Fahrbahnerkennung und automatisierte Fahrbahnführung anhand einer Farbkamera mit einem Modellfahrzeug

Lane Detection and automated Lane Following using a Color Camera with a Model Car

Bachelor-Thesis eingereicht von
Nicolas Acero Sepulveda
am 7. Oktober 2016

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachgebiet Echtzeitsysteme

Elektrotechnik und Informationstechnik (FB18)

Zweitmitglied Informatik (FB20)

Prof. Dr. rer. nat. A. Schürr
Merckstraße 25
64283 Darmstadt

www.es.tu-darmstadt.de

Gutachter: Prof. Dr. rer. nat. Andy Schürr
Betreuer: Géza Kulcsar

ES-B-0117

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis selbstständig und ohne Hilfe Dritter angefertigt zu haben. Gedanken und Zitate, die ich aus fremden Quellen direkt oder indirekt übernommen habe, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und wurde bisher nicht veröffentlicht.

Ich erkläre mich damit einverstanden, dass die Arbeit auch durch das Fachgebiet Echtzeitsysteme der Öffentlichkeit zugänglich gemacht werden kann.

Darmstadt, den 7. Oktober 2016

_____

(N. Acero Sepulveda)

Zusammenfassung

Die vorliegende Bachelorarbeit präsentiert, basierend auf bereits vorgestellten Ansätzen, ein robustes Verfahren zur kamerabasierten Fahrbahnerkennung und automatischen Fahrbahnführung auf einem Modellfahrzeug in Indoor-Szenarien. Dabei wird als erstes die IPM-Technik verwendet, um eine projektive Transformation zwischen Bild- und Fahrzeugkoordinatensystem durchzuführen. Als nächstes erfolgt eine Kantenfilterung, die die Fahrbahnmarkierungen im Bild verschärft. Anschließend werden die Fahrbahnmarkierungen mit Hilfe von Geraden und kubischen Splines im Zusammenspiel mit dem RANSAC-Algorithmus modelliert. Danach werden die gefundenen Modellinstanzen validiert, um die zu befahrende Fahrspur zu ermitteln. Im Anschluss daran wird die Fahrspurmitte berechnet. Diese wird schließlich dafür verwendet, das erforderliche Lenkmanöver einzuleiten.
Um die Robustheit gegen Ausreißer und mögliche Verarbeitungsfehler zu erhöhen, wird das beschriebene System von einem Tracking-Verfahren unterstützt.
Die Ergebnisse dieser Thesis zeigen, dass das vorgestellte Verfahren mit einer Richtig-Positiv-Rate von mindestens 97,71% sowie mit einer gesamten Rechenzeit von 41,58ms vielversprechend ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Fahrbahnerkennung und die automatische Fahrspurführung sind wesentliche Aufgaben des autonomen Fahrens. Diese Aufgaben ermöglichen komplexere Funktionen wie Parkassistenz, automatisierte Überholmanöver und Ausweichen von Hindernissen. Durch die Ergebnisse dieser Bachelorarbeit soll weiteren Studierenden eine Basis zum Lösen neuer Herausforderungen gegeben werden, die im Rahmen des Projektseminars Echtzeitsysteme der Technischen Universität Darmstadt erarbeitet werden. Dabei soll den Studenten der genannten Universität in naher Zukunft die Teilnahme an Wettbewerben für autonomes Fahren ermöglichen. Im folgenden Absatz wird einer dieser Wettbewerbe vorgestellt.

1.1 Carolo-Cup

Der Carolo-Cup ist ein Wettbewerb für autonomes Fahren, der seit 2008 von der Technischen Universität Braunschweig veranstaltet wird. Die Zielgruppe dieser Veranstaltung sind studentische Teams, die gemeinsam Konzepte bei drei verschiedenen Disziplinen entwickeln und diese präsentieren.

Die Teilnahme am Carolo-Cup beinhaltet die Konstruktion eines Modellfahrzeugs im Maßstab von 1 : 10, das sowohl nach Funktionalität als auch Energieeffizienz sowie Kostenminimierung bewertet wird. Zudem sind während des Wettbewerbs die folgenden Disziplinen zu bestehen und bewerten[Car14]:

- **Paralleles Einparken:** Das Fahrzeug soll auf einer Straße fahren und möglichst schnell eine ausreichend lange Parklücke finden und berührungslos zwischen eventuellen Hindernissen einparken.
- **Rundstrecke ohne Hindernisse:** Fahrt auf einer Straße ohne Hindernisse und ohne Beachtung von Vorfahrtregeln. Die Fahrt soll zwei Minuten dauern und Teile der Fahrbahnmarkierungen können unterbrochen bzw. verdeckt sein.
- **Rundstrecke mit Hindernissen:** Ergänzung der zweiten Disziplin mit Hindernissen und Berücksichtigung der Vorfahrtregeln.

Die Erkennung der Fahrbahn und ihre Einhaltung haben einen großen Einfluss auf den Erfolg bei den genannten Aufgaben und auf eine gute Bewertung. Punkte beim Verlassen der Fahrbahn sind abzuziehen.

Beim Ansatz dieser Arbeit wird von Fahrbahneigenschaften ausgegangen, die sich am Regelwerk des Carolo-Cups orientieren und der Fokus wird auf die Disziplin *„Rundstrecke ohne Hindernisse“* gelegt.

1.2 Problemstellung und Zielsetzung

Zur Verwirklichung des autonomen Fahrens sind bereits in der Wissenschaft verschiedene Techniken vorhanden. Eine dieser Techniken basiert sowohl auf einer robusten

kamerabasierten Fahrbahnerkennung als auch auf einer automatischen Fahrspurführung. Für diese zwei wesentlichen Aufgaben sind zahlreiche Techniken in der Literatur beschrieben. Es gibt jedoch weder eine perfekte Lösung noch einen überall empfohlenen Ansatz, da die Ergebnisse der verschiedenen Ansatzvarianten vom Anwendungsszenario sowie von Einflussfaktoren abhängen. Es bietet sich dennoch an, basierend auf bereits vorgestellten Ansätzen, eine auf einen Anwendungsfall spezialisierte Umsetzung zu entwickeln und zu evaluieren. Aus diesem Grund ist Ziel dieser Arbeit, basierend auf bereits vorgestellten Ansätzen, eine Variante zur kamerabasierten Fahrbahnerkennung und Fahrbahnführung zu entwickeln, die auf Modellfahrzeugen in *Indoor*-Szenarien spezialisiert ist. Darüber hinaus wird im Rahmen dieser Arbeit die Funktionalität und Verwendbarkeit von der resultierenden Umsetzung evaluiert.

Die Fahrbahnerkennung ist ein aktives Forschungsthema, das mit verschiedenen Herausforderungen verbunden ist. Die Fahrbahnerkennung lässt sich im Allgemein in drei Phasen aufteilen. Diese und die jeweiligen Herausforderungen sind im Folgenden beschrieben.

- **Vorverarbeitung:** In diesem Schritt ist eine passende Filterung bzw. Transformation des Bildes zu finden, sodass die Fahrbahnmarkierungen verschärft und die nicht relevanten Informationen unterdrückt werden. Dabei liegt die Schwierigkeit in der Abhängigkeit von externen Faktoren. Eine Aufnahme kann beispielsweise aufgrund der Fahrdynamik verwackelt sein oder wegen mangelnder Lichtintensität verrauscht werden.

- **Extraktion der Merkmale:** Alle Fahrbahnmarkierungen sind aus dem bearbeiteten Bild zu extrahieren. Die verschiedenen Arten von Fahrbahnmarkierungen und Ausreißern stellen eine Herausforderung bei der Detektion von Merkmalen dar. Eine Fahrstrecke kann aus geraden, gebogenen und unterbrochenen Teilen bestehen.

- **Modellierung der Fahrbahn:** Allein durch die extrahierten Merkmale ist man nicht in der Lage, zu erkennen, welche davon tatsächlich zu Fahrbahnmarkierungen gehören und damit gültige Fahrspuren definieren. Dafür ist ein Modell der Fahrbahn zu erstellen und die Merkmale nach diesem zu bewerten.

Weitere Herausforderungen bei der Fahrbahnerkennung sind die Echtzeitfähigkeit und das begrenzte Sichtfeld der Kamera. Zum einen sind viele Bilder pro Sekunde durch die drei genannten Phasen zu bearbeiten, um eine ausreichende Regelung der Spurführung zu gewinnen. Dies setzt eine optimale Ressourcennutzung voraus. Zum anderen sind alle Fahrbahnmarkierungen trotz tiefer Position und begrenztem Sichtfeld der Kamera im Bild aufzunehmen.

Andererseits stellt die Fahrzeugregelung bei der Fahrspurführung auch eine Herausforderung dar. Eine Regelvorgabe ist aus der im Bild erkannten Fahrbahn abzuleiten. Dafür ist die Herleitung einer Beziehung zwischen Bild- und Weltkoordinaten erforderlich.

1.3 Struktur der Arbeit

In Kapitel 2 werden zunächst die wichtigsten Grundlagen vorgestellt. Diese sollen das Verständnis dieser Arbeit erleichtern. Als nächstes wird ein bereits an der TU-Darmstadt entwickelter Ansatz zur Fahrbahnerkennung zusammengefasst. Außerdem werden die Schwächen des genannten Ansatzes erläutert. Im Anschluss daran werden die Lösungsansätze, auf denen diese Arbeit basiert, vorgestellt und diskutiert. Am Ende des Kapitels erfolgt die Wahl des Ansatzes dieser Arbeit.
In Kapitel 3 geht es um die Umsetzung. Hier wird als erstes die für das Testen der Fahrbahnerkennung und Fahrbahnführung benutzte Testfahrbahn beschrieben. Als nächstes wird sowohl das verwendete Modellfahrzeug als auch die vorhandene Hardware vorgestellt. Danach werden die verwendeten Softwarebibliotheken und Programme erläutert. Anschließend wird der Programmablaufplan vorgestellt. Im Anschluss daran wird auf jede Phase des Programmablaufplans näher eingegangen.
In Kapitel 4 werden hauptsächlich die Ergebnisse dieser Arbeit präsentiert und diskutiert. Außerdem erfolgt eine Untersuchung der Auswirkung ausgewählter Parameter auf die Funktionalität des Algorithmus für Fahrbahnerkennung. Im Anschluss daran werden die durchschnittlichen Rechenzeiten je Phase des Algorithmus vorgestellt und diskutiert. Darüber hinaus werden als Ausblick auf diese Bachelorarbeit Optimierungen zur Reduzierung der Rechenzeit bzw. des Rechenaufwands vorgeschlagen. Am Ende des Kapitels erfolgt eine qualitative Auswertung der Funktionalität der Fahrbahnerkennung sowie der Fahrbahnführung auf dem Modellfahrzeug. Dabei werden mögliche Probleme erörtert.
In Kapitel 5 wird hauptsächlich der Stand der Technik beschrieben. Außerdem werden drei verschiedene Ansätze beleuchtet. Dabei werden etwaige Gemeinsamkeiten und Unterschiede mit dem Ansatz dieser Thesis aufgezeigt.
Zum Schluss werden in Kapitel 6 die Erkenntnisse sowie die Ergebnisse dieser Arbeit rekapituliert und mögliche zukünftige Verbesserungen bzw. Erweiterungen in Aussicht gestellt.

2 Grundlagen

In diesem Kapitel werden einige Grundlagen vorgestellt, die für das Verständnis dieser Arbeit relevant sind. Als erstes werden die physikalischen Eigenschaften der Fahrstrecke im Carolo-Cup vorgestellt, da die in dieser Arbeit benutzte Testfahrbahn sich am Regelwerk dieses Wettbewerbes orientiert. Anschließend wird eine Technik namens *IPM* vorgestellt, die mittels einer üblichen Kamera und ihren Parametern eine Draufsicht eines Bildbereiches ermöglicht. Da die Fahrbahnmarkierungen als zusammenhängende Kanten zu betrachten sind, erfolgt eine kurze Einführung in die Methoden zur Verschärfung von Kanten in Bildern. Danach wird der *RANSAC*-Algorithmus vorgestellt. Dieser Algorithmus kommt häufig zur Modellauswahl in der Computer Vision zum Einsatz. *RANSAC* ist für diese Arbeit sehr wichtig, da er eine große Rolle beim Auffinden von möglichen Fahrbahnmarkierungen spielt. In diesem Kapitel wird auch die Interpolation mit kubischen Splines erläutert, da dieses Verfahren im Zusammenspiel mit *RANSAC* zur Modellierung der Fahrbahnmarkierungen verwendet wird. Als nächstes erfolgt eine kurze Einführung in den Kalman-Filter, da dieser für das Tracking der erkannten Markierungen verwendet werden kann. Zusätzlich wird der Ungarische Algorithmus beschrieben. Dieser wird für das Tracking der Fahrbahnmarkierungen benötigt. Anschließend wird auf den bereits an der TU-Darmstadt entwickelten Ansatz zur Fahrbahnerkennung eingegangen und dessen Probleme erläutert. Zum Schluss dieses Kapitels werden einige Lösungsansätze vorgestellt und diskutiert. Dies führt zur Wahl des Ansatzes dieser Arbeit, der als Letztes kurz vorgestellt wird.

2.1 Eigenschaften der Fahrstrecke im Carolo-Cup

Die im Carolo-Cup verwendete Fahrstrecke ist die einfache Abbildung einer realen Verkehrsstraße. Der Einfachheit halber setzen die Veranstalter des Wettbewerbes die folgenden relevanten Messgrößen und Eigenschaften der Fahrstrecke im Voraus fest. Die Fahrstrecke setzt sich aus zwei entgegen gerichteten Fahrspuren zusammen, die durch drei weiße Spurmarkierungen gekennzeichnet sind. Diese drei Spurelemente haben eine feste Breite von 20 mm. Die linke und rechte Markierungen bestehen in der Regel aus durchgezogenen Linien und Kurven, während die mittige Markierung nur aus gestrichelten Linien besteht. Darüber hinaus ist die gesamte Breite der Fahrbahn mit 820 mm und der minimale Kurvenradius mit 1000 mm festgelegt. Siehe Abbildung 2.1.

Andererseits lassen sich auch Kreuzungen entlang der Fahrstrecke vom Carolo-Cup sehen. Dabei gelten dieselben Messgrößen und Vereinfachungen wie bei einem einfachen Straßenausschnitt. Der Unterschied liegt nur daran, dass nun Stopplinien vorhanden sind. Diese sind horizontale weiße Linien, die 40 mm breit sind und direkt vor der Kreuzung stehen. Siehe Abbildung 2.1

Es ist noch zu beachten, dass auf der Fahrstrecke vom Carolo-Cup Rechtsverkehr gilt. Außerdem wird je nach Disziplin die Vorfahrtregel berücksichtigt. Siehe Unterabschnitt 1.1. Darüber hinaus ist es möglich, dass alle drei Fahrbahnmarkierungen an beliebiger Stelle auf einer Länge von bis zu 1000 mm unterbrochen werden. [Car14]

Abbildung 2.1: Links: Messgrößen eines üblichen Straßenausschnittes am Carolo-Cup. Rechts: Kreuzung am Carolo-Cup [Car14]

2.2 IPM-Methode

Die **IPM**-Methode (englisch ***I**nverse **P**erspective **M**apping*, deutsch etwa „Abbildung der umgekehrten Perspektive“) ist eine etablierte Methode zur perspektivischen Entzerrung von Bildern. Durch diese Methode wird jedes Pixel eines Bildausschnitts auf eine andere Position abgebildet, sodass eine neue zweidimensionale Pixelmatrix entsteht. Diese neue Matrix stellt eine Draufsicht von dem Bildausschnitt dar, aus der man mit Hilfe von Skalierungsfaktoren die Dimensionen eines Objektes und Abstände abschätzen kann. Aus diesem Grund kommt die IPM-Methode häufig zur Objektdetektion sowie zur Fahrbahnerkennung zum Einsatz [SDJM10, LFW12]. Die IPM-Methode wird für die Fahrbahnerkennung eingesetzt, da durch die genannte Abbildung die Fahrstreifen vertikal und parallel zueinander dargestellt werden. Siehe Abbildung 2.3. Dies bedeutet eine Erleichterung der Fahrbahnerkennung aufgrund der regelmäßigen Breite der Fahrspur und der Fahrbahnmarkierungen. Im Folgenden wird auf die IPM-Methode näher eingegangen.

Zur Durchführung der IPM-Methode sind sowohl intrinsische als auch extrinsische Kameraparameter zu kennen. Zum einen sind die folgenden 4 intrinsischen Kameraparameter zu ermitteln: Brennweiten f_u und f_v sowie die Koordinaten des Bildmittelpunktes c_u und c_v. Diese Parameter lassen sich durch eine Kamerakalibrierung bestimmen,

wie im Buch „*Learning OpenCV: Computer Vision with the OpenCV Library*“ beschrieben wird [BK08]. Zum anderen sind die erforderlichen extrinsischen Kameraparameter die Folgenden: Translation in z-Richtung bzw. die Höhe der Kamera $\boldsymbol{h}$, Yaw-Winkel $\boldsymbol{\beta}$, Pitch-Winkel $\boldsymbol{\alpha}$. Siehe Abbildung 2.2. Im Folgenden werden die Schritte eines IPM-Ansatzes

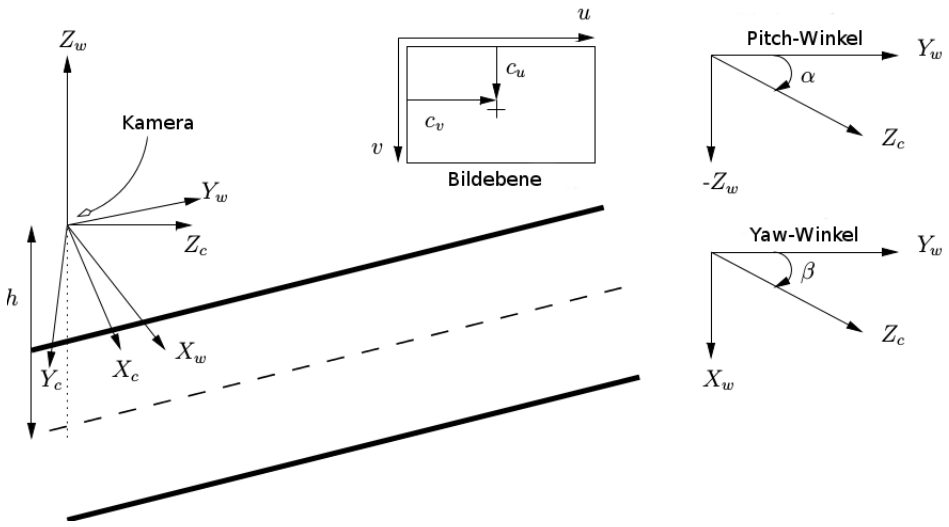


Abbildung 2.2: Links: Koordinatensysteme (Boden-, Kamera-, und Bildbezugssystem). Rechts: Definition von dem Pitch-Winkel α und Yaw-Winkel β.[Aly08]

aus der Arbeit von [Aly08] beschrieben.

