. Weiterhin stehen verschiedene Objektive (Normal, Tele, Weitwinkel), die jeweils über eine Blende verfügen zur Auswahl. Dadurch hat man die Möglichkeit die Kamera selbst bei sehr hellen Lichtverhältnissen einsetzen zu können. Nachfolgend ein Ausschnitt aus Photonfocus (2008) $(S.2)$ um einen kurzen Überblick über die wichtigen Daten der Kamera zu bekommen:

Technologie	Aktivpixel-CMOS-Sensor
Auflösung	750×400px
Farbformat	8-Bit Monochrom
Bildwiederholrate	60fps
Schnittstelle	USB 2.0 (full speed)
Spannungsversorgung	+12V DC ($\pm 10\%$)
Leistungsaufnahme	2.2W
Abmessungen	ca. 55×55×43mm

Tabelle 4.2.: Wichtige Daten der Kamera im Überblick

Die Spannungsversorgung geschieht normalerweise über ein mitgeliefertes Netzteil - für den Betrieb auf dem Fahrzeug wird diese jedoch umgerüstet um einen direkten Betrieb an 12V-Akkus zu ermöglichen. Das mitgelieferte Software-Packet stellt alle nötigen Programme zum Betrieb der Kamera bereit und enthält weiterhin ein SDK (Software-Development-Kit) mit dem die Kamera mit wenigen Schritten in eigene Softwareprojekte integriert werden kann.

Leider weist die Kamera einen Defekt in Form einer komplett ausgefallenen Pixel-Zeile auf. Weiterhin waren sporadisch auftretende vereinzelte Pixelfehler unterhalb der betreffenden Zeile zu vermerken. Um die fehlerbehafteten Bereiche nicht mit in die Fahrspurerkennung einzubeziehen wurde bei *Y=375px* ein Horrizont festgelegt. Nur Bilddaten, die sich über diesem Horrizont befinden werden verarbeitet, da diese fehlerfrei sind.

4.2. Der Computer

Abbildung 4.4.: Der PC

Für den PC kommen handelsübliche Bauteile zum Einsatz. Als Prozessor dient eine 2,2GHz Dual-Core CPU aus dem Hause Intel um hohen Rechenanforderungen gewachsen zu sein. Der Arbeitsspeicher wurde mit 1GB dimensioniert. Um ein Speichermedium nicht durch Erschütterungen zu zerstören verfügt der PC über eine 8GB große Solid-State-Disc. Da alles auf kleinstem Raum untergebracht werden muss wurde die Hardware auf einem Mainboard mit μATX-Formfaktor vereint. Weiterhin wurde eine W-LAN-Karte verbaut um via Remote-Konsole Zugriff auf das Fahrzeug zu haben. Die Spannungsversorgung erfolgt direkt über die 12V-Boardspannung.

4.3. Das CAN-Interface

Abbildung 4.5.: Das CAN-Interface

Bei dem ausgewählten „USB-to-CAN Compact“ aus dem Hause IXXAT handelt es sich um ein Interface, welches gleich mehrere Vorzüge mit sich bringt. Als erstes weist er eine sehr kompakte Bauform auf. Dadurch kann es sehr einfach auf dem Fahrzeug installiert werden. Weiterhin wird es lediglich über eine USB-Schnittstelle mit dem PC verbunden. Diese übernimmt zugleich die Stromversorgung - dadurch fallen zusätzliche Kabel weg. Möglich ist dies, da das Interface nur einen sehr kleinen Strom benötigt und dieser über den USB-Port abgegriffen werden kann. Trotzdem ist es mit einem leistungsfähigem Microcontroller ausgestattet, welcher in der Lage ist trotz hoher Bit- und Nachrichtenraten alle Nachrichten problemlos zu empfangen. Zu guter Letzt ermöglicht das mitgelieferte SDK ein einfaches und schnelles Implementieren in eigene Softwareprojekte. Detailiertere Informationen zu dem CAN-Interface sind auf der entsprechenden Internetseite des Herstellers zu finden [IXXAT (2008)].

PC-Businterface	USB 2.0 (full speed)
CAN-Controller	Philips SJA 1000
Mögliche Bit-Raten	10kBaud bis 1MBaud
Spannungsversorgung	Per USB (Verbrauch: ca. 250mA)
Abmessungen	ca. 80×45×20mm

Tabelle 4.3.: Wichtige Daten des CAN-Interface im Überblick

4.4. Verfügbare Microcontroller

4.4.1. Atmel AT90CAN128 (AVR)

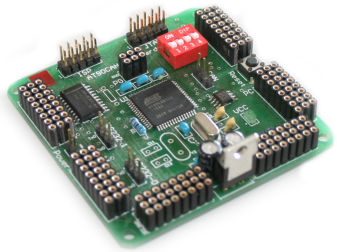

Abbildung 4.6.: Der Microcontroller

Als erstes stand der im Labor verfügbare AT90CAN128 Microprozessor aus dem Hause Atmel für die Umsetzung der Lenkansteuerung zur Auswahl. Der Vorteil dieses Prozessors ist, dass es ihn schon über einen sehr langen Zeitraum gibt und es somit unzählige Informationsquellen im Internet, sowie viele Erfahrungen durch den Einsatz in Vorlesungen an der HAW-Hamburg gibt. Eine kurze Übersicht über die technischen Daten findet sich in Tabelle 4.4. Weitere Detailinformationen finden sich in (Atmel, 2006, S.1f).

Architektur	Enhanced RISC
Datenbreite	8Bit
Prozessortakt	16MHz
Speichergröße (Flash)	128kByte
Schnittstellen	CAN (2.0 A/B), 2xUSART
Besonderheiten	Keine FPU

Tabelle 4.4.: Wichtige Daten des AT90CAN128 im Überblick

Zwar reichen die 8-Bit-Architektur und die Taktfrequenz von 16MHz zum übertragen der Lenkwinkel und somit für diese Arbeit aus, jedoch stößt der Prozessor in Hinblick auf andere Aufgaben an seine Grenzen. So reichte er Beispielsweise für die Arbeit von Nicolai Glatz (vgl. Glatz (2008)) bei der Ansteuerung eines Radarsensors nicht mehr aus. Deshalb ist der Einsatz eines leistungsstärkeren Microprozessors vorgesehen. Zwar gibt es auch leistungsstärkere Prozessoren mit verschieden Features aus dem Hause Atmel, jedoch sind diese nicht im Labor verfügbar.

4.4.2. NXP LPC2468 (ARMv7)

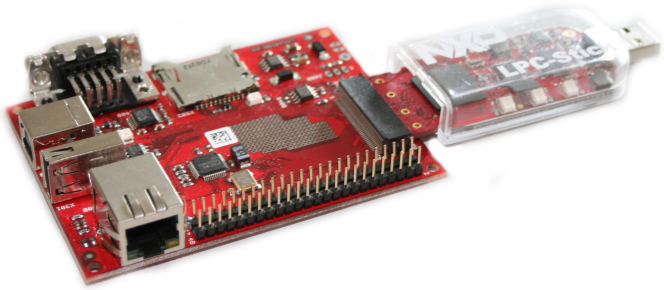

Abbildung 4.7.: Der Microcontroller

Der zweite im Labor verfügbare Microcontroller ist der auf dem ARMv7 basierende LPC2468 aus dem Hause NXP. Wegen der höheren Rechenleistung von 72MHz, der 32-Bit-Architektur, sowie der Vielzahl verschiedener Schnittstellen bietet dieser im direkten Vergleich eine bessere Basis für das Fahrzeug. Weiterhin besteht die Möglichkeit Echtzeitbetriebssysteme (kurz RTOS) (siehe dazu eCos (2008)) auf dem Controller zu installieren. Die Softwareentwicklung findet in der Entwicklungsumgebung „HiTOP“ statt, welche von der Firma Hitex mitgeliefert wird. Eine kurze Übersicht über die technischen Daten findet sich in Tabelle 4.5. Weitere Detailinformationen sind im Datenblatt ((NXP, 2008, S.1f)) zu finden.

Architektur	Enhanced RISC
Datenbreite	32Bit
Prozessortakt	72MHz
Flash	512kByte
Schnittstellen	CAN (2.0 A/B), Ethernet, 3x I^2C, I^2S, 2xSSP, 4xUART, USB
Besonderheiten	Keine FPU, RTOS fähig

Tabelle 4.5.: Wichtige Daten des LPC2468 im Überblick

5. Umsetzung der Fahrspurerkennung

Ursprünglich wurde der TFALDA in Hinblick darauf entwickelt, dass sich das autonome Fahrzeug innerhalb von zwei Fahrstreifen bewegen soll. Gemäß Zielsetzung wurden bei dieser Arbeit in der Implementierung teile des Algorithmusses abgeändert um sich somit entlang nur eines Fahrstreifens bewegen zu können. Wie genau der Algorithmus implementiert wurde ist in folgendem Kapitel zu lesen.

5.1. Pre-Processing

Bildausschnitt festlegen

Da wie bereits in Kapitel 4.2 beschrieben ein verhältnismäßig leistungsstarker PC zum Einsatz kommt bräuchte man sich prinzipiell keine großen Gedanken über eine effiziente Software machen und könnte daher problemlos das gesamte aufgezeichnete Bild als Informationsquelle für die Fahrspurerkennung benutzen. Dies bringt jedoch einige Probleme mit sich. So würde es zu viel Zeit beanspruchen ein vollständiges Bild abzuarbeiten. Dadurch wäre es nicht ohne Weiteres möglich, die Software auf einem leistungsschwächeren System einzusetzen. Weiterhin würden sich zu viele Informationen im Bild befinden, sodass Kanten welche garkeine Fahrspur darstellen als eine solche erkannt werden könnten.

