INSTITUT FÜR INFORMATIK

BACHELORARBEIT

Fahrspurerkennung mit RANSAC bei autonomen Modellfahrzeugen

von Conrad Läßig

Betreuer:
Prof. Dr. Raúl Rojas

9. Mai 2015

Inhaltsverzeichnis

Zusammenfassung

Autonome Fahrzeuge sind Robotersysteme, die eine Vielzahl von komplexen Aufgaben bewältigen müssen, um selbstständig am Straßenverkehr teilnehmen zu können. Eine davon ist das Erkennen und Verfolgen von Fahrspuren. Diese Arbeit beschreibt die Implementierung einer Fahrspurerkennung für ein autonomes Modellfahrzeug mit einer omnidirektionalen Kamera, welches im jährlich stattfindenden Carolo-Cup-Wettbewerb zum Einsatz kommt und dort auf einer simulierten Landstraße verschiedene Disziplinen bestehen muss. Der in der Arbeit vorgestellte Ansatz basiert auf einer schon vorhandenen Kantenerkennung. Es wird beschrieben, wie mithilfe des RANSAC-Algorithmus und der newtonschen Polynominterpolation aus den Kanten der Spurmarkierungen Polynome 2. Grades modelliert werden und wie aus diesen ein Modell der rechten Fahrspurmitte gewonnen wird. Des weiteren wird gezeigt, wie dieses Modell unter Verwendung der vorhandenen Regel-Architektur in das Fahrverhalten integriert wird.

1 Einleitung

Die Erforschung und Entwicklung mobiler Robotersysteme ist ein rasant wachsender Bereich. Besonders interessant ist dabei die Konstruktion "autonomer", also selbstständig fahrender Fahrzeuge. Viele Unfälle, die auf menschliches Versagen zurück zu führen sind, könnten so in Zukunft vermieden werden und der Komfort im Straßenverkehr würde sich deutlich erhöhen. Mittlerweile gibt es eine ganze Reihe von Projekten an Universitäten, wie zum Beispiel an der FU Berlin das Forschungsprojekt *AutoNOMOS Labs* oder auch bei namhaften Unternehmen wie Volvo (siehe Söldner, 2014) oder Google (siehe Markoff, 2010), in denen Prototypen solcher Fahrzeuge entwickelt und getestet werden.

Autonome Fahrzeuge müssen viele komplexe Aufgaben wie die Lokalisierung in der Umgebung, das Erkennen von Hindernissen oder das Beachten von Verkehrsregeln bewältigen. Eine grundlegende Fähigkeit ist die sichere, selbstständige Fahrt innerhalb vorhandener Fahrspuren. In der vorliegenden Arbeit wird ein Ansatz für die dafür notwendige Fahrspurerkennung vorgestellt.

1.1 Carolo-Cup

Der Carolo-Cup ist ein von der Technischen Universität Braunschweig seit 2008 jährlich veranstalteter Wettbewerb, der den teilnehmenden studentischen Teams die Möglichkeit bietet, sich mit den Herausforderungen des autonomen Fahrens anhand von Modellautos auseinander zu setzen. Jedes Team muss ein autonomes Modellauto im Maßstab 1:10 konstruieren, welches auf einer an die Realität angenäherten Modellstrecke folgende drei Disziplinen meistern muss:

- **Rundstrecke ohne Hindernisse:** Das Auto soll innerhalb einer festen Zeit so weit wie möglich die Strecke abfahren, ohne dabei die rechte Fahrspur zu verlassen. Es können auf einer Länge von bis zu 1 m Spurmarkierungen fehlen. Eine der drei Markierungen muss jedoch immer sichtbar bleiben. An Kreuzungen muss nicht gehalten werden. Kommt das Auto von der Spur ab, darf es gegen eine Zeitstrafe mit einer Fernbedienung langsam wieder auf die Strecke zurück gesteuert werden.

- **Rundstrecke mit Hindernissen:** Genauso wie oben muss hier soweit wie

möglich auf der Strecke gefahren werden. Erschwerend kommt nun aber hinzu, dass sich statische oder dynamische Hindernisse in Form von weißen Kartons auf der Strecke befinden, die unter korrekter Verwendung des Blinkers mit Spurwechseln überholt werden müssen. An Kreuzungen muss bei durchgezogener Linie gehalten und ggf. einem vorbeifahrenden Hindernis Vorfahrt gewährt werden.

- **Paralleles Einparken:** Hier gibt es rechts von der Strecke einen Parkstreifen, auf dem zwischen weißen Kartons Parklücken unterschiedlicher Größe vorhanden sind. Das Auto muss so schnell wie möglich eine ausreichend große Parklücke finden, möglichst ohne Kollisionen seitlich einparken und parallel zum Stehen kommen. Der Einsatz der Fernbedienung ist hier untersagt.

In die Wertung geht außerdem eine Präsentation ein, in der das Team das Auto und die entwickelten Konzepte vorstellt. Hier kommt es auch auf die Wirtschaftlichkeit und Energieeffizienz der Konstruktion an. Weitere Details zu den Regeln finden sich im Regelwerk (Technische Universität Braunschweig, 2014).

1.2 Zielstellung

Das Ziel dieser Arbeit ist, im Kontext des Carolo-Cup-Wettbewerbs ein Softwaremodul zu entwickeln, mit dem die vor dem Auto liegende Fahrspur erkannt wird. Dafür werden die Spurmarkierungen genutzt, die auf den Bildern der auf dem Auto montierten Kamera zu sehen sind. Außerdem soll das Auto in die Lage versetzt werden, der modellierten Fahrspur zu folgen.

1.3 Struktur der Arbeit

Bis hier wurden die Rahmenbedingungen der Arbeit vorgestellt. Den Schluss dieses Kapitels bildet eine Beschreibung der Entwicklungsumgebung, in der die Hardware des Modellautos und die vorhandene Software vorgestellt werden. Im zweiten Kapitel werden einige verwandte Arbeiten mit ähnlicher Zielstellung angeführt und bewertet. Außerdem werden einige Techniken vorgestellt, die in dieser Arbeit verwendet werden und häufig mit dem Thema Spurerkennung im Zusammenhang stehen. Das dritte Kapitel bildet den Hauptteil der Arbeit. Hier wird der verwendete

Ansatz beschrieben. Es wird dafür zuerst erklärt, welche Daten als Grundlage für die Spurerkennung verwendet werden und wie sie gewonnen werden. Dann wird die Implementierung der Spurerkennung im Detail erläutert. Im vierten Kapitel wird dargestellt, wie die erkannte Fahrspur für das Fahrverhalten verwendet wird. Dafür wird kurz beschrieben, wie das Modellauto bisher fährt und dann die Integration der Fahrspurverfolgung in das bestehende Softwaresystem dargelegt. Im letzten Kapitel werden die Ergebnisse der Arbeit hinsichtlich der Zielstellung diskutiert.

1.4 Die Entwicklungsumgebung

Als ich im Oktober 2014 Mitglied im "Berlin United Racing Team"der Freien Universität Berlin wurde, war bereits ein funktionstüchtiges Modellauto vorhanden, welches in früheren Carolo-Cup-Wettbewerben bereits zum Einsatz gekommen war und als Testfahrzeug für die Arbeit zur Verfügung stand. An einem zweiten Auto wurden gerade eine neue Stereokamera und eine angepasste Stromversorgungsplatine getestet. Auch stand bereits Software zur Verfügung, auf der aufgebaut werden konnte. Die Hard- und Software werden in den folgenden Abschnitten kurz beschrieben.

1.4.1 Hardware

Eine detailierte Beschreibung der Hardware findet sich in Boldt und Junker (2012), deswegen folgt hier nur eine Zusammenfassung.

Als Basis des Testfahrzeugs dient das Chassis *Xray T3 Spec. 2012*. Auf diesem befindet sich als zentrale Komponente der Einplatinencomputer *ODROID-X2* mit einem 1,7 GHz *ARM Cortex-A9* Quad Core Prozessor, 2 GB RAM und verschiedenen Schnittstellen wie USB 2.0, 100 MBit/s oder UART. Auf ihm werden sämtliche kognitiven Prozesse ausgeführt. Ein USB-WLAN-Adapter und der Ethernet-Port werden für das Aufspielen der Software und Debugging verwendet.

Über die serielle UART-Schnittstelle ist an den Hauptrechner das vom Team selbst entwickelte *OttoBoard V2* angeschlossen. Auf diesem befindet sich ein *STM32 F4* Mikrocontroller mit einem 168 MHz *ARM Cortex-M4F* Prozessor. Das *OttoBoard V2* dient zur Ansteuerung und Kontrolle der im Folgenden beschriebenen Hardwarekomponenten.