1. Berechne mit Gleichung 1 die Faktoren c_1, c_2, s_1 und s_2.

$$c_1 = \cos(\alpha);\ c_2 = \cos(\beta);\ s_1 = \sin\alpha;\ s_2 = \sin(\beta) \tag{1}$$

2. Bestimme die Transformationsmatrix von dem Bild- zum Bodenbezugssystem durch Gleichung 2.

$$T_{Bild}^{Boden} = h \begin{pmatrix} -\frac{1}{f_u}c_2 & \frac{1}{f_v}s_1s_2 & \frac{1}{f_u}c_uc_2 - \frac{1}{f_v}c_vs_1s_2 - c_1s_2 & 0 \\ \frac{1}{f_u}s_2 & \frac{1}{f_v}s_1c_1 & -\frac{1}{f_u}c_us_2 - \frac{1}{f_v}c_vs_1c_2 - c_1c_2 & 0 \\ 0 & \frac{1}{f_v}c_1 & -\frac{1}{f_v}c_vc_1 + s_1 & 0 \\ 0 & -\frac{1}{hfv}c_1 & \frac{1}{hf_v}c_vc_1 - \frac{1}{h}s_1 & 0 \end{pmatrix} \tag{2}$$

3. Wähle die Eckpunkte des zu transformierenden Bildausschnitts aus. Siehe Abbildung 2.3a.

4. Transformiere jeden Eckpunkt des zu transformierenden Bildausschnitts von dem Bild- zu dem Bodenbezugssystem anhand Gleichung 3 und speichere die Ergebnisse in das Array **groundPoints**. Damit definiert sich das Array groundPoints folgendermaßen: groundPoints $= \{P_0(x|y), P_1(x|y), ..., P_n(x|y)\}$. Hier sind die x- und

y-Werte auf den Boden bezogen und sind entweder in Millimeter oder in Meter angegeben.

$$P_{Boden} = T_{Bild}^{Boden} \cdot P_{Bild} \tag{3}$$

5. Bestimme anhand Gleichung 4 und Gleichung 5 die Skalierungsfaktoren $\boldsymbol{s_x}$ und $\boldsymbol{s_y}$, um den transformierten Bildausschnitt in einem Bild (*IPM-Bild*) darzustellen. Dieses Bild hat eine feste Breite $\boldsymbol{b}$ und Höhe $\boldsymbol{a}$.

$$s_x = \frac{b}{\max_{P_x}\{groundPoints\} - \min_{P_x}\{groundPoints\}} \tag{4}$$

$$s_y = \frac{a}{\max_{P_y}\{groundPoints\} - \min_{P_y}\{groundPoints\}} \tag{5}$$

6. Berechne mit Hilfe von den Schrittweiten $\frac{1}{s_x}$ und $\frac{1}{s_y}$ die restlichen Punkte des transformierten Bildausschnitts und speichere sie in groundPoints. Siehe Abbildung 2.3b

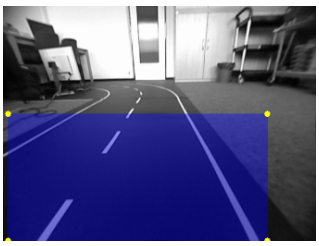
7. Skaliere anhand der Skalierungsfaktoren die x- und y-Werte aller Punkte aus groundPoints, sodass die Punkte einem Pixel von dem IPM-Bild entsprechen.

8. Berechne für jedes skaliertes Element aus groundPoints den zugehörigen Farbwert und setze schließlich diese Information in das IPM-Bild ein. Siehe Abbildung 2.3c.

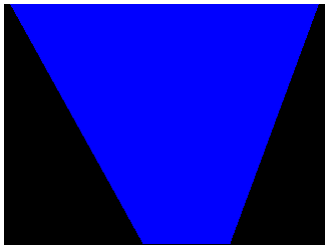
9. Mit Hilfe von Gleichung 6 und Gleichung 7 ist man ggf. in der Lage, die Punkte aus groundPoints zurückzutransformieren. Außerdem ist es auch möglich, anhand der Skalierungsfaktoren die Position der Pixel aus dem IPM-Bild wieder in Meter bzw. Millimeter umzuwandeln.

$$T_{Boden}^{Bild} = \begin{pmatrix} f_u c_2 + c_u c_1 s_2 & c_u c_1 c_2 - s_2 f_u & -c_u s_1 & 0 \\ s_2(c_v c_1 - f_v s_1) & c_2(c_v c_1 - f_v s_1) & -f_v c_1 - c_v s_1 & 0 \\ c_1 s_2 & c_1 c_2 & -s_1 & 0 \\ c_1 s_2 & c_1 c_2 & -s_1 & 0 \end{pmatrix} \tag{6}$$

$$P_{Bild} = T_{Boden}^{Bild} \cdot P_{Boden} \tag{7}$$



(a) Zu transformierender Bildausschnitt (blaues Rechteck)



(b) Punkte des transformierten Bildausschnitts

(c) Ergebnisbild der IPM-Methode

Abbildung 2.3: Anwendungsbeispiel für die IPM-Methode

2.3 Kantendetektion

Zur Fahrbahnerkennung ist eine passende Filterung des Bildes zu finden, sodass die Fahrbahnmarkierungen verschärft und die nicht relevanten Informationen unterdrückt werden. Dafür bietet es sich an, die Bildkanten mit Hilfe von einem sogenannten Kantenfilter hervorzuheben. Diese Art von Filtern berechnen die erste, zweite oder dritte Ableitung des Farb- bzw. Grauwertes eines Bildes in x- oder y-Richtung (oder beidem). Dieses Verfahren erfolgt mit Hilfe von einer Faltungsmatrix (auch *Kern* genannt), die über das Bild geschoben und mit den entsprechenden Bildelementen multipliziert wird. Das Ergebnis dieser Operation liefert ein neues Bild mit der gleichen Dimension wie das Ursprungsbild. Dieses neues Bild markiert mit einem hohem Grauwert die Stellen, an denen sich die Ableitung des Helligkeitswertes von dem Ursprungsbild am stärksten änderte. Im Gegenteil dazu werden die Stellen, an denen es eine kleine Änderung der Ableitung gab, mit einem niedrigen Grauwert gekennzeichnet (siehe Abbildung 2.4). Aufgrund dieses Verhaltens und des Kontrastes der hellen Fahrbahnmarkierungen zum dunklen Fahrbahnbelag ist diese Art von Filtern bei der Fahrbahnerkennung hilfreich.

Einige standardisierten und breit verwendeten Kantenfilter sind:

- Sobel-Operator
- Laplace-Filter
- Canny-Algorithmus
- Scharr-Operator

Diese Kantenfilter basieren auf dem oben beschriebenen Prinzip und unterscheiden sich hauptsächlich in der Faltungsmatrix. Entscheidend für die Ergebnisse eines Kantenfilters ist die Auswahl der Größe der Faltungsmatrix, da diese Große das zu untersuchende Feld definiert. Darüber hinaus erbringen die Kantenfilter je nach Anwendungssituation unterschiedliche Leistungen.

Andererseits ist man auch in der Lage, spezialisierte Kantenfilter zu definieren. Dafür ist eine Faltungsmatrix zu definieren, die für bestimmte Situationen gewünschte

Ergebnisse liefert. Im Ansatz dieser Arbeit wurde ein für die Fahrbahnerkennung spezialisierter Filter verwendet. Dieser Filter ist von einem Hoch- und Tiefpass abgeleitet und wurde der Arbeit von Mohamed in [Aly08] entnommen.

Abbildung 2.4: Anwendungsbeispiel für einen Kantenfilter: Canny-Algorithmus [BK08]

2.4 Spline-Interpolation

In dieser Arbeit sind Fahrbahnmarkierungen zu modellieren. Entscheidend für die Auswahl des Modells ist, dass sowohl gerade Linien als auch gekrümmte Linien mit möglichst hoher Genauigkeit beschrieben werden. Außerdem soll die Modellierung nicht rechenintensiv sein. Dafür eignet sich die Interpolation mit natürlichen kubischen Splines besonders gut. Bei dieser Art von Interpolation wird versucht, mit gegebenen Stützstellen und anhand stückweise stetiger Polynome 3. Grades, ein genaueres Polynom zu approximieren. Auf das allgemeine Verfahren wird im Buch „Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens“ näher eingegangen [HB09, S. 364–372]. Durch Testen stellte man im Bezug auf diese Arbeit fest, dass natürliche kubische Splines mit vier Stützstellen eine zufriedenstellende Genauigkeit bei der Modellierung von Fahrbahnmarkierungen anbieten. Im Folgenden wird die Berechnung der genannten Splines zur Zerlegung $\Delta = \{x_0, x_1, x_2, x_3\}$ (vier Stützstellen) erläutert.

Eine allgemeine Voraussetzung für dieses Verfahren ist die aufsteigende Reihenfolge von den Konstanten in der Zerlegung. Nämlich gilt $x_0 < x_1 < x_2 < x_3$.

Als erstes ist die Breite h jedes Intervalls der Zerlegung zu bestimmen.

$$h_i = x_{i+1} - x_i, \quad i = 0, 1, 2 \tag{8}$$

Anschließend ist das folgende Gleichungssystem aufzustellen, um die sogenannten *Momente* M_1, M_2 zu berechnen. Die Momente M_0 und M_3 sind hier aufgrund der natürlichen Randbedingung gleich null.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{h_0}{6} & \frac{h_0+h_1}{3} & \frac{h_1}{6} & 0 \\ 0 & \frac{h_1}{6} & \frac{h_1+h_2}{3} & \frac{h_2}{6} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ M_1 \\ M_2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0} \\ \frac{y_3-y_2}{h_2} - \frac{y_2-y_1}{h_1} \\ 0 \end{pmatrix} \tag{9}$$

Mit dem Ergebnis aus Gleichung 9 lassen sich mit den nächsten zwei Gleichungen alle Konstanten d_i und c_i bestimmen.

$$d_i = y_i - \frac{h_i^2}{6} M_i, \quad i = 0, 1, 2 \tag{10}$$

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6}(M_{i+1} - M_i), \quad i = 0, 1, 2 \tag{11}$$

Durch Einsetzen der Ergebnisse aus Gleichung 9, Gleichung 10 und Gleichung 11 in Gleichung 12 lässt sich das gesuchte Interpolationspolynom jedes Intervalls der Zerlegung Δ ermitteln.

$$s_i(x) = \frac{1}{6}\left(\frac{(x_{i+1} - x)^3}{x_{i+1} - x_i} M_i + \frac{(x - x_i)^3}{x_{i+1} - x_i} M_{i+1}\right) + c_i(x - x_i) + d_i, \quad i = 0, 1, 2 \tag{12}$$

Schließlich bietet es sich an, anhand der berechneten Polynome den Wert für ein x zwischen x_0 und x_3 zu approximieren. Hierfür ist das entsprechende Polynom zu verwenden. s_0 falls $x < x_1$. s_1 falls $x < x_2$. s_2 falls $x \leq x_3$.

2.5 RANSAC-Algorithmus

Wie in Unterabschnitt 1.2 schon beschrieben wurde, stellen Ausreißer und die verschiedenen Arten von Fahrbahnmarkierungen Herausforderungen bei der Extraktion von Merkmalen dar. Aus diesem Grund ist es sinnvoll, einen Ansatz zur Fahrbahnerkennung mit den folgenden Anforderungen zu gestalten. Der Ansatz soll robust gegen Ausreißer sein und soll die Fahrbahnmarkierungen mathematisch beschreiben. Dafür eignet sich der RANSAC-Algorithmus mit einer Modellierung mittels Interpolation. Der genannte Algorithmus wird im Folgenden vorgestellt.

RANSAC (englisch **ran**dom **sa**mple **c**onsensus) ist ein weit verbreiteter Algorithmus zur Schätzung der Parameter eines mathematischen Modells innerhalb einer Datenmenge. Dieser Algorithmus wurde erstmals von Fischler und Bolles in [FB81] vorgestellt. Die Gründe für seine breite Verwendung sind seine Einfachheit und seine Fähigkeit, mit Datenmengen umzugehen, die mit mehr als 50% Ausreißer-Anteil kontaminiert sind [Yan10]. Daher findet RANSAC häufig Anwendung in Bereichen, bei denen Daten meistens verrauscht und kontaminiert vorliegen, wie in Computer Vision und in der Bildverarbeitung.

Der RANSAC-Algorithmus ist iterativ und basiert auf dem folgenden Verfahren: In jeder Iteration werden $\boldsymbol{n}$ Punkte bzw. Elemente aus einer Datenmenge $\boldsymbol{D}$ ausgewählt, um eine Modellinstanz $\boldsymbol{M}$ zu bestimmen. Anschließend erfolgt eine Bewertung der geschätzten Modellinstanz. Dafür wird versucht, alle anderen Punkte aus der Datenmenge D in M zu fitten. Daraus ergibt sich für jeden dieser Punkte ein sogenannter Fitting-Fehler $\boldsymbol{f}$. Dieser Fehler ist danach mit einem konstanten Schwellwert $\boldsymbol{t}$ zu vergleichen. Ist

f kleiner als t, dann wird der aktuelle Punkt als *inlier* (deutsch „Einlieger“) gekennzeichnet und gespeichert. Die Bewertung der Modellinstanz entspricht der Anzahl an gekennzeichneten inliers. Schließlich wird nach einer fester Anzahl an Iterationen $\boldsymbol{k}$ die bestbewertete Modellinstanz mit der zugehörigen Menge von inliers zurückgegeben. Somit ist gewährleistet, dass der Algorithmus nach $\boldsymbol{i}$-Iterationen eine Teilmenge von D mit möglichst wenigen Ausreißern liefert. Hierbei gilt $0 \leq i \leq k$. Pseudocode ist im Algorithmus 1 zu finden. Zur Verdeutlichung des RANSAC-Algorithmus ist ein Anwendungsbeispiel in Abbildung 2.5 zu sehen.

Aus der vorherigen Beschreibung des RANSAC-Algorithmus ist zu erkennen, dass er von der maximalen Anzahl an Iterationen k abhängt. Laut Ziv Yaniv in [Yan10] lässt sich diese Abhängigkeit mathematisch lösen, indem die maximale Anzahl an Iterationen aus probabilistischer Sicht dynamisch geschätzt wird. Dabei wird dieser Grenzwert aus der Wahrscheinlichkeit, dass keiner der n zufällig ausgewählten Punkte ein Ausreißer ist, abgeleitet. Siehe Zuweisung der Variable s in Zeile 22 vom Algorithmus 1. Diese Variante vom Algorithmus wird „RANSAC mit adaptiver Bestimmung der Parameter“ genannt. In der Abbruchbedingung für die **while** Schleife in Zeile 6 vom Algorithmus 1 ist folgendes zu sehen. Der Algorithmus terminiert, wenn die Anzahl an Iterationen i größer als s oder größer als k ist. Hierbei ist die Variable s der probabilistisch geschätzte Grenzwert für die Anzahl an Iterationen und k ein fester Grenzwert, der für diese Arbeit aufgrund der Echtzeitanforderung eingeführt wurde. Diese Konstante k wurde im Rahmen dieser Arbeit durch Testen bestimmt, hier sei $k = 100$ bei der Modellierung mit kubischen Splines. Weitere Abhängigkeiten vom Algorithmus sind die Anzahl an zufällig auszuwählenden Punkten n und der Schwellwert t. Bei der Modellierung anhand kubischer Splines mit vier Stützstellen ist im Ansatz dieser Arbeit $n = 4$ gegeben. Zudem wurde der Schwellwert auch wie bei k durch Testen ermittelt. Der Schwellenwert t ist in diesem Ansatz gleich eins.

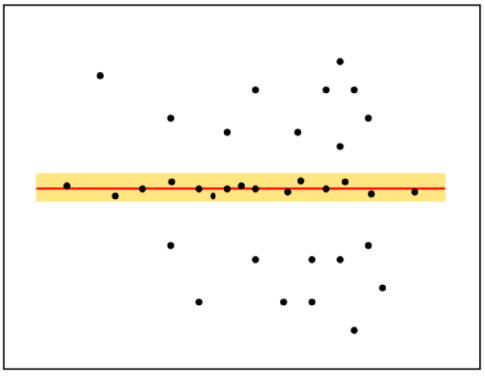


Abbildung 2.5: Anwendungsbeispiel für den RANSAC-Algorithmus für das Fitten einer Gerade innerhalb einer Punktmenge [Kr"15]. Die Punkte innerhalb des gelben Bereichs sind die inliers. In Rot ist die durch RANSAC gefundene Modellinstanz.

Algorithmus 1 RANSAC-Algorithmus mit adaptiver Bestimmung der Parameter

Eingabe:

D → Datenmenge von Punkten, die mit Ausreißern kontaminiert ist
n → Anzahl an Punkten, die für das Fitten des Modells zufällig auszuwählen sind
k → Fester Grenzwert für maximale Anzahl an Iterationen
t → Threshold zur Festlegung, ob ein Punkt, p_i, aus D zu einem Modell, M_i, passt

Ausgabe:

$inliers$ → Untermenge aus D, die möglichst wenige Ausreißer enthält
$bestModel$ → Parameter des Modells, welches die Punkte aus D am besten fittet

$i \leftarrow 0$
$s \leftarrow 1$
$bestModel \leftarrow \emptyset$
$inliers \leftarrow \emptyset$
$bestScore \leftarrow 0$
while $i \leq s$ **or** $i \leq k$ **do**
    $U \leftarrow$ n zufällig ausgewählte Punkte aus der Datenmenge D
    $model \leftarrow$ FITMODELTOPOINTS(U)
    $inliers \leftarrow \emptyset$
    for all $p \in D$ **where** $p \notin U$ **do**
        if p passt zu $model$ mit einem Fehler f **and** $f \leq t$ **then**
            Füge p zu $inliers$ hinzu
        end if
    end for
    $numInliers \leftarrow$ GETSIZE($inliers$)
    if $numInliers > bestScore$ **then**
        $bestScore \leftarrow numInliers$
        $bestModel \leftarrow model$
        $totalNumPoints \leftarrow$ GETSIZE(D)
        $\epsilon \leftarrow \frac{numInliers}{totalNumPoints}$
        $p \leftarrow 1 - \epsilon^n$
        $s \leftarrow \frac{\log(1-p)}{log(p)}$
    end if
    $i \leftarrow i + 1$
end while

2.6 Kalman-Filter

Die Fahrbahnerkennung soll robust gegen Ausreißer und verrauschte Bilder sein, wie in Unterabschnitt 1.2 schon beschrieben wurde. Dafür ist es sehr hilfreich ein sogenanntes *Tracking* der Fahrbahnmarkierungen zu implementieren. Dabei soll die Position der Fahrbahnmarkierungen im Bild F_i gespeichert werden. Anschließend ist anhand dieser Information die Position der Markierungen im nächsten Bild vorauszusagen. Somit ist

es möglich eine Korrektur im Bild F_{i+1} durchzuführen. Der große Vorteil dieses Verfahrens ist die Verbesserung der Ergebnisse und die Erkennung der Fahrbahnmarkierungen auch auf verrauschten Bildern. Zur Implementierung des Trackings kommt häufig in der Literatur der Kalman-Filter zum Einsatz. Dieser Filter wird im Folgenden vorgestellt.

