Institut für Informatik

Bachelorarbeit

Kreuzungserkennung und -verhalten bei autonomen Modellfahrzeugen

---

von Benjamin Zengin

Betreuer:

Prof. Dr. Raúl Rojas

18.09.2014

Inhaltsverzeichnis

Zusammenfassung

Im jährlich stattfindenden Carolo-Cup treten studentische Teams mit selbstentwickelten autonomen Modellfahrzeugen in verschiedenen Disziplinen gegeneinander an. In der Disziplin „Rundkurs mit Hindernissen“ müssen unter anderem auch Kreuzungen erkannt und an diesen angehalten werden. In dieser Arbeit wird eben dieses Erkennen und Verhalten an Kreuzungen behandelt. Zunächst wird die Entwicklungsumgebung und die Grundlagen auf denen diese Arbeit aufbaut beschrieben, anschließend folgt die Problemstellung und es wird analysiert anhand welcher Merkmale eine Kreuzung zu klassifizieren ist. Danach wird ein Ansatz zur Erkennung dieser Merkmale vorgestellt und eine Vorgehensweise entwickelt, die anhand der Erkennung das Verhalten an einer Kreuzung steuert. Abschließend werden Praxiserfahrungen präsentiert und ein Ausblick auf weiterführende Arbeiten gegeben.

1 Einleitung

Die Erforschung und Entwicklung autonomer Roboter und Fahrzeuge ist in vollem Gange. Die Idee, dass ein Auto völlig selbständig und sicher seinen Weg von einem Punkt zum anderen findet, ist faszinierend. Durch die Steuerung des Autos von einem Computer, soll der Straßenverkehr sicherer werden [1]. Verschiedene namenhafte Unternehmen haben bereits Prototypen in der Entwicklung, darunter Google [2] und Volvo [3]. Laut einer Umfrage des IEEE, werden bereits im Jahre 2035 seriengefertigte Automobile weder Lenkräder noch Pedale besitzen [4].

Das autonome Fahren ist jedoch sehr komplex, denn viele verschiedene Aspekte müssen beachtet werden. Spurerkennung, Hindernisse, Vorfahrt und Lokalisierung sind nur einige davon. Mit diesen Themen beschäftigen sich in kleinerer Ausgabe auch die Teilnehmer des jährlich von der TU Braunschweig veranstaltetem „Carolo-Cup“. Für diesen Wettbewerb entwickeln studentische Teams autonome Modellfahrzeuge und lassen diese in verschiedenen Disziplinen gegeneinander antreten.

1.1 Carolo-Cup

Der Carolo-Cup wird, wie bereits erwähnt, von der TU Braunschweig veranstaltet und findet seit 2008 jährlich statt. Ziel des Wettbewerbs ist es, ein autonomes Modellauto im Maßstab 1:10 zu entwickeln und mit diesem verschiedene Aufgaben zu bewältigen. Der Wettbewerb besteht aus einem statischen Teil, in dem das Konzept des Fahrzeuges präsentiert werden soll, sowie aus einem dynamischen, der folgende Disziplinen umfasst [5]:

1. Rundkurs ohne Hindernisse

   In diesem Teil muss das Auto auf der rechten Spur einen Rundkurs abfahren. Kreuzungen können ignoriert werden. Vor dem Start der Fahrt werden an einige Stellen die Fahrbahnmarkierungen entfernt. Dabei muss allerdings immer mindestens eine der drei Linien vorhanden bleiben. Falls das Auto von der Bahn

abkommt, kann es gegen Strafpunkte mithilfe einer Fernbedienung zurück auf die Strecke geführt werden.

2. Rundkurs mit Hindernissen

   Diese Disziplin findet auf dem gleichen Rundkurs statt, allerdings können sich nun stehende, oder sich bewegende Hindernisse (Weiße Würfel aus Pappe) auf der Strecke befinden. Diesen muss ausgewichen werden, indem auf die linke Spur gewechselt wird und nach dem Hindernis wieder eingeschert wird. Beim Spurwechsel muss der Blinker gesetzt werden. In dieser Disziplin müssen nun auch Kreuzungen beachtet werden, d.h. an der Stopplinie angehalten werden. Hier kann ebenfalls ein Hindernis auftreten, das die Kreuzung passiert. Es darf erst weiter gefahren werden, sobald die Kreuzung wieder frei ist.

3. Paralleles Einparken

   Hier ist die Aufgabe an einem Parkstreifen entlang zu fahren, eine ausreichend große Parklücke zu finden und anschließend rückwärts einzuparken. Auf dem Parkstreifen stehen mehrere weiße Würfel, die verschieden große Lücken bilden.

1.2 Ziel der Arbeit

Diese Arbeit umfasst folgende zwei Ziele:

1. Entwickeln einer Kreuzungserkennung

   Auf Bildern, die von einer am Fahrzeug befestigten Kamera geliefert werden, sollen Kreuzungen erkannt werden.

2. Entwickeln eines Kreuzungsverhaltens

   Auf Basis der entwickelten Kreuzungserkennung soll das Fahrzeug, sofern notwendig, an der Kreuzung anhalten.

Diese Ziele stehen im Kontext zum Carolo-Cup, d.h. das Analysieren der Problemstellung und das Entwickeln des Ansatzes stehen im direkten Bezug auf das Carolo-Cup-Regelwerk.

2 Entwicklungsumgebung

Das Auto (Testträger) auf dem die entwickelte Software läuft, war bereits vorhanden als ich mit meiner Arbeit begann. Da der hier vorgestellte Ansatz für den Testträger entwickelt und auch mit ihm getestet wurde, werde ich einen kurzen Überblick über die verwendete Hardware und Software geben, welche genauer in [6] beschrieben sind.