Das Auto wird mit einem sehr präzisen, bürstenlosen Elektromotor angetrieben, der zusammen mit dem verbauten Getriebe Geschwindigkeiten von bis zu 4 m/s erreicht. Der Motor wird über einen eigenen Motorcontroller gesteuert, mit dem sich auch die aktuelle Geschwindigkeit auslesen lässt.

Gesteuert wird das Auto mithilfe eines Lenkservos an der Vorderachse.

Im *OttoBoard* ist ein Inertialsensor (IMU) integriert, der ein Accelerometer, ein Magnetometer und ein Gyroskop enthält, mit denen Winkeländerung, Beschleunigung und Ausrichtung des Fahrzeugs bestimmt werden können.

Weiterhin ist ein 40 MHz Empfänger verbaut, über den das Auto ggf. auch mit einer Fernbedienung gesteuert werden kann.

Am Auto sind Bremslichter und Blinker vorhanden und eine blaue LED, die das Benutzen der Fernbedienung anzeigt. So können die im Carolo-Cup-Regelwerk geforderten Lichtsignale gegeben werden.

Hinten am Auto befindet sich ein Bedienpanel mit einem Display, einem Drehknopf und zwei Auswahlknöpfen, mithilfe dessen die Programme für die verschiedenen Disziplinen gestartet und Hardwarekomponenten kalibriert werden können. Neu hinzugekommen sind drei weitere im Regelwerk geforderte Knöpfe, mit denen die Wettbewerbsjury die Programme komfortabler starten kann.

Die Stromversorgung wird über ein selbst entwickeltes Powerboard aus einem leicht wechselbaren Lithium-Polymer-Akku gewährleistet. Es gibt einen Hauptschalter und aus Sicherheitsgründen einen separaten Schalter für den Antrieb. Das Board zeigt mit einer RGB-LED den Akkustand an und trennt den Akku ggf. von allen Verbrauchern, um Schäden durch Tiefenentladung zu verhindern.

Auf einem Mast ist über dem Auto eine Webcam angebracht, die über USB mit dem Hauptrechner verbunden ist. Eine Besonderheit ist, dass die Kamera von unten auf einen Parabolspiegel gereichtet ist. Über diesen wird eine 360°-Sicht um das Auto ermöglicht.

Die Abbildungen 1 und 2 zeigen das Modellauto in verschiedenen Zuständen.

1.4.2 Software

Auf dem Hauptrechner ist ein ARM-Linux installiert, was den Vorteil hat, dass viele Standardbibliotheken unkompliziert genutzt werden können. Die Software für den Carolo-Cup wird auf den Rechnern der Teammitglieder entwickelt, mit GCC

Abbildung 1: Das Modellauto mit der omnidirektionalen Kamera

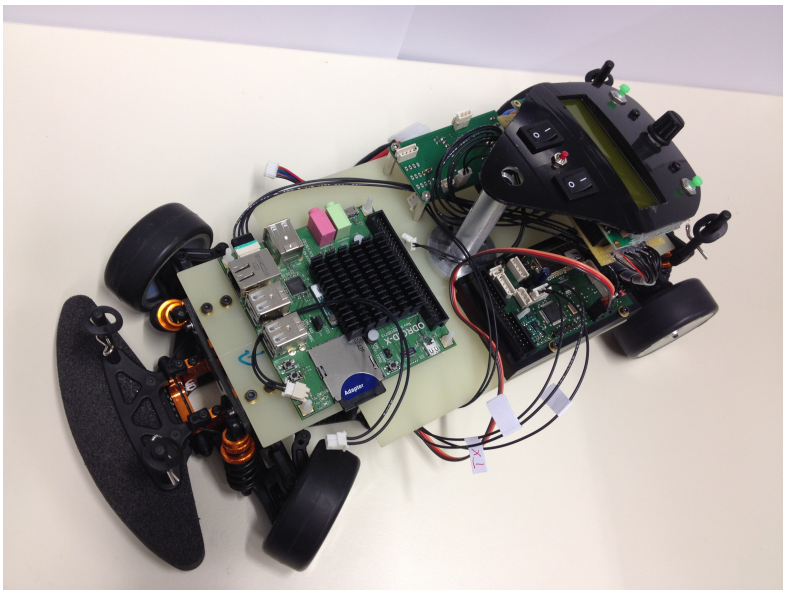

Abbildung 2: Das Chassis mit montierten Hardwarekomponenten

für ARM cross-kompiliert und per SSH auf das Auto übertragen. Sie ist in C++ geschrieben.

Auf dem Mikrocontroller läuft in C geschriebene Software, die Schnittstellen zu den meisten Hardwarekomponenten bietet, auf die der Hauptrechner zugreifen kann. Außerdem werden bei Nutzung der Fernbedienung die Steuerkommandos des Hauptrechners überschrieben.

Die Carolo-Cup-Software wird mithilfe des *BerlinUnited* Frameworks entwickelt, welches ursprünglich in einer Zusammenarbeit zwischen der *Humboldt-Universität Berlin* und der *Freien Universität Berlin* für die Robo-Cup Fußballroboter entstand.

Das Framework basiert auf einer Blackboard-Architektur, in der der Quellcode in Module und Repräsentationen aufgeteilt ist. In den Repräsentationen werden Daten gespeichert und zur Verfügung gestellt. Die Module verweisen auf Repräsentationen, die sie benötigen oder anbieten und dienen der Verarbeitung der Daten. Basierend auf den Abhängigkeiten zwischen Repräsentationen und Modulen wird eine Ausführungskette berechnet, die dann immer wieder sequentiell durchlaufen wird. Die Ausführung findet syncronisiert zur Kamerabildrate von 30 Hz statt, sodass jedes Modul nach 1/30 s neu ausgeführt wird und alle Berechnungen innerhalb dieser Zeit beendet sein müssen. Dabei wird das Sense-Plan-Act-Paradigma verfolgt:

1. Sense: Die Daten der Sensoren und der Kamera werden eingeholt.

2. Plan: Die Daten werden ausgewertet und aus den gewonnenen Informationen Modelle generiert, wie z.B. im Bild sichtbare Kanten oder die aktuelle Position auf der Strecke. Auf Basis dieser Modelle wird das Verhalten geplant.

3. Act: Das geplante Verhalten wird durch Ansteuerung der entsprechenden Hardwarekomponenten umgesetzt.

2 Forschungsstand

Es gibt bereits viele auf Kamerabildern basierende Algorithmen zur Fahrspurerkennung. Diese können die Eigenschaften der sichtbaren Spuren, wie beispielsweise

Gradient oder Farbe, nutzen oder auf dem Erzeugen von Modellen basieren. Da in der vorhandenen Software schon modellbasierte Algorithmen benutzt werden und die nötigen Grundlagen gelegt sind, wird auch in dieser Arbeit ein modellbasierter Ansatz gewählt. Es finden sich kaum Arbeiten, die Bilder einer vergleichbaren omnidirektionalen Kamera nutzen. Viele Arbeiten, die meist andere Kamerasysteme nutzen und im realen Straßenverkehr Anwendung finden, bieten dennoch Anhaltspunkte für Entwicklung einer geeigneten Herangehensweise für das der Arbeit zugrunde liegende Szenario des Carolo-Cups. Einige wenige werden im Folgenden kurz vorgestellt.

2.1 Verwandte Arbeiten

Deng und Han (2013) zeigen einen Ansatz, bei dem die Bilder der schräg nach unten auf die Straße gerichteten Kamera zuerst entzerrt werden und danach eine Kantenerkennung mittels eines Gauß-Filters durchgeführt wird. Eine solche Entzerrung wäre mit der vorhandenen Hardware zu rechenaufwändig. Genau so verhält es sich mit der nach dem Entzerren angewandten Hough-Transformation zum Eingrenzen der Spurmarkierungen. Für diese Arbeit werden stattdessen weniger aufwändige, statische Begrenzungen gewählt. Interessant ist jedoch die vorgestellte Anpassung des RANSAC-Algorithmus, bei der die Bereiche, in denen Modelle gesucht werden, weiter unterteilt werden, um schneller zu guten Modellen zu gelangen. Die Anwendung einer solchen Anpassung ist auch für die vorliegende Arbeit sinnvoll. Jedoch wird die Nutzung von Polynomen als Modell wegen ihrer einfacheren Handhabung der Nutzung von B-Splines vorgezogen, wobei letztere sehr häufig als Modell verwendet werden, wie z.B. auch in Wang, Shen et al. (2000) und in Aly (2008).

Ein etwas anderer Ansatz wird von Tan et al. (2014) verfolgt. Hier wird das Sichtfeld in nah und fern unterteilt. Im direkt vor dem Fahrzeug liegenden Nahbereich werden mit einer Hough-Transformation die noch gerade verlaufenden Spurmarkierungsabschnitte als Geraden modelliert. In der Ferne werden die Datenpunkte der Spuren durch eine angepasste Variante der von Lim et al. (2012) vorgestellten *Riverflow*-Methode gewonnen und daraus unter Verwendung des RANSAC-Algorithmus Hyperbeln modelliert. Da in der vorliegenden Arbeit durch die ver-

wendete Kamera die Sichtweite sehr beschränkt ist, lohnt sich das Unterteilen des Sichtbereichs und die Anwendung dieser Methode nicht.