Der Kalman-Filter wurde erstmals 1960 in [Kal60] eingeführt. Dieser Filter ermöglicht die Verwendung einer linearisierten Version der Systemdynamik, um optimale Schätzungen eines Prozesses unter der Annahme von gaußschem Rauschen zu generieren. Der Kalman-Filter schätzt den Prozesszustand mit Hilfe von einer Art Regelung. Der Filter schätzt den Prozesszustand zu einem bestimmten Zeitpunkt und erhält anschließend Feedback in Form von Messungen. Daher lassen sich die Gleichungen des Kalman-Filters in zwei Gruppen aufteilen. Nämlich Zustandsgleichungen und Beobachtungsgleichungen. Zum einen sind die Zustandsgleichungen verantwortlich für die Vorwärtsprojektion (in der Zeit) von dem aktuellen Prozesszustand sowie von der Kovarianzmatrix. Zum anderen sind die Beobachtungsgleichungen verantwortlich für das Feedback. Mit anderen Worten sind die Zustandsgleichungen für die Prädiktion des Prozesses und die Beobachtungsgleichung für seine Korrektur gedacht. Somit basiert der Algorithmus des Kalman-Filters auf einem Prädiktion-Korrektur-Verfahren, um numerische Probleme zu lösen.

Bei einem üblichen Kalman-Filter sind die genannten Zustandsgleichungen mit der einfachen Gleichung 13 definiert. Darüber hinaus sind die Beobachtungsgleichungen mit der Gleichung 14 definiert.

$$X_t = A_{t-1}X_{t-1} + B_{t-1}u_{t-1} + \epsilon_{t-1} \tag{13}$$

$$Z_t = HX_t + \delta_t \tag{14}$$

In der Zustandsgleichung ist A eine $n\ x\ n$ Matrix, die die Übergänge zwischen zeitlich aufeinanderfolgenden Zuständen X_{t-1} und X_t beschreibt. Die Matrix B setzt das Systeminput u_{t-1} in Beziehung zu dem Zustand X_t. Der Term ϵ_{t-1} stellt das Prozessrauschen dar.

In der Beobachtungsgleichung ist H die sogenannte Beobachtungsmatrix, während δ_t das Messrauschen darstellt.

In dem Ansatz dieser Arbeit entsprechen die Messungen der Position des Schwerpunktes einer erkannten Fahrbahnmarkierung. Aus diesem Grund wird der Prozesszustand X_t mit den x- und y- Koordinaten des Schwerpunktes sowie mit den zugehörigen Geschwindigkeiten v_x und v_y modelliert, wie es in Gleichung 15 zu sehen ist.

$$X_t = \begin{pmatrix} x_t \\ y_t \\ v_{x;t} \\ v_{y;t} \end{pmatrix} = \begin{pmatrix} x_{t-1} + v_{x;t-1}\ \Delta t + \frac{1}{2}a_x\Delta t^2 \\ y_{t-1} + v_{y;t-1}\ \Delta t + \frac{1}{2}a_y\Delta t^2 \\ v_{x;t-1} + a_x\Delta t \\ v_{y;t-1} + a_y\Delta t \end{pmatrix} \tag{15}$$

In diesem Ansatz wird das Prozessrauschen ϵ_t mit den unbekannten Beschleunigungskomponenten modelliert. Außerdem ist hier $u_{t-1} = 0$, weil es kein Systeminput vorhanden ist. Unter diesen Annahmen und durch Koeffizientenvergleich zwischen Gleichung 13 und Gleichung 15 erhält man die Gleichung 16.

$$X_t = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} X_{t-1} + \begin{pmatrix} \frac{1}{2} a_x \Delta t^2 \\ \frac{1}{2} a_y \Delta t^2 \\ a_x \Delta t \\ a_y \Delta t \end{pmatrix} \tag{16}$$

Aus Gleichung 16 ergibt sich die Matrix A und der Term ϵ_t, wie es in Gleichung 17 zu sehen ist.

$$A_t = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \epsilon_t = \begin{pmatrix} \frac{1}{2} a_x \Delta t^2 \\ \frac{1}{2} a_y \Delta t^2 \\ a_x \Delta t \\ a_y \Delta t \end{pmatrix} \tag{17}$$

Um aus X_t nur die Messungen x und y zu erhalten, ist die Beobachtungsmatrix H zu bestimmen. Für diesen Ansatz ist H in Gleichung 18 definiert.

$$H_t = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{18}$$

Die Prädiktion und Korrektur der Position des Schwerpunktes anhand der Zustands- und Beobachtungsgleichungen wird zwar in dieser Arbeit verwendet, aber die ausführliche Erklärung ist nicht relevant für das Verständnis dieses Ansatzes. Der geneigte Leser kann gerne im Buch „Probabilistic Robotics“ nachlesen [TFB05, S. 34–48].

Im Bild befinden sich normalerweise mehrere Fahrbahnmarkierungen. Dabei stellt sich die Frage, welche Fahrbahnmarkierung welche Messung generiert hat. Dieses Problem ist das sogenannte Zuordnungsproblem. Eine Methode zur Lösung dieses Problems wird im Unterabschnitt 2.7 vorgestellt.

2.7 Ungarische Methode

Bisher wurde das Tracking einer Fahrbahnmarkierung mit Hilfe vom Kalman-Filter erklärt. Jedoch befinden sich in jedem Bild F_i normalerweise mehrere Fahrbahnmarkierungen. Aus diesem Grund sind mehrere Instanzen des Kalman-Filters zu benutzen, um alle Fahrbahnmarkierungen gleichzeitig zu verfolgen. Dabei stellt sich die Frage, welche Fahrbahnmarkierung im Bild F_{i+1} zu welchem Kalman-Filter gehört. Diese Frage ist ein typisches Beispiel für ein lineares Zuordnungsproblem. Zur Lösung dieses Problems wird in der Literatur häufig die Ungarische Methode verwendet, weil die Lösungen dieser Problemklasse binär und damit insbesondere ganzzahlig sind. Außerdem existieren für die Ungarische Methode relativ schnelle Algorithmen mit einer Komplexität der Größenordnung $O(n^3)$. Diese Methode wird im Folgenden vorgestellt.

Die Ungarische Methode wurde erstmals 1955 von Kuhn in [Kuh55] eingeführt. Diese Methode basiert auf dem iterativen Suchen der optimalen Zuordnung mit Hilfe von einer sogenannten *Kostenmatrix*. Bei dieser Matrix entspricht jede Zeile einer Quelle und jede Spalte einem Ziel (oder umgekehrt). Dabei enthält jedes Matrixelement die Kosten bzw. die Bewertung der Zuordnung ihrer Quelle zu ihrem Ziel. Die Methode garantiert, dass nach $i \geq 0$ Iterationen die günstigste Zuordnung geliefert wird. Dafür werden in jeder Iteration jedes Zeilen- und Spaltenminimum der Kostenmatrix gesucht. Anschließend werden die Zeilen- und Spaltenelemente um das Zeilen- bzw. Spaltenminimum reduziert. Dieser Vorgang wird solange wiederholt, bis die optimale Zuordnung gefunden wird.

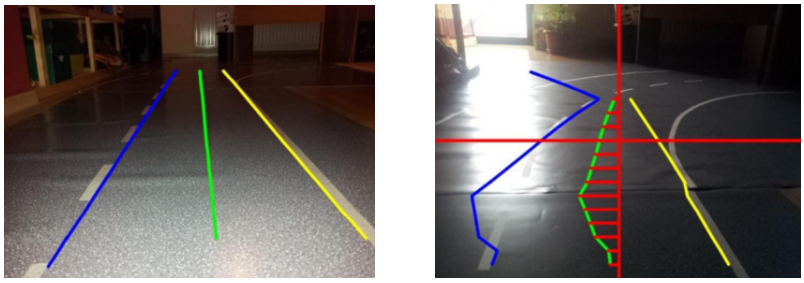
Im Ansatz dieser Arbeit sind die im aktuellen Bild erkannten Fahrbahnmarkierungen als die oben genannten Quellen zu betrachten, während die in vorherigen Bildern erkannten Fahrbahnmarkierungen als Ziele zu sehen sind.

2.8 Lösungsansätze

An der Technischen Universität Darmstadt wurde bereits von Handau ein Ansatz zur kamerabasierten Fahrbahnerkennung mit einem Modellfahrzeug entwickelt [Han15]. Dort werden die Kanten im Bild mittels Canny-Algorithmus hervorgehoben. Anschließend werden mit Hilfe von der Hough-Transformation vertikale Linien erkannt. Danach werden aus dem Bild zwei sogenannten ROIs (englisch Regions of Interest) gewählt. Diese ROIs entsprechen jeweils dem unteren und oberen Bereich der Fahrbahn. Anhand der erkannten Linien und durch das Fitten des Modells einer Gerade werden Abschnitte der Fahrbahnmarkierungen in jedem ROI approximiert. Im Anschluss daran werden beide ROIs zusammengefügt und aus den zuvor gewonnenen Informationen wird die Mittellinie der Fahrspur berechnet. Letztlich wird die Position der Fahrspurmitte gespeichert und für das Tracking von dem Fahrbahnverlauf verwendet. Bei dieser Tracking-Phase lassen sich laut Handau Lenkmanöver für die Fahrzeugregelung einleiten. Die Ergebnisse von der Handau's Umsetzung sind in Abbildung 2.6a zu sehen.

Der vorgestellte Ansatz von Handau ist jedoch mit verschiedenen Problemen verbunden. Zum einen ist der Ansatz nicht robust gegen Ausreißer und unregelmäßige Lichtintensität. Zum anderen ist diese Umsetzungsvariante aufgrund der einfachen Modellierung mit Geraden nicht für gekrümmte Fahrstrecken geeignet (siehe Abbildung 2.6b). Außerdem geht Handau in seiner Umsetzung davon aus, dass das Fahrzeug stets die richtige Fahrspur befährt. Aus diesem Grund erkennt Handau nur die nächsten zwei Fahrbahnmarkierungen zur Bildmitte. Dies verhindert die Erkennung von weiteren Fahrspuren und damit auch eventuelle Überholmanöver oder Korrekturen beim Fehlverhalten. Darüber hinaus besteht kein Zusammenhang zwischen der gewonnenen Position der Fahrspurmitte im Bild und ihrer tatsächlichen Position bezüglich des Fahrzeugs bzw. des Bodenkoordinatensystems. Dies erschwert eine passende Regelung zur automatischen Fahrspurführung.

Um diese Fehler zu vermeiden, soll einerseits eine andere Methode zur Kantendetektion verwendet werden, die robuster gegen unregelmäßige Lichtintensität ist. Andererseits sei hier auch notwendig, eine erweiterte Modellierung der Fahrbahnmarkierungen zu implementieren, die sich sowohl für gerade Fahrstrecke als auch für gekrümmte Abschnitte eignet. Darüber hinaus soll der Ansatz nicht auf die Erkennung der nächsten zwei Fahrbahnmarkierungen beschränkt werden, sondern auf die Erkennung von allen sichtbaren Fahrbahnmarkierungen. Dadurch müssen jedoch viele Instanzen des Modells in jedem Bild gefunden werden. Um die davor beschriebenen Anforderungen erfüllen



(a) Fahrbahnerkennung mit dem Ansatz von Handau bei einer geraden Fahrstrecke

(b) Verarbeitungsfehler vom Handau's Ansatz bei einer Kurve und unregelmäßiger Lichtintensität

Abbildung 2.6: Ergebnisse von Handau's Umsetzung [Han15]

zu können, basiert die Umsetzung dieser Arbeit hauptsächlich auf drei verschiedenen Ansätzen, die im Folgenden beschrieben werden.

Die ersten zwei Ansätze wurden 2015 von Läßig und Krüger an der Freien Universität Berlin entwickelt und werden in [Lä15] und in [Kr"15] beschrieben. Diese Arbeiten bauen sich aufeinander und wurden zwecks der Teilnahme am Carolo-Cup konzipiert. Bei beiden Arbeiten wird eine omnidirektionale Kamera verwendet, die ca. 20 cm hoch auf einem Modellfahrzeug platziert ist. Mit Hilfe von dieser Kamera wird eine Vogelperspektive von dem gesamten Bereich in einem Radius von etwa 1.5 m um das Fahrzeug gewonnen. Anschließend wird in beiden Arbeiten eine auf dem Sobel-Operator basierende Kantendetektion durchgeführt. Im Anschluss daran wird beim Läßig's Ansatz der *RANSAC*-Algorithmus im Zusammenspiel mit der Newtonschen Polynominterpolation auf drei verschiedene kleine Teilbereiche des Bildes angewendet, um nach Fahrbahnmarkierungen zu suchen. Im Krüger's Ansatz wird hingegen diese Suche nicht auf Teilbereiche beschränkt, sondern auf das gesamte Fläche vor dem Fahrzeug. Dabei werden mehrere Instanzen eines Modells mit Hilfe von *MultiRansac* gefunden. Nach diesen Schritten werden in beiden Ansätzen die gefundenen möglichen Fahrbahnmarkierungen durch Abstandsbestimmung und anschließenden Vergleich mit den Spezifikationen im Regelwerk des Carolo-Cups validiert. Letztlich wird die Fahrspurmitte berechnet und damit ein passendes Lenkmanöver eingeleitet.
In den Ansätzen von Läßig und Krüger lassen sich sowohl Vorteile als auch Nachteile feststellen. Diese werden im Folgenden erläutert. Zur Vereinfachung und aufgrund

der Ähnlichkeit zwischen beiden Ansätzen werden sie ab hier als ein einzelner Ansatz betrachtet.

Vorteile vom Läßig's und Krüger's Ansatz

- die Idee von der Validierung der erkannten Fahrbahnmarkierungen anhand bestimmter Spezifikationen ist interessant und lässt sich als sinnvoll für den Ansatz dieser Arbeit betrachten.
- Ein einfaches Tracking-Verfahren wird verwendet, um die Position der Fahrspurmitte zu verfolgen. Dies kann hilfreich sein, wenn eine Fahrbahnmarkierung nicht erkannt werden konnte.

Nachteile vom Läßig's und Krüger's Ansatz

- Aufgrund des begrenzten Sichtfeldes und tiefer Position der in dieser Arbeit verwendeten Kamera ist der Ansatz von Läßig und Krüger hier nicht direkt einsetzbar.
- Es werden zwar mehrere Fahrbahnmarkierungen mittels *MultiRansac* erkannt, aber das führt zu einer signifikanten Erhöhung der von RANSAC benötigten Anzahl an Iterationen. Aus diesem Grund kann durch die Nutzung von MultiRansac die Recheneffizienz reduziert werden.
- Durch Verwendung von einfachen Polynomen zweiten Grades werden einige Fahrbahnmarkierungen nicht ausreichend modelliert [Jen08, S. 9–10].

Der dritte Ansatz, der auch die Basis von dieser Arbeit bildet, ist der Ansatz von Aly. Diese Variante der Fahrbahnerkennung wurde 2008 an dem *Caltech Institute of Technology* entwickelt und wird in [Aly08] beschrieben. Aly entwickelte seine Umsetzung zwecks der Teilnahme am *DARPA Urban Challenge* und ist auf die Fahrbahnerkennung in städtischen Umgebungen spezialisiert. Aly verwendet eine übliche Farbkamera, aus deren Bildern mit Hilfe von der IPM-Methode eine Draufsicht erzeugt wird. Danach wird eine Kantendetektion auf die Draufsicht angewendet. Diese erfolgt mittels eines Hochpass- und eines Tiefpassfilters mit anschließender Threshold-Operation. Als nächstes werden durch die Hough-Transformation vertikale Liniensegmente in der Draufsicht detektiert. Diese werden als Hinweise für die Position möglicher Fahrbahnmarkierungen betrachtet. Im Anschluss daran wird in der unmittelbaren Nähe von jedem Hinweis nach Fahrbahnmarkierungen gesucht. Dies erfolgt mit Hilfe von *RANSAC* für Geraden und *RANSAC* für B-Splines, um die Robustheit gegen Ausreißer gewährzuleisten. Darüber hinaus werden dank dieses Verfahrens sowohl gerade als auch gekrümmte Fahrbahnmarkierungen in der Draufsicht gefunden.
In dem Ansatz von Aly lassen sich sowohl Vorteile als auch Nachteile feststellen. Diese werden im Folgenden erläutert.

Vorteile vom Aly's Ansatz

- Eine übliche Farbkamera wird verwendet.
- Robust gegen Ausreißer und unregelmäßige Lichtintensität.
- Erkennung mehrerer Fahrbahnmarkierungen.

Nachteile vom Aly's Ansatz

- Keine Validierung der erkannten Fahrbahnmarkierungen bzw. der Fahrspur anhand bekannten Abmessungen findet statt, wie es hingegen im Ansatz von Läßig und Krüger schon der Fall ist. Dies erhöht sowohl die Falsch-Positiv-Rate als auch das Fehlverhalten.
- Kein Tracking-Verfahren ist vorhanden. Aus diesem Grund kann der Ansatz zur Fahrbahnerkennung weder auf fehlende Fahrbahnmarkierungen noch auf Verarbeitungsfehler reagieren.
- Die Handhabung von B-Splines ist nicht einfach. Aus diesem Grund ist eine Nachbearbeitung der erkannten Fahrbahnmarkierungen kompliziert. Außerdem kann durch die Nutzung von B-Splines die Recheneffizienz reduziert werden.

2.9 Wahl des Ansatzes

Nach Vorstellung und Diskussion aller relevanten Lösungsansätze lässt sich im Rahmen dieser Arbeit die Wahl des Ansatzes feststellen. Es bietet sich an, die in den letzten Abschnitten beschriebenen Lösungsansätze teilweise zu reproduzieren und zu ergänzen. Da der Lösungsansatz von Läßig und Krüger aufgrund der verwendeten Kamera nicht direkt einsetzbar ist, wird in dieser Arbeit hauptsächlich der Lösungsansatz von Aly verwendet. Allerdings mit einigen Erweiterungen, die anschließend erläutert werden.

Ein Verfahren zur Validierung der Fahrspur wird implementiert, so ähnlich wie bei dem Ansatz von Läßig und Krüger. Da Polynome zweiten Grades in einigen Fällen nicht ausreichend zur Modellierung der Fahrbahnmarkierungen sind und die Handhabung von B-Splines nicht einfach ist, wird in dieser Arbeit eine Modellierung mit Polynomen 3. Grades realisiert. Das Tracking-Verfahren von Läßig und Krüger wird weiterverwendet, jedoch mit Erweiterungen, die Korrekturen sowie Vorhersagen ermöglichen.

Im nächsten Kapitel wird auf die tatsächliche Realisierung des gewählten Ansatzes näher eingegangen.

3 Umsetzung

In diesem Kapitel wird als erstes die für das Testen der Fahrbahnerkennung und Fahrbahnführung benutzte Testfahrbahn beschrieben. Als nächstes wird sowohl das verwendete Modellfahrzeug als auch die vorhandene Hardware vorgestellt. Danach werden die verwendete Softwarebibliotheken und Programme erläutert. Anschließend wird der Programmablaufplan vorgestellt. Im Anschluss daran wird auf jede Phase des Programmablaufplans näher eingegangen.