2.1 Hardware

Das Auto benutzt als zentrale Komponente den Einplatinencomputer Odroid-X2 auf dem sämtliche kognitive Prozesse ablaufen. Er ist mit einem 1.7 GHz ARM Cortex-A9 Quad Core Prozessor ausgestattet, besitzt 2 GB Arbeitsspeicher und verschiedene Schnittstellen. Das Auto verfügt über eine Webcam mit omnidirektionaler Linse, die über USB mit dem Hauptrechner verbunden ist. Sie ist an einem Mast über dem Auto angebracht und ermöglicht von dort eine 360° Sicht um das Auto. Die Rundumsicht wird realisiert, indem die Webcam von unten auf einen parabolischen Spiegel gerichtet ist, der die Umgebung auf ein Bild krümmt. Über eine serielle Schnittstelle ist der Rechner mit einem Mikrocontrollerboard, dem „OttoV2“ verbunden. Es handelt sich bei dem Board um eine Eigenentwicklung des Teams, welche dazu verwendet wird die im Folgenden genannte Hardware anzusteuern und zu kontrollieren. Der Antrieb ist ein Elektromotor mit präziser Geschwindigkeitsregelung, was dazu führt das er zum einen sehr langsam, aber zum anderen auch bis zu 4 m/s schnell fahren kann. Die Lenkung wird mit einem Lenkservo realisiert. Das Auto besitzt außerdem einen Empfänger, der im 40 MHz Band operiert. Dieser ist dafür da, um im Notfall von autonomer auf manuelle Fahrt umzuschalten und das Auto mit einer Fernbedienung zu steuern. Da im Carolo-Cup auch Lichtsignale gefordert sind, verfügt es weiterhin über Bremslichter, Blinker und ein Licht, welches leuchtet wenn das Fahrzeug ferngesteuert wird. Um am Auto die Programme für die verschiedenen Disziplinen starten zu können, ist ein Bedienpaneel oben in die Karosserie eingelassen. Es enthält einen LCD Bildschirm, der ein Menü anzeigt, durch das sich mit einem Einstellrädchen und zwei Knöpfen navigieren lässt. Der benötigte Strom wird mit einem selbst entwickelten Powerboard aus einem Lithium-Polymer-Akku zur Verfügung gestellt. Da Lithium-Polymer-Akkus

durch Tiefenentladung beschädigt werden können, trennt das Powerboard den Akku bei zu großer Entladung von allen Verbrauchern.

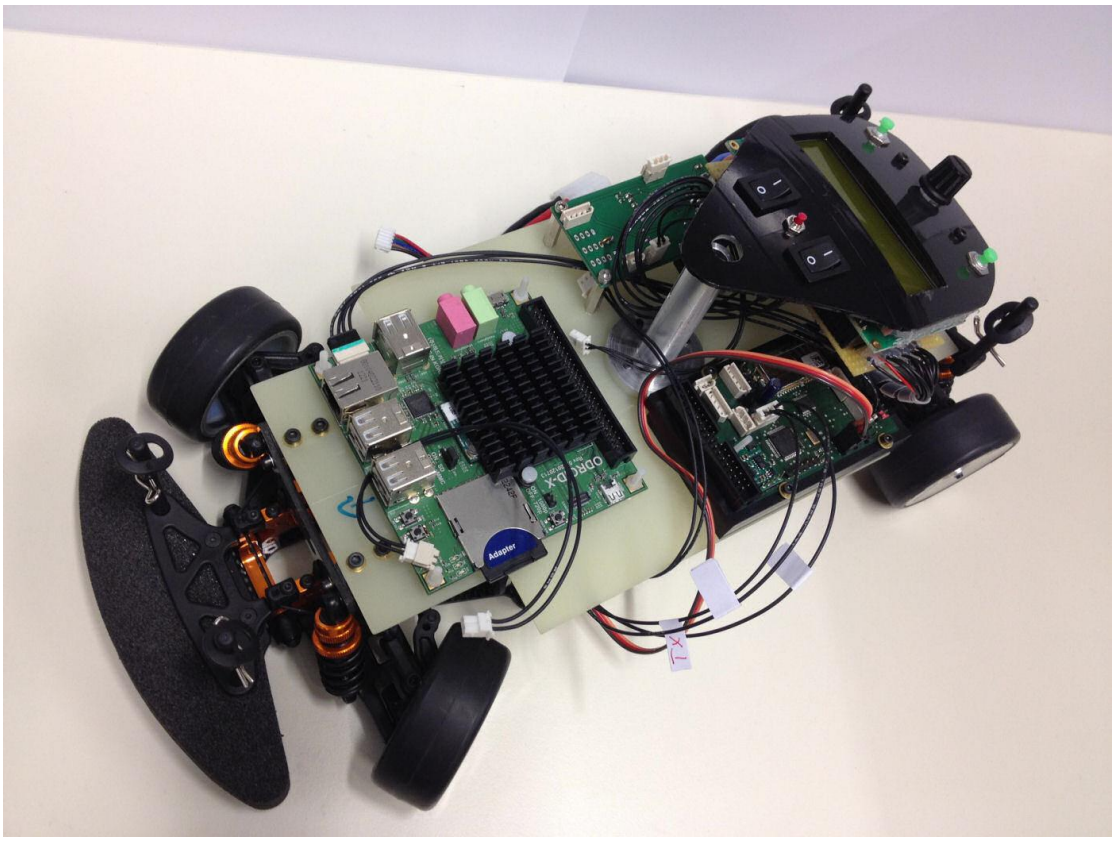

Abbildung 2.1: Fahrgestell mit montierten Komponenten

(Odroid: links, Powerboard: oben, OttoV2: unten, Bedienpaneel: rechts) [6]