Lipski et al. (2008) stellen einen Ansatz vor, bei dem die vor dem Fahrzeug liegende Strecke aus kurzen, aufeinander folgenden Segmenten modelliert wird. Die Segmente sind in Bereiche für die Erkennung der einzelnen Spurmarkierungen unterteilt. Die Informationen des vorherigen Segments werden zur Erstellung des nächsten genutzt. Um die Krümmung in Kurven zu modellieren, wird ein Segment im Gesamten gedreht. Die erreichte Performance ist mit den angegebenen 10 Bildern pro Sekunde für diese Arbeit zu gering, wobei sie wahrscheinlich auf die relativ aufwendige Vorverarbeitung der Bilder zur Erkennung von Spurmarkierungspunkten zurückzuführen ist. Die Idee, die Erkennungsbereiche dynamisch entlang der Spurmarkierungen auszurichten, ist für die Arbeit jedoch sehr interessant.

2.2 RANSAC

Ergebnisse von empirischen Messungen liegen meist in Form von Datenpunkten vor. Es soll ein Modell gefunden werden, welches an diese möglichst genau angepasst ist. Durch die Verwendung digitaler Messtechnik, wie sie im Bereich des maschinellen Sehens und der Robotik zum Einsatz kommt, erhält man sehr große Datensätze mit vielen Ausreißern. Dies birgt Schwierigkeiten für das Finden der bestmöglichen Modellparameter. Mit klassischen Verfahren wie der Methode der kleinsten Quadrate können nur kleine Fehler ausgeglichen werden, größere Fehler können jedoch zu einer Fehlklassifikation der Daten führen, aus der falsche Parameter resultieren (siehe Fischler und Bolles, 1981). Eine Möglichkeit, dem zu entgegnen, ist die Verwendung des von Fischler und Bolles (1981) vorgestellten RANdom SAmple Consensus Verfahrens (RANSAC), welches gegenüber großen Ausreißern resistent ist. In diesem wird ein iterativer Ansatz verfolgt, bei dem, statt alle Datenpunkte gemeinsam auszugleichen, in mehreren Schritten zufällig Punktmengen gewählt werden, die eine für die Berechnung der Modellparameter benötigte Mindestgröße haben müssen. Der prinzipielle Ablauf ist in Algorithmus 1 dargestellt.

Der Algorithmus besitzt vier freie Parameter:

- die maximale Anzahl an Iterationen k

Algorithmus 1 : RANSAC Algorithmus (aus Fischler und Bolles, 1981)

Data : Punkte P, an die das Modell angepasst werden soll
Result : Modell M^*, falls eines gefunden wird
while *maximale Anzahl von Iterationen k noch nicht erreicht* **do**
    $S \longleftarrow$ n zufällig gewählte Punkte aus P;
    Berechne ein Modell M aus S;
    $S^* \longleftarrow$ alle Punkte aus P, deren Abstand zu M kleiner als der Schwellwert m ist; `// Consensus Set`
    if *S^* hat genug Punkte (Schwellwert t)* **then**
        Berechne an S^* angepasstes Modell M^*;
    end
end

- die Anzahl an Punkten n, mit der das Anfangsmodell M gebildet wird
- der Schwellwert m, der die Distanz bestimmt, innerhalb derer ein Punkt p liegen muss, um das Modell M zu stützen
- der Schwellwert t, der die Anzahl von Punkten bestimmt, die das gefundene Anfangsmodell stützen müssen, damit ein gültiges Ergebnis erreicht wird. Diese Punktmenge wird *Consensus Set* genannt.

Die Anzahl n muss mindestens so groß sein, dass die Parameter des gewählten Modells eindeutig bestimmbar sind. Verfahren wie die lineare Regression können auch bei größeren Punktmengen angewandt werden, so dass auch eine höhere Anzahl gewählt werden kann.

Der Schwellwert m sollte so gewählt werden, dass ausreichend viele Ausreißer als solche erkannt und somit keine falschen Modelle gewählt werden. In den meisten Fällen empfiehlt es sich, diesen Wert empirisch zu bestimmen.

Der Schwellwert t bietet die Möglichkeit, den Algorithmus statt mit einem schlechten Ergebnis ohne Ergebnis terminieren zu lassen. Er sollte so groß sein, dass sichergestellt ist, dass ein korrektes Modell gefunden wurde. Oft wird t so gewählt, dass die Größe des *Consensus Sets* der Gesamtzahl der Datenpunkte abzüglich der Zahl vermuteter Ausreißer entspricht (siehe Hartley und Zisserman, 2004: 120).

Anzahl Punkte	Anteil an Ausreißern ϵ						
n	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Tabelle 1: Anzahl der nötigen Iterationen k, damit mit einer Wahrscheinlichkeit $p = 0.99$ wenigstens in einer Iteration kein Ausreißer unter den n gewählten Punkten ist, bei einer Wahrscheinlichkeit von ϵ Ausreißern (aus Hartley und Zisserman, 2004)

Die Anzahl an Iterationen k kann stochastisch bestimmt werden. Der Wert wird so hoch angesetzt, dass mit einer Wahrscheinlichkeit p sichergestellt ist, dass in wenigstens einer Iteration eine zufällige Auswahl von n Startmodellpunkten getroffen wird, in der keine Ausreißer enthalten ist. Angenommen w ist die Wahrscheinlichkeit, dass ein zufällig gewählter Punkt zu den Punkten gehört, die das Modell stützen und damit $\epsilon = 1 - w$ die Wahrscheinlichkeit, dass es sich um einen Ausreißer handelt. Dann sind nach Hartley und Zisserman (2004: 119) k Iterationen von je n Punkten nötig, wobei $(1 - w^n)^k = 1 - p$ und damit

$$k = \frac{log(1-p)}{log(1-(1-\epsilon)^n)}. \tag{1}$$

Tabelle 1 führt Beispielwerte für k bei $p = 0.99$ für gegebene n und ϵ an.

Es gibt verschiedene Erweiterungen des RANSAC-Algorithmus, die in dieser Arbeit jedoch nicht zum Einsatz kommen. Der MLESAC-Algorithmus beispielsweise kombiniert RANSAC mit einem Maximum Likelihood Schätzer (siehe Torr und Zisserman, 2000). Der LO-RANSAC-Algorithmus erweitert den ursprünglichen Algorithmus um einen zusätzlichen Schritt, bei dem das *Consensus set* weiter optimiert wird (siehe Chum et al. 2003).

2.3 Polynominterpolation

Das in dieser Arbeit für die Spurmarkierungen und die daraus abgeleitete Fahrspurmitte gewählte Modell ist ein Polynom 2. Grades. Um aus gewonnenen Datenpunkten ein Polynom zu erstellen, kann eine Polynominterpolation durchgeführt werden. Dabei wird für $n+1$ gegebene Wertpaare (x_i, f_i mit paarweise verschiedenen Stützstellen x_i ein Polynom P maximal vom Grad n gesucht, das für $i = 0, ..., n$ die Gleichungen $P(x_i) = f_i$ erfüllt. So ein Polynom existiert immer und ist eindeutig bestimmt (vgl. Freund und Hoppe, 2007: 39).

Für dieses Problem existieren verschiedene Lösungsansätze wie die Lagrangesche Interpolationsformel oder der Algorithmus von Neville-Aitken. Ein vergleichsweise effizientes und damit in der Numerik häufiger genutztes Verfahren ist der Newtonsche Algorithmus (vgl. Freund und Hoppe, 2007: 43f.). In diesem wird das gesuchte Interpolationspolynom mithilfe der Newton-Basisfunktionen $N_0(x) = 1$ und $N_i(x) = \Pi_{j=0}^{i-1}(x - x_j) = (x - x_0) \cdots (x - x_{i-1})$ mit $i = 1, \ldots, n$ dargestellt. Es ergibt sich aus der Newtonschen Interpolationsformel

$$P(x) = \sum_{i=0}^{n} c_i \cdot N_i(x) \tag{2}$$

$$= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0) \cdots (x - x_{n-1}) \ . \tag{3}$$

Die Koeffizienten können nun mit dem Gleichungssystem der Gleichungen $P(x_i) = f_i$

$$\begin{pmatrix} 1 & & & 0 \\ 1 & (x_1 - x_0) & & \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & \\ \vdots & \vdots & & \ddots \\ 1 & (x_n - x_0) & \cdots & \Pi_{i=0}^{n-1}(x_n - x_i) \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} f_0 \\ \vdots \\ f_n \end{pmatrix} \tag{4}$$

bestimmt werden. Dies würde n Divisionen und $n(n-1)$ Multiplikationen kosten.