3.1 Testfahrbahn

Für das Testen im Rahmen dieser Arbeit wurde eine Testfahrbahn erstellt. Diese orientiert sich am Regelwerk vom Wettbewerb für autonomes Fahren *Carolo-Cup*. Daher besitzt die Testfahrbahn dieselben physikalischen Eigenschaften, die in Unterabschnitt 2.1 beschrieben sind. Darüber hinaus besteht der Fahrbahnuntergrund aus einem mattschwarzen PVC-Bodenbelag. Die Fahrbahnmarkierungen wurden mit weißem Isolierband auf diesem Material angebracht. Die Testfahrbahn besteht aus zwei Fahrstrecken. Bei der ersten handelt es sich um eine gerade Fahrstrecke, die etwa zwei Meter lang ist. Bei der zweiten handelt es sich hingegen um eine gekrümmte Fahrstrecke mit einem Kurvenradius von ca. einem Meter. Somit entspricht die Kurve von der Testfahrbahn der engsten Kurve bei dem Corolo-Cup. Dies wurde absichtlich gemacht, um die Umsetzung dieser Arbeit auch unter herausfordernden Situationen zu testen. An dieser Stelle ist noch zu erklären, dass die verwendete Testfahrbahn im Vergleich zu der gesamte Fahrstrecke im Carolo-Cup sehr klein ist. Allerdings war diese für die Durchführung kleiner Teste ausreichend. Die verwendete Testfahrbahn ist in Abbildung 3.1 zu sehen.

Abbildung 3.1: Testfahrbahn

3.2 Hardware

In den nächsten Unterabschnitten werden die für die Umsetzung dieser Arbeit relevanten Hardware-Komponenten beschrieben. Als erstes wird das verwendete Modellfahrzeug vorgestellt. Danach werden die Eigenschaften vom benutzten Mikrocontroller und

Mainboard erläutert. Als letztes erfolgt eine kurze Beschreibung der Kamera, die zur Fahrbahnerkennung verwendet wird.

3.2.1 Modellfahrzeug

Um der Ansatz dieser Arbeit in der Praxis zu testen, wurde das Modellfahrzeug aus Abbildung 3.2 verwendet. Dieses wurde für das Projektseminar Echtzeitsysteme an der Technischen Universität Darmstadt angefertigt. Das Chassis sowie die Lenkung, der Motorstrang und die Motorregelung stammen von einem Modellbau der japanischen Firma Tamiya. Mit dem Modellfahrzeug ist eine maximale Geschwindigkeit von etwa 1 m/s und ein minimaler Lenkradius von ca. 90cm zu erreichen.
Das Modellfahrzeug wurde im Rahmen des Projektseminars zur Erarbeitung verschiedener Projekte mit einem 3200mAh LiFe-Akku, einer Kinect, einem Mainboard, einem Mikrocontroller und verschiedenen Sensoren ausgerüstet. Allerdings werden im Rahmen dieser Arbeit nur einige Komponenten verwendet. Diese werden in den nächsten Abschnitten erläutert.

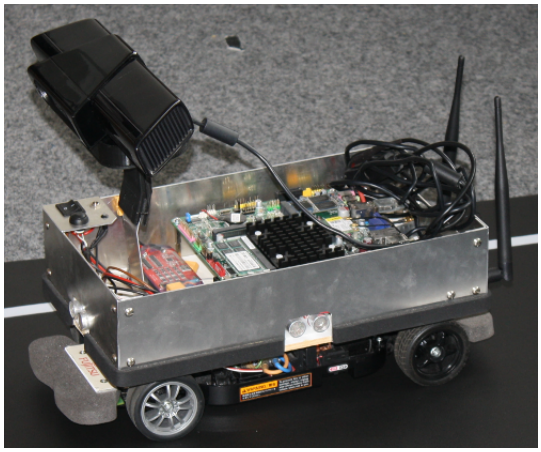


Abbildung 3.2: Verwendetes Modellfahrzeug

3.2.2 Mikrocontroller und Mainboard

Auf dem Modellfahrzeug ist sowohl ein Mikrocontroller als auch ein Mainboard vorhanden. Bei dem Mikrocontroller handelt es sich um das Modell MB96300 der Firma Fujitsu. Dieser ist für die Ansteuerung der Lenkung und des Antriebs verantwortlich und wird über einen USB-Anschluss an das Mainboard angeschlossen.
Bei dem verwendeten Mainboard handelt es sich um ein eingebettetes Board der Serie „PD10BI-MT ThinMini-ITX“ von der Firma MiTAC. Dieses verfügt über einen Intel Quadcore-Prozessor, der mit 2.0 GHz getaktet wird und eine einfache Onboard-Grafikkarte enthält. Darüber hinaus ist das Mainboard mit 8GB Arbeitsspeicher und einer 60GB großen SSD-Festplatte ausgerüstet. Zudem sind zwei 2.0 USB- und zwei 3.0 USB-Anschlüsse vorhanden. Zusätzlich bietet das Mainboard einen 1Gbit/s Ethernet-Anschluss sowie einen Wireless Adapter, der an zwei externe Antennen angeschlossen ist. Die Stromversorgung erfolgt mit Hilfe von dem installierten LiFe-Akku.

3.2.3 Kamera

Bei der in dieser Arbeit verwendeten Kamera handelt es sich um die erste Version der Kinect. Diese wurde 2010 von der Firma Microsoft eingeführt und besteht aus einer Tiefen- und Farbkamera. Zum einen hat die Tiefenkamera eine Auflösung von 320x240 Pixeln. Zum anderen beträgt die Auflösung der Farbkamera 640x480 Pixel. Beide Kameras haben laut dem Hersteller ein vertikales Sichtfeld von 43 Grad und ein horizontales Sichtfeld von 57 Grad. Die Nutzung der Kinect bot sich an, da diese auf dem Modellfahrzeug schon vorhanden war. Allerdings wird im Rahmen dieser Arbeit nur die Farbkamera der Kinect verwendet.
Im Laufe der Umsetzung ließ sich feststellen, dass das horizontale Sichtfeld der Kamera für die Fahrbahnerkennung nicht ausreichend war. Bei der Kurve in der Testfahrbahn war nur eine der drei Fahrbahnmarkierungen aufzunehmen. Aus diesem Grund wurde eine Weitwinkel-Linse der Firma Nyko an der Kinect angebracht. Diese soll laut dem Hersteller das Sichtfeld der Kamera um 40% vergrößern. Mit Hilfe von der angebrachten Linse war es danach möglich, bei der Kurve zwei der drei Fahrbahnmarkierungen aufzunehmen. Dieses letztes Verhalten stellt für die Fahrbahnerkennung und die anschließende Fahrbahnführung im Ansatz dieser Arbeit keine Problematik dar.
Die Kinect ist mit einer Halterung auf dem Modellfahrzeug montiert und hat einen Abstand von 32,5cm zum Boden. Darüber hinaus beträgt der Neigungswinkel der Kamera etwa 27 Grad zum Boden. Zudem werden von der Kinect 30 Bilder pro Sekunde aufgenommen, die über ein USB-Kabel an das Mainboard des Fahrzeugs übertragen werden. Die Kinect mit der angebrachten Linse ist in Abbildung 3.3 zu sehen.

Abbildung 3.3: Kinect mit Weitwinkel-Linse

3.3 Software

In den folgenden Abschnitten wird als erstes die Entwicklungsumgebung und die im Rahmen dieser Arbeit verwendete Software beschrieben. Im Anschluss daran werden die relevanten Annahmen und Voraussetzungen für die korrekte Funktionalität des Algorithmus dieser Arbeit erläutert. Danach wird der Programmablaufplan von dem implementierten Algorithmus für die Fahrbahnerkennung und automatische Fahrbahnführung vorgestellt. Als letztes wird auf die Implementierung des Algorithmus näher eingegangen. Dabei erfolgt eine Erläuterung jeder Phase des Programmablaufplans.

3.3.1 Entwicklungsumgebung und verwendete Software

Zur Implementierung des Ansatzes dieser Arbeit wurde das in Unterabschnitt 3.2.2 vorgestellte Mainboard verwendet. Auf dem Mainboard wurde eine Version des Linux-Betriebssystems *Lubuntu* installiert. Diese stellt eine kompakte und schnelle Variante vom Betriebssystem *Ubuntu* 14.04 dar. Darüber hinaus erfolgt die Programmierung dieser Umsetzung hauptsächlich in *C++* und teilweise auch in C. Zudem wird der Code durch die GCC und G++ Compiler kompiliert.

Im Rahmen der Umsetzung wurde die Version *Indigo* von dem Software-Framework *ROS* verwendet. ROS steht für ***R**obotic **O**perating **S**ystem* und wurde für die Roboterprogrammierung entwickelt. ROS ist laut der Onlinereferenz[ROS14] frei zugänglich und wird ständig gepflegt und weiterentwickelt. Darüber hinaus bietet ROS zahlreiche open source Bibliotheken, Werkzeugen und Gerätetreiber. Diese werden als ROS-Pakete bezeichnet und ermöglichen die Entwicklung von Robotern und damit verwandten Projekten. Jedes ROS-Paket enthält einen oder mehrere der sogenannten Nodes. Bei diesen handelt es sich prinzipiell um einzelne Programme bzw. Prozesse, die entweder in C++, *Python* oder *Lisp* geschrieben wurden und die mit dem ROS-Framework gestartet werden können. Zudem wird mit Hilfe von ROS eine Art Netzwerk aufgebaut, über das alle Nodes miteinander kommunizieren können. Diese Kommunikation erfolgt mit Nachrichten von unterschiedlichen ROS-Objekten, die durch Busse übertragen werden. Diese Busse werden als ROS-Topics bezeichnet und können von jedem Node veröffentlicht und abonniert werden. In diesem Zusammenhang operiert ein Node, der eine Nachricht über ein Topic sendet, als *Publisher*. Ein Node, der hingegen eine Nachricht empfängt, agiert als *Subscriber*.

Um die Implementierung verschiedener Funktionen und Algorithmen aus der Bildverarbeitung zu erleichtert, kommt im Rahmen dieser Arbeit die freie Programmbibliothek *OpenCV* zum Einsatz. Diese unterstützt laut der Onlinereferenz [Ope16] die Entwicklung von Software mit zahlreichen Algorithmen für die Bildverarbeitung und maschinelles Sehen. Diese Algorithmen sind frei zugänglich und werden für die Programmiersprachen C und C++ vollständig und effizient entwickelt.

Zur Umsetzung von RANSAC wurde eine frei zugängliche Vorlage des Algorithmus verwendet. Diese wurde von der Programmbibliothek ***M**obile **R**obot **P**rogramming **T**oolkit (MRPT)* zur Verfügung gestellt. MRPT ist laut der Onlinereferenz [MRP16] quelloffen und bietet zahlreichen C++ Algorithmen für die Roboterprogrammierung, Computer Vision, Trajektorienplanung sowie für das Mapping mit Robotern.
Die verwendete Vorlage implementiert den RANSAC-Algorithmus mit adaptiver Bestimmung der Parameter. Diese Variante wurde in Unterabschnitt 2.5 beschrieben. Darüber hinaus wird die genannte Vorlage in der Umsetzung dieser Arbeit durch die Nutzung von Spline-Interpolation auf die Modellierung von Fahrbahnmarkierungen spezialisiert. Diese Interpolation-Methode wurde wiederum anhand vorgefertigter Funktionen der Programmbibliothek MRPT implementiert.

Wie in Unterabschnitt 2.9 schon beschrieben wurde, basiert der Ansatz dieser Arbeit auf die Arbeiten von Aly, Krüger und Läßig. Der erste genannte Autor veröffentlichte in [Aly11a] seine Implementierung von Fahrbahnerkennung. Diese ist frei erhältlich und ist in C geschrieben. Mit freundlicher Genehmigung von Aly wurden im Rahmen dieser Arbeit einige Codeabschnitte seiner Implementierung weiterverwendet. Allerdings wurde die Mehrheit der verwendeten Abschnitte angepasst und erweitert, um sowohl die Integration mit dem ROS-Framework als auch die Fahrbahnerkennung mit Modellfahrzeugen zu ermöglichen. Darüber hinaus wurde zur Evaluation der Umsetzung dieser Arbeit ein von Aly entwickeltes *Matlab-Tool* verwendet. Dieses ist ebenfalls frei zugänglich und wurde in [Aly11b] veröffentlicht.

3.3.2 Annahmen und Voraussetzungen

Bei der durch diese Arbeit vorgestellten Variante zur Fahrbahnerkennung und Fahrbahnführung betrachtet man die Fahrbahnmarkierungen als zusammenhängende Kanten, für die bestimmte Eigenschaften gelten. Einige Annahmen und Voraussetzungen werden für die Eingrenzung dieser Eigenschaften sowie für die Gewährleistung der erwarteten Funktionalität des Algorithmus dieser Arbeit benötigt. Im Folgenden werden die genannten Annahmen und Voraussetzungen beschrieben.

Eine wichtige Voraussetzung ist eine einmalige Kalibrierung der verwendeten Kamera vor der ersten Nutzung des Algorithmus. Dies kann mit Hilfe von OpenCV realisiert werden, wie im Buch „*Learning OpenCV: Computer Vision with the OpenCV Library*“ beschrieben wird [BK08]. Eine Vorkalibrierung der Kamera ermöglicht die Verwendung von der IPM-Technik und die Entzerrung der Bilder. Darüber hinaus wird auch für die richtige Funktionalität von der IPM-Technik vorausgesetzt, dass die Kamera fest positioniert ist. Dabei soll die durch die Fahrdynamik verursachte Bewegung der Kamera vernachlässigbar sein. Außerdem soll die Kamera so positioniert sein, dass mindestens zwei Fahrbahnmarkierungen in den Aufnahmen zu sehen sind.

Für die Kantendetektion wird ein dunkler Fahrbahnuntergrund mit hellen Fahrbahnmarkierungen angenommen. Diese Annahme ist angelehnt an den zuvor vorgestellten Wettbewerb für autonomes Fahren *Carolo-Cup*.

Es wird auch angenommen, dass die physikalischen Eigenschaften und Abmessungen der Fahrbahn bekannt und gleichbleibend sind. Dies wird für die Validierung von den erkannten Fahrbahnmarkierungen und der Fahrspur benötigt. Darüber hinaus ist zur Vermeidung von falsch-positiven Erkennungen und zur Reduzierung des Rechenaufwands die maximale Anzahl an zu erkennenden Fahrbahnmarkierungen über einen Parameter zu limitieren.

3.3.3 Programmablaufplan

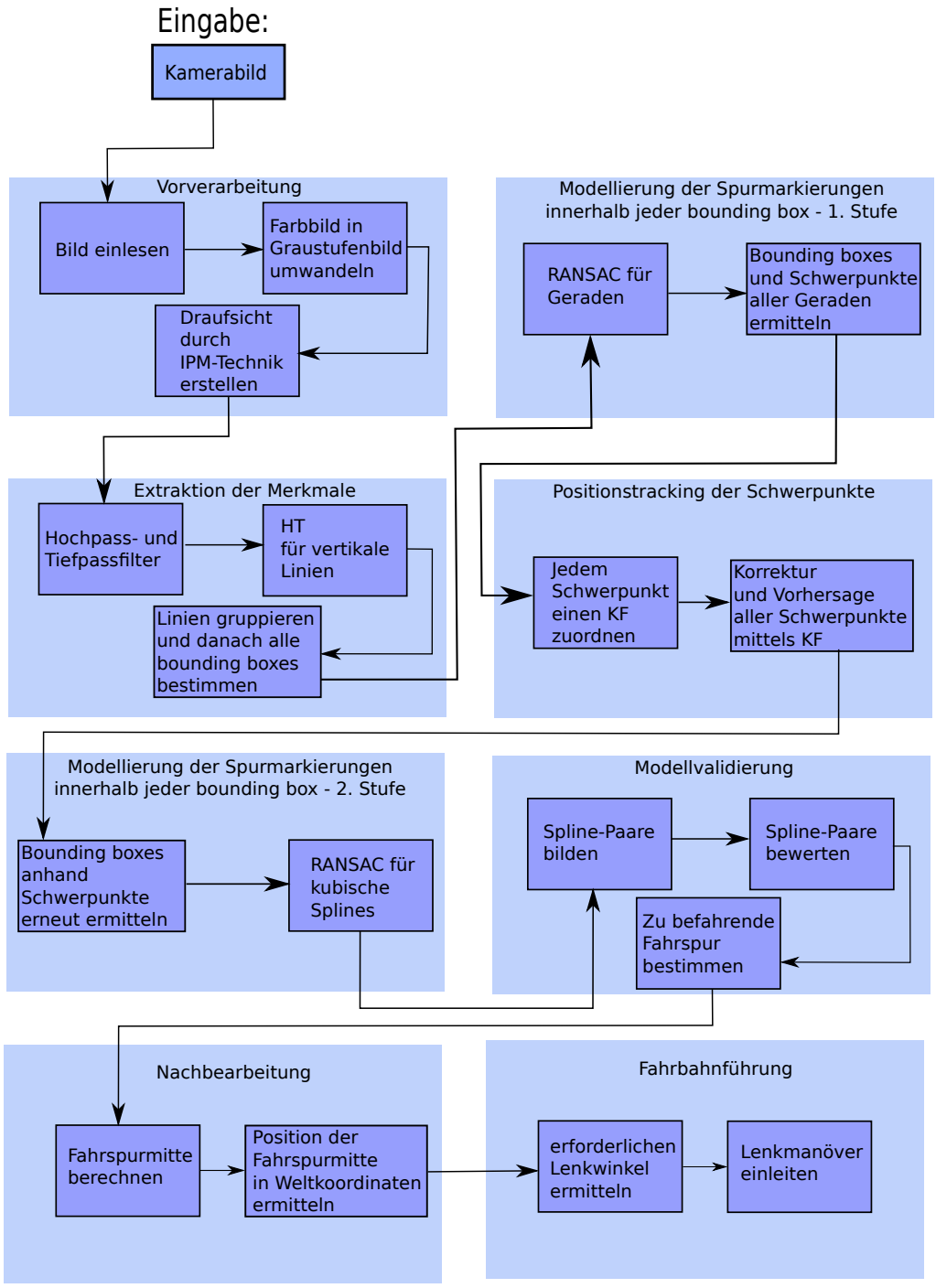


Abbildung 3.4: Programmablaufplan

In Abbildung 3.4 ist der Programmablaufplan von der Umsetzung dieser Arbeit zu finden. Das damit beschriebene Verfahren wird mit dem Empfang eines Bildes gestartet. Da die verwendete Kamera 30 Bilder pro Sekunde sendet, findet eine periodische Durchführung des Verfahrens statt.

Der Programmablaufplan wird in verschiedenen Phasen aufgeteilt. Eine allgemeine Vorstellung von den Phasen und der damit verbundenen Reihenfolge wird für das Verständnis der nächsten Abschnitten dieses Kapitels benötigt. Aus Platzgründen wurde ein paar Abkürzungen verwendet. HT steht für Hough-Transformation, KF für Kalman-Filter.

3.3.4 Vorverarbeitung

Am Anfang wird das von der Kinect aufgenommene Bild über ein ROS-Topic empfangen. Dieses Bild wird dank des ROS-Pakets *Freenect*[1] als ein Objekt des ROS-Datentyps *Image* bereitgestellt. Dieser Datentyp ist jedoch mit der hilfreichen Bibliothek für Bildverarbeitung *OpenCV* nicht kompatibel. Aus diesem Grund ist als erstes das empfangene Bild vom Typ Image in ein OpenCV-*Mat*-Objekt umzuwandeln. Dies erfolgt mit Hilfe vom ROS-Paket *cv_bridge*[2] und ermöglicht die Matrixdarstellung des Bildes und die weitere Verarbeitung entlang der Umsetzung dieser Arbeit. Das *Mat*-Objekt stellt eine zweidimensionale Matrix dar. Jedes Matrixelement entspricht einem Bildpunkt(Pixel) und enthält die Farbwerte (Rot, Grün, Blau).