Unter Berücksichtigung dieser Punkte wird aus dem eingelesenen Bild ein bestimmter Bereich zur weiteren Verarbeitung freigestellt. Im nachfolgenden wird dieser Bereich nur noch kurz als ROI bezeichnet. Es wird mit einem Recheck aufgespannt, welches durch zwei Punkte definiert ist. Im Labor wurde dieser Bereich ermittelt indem ein Szenario erstellt wurde, von dem angenommen wird, dass es sich um eine realitätsnahe Startsituation handelt. Dieses Szenario ist durch folgende Parameter definiert:

Bildgröße	750px $\times$ 400px
Kamerahöhe	30cm
Kamerawinkel	-7°
Fahrstreifendistanz	30cm
Horrizont	Bei 375px in Y-Richtung

Tabelle 5.1.: Szenario zur Ermittlung des Start-ROIs

Aus dem unter diesen Bedingungen aufgenommenem Bild wurde dann ein Bereich bestimmt, der für den Start der Fahrspurerkennung ausreichend ist um den Startpunkt einer Fahrspur zu finden. Mit einbezogen wurde der in Kapitel 4.1.2 festgelegte Horrizont zur Unterbindung von Fehlinformation durch Pixelfehler.

Es ergaben sich zwei Punkte welche beim Start der Anwendung wie folgt festgelegt werden:

$$P_{1,x} : 320px, P_{1,y} : 300px$$
$$P_{2,x} : 428px, P_{2,y} : 374px$$

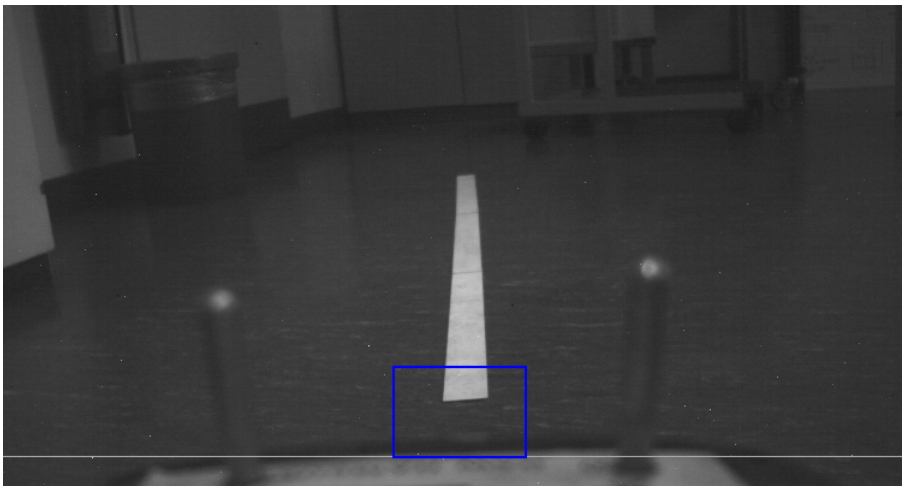

Abbildung 5.1.: Pre-Processing: Gesamtbild mit eingezeichnetem Start-ROI

Resampling

Auf Grund der hohen Auflösung der Kamera bietet es sich an das Bild zu skalieren. Dies bringt den Vorteil mit sich, dass die wesentlichen Bildinformationen erhalten bleiben, jedoch die Anzahl der Bildpunkte verringert wird, was einen weiteren Performancegewinn mit sich bringt. Es hat sich durch Ausprobieren verschiedener Faktoren gezeigt, dass die Fahrspur bei der Skalierung des ROIs um den Faktor 3 noch ausreichend zu erkennen ist. Daher wird dieser Faktor übernommen.

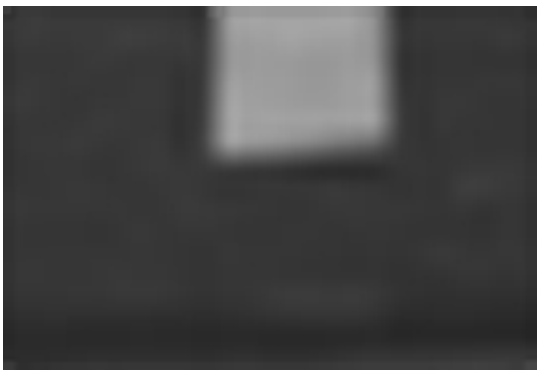

Abbildung 5.2.: Pre-Processing: Skaliertes ROI

Kantenverstärkung

Das festgelegte und skalierte ROI wird zum Schluss mit einem horrizontalen Sobel-Operator gefaltet um die enthaltenen Kanten zu verstärken. Berechnet wird das Zielbild mit Hilfe der folgenden Formel:

$$G_y = S_y * A = \frac{1}{8} * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \tag{5.1}$$

wobei:

- A: dem ROI als Ursprungsbild entspricht
- G_y: dem Zielbild entspricht

Es wird spalten- und zeilenweise über das Ursprungsbild iteriert und für jeden Pixel mit Hilfe der Faltungsmaske (3x3 Matrix) der Pixelwert des Zielbildes berechnet. Eine detailiertere Beschreibung des Verfahrens ist im Grundlagenkapitel unter Punkt 2.2 zu finden. Der verwendete Sobel-Operator ist näher in Bräunl (1995) beschrieben.

Abbildung 5.3.: Pre-Processing: Endergebnis nach Faltung

5.2. Einstellen der Kantengewichte

In Abschnitt B bis D der Veröffentlichung des TFALDAs ((Yim und Oh, 2003, S.221ff)) wird Beschrieben wie mit Hilfe von Testbildern sowie einem evolutionären Algorithmus die Kantengewichte bestimmt wurden.

Es wurden mehrere Testszenarien festgelegt und Bildersequenzen dieser Szenarien aufgezeichnet. Anschließend wurde manuell für jedes Bild bestimmt, wo sich die Fahrspur befindet. Danach wurde ein evolutionärer Algorithmus verwendet, der bei jedem Iterationsschritt die Kantengewichte verändert hat und für jedes Set von Kantengewichten die Fahrspurerkennung auf die aufgezeichneten Bilder angewendet hat. Dabei wurde aufgezeichnet wieviele Kanten jeweils bei welcher kombination der eingestellten Kantengewichten erkannt wurden. Als optimal wurden dann die Kantengewichte festgelegt, bei denen die meisten Fahrspuren erkannten wurden.

Für die von den Autoren gewählten Testszenarien ergaben sich dabei folgende Kantengewichte:

$$\begin{aligned} K_D &= 10{,}67 \\ K_I &= 1{,}00 \\ K_P &= 7{,}33 \end{aligned}$$

Die auf diese Weise ermittelten Kantengewichte wurden vorerst für diese Arbeit übernommen und im Labor getestet. Dabei stellte sich heraus, dass die Werte unter verschiedenen Bedingung ein gutes Ergebnis liefern. Aus diesem Grunde wurden die Werte auch für den Fortverlauf der Arbeit beibehalten.

5.3. Ermittlung der Fahrspur

Um die Fahrspur in einem Bild zu berechnen wird auf den bereits in Kapitel 3.1.2 beschriebenen Automatic Lane Detector zurückgegriffen. Die ebenfalls beschriebene I-Operation spannt von jedem P_i zu jedem Q_j innerhalb des ROIs einen Vektor auf und berechnet die dazugehörigen Pixelintensitäten. Jeder Vektor stellt dabei einen möglichen lane-candidate dar. Zu jedem P_i gibt es einen möglichen lane-candidate dessen Pixelintensität am größten ist. Die Pixelintensität dieses lane-candidates wird zusammen mit den X-Werten der dazugehörigen Start- und End-Punkte in einem drei-dimensionalen Array abgespeichert. Die Pixelintensität eines Vektors ergibt sich aus der Aufsummierung der Grauwerte aller Bildpunkte des Vektors.

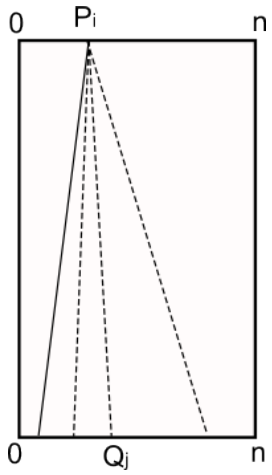

Abbildung 5.4.: Fahrspurermittlung: Funktionsweise der I-Operation

Nachdem für jeden möglichen lane-candidate aller Punkte P_i die Informationen gesammelt sind, wird mit nachfolgender Gleichung (5.2) zu jedem P_i ein Wert λ_i berechnet. Dieser Wert gibt Auskunft über die Distanz eines möglichen lane-candidates in Bezug auf den aktuellen. Zur Gewichtung der Größen werden dabei die im vorherigen Abschnitt berechneten Kantengewichte mit einbezogen.

$$\lambda_i = K_P * |P(C_i) - P(C_p)| + K_D * |D(C_i) - D(C_p)| + K_I * |I(C_p) - I(C_i)| \quad (5.2)$$

wobei:

- C_i: Wert des möglichen lane-candidate
- C_p: Wert des aktuellen lane-candidate
- K: Kantengewichte der Dimensionen

- P: X-Wert des Startpunkts des möglichen lane-candidates
- D: X-Wert des Endpunkts des möglichen lane-candidates
- I: Pixelintensität des möglichen lane-candidates

Sind alle Werte bestimmt, so wird das λ_i gesucht, welches am kleinsten ist. Dieses beschreibt dann den am nahe liegensten und somit den folgenden lane-candidate. Der ermittelte lane-candidate wird zur Visualisierung nach seiner Bestimmung zusammen mit dem aktuellen ROI in den Farben Rot bzw. Blau in das Ausgabefenster (750px $\times$ 400px; Zeigt das aufgenommenem Bild der Kamera) eingezeichnet.

5.4. Bestimmung des Lenkwinkels

Um den nötigen Lenkwinkel zu bestimmen wird davon ausgegangen, dass sich die Kamera mittig auf dem Fahrzeug befindet. Somit entspricht die vertikale Bildmitte der Fahrzeugmitte. Eine entsprechende Vertikale wird mit folgenden Koordinaten und der Farbe Grün ebenfalls in das Ausgabefenster eingezeichnet:

$$P_{1,x} : 375px, P_{1,y} : 0px$$
$$P_{2,x} : 375px, P_{2,y} : 399px$$

Um einen Lenkwinkel bestimmen zu können, der das Fahrzeug veranlasst einer Fahrspur nachzufahren sind vorerst einige Zwischenschritte notwendig. Diese werden im Folgenden aufgezeigt.

Ausmessen der PWM-Signale

Im ersten Schritt wurde mit Hilfe eines Oszilloskops ausgemessen für welchen Zeitraum das PWM-Signal eine positive Flanke haben muss, damit die maximalen und der neutrale Lenkwinkel angefahren wird. Daraus ergaben sich die Werte in Tabelle 5.2. Siehe dazu auch Abbildung 5.5 und 5.6.