Mit nur $n(n+1)/2$ Divisionen ist es jedoch effektiver, die Koeffizienten mit dem Schema der dividierten Differenzen zu berechnen. Die Koeffizienten c_i sind durch

die dividierten Differenzen definiert als

$$c_i = f\left[x_0, \ldots, x_i\right], \tag{5}$$

wobei die dividierten Differenzen für $i < j$ rekursiv definiert sind durch

$$f\left[x_i\right] = f_i = f(x_i) \text{ (bekannte Funktionswerte an den Stützstellen)} \tag{6}$$

$$f\left[x_i, \ldots, x_j\right] = \frac{f\left[x_{i+1}, \ldots, x_j\right] - f\left[x_i, \ldots, x_{j-i}\right]}{x_j - x_i} \tag{7}$$

(vgl. Freund und Hoppe, 2007: 44).

Zu beachten ist, dass ein interpoliertes Polynom nur innerhalb des Bereiches der verwendeten Stützstellen eine ausreichend gute Näherung an die Daten bietet. Außerhalb wächst der Approximationsfehler schnell an und das Polynom ist dort nicht zu verwenden.

3 Fahrspurerkennung

Die im Carolo-Cup verwendete Strecke ist zwar einer realen Landstraße nachempfunden, in ihrer Gestaltung aber sehr schlicht gehalten. Es gibt lediglich auf ein dunkles, ebenes Bodenmaterial weiß aufgezeichnete Spurmarkierungen, die Geraden, scharfe Kurven und Kreuzungen nachbilden und die einzigen Informationen für die Modellierung der Fahrspur liefern.

Die Straße hat im Regelwerk klar definierte Abmessungen. Sie ist konstant 820 mm breit und von durchgezogenen Linien begrenzt. Die Straße wird durch eine Mittellinie in zwei Fahrstreifen aufgeteilt. Diese hat alle 200 mm eine 200 mm große Lücke. Die Markierungslinien sind ca. 20 mm breit. Zusätzlich gibt es eine über beide Fahrstreifen verlaufende Startlinie und an den Kreuzungen über einen Fahrstreifen verlaufende Haltelinien, die jeweils 40 mm breit sind. Wenn zwei Straßen parallel verlaufen, haben sie gemessen an der Außenkante der Markierungen einen Mindestabstand von 50 mm. Der minimale Innenradius von Kurven beträgt 1000 mm (vgl. Technische Universität Braunschweig, 2014: 15). Eine besondere Herausforderung für die Fahrspurerkennung besteht in den vor dem Wettbewerb entfernten Spurmarkierungen. Der Algorithmus sollte so robust sein, dass auch

Abbildung 3: Eine Kreuzung der Strecke

dann noch eine Fahrspur erkannt wird, wenn kurzzeitig nur eine der drei Spurmarkierungen vorhanden ist.

Während der Disziplin *Rundkurs mit Hindernissen* können sich statische und dynamische Hindernisse im Abstand von mindestens 1 m auf beiden Fahrspuren befinden. Sie bestehen aus weißen Kartons mit einer Höhe von bis zu 24 cm und einer Breite von bis zu 40 cm und können die Spurmarkierungen daher ebenso verdecken, was die Erkennung weiter erschwert.

In den Abbildungen 3 und 4 sind Ausschnitte der Strecke zu sehen und in Abbildung 5 wird eine mögliche Wettbewerbsstrecke gezeigt. Die Abbildungen sind dem Regelwerk entnommen (siehe Technische Universität Braunschweig, 2014).

3.1 Kantenerkennung

Das *Berlin United* Team nimmt bereits seit 2012 am Carolo-Cup teil und für die vorangegangenen Wettbewerbe wurden schon viele Funktionalitäten implementiert, von denen einige Grundlagen für diese Arbeit darstellen. Im folgenden soll die schon vorhandene Kantenerkennung kurz erläutert werden.