Es wird nun geprüft, ob das Mat-Objekt entweder ein Graustufenbild oder ein Farbbild darstellt. Im letzteren Fall ist zuerst das Bild mittels OpenCV in ein Graustufenbild umzuwandeln. Dieser Schritt sei hier notwendig, da diese Darstellung die Fahrbahnerkennung vereinfacht. Das liegt daran, dass durch diese Informationsreduktion nur ein Farbkanal statt drei zu verarbeiten sind. Mit anderen Wörtern durch diese Umwandlung enthält jedes Matrixelement von *Mat* nur einen einzelnen Wert. Dies bedeutet sowohl weniger Speicherverbrauch als auch weniger Rechenaufwand.

Im Anschluss daran wird nun mit Hilfe von der IPM-Technik und den eigenen Kameraparametern eine Draufsicht eines ausgewählten Bildbereiches erzeugt. Diese Technik wurde in Unterabschnitt 2.2 beschrieben. Die resultierende Draufsicht wird in einem Mat-Objekt gespeichert, das entlang dieser Arbeit als *IPM-Bild* genannt wird. Ein IPM-Bild, das während eine Testfahrt aufgenommen wurde, ist in Abbildung 2.3c zu finden.

IPM kann effizient durchgeführt werden, indem man die Abbildungen zur Transformation (englisch Maps) sowie die Skalierungsfaktoren nur in der ersten Iteration berechnet und diese speichert. Aufgrund der statischen Position der Kamera sind die Abbildungen auch als statisch zu sehen und werden in den darauf folgenden Iterationen zur Erstellung der Draufsicht weiterverwendet. *IPM* bringt mit sich verschiedene Vorteile, die sowohl die Fahrbahnerkennung als auch die Fahrbahnführung erleichtern. Im folgenden werden die zwei wichtigsten Vorteile von IPM erläutert.

- Aufgrund der Perspektive sind die Fahrstreifen im Rohbild windschief und schneiden sich in einem gemeinsamen Fluchtpunkt. Dank der IPM-Technik werden nun die Fahrstreifen parallel zu einander so dargestellt, wie es in der Realität tat-

[1] http://wiki.ros.org/freenect_stack
[2] http://wiki.ros.org/cv_bridge

sächlich ist. Das kann in den nächsten Phasen der Umsetzung dieser Arbeit als Annahme sehen, um die Erkennung der Fahrstreifen zu erleichtern.

- Die Informationen im IPM-Bild werden durch die resultierenden Skalierungsfaktoren mit ihren tatsächlichen Positionen in der Welt (bezüglich des Fahrzeugs) in Zusammenhang gebracht. Dies ermöglicht eine Schätzung von Abständen und Größen im IPM-Bild. Das kann sehr hilfreich für die Validierung der erkannten Fahrspur und für die automatische Spurführung sein.

3.3.5 Extraktion von Merkmalen

In dieser Phase sind die Fahrbahnmarkierungen zu verschärfen und die nicht relevanten Informationen zu unterdrücken. Dafür wird als erstes das *IPM-Bild* anhand eines Kantenfilters gefiltert. Diese Art von Filtern wurden in Unterabschnitt 2.3 beschrieben. Der im Ansatz dieser Arbeit verwendete Kantenfilter besteht aus zwei Faltungsmatrizen, eine für die Filterung des Bildes in vertikaler Richtung und die Andere für die Filterung in horizontaler Richtung. Diese zwei Faltungsmatrizen sind vom Gauß-Filter abgeleitet und reagieren besonders gut auf Fahrbahnmarkierungen.
Die Faltungsmatrix für die Filterung in vertikaler Richtung $\boldsymbol{k_v}$ ist ein üblicher Gauß-Filter, d.h. ein Tiefpass, der mit Hilfe von Gleichung 19 bestimmt wird. Die Faltungsmatrix k_v hat die Dimension $r \times 1$, wobei $\boldsymbol{r}$ als ein einzustellender Parameter zu betrachten ist. Die genannte Faltungsmatrix wird über alle Spalten des IPM-Bildes geschoben und glättet das Bild in vertikaler Richtung, um das Rauschen zu reduzieren. Hierfür ist der Parameter σ_y so einzustellen, dass er der gewünschten Länge der erkannten Fahrbahnmarkierungen entspricht.

$$f_v(y) = e^{\left(-\frac{y^2}{2\sigma_y^2}\right)}, \text{ mit } \frac{-r+1}{2} \leq y \leq \frac{r-1}{2} \tag{19}$$

$$r \in \mathbb{N}, r \geq 3 \tag{20}$$

Die Faltungsmatrix für die Filterung in horizontaler Richtung $\boldsymbol{k_u}$ ist ein Hochpass, der mit Hilfe von Gleichung 21 bestimmt wird. Diese Faltungsmatrix ergibt sich aus eins minus der zweiten Ableitung eines Gauß-Filters und wird über alle Zeilen des IPM-Bildes geschoben. Darüber hinaus hat die Faltungsmatrix k_u die Dimension $w \times 1$, wobei $\boldsymbol{w}$ als ein einzustellender Parameter zu betrachten ist. Der resultierenden Hochpass reagiert auf Änderungen des Gradienten in horizontaler Richtung und eignet sich somit besonders gut dafür, die Fahrbahnmarkierungen hervorzuheben. Hierfür ist der Parameter σ_x so einzustellen, dass er der Breite der Fahrbahnmarkierungen entspricht.

$$f_u(x) = 1 - \frac{\partial^2}{\partial x^2} e^{\left(-\frac{x^2}{2\sigma_x^2}\right)} = \frac{1}{\sigma_x^2} e^{\left(-\frac{x^2}{2\sigma_x^2}\right)\left(1-\frac{x^2}{\sigma_x^2}\right)}, \text{ mit } \frac{-w+1}{2} \leq x \leq \frac{w-1}{2} \tag{21}$$

$$w \in \mathbb{N}, w \geq 3 \tag{22}$$

Entsprechend der verwendeten Testfahrbahn, ist in dieser Umsetzung die Parameter des Filters folgendermaßen eingestellt. $r = 5$, $w = 7$, $\sigma_y = 500\,\text{mm}$ und $\sigma_x = 20\,\text{mm}$. Das Ergebnis von der Filterung eines IPM-Bildes der Testfahrbahn ist in Abbildung 3.5a zu sehen.

Nach der vorherigen Filterung wird versucht, das übrige Rauschen sowie die nicht relevanten Informationen zu entfernen. Dies erfolgt anhand einer sogenannte *Threshold-Operation*. Dabei werden alle Pixel des IPM-Bildes unter einem Schwellenwert φ zu null gesetzt. Der Schwellenwert φ ist im Ansatz dieser Arbeit in Form von einem Quantil einzustellen und wurde durch Testen bestimmt. Die Darstellung des Schwellenwertes als ein Quantil und nicht als ein fester Grauwert ermöglicht eine dynamische Anpassung in Situationen mit unregelmäßigen Lichtbedingungen. In diesem Ansatz sei der Schwellenwert φ mit dem $0,9875$-Quantil definiert. Das bedeutet, dass nur die Pixel vom IPM-Bild mit einem Grauwert größer als $98,75\%$ aller Grauwerte im Bild behalten werden. Das Ergebnis vom gefilterten IPM-Bild nach der Threshold-Operation ist in Abbildung 3.5b zu sehen. Im Anschluss an die Threshold-Operation kommt eine einfache Version der

(a) Durch den Hochpass und Tiefpass gefiltertes IPM-Bild

(b) Gefiltertes IPM-Bild nach der Threshold-Operation

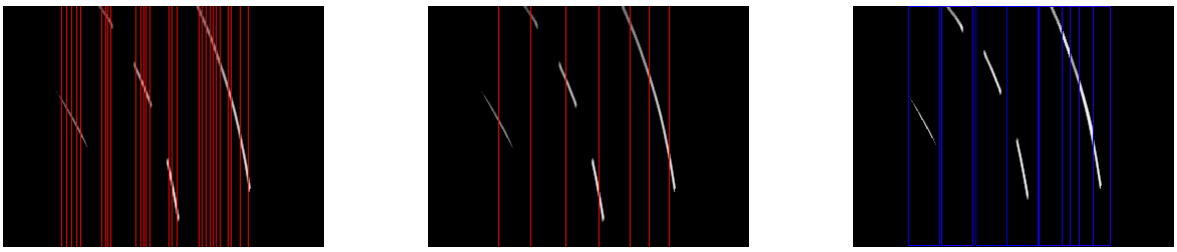
Abbildung 3.5: Filterung und Thresholding des IPM-Bildes

Hough-Transformation zum Einsatz. Dabei wird das IPM-Bild bzw. das *Mat*-Objekt spaltenweise durchgelaufen. In jeder Iteration werden alle Grauwerte in der i-ten Spalte aufaddiert und das Ergebnis in einer Variable gespeichert. Nach dem Durchlauf werden lokale Maxima bezüglich der gespeicherten Ergebnisse bestimmt. Nun werden vertikale Linien an der Stelle von jedem lokalen Maximum platziert (siehe Abbildung 3.6a). Diese Linien sind die sogenannten *Hough-Lines* und geben Hinweis dafür, wo sich Fahrbahnmarkierungen möglicherweise befinden. Dies wird in der nächsten Phase dieser Umsetzung ausgenutzt, um RANSAC in bestimmten Bildbereichen anzuwenden. Damit werden mehrere Fahrbahnmarkierungen erkannt, ohne dass man im ganzen Bild danach suchen muss. Dafür sind allerdings in der aktuellen Phase noch einige Schritte erforderlich. Diese werden im Folgenden beschrieben.

Wie es in Abbildung 3.6a zu sehen ist, können durch die Hough-Lines mehrere Hinweise auf die Position jeder Fahrbahnmarkierung entstehen. Um die Uneindeutigkeit der Position zu reduzieren und damit mehrere Antworten auf dieselbe Fahrbahnmarkierung zu vermeiden, ist eine Gruppierung der Hough-Lines durchzuführen. Bei dieser Gruppierung werden durch Mittelwertbildung alle Linien, deren Abstand voneinander

kleiner als ein Schwellenwert ist, zu einer einzelnen Linie zusammengefasst. Siehe Abbildung 3.6b. Der genannte Schwellenwert wurde im Rahmen dieser Arbeit durch Testen bestimmt.

Als nächstes wird in dieser Phase ein Suchbereich zur Anwendung des RANSAC-Algorithmus an der Stelle jeder resultierenden Linie definiert. Dabei entspricht die Länge jedes Suchbereiches der Höhe des IPM-Bildes während dessen Breite durch einen Parameter bestimmt wird. Dieser wurde im Ansatz dieser Arbeit durch Testen ermittelt und beträgt 30 Pixel. Die beschriebenen Suchbereiche werden in Form mehrerer *Rect*-Objekte definiert und gespeichert (siehe Abbildung 3.6c). Anschließend werden diese zur Modellierung der Fahrbahnmarkierungen an die nächste Phase weitergegeben. In den folgenden Abschnitten dieser Arbeit werden die Suchbereiche als *bounding boxes* bezeichnet.



(a) Hough-Lines bei einer gekrümmten Fahrstrecke **(b)** Nach der Gruppierung resultierende Linien **(c)** Suchbereiche bzw. Bounding boxes

Abbildung 3.6: Hough-Lines und die daraus resultierenden bounding boxes im IPM-Bild

3.3.6 Modellierung der Spurmarkierungen - 1. Stufe

Nach Bestimmung der bounding boxes werden diese als Masken für das IPM-Bild benutzt. Dabei wird das IPM-Bild in kleinere Bildbereiche zerschnitten, deren Größe einer bounding box entspricht. Diese Teilbereiche des Bildes werden in verschiedenen *Mat*-Objekte gespeichert. Im Anschluss daran wird versucht, in jedem Teilbereich eine Gerade zu fitten. Dies erfolgt mit Hilfe vom RANSAC-Algorithmus im Zusammenspiel mit der mathematischen Methode *Linear Least Squares*. Dabei werden n Pixel aus der Menge aller nicht schwarzen Pixel im Teilbereich zufällig gewählt. Mit den gewählten Pixeln wird eine Gerade gebildet. Anschließend wird bewertet, wie gut diese Gerade die Menge der restlichen nicht-schwarzen Pixel fittet. Dafür werden die durch die Gerade getroffenen nicht-schwarzen Pixel gezählt. Mit anderen Wörtern wird die Anzahl an *inliers* (deutsch Einlieger) bestimmt. Die Anzahl an inliers wird als Kriterium für die Bewertung der gebildeten Gerade verwendet und wird zum weiteren Vergleich zusammen mit der Gerade in einer Variable gespeichert. Dieses Verfahren wird solange wiederholt, bis entweder die gebildete Gerade eine minimale Anzahl an inliers enthält oder bis eine bestimmte Anzahl an Iterationen erreicht wurde. Diese zwei Parameter sowie die Anzahl an zu wählenden Pixeln wurden im Rahmen dieser Arbeit durch Testen bestimmt. Am Ende des Algorithmus wird die bestbewertete Gerade in Form von zwei *Point*-Objekten

zusammen mit der zugehörigen Bewertung zurückgegeben. Eine Beschreibung des genannten Algorithmus ist in Unterabschnitt 2.5 zu finden.

Im Anschluss daran werden die Ergebnisse von der Anwendung von RANSAC in allen Teilbereichen vereinigt. Falls die Gesamtzahl an erkannten Geraden den Parameter von der maximalen Anzahl an zu erkennenden Fahrbahnmarkierungen übersteigt, werden die Geraden mit schlechtesten Bewertungen ignoriert und verworfen. Dies führt zu einer Verminderung von falsch-positiven Erkennungen sowie zu einer Reduzierung der Rechenaufwand in den nachfolgenden Phasen.
Das vereinigte Ergebnis von der Anwendung von RANSAC in allen Teilbereichen ist in Abbildung 3.7 zu sehen. Wie man hier erkennen kann, werden die Fahrbahnmarkierungen durch die resultierenden Geraden gut approximiert. Allerdings werden diese im Fall einer gekrümmten Fahrstrecke nur tangential approximiert. Dadurch entstehen mehrere Antworten auf dieselbe Markierung aufgrund der für diesen Fall ungenügenden Modellierung. Dieses Verhalten lässt sich in Abbildung 3.8a beobachten. Um dieses Problem zu lösen, ist eine Verfeinerung der Modellierung der Fahrbahnmarkierungen mit Hilfe von komplexeren Polynomen durchzuführen. Dies wird im weiteren Verlauf der Umsetzung dieser Arbeit anhand von RANSAC für kubische Splines realisiert. Als nächstes

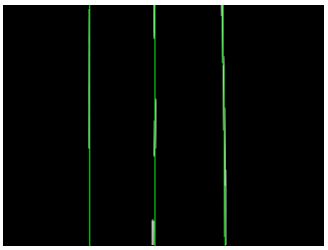
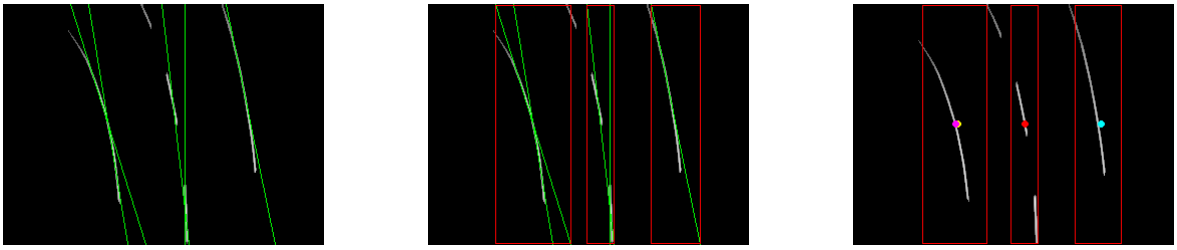


Abbildung 3.7: Die von RANSAC für Geraden gefundenen Modellinstanzen bei einer geraden Fahrstrecke.

sind in der aktuellen Phase dieser Umsetzung noch einige Schritte erforderlich, um die Verarbeitung in den nächsten Phasen zu erleichtern. Diese werden im Folgenden erläutert. Als erstes sind nach der Anwendung von RANSAC die bounding boxes aller resultierenden Geraden zu bestimmen. Dafür wird jede Gerade mit einem Rechteck umschlossen. Falls es Gruppen von sich überlappenden Rechtecken gibt, wird jede von diesen zu einem einzelnen Rechteck zusammengefasst. Dieser Schritt erfolgt analog zu der Gruppierung der Hough-Lines in der vorherigen Phase dieser Umsetzung. Jedes resultierende Rechteck stellt eine bounding box dar. Die daraus resultierenden bounding boxes sind in Abbildung 3.8b zu sehen. Im Anschluss daran werden die Schwerpunkte aller bounding boxes bestimmt und als *Point*-Objekte gespeichert. Diese sind in Abbildung 3.8c zu erkennen. Als letztes werden die Schwerpunkte bzw. *Point*-Objekte zum Positionstracking an die nächste Phase weitergegeben.



(a) Die von RANSAC für Geraden gefundenen Modellinstanzen bei einer gekrümmten Fahrstrecke.

(b) Resultierende bounding boxes nach Gruppierung der einzelnen Geraden

(c) Bounding boxes mit ihren jeweiligen Schwerpunkten (Kreise im Bild).

Abbildung 3.8: Ergebnis von RANSAC für Geraden und die daraus resultierenden bounding boxes und Schwerpunkte

3.3.7 Positionstracking

In dieser Phase kommt ein Tracking-Verfahren zum Einsatz. Dabei werden die in der letzten Phase ermittelten Schwerpunkte über mehrere aufeinanderfolgenden Bilder verfolgt, um die Korrektur und Vorhersage der Position der Schwerpunkte zu ermöglichen. Dies erlaubt wiederum die Erkennung von Fahrbahnmarkierungen durch die nächste Phase, auch wenn Verarbeitungsfehler bei den letzten zwei Phasen aufgetaucht sind. Das beschriebene Tracking-Verfahren erfolgt anhand mehrerer Kalman-Filter im Zusammenspiel mit der Ungarischen Methode. Dabei ist jeder Kalman-Filter für die Korrektur und Vorhersage der Position eines Schwerpunktes verantwortlich während die Ungarische Methode bei der richtigen Zuordnung von Kalman-Filtern hilft. Dadurch wird in der Iteration i jeder der erkannten Schwerpunkte mit einem Kalman-Filter assoziiert. In der Iteration $i+1$ antwortet die Ungarische Methode auf die Frage, welcher Schwerpunkt aus dieser neuen Iteration zu welchem Kalman-Filter aus der vorherigen Iteration gehört.
Der Kalman-Filter sowie die genannte Methode wurden jeweils in Unterabschnitt 2.6 und Unterabschnitt 2.7 vorgestellt. Im Folgenden wird auf die Implementierung des Tracking-Verfahrens näher eingegangen.