Linker Anschlag	$801\mu s$
Zentriert	$1287\mu s$
Rechter Anschlag	$1801\mu s$

Tabelle 5.2.: Gemessene Einschaltzeiten bestimmter Lenkwinkel

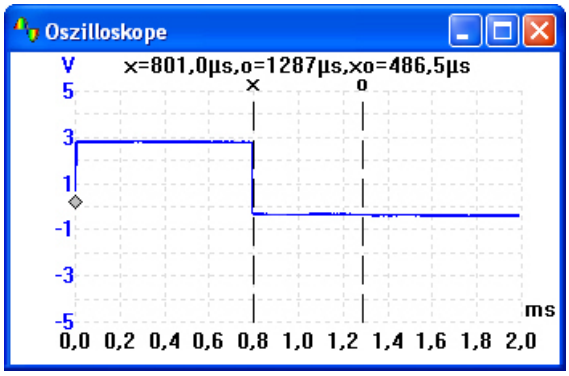

Abbildung 5.5.: Lenkwinkelbestimmung: Ausgabe in PicoScope - Linker Lenkanschlag

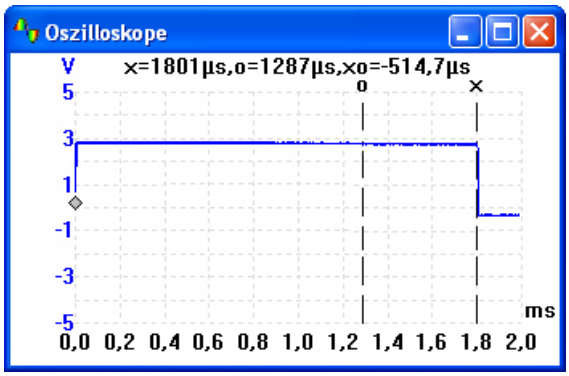

Abbildung 5.6.: Lenkwinkelbestimmung: Ausgabe in PicoScope - Rechter Lenkanschlag

Bei der Überprüfung der Werte stellt sich heraus, dass die Lenkung nicht ganz Symetrisch ist. Überprüft werden kann dies indem man mit nachfolgender Gleichung die Einschaltzeit für die neutrale Lenkstellung aus den Einschaltzeiten der beiden maximalen Lenkeinschlägen berechnet:

$$t_{ein,mid} = t_{ein,min} + \frac{t_{ein,max} - t_{ein,min}}{2} \tag{5.3}$$

Setzt man die gemessenen Werte in die Gleichung 5.3 ein so erhält man folgenden Wert für die Einschaltzeit in neutraler Lenkstellung:

$$t_{ein,mid} = 801\mu s + \frac{1801\mu s - 801\mu s}{2} = 1301\mu s \tag{5.4}$$

Zwischen der errechneten Einschaltzeit von 1301μs und der gemessenen tatsächlichen von 1287μs liegt eine Differenz von 14μs, die auf eine ungenau eingestellte Lenkung zurückzuführen ist. Aus diesem Grund wird der gemessene Wert als Wert für die neutrale Lenkstellung übernommen. Weiterhin werden die Einschaltzeiten von linkem und rechtem Lenkanschlag

so gewählt, das beide Richtungen mit einem identischen maximalen Lenkwinkel angesteuert werden können.

$$\begin{aligned} 1287\mu s - 801\mu s &= 486\mu s \\ 1801\mu s - 1287\mu s &= 514\mu s \end{aligned} \tag{5.5}$$

Zur Ansteuerung der maximalen Lenkwinkel wird das Minimum dieser beiden Differenzen verwendet.

Es ergeben sich für die weitere Verwendung die in nachfolgender Tabelle zusammengefassten Werte.

Linker Anschlag	*801μs*
Zentriert	*1287μs*
Rechter Anschlag	*1773μs*

Tabelle 5.3.: Verwendete Einschaltzeiten bestimmter Lenkwinkel

Linearisierung der Lenkung

Beim Betrachten der Lenkung machte es den Eindruck, dass sich der Lenkwinkel nicht linear zum Lenk-Servo verhält. Deshalb wurde die Lenkung des Fahrzeugs vermessen um den verlauf der Lenkung bestimmen zu können.

Dazu wurde das Fahrzeug auf einen ebenen Untergrund gestellt, seine exakte Position markiert und dann von diesem Startpunkt aus Kurven mit bestimmten Einschaltzeiten gefahren. Diese Endpunkte wurden dann ausgemessen, um aus ihnen den Lenkwinkel bestimmen zu können. Es ergaben sich folgende Werte:
Die berechneten Winkel wurden dann zusammen mit den zugehörigen Einschaltzeiten in ein Koordinatensystem eingetragen. Das Ergebnis dieser Messung schaute wie folgt aus:

y	38									
x	800	801	900	975	1000	1100	1200	1287	1300	1400
α	-40,54	-40,45	-34,38	-28,93	-23,85	-14,46	-5,41	0,00	2,86	9,71

y	38					
x	1500	1600	1625	1700	1773	
α	19,56	24,10	28,35	36,19	40,54	

Tabelle 5.4.: Gemessene Winkel bestimmter Einschaltzeiten

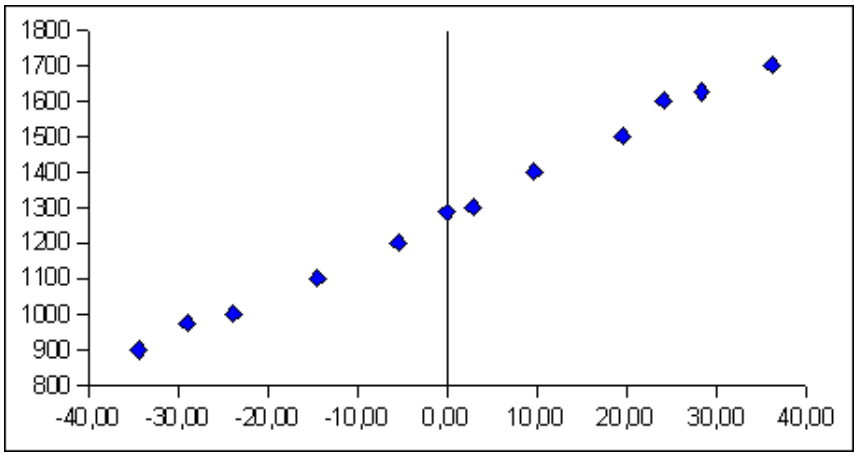

Abbildung 5.7.: Lenkwinkelbestimmung: Messung der Lenkwinkel

Überraschender Weise stellte sich dabei heraus, dass die Lenkung doch ein lineares Verhalten aufweist. Mit dieser Erkenntnis und dem Wissen über die in Tabelle 5.3 festgehaltenen Einschaltzeiten lässt sich eine lineare Funktion bestimmen, mit der sich aus einem gegebenen Lenkwinkel die benötigte Einschaltzeit berechnen lässt. Diese schaut wie folgt aus:

$$f(\alpha) = \frac{486}{40} * \alpha + 1287 = \frac{243}{20} * \alpha + 1287 \tag{5.6}$$

Berechnung des Lenkwinkels

In dieser Arbeit wird eine Methode aus Pareigis (2008). Dabei wird die Distanz der Fahrspur zur Fahrzeuglängsachse wird betrachtet. Es fließen die Distanz eines definierten Punktes zur Fahrzeuglängsachse (P-Anteil), sowie die im folgenden als „Drift“ bezeichnete Distanz mit in die Berechnung des Lenkwinkels ein (D-Anteil). Der Drift gibt Auskunft darüber, wieviel Pixel sich der betrachtete Punkt in Bezug auf das zuvor ausgewertete Bild in X-Richtung bewegt hat. Diese beiden Werte werden verschieden gewichtet und miteinander addiert. Verwendet wird hier der Startpunkt des lane-candidate (P_i). Als Gewichte werden vorerst die in Pareigis (2008) festgelegten Wert (P-Anteil: 0,25; D-Anteil: 1) übernommen.

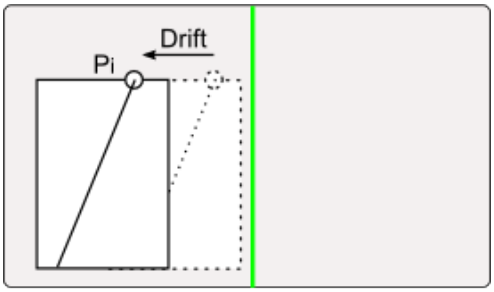

Abbildung 5.8.: Größen zur Lenkwinkelberechnung

Es ergibt sich aus der Summe der sogenannte *„leadValue“*. Der leadValue gibt den prozentualen Ausschlag der Lenkung an. Dieser wird auf die maximalen $\pm 40^\circ$ Lenkausschlag umgerechnet und er gibt den anzusteuernden Lenkwinkel in Grad. Dieser wird letzten Endes in die zuvor berechnete Funktion (vgl. Gl. 5.6) eingesetzt. Es ergibt sich die entsprechende Einschaltzeit für das PWM-Signal.

5.5. Dynamisierung des ROIs

Mit einem statischen ROI funktioniert die Fahrspurerkennung bereits hinreichend gut. Es bringt aber auch einen erheblichen Nachteil mit sich. Um nachfolgende Fahrstreifen erkennen zu können, die in einem größeren Winkel zum aktuellen Fahrstreifen liegen, muss das ROI sehr groß gehalten werden. Bedingt dadurch verschlechtet sich die Performanz und die Robustheit des Systems. Zum einen müssen mehr Bildpunkte als nötig abgearbeitet werden und zum anderen können Linien, die sich im Bildbereich befinden aber keine Fahrspur darstellen fälschlicher Weise als eine solche interpretiert werden. Um dies zu vermeiden benutzt die Anwendung ein dynamisches ROI.