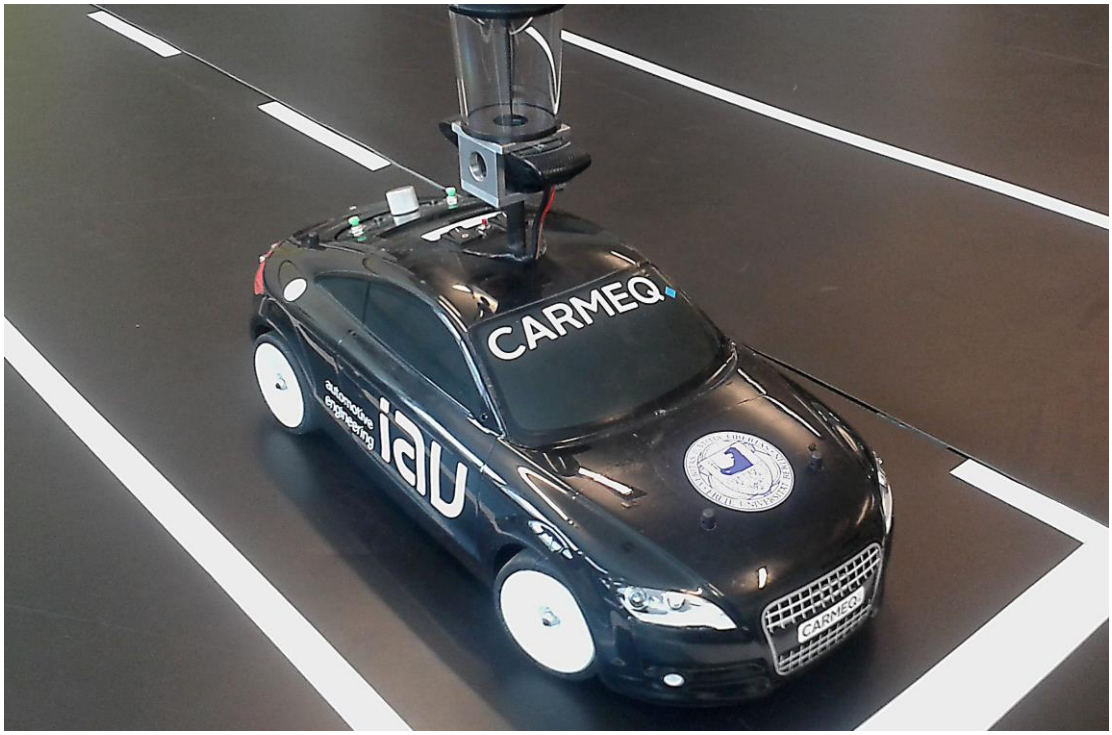

Abbildung 2.2: Fahrzeug mit Karosserie und montierter omnidirektionaler Kamera

2.2 Software

Die Software für den Carolo-Cup wird zum Großteil in C++ geschrieben, wobei das Mikrocontrollerboard in C programmiert wird. Für die Entwicklung der kognitiven Prozesse wird das Framework von Berlin United verwendet, das ursprünglich nur für die Verwendung mit den RoboCup Fußball Robotern gedacht war. Da die Architektur sich jedoch, aufgrund des ähnlichen Vorgehensschemas, auch für autonomes Fahren eignet, wurde es für das Carolo-Cup Projekt übernommen.

Das Framework besteht aus zwei verschiedenen Teilen – Module und Repräsentationen. Repräsentationen sind ausschließlich dafür da Daten bereit zu stellen. Module hingegen sind dafür gedacht Berechnungen durchzuführen. Dabei kann es Repräsentationen referenzieren, um Daten einzufordern oder bereitzustellen. Auf Basis der Referenzen zwischen den Modulen und Repräsentationen berechnet das Framework eine Ausführungskette. Diese wird immer wieder sequentiell durchlaufen. Dabei folgt das Framework dem Sense-Think-Act Paradigma, was bedeutet, dass zuerst alle Daten der Sensoren eingeholt werden (Sense), diese anschließend verarbeitet werden (Think) und zum Schluss ein Verhalten daraus bestimmt wird (Act).

3 Grundlagen

Da das Team von „Berlin United“ schon seit 2012 am Carolo-Cup teilnimmt, ist die Menge von bereits implementierten Funktionalitäten natürlich groß. Diese Arbeit baut auf dem Vorhandenen auf, weshalb die relevanten Mechanismen im Folgenden erläutert werden.

3.1 Kamera

Bevor Spuren, Hindernisse oder Kreuzungen erkannt werden können, muss es eine Möglichkeit der Wahrnehmung geben. Beim verwendeten Auto besteht diese aus einer omnidirektionalen Kamera, die pro Sekunde 30 Bilder liefert. Diese hat den Vorteil, dass in alle Richtungen gesehen werden kann, allerdings auch den Nachteil, dass nicht sehr weit geblickt wird, da das Bild stark verzerrt ist.

Momentan wird auch eine Stereokamera in das Auto eingepflegt, die frontal auf dem Auto angebracht wird. Sie soll dazu dienen Hindernisse erkennen zu können. Für die Kreuzungserkennung wäre diese auch dienlich, da weiter nach vorne geblickt werden kann, die Kreuzung früher erkannt wird und somit z.B. ein langsames Anfahren an die Kreuzung realisiert werden könnte.

3.2 Fahren

Das Halten der Spuren und Fahren entlang des Kurses wird zurzeit durch die von Lukas Maischak implementierte Lokalisierung realisiert. Dabei muss zunächst eine Karte von der Strecke erstellt werden. Dies geschieht durch manuelles Abfahren. Anschließend wird auf Basis der aufgezeichneten Karte ein „Force Field“ berechnet, welches verwendet wird, um Streckenabschnitte lokalisieren zu können, auch wenn diese im Vergleich zur ursprünglichen Karte gedreht sind. Nachdem ein abzufahrender Pfad auf der Karte definiert wurde, kann das Auto auf die Startposition gesetzt werden, um diesem zu folgen. Dies geschieht, indem zu jedem Bild mithilfe des Algorithmus die globale Position und Orientierung („Pose“) erkannt und aktualisiert wird. Anhand dieser kann der nächste Wegpunkt des Pfades bestimmt und das entsprechende Fahrmanöver

geplant werden. Mit der Lokalisierung ließe sich zwar auch ein Kreuzungsverhalten realisieren, allerdings kann es passieren, dass die Lokalisierung z.B. auf langen Geraden aufgrund zu weniger Merkmale ihre Position verliert. Deshalb braucht es sowohl für das Spurhalten, als auch für die Kreuzungserkennung eine Absicherung.

3.3 Kantenerkennung

Wenn man über einen Ansatz nachdenkt, Spuren auf einem Bild zu erkennen, dann wäre eine naive Idee alle Pixel des Bildes zu überprüfen und die Bereiche zu markieren in denen sich besonders viel weiß befindet. Dieser Ansatz ist jedoch sehr rechenintensiv. Stattdessen werden Kanten im Bild berechnet. Kanten entsprechen Stellen im Bild, an denen es Helligkeitsübergänge gibt. Diese werden auf dem Testträger berechnet, indem mehrere Scanlinien horizontal und vertikal über das Bild gelegt werden und mithilfe eines Kantendetektions-Filters die Gradienten der Helligkeitswerte entlang der Linien berechnet werden. Im vorliegenden System wird dieses Verfahren unter Benutzung des Prewitt-Operators [7] als Filter implementiert. Abschließend wird noch eine „Non-maximum suppression“ (NMS) durchgeführt. Diese sorgt dafür, dass an allen lokalen Gradientenmaxima alle Werte, die sich um das Maximum herum befinden, unterdrückt werden. Dadurch wird bewirkt, dass nur noch allein das Maximum vorhanden ist und somit doppelte Kanten verhindert werden.