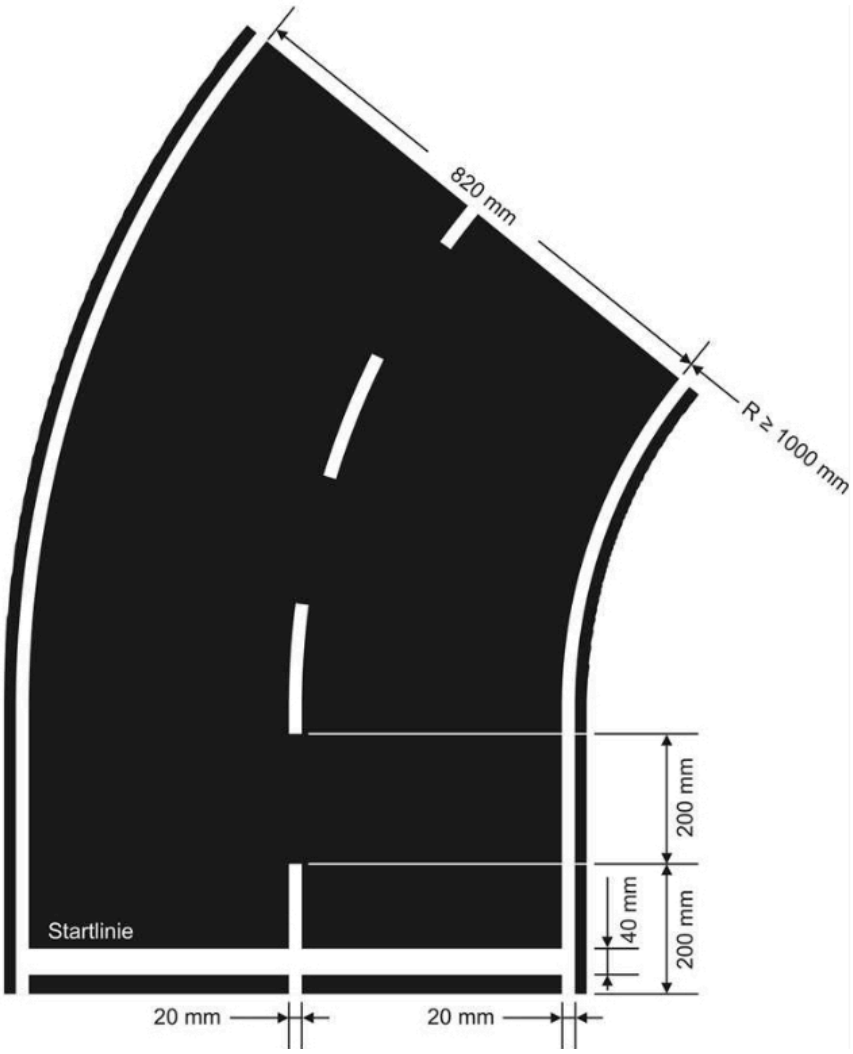

Abbildung 4: Eine Kurve mit Angaben zu den Streckenabmessungen

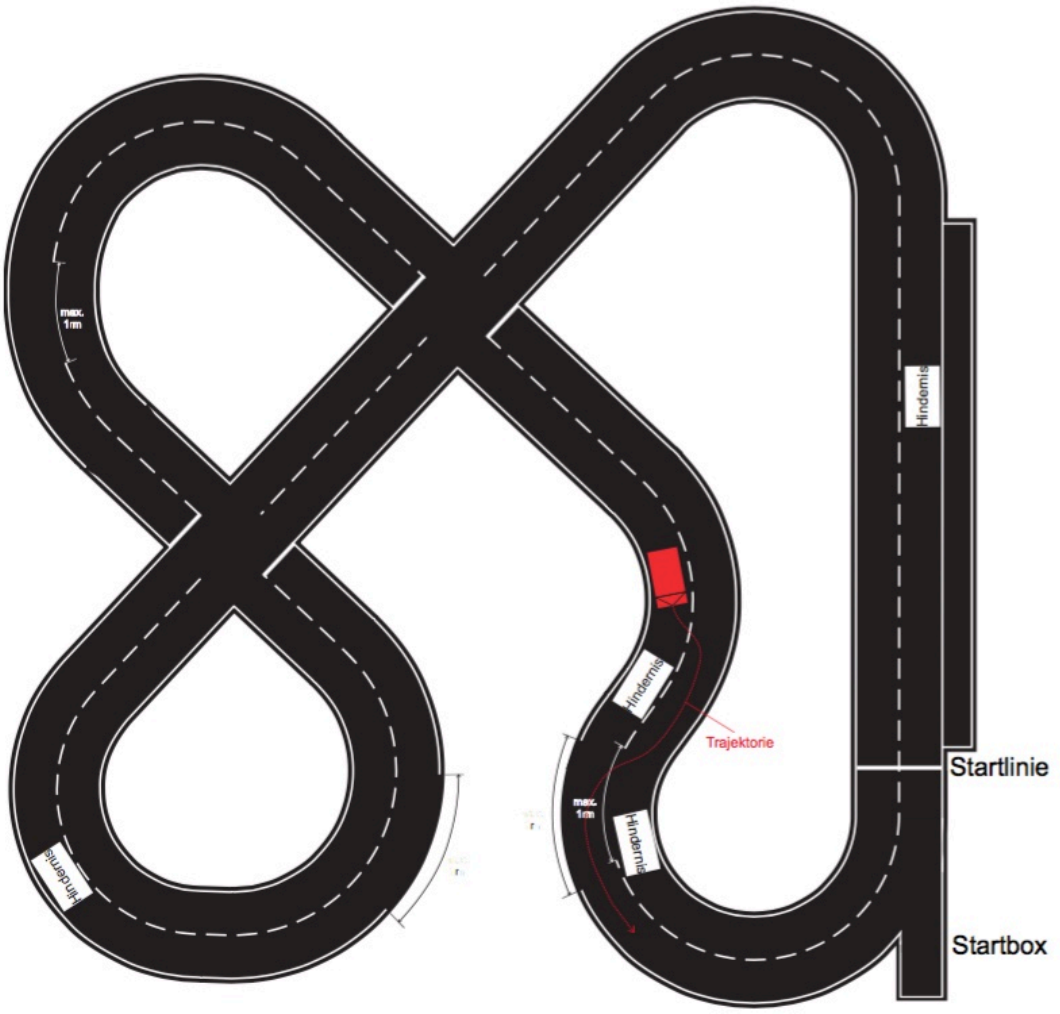

Abbildung 5: Ein möglicher Verlauf der Wettbewerbsstrecke mit Hindernissen und Fehlstellen

Als Eingabe erhält die Kantenerkennung Bilder von der omnidirektionalen Kamera auf dem Auto mit einer Rate von 30 Bildern pro Sekunde. Auf den Bildern sind die Spurmarkierungen rund um das Auto zu sehen. Ein naiver Ansatz, diese Markierungen zu erkennen, wäre es, über alle Pixel im Bild zu laufen und die Regionen mit weißen Pixeln zu finden. Dies wäre jedoch rechenaufwändig und die Flächen der Hindernisse, durch das leicht spiegelnde Bodenmaterial auftretende Reflexionen von Lichtquellen und sonstiges Rauschen müssten umständlich heraus gefiltert werden. Deswegen wird ein kanten-basierter Ansatz gewählt. In der vorliegenden Implementierung werden als erstes die Kantenpixel (*Edgels*) von weißen Regionen gesucht. Dafür werden über das Bild horizontal und vertikal in kleinem Abstand Scanlinien gelegt. Entlang dieser Linien werden mit dem *Sobel-Operator* (siehe Sobel, 2015) die Gradienten der Helligkeitswerte berechnet und anschließend durch eine *non-maximum surpression* die Stellen mit den stärksten Helligkeitsübergängen ermittelt, sodass keine doppelten Kanten entstehen. Abbildung 6 zeigt die auf einem Kamerabild auf diese Weise erkannten Edgels. An die Lichtverhältnisse angepassten Einstellungen der Kamera sind eine Voraussetzung für eine gute Erkennung und ein minimales Rauschen.

Um später Fahrmanöver planen zu können, ist es erforderlich, dass die gesuchte Fahrspur in einem zum Auto relativen Koordinatensystem mit realen Distanzen modelliert wird, welches quasi einen Blick aus der Vogelperspektive auf das Auto und seine Umgebung erlaubt. Dafür müssen die Positionen bestimmter Pixel des durch den Parabolspiegel verzerrten Kamerabildes in dieses relative Koordinatensystem transformiert werden. Das relative Koordinatensystem ist so ausgerichtet, dass sein Ursprung sich in der Symmetrieachse des Parabolspiegels befindet. Die x-Achse erstreckt sich geradeaus vor das Auto und die y-Achse ist orthogonal dazu nach links gerichtet. Die Kameralinse ist zentral auf den Parabolspiegel gerichtet, sodass die mittig vertikal durch den Spiegel verlaufende Symmetrieachse eine Normale zur Kamerasensorfläche bildet. Dies hat zur Folge, dass alle Pixel, die auf dem verzerrten Kamerabild den gleichen Abstand haben, auch im relativen Koordinatensystem gleich weit voneinander entfernt liegen. In letzterem wird jedoch die Dichte der Edgels aufgrund der Entzerrung ausgehend vom Koordinatenursprung nach außen geringer. Die Transformation der Bildpunkte erfolgt über eine für den verwendeten Parabolspiegel spezifische Funktion, welche durch ein Polynom appro-

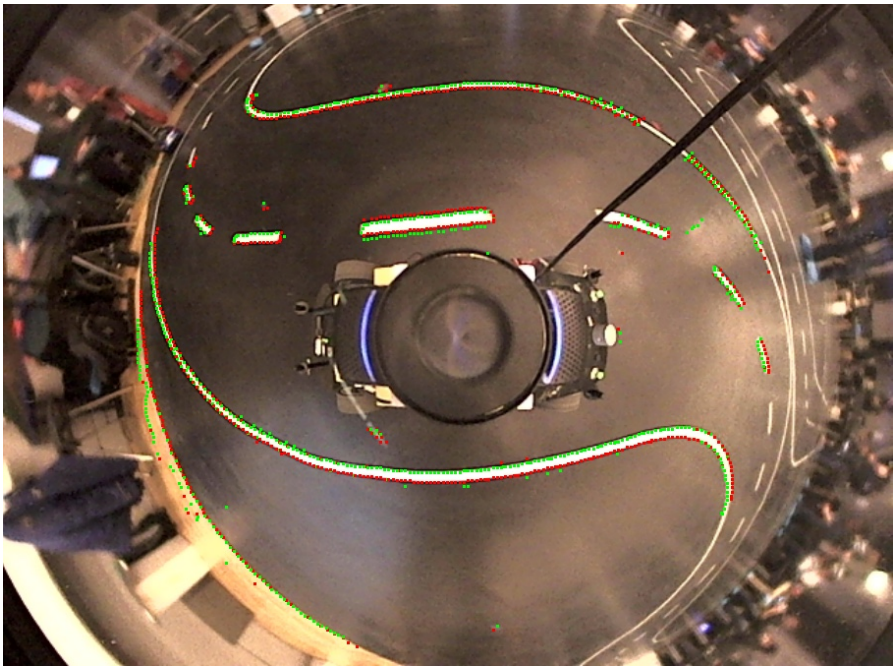

Abbildung 6: Ein Bild der omnidirektionalen Kamera mit in Rot und Grün eingezeichneten Positionen der Edgels

ximiert wird. Dieses wird im Vorfeld über eine lineare Regression aus mit einem Kalibrierungsbild ermittelten Abstandspaaren erzeugt.

Die Spurmarkierungen werden durch die Scanlinien so geschnitten, dass die an den beiden Markierungsrändern entstehenden Edgels ungefähr eine Spurmarkierungsbreite voneinander entfernt liegen. Von diesen Edgel-Paaren werden jene gewählt, deren Abstand zueinander nicht übermäßig groß ist und von diesen wird der jeweilige Mittelpunkt berechnet. Die so gewonnene Menge von Mittelpunkten von Spurmarkierungen (siehe Abbildung 7) bildet die Eingabe für die in dieser Arbeit betrachtete Fahrspurerkennung (vgl. Maischak, 2014: 29f.).

3.2 Implementierung

Hier folgt nun die Beschreibung des implementierten Algorithmus zur Fahrspurerkennung.

Der Ansatz besteht darin, dass aus den drei vor dem Auto liegenden Spurmarkierungen im relativen Koordinatensystem jeweils ein Polynom 2. Grades modelliert

Abbildung 7: Die aus den Edgels gewonnenen Mittelpunkte der Spurmarkierungen

wird. Aus diesen drei Polynomen wird ein weiteres Polynom abgeleitet, welches die Mitte der rechten Fahrspur darstellt und für die Implementierung des Fahrverhaltens weiterverarbeitet wird.

Eine für diesen Algorithmus getroffene Annahme ist, dass das Auto sich zu Beginn der Ausführung ungefähr mittig auf der rechten Fahrspur befindet und parallel zur Fahrtrichtung ausgerichtet ist, denn nur so sind die nächsten Schritte möglich. Da dies sowieso die für die Fahrt gewünschte Position ist, ist diese Annahme vertretbar.