Als erstes wird jedem Schwerpunkt mit Hilfe von der Ungarischen Methode einen Kalman-Filter zugeordnet. Dabei ergeben sich die Kosten für die Kostenmatrix der Ungarischen Methode aus den euklidischen Distanzen zwischen den Schwerpunkten aus der Iteration i und denen aus der Iteration $i+1$. Bei der Zuordnung ist die folgende Fallunterscheidung erforderlich. Der erste Fall tritt ein, wenn die Kostenmatrix noch nicht initialisiert ist oder wenn es sich um einen neuen Schwerpunkt handelt. Der zweite Fall taucht auf, wenn eine initialisierte Kostenmatrix vorhanden ist aber nicht alle Schwerpunkte zugeordnet werden konnten. Der dritte Fall tritt ein, wenn die Kostenmatrix initialisiert ist und alle Schwerpunkte erfolgreich zugeordnet wurden. Beim ersten Fall wird die Kostenmatrix mit den neuen Schwerpunkten ergänzt bzw. initialisiert. Außerdem wird für jeden neuen Schwerpunkt ein Kalman-Filter erstellt, der mit dem neuen

Schwerpunkt assoziiert ist und mit seiner Position initialisiert wird. Beim zweiten Fall werden die nicht zugeordneten Schwerpunkte als *Miss*, d.h. nicht-getroffen, markiert. Darüber hinaus werden beim zweiten Fall die erfolgreich zugeordneten Schwerpunkte als *Hit*, d.h. getroffen, markiert. Beim dritten Fall werden alle Schwerpunkte als Hit gekennzeichnet. Anschließend erfolgt bei den letzten zwei Fällen eine Aktualisierung der Kalman-Filter mit der Position von den jeweiligen Hits und eine Vorhersage der Position der jeweiligen Misses. Danach findet durch die Kalman-Filter eine Korrektur der Position aller Hits und Misses statt.
Falls ein Schwerpunkt als Miss markiert wurde, wird ein Zähler in Form von einer einfachen Integer-Variable angelegt. Diesen verwendet man als Zähler für aufeinanderfolgende Bilder, bei denen der Schwerpunkt als Miss bezeichnet wird. Zeigt der Zähler, dass ein Schwerpunkt bei vielen aufeinanderfolgenden Bildern gefehlt hat, dann wird der Schwerpunkt von der Kostenmatrix entfernt. Darüber hinaus werden der Schwerpunkt und der dazugehörige Kalman-Filter gelöscht, um das Tracking vom fehlenden Schwerpunkt zu beenden. Die erlaubte Anzahl von aufeinanderfolgenden Bildern, bei denen ein Schwerpunkt fehlen darf, wird über einen Parameter eingestellt. Die passende Einstellung von diesem Parameter hängt von der Geschwindigkeit des Fahrzeugs ab und wird durch Testen bestimmt.
Ähnlich kommt ein Zähler im Fall eines neuen Schwerpunktes zum Einsatz. Dieser zählt hingegen die aufeinanderfolgenden Treffer von dem neuen Schwerpunkt. Ein neuer Schwerpunkt wird erst als gültig deklariert und somit für die weitere Modellierung von Spurmarkierungen verwendet, wenn er eine minimale Anzahl an aufeinanderfolgenden Treffern erreicht hat. Dieses Verhalten wird auch über einen von der Fahrzeuggeschwindigkeit abhängigen Parameter eingestellt. Die beschriebene Maßnahme hilft bei der Verminderung von falsch-positiven Erkennungen von Fahrbahnmarkierungen.

Zu guter Letzt werden die resultierenden Schwerpunkte mit korrigierten bzw. vorhergesagten Positionen zur anschließenden Modellierung von Spurmarkierungen an die nächste Phase dieser Umsetzung weitergegeben.

3.3.8 Modellierung der Spurmarkierungen - 2. Stufe

Hier wird nun eine verfeinerte Modellierung der Fahrbahnmarkierungen durchgeführt, um die Erkennung von gekrümmten Fahrstrecken zu verbessern. Diese Phase ergänzt daher die vorletzte Phase und erfolgt analog mit Hilfe vom RANSAC-Algorithmus.

Durch Testen stellte man fest, dass kubische Splines mit vier Stützstellen eine zufriedenstellende Genauigkeit bei der Modellierung von Fahrbahnmarkierungen anbieten. Aus diesem Grund werden in der aktuellen Phase im Gegensatz zu der Vorletzten kubische Splines statt Geraden zur Modellierung der Fahrbahnmarkierungen verwendet. Im Folgenden werden die dafür benötigten Schritte beschrieben.

Als erstes werden die aus der letzten Phase resultierenden Schwerpunkte dafür verwendet, die bounding boxes, d.h. die Suchbereiche, erneut zu ermitteln. Dabei werden die bounding boxes aus der vorletzten Phase (siehe Abbildung 3.8c) anhand ihrer Schwer-

punkte mit korrigierten bzw. vorhergesagten Positionen rekonstruiert.

Nach erneuter Bestimmung der bounding boxes werden diese zur Aufteilung des IPM-Bildes in kleinere Bildbereiche verwendet, so ähnlich wie in der vorletzten Phase. Im Anschluss daran wird der RANSAC-Algorithmus bei jedem Teilbereich angewendet. Dieser Schritt erfolgt auch analog zu der vorletzten Phase, jedoch mit den folgenden Unterschieden. Zum einen wird nun versucht, in jedem Teilbereich mittels Spline-Interpolation einen kubischen Spline zu fitten. Zum anderen werden zur Modellbildung bei jeder Iteration des RANSAC-Algorithmus genau vier nicht-schwarze Pixel aus dem Teilbereich zufällig gewählt. Dabei entsprechen die zufällig gewählten Pixel den Stützstellen vom zu bildenden Spline.
Am Ende des RANSAC-Algorithmus wird der bestbewertete Spline in Form von einer Menge *Point*-Objekten zurückgegeben. Bei dem bestbewerteten Spline handelt es sich um den Spline mit der größten Anzahl an inliers. Also der Spline, der die Pixelmenge im Teilbereich am besten fittet.

Als nächstes werden die Ergebnisse von der Anwendung von RANSAC in allen Teilbereichen vereinigt. Darüber hinaus wird zwecks weiterer Verarbeitung jeden resultierenden Spline mit dem Schwerpunkt von der zugehörigen bounding box assoziiert. Jeder dieser Punkte wird im weiteren Verlauf dieser Umsetzung als Referenzpunkt von einem Spline bezeichnet. Als letztes werden die resultierenden Splines zusammen mit den jeweiligen Referenzpunkten an die nachfolgende Phase weitergegeben. Das Endergebnis dieser Phase ist in Abbildung 3.9 zu sehen.

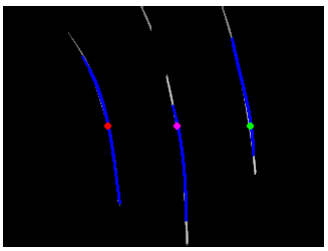


Abbildung 3.9: Ergebnis von RANSAC für kubische Splines und die dazugehörenden Referenzpunkte

3.3.9 Modellvalidierung

Bisher handelt es sich bei den erkannten Splines um reine Kandidaten für Fahrbahnmarkierungen. Außerdem besteht keine Information darüber, welches Paar von Kandidaten die zu befahrende Fahrspur bilden. Aus diesem Grund ist in dieser Phase zur Verminderung von falsch-positiven Ergebnissen die Gültigkeit der gefundenen Modellinstanzen zu validieren und damit Entscheidungen über die zu befahrende Fahrspur zu treffen. Dafür bietet es sich an, die Bestimmung der zu befahrenden Fahrspur als ein Optimierungsproblem zu betrachten. Dabei werden anhand der bekannten Abmessungen von der Fahrbahn und verschiedener Kriterien die Gültigkeit der Erkennungen bewertet. Im

Folgenden erfolgt eine detaillierte Beschreibung der genannten Methode.

Als erstes werden alle möglichen Spline-Paare gebildet. Dabei wird jedes Spline-Paar zusammen mit den zugehörigen Referenzpunkten in einer Variable gespeichert. Danach werden beide Splines jedes Spline-Paars anhand der x-Werte ihrer Referenzpunkte miteinander verglichen. Dadurch bestimmt sich ob der Spline $\boldsymbol{s_i}$ ein Kandidat für die linke oder rechte Fahrbahnmarkierung ist. Dabei wird der Spline s_i als der linke Spline markiert, falls der x-Wert des dazugehörigen Referenzpunktes kleiner als der x-Wert vom zweiten Referenzpunkt ist. Wenn das nicht der Fall ist, dann wird s_i als der rechte Spline markiert. Anschließend wird anhand der folgenden zu minimierenden Bewertungsfunktion die Bewertung für jedes Spline-Paar bestimmt.

$$Bewertung = \sqrt{k_1^2 + k_2^2 + k_3^2 + k_4^2 + k_5^2}$$

Wie es bei der Bewertungsfunktion zu erkennen ist, basiert der Bewertung auf fünf verschiedenen Bewertungskriterien, k_1 k_2, k_3, k_4, und k_5. Diese wurden im Rahmen dieser Arbeit festgelegt und werden im Folgenden vorgestellt.

Bewertungskriterien

- k_1: Die Definition von diesem Kriterium hängt von der eingestellten Fahrtrichtung ab. Im Fall von Rechtsverkehr entspricht k_1 dem horizontalen Abstand vom Referenzpunkt von dem rechten Spline zur rechten Bildkante. Im Fall von Linksverkehr ist k_1 gleich dem horizontalen Abstand vom Referenzpunkt von dem linken Spline zur linken Bildkante. Dieses Kriterium bestraft somit die Spline-Paare, die von der entsprechenden Bildkante zu weit entfernt sind. Daher entscheidet dieses Kriterium, ob die linke oder rechte Fahrspur als die zu befahrende Fahrspur zu deklarieren ist.

- k_2: Dieses Kriterium ergibt sich aus der Differenz zwischen der erwarteten Länge (in Pixel) von einer Fahrbahnmarkierung im IPM-Bild und der Länge von dem kürzesten Spline aus dem Spline-Paar. Dank dieses Kriteriums werden zu kurze Splines bestraft, damit mögliche Verarbeitungsfehler aus vorherigen Phasen die Wahl der zu befahrenden Fahrspur nicht beeinflussen.

- k_3: Dieses Kriterium wird anhand der folgenden Gleichung definiert.
  $k_3 = c \cdot (b_e - b_a)$. Dabei entspricht b_e der erwarteten Breite der Fahrspur (in Pixel), d.h. dem bekannten Abstand zwischen zwei Fahrspurmarkierungen. b_a ist gleich dem rechnerischen Abstand zwischen beiden Splines aus dem Spline-Paar. Der Faktor c entspricht einem Gewichtungsfaktor für dieses Kriterium. Anhand des Kriteriums k_3 werden dann große Abweichungen zwischen der Breite eines Spline-Paares und der gemäß bestimmten Spezifikationen erwarteten Breite der Fahrspur bestraft. Dadurch wird festgestellt, ob es sich bei dem Spline-Paar um eine Fahrspur handelt oder nicht.

- k_4: Das vierte Kriterium ergibt sich aus der Differenz zwischen dem x-Wert des Referenzpunktes von dem linken Spline und dem x-Wert des Referenzpunktes der

linken Fahrbahnmarkierung von der letzten Fahrspur, die in vorherigen Iterationen als gültig deklariert wurde.

- k_5: Das letzte Kriterium wird analog k_4 berechnet. Allerdings bezüglich dem x-Wert des Referenzpunktes von dem rechten Spline und dem x-Wert des Referenzpunktes der rechten Fahrbahnmarkierung von der letzten Fahrspur, die in vorherigen Iterationen als gültig deklariert wurde. Dank den Kriterien k_4 und k_5 werden große Positionsabweichungen zwischen der letzten gültigen Fahrspur und dem Spline-Paar bestraft, um die Robustheit gegen mögliche Verarbeitungsfehler aus den vorherigen Phasen zu erhöhen.

Die berechnete Bewertung wird als Maßstab für die Gültigkeit jedes Spline-Paars betrachtet. Im Anschluss daran wird das Spline-Paar χ, das die *niedrigste* Bewertung bekommen hat, mit einem Schwellenwert τ verglichen. Ist die Bewertung von χ kleiner als τ, dann wird das Spline-Paar χ als die zu befahrende Fahrspur deklariert. Siehe Abbildung 3.10. Im Rahmen dieser Arbeit wurde der genannte Schwellenwert durch Testen bestimmt.

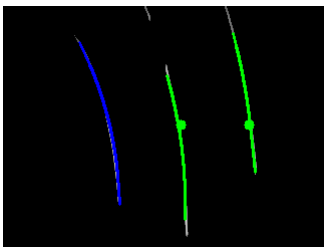


Abbildung 3.10: Nach der Modellvalidierung deklarierte zu befahrende Fahrspur (grüne Splines) im Fall von Rechtsverkehr

An dieser Stelle sind die folgenden Anmerkungen und Erklärungen für ein besseres Verständnis dieser Phase hilfreich.

- Die erwartete Breite der Fahrspur wird mit Hilfe von den aus der IPM-Technik resultierenden Skalierungsfaktoren von Millimeter in Pixel umgewandelt.
- Falls bis zur aktuellen Iteration des Algorithmus keine gültige Fahrspur deklariert wurde, dann ist zur Modellvalidierung kein Vergleich mit vorherigen Informationen möglich. Daher werden in diesem Fall sowohl k_4 als auch k_5 zu null gesetzt.
- Bei der Umstellung von Links- auf Rechtsverkehr oder umgekehrt tritt aufgrund des Kriteriums k_1 eine große Änderung bei den Bewertungen ein. Dies kann bei eventuellem Überholmanöver genutzt werden, um dynamisch zur Laufzeit in die linke bzw. in die rechte Fahrspur zu wechseln. Bei der Umsetzung dieser Arbeit kann die Fahrtrichtung sowohl über einen Parameter als auch über ein *ROS-Topic* umgestellt werden.

Als letztes wird in dieser Phase die deklarierte zu befahrende Fahrspur im Form von einem Spline-Paar an die nächste Phase weitergegeben. Falls keine gültige Fahrspur ermittelt wurde, wird die letzte gültige Fahrspur übergeben.

3.3.10 Nachbearbeitung und Fahrbahnführung

In Vorbereitung auf die Fahrbahnführung ist als erstes die Fahrspurmitte zu bestimmen. Dabei ist ein neues Polynom in die Mitte der zu befahrenden Fahrspur zu erzeugen. Dafür bietet es sich an, einen der beiden Splines horizontal in die Mitte der Fahrspur zu verschieben und die Menge verschobener Punkte als die Fahrspurmitte zu betrachten. Welcher der beiden Splines zu verschieben ist, hängt von der eingestellten Fahrtrichtung ab. Im Fall von Rechtsverkehr wird der rechte Spline verschoben. Bei Linksverkehr ist hingegen der linke Spline zu verschieben. Der Grund dafür ist, dass die mittige Fahrbahnmarkierung im Vergleich zu den anderen Markierungen in der Regel die Erzeugung von kürzeren Splines verursacht. Darüber hinaus ist die Fahrbahnerkennung bei der mittigen Fahrbahnmarkierung aufgrund der gestrichelten Linien fehleranfälliger. Deshalb ist diese Fahrbahnmarkierung bei der Bestimmung von der Fahrspurmitte zu vermeiden. Ein Beispiel für die resultierende Fahrspurmitte im Fall von Rechtsverkehr ist in Abbildung 3.11 zu finden. Im Anschluss daran werden die x- und y-Werte

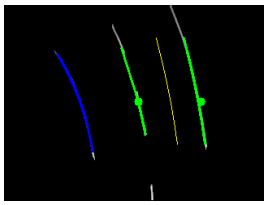


Abbildung 3.11: Fahrspurmitte (gelber Spline)

aller Punkte, die die berechnete Fahrspurmitte bilden, anhand der aus der IPM-Technik resultierenden Skalierungsfaktoren in Meter umgewandelt. Damit wird die tatsächliche Position der Fahrspurmitte bezüglich des Modellfahrzeugs approximiert.
Danach wird unter der Annahme, dass die Kamera ausgehend von der Fahrzeugbreite mittig auf dem Modellfahrzeug platziert ist, das Fahrzeug als Ursprung des Koordinatensystems betrachtet. Anschließend wird der Winkel $\boldsymbol{\alpha}$ zwischen der x-Achse und dem letzten Punkt der Fahrspurmitte berechnet, so wie es in Abbildung 3.12 dargestellt ist. Als nächstes wird α mit einem Proportionalitätsfaktor $\boldsymbol{p}$ multipliziert. Das daraus resultierende Ergebnis $\boldsymbol{\alpha'}$ wird als eine Approximation für den erforderlichen Lenkwinkel betrachtet. Anschließend erfolgt anhand einer für das verwendete Modellfahrzeug bestimmten Korrespondenztabelle die Umwandlung vom Wert α' in den entsprechenden Lenkanschlag für den Servomotor, der für die Lenkung verantwortlich ist. Als letztes wird der Wert vom gewünschten Lenkanschlag über ein *ROS-Topic* an einen weiteren *ROS-Node* weitergegeben, um das erforderliche Lenkmanöver einzuleiten. Der genannte Node wurde im Rahmen des Projektseminars Echtzeitsysteme an der TU-Darmstadt entwickelt.
Zur zufriedenstellenden Funktionalität der Fahrbahnführung ist der oben genannte Proportionalitätsfaktor p durch Testen passend zu bestimmen. Außerdem wird hierfür vorausgesetzt, dass das Modellfahrzeug zu Beginn mittig und korrekt in seiner Fahrspur ausgerichtet ist. Diese Voraussetzung entspricht dem Regelwerk von dem zuvor vorgestellten Wettbewerb für autonomes Fahren Carolo-Cup[Car14].

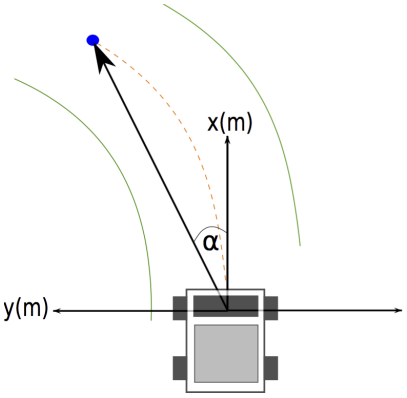


Abbildung 3.12: Darstellung der implementierten Fahrbahnführung. In Blau ist der Zielpunkt. In Grün ist die erkannte zu befahrende Fahrspur. In Gelb ist die berechnete Fahrspurmitte.

3.3.11 Optionale Rücktransformation der Modelle

Im Folgenden wird eine zusätzliche Phase des Algorithmus dieser Arbeit vorgestellt. Diese hat weder Einfluss auf die Fahrbahnerkennung noch auf die Fahrbahnführung. Allerdings dient diese Phase zur besseren Visualisierung der Erkennungen und evtl. zur Evaluierung der Ergebnisse.
Am Ende des Programmablaufs bietet es sich an, die im IPM-Bild erkannten Modellinstanzen, d.h. die Splines, auf dem Eingangsbild darzustellen. Dafür lassen sich die x- und y-Werte aller Punkte, die eine Modellinstanz bilden, in die zu dem Eingangsbild zugehörigen Werte zurücktransformieren (siehe Abbildung 3.13). Dies ist mit Hilfe von der Rücktransformation der IPM-Technik und von den daraus resultierenden Skalierungsfaktoren durchzuführen. Die Grundlagen dafür sind in Unterabschnitt 2.2 zu finden.
Da diese Rücktransformation für die normale Funktionalität des Algorithmus dieser Arbeit nicht erforderlich ist, ist diese Phase aufgrund des zusätzlichen Rechenaufwands standardmäßig deaktiviert. Diese kann jedoch über einen Parameter jederzeit aktiviert werden.