3.4 Pixel vs. relative Koordinate

Bei Verarbeiten des Bildes, egal ob Kantenerkennung oder Farbanalyse, sind die Ergebnisse der Algorithmen immer Pixel oder Pixelgruppen. Um jedoch ein Fahrmanöver planen zu können, müssen die Pixel des Bildes in reale Entfernungen umgerechnet werden. Das Auto verwendet wie erwähnt eine omnidirektionale Kamera. Da die Linse der Kamera zentral auf den parabolischen Spiegel gerichtet ist, entsteht eine Symmetrieachse, die mittig und senkrecht auf dem resultierenden Bild steht. Zwei Pixel, die denselben Abstand zur Mitte des Bildes haben, sind auch in der Realität gleich weit entfernt. Es muss folglich eine Funktion gefunden werden, welche den Pixelabstand zum Zentrum in das relative Koordinatensystem des Autos umrechnet. Diese Abbildung kann durch ein Polynom beschrieben werden, das mithilfe einer Kalibriermatte mit bekannten Abständen interpoliert wird. Das relative Koordinatensystem wird dazu verwendet um Punkten den realen Abstand (in cm) zum Auto zuordnen zu können. Der Koordinatenursprung ist dort, wo sich die

Symmetrieachse der Kamera befindet. Die x-Achse verläuft parallel zum Auto nach vorne und die y-Achse nach links.

3.5 Forschungsstand

Es gibt bereits verschiedene Ansätze zur Erkennung von Kreuzungen. Diese sind zwar auf den echten Straßenverkehr bezogen, aber bieten trotzdem einen guten Einblick in die Thematik. Ein Ansatz zur Erkennung ist es, die seitlichen Spurbegrenzungen mittels verschiedener Constraints zu erkennen und eine Kreuzung daran zu definieren, dass diese Spurbegrenzungen eine Lücke enthalten und es von den Seiten aus Spurbegrenzungen quer zur Spur gibt, die diese Theorie unterstützen [8]. Es wird dort jedoch auch zugegeben, dass das Fehlen von Spurbegrenzungen nur ein schwacher Beleg für die Existenz einer Kreuzung ist. Ein anderer Ansatz ist es die Kreuzungserkennung erst durchzuführen, wenn mittels einer Karte festgestellt wird, dass demnächst eine erreicht wird [9]. Da meine Arbeit allerdings eine Alternative zur Lokalisierung bieten soll, steht keine Karte zur Verfügung.

Die Anzahl der Arbeiten zum Thema Kreuzungserkennung ist eher gering. Viel mehr findet man zum Thema Spurerkennung. Diese Thematik ist für diese Arbeit jedoch auch von Bedeutung, da bei der Spurerkennung Linien erkannt werden müssen, was bei der Kreuzung genauso der Fall ist. Ein Ansatz ist das Verwenden von neuronalen Netzen [10]. Der Vorteil ist, dass sie auf vielen verschiedenen Straßentypen unter verschiedenen Verhältnissen funktionieren. Allerdings wäre dieser Ansatz für das Kreuzungserkennungsproblem zu weit gegriffen, da die Verhältnisse im Carolo-Cup gleichbleibend sind. Eine andere Möglichkeit ist das Aufsetzen der Erkennung auf den Farben des Bildes [11]. Hier müssen allerdings alle Pixel in die Berechnung mit einbezogen werden. Deshalb werden in anderen Ansätzen die Kanten als Grundlage für eine Linienerkennung verwendet, was effizienter ist [12], [13]. Da die Kreuzung im Carolo-Cup eindeutig definiert ist und keine Variation zulässt, habe ich mich für diese Alternative entschieden.

4 Problemstellung

In der Disziplin „Rundkurs mit Hindernissen“ müssen Kreuzungen beachtet werden, d.h. das Auto muss an den Stopplinien halten. Dabei ist zu beachten, dass mindestens zwei Sekunden lang gehalten wird und, dass sich die Fahrzeugvorderkante vor der Linie befindet, allerdings nicht mehr als 15 cm von dieser entfernt ist [5].

Es kann der Fall auftreten, dass während des Haltevorgangs ein dynamisches Hindernis die Kreuzung überquert. Dann muss zusätzlich zu den zwei Sekunden so lange gewartet werden, bis das Hindernis die Kreuzung passiert hat. Diese Erkennung ist allerdings nicht Bestandteil dieser Arbeit.

Wenn das Auto die Kreuzung parallel zu Stopplinie erreicht, so muss nicht gehalten werden.

Bei einer Kreuzung treffen zwei Fahrbahnen im rechten Winkel aufeinander. Auf zwei gegenüber liegenden Seiten der Kreuzung befindet sich jeweils auf der rechten Fahrbahn eine 40 mm breite Stopplinie (Abbildung 4.1).

Zusätzlich gibt es eine Startlinie (Abbildung 4.2), die genauso breit wie die Stopplinie ist, allerdings die linke und rechte Spur bedeckt. An dieser soll nicht gehalten werden.

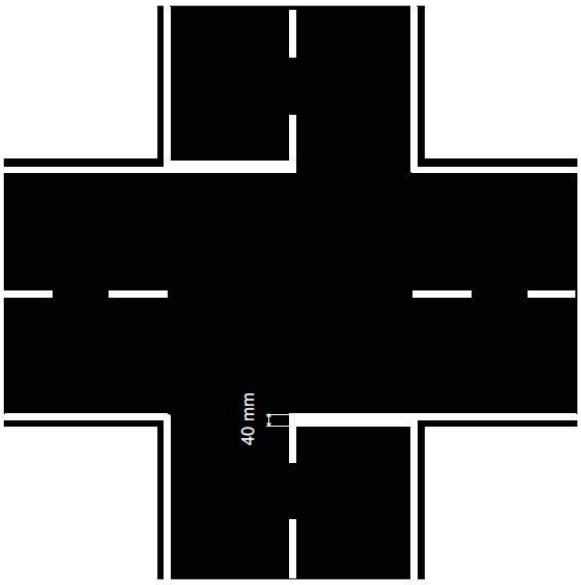

Abbildung 4.1: Stopplinie [5]

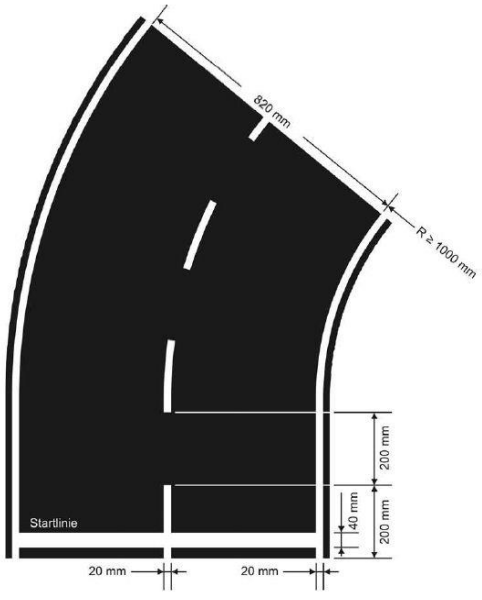

Abbildung 4.2: Startlinie [5]

5 Kreuzungserkennung

Im folgenden Abschnitt wird der Ansatz zur Erkennung der Kreuzung vorgestellt. Zuerst wird analysiert, auf Basis welcher Merkmale die Erkennung stattfindet und anschließend der entwickelte Algorithmus beschrieben.