3.2.1 Vorverarbeitung der Eingabedaten

Um die einzelnen Polynome aus der Gesamtmenge der während der Kantenerkennung gewonnenen Mittelpunkte zu modellieren, muss diese Menge zuerst aufgeteilt werden. Dafür werden im relativen Koordinatensystem zumindest für den Anfang drei rechteckige *Regions of Interest* (ROI) angelegt, die so dimensioniert und positioniert sind, dass in ihnen jeweils jene Teilmenge von Punkten liegt, die zu einer der drei vor dem Auto liegenden Spurmarkierungen gehört. Es gibt also je eine ROI für die linke, die gestrichelte mittlere und die rechte Spurmarkierung. Die Breite einer solchen ROI ist so bemessen, dass zum einen sowohl gerade als auch für eine Kurve gekrümmte Spurmarkierungen zu einem für die erstmalige Modellierung ausreichenden Anteil abgedeckt sind. Des weiteren ist eine Toleranz bei der Positionierung des Autos innerhalb der rechten Fahrspur gegeben, sodass auch

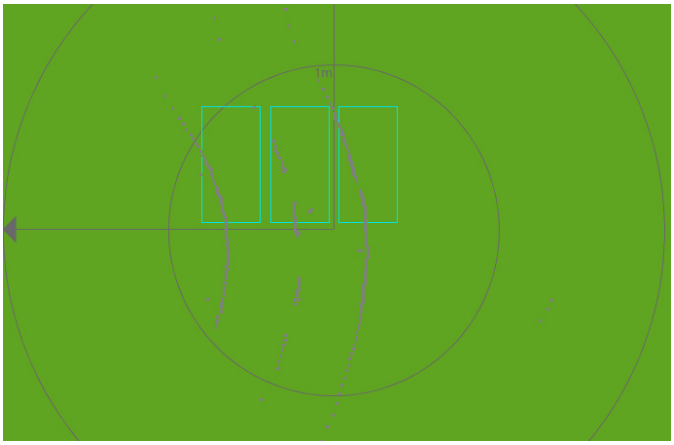

Abbildung 8: Drei für die Unterteilung der Spurmarkierungspunkte angelegte Regions of Interest in Blau

dann noch zur den Spurmarkierungen gehörende Punkte in einer ROI liegen, wenn das Auto ein wenig von der rechten Spurmitte abkommt. Vertikal sind diese Regionen ausgehend von der y-Achse des Koordinatensystems so weit vor das Auto ausgedehnt, dass in einer scharfen Kurve noch keine zu einer anderen Fahrspur gehörenden Punkte in ihnen liegen. Steht das Auto parallel zur Fahrtrichtung auf einem geraden Streckenabschnitt genau in der Mitte der rechten Fahrspur, liegen die Punkte der Spurmarkierungen auf der senkrechten Mittellinie einer solchen ROI. Ein Beispiel ist in Abbildung 8 zu sehen.

Die drei Teilmengen der Mittelpunkte werden erzeugt, indem für jeden Punkt der Gesamtmenge überprüft wird, ob er in einer der drei Regions of Interest liegt und der Punkt dann ggf. zur entsprechenden Teilmenge hinzugefügt wird.

3.2.2 Modellierung der Spurmarkierungen

Nun wird versucht, aus den drei Teilmengen jeweils mittels Polynominterpolation in Kombination mit dem RANSAC-Algorithmus ein passendes Polynom zu generieren. Als erstes werden dafür die Punkte der Teilmenge nach ihrer x-Koordinate sortiert. Die sortierten Punkte werden in drei Untermengen aufgeteilt, in obere, mittlere und untere Punkte, so ähnlich wie bei Deng und Han (2013). Während des RANSAC-Algorithmus werden die folgenden Schritte in mehreren Iterationen wiederholt. Aus jeder der Untermengen wird zufällig ein Punkt gewählt. Aus die-

sen drei Punkten wird ein neues Newton-Polynom erzeugt. Dafür kommt ein in der Software bereits vorhandenes Modul zur Anwendung, welches eine Datenstruktur auf Basis einer Matrix zur Verfügung stellt, für die die Methode der dividierten Differenzen und die Auswertung des Polynoms an einer gewünschten Stelle mit dem Hornerschema bereits implementiert sind. Jedes Mal, wenn ein Punkt zur Datenstruktur hinzu gefügt wird, wird die Matrix vergrößert und mithilfe der schon vorhandenen Daten der nächste Koeffizient des Newton-Polynoms berechnet, wie in Abschnitt 2.3 beschrieben.

3.2.3 Bewertung der Modelle

Das erzeugte Polynom muss einige Kriterien erfüllen, damit es als fehlerhaftes Modell nicht direkt wieder verworfen und der Iterationsschritt übersprungen wird. Es darf zum einen im relativen Koordinatensystem einen vor dem Auto liegenden, vordefinierten Korridor nicht verlassen, also nicht zu stark gekrümmt sein. Solche invaliden Polynome können entstehen, wenn in größerer Distanz zum Auto, etwa an einer Kreuzung oder durch eine Verdeckung nicht genügend Kantenpunkte gefunden wurden. Zum anderen darf aus Plausibilitätsgründen das gefundene Polynom nicht zu weit vom im letzten Bild an dieser Position erkannten Polynom entfernt sein.

Als nächstes wird jeweils für alle Punkte der Untermengen überprüft, ob sie von dem berechneten Polynom abgedeckt werden, also ob ihre Distanz zum Polynom kleiner als ein vorher festgelegter Wert m ist. Aus Effizienzgründen wird an dieser Stelle nur die horizontale Distanz betrachtet. Liegt ein Punkt innerhalb dieser Distanz, zählt er zu den Unterstützern des Polynoms innerhalb der Untermenge. Im Anschluss wird der Anteil von Unterstützern an der Gesamtmenge der in den Untermengen vorhandenen Punkten berechnet, wobei hier die Unterstützer aus der oberen Untermenge, also der am weitesten vor dem Auto liegenden Punkte, stärker gewichtet werden als die restlichen. Dies geschieht, um den durch Ausreißer entstehenden Fehlern entgegen zu wirken, die hier durch die geringere Punktdichte größer ausfallen. Wenn der errechnete Anteil kleiner als ein vordefinierter Schwellwert ist, wird das Polynom ebenso als nicht ausreichend gutes Modell verworfen.

Jetzt wird überprüft, ob in einer vorherigen Iteration für das aktuelle Bild bereits ein Polynom mit einem höheren Anteil von Unterstützern gefunden wurde. Ist dies

nicht der Fall, wird das aktuelle Polynom zusammen mit seinem Unterstützeranteil als das an dieser Position für dieses Bild beste Polynom gespeichert.

Falls in keiner der Iterationen ein ausreichend gutes Polynom gefunden wurde, gilt das Polynom für diese Spurmarkierung in diesem Bild als nicht erkannt. Wenn jedoch ein Polynom erkannt wurde, kann für das nächste Bild eine an das Polynom angepasste Region of Interest verwendet werden, um die Punktmenge zur Bestimmung des nächsten Polynoms weiter einzuschränken.

3.2.4 Angepasste ROI

Eine solche angepasste ROI besteht darin, dass beim Aufteilen der gesamten Punktmenge nur noch jene Punkte zu einer Teilmenge für die Modellierung des Polynoms des nächsten Bildes hinzu gefügt werden, die zum im aktuellen Bild erkannten Polynom eine maximale Distanz von 5 cm haben. Mit diesem Abstand werden Punkte von Markierungen angrenzender, parallel verlaufender Streckenabschnitte gerade noch ignoriert. Die angepasste ROI reicht im Vergleich zur anfänglichen ROI weiter vor das Auto, da sie nahe um die Spurmarkierungspunkte liegt und so auch in scharfen Kurven Punkte anderer Spurmarkierungen nicht in ihr liegen, da die Markierungen sich nicht schneiden. Es wird bei dieser Vorgehensweise der Umstand genutzt, dass die Spurmarkierungen sich von einem zum nächsten Bild nur wenig verändern und somit alle Punkte der Spurmarkierung des nächsten Bildes sich noch innerhalb der angepassten ROI des erkannten Polynoms des aktuellen Bildes befinden. Die so erzeugten Regions of Interest werden mit den erkannten Polynomen mitgeführt. Durch ihre geringe Breite wird die Zahl der von Bildartefakten wie Spieglungen stammenden Ausreißer oder von unerwünschten Punkten der Start- und Haltelinien vermindert und so die Qualität der Eingabedaten für die Modellierung konstant hoch gehalten. In Abbildung 9 sind die modellierten Polynome mit ihren angepassten ROI zu sehen. Falls sich dennoch einmal innerhalb der angepassten ROI kein ausreichend gutes Polynom finden lässt, wird für das nächsten Bild wieder die in Abschnitt 3.2.1 beschriebene rechteckige ROI verwendet.