(a) Ergebnisse der Fahrbahnerkennung im IPM-Bild.

(b) die grünen Modellinstanzen aus dem IPM-Bild (linke Abbildung) nach der Rücktransformation.

Abbildung 3.13: Beispiel für die Rücktransformation von erkannten Modellinstanzen

4 Evaluierung und Diskussion

In diesem Kapitel wird als erstes die Funktionalität des in dieser Arbeit vorgestellten Algorithmus für Fahrbahnerkennung evaluiert. Dafür werden die Erkennungen von der zu befahrenden Fahrspur entlang einer bestimmten Anzahl an aufeinanderfolgenden Bildern mit den eigentlichen Positionen der Fahrbahnmarkierungen verglichen. Im Anschluss daran wird über die Auswirkungen ausgewählter Parameter auf die Ergebnisse der Fahrbahnerkennung eingegangen. Danach werden die durchschnittlichen Rechenzeiten je Phase des Algorithmus vorgestellt. Als letztes erfolgt eine Diskussion über das Fahrverhalten bei einer Fahrt auf der Testfahrbahn und über die dabei auftretenden Probleme.

4.1 Evaluierung der Fahrbahnerkennung

Zur Evaluierung der Fahrbahnerkennung wurde als erstes die Testfahrbahn in zwei verschiedenen Szenarien mit Hilfe von der implementierten Fahrbahnführung befahren. Dabei wurden die Rohbilder mit einer Bildwechselfrequenz von 10 Hz aufgezeichnet. Bei dem ersten Szenario handelt es sich um eine gerade Fahrstrecke mit anschließender Linkskurve unter optimalen Lichtbedingungen. Beim zweiten Szenario handelt es sich hingegen um eine Rechtskurve mit anschließender gerader Fahrstrecke unter unregelmäßigen Lichtbedingungen. Aus dem ersten Szenario ergab sich der Datensatz **A**, der aus 104 Bildern besteht. Aus dem zweiten Szenario resultierten 88 Bilder, die den Datensatz **B** bilden.
Danach wurden mit Hilfe vom Matlab-Tool *CaltechCV Image Labeler*[3] die zwei Spurmarkierungen der zu befahrenden Fahrspur in alle Bilder der beiden Datensätze per Hand eingezeichnet (siehe Abbildung 4.1), wobei Rechtsverkehr angenommen wurde. Im Anschluss daran wurden die Positionen der eingezeichneten Spurmarkierungen in Form einer Menge von Punkten gespeichert und in einer Datei $\boldsymbol{t}$ abgelegt. In dieser Datei ist jedes Bild mit den Positionen der zugehörigen Spurmarkierungen assoziiert.
Anschließend wurde der in dieser Arbeit vorgestellte Algorithmus für Fahrbahnerkennung zweimal ausgeführt, je einmal für alle Bilder aus Datensatz A und Datensatz B. Dabei wurde die in Unterabschnitt 3.3.11 beschriebene Rücktransformation durchgeführt, um eine Darstellung der erkannten Fahrspurmarkierungen auf dem Eingangsbild zu gewinnen. Im Anschluss daran wurden die erkannten Spurmarkierungen der zu befahrenden Fahrspur in Form einer Menge von Punkten gespeichert und in einer Datei $\boldsymbol{r}$ abgelegt.
Anschließend wurden die Informationen aus der Datei r mit den Daten aus der Datei t verglichen, um die Erkennungen zu validieren. Dabei wurden die Positionen der per Hand eingezeichneten Fahrspurmarkierungen als richtig angenommen. Der genannte Vergleich erfolgte mit Hilfe von frei zugänglichem Matlab-Code, der von Aly im Rahmen seiner Arbeit in [Aly08] geschrieben wurde und in [Aly11a] zu erhalten ist. Laut Aly wird das folgende Verfahren durchgeführt, um zu bestimmen, ob eine vom Algorithmus für Fahrbahnerkennung erkannte Spurmarkierungen $\boldsymbol{s_1}$ mit einer per Hand markierten Spurmarkierung $\boldsymbol{s_2}$ übereinstimmt. Als erstes wird für jeden Punkt p_i^1 der

[3] http://vision.caltech.edu/malaa/software/research/ccvLabeler/

Spurmarkierung s_1 der nächstliegendste Punkt p_j^2 auf der Spurmarkierung s_2 ermittelt. Danach wird die euklidische Distanz d_i^1 zwischen p_i^1 und p_j^2 berechnet. Dieses Verfahren wird für jeden Punkt p_i^2 der Spurmarkierung s_2 wiederholt. Daraus ergibt sich die euklidische Distanz d_i^2. Anschließend werden sowohl die Medianwerte beider Distanzen $\widehat{d_1}$ bzw. $\widehat{d_2}$ als auch die Mittelwerte $\overline{d_1}$ und $\overline{d_2}$ berechnet. Als letztes werden s_1 und s_2 als dieselbe Spurmarkierung deklariert, falls Gleichung 23 und Gleichung 24 erfüllt sind. Bei diesen Gleichungen stellen w_1 und w_2 die erlaubte Abweichung zwischen beiden Spurmarkierungen dar. Die Werte für diese zwei Parameter wurden ebenfalls der renommierten Arbeit von Aly entnommen und betrugen somit bei der Evaluierung der Fahrbahnerkennung jeweils 20 und 15 Pixel.

$$\min(\widehat{d_1}, \widehat{d_2}) \leq w_1 \tag{23}$$

$$\min(\overline{d_1}, \overline{d_2}) \leq w_2 \tag{24}$$

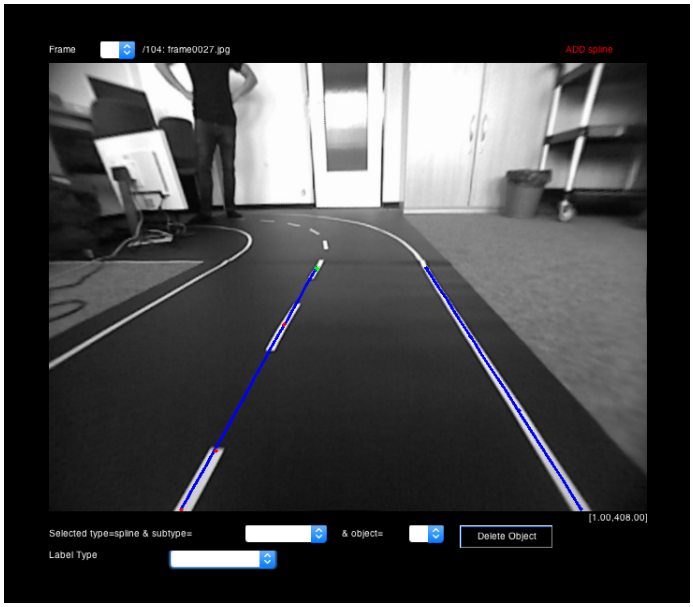


Abbildung 4.1: Manuelle Einzeichnung von Fahrbahnmarkierungen mit Hilfe vom Matlab-Tool *CaltechCV Image Labeler*.

4.1.1 Auswertung der Evaluationsergebnisse

Aus der zuvor beschriebenen Evaluierung ergaben sich die in Tabelle 4.1 präsentierten Ergebnisse. Dabei ist **RP** die Anzahl an richtig-positiven Erkennungen, d.h. die Anzahl an richtig erkannten Fahrspurmarkierungen. **FP** steht für die Anzahl an falsch-positiven Erkennungen, also die Anzahl an erkannten Fahrspurmarkierungen, die weder Gleichung 23 noch Gleichung 24 erfüllten. **FN** steht für die Anzahl an falsch-negativen

Erkennungen, d.h. die Anzahl an Fahrspurmarkierungen, die vom Algorithmus nicht erkannt werden konnten. Darüber hinaus ist **RPR** die Richtig-Positiv-Rate. Diese wurde anhand Gleichung 25 berechnet. Außerdem steht **FNR** für die Falsch-Negativ-Rate, die durch Gleichung 26 berechnet wurde.
An dieser Stelle wird noch erklärt, dass bei dieser Evaluierung die Anzahl an richtig-negativen Erkennungen nicht untersucht wurde, da bei unseren Experimenten die richtig-negativen Erkennungen nicht eindeutig definiert werden können. Die Grundlagen der verwendeten Klassifizierung sowie der statistischen Gütekriterien wurden der Arbeit von Powers entnommen. [Pow11]

$$RPR = \frac{RP}{RP + FN} \cdot 100 \tag{25}$$

$$FNR = \frac{FN}{RP + FN} \cdot 100 \tag{26}$$

Tabelle 4.1: Evaluationsergebnisse

Datensatz	Anz. Markierungen	RP	FP	FN	RPR	FNR
A	208	204	0	4	98,08%	1,92%
B	176	171	1	4	97,71%	2,29%

Den Evaluationsergebnissen ist eine zufriedenstellende Funktionalität des Algorithmus für Fahrbahnerkennung zu entnehmen. Wie man betrachten kann, ist die Richtig-Positiv-Rate bei beiden Datensätzen sehr hoch. Dies bedeutet, dass die implementierte Fahrbahnerkennung sowohl bei den geraden als auch bei den gekrümmten Fahrstrecken von unserer Testfahrbahn eine gute Leistung erbringt. Darüber hinaus lassen sich beim Testszenario B trotz unregelmäßiger Lichtbedingungen auch gute Ergebnisse betrachten. Dank der robusten Kantenfilterung ist beim Datensatz B nur eine falsch-positive Erkennung innerhalb von 176 Fahrbahnmarkierungen aufgetaucht. Zur Verdeutlichung der Lichtverhältnisse beim Szenario B dient Abbildung 4.2.
Andererseits wurden einige Spurmarkierungen bei beiden Testszenarien nicht erkannt (dies lässt sich der Falsch-Negativ-Rate entnehmen). Der Grund dafür ist die Funktionsweise vom verwendeten Tracking-Verfahren. Da zu Beginn der Fahrbahnerkennung das Tracking-Verfahren initialisiert werden muss, werden bei den ersten Bildern keine Spurmarkierungen erkannt.

An dieser Stelle wird erklärt, dass die Testszenarien mit den Bedingungen bei einer echten Straße nicht vergleichbar sind. Nämlich handelt es sich bei den Testszenarien um ideale Situationen, die nur mit den Bedingungen beim zuvor vorgestellten Wettbewerb für autonomes Fahren *Carolo-Cup* zu vergleichen sind. Darüber hinaus war aus Platzgründen nicht möglich, die Fahrbahnerkennung bei längeren und komplizierteren Fahrstrecken zu testen. Aus diesem Grund sind die hier präsentierten Ergebnisse nur unter Berücksichtigung der verwendeten Testfahrbahn zu betrachten.

Außerdem hängen die Ergebnisse zum großen Teil von der Einstellung verschiedener Parameter ab, da diese die Funktionalität des Algorithmus für Fahrbahnerkennung beeinflussen. Eine Liste von den wichtigsten Parametern, mit den standardmäßig eingestellten Werten ist in Tabelle A.1 im Anhang zu finden. Anhand der genannten Einstellung wurden die in diesem Unterabschnitt präsentierten Ergebnisse erhalten.

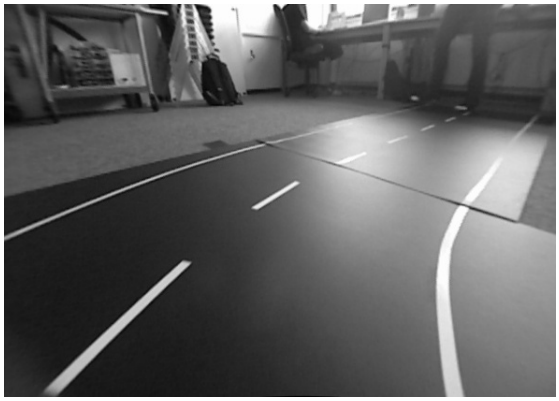


Abbildung 4.2: Bild aus Testszenario B: Rechtskurve mit unregelmäßigen Lichtverhältnissen

4.1.2 Auswirkungen von verschiedenen Parametern

In den folgenden Absätzen erfolgt nun eine Diskussion über ausgewählte Parameter, die eine große Auswirkung auf die Ergebnisse der Fahrbahnerkennung haben. Wenn möglich wird auch eine empfohlene Einstellung dieser Parameter ermittelt. Dafür werden die in Tabelle 4.1 präsentierten Ergebnisse beim Testszenario **B** als Referenzwerte betrachtet. Danach werden die ausgewählten Parameter mehrmals umgestellt. Nach jeder Umstellung eines Parameters wird die Evaluierung der Fahrbahnerkennung mit den Bildern vom Datensatz B erneut durchgeführt. Die neuen Ergebnisse werden anschließend mit den Referenzwerten verglichen, um die Auswirkungen je Parameter zu ermitteln.
Zur Erleichterung des Vergleichs werden in den folgenden Tabellen sowohl die Referenzwerte als auch die standardmäßige Einstellung jedes Parameters mit **Blau** gekennzeichnet.

Als erstes wird über den Einfluss vom Schwellenwert τ diskutiert. Dieser Parameter kommt in der Phase „Modellvalidierung“ zum Einsatz und wurde in Unterabschnitt 3.3.9 beschrieben. Der Schwellenwert τ ist entscheidend für die Validierung der erkannten Fahrbahnmarkierungen und damit für die Bestimmung der zu befahrenden Fahrspur.
Den Ergebnissen in Tabelle 4.2 ist zu entnehmen, dass die Falsch-Negativ-Rate bei kleinen τ-Werten zunimmt. Der Grund dafür ist, dass bei kleinen τ-Werten sehr wenige Fahrbahnmarkierungen als gültige Spurmarkierungen deklariert werden. Dies führt zu einer großen Anzahl an Bildern, in denen keine Fahrspurmarkierungen bestimmt werden konnten. Im Gegensatz dazu ist auch aus der Tabelle zu ersehen, dass bei zu großen

τ-Werten einige falsch-positiven Erkennungen auftauchen können.
Bei τ-Werten zwischen 180 und 250 lassen sich die besten Ergebnisse erkennen.

Tabelle 4.2: Ergebnisse der Fahrbahnerkennung für verschiedene Werte vom Schwellenwert τ

τ	RP	FP	FN	RPR	FNR
150	34	0	142	19,32%	80,68%
160	68	0	108	38,64%	61,36%
170	131	1	44	74,86%	25,14%
180	171	1	4	97,71%	2,29%
250	171	1	4	97,71%	2,29%
500	170	2	4	97,70%	2,30%

Als nächstes wird die Auswirkung vom Parameter w untersucht. Dieser Parameter wird in der Phase „Extraktion der Merkmale“ verwendet und wurde in Unterabschnitt 3.3.5 beschrieben. Anhand vom Parameter w wird die Dimension der Faltungsmatrix k_u zur Kantenfilterung in horizontaler Richtung festgesetzt. Mit anderen Worten wird mit Hilfe vom Parameter w die Verschärfung von Kanten im IPM-Bild eingestellt.
Wie die Tabelle 4.3 zeigt, ist die Falsch-Negativ-Rate bei w-Werten kleiner als 5 sehr hoch. Das liegt daran, dass bei kleinen w-Werten die Faltungsmatrix k_u zu klein wird. Das führt dazu, dass bei der Faltung der Kantenfilterung eine ungenügende Anzahl an benachbarten Pixeln betrachtet wird. Aus diesem Grund werden in diesem Fall die Kanten schlecht hervorgehoben und somit wird die Erkennung von Fahrbahnmarkierungen erschwert. Im Gegensatz dazu treten Verarbeitungsfehler bei zu großen Dimensionen der Faltungsmatrix k_u ein. Dies erklärt die schlechten Ergebnisse bei großen w-Werten. Aus diesem Grund wird im Rahmen dieser Arbeit die Einstellung vom Parameter w zwischen 5 und 9 empfohlen.

Tabelle 4.3: Ergebnisse der Fahrbahnerkennung für verschiedene Werte vom Parameter w

kernel_width	**RP**	**FP**	**FN**	**RPR**	**FNR**
3	89	9	78	53,29%	46,71%
4	93	5	78	54,39%	45,61%
5	162	6	8	95,29%	4,71%
7	171	1	4	97,71%	2,29%
14	105	35	36	74,47%	25,53%
30	88	62	26	77,19%	22,81%

Als letztes wird die Auswirkung vom Schwellenwert φ betrachtet. Dieser Parameter wird in der Phase „Extraktion der Merkmale“ für die Einstellung der Thresholding-Operation verwendet. Der genannte Schwellenwert wurde in Unterabschnitt 3.3.5 beschrieben.

Anhand des Schwellenwerts φ wird nach der Kantenfilterung bestimmt, welche Pixel zu null gesetzt werden müssen. φ wird in Form von einem Quantil dargestellt und kann deswegen nur reelle Zahlen zwischen 0 und 1 annehmen.
Die Tabelle 4.4 zeigt, dass bei φ-Werten kleiner als 0,95 viele falsch-positiven Erkennungen auftauchen. Der Grund dafür ist, dass bei kleinen φ-Werten viele nicht-schwarzen Pixel im IPM-Bild bleiben. Dies führt aufgrund der hohen Anzahl an Ausreißern sowohl zu Verarbeitungsfehlern als auch zu einer Erhöhung des Rechenaufwands. Wenn man hingegen den Parameter φ mit Werten größer als 0,99 einstellt, dann wird durch die Thresholding-Operation zu viel Information gelöscht. Dabei wird die Erkennung von vielen Fahrbahnmarkierungen nicht mehr möglich. Aus diesem Grund wird im Rahmen dieser Arbeit die Einstellung von φ mit Werten zwischen 0,97 und 0,99 empfohlen.

Tabelle 4.4: Ergebnisse der Fahrbahnerkennung für verschiedene Werte vom Schwellenwert φ

φ	RP	FP	FN	RPR	FNR
0,8500	69	103	4	94,52%	5,48%
0,9000	76	94	6	92,68%	7,32%
0,9500	154	18	4	97,47%	2,53%
0,9875	171	1	4	97,71%	2,29%
0,9900	170	2	4	97,70%	2,30%

4.2 Durchschnittliche Rechenzeiten

In den folgenden Absätzen werden die durchschnittlichen Rechenzeiten je Phase des Algorithmus vorgestellt und diskutiert.
Zur Messung der Rechenzeiten wurde der gesamte Algorithmus dieser Arbeit auf dem Mainboard des Modellfahrzeugs ausgeführt, während die Testfahrbahn befahren wurde. Dabei wurde mit Hilfe von der Kinect-Kamera alle 33,33ms ein neues Bild von der Testfahrbahn erhalten. Falls der Haupt-Thread des Algorithmus zum Zeitpunkt des Empfangs eines neuen Bildes frei war, wurde die Fahrbahnerkennung auf das neue Bild angewendet. Dabei wurde die Rechenzeit je Phase gemessen. Anschließend wurde der Mittelwert der benötigten Rechenzeiten bei 500 aufeinanderfolgenden Iterationen berechnet. Die daraus resultierenden Ergebnisse sind in Tabelle 4.5 zu finden.