Das ROI wird nach jedem analysierten Bild in seiner horrizontalen Position angepasst. Die Y-Koordinaten werden beibehalten, da eine Änderung derer nur eine Änderung der Sichtweite bewirkten würde. Zur Berechnung des neuen ROIs werden die Start- und End-Koordinaten des aktuellen lane-candidate zur Hilfe genommen.
Aus den X-Werten der Punkte P_i und Q_j wird das Mittel gebildet. Dieser Wert bildet dann die neue Mitte des ROIs. Als optimal hat sich während der Entwicklung eine Breite von 90px für das ROI. Damit ist gewährleistet, dass auch schnell ausbrechende Fahrspuren noch rechtzeitig erkannt werden.
Stößt das ROI an die linke oder rechte Grenze des Bildes, so bleibt es dort in seiner vollen Größe stehen bis sich der Fahrstreifen wieder in Richtung Fahrzeugmitte bewegt. Dies verhindert, dass das ROI zu einer Seite aus dem Bild hinaus läuft.

5.6. Ansteuerung der Lenkung

Ist die Einschaltzeit für einen neuen Lenkwinkel berechnet wird dieser in einer CAN-Nachricht verpackt und diese dann mittels eines USB-to-CAN-Converters auf dem von ihm geöffneten CAN-Bus abgelegt. Der Microcontroller ist ein Knoten auf dem CAN-Bus, welcher die gesendete Nachricht empfängt. Gesendet wird die Einschaltzeit in der Einheit μs. Die Anzahl der zu übertragenen Bytes pro CAN-Nachricht richtet sich dabei nach der Größe der maximalen Einschaltzeit.

Wert für maximale Einschaltzeit: 1773

$$2^{10} < 1773 < 2^{11}$$

Es ergibt sich also eine Mindestgröße der zu übertragenden Nutzdaten von 11 Bit pro CAN-Nachricht. Da in einer CAN-Nachricht die Größe der zu übertragenden Daten nur Byteweise angeben werden können ist es notwendig die nächst größere Bytezahl anzugeben. Somit

Potenz	Maximaler Dezimalwert
2^{10}	1024
2^{11}	2048

Tabelle 5.5.: Maximaler Dezimalwert einer Potenz

ergibt sich, dass pro gesendeter CAN-Nachricht 2 Bytes an Nutzdaten übertragen werden.

Für den Microcontroller wurde ein kleines Programm geschrieben, welches die vom PC gesendeten CAN-Nachrichten empfängt. Kommt eine neue Nachricht an, so wird die Einschaltzeit den Nutzdaten entnommen und als neuer Wert für das PWM-Signal übernommen. Das PWM-Signal wird über einen Pin, an den der Servomotor der Lenkung angeschlossen ist, nach außen geführt. Der Servomotor verarbeitet das Signal und steuert einen entsprechenden Lenkwinkel an.

6. Test der Fahrspurerkennung

Neben kleinen Verhaltenstests am Arbeitsplatz, die während der Entwicklung statt fanden wurde das System anschließend auf unterschiedliche Aspekte hin untersucht. Die einzelnen Tests und deren Ergebnisse werden in dem nachfolgenden Abschnitt aufgezeigt.

6.1. Bildhelligkeit

Testziel

Der erste Test wurde durchgeführt um zu ermitteln, wie sich die Fahrspurerkennung auf unterschiedliche Lichtverhältnisse auswirkt. Es soll getestet werden welche Mindest-Pixelintensitäten von Eingangsbild und lane-candidate voraussetzung für eine einwandfreie Erkennung der Fahrspur zu verwenden sind.

Testdurchführung

Für diesen Test wurde eine Messung im Labor und eine im Freien bei Sonneneinstrahlung durchgeführt. Als Lichtquelle bei der Messung im Labor diente die Deckenbeleuchtung. Die Belichtungszeit der Kamera wurde fest eingestellt. Während des Tests wurde der Blendenöffnungswert verändert und dabei die aufsummierte Pixelintensität des erkannten lane-candidate, die durchschnittliche Pixelintensität des Eingangsbilds sowie allgemein das Verhalten der Fahrspurerkennung betrachtet. Zusätzlich wurde die Fahrspur bei den verschiedenen Einstellungen abrupt aus dem Bild gezogen um erkennen zu können, ob das System der Fahrspur noch folgen kann. Als Parameter für die Messungen dienten folgende Werte:

Kamera	
Belichtungszeit	20.0ms
Winkel	-7°

Lichtquelle		
Art	Leuchtstoffröhre	Sonnenlicht
Einstrahlwinkel	Senkrecht	60°
Lichtstärke	38Watt	—
Abstand	3m	—

Tabelle 6.1.: Testparameter

Es wurden folgende Werte gemessen:

Deckenbeleuchtung					
Blendenöffnungswert	1.5	2	2.8	4	5.6
Summierte Pixelintensität (lane-candidate)	16620	16242	8757	4683	1919
Gemittelte Pixelintensität (Eingangsbild)	47	46	38	34	32
Blendenöffnungswert	8	11	16	Zu	
Summierte Pixelintensität (lane-candidate)	874	Nicht messbar		754	
Gemittelte Pixelintensität (Eingangsbild)	31	30	29	28	

Sonneneinstrahlung					
Blendenöffnungswert	1.5	2	2.8	4	5.6
Summierte Pixelintensität (lane-candidate)	Nicht messbar				
Gemittelte Pixelintensität (Eingangsbild)	255	254	253	240	200
Blendenöffnungswert	8	11	16	Zu	
Summierte Pixelintensität (lane-candidate)	Nicht messbar	18873	16922	378	
Gemittelte Pixelintensität (Eingangsbild)	158	111	45	30	

Tabelle 6.2.: Messergebnis

Testergebnis

Aus dem durchegeführten Test lässt sich erkennen, dass eine gewisse Mindest-Pixelintensität im Bild vorhanden sein muss damit die Fahrspur korrekt erkannt werden kann. Während der durchführung wurden sowohl Eingangsbild, als auch gefiltertes Bild genauestens beobachtet. Es war zu erkennen, dass das System der aus dem Bild gezogenen Fahrspur nur dann korrekt folgen konnte, wenn die aufsummierte Pixelintensität des lane-candidate einen Wert oberhalb von 4500 hatte. Lag die Intensität niedriger, so konnte der angewandte Sobel-Operator die Kante der Fahrspur nicht mehr verstärken (vgl. Abb. 6.1). Aus diesem Grund verschwand die Fahrspur aus dem gefilterten und betrachteten Bildbereich, was zu Folge hatte, dass das ROI zum rechten Bildrand hin abdriftete. Im Grenzbereich dieses Wertes konnte das System der Fahrspur nur noch mit abnehmender Geschwindigkeit folgen.
Wurde das System einer direkten Sonneneinstrahlung ausgesetzt, so stieg die durchschnittliche Helligkeit des Eingangsbildes rapide an. Bereits bei großtem Blendenwert lag sie auf dem Niveau der beiden Kleinsten bei der Messung im Labor. Dies hatte zur Folge, dass die Fahrspur nur bei sehr hohen Blendenwerten erkannt werden konnte. Grund dafür ist, dass das sehr helle Licht für einen sehr geringen Kontrast zwischen Boden und Fahrspur sorgt. Somit konnte der Sobel-Operator keine Kante erkennen und verstärken. Dies führte wie zuvor zum Abdriften des ROIs.

Abbildung 6.1.: Unterschied zwischen geringer und normaler Intensität

Des Weiteren lässt sich aus den Werten mit geschlossener Blende erkennen, dass das Bild ein bestimmtes Grundrauschen enthält. Dieses Rauschen hat sich jedoch nicht negativ auf die Spurerkennung ausgewirkt.

Testfazit

Es ist stets darauf zu achten, dass der lane-candidate stets eine gewisse Pixelintensität besitzt. Die Tests ergaben eine ideale Intensität von 7500 bis 15000. Liegt der Wert außerhalb dieser Grenzen wäre über eine Anpassung der Belichtungszeit eine Korrektur vorzunehmen. Zur Berechnung der idealen Belichtungszeit wäre die durchschnittliche Helligkeit des vollständigen Bereichs, auf dem sich das ROI bewegen kann, zu betrachten. Für die optimale durchschnittliche Helligkeit bezogen auf das gesamte Eingangsbild ergab sich ein Wert von 45 bis 55.

	Minimum	Maximum
Durchschnittliche Bildhelligkeit	45	55
Summierte Pixelintensiät	7500	15000

Tabelle 6.3.: Ideale Helligkeitswerte

6.2. Beeinflussung der Fahrspur

Testziel

Dieser Test soll aufschluss über die Robustheit des verwendeten Algorithmusses in Zusammenhang mit den verwendeten Kantengewichten geben. Es soll getestet werden, inwiefern sich die Fahrspurerkennung durch andere mögliche Fahrspuren beeinflussen lässt.

Testdurchführung

Der erste Teil des Tests wurde auf dem Flur vor dem Labor durchgeführt. Dieser Ort wurde ausgewählt, da sich der Bodenbelag, sowie die Deckenbeleuchtung hervorragend eignen um einfallendes Licht auf die Fahrspur zu simulieren. Es wurde eine Fahrspur so ausgelegt, dass sie zwei Mal von der Reflektion der Deckenbeleuchtung auf dem Fußboden gekreuzt wird.

Abbildung 6.2.: Das Testszenario

Es wurden beim Test folgende Parameter verwendet:

Fahrspur	
Kurvenradius 1	°
Kurvenradius 2	°
Kurvenradius 3	°
Kurvenradius 4	°
Gesamtlänge	1020cm
Fahrzeug	
Geschwindigkeit	Schrittgeschwindigkeit
Kamera	
Belichtungszeit	20ms
Blendenwert	
Winkel	-7°
Sonstiges	
Bild-Helligkeit (Mittlerer Grauwert)	
Boden-Reflektion	10%

Tabelle 6.4.: Testparameter

Der Grad der Reflektion des Bodens wurde mit Hilfe eines Belichtungsmessers aus der Fotografie gemessen. Das Messgerät wurde direkt auf die Deckenbeleuchtung sowie die Reflektion auf dem Boden gerichtet. Bei einem eingestellten Blendenwert von 5.6 sowie einem ISO-Wert von 400 ergaben sich folgende Belichtungszeiten:

Deckenbeleuchtung	1/125s
Reflektion	1/13s

Tabelle 6.5.: Belichtungszeiten bei unterschiedlichen Lichtverhältnissen

Nachdem beide gemessenen Werte ins Verhältnis gesetzt wurden ergab sich ein Grad der Reflektion von 10%.