5.1 Mögliche Merkmale

Um eine Vorgehensweise für das Erkennen von Kreuzungen entwickeln zu können, muss zuerst einmal analysiert werden, welche Merkmale eine Kreuzung als solche kennzeichnen. Folgende zwei Merkmale kommen in Frage:

1. Unterbrechung der seitlichen Spurbegrenzungen

   Da sich an der Kreuzung zwei Spuren treffen, sind die Spurbegrenzungen an den Seiten nicht vorhanden. Es wäre also möglich die Erkennung auf das Fehlen der Begrenzungen zu stützen, wie z.B. auch in [8]. Dieser Ansatz ist jedoch in zweierlei Hinsicht problematisch. Zum einen könnte es sein, dass ein dynamisches Hindernis fälschlicherweise als Seitenbegrenzung eingestuft und daher die gesamte Kreuzung ignoriert wird. Zum anderen, könnte es sein, dass vor dem Wettbewerb entfernte Seitenbegrenzungen beim Fahren zum Erkennen einer Kreuzung führen, obwohl keine vorhanden ist.

2. Stopplinie

   Da jede Kreuzung eine Stopplinie an der Einfahrt besitzt, könnte diese zur Definition herangezogen werden. Die Stopplinie ist im Kontext der gesamten Strecke eine einzigartige Erscheinung. Lediglich die Startlinie erstreckt sich ebenfalls quer zur Strecke. Um die Erkennung basierend auf der Stopplinie zu entwickeln, müsste folglich eine explizite Differenzierung zur Startlinie erfolgen.

Die Stopplinie bietet augenscheinlich die besten Voraussetzungen für eine erfolgreiche Erkennung und wird deshalb auch für den vorgestellten Ansatz als Merkmal verwendet.

5.2 Ansatz

Zur Erkennung der Kreuzung gehört nicht nur die reine Klassifizierung des Bildes (Kreuzung: Ja oder Nein?), sondern auch das Feststellen der Position der Kreuzung. Dies ist erforderlich, damit auf Basis der Erkennung ein Verhalten stattfinden kann, welches das Auto an der richtigen Stelle anhalten lässt. Gesucht ist also ein Algorithmus, der Linien in einem Bild erkennt und diesen auch eine Position zuordnet. Diese Bedingungen erfüllt die Hough-Transformation [14].

5.2.1 Hough-Transformation

Die Hough-Transformation ist ein Verfahren, das vorhandene Geraden in einer Punktmenge berechnet. Als Erstes muss definiert werden, wie die zu berechnenden Geraden parametrisiert dargestellt werden können. Hierfür wird die Hessesche Normalform $d = x \cdot \cos(\alpha) + y \cdot \sin(\alpha)$ verwendet, wobei d der Abstand der Gerade zum Koordinatenursprung und α der Winkel zwischen der Normalen der Gerade und der x-Achse ist. Sie hat gegenüber der sonst in der Mathematik üblichen Form $y = m \cdot x + n$ den Vorteil, dass vertikale Linien dargestellt werden können und, dass der Abstand vom Ursprung direkt in der Darstellung enthalten ist. Über diesen Parametern wird eine Matrix angelegt. Anschließend werden für jeden Punkt der eingegebenen Punktmenge alle durch ihn verlaufenden Geraden bestimmt und die entsprechenden Stellen für d und α in der Matrix inkrementiert. Dieser Vorgang nennt sich „Voting“, weshalb die zu Grunde liegende Matrix auch „Voting-Matrix“ heißt. Am Ende werden besonders hohe Werte mittels eines Schwellwerts in der Matrix lokalisiert und die dazugehörigen Parameter vermerkt. Diese beschreiben die im Bild enthaltenen Geraden.

5.2.2 Anwendung

Die Erkennung benötigt zuerst eine Punktmenge auf der gearbeitet wird. Hierfür werden die auf dem Bild festgestellten Kanten verwendet. Da es positive (dunkel zu hell) und negative Kanten (hell zu dunkel) gibt, müssen für diese zwei verschiedene Voting-Matrizen angelegt werden. Da die Stopplinie immer quer zum Auto steht, kann der Prozess dadurch beschleunigt werden, dass man ihn auf horizontale Linien reduziert. Dies kann zum einen dadurch erreicht werden, dass nur die parallel zum Auto verlaufenden Scanlinien verwendet werden und zum anderen dadurch, dass für den Parameter α nur Winkel mit kleinem Betrag in Betracht gezogen werden, also z.B. $-30° \leq \alpha \leq 30°$. Außerdem können alle Kanten, die sich hinter der Nase des Autos befinden, ignoriert werden, da man nur auf die Stopplinie reagieren muss, wenn sie sich vor dem Auto befindet. Letztlich muss durch Testen noch ein geeigneter Schwellwert

für die Werte in der Voting-Matrix festgelegt werden. Der Wert ist dann gut gesetzt, wenn so wenig wie möglich Geraden im Bild erkannt werden und gleichzeitig die Stopplinie aber immer als Gerade vorhanden ist.

Mit der Durchführung der Hough-Transformation erhält man nun also eine Menge von Geraden, die auf dem gesamten vorderen Teil des Bildes liegen. Aus diesen Daten muss nun eine Schlussfolgerung auf die Existenz einer Kreuzung gezogen werden. Dies ist aber komplizierter als gedacht, da nun zwar Linien erkannt wurden, diese aber keine Informationen über Start- und Endpunkt enthalten. Es ist schwer zu differenzieren, ob eine Gerade die Stopplinie beschreibt, oder aber die Spurbegrenzung der quer verlaufenden Spur. Würde man nun die Stopplinie nur anhand der Existenz von horizontalen Geraden definieren, so würde sie auch erkannt werden, wenn man in Vorfahrtsrichtung in die Kreuzung einfährt. Dann würden nämlich die Spurbegrenzungen der kreuzenden Spur als horizontale Linie erkannt und fälschlicherweise als Stopplinie interpretiert werden.