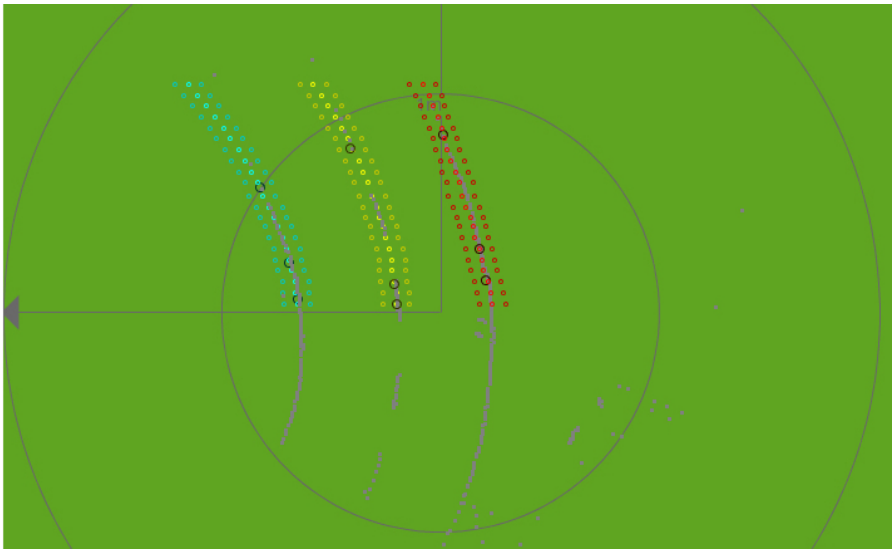

Abbildung 9: Die Polynome in hellen Farben umgeben von den an sie angepassten Regions of Interest in dunkleren Farben

3.2.5 Modellierung der Fahrspur

Als Ergebnis der Modellsuche liegen nun bis zu drei die Spurmarkierungen modellierende Polynome vor. Aus diesen wird nun ein Modell für die Fahrspur abgeleitet, welches in der Implementierung des Fahrverhaltens genutzt wird. Dafür wird das beste Polynom aus den vorhandenen ausgewählt. Wurden zwei oder alle drei Polynome erkannt, werden sie zuerst hinsichtlich ihrer Krümmung verglichen. Dies erfolgt durch Vergleichen der Ableitungen der Polynome orthogonal zum Streckenverlauf an einer ausgewählten Stelle vor dem Auto. Weicht ein Polynom in seiner Krümmung zu stark von den anderen ab, wird es verworfen. Von den übrigen wird das Polynom mit dem größeren Anteil von Unterstützern gewählt. Wurde nur ein Polynom erkannt, dann wird dieses verwendet. Wenn jedoch gar kein Polynom gefunden wurde, kann für dieses Bild kein Modell der Fahrspur erzeugt werden.

Das Fahrspurmodell wird nun erstellt, indem auf Basis des gewählten Polynoms in die Mitte der rechten Fahrspur neues Polynom erzeugt wird. Dafür werden drei ausreichend voneinander entfernt liegende Punkte des gewählten Polynoms entlang der Normalen an das Polynom im entsprechenden Punkt in die Mitte der rechten Fahrspur verschoben. Aus den drei verschobenen Punkten kann durch die

oben beschriebene newtonsche Polynominterpolation das gesuchte Polynom zur Modellierung der Fahrspur erstellt werden.

4 Fahrverhalten

Für vergangene Carolo-Cup-Wettbewerbe wurden bereits Softwarekomponenten entwickelt, durch die das Auto in der Lage ist, autonom auf der Wettbewerbsstrecke zu fahren. Zum einen kommt hierfür eine von Maischak (2014) entwickelte Lokalisierung zum Einsatz und zum anderen ein von Krakowczyk (2014) entwickelter Controller zur Pfadverfolgung. Da die Implementierung des auf der erkannten Fahrspur basierenden Fahrverhaltens auf diesen vorhandenen Funktionalitäten aufbaut, werden sie hier kurz erläutert.

Grundlage für die Lokalisierung ist eine vor der Fahrt manuell aus Fotos erstellte Karte der Strecke, über die ein globales Koordinatensystem gelegt wird. Mit einem Hilfsprogramm wird ein Pfad in die auf der Karte sichtbare Strecke eingezeichnet, den das Auto später auf der Strecke abfahren soll. Er wird in Form einer Liste von Punkten mit globalen Koordinaten zurück gegeben. Alternativ kann er auch während des Abfahrens der Strecke aufgezeichnet werden. Der Pfad und die in eine Bitmap umgerechnete Karte werden dann auf dem Hauptrechner des Autos abgespeichert.

Das Auto fährt den Pfad ab und dabei wird versucht, mit dem auf dem *Otto-Board* verbauten Inertialsensor und den Informationen vom Motorregler die abgefahrene Trajektorie und damit die aktuelle *Pose* (Position und Ausrichtung) auf der Strecke nachzuvollziehen. Dies reicht jedoch wegen der Messfehler der Sensoren nicht aus. Um die über die *Pose* getroffene Annahme zu korrigieren, wird zum einen ein aus der Karte berechnetes *Force Field* verwendet. Mit diesem wird die hypotetische, möglicherweise verschobene und verdrehte *Pose* zur aus den im Kamerabild sichtbaren Spurmarkierungen abgeleiteten tatsächlichen *Pose* "gezogen". Eine weitere Korrekturmöglichkeit besteht in der Verwendung eines Partikelfilter-Algorithmus, bei dem mehrere Hypothesen zur aktuellen *Pose* aufgestellt werden und mithilfe der aus dem Kamerabild gewonnenen Informationen bewertet werden. Durch Entfernen der schlechter bewerteten Hypothesen wird die Menge der Hypothesen eingeschränkt und für die folgenden Bilder weiterverwendet. Die der

tatsächlichen *Pose* am nächsten liegende wird für das Fahrverhalten weiter verarbeitet.

Das Verhalten des Autos wird über ein Modul geregelt, welches aus verschiedenen Strategien auswählt. So gibt es für jede der drei dynamischen Wettbewerbsdisziplinen eine eigene Strategie und noch eine für den Leerlauf und eine weitere zum Konfigurieren des Autos. Während der Ausführung einer Strategie werden Bild für Bild eine Reihe verschiedener Verhaltensweisen ausgeführt.

Das Abfahren des Pfades erfolgt mithilfe des Pfadverfolgungsverhaltens. Für dieses werden ausgehend von der in der Lokalisierung ermittelten aktuellen *Pose* einige vor dem Auto liegende Pfadpunkte aus dem globalen in das zum Auto relative Koordinatensystem transformiert und somit ein lokaler Pfad erstellt. Mithilfe des Controllers von Krakowczyk (2014) werden die Bewegungskommandos für den Motorregler und den Lenkservo erzeugt, die notwendig sind, um den Pfad möglichst genau und ohne Oszillationen mit dynamischen Geschwindigkeitsanpassungen abzufahren. Als Parameter benötigt der Controller nur eine angestrebte Zielgeschwindigkeit und eine *lookahead distance*, also die Distanz, ab der der nächste Pfadpunkt vor dem Auto gewählt wird.

4.1 Integration

Da sich das Verfolgen einer erkannten Fahrspur vom Verfolgen eines Pfades kaum unterscheidet, lag es nahe, das aus der Fahrspurerkennung resultierende Verhalten unter Nutzung der vorhandenen Softwarearchitektur zu implementieren. Statt aus dem globalen Pfad einen lokalen zu berechnen, wird der für den Controller erforderliche lokale Pfad aus der erkannten rechten Fahrspurmitte bestimmt. Dies geschieht in einem neuen Modul, welches das von der Fahrspurerkennung zur Verfügung gestellte Polynom der rechten Spurmitte als Eingabe verwendet und daraus eine Liste von Punkten ableitet, die in gleichmäßigen Abständen auf dem Polynom liegen. Diese Punkte sind bereits in lokalen Koordinaten gegeben. Der erzeugte Pfad ist in Abbildung 10 dargestellt.

Das neue Modul für das Verhalten der Spurverfolgung ist dem für das Verhalten der Pfadverfolgung sehr ähnlich, allerdings wird direkt der eben beschriebene lokale Pfad verwendet. Das Fahrverhalten wird dann genau wie bei der Pfadver-

Abbildung 10: Der für das Fahrverhalten erzeugte Pfad aus einzelnen Pfadpunkten in Schwarz

folgung vom oben beschriebenen Controller umgesetzt. Wenn einmal für ein Bild keine Fahrspur modelliert werden und somit auch kein lokaler Pfad erstellt werden konnte, hält das Auto im Gegensatz zur Pfadverfolgung nicht einfach an, sondern fährt als Notlösung für dieses Bild auf einem geraden Pfad weiter, um im nächsten Bild möglichst wieder einen der Spur entsprechenden Pfad zu bekommen.

5 Ergebnisse

Bisher wurden der Ansatz und seine Implementierung beschrieben. Hier folgt eine Erörterung über die Praxistauglichkeit.

In Tests, die auf der im Vergleich zu Wettbewerbsstrecke viel kleiner dimensionierten Laborstrecke durchgeführt wurden, war das Auto in der Lage, Teilstrecken autonom zu fahren. Durch die geringe Sichtweite der Kamera mit in der Entfernung spärlicher werdenden Kantenpunkten können jedoch die Spurmarkierungen nur in einer kurzen Distanz von ca. 1 m vor dem Auto erkannt werden. Dies hat zur Folge, dass für eine sichere Fahrt innerhalb der Spur die Geschwindigkeit eingeschränkt werden muss.