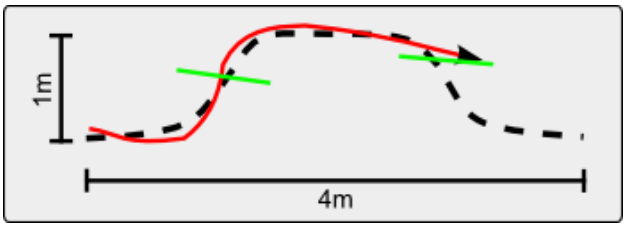

Abbildung 6.3.: Skizze des zurückgelegten Wegs

Wie in der Skizze der gefahrenen Strecke zu erkennen kam es auf dem ersten Teil der Strecke zu keinen Problemen bei der Erkennung der Fahrspur. Die erste Kreuzung der Lichtreflektion (grün eingezeichnet) mit der Fahrspur verlief in einem so ungünstigen Winkel, dass sie deren Kante nicht vom verwendeten Filter nicht verstärkt werden konnte und somit auch nicht vom System wahrgenommen werden. Anders sah dies jedoch bei der zweiten kritischen Stelle aus. Ein Blick aus der Sicht des Fahrzeugs verdeutlicht dieses:

Abbildung 6.4.: Probem durch einfallendes Licht

Die Reflektion lag dieses Mal in einem wesentlich besseren Winkel zur aktuellen Fahrspur. Er betrug hier 15°. Weiterhin lag die eigentliche Folgefahrspur in einem wesentlich ungünstigeren Winkel von 35°.

Haben die Reflektion und der eigentliche Fahrstreifen eine ähnliche Pixelintensität, so wird der lane-candidate ausgewählt, dessen Winkelabweichung in Bezug auf den aktuellen lane-candidate am geringsten ist. Da dieses in dieser Situation der Fall war entschied der Algorithmus die Reflektion weiter zu verfolgen.

Der zweite Teil des Tests wurde im Freien durchgeführt. Es wurde eine Gerade ausgelegt. Auf diese Gerade wurde auf halber Strecke ein „falscher“ Fahrstreifen mit variierendem Winkel gelegt. Die Strecke wurde mehrfach abgefahren um zu ermitteln, wann die Fahrspurerkennung dem „falschen“ Fahrstreifen folgt.

Auf den nachfolgenden Bildern aus der Perspektive ist das gewählte Szenario, sowie die ausgewählte Fahrspur zu erkennen:

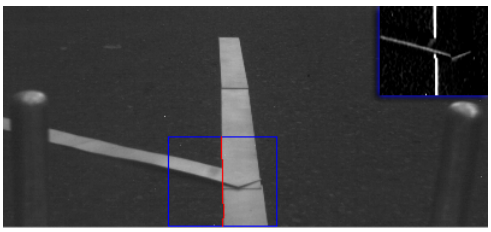

Abbildung 6.5.: Winkel: 45°

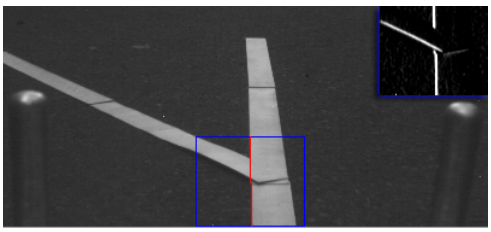

Abbildung 6.6.: Winkel: 30°

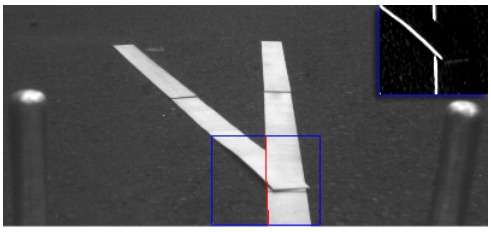

Abbildung 6.7.: Winkel: 20°

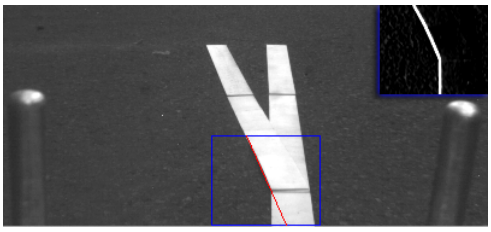

Abbildung 6.8.: Winkel: 10°

Dadurch, dass der TFALDA die Richtung der Fahrspur mit in seine Berechnungen einfließen lässt wurden die „falschen“ Fahrstreifen im Winkel größer als 10° nicht beachtet. Lediglich die Fahrspur im Winkel von 10° wurde als plausibel beurteilt und weiter verfolgt. Den Grund dafür kann man im betrachteten Bereich (Bildausschnitt oben rechts) erkennen. Durch die Breite der Fahrstreifen wird die Kante der eigentliche Fahrspur verdeckt, sodass diese nicht durch den Sobel-Operator verstärkt werden kann. Somit bleibt sie verborgen und der Algorithmus folgt der „falschen“ Spur.

Abschließend wurde das folgende Szenario überprüft:

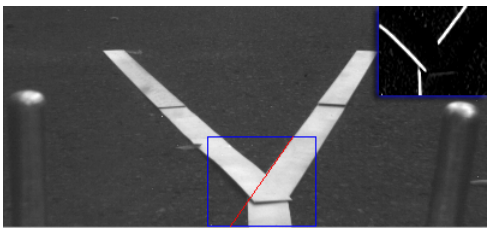

Abbildung 6.9.: Gabelung der Fahrspur

In dieser Situation ist keine Prognose zu treffen, wie sich das Fahrzeug verhalten wird. Es kommt dabei ganz auf die Ausrichtung des Fahrzeugs zur Fahrspur an. Je nach dem, von welcher Folge-Fahrspur mehr Bildpunkte im ROI liegen, wählt der Algorithmus den Nachfolger aus.

Testergebnis

Der Test hat gezeigt, dass sich die Fahrspurerkennung bei nicht plausiblen Störungen sehr robust verhält. Wenn jedoch eine Störung im Bild lag, deren verstärkte Kante als eine plausible Fahrspur erkannt wurde, so wurde diese statt der eigentlichen Fahrspur weiter verfolgt. Weitere Plausibilitätsprüfungen wären an dieser Stelle zu überlegen um einen höheren Grad der Robustheit zu erreichen.

6.3. Systemauslastung

Da auch andere Anwengungen parallel auf der benutzten Hardware zum Einsatz kommen ist es wichtig zu wissen, wie sehr die Fahrspurerkennung die Systeme auslastet.

Computer

Gerade bei dem eingesetzten PC ist es von Interesse, da auf diesem später die gesamte Architektur aufgesetzt werden soll. Es kommen dann weitere bisher entwickelte Systeme zum Einsatz die ebenfalls eine gewisse Last erzeugen. Daher ist darauf zu achten das System nicht mit einzelnen Systemen zu überlasten.

Um dies zu testen wurde die erzeugte Prozessorauslastung durch die Fahrspurerkennung über einen Zeitraum von 30sek. aufgezeichnet. Dabei wurde einmal mit eingeschalteter und einmal mit ausgeschalteter Visualisierung gemessen. Es ergab sich folgendes Ergebnis: Wie bereits erwartet erzeugt die Grafikverarbeitung eine vergleichsweise hohe Auslastung. Bei eingeschalteter Visualisierung lag die Auslastung der CPU bei durchschnittlich 22%. Durch das Ausschalten der Visualisierung konnte ein Gewinn von 7% auf 16% erreicht werden.

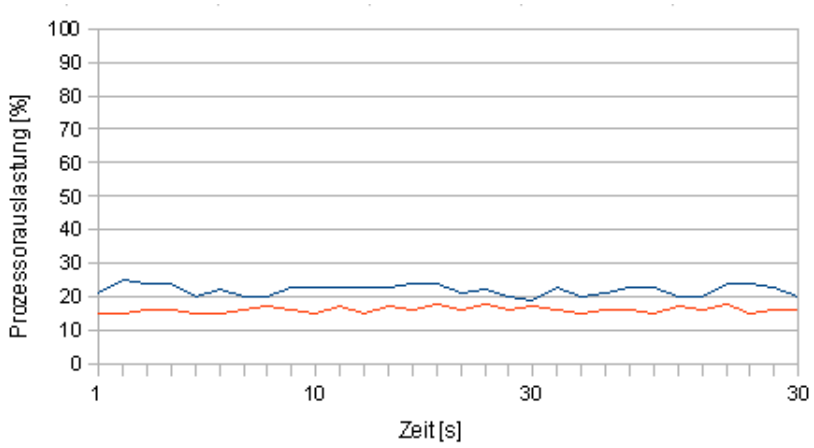

Abbildung 6.10.: Auslastung der CPU

Microcontroller

Bei den Microcontrollern hat man zwar theoretisch die Möglichkeit (bedingt durch den USB) bis zu 127 Geräte anzuschließen. Da diese jedoch teuer sind und der Platz an Bord des Fahrzeugs beschränkt ist, wird auch hier versucht möglichst viele Anwendungen auf einem Controller unterzubringen. Aus diesem Grund wird auch hier die Auslastung betrachtet.