Für dieses Problem gibt es zwei mögliche Lösungen. Zum einen könnte man zusätzlich zur Erkennung der Geraden noch ein Feststellen der Start- und Endpunkte implementieren. Dann würden allerdings noch weitere Probleme aufkommen, wie z.B. wenn zwei Linien auf derselben Gerade liegen, aber nicht miteinander verbunden sind. Die andere Möglichkeit wäre eine „Region of Interest“ (ROI) anzulegen, welche dafür sorgt, dass die Hough-Transformation nicht auf der gesamten Punktmenge durchgeführt wird, sondern nur auf einem bestimmten und „interessanten“ Teil. Diesen Ansatz habe ich gewählt, da er keine weiteren Probleme aufwirft und wie im Folgenden erläutert, auch noch andere Vorzüge hat.

5.2.3 Anlegen einer Region of Interest

Wenn sich das Auto auf eine Kreuzung zu bewegt, dann ist die Stopplinie an immer der gleichen Position zu erwarten. Es reicht also aus um diese Position einen Rahmen zu legen und die Linienfindung ausschließlich auf diesen Bereich zu fokussieren. Wenn bei der Verarbeitung innerhalb dieses Bereiches eine horizontale Linie gefunden wird, so ist die Wahrscheinlichkeit sehr hoch, dass es eine Stopplinie ist. Außerdem nimmt die Hough-Transformation durch die erheblich verkleinerte Eingabemenge deutlich weniger Zeit in Anspruch. Die ROI wird so gelegt, dass sie den rechten Fahrstreifen bedeckt. Die Länge kann z.B. auf die Reichweite der Kamera gesetzt werden, wobei eine Verkürzung auch sinnvoll ist, da dadurch die Häufigkeit von „False Positives“ verringert werden kann (siehe Abschnitt 7).

5.2.4 Interpretieren der Daten

Nun da eine ROI angelegt ist und innerhalb dieser Geraden erkannt werden, müssen die gewonnenen Daten noch interpretiert werden, sodass eine Kreuzungserkennung stattfinden kann. Wird nun eine Gerade erkannt, so kann man sich sicher sein, dass diese an der Position der Stopplinie ist, da ausschließlich Punkte die in der ROI sind für die Transformation verwendet werden. Dadurch fällt die Arbeit für die Überprüfung weg, ob sich die Gerade an der Position der Stopplinie befindet und es muss nur noch geprüft werden, ob generell eine Linie vorhanden ist. Dazu werden beide Kantentypen mit einbezogen. Die negativen Kanten befinden sich vom Auto aus gesehen vor der Linie, die positiven Kanten dahinter. Es wird daher zuerst geprüft, ob in der Eingabemenge der positiven Kanten eine Gerade erkannt wurde. Dies dient als Absicherung, da z.B. bei Reflexionen auf der Fahrbahn manchmal eine Gerade in den negativen Kanten gefunden wird, aber nicht in den positiven. Bei einer Stopplinie wird hingegen beides gefunden. Nachdem die positiven Kanten überprüft wurden, werden die negativen Kanten behandelt. Aus den gefundenen Geraden muss nun, sofern vorhanden, eine ausgewählt werden, welche die Stopplinie repräsentieren soll. Dies kann durch Bilden des Durchschnitts über alle Geraden erfolgen oder aber durch finden des maximalen Wertes in der Voting-Matrix. Man erhält mit der gefundenen Gerade die Werte der Parameter d und α . Aus diesen berechnet man mittels trigonometrischer Beziehungen die relativen Koordinaten des Wegpunktes, an dem das Auto zum Stillstand kommen soll, indem man von d den Abstand zwischen Kamera und Fahrzeugfront abzieht und den resultierenden Wert als Hypotenuse verwendet.

Zu diesem Zeitpunkt ist die Kreuzungserkennung bereits funktionsfähig, problematisch ist nun aber noch, dass eine Startlinie in der ROI auch eine Geradenerkennung verursachen würde. Um diesen Fehler zu beheben, wird eine zweite ROI angelegt. Sie ist von der Größe her identisch mit der ersten, allerdings wird sie auf dem linken Fahrstreifen platziert. Bei der Transformation werden nun die Punkte in der linken ROI in einer eigenen Matrix verarbeitet. Falls nach Anwenden des Schwellwerts eine Gerade erkannt wird, dann liegt eine Startlinie vor und es muss kein Wegpunkt zum Anhalten berechnet werden.

6 Kreuzungsverhalten

Die Ergebnisse der Kreuzungserkennung allein sorgen noch nicht dafür, dass das Auto an der Stopplinie anhält. Dafür muss ein weiteres Modul entworfen werden, das die Daten der Kreuzungserkennung als Eingabe erhält und daraus ein Verhalten festlegt. Da das Auto bereits mittels Lokalisierung einen Pfad abfährt, liegt schon ein Modul zur Steuerung des Verhaltens vor. Die Kreuzungserkennung kann dort direkt integriert werden.

6.1 Integration

Das vorhandene Verhaltensmodul ist ein Strategiemodul, d.h. es wechselt zwischen verschiedenen konkreten Verhaltensmodulen. Bisher tat das Modul nichts anderes als das Pfad-Folgen-Verhalten zu aktivieren und wiederholt für eine gewisse Reichweite die Geschwindigkeit zu setzen. Die Integration des Kreuzungsverhaltens erfolgt direkt in das Strategiemodul, da für das Anhalten an der Kreuzung lediglich das Leerlaufverhalten zum richtigen Zeitpunkt aktiviert werden muss, damit das Auto anhält. Es muss also kein komplexes Verhaltensmodul für das Verhalten an Kreuzungen geschrieben werden. Um das Kreuzungsverhalten zu integrieren, erhält das Modul eine Variable zur Speicherung des aktuellen Status (Pfad folgen oder Kreuzung), über den geregelt wird welche Aktion als nächstes möglich ist.

6.2 Schnittstelle

Die Daten der Kreuzungserkennung müssen bereitgestellt werden, damit auf Basis dieser ein Verhalten stattfinden kann. Dafür wird eine Repräsentation angelegt, die einerseits einen Wahrheitswert besitzt, der wiedergibt, ob eine Kreuzung erkannt wurde. Andererseits enthält sie den konkreten Wegpunkt, an dem angehalten werden muss, um vor der Kreuzung zu stehen. Die Variablen der Repräsentation werden in der Kreuzungserkennung bereitgestellt und im Strategiemodul als Eingabe referenziert.