Gegen dieses Problem kann ohne Änderungen am Hardwaresetup wenig unternommen werden. Eine Möglichkeit wäre, die Scanlinien in den äußeren Bereichen

des Kamerabildes dichter anzuordnen, um die Dichte der äußeren Kantenpunkte im relativen Koordinatensystem zu erhöhen. Dadurch könnte der Bereich vor dem Auto, in dem eine stabile Modellierung der Spuren möglich ist, vergrößert werden, was höhere Geschwindigkeiten erlaubt.

Außerdem ist die Spurerkennung ziemlich fragil. Auf geraden Strecken und in Kurven, die nicht von Markierungen anderer Streckenabschnitte umgeben sind, funktioniert sie zuverlässig und das Auto folgt wie gewünscht der rechten Spurmitte. An anderen Stellen, an denen angrenzende Spuren sichtbar sind oder Linien die Strecke kreuzen, ist sie jedoch noch sehr instabil. Dort wird wegen der hohen Zahl an Ausreißern zu häufig nur eine der drei Spurmarkierungen erkannt, was für das Wettbewerbsszenario, in dem bekanntermaßen Markierungen auch fehlen können, nicht ausreichend ist. In seltenen Fällen wird auch ein falsches oder gar kein Modell gefunden. Geschieht dies in Kurven in mehreren aufeinander folgenden Bildern, verlässt das Auto die Spur.

Ein möglicher Lösungsansatz wäre, die wegen ihrer höheren Effizienz für die Modellierung gewählte Polynominterpolation durch eine lineare Regression zu ersetzen, die bei einer großen Zahl von Ausreißern zwar ungenauere, aber weniger invalide Modelle erzeugt. So könnte die Anzahl der Bilder ohne erkanntes Spurmodell verringert werden.

Ebenso schwierig gestaltet sich die Erkennung an Kreuzungen. Hier fehlen die meisten Markierungen generell. Einzig die Haltelinien können beim Überfahren der Kreuzung in Richtung der Vorfahrtsstraße genutzt werden. Sonst besteht bisher nur die Möglichkeit, die Kreuzung mangels erkannter Spur ”blind”und damit einfach geradeaus zu überfahren. In manchen Fällen kommt es zusätzlich durch die Markierungen der kreuzenden Spur in zu Fehlern bei der Modellierung. Zum Beispiel wird fälschlicherweise kurzzeitig eine Kurve erkannt, sodass das Auto beim Überfahren der Kreuzung abgelenkt werden kann und nach der Kreuzung die korrekte Spur verfehlt.

Eine nahende Kreuzung müsste über ein anderes Modul anhand Charakteristika wie im rechten Winkel aufeinander treffender Linien und fehlender Mittellinien frühzeitig erkannt und im Fahrverhalten gesondert behandelt werden.

Wegen diesen noch bestehenden Problemen ist der vorgestellte Ansatz im Carolo-Cup praktisch leider nur begrenzt einsetzbar.

6 Fazit

Es muss festgestellt werden, dass die Zielstellung der Arbeit, eine stabile Fahrspurerkennung mit darauf aufbauendem Fahrverhalten für den Einsatz im Carolo-Cup zu implementieren, nur in Ansätzen erfüllt werden konnte. Stellenweise konnten mit dem vorgestellten Ansatz gute Ergebnisse erzielt werden, aber im Allgemeinen bedarf er weiterer Optimierung.

Daneben bestehen als Ausblick einige Möglichkeiten der Weiterentwicklung. Sobald die neue Stereokamera uneingeschränkt zur Verfügung steht, können die Kantenerkennung und die Spurerkennung an diese angepasst werden. Ein Vorteil liegt in der viel größeren Sichtweite der Kamera, die eine Modellierung der Fahrspuren über eine größere Distanz ermöglicht und so die Planbarkeit der Fahrt erhöht. Es wird dann aber auch ein anderes Spurmodell nötig, dass den komplexeren Verlauf der sichtbaren Spuren besser abdeckt, zum Beispiel eine B-Snake wie in Wang, Teoh et al. (2004).

Wenn die mithilfe der Stereokamera ermöglichte Hinderniserkennung fertig gestellt ist, wird es erforderlich, die Spurerkennung so zu modifizieren, dass Spurwechselmanöver möglich werden. Dafür könnten die rechteckigen *Regions of Interest* mit einer Offset-Variable horizontal dynamisch verschoben werden.

Es bleibt also noch viel zu tun.

Eigenständigkeitserklärung

Ich erkläre hiermit, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Conrad Läßig
Berlin, 9. Mai 2015

Literatur

Aly, Mohamed (2008): Real time Detection of Lane Markers in Urban Streets, in: *IEEE Intelligent Vehicles Symposium*, IEEE, S. 7–12.

Boldt, Jan Frederik und Severin Junker (2012): Evaluation von Methoden zur Umfelderkennung mit Hilfe omnidirektionaler Kameras am Beispiel eines Modellfahrzeugs, Freie Universität Berlin, Institut für Informatik.

Chum, Ondřej, Jiří Matas und Josef Kittler (2003): Locally Optimized RANSAC, in: *Pattern Recognition, 25th DAGM Symposium, Magdeburg, Germany, September 10-12, 2003, Proceedings*, hrsg. von Bernd Michaelis und Gerald Krell, Lecture Notes in Computer Science 2781, Berlin: Springer, S. 236–243.

Deng, Jiayong und Youngjoon Han (2013): A Real-time System of Lane Detection and Tracking Based on Optimized RANSAC B-spline Fitting, in: *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, RACS '13, ACM, S. 157–164.

Fischler, Martin A. und Robert C. Bolles (1981): Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, in: *Communications of the ACM,* Jg. 24, Nr. 6, S. 381–395.

Freund, Roland W. und Ronald H. W. Hoppe (2007): *Stoer/Bulirsch: Numerische Mathematik 1*, 10. Auflage, Berlin: Springer.

Hartley, Richard und Andrew Zisserman (2004): *Multiple View Geometry in Computer Vision*, 2. Auflage, Cambridge: Cambridge University Press.

Krakowczyk, Daniel (2014): A path following control architecture for autonomous vehicles, Bachelorarbeit im Fach Informatik, Freie Universität Berlin.

Lim, King Hann, Kah Phooi Seng und Li-Minn Ang (2012): River Flow Lane Detection and Kalman Filtering-Based B-Spline Lane Tracking, in: *International Journal of Vehicular Technology,* Ausgabe 2012.

Lipski, Christian, Björn Scholz, Kai Berger, Christian Linz und Timo Stich (2008): A Fast and Robust Approach to Lane Marking Detection and Lane Tracking, in: *IEEE Southwest Symposium on Image Analysis and Interpretation*, SSIAI 2008, IEEE, S. 57–60.

Maischak, Lukas (2014): Lane Localization for Autonomous Model Cars, Masterarbeit im Fach Informatik, Freie Universität Berlin.

Markoff, John (2010): Google Cars Drive Themselves, in Traffic, URL: `http://www.nytimes.com/2010/10/10/science/10google.html` (besucht am 06. 05. 2015).

Sobel, Irwin (2015): History and Definition of the Sobel Operator, URL: `https://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator` (besucht am 02. 06. 2014).

Söldner, Michael (2014): 100 selbstfahrende Autos von Volvo kurven durch Göteborg, URL: `http://www.pcwelt.de/news/100_selbstfahrende_Autos_von_Volvo_kurven_durch_Goeteborg-Autonom-8691067.html` (besucht am 06. 05. 2015).

Tan, Huachun, Yang Zhou, Yong Zhu, Danya Yao und Keqiang Li (2014): A novel curve lane detection based on Improved River Flow and RANSAC, in: *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, S. 133–138.

Technische Universität Braunschweig (2014): Carolo-Cup Regelwerk 2015, URL: `https://wiki.ifr.ing.tu-bs.de/carolocup/system/files/Hauptwettbewerb2015.pdf` (besucht am 26. 04. 2015).

Torr, Philip H. S. und Andrew Zisserman (2000): MLESAC: A New Robust Estimator with Application to Estimating Image Geometry, in: *Computer Vision and Image Understanding,* Jg. 78, Nr. 1, S. 138–156.

Wang, Yue, Dinggang Shen und Eam Khwang Teoh (2000): Lane detection using spline model, in: *Pattern Recognition Letters,* Jg. 21, Nr. 8, S. 667–689.

Wang, Yue, Eam Khwang Teoh und Dinggang Shen (2004): Lane detection and tracking using B-Snake, in: *Image and Vision Computing,* Jg. 22, Nr. 4, S. 269–280.