Über einen Pin auf dem I/O-Board wird ein Signal ausgegeben. Dieses Signal wird mit einem Oszilloskop ausgemessen und dessen Flankendauer bestimmt. Über den eingestellten Prozessortakt von 48MHz lässt sich dann dessen Auslastung berechnen. Es wurden folgende durchschnittliche Bearbeitungszeiten des Microcontrollers gemessen:

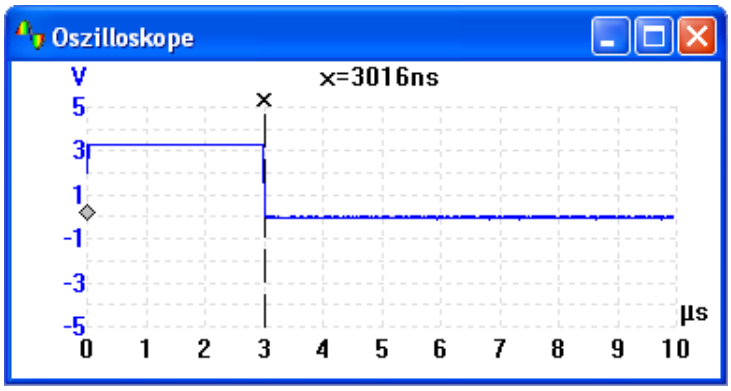

Abbildung 6.11.: Auslastung des Microcontrollers im Ruhezustand

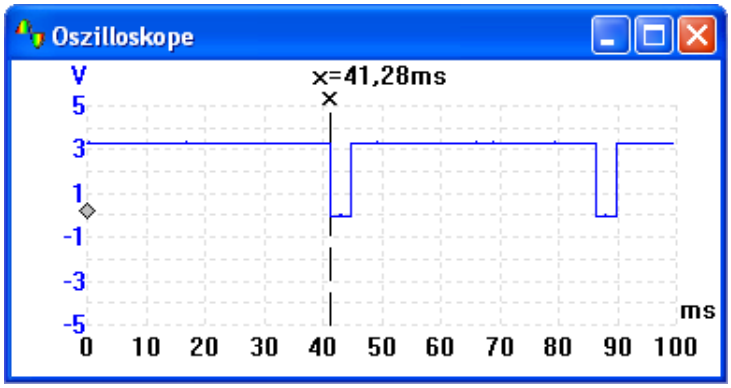

Abbildung 6.12.: Auslastung des Microcontrollers unter Last

Da der Prozessortakt auskunft darüber gibt, wieviele Rechenvorgänge die CPU pro sekunde bearbeiten kann, besteht die Möglichkeit über diesen Wert die Auslastung der CPU zu berechnen. Es ergibt sich folgendes:

$$48\text{MHz} \mathrel{\widehat{=}} 48 \text{ Mio. Rechenvorgänge pro s}$$

$$\text{Dauer eines vollständigen Rechenzyklusses: } 41,28ms$$

$$\text{Anzahl Rechenvorgänge pro Zyklus: } 48 * 10^6 * \tfrac{1}{s} * 41,28 * 10^{-3}s = 1981440$$

$$\text{Auslastung des Prozessors: } \tfrac{1981440*100}{48*10^6} = 4{,}128\%$$

6.4. Geradeausfahrt

Testziel

Bei diesem Test sollte die implementierte Lenkwinkelberechnung im Zusammenspiel mit der Fahrspurerkennung getestet werden. Ziel war es herauszufinden wie gut das Fahrzeug einer geraden Fahrspur folgen kann und ob sich durch das Verändern der aus Pareigis (2008) übernommenen Gewichte eine Verbesserung erzielen lässt.

Testdurchführung

Zur Durchführung des Tests wurde eine Gerade Fahrspur der Länge 5m ausgelegt. Das Fahrzeug wurde an den Anfang der Linie gestellt. Dann wurde die Fahrspurerkennung gestartet und die Strecke mehrfach abgefahren. Bei jedem Durchgang wurden die Gewichte etwas variiert.Während des Test wurde das Fahrzeug mit einer Geschwindigkeit von bis zu 10km/h bewegt.

Nachfolgend soll anhand der Skizzen gezeigt werden, welchen Weg das Fahrzeug bei welchen Gewichten zurückgelegt hat.

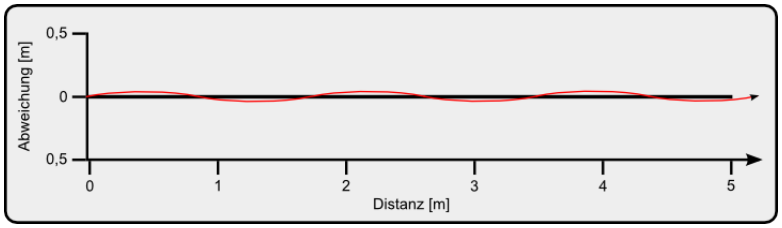

Abbildung 6.13.: P-Anteil: 0,25; D-Anteil: 1,00

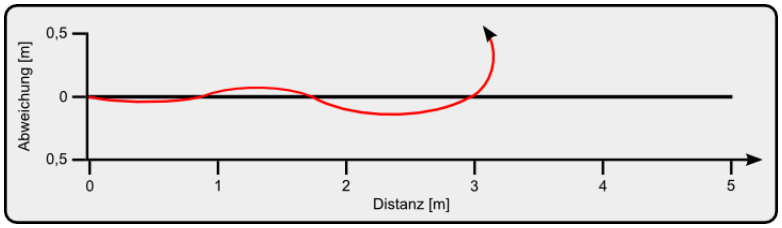

Abbildung 6.14.: P-Anteil: 0,15; D-Anteil: 1,00

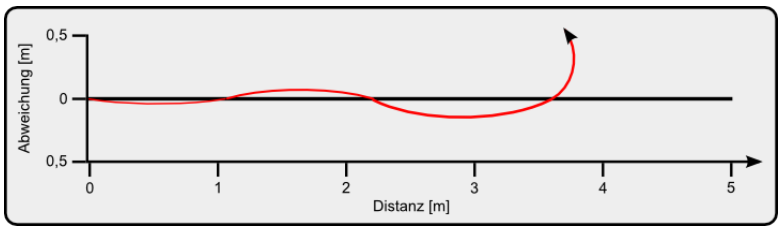

Abbildung 6.15.: P-Anteil: 0,50; D-Anteil: 1,00

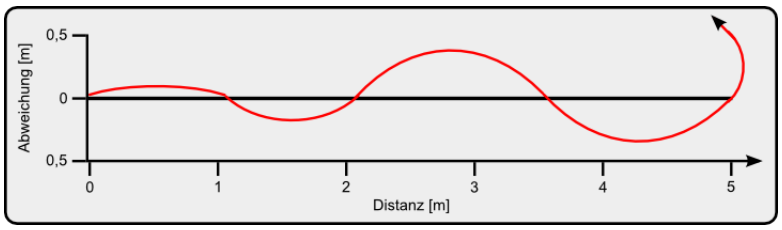

Abbildung 6.16.: P-Anteil: 0,25; D-Anteil: 0,50

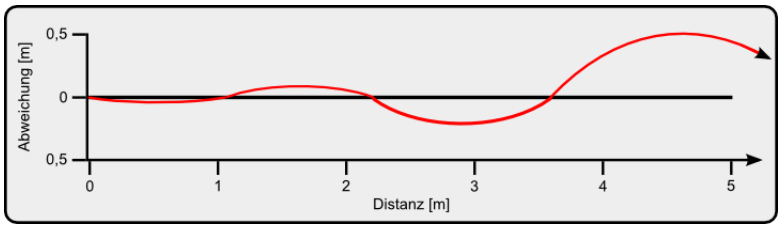

Abbildung 6.17.: P-Anteil: 0,25; D-Anteil: 1,50

Testergebnis

Es stellte sich bei diesem Test heraus, dass die übernommenen Gewichte das beste Ergebnis erzielen (vgl. Abb. 6.13). Bei Veränderung des P-Anteils schaukelte sich das Fahrzeug leicht auf und verlor dann die Fahrspur (vgl. Abb. 6.14 und 6.15). Wurde der D-Anteil verändert, so schaukelte sich das Fahrzeug gegen Ende der ausgelegten Strecke zunehmend stärker auf (vgl. Abb. 6.16 und 6.17). Zwar konnte das Fahrzeug der Fahrspur mit den übernommenen Gewichten gut und mit nur sehr geringen Schlenkern folgen, doch galt dies nur bis zu einer Geschwindigkeit von ca. 10km/h. Lag die Geschwindigkeit höher, so schaukelte sich das Fahrzeug ebenfalls auf und verlor die Fahrspur. Es macht den Eindruck, dass für unterschiedliche Geschwindigkeiten verschiedene Gewichte gewählt werden müssen.

6.5. Kurvenfahrt

Testziel

Dieser Test knüpfte an den Test der Geradeausfahrt an und sollte zeigen, wie sich das Fahrzeug mit den zuvor als ideal bestimmten Gewichten während einer Kurvenfahrt verhält.

Testdurchführung

Diesmal wurde zur Durchführung des Tests eine S-Kurve ausgelegt. Die beiden Kurvenradien betrugen dabei 2m und 4m.

Die nachfolgende Skizze zeigt wie sich das Fahrzeug während des Tests verhalten hat:

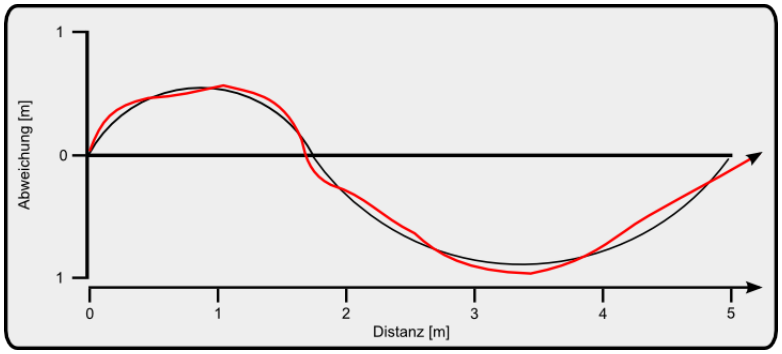

Abbildung 6.18.: P-Anteil: 0,25; D-Anteil: 1,00

Testergebnis

Der Skizze lässt sich entnehmen, dass die übernommenen Gewichte wieder ein hervorragendes Ergebnis erzielen konnten. Dies galt jedoch nur, wenn die Strecke mit Schrittgeschwindigkeit befahren wurde. Ein Anheben der Geschwindigkeit führte ebenso wie eine Änderung der Gewichte zum Verlust der Fahrspur. Auf dem empirischen Weg konnte keine Verbesserung des Ergebnisses erreicht werden.

6.6. Fahrzeugerschütterungen

Testziel

Dieser Test soll zeigen, wie sich das System bei Erschütterungen und unterschiedlichen Fahrbahnbelägen verhält.