6.3 Steuerung des Anhaltens

Das Strategiemodul hat nun wie erwähnt zwei verschiedene Status um das Verhalten festzulegen. Zu Anfang befindet sich das Modul im Pfad-Folgen-Verhalten. Ist das Modul in diesem Status, so überprüft es wiederholt, ob eine Kreuzungserkennung erfolgreich war. Ist dies der Fall, so wird weiterhin überprüft, ob ein Anhalten überhaupt erlaubt ist. Das Anhalten wird nämlich für eine gewisse Zeit verboten, sobald an einer Kreuzung gehalten wurde. Dadurch wird doppeltes Anhalten verhindert. Sind alle Kriterien erfüllt, dann wird berechnet, wie weit der Wegpunkt vom Auto entfernt ist. Ist die Entfernung kleiner als ein vorher definierter Wert, dann wird das Leerlauf-Verhalten aktiviert und das Auto kommt zum Stillstand. Der definierte Entfernungswert muss dem Bremsweg entsprechen, damit das Auto die Bremsung so früh startet, dass beim Wegpunkt der absolute Stillstand erreicht ist. Der Bremsweg ist von der Geschwindigkeit des Autos abhängig. Da diese beim aktuellen Pfad-Folgen-Verhalten jedoch konstant ist, kann der Bremsweg experimentell bestimmt werden. Falls sich allerdings die Geschwindigkeit des Autos ändern würde, so müsste die Entfernung mithilfe folgender Formel bestimmt werden:

$$s_B = \frac{{v_0}^2}{2a} \qquad s_B: \text{Bremsweg} \qquad v_0: \text{Geschwindigkeit} \qquad a: \text{Bremsverzögerung}$$

Die Bremsverzögerung a müsste in einer Versuchsreihe mit verschiedenen Geschwindigkeiten ermittelt werden. Anschließend könnte man den Schwellwert für die Entfernung zum Anhalten dynamisch festlegen.

Sobald das Auto zum Stillstand gekommen ist wird ein Countdown gestartet. Wenn dieser abgelaufen ist, wird wieder auf das Pfad-Folgen-Verhalten gewechselt. Der Countdown muss aufgrund der Regelung des Carolo-Cups mindestens zwei Sekunden betragen. Außerdem müsste hier ebenfalls die Weiterfahrt blockiert werden, falls ein dynamisches Hindernis in die Kreuzung einfährt. Nach dem Halten an der Stopplinie wird ein weiterer Countdown gestartet, der das Anhalten für eine gewisse Zeit verhindert (s.o.). Danach wird wieder vom Anfang begonnen.

7 Zuverlässigkeit und Probleme

Bis jetzt wurde nur die Vorgehensweise beschrieben, aber noch nichts von der Praxistauglichkeit erwähnt. Im folgenden Teil wird die Zuverlässigkeit des Algorithmus analysiert, Stärken und Schwächen aufgezeigt und ein Ausblick auf Ansätze zur Verbesserung gegeben.

7.1 Erkennungsrate

Die Erkennungsrate des vorgestellten Ansatzes ist sehr gut. In den durchgeführten Testläufen auf der Teststrecke im Labor wurde die Kreuzung immer erkannt. Allerdings ist der Algorithmus natürlich an verschiedene Faktoren gebunden, damit er funktioniert. Zum einen darf die Geschwindigkeit nicht so groß werden, dass die Anzahl der Bilder pro Sekunde nicht ausreicht, um das Fahrzeug rechtzeitig zu stoppen. Aktuell fährt das Auto mit konstanten 1 m/s, was bei 30 Bildern pro Sekunde bedeutet, dass alle 3 cm ein neues Bild zum Auswerten verfügbar ist. Wird dieser Abstand zu groß, dann könnte es passieren, dass der Bremsvorgang zu spät gestartet wird. Problematisch könnten auch die Lichtverhältnisse sein. Ist die Strecke nicht gut beleuchtet, sodass die Kantenerkennung keine korrekten Daten liefert, dann kann folglich auch die Kreuzungserkennung nicht erfolgreich sein.

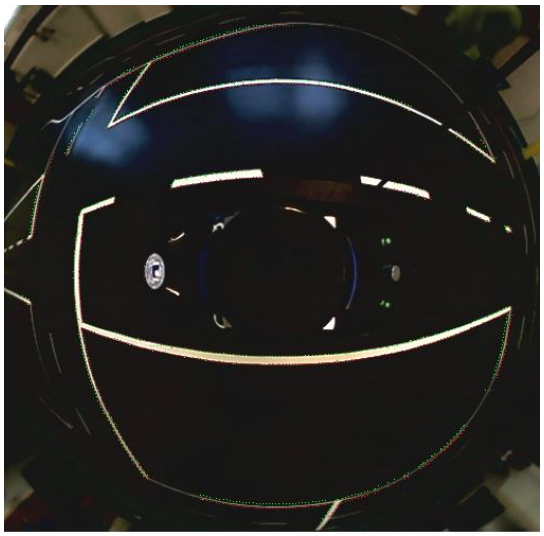

Abbildung 7.1: Kreuzung mit markierten Kanten

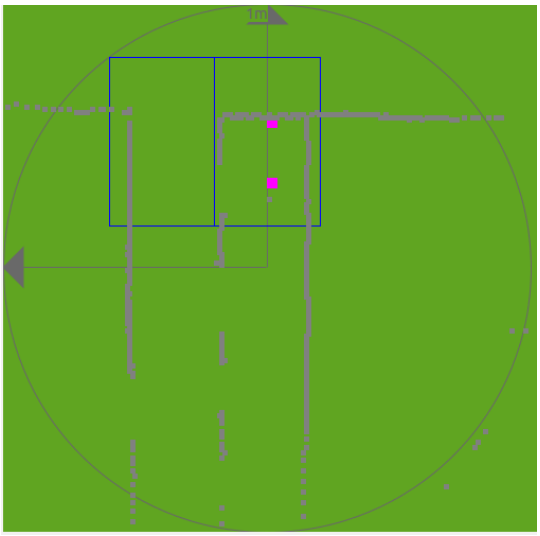

Abbildung 7.2: Relative Sicht mit ROI (blau) und erkannter Kreuzung (Lotfußpunkt und Wegpunkt, lila)