Aus Platzgründen wurden die folgenden Abkürzungen verwendet.

- Vva: Vorverarbeitung
- EdM: Extraktion der Merkmale
- MdS_1: Modellierung der Spurmarkierungen - 1. Stufe
- MdS_2: Modellierung der Spurmarkierungen - 2. Stufe
- Mv: Modellvalidierung
- Pt: Positionstracking
- Nba: Nachbearbeitung

Tabelle 4.5: Durchschnittliche Rechenzeiten je Phase

Vva	EdM	MdS_1	MdS_2	Pt	Mv	Nba	Gesamt
4,58ms	3,24ms	7,10ms	24,88ms	105,53μs	58,57μs	1,62ms	41,58ms

Die Tabelle zeigt, dass der Algorithmus im Durchschnitt 41,58ms pro Durchlauf benötigt. Somit ist der Algorithmus etwa 8ms langsamer als die verwendete Kamera. Dies bedeutet, dass alle empfangenen Bilder nicht bearbeitet werden können. Aus diesem Grund sind einige Bilder zu verwerfen. Dies stellt jedoch kein Problem dar, weil aufgrund der hohen Bildwiederholungsfrequenz die Änderungen zwischen zwei aufeinanderfolgenden Bildern nicht signifikant sind.
Die Höchstgeschwindigkeit des Modellfahrzeugs beträgt etwa 1m/s. Dies bedeutet, dass das Fahrzeug bei dieser Geschwindigkeit eine Strecke von ca. 4,2cm pro Durchlauf des Algorithmus befahren würde. Diese theoretische Strecke wäre für eine zufriedenstellende Fahrbahnführung bzw. Fahrzeugregelung noch akzeptabel. Daraus kann man folgern, dass die Echtzeitanforderungen durch den implementierten Algorithmus erfüllt sind.
Der Tabelle ist auch zu entnehmen, dass die Phase MdS_2 am aufwändigsten ist. Der Grund dafür ist der Rechenaufwand bei der Modellierung von Fahrbahnmarkierungen mit Hilfe von RANSAC und kubischen Splines. Um den Rechenaufwand zu reduzieren und damit auch die gesamte Rechenzeit, können folgende Verbesserungsvorschläge eingebracht werden. Zum einen kann die Größe des IPM-Bildes verkleinert werden, damit der RANSAC-Algorithmus weniger Iterationen benötigt. Allerdings kann dadurch ungenauere Modelle der Fahrbahnmarkierungen entstehen. Zum anderen lässt sich die Durchführung vom RANSAC-Algorithmus mit Hilfe von einer Grafikkarte beschleunigen. Dafür wäre eine Parallelisierung von RANSAC anhand *OpenCL*[4] oder *CUDA*[5] erforderlich. Da auf dem Modellfahrzeug nur eine sehr einfache On-Board-Grafikkarte zur Verfügung steht, kam im Rahmen dieser Arbeit eine solche Parallelisierung nicht in Frage. Jedoch ist es möglich die Fahrbahnerkennung auf einem zweiten Rechner mit besserer Ausstattung auszulagern und die Daten über das ROS-Netzwerk und eine WLAN-Verbindung zu übertragen.

Andererseits wurde neben den Rechenzeiten die gesamte CPU-Auslastung gemessen. Diese betrug im Durchschnitt 54%. Man kann sich fragen, warum keine 100% CPU-Auslastung erreicht wird, um die benötigten Rechenzeiten zu minimieren. Diese Frage lässt sich wie folgt beantworten. Die Durchführung des Algorithmus wird durch verschiedene Speicherzugriffe und nebenläufige Prozesse beschränkt. Aus diesem Grund ist die völlige Nutzung der vier Kerne vom vorhandenen Prozessor für die Durchführung des Algorithmus nicht möglich.

[4] https://www.khronos.org/opencl/
[5] http://www.nvidia.de/object/cuda-parallel-computing-de.html

4.3 Testfahrt

Um die Ergebnisse dieser Arbeit auch qualitativ auszuwerten, wurde ein Funktionstest der Fahrbahnerkennung und Fahrspurführung durchgeführt. Dabei wurde die Testfahrbahn zweimal befahren, je einmal auf der linken Fahrspur und auf der rechten Fahrspur. Während die Fahrten wurde die Visualisierung der erkannten Fahrbahnmarkierungen aufgezeichnet. Darüber hinaus wurde mit Hilfe einer externen Kamera ein Video von den Fahrten aufgenommen. Danach erfolgte eine qualitative Auswertung der gewonnenen Daten.
In Abbildung 4.3 ist ein Screenshot der gewonnenen Daten bei der Fahrt auf der rechten Fahrspur zu sehen. Das größte Bild stammt aus der externen Kamera. Oben rechts befinden sich die erkannten Fahrbahnmarkierungen(grün) und die berechnete Fahrspurmitte(gelb). Unten rechts sieht man das Rohbild der Kinect-Kamera.

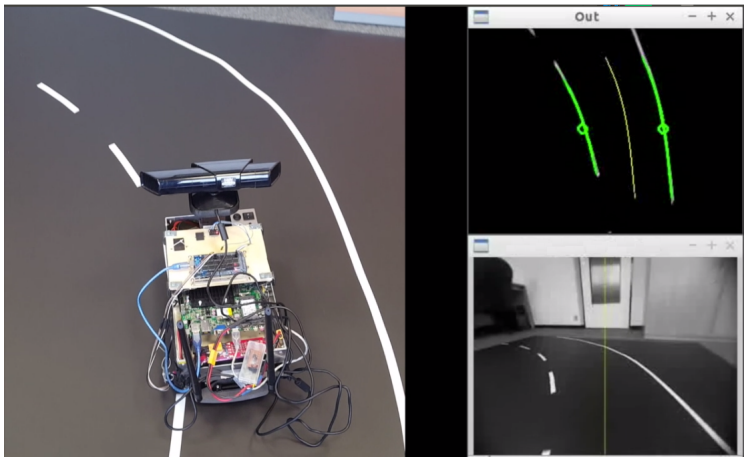


Abbildung 4.3: Screenshot von der Testfahrt

Der Screenshot zeigt, dass beide Spurmarkierungen der zu befahrenden Fahrspur sowie die Fahrspurmitte mit zufriedenstellender Genauigkeit ermittelt wurden. Allerdings sieht man auch, dass das Modellfahrzeug die mittige Fahrbahnmarkierung überfuhr. Dieses Verhalten ist nicht auf die implementierte Fahrbahnerkennung zurückzuführen, sondern auf den verwendeten Ansatz zur Fahrbahnführung bzw. Fahrzeugregelung. Dies liegt daran, dass die eingesetzte Fahrbahnführung bei gekrümmten Fahrstrecken nur begrenzt einsetzbar ist. Nämlich wird im Fall eines zu weit entfernten Zielpunktes ein zu großer Lenkanschlag berechnet. Dies kann dazu führen, dass das Modellfahrzeug bei Kurven die mittige Fahrbahnmarkierung überfährt. Zur Verdeutlichung dieser Problematik dient Abbildung 4.4. Eine mögliche Lösung dieses Problems wäre, anhand der berechneten Fahrspurmitte einen Pfad zu bestimmen, den das Auto später auf der Fahrstrecke abfahren soll. Dabei wäre nicht nur der letzte Punkt der Fahrspurmitte als Zielpunkt zu betrachten, sondern auch verschiedene Punkte entlang der Fahrspurmitte.
Eine weitere Lösung wäre, den Kurvenradius an einer beliebigen Stelle x anhand der berechneten Fahrspurmitte zu bestimmen. Anschließend wäre es möglich mit Hilfe von der Vorwärtskinematik der Ackermann-Lenkung und von den Sensoren im Modellfahr-

zeug, den erforderlichen Lenkwinkel an der Stelle x mit mehr Genauigkeit zu ermitteln. Die Grundlagen eines solchen Ansatzes lassen sich der Masterthesis von Jenning[Jen08, S. 34–36] entnehmen. Darüber hinaus sind die Grundlagen der Vorwärtskinematik der Ackermann-Lenkung in der Arbeit von Ehmes[Ehm16, S. 7–8] zu finden.

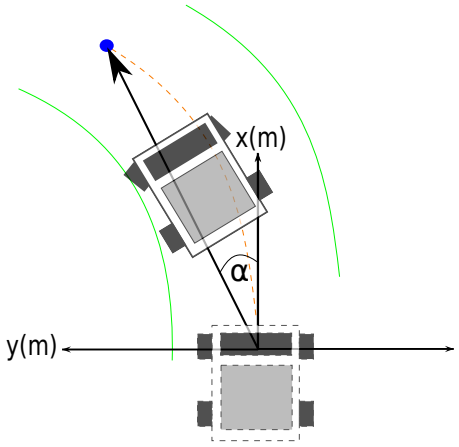


Abbildung 4.4: Darstellung des Problems mit der implementierten Fahrbahnführung. In Blau ist der Zielpunkt. In Grün ist die erkannte zu befahrende Fahrspur. In Gelb ist die berechnete Fahrspurmitte.

5 Verwandte Arbeiten

Die Fahrbahnerkennung sowie die automatische Fahrbahnführung sind hochaktuelle Forschungsthemen. Aus diesem Grund sind in der Literatur zahlreiche Techniken für diese wesentlichen Aufgaben des autonomen Fahrens beschrieben. Verschiedene Arbeiten beschäftigen sich mit der Zusammenfassung von bereits vorgestellten Ansätzen. Ein Beispiel dafür ist die Arbeit von A. M. Kumar und P. Simon, die in [KS15] die Entwicklung der Fahrbahnerkennung von 2003 bis 2014 zusammenfassen. Als weiteres Beispiel dient die Arbeit von Yenikaya et al. Sie begutachten und fassen die verschiedenen Ansätze von 1991 bis 2009 in [YYD13] zusammen.
Beide Arbeiten zeigen, dass etwa die Hälfte der zusammengefassten Ansätze analog zu der Umsetzung dieser Thesis die Hough-Transformation bei der Phase „Extraktion der Merkmale“ benutzen. Die anderen Autoren verwenden hingegen unterschiedliche Methoden wie *Haar-like Features* oder Methoden aus der künstlichen Intelligenz. Zusätzlich kann man auch in beiden Arbeiten sehen, dass bei den anderen allgemeinen Phasen der Fahrbahnerkennung die eingesetzten Methoden je nach Ansatz stark variieren. Dabei hängen die Ergebnisse und die verwendeten Methoden vom Anwendungsszenario ab.
In den folgenden Absätzen werden nun ausgewählte Ansätze beleuchtet. Diese sollen einen Überblick über Alternativen zur Implementierung dieser Arbeit geben. Darüber hinaus wird erklärt, wo Unterschiede bestehen und Gemeinsamkeiten auftreten und warum diese Ansätze keine Anwendung in dieser Bachelorarbeit fanden.

Die Arbeit von Jung et al. [JMK13] beschreibt einen der schnellsten Algorithmen für Fahrbahnerkennung. Laut der Autoren beträgt die gesamte Rechenzeit des Algorithmus nur 0,12ms pro Durchlauf. Dafür verwenden Jung et al. nur einfache Methoden, die sehr effizient durchführbar sind. Als erstes wird das Rohbild in ihre linke und rechte Hälfte zerschnitten. Danach wird ein sogenannter *steerable filter* auf jede Hälfte angewendet. Dadurch werden Linien mit einer bestimmten Steigung hervorgehoben. Als nächstes werden Punkte aus dem gefilterten Bild extrahiert. Anhand dieser Punkte werden zwei Geraden gebildet, die je einen Kandidaten für die linke und rechte Fahrspurmarkierung darstellen. Anschließend werden diese Kandidaten mit bestimmten Kriterien validiert. Als letztes wird mit Hilfe von den erkannten Fahrspurmarkierungen ein Verfahren zur Spurverlassenswarnung durchgeführt.
Der Algorithmus von Jung et al. liefert mit einer Richtig-Positiv-Rate von 90,16% befriedigende Ergebnisse. Allerdings ist dieser Algorithmus sehr simpel und nicht robust gegen Ausreißer. Darüber hinaus basiert dieser nur auf der Modellierung mit Geraden und eignet sich nur für Spurverlassenswarnung und nicht für Fahrbahnführung. Aus diesem Grund kam die Verwendung dieses Ansatzes im Rahmen dieser Arbeit nicht in Frage.

Eine weitere Alternative zur Fahrbahnerkennung bietet die Arbeit von Lipski et al. [LSB+08]. Diese Arbeit beschreibt einen robusten Ansatz, der 2007 beim Wettbewerb für autonomes Fahren „DARPA Urban Challange“ auf einem echten Fahrzeug erfolgreich getestet wurde. Gemeinsamkeiten zwischen diesem Ansatz und dem Algorithmus

dieser Bachelorarbeit sind die Verwendung von der IPM-Technik und einem Tracking-Verfahren. Allerdings gibt es auch Unterschiede, die im Folgenden erläutert werden. Zu Beginn wird das Rohbild nicht in ein Graustufenbild umgewandelt, sondern dieses wird mit der gesamten Farbinformation in ein IPM-Bild transformiert. Danach werden die Farbwerte vom RGB-Farbraum in den HSV-Farbraum umgewandelt, damit neben weißen Fahrbahnmarkierungen auch farbige Markierungen durch Detektion von starken Änderungen der Farbsättigung erkannt werden können. Anschließend wird zur Extraktion der Merkmale im Gegensatz zum Algorithmus dieser Thesis keine Kantenfilterung durchgeführt, sondern mehrere Threshold-Operationen. Diese werden auf kleine Bildbereiche angewendet, wobei die Farbsättigung und Farbhelligkeit jedes Pixels mit bestimmten Schwellenwerten verglichen werden. Interessant ist, dass beim Lipski et al.'s Ansatz alle vorherigen Schritten zur Beschleunigung des Algorithmus auf einer Grafikkarte durchgeführt werden. Als letztes wird analog zu dieser Thesis der RANSAC-Algorithmus verwendet, um Modelle der Fahrbahnmarkierungen auf dem IPM-Bild zu fitten. Allerdings werden diese nicht durch Splines modelliert, sondern mit mehreren Fahrspurschablonen, d.h. mit mehreren Geraden.
Laut Lipski et al. erreicht der Algorithmus eine Wiederholrate von 10 Bilder pro Sekunde. Somit ist Lipski et al.'s Ansatz etwa doppelt so langsam wie die Variante dieser Thesis. Darüber hinaus ist für diesen Algorithmus eine hochmoderne Grafikkarte erforderlich. Aufgrund der Tatsache, dass auf unserem Modellfahrzeug nur eine sehr einfache Grafikkarte zur Verfügung steht und dass der Rechenaufwand hoch ist, wurde Lipski et al.'s Ansatz im Rahmen dieser Thesis nicht verwendet.

Interessant ist die Masterarbeit von Jenning [Jen08]. Diese wurde an der Hochschule für Angewandte Wissenschaften Hamburg zwecks der Teilnahme am Carolo-Cup verfasst und beschreibt sowohl einen Ansatz zur Fahrbahnerkennung als auch die Grundlage eines Ansatzes zur Fahrspurführung. Jenning verwendet analog zu dieser Thesis eine Kantenfilterung und die IPM-Technik für die Fahrbahnerkennung. Allerdings wird bei Jenning's Ansatz statt Gauß-Filter der Sobel-Operator zur Kantenfilterung eingesetzt. Darüber hinaus wird die Kantenfilterung vor der Anwendung der IPM-Technik durchgeführt. Des Weiteren verwendet Jenning der RANSAC-Algorithmus nicht. Stattdessen erfolgt die Modellierung von jeder Fahrspurmarkierung durch mathematische Approximation der Parameter eines einfachen Polynoms 3. Grades. Andererseits bestimmt Jenning anhand der approximierten Polynome den lateralen Abstand des Fahrzeugs zum Mittelpunkt der Fahrspur, den tangentialen Steuerkurswinkel des Fahrzeugs zur Fahrspur, sowie den Radius an einer beliebigen Stelle des Polynoms, um die Fahrbahnführung bzw. Fahrzeugregelung zu realisieren.
Jenning berichtet in seiner Arbeit vor allem über die mathematischen Grundlagen hinter seiner Umsetzung und erklärt nicht viel über die genaue Implementierung. Zudem veröffentlichte der genannte Autor keinen Quellcode von seiner Arbeit. Aus diesem Grund fällt es schwer, seine Ergebnisse zu reproduzieren oder auf diesen aufzubauen. Es bietet sich jedoch als Ausblick auf diese Bachelorarbeit an, die von Jenning vorgestellten Grundlagen nachzuvollziehen, um die Rechenzeit der Fahrbahnerkennung zu reduzieren und die Fahrbahnführung bzw. Fahrzeugregelung zu verbessern.

6 Fazit

Ziel dieser Arbeit war es, basierend auf bereits vorgestellten Ansätzen, ein Verfahren zur kamerabasierten Fahrbahnerkennung und automatischen Fahrbahnführung zu entwickeln und zu evaluieren. Dieses sollte auf Modellfahrzeugen in *Indoor*-Szenarien spezialisiert sein.
Um das Ziel zu erreichen, mussten zuerst andere Ansätze untersucht werden. Aus der Literaturrecherche ergab sich, dass zahlreiche Techniken für Fahrbahnerkennung bereits entwickelt wurden. Dabei stellte sich heraus, dass es weder eine perfekte Lösung noch einen überall empfohlenen Ansatz gibt. Die verwendeten Methoden und die daraus resultierenden Ergebnisse hängen vom Anwendungsszenario sowie von Einflussfaktoren ab. Ausgehend von diesen Thesen wurde der Ansatz dieser Arbeit folgendermaßen gezielt gewählt. Erstens wurde die IPM-Technik verwendet, um eine projektive Transformation zwischen Bild- und Fahrzeugkoordinatensystem durchzuführen. Zweitens wurde eine Kantenfilterung eingesetzt, die die Fahrbahnmarkierungen im Bild anhand den bekannten Spezifikationen bzw. Abmessungen der Fahrbahn verschärft. Drittens wurden Geraden und kubische Splines im Zusammenspiel mit dem RANSAC-Algorithmus benutzt, um die Fahrbahnmarkierungen mit zufriedenstellender Genauigkeit zu modellieren. Viertens wurde eine Modellvalidierung implementiert, um die erkannten Fahrbahnmarkierungen zu validieren und damit die zu befahrende Fahrspur zu ermitteln. Fünftes wurde die Fahrspurmitte dafür verwendet, das erforderliche Lenkmanöver einzuleiten. Zu guter Letzt wurde der gesamte Ansatz mit einem Tracking-Verfahren unterstützt, um die Robustheit gegen Ausreißer und mögliche Verarbeitungsfehler zu erhöhen.

Das gesamte Verfahren zur kamerabasierten Fahrbahnerkennung und automatischen Fahrbahnführung wurde auf einer Testfahrbahn getestet. Dabei handelte es sich um eine Fahrstrecke, die sich am Regelwerk vom Wettbewerb