Testdurchführung

Für den Test wurde ein Worst-Case-Szenario gewählt. Es wurde auf Kopfsteinpflaster eine gerade Fahrspur ausgelegt, welcher das Fahrzeug folgen soll. Zum großteil bedingt durch die Pflastersteine, aber auch zusätzlich durch die Vibrationen des Motors und das grobstollige Profil der Reifen beeinflusst ändert sich kontinuierlich der Blickwinkel auf die Fahrspur. Während der Fahrt glich das System die Lenkung stets minimal an, sodass der Fahrspur

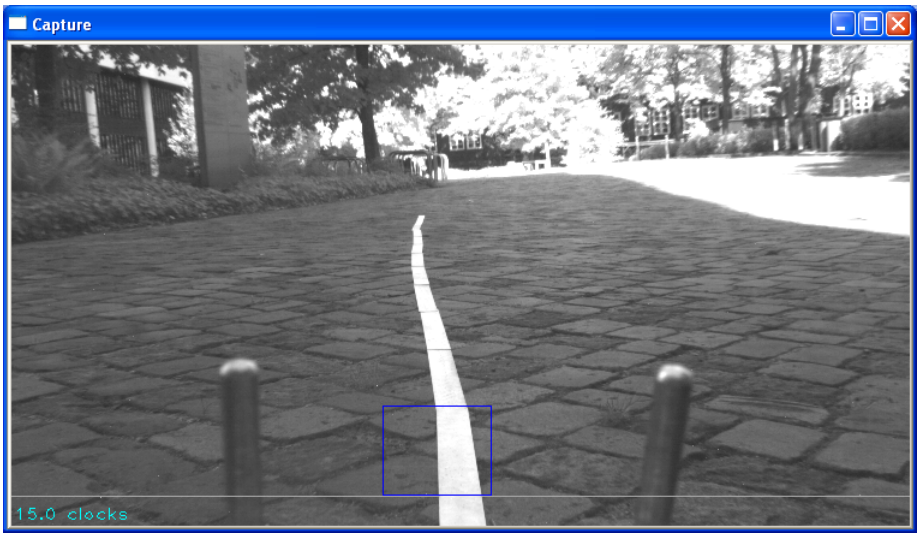

Abbildung 6.19.: Testszenario aus Fahrzeugperspektive

über den gesamten Verlauf der Teststrecke erfolgreich gefolgt werden konnte. Dies verdeutlicht folgende Grafik über die zurückgelegte Strecke des Fahrzeugs:

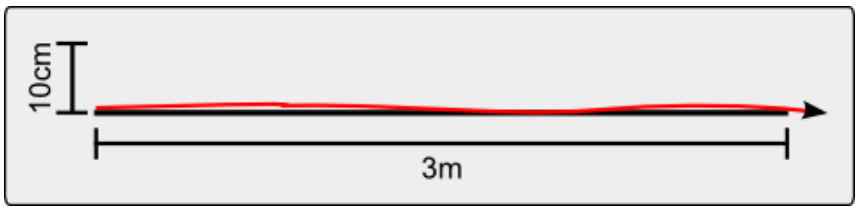

Abbildung 6.20.: Zurückgelegter Weg beim Test

Testergebnis

Der Test zeigte, dass sich das Fahrzeug trotz widrigster Umstände entlang der Fahrspur bewegen konnte. Durch die hohe Bearbeitungsgeschwindigkeit des Systems wurde ein Ausbrechen der Fahrspur rechtzeitig erkannt, sodass das ein rechtzeitiges Reagieren möglich war. Es lässt sich ableiten, dass sich das Fahrzeug problemlos auf den unterschiedlichsten Fahrbahnbelägen bewegen kann.

7. Zusammenfassung und Verbesserungsmöglichkeiten

7.1. Zusammenfassung

In dieser Arbeit sollte im Rahmen des „intelliTruck“-Projekts der HAW-Hamburg ein kamergestütztes Fahrspurerkennungssystem entwickelt werden mit dem ein Modellfahrzeug autonom einer ausgelegten Fahrspur folgen kann. Es galt die Erkenntnisse aus vorherigen Arbeiten für diese Arbeit miteinzubeziehen.

Aufbauend auf den Arbeiten von Dennis Berger und Alexander Kant wurde ein System entwickelt, welches den TFALD-Algorithmus zur Erkennung der Fahrspur nutzt. Dieser hat sich bedingt durch das Einbeziehen verschieden gewichteter Informationen über die Fahrspur als sehr robust und hervorragend geeignet bewiesen.

Das Ergebnis dieser Arbeit ist ein System, welches zuverlässig und unter den verschiedensten Bedingungen eine Fahrspur in einem Kamerabild erfassen und dieser folgen kann. Es stellte sich bei Tests heraus, dass sich der verwendete Algorithmus nur von anderen Einflüssen wie Reflektion oder anderen im Bild enthaltenen Kanten ablenken ließ, wenn diese seine sehr hohe Ähnlichkeit in Bezug auf die drei betrachteten Dimensionen einem aktuellen lane-candidate darstellten. Weiterhin hat sich die für den Microcontroller entwickelte Software als sehr robust gezeigt. Während den Testläufen kam es nur bedingt durch Wackelkontakten zu kurzen Aussetzern.

Zwar funktionieren diese beiden Komponenten zum jetzigen Zeitpunkt bereits sehr gut, jedoch ist das autonome Fahren noch von einer dritten Komponente abhängig. Dabei handelt es sich um die Berechnung des Lenkwinkels. Dieses funktioniert zum jetzigen Zeitpunkt nur bei Geschwindigkeiten bis zu 10km/h. Im verhältnis zur theoretisch Möglichen Geschwindigkeit von 65km/h ist dies sehr gering. Bei höheren Geschwindigkeiten verliert das Fahrzeug die Fahrspur. Zurückzuführen ist dieser Umstand auf die Gewichte mit denen bestimmt wird, wie sehr sich die Informationen über den ausgewähltenn lane-candidate auf die Berechnung des Lenkwinkels auswirken. Zwar wurden wie im Testkapitel nachzulesen verschiedene Gewichte ausprobiert, jedoch konnte bis zum jetzigen Zeitpunkt kein besseres Ergebnis erzielt werden.

Da die Erkennung der Fahrspur, sowie die Ansteuerung der Lenkung bereits sehr gut funktionieren, bilder das in dieser Arbeit umgesetzte Fahrspurerkennungssystem eine gute Basis für Arbeiten, die auf diesem Thema aufbauen sollen. Zum Abschluss sollen im nachfolgenden Abschnitt einige Ansätze mit auf den Weg gegeben die bei einer Weiterentwicklung des Systems betrachtet werden solten.

7.2. Verbesserungsmöglichkeiten

Während der Implementierung der Fahrspurerkennung, sowie bei den Tests sind einige Dinge aufgefallen, die bei der Verbesserung des in dieser Arbeit implementierten Systems betrachtet werden sollten. Diese Gedanken werden im Folgenden kurz zusammengefasst.

7.2.1. Anpassen der Kantengewichte

In Kapitel 5.2 wurde angenommen, dass die im IEEE-Paper genannten Kantengewichte optimal seien. Dies stimmt insofern, als dass die Gewichte optimal für die von den Autoren getesteten Szenarien sind. Unklar ist, ob die Gewichte ebenfalls im Einsatzgebiet des „intelliTrucks“ als optimal zu bezeichnen sind. Deshalb wäre es sinnvoll eine Software zu schreiben, die die Fahrspurerkennung auf eine Sequenz von Bildern einer Testfahrt mit variierenden Kantengewichten anwenden kann. Aufzuzeichnen wäre dabei, wieviele Fahrspuren erkannt wurden und wieviele nicht. Daraus ließe sich ableiten, welche Kantengewichte für das Einsatzgebiet des „intelliTrucks“ als ideal zu bezeichnen sind. Es empfiehlt sich dazu ein Blick in Berger (2008).

7.2.2. Dynamische Belichtungszeit

Aus dem Test in Kapitel 6.1 geht hervor, dass es sinnvoll wäre, die Belichtungszeit fortlaufend anzupassen. Somit bestünde die Möglichkeit trotz welchselnder Lichtverhältnisse stets eine als ideal ermittelte durchschnittliche Bildhelligkeit zu erreichen und somit für ein für die Kantenverstärkung ausreichendes Kontrastverhältnis zwischen Fahrstreifen und Untergrund zu sorgen.

7.2.3. Verwendung zweier ROIs

Weiterhin wäre zu testen, ob eine Fahrspurerkennung mit der verwendung von zwei aneinandergeknüpften und untereinander liegenden ROIs Vorteile mit sich bringt. Denkbar wäre, dass sich das Fahrzeug bei höheren Geschwindigkeiten besser kontrollieren lässt, da das System weiter vorausschauen kann und somit früher Kenntnis über den Fortverlauf der Strecke hat.

Literaturverzeichnis

[Atmel 2006] ATMEL: *AT90CAN128 Datasheet*. 2006. – URL http://www.atmel.com/dyn/resources/prod_documents/doc7522.pdf

[Berger 2008] BERGER, Dennis: *Verbesserung des TFALDA-Algorithmus zur Fahrspurerkennung*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2008

[Bosch 2008] BOSCH: *Fahrsicherheit*. 2008. – URL http://rb-k.bosch.de/de/sicherheitkomfort/fahrsicherheit/index.html

[Bräunl 1995] BRÄUNL, Thomas: *Parallele Bildverarbeitung*. Addison-Wesley, 1995. – ISBN 3-893-19951-9

[Citroën 2008] CITROËN, Automobiles: *Lane Departure Warning System (LDWS)*. 2008. – URL http://www.citroen.com/CWW/en-US/TECHNOLOGIES/SECURITY/AFIL/

[DARPA 2008] DARPA: *DARPA Urban Challenge*. 2008. – URL http://www.darpa.mil/GRANDCHALLENGE/

[Dössel 2000] DÖSSEL, O.: *Bildgebende Verfahren in der Medizin - Von der Technik zur medizinischen Anwendung*. Springer-Verlag GmbH, 2000. – ISBN 3-540-66014-3

[eCos 2008] ECOS: *eCos - Realtime OS for embedded applications*. 2008. – URL http://ecos.sourceware.org/

[FAUST 2008] FAUST: *FAUST - Fahrerassistenz- und Autonome Systeme*. 2008. – URL http://www.informatik.haw-hamburg.de/faust.html

[Glatz 2008] GLATZ, Nicolai: *Möglichkeiten eines Radartransceivers zur Hinderniserkennung an einem autonomen Fahrzeug*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2008. – URL http://www.informatik.haw-hamburg.de/fileadmin/faust_upload/Arbeiten/2008Ba_-_Nicolai_Glatz_Moeglichkeiten_eines_Radartransceivers_zur_Hinderniserkennung_an_einem_autonomen_Fahrzeug.pdf

[IDS-Imaging 2008] IDS-IMAGING: *uEye LE Datenblatt*. 2008. – URL ftp://ftp2.imaging.de/websites/documents/catalogue_us/cameras/IDS/en_US_IDS_uEye_1220.pdf

[IXXAT 2008] IXXAT: *USB-to-CAN compact*. 2008. – URL http://www.ixxat.de/usb-to-can-compact-interface_de.html

[Kant 2007] KANT, Alexander: *Bildverarbeitungsmodul zur Fahrspurerkennung für ein autonomes Fahrzeug*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2007. – URL http://www.informatik.haw-hamburg.de/fileadmin/faust_upload/Arbeiten/2007Ba_-_Alexander_Kant_-_Bildverarbeitungsmodul_zur_Fahrspurerkennung_fuer_ein_autonomes_Fahrzeug.pdf

[Kirchner u. a. 2005] KIRCHNER, Dr.-Ing. A. ; KRÜGER, Prof. Dr.-Ing. K. ; MILDNER, Oberstleutnant Dr.-Ing. F. ; SCHMIDT, Dr.-Ing. R.: Laserscanner für Fahrerassistenzsysteme. In: *Automobiltechnische Zeitschrift* (2005), Nr. 9

[Korda 2005] KORDA, Martin: *Städtebau - Technische Grundlagen*. 5. Auflage. Vieweg+Teubner Verlag, 2005. – ISBN 3-519-45001-1

[Martin 2007] MARTIN, Trevor: *The Insider's Guide To The NXP LPC2300/2400 Based Microcontrollers*. 2007. – URL http://www.informatik.haw-hamburg.de/faust.html

[NXP 2008] NXP: *LPC2468 Datasheet*. 2008. – URL http://www.standardics.nxp.com/products/lpc2000/datasheet/lpc2468.pdf

[Pareigis 2008] PAREIGIS, Stephan: *Methoden zur Regelung der Lenkung und Geschwindigkeit eines autonomen Modellfahrzeugs basierend auf dem Bildverarbeitungsalgorithmus TFALDA - Internes Arbeitspapier, Department Informatik, HAW Hamburg*. 2008

[Photonfocus 2008] PHOTONFOCUS: *MV-D750E Serie*. 2008. – URL http://www.photonfocus.com/upload/flyers/flyer_A4_MV-D750E_de_1.0_n4.pdf

[Volkswagen 2008] VOLKSWAGEN: *Lane Assist*. 2008. – URL http://www.volkswagen.de/vwcms_publish/vwcms/master_public/virtualmaster/de3/unternehmen/Innovation/fahrassistenzsysteme/lane_assist.html

[Wallentowitz 2006] WALLENTOWITZ, Reif: *Handbuch Kraftfahrzeugelektronik - Grundlagen, Komponenten, Systeme, Anwendungen*. Vieweg Verlag GmbH, 2006. – ISBN 3-528-03971-X

[Yim und Oh 2003] YIM, Young U. ; OH, Se-Young: Three-feature based automatic lane detection algorithm (TFALDA) for autonomous driving. In: *IEEE Transactions on Intelligent Transportation Systems* 4 (2003), Nr. 4, S. 219–225

[Zeppelin 2006] ZEPPELIN, MVS: Baustellen- & Verkehrssicherung. In: *MVS Magazin* (2006), Nr. 1

A. Versuchsaufbau

Zur Inbetriebnahme der Fahrspurerkennung benötigt es im Wesentlichen nachfolgende Komponenten:

- PC
- NXP LPC2468 Microcontroller
- Photonfocus MV-D750E-20-U2 USB-Kamera
- IXXAT USB-to-CAN-Compact CAN-Converter
- Spannungsversorgung (+12.0V & +6.0V)

In der Abbildung A.1 ist das auf dem Labortisch aufgebaute Fahrspurerkennungssystem zu erkennen.

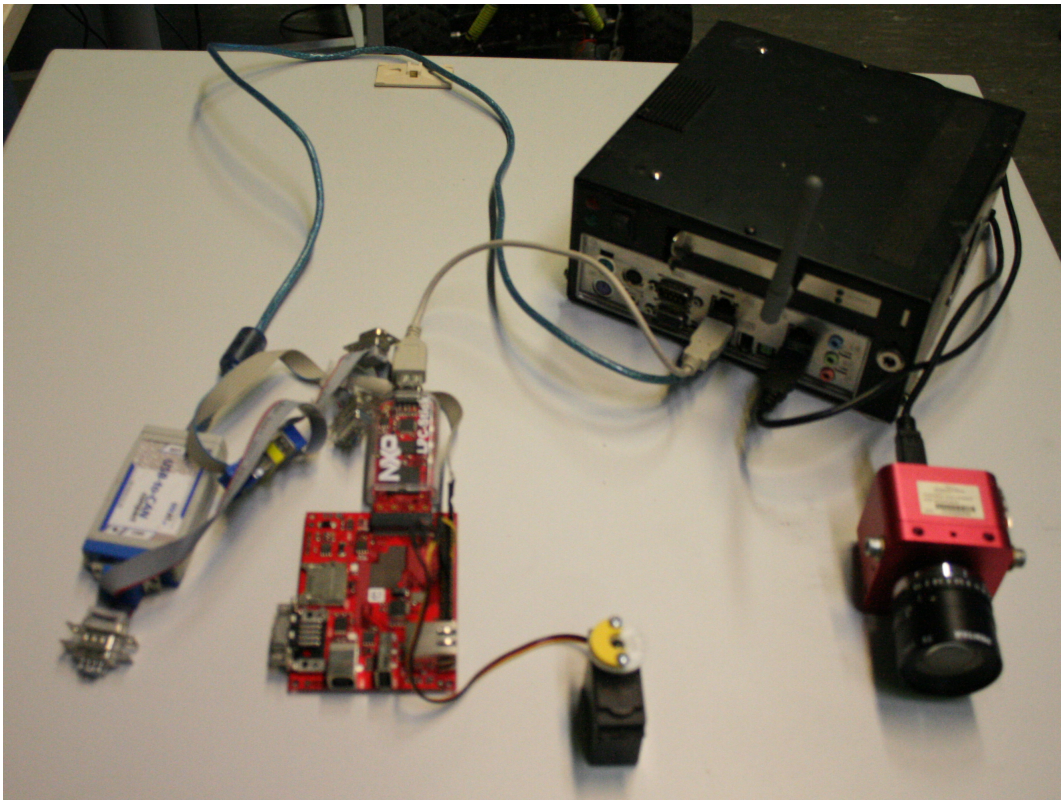

Abbildung A.1.: Versuchsaufbau auf Labortisch

B. Pin-Listing

Im nachfolgenden eine Auflistung, wie das IO-Board des Microcontrollers für den Betrieb verkabelt werden muss:

Pin	Beschreibung	Angeschlossen an		Pin	Beschreibung	Angeschlossen an
1	GND			2	CAN BM	CAN Low
3	GND			4	CAN BP	CAN High
5	+9V			6	P2.11	
7	+9V			8	P2.12	
9	+5V			10	P2.13	
11	+5V			12	P2.9	
13	+3.3V			14	P2.8	
15	+3.3V			16	P2.5	
17	GND	Masse Servo		18	P2.4	
19	GND			20	P2.3	
21	P0.26			22	P2.2	
23	P0.25			24	P2.1	
25	P0.24			26	P2.0	PWM-Signal Servo
27	P0.23			28	P0.5	
29	P0.20			30	P0.4	
31	P0.19			32	P0.22	
33	P0.18			34	P0.21	
35	P0.17			36	P1.12	
37	P0.16			38	P1.11	
39	P0.15			40	P1.7	
41	P1.20			42	P1.21	
43	GND			44	Power_enable	
45	GND			46	Reset	
46	n.c			48	n.c	

Tabelle B.1.: Pin-Belegung des IO-Boards

C. CD-ROM Inhalt

Nachfolgend eine Auflistung der Inhalte auf der beigelegten CD-ROM:

Verzeichnis	Inhalte
Root	- Bachelor Thesis
Literatur	- IEEE: TFALDA for Autonomous Driving
Manuals	- Commell LV-668 Mainboard - NXP LPC2468 Desciption - NXP LPC24xx Datasheet - NXP COM-Board Desciption
Software	- PC (Fahrspurerkennung) - LPC (Lenkansteuerung)
Treiber	- IXXAT USB-to-CAN Converter - IXXAT USB-to-CAN Converter

Glossar

t_{ein} *Einschaltzeit*
Dauer des ‘high’-Pegels eines PWM-Signals

ACC *Adaptive Cruise Control*
System zur automatischen Abstandseinhaltung mittels Radartechnik in KFZ

AFIL *Alarm bei Fahrspurabweichung durch Infrarot Linienerkennung*
Assistenzsystem zur Warnung bei Fahrbahnüberschreitungen im KFZ

ESP *Elektronisches Stabilitäts Programm*
Assistenzsystem zur Vermeidung ungewollten Schleuderns in KFZ

fps *Frames per second*
Bildwiederholrate angegeben in Bildern pro Sekunde

Lane Assist
Assistenzsystem zur einhaltung von Fahrspuren mit einem KFZ

lane-candidate
Eine erkannte mögliche Fahrspur innerhalb eines Bildes.

PWM *Pulsweitenmodulation*
Art zur Modulation eines Tastverhältnisses mit gleichbleibender Frequenz

ROI *Region Of Interrest*
Ausschnitt eines Bildes, welcher relevante Daten zur bearbeitung enthält

RTOS *Real-Time Operating System*
Um Echtzeit-Funktionen erweitertes Betriebssystem um ein bestimmtes Zeitverhalten zu erreichen und gewisse Prozessverhalten voraussagen zu können

SDK *Software-Development-Kit*
Dokumentations- und Softwarepacket zu Hard-/Software um Entwicklern das Implementieren derer in eigene Anwendungen zu erleichtern

TFALDA *Three-Feature Based Automatic Lane Detection Algorithm*
Bezeichnet einen robusten Algorithmus zur Erkennung von Fahrspuren in einem Bild.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 8. September 2008	
Ort, Datum	Unterschrift