7.2 False Positives

Ein „False Positive“ ist in diesem Fall die Erkennung einer Kreuzung, obwohl keine vorhanden ist. Die Auswirkungen sind zwar nicht so schwerwiegend, wie wenn eine Kreuzung nicht erkannt wird, obwohl eine vorhanden ist, doch ein unnötiges Anhalten kostet Zeit. Die größte Schwachstelle des vorgestellten Ansatzes sind eben diese False Positives. Ist das Auto in einer so engen Kurve, dass die rechte ROI die Spurbegrenzung beinhaltet, kann es vorkommen, dass diese fälschlicherweise als Stopplinie erkannt wird und das Auto anhält. Um dieses Verhalten zu verhindern, muss der Winkelbereich, der bei der Hough-Transformation in Betracht gezogen wird, verkleinert werden. Hier muss ein guter Kompromiss gefunden werden. Wenn man den Bereich nämlich zu klein setzt, so würde eine Kreuzung nicht mehr erkannt werden, wenn man, z.B. aus einer Kurve heraus zu schräg auf sie zu fährt. False Positives können ebenfalls auftreten, wenn das Auto einem Hindernis ausweicht, da hier die seitlichen Spurbegrenzungen manchmal ebenfalls in einem ungünstigen Winkel zum Auto stehen. Die Fehleinschätzung kann hier allerdings verhindert werden, wenn die Hinderniserkennung höher priorisiert wird als die Kreuzungserkennung.

7.3 Mögliche Verbesserungen

Da es in Kurven zu False Positives kommen kann, wäre hier ein Ansatzpunkt für Verbesserungen. Das Problem entsteht, da die ROI immer statisch vor dem Auto platziert ist. Sie sollte allerdings immer auf dem rechten bzw. linken Fahrstreifen sein. Man müsste folglich in Kurven die ROIs entsprechend anders platzieren. Dies wäre möglich, indem der Lenkwinkel des Autos ausgelesen wird und anhand dessen die ROI mit einer Koordinatentransformation auf die Fahrbahn gebracht wird. Eine andere vorübergehende Lösung wäre es, die ROI zu verkürzen. Da der Anhaltevorgang erst gestartet wird, wenn das Auto nahe genug an der Stopplinie ist, kann die ROI auf eben diese Distanz abgestimmt sein. Dadurch wird gleichzeitig erreicht, dass die ROI erst bei schärferen Kurven die seitlichen Spurbegrenzungen bedeckt und damit seltener die Gefahr eines False Positives besteht.

Ein weiterer Ansatzpunkt wäre, die Performanz des Algorithmus zu verbessern. Zurzeit benötigt er auf dem Testträger zwischen 6 und 13 ms. Dieser Wert ist nicht optimal, da alle ausgeführten Prozesse die Gesamtzeit von 33 ms pro Durchlauf nicht überschreiten sollten, damit die Bildaktualisierungsrate der Kamera voll ausgenutzt werden kann. Die Verbesserung der Performanz könnte erreicht werden, indem die Hough-Transformation durch das effizientere RANSAC Verfahren [15] ersetzt werden würde. Eine andere oder

auch zusätzliche Möglichkeit wäre das Verfahren zu parallelisieren. Da die Transformation auf n unterschiedlichen Scanlinien stattfindet, kann die Berechnung auch auf bis zu n Prozesse aufgeteilt werden. Hier müsste getestet werden welche Anzahl die größte Leistungssteigerung bringen würde.

Literaturverzeichnis

[1] F. G. Heide und C. Kerkmann, „Ein feinfühliger Chauffeur namens Computer," Handelsblatt, 2014. [Online]. Available: http://www.handelsblatt.com/technologie/vernetzt/autonome-autos-ein-feinfuehliger-chauffeur-namens-computer/10263146.html.

[2] „Selbstfahrende Autos: Google baut ein eigenes Auto," heise online, [Online]. Available: http://heise.de/-2199035.

[3] M. Söldner, „100 selbstfahrende Autos von Volvo kurven durch Göteborg," PC WELT, 2014. [Online]. Available: http://www.pcwelt.de/news/100_selbstfahrende_Autos_von_Volvo_kurven_durch_Goeteborg-Autonom-8691067.html.

[4] IEEE Transportation Electrification, 2014. [Online]. Available: http://electricvehicle.ieee.org/telematics/ieee-survey-reveals-mass-produced-cars-will-steering-wheels-gasbrake-pedals-horns-rearview-mirrors-2035/.

[5] Technische Universität Braunschweig, „Carolo-Cup Regelwerk 2015," 3 Juni 2014. [Online]. Available: https://wiki.ifr.ing.tu-bs.de/carolocup/system/files/Hauptwettbewerb2015.pdf. [Zugriff am 12 September 2014].

[6] J. F. Boldt und S. Junker, „Evaluation von Methoden zur Umfelderkennung mit Hilfe omnidirektionaler Kameras am Beispiel eines Modellfahrzeugs," Freie Universität Berlin, Institut für Informatik, 2012.

[7] J. M. S. Prewitt, „Object enhancement and extraction," in *Picture Processing and Psychopictorics*, Academic Press, 1970.

[8] D. Kuan, G. Phipps und A.-C. Hsueh, „A Real-Time Road Following and Road Junction Detection Vision System for Autonomous Vehicles.," in *Proceedings of the 5th National Conference on Artificial Intelligence,* 1986.

[9] M. Ekinci und B. T. Thomas, „Road Junction Recognition and Turn-offs for Autonomous Road Vehicle Navigation," in *Proceedings of the ICPR,* 1996.

[10] D. Pomerleau, „Ralph: "Rapidly adapting lateral position handler."," in *Proceedings of the IEEE Symposium on Intelligent Vehicles,* 1995.

[11] J. Crisman und C. Thorpe, „Color Vision for Road Following," in *Proceedings of SPIE Conference on Mobile Robots,* 1988.

[12] M. Aly, „Real time detection of lane markers in urban streets," in *IEEE Intelligent Vehicles Symposium*.

[13] V. Voisin, M. Avila, B. Emile, S. Begot und J.-C. Bardet, „Road Markings Detection and Tracking Using Hough Transform and Kalman Filter," in *Proceedings of the 7th international conference on ACIVS,* 2005.

[14] R. C. Gonzales und R. E. Woods, Digital Image Processing, New Jersey: Prentice Hall, 2002.

[15] M. A. Fischler und R. C. Bolles, „Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM,* pp. 381-395, 1981.

Eigenständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Bachelorarbeit selbstständig und ohne unerlaubte Hilfe angefertigt habe. Alle verwendeten Hilfsmittel und Quellen sind vollständig angegeben und die aus den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht.

Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

_____

Benjamin Zengin

Berlin, 18.09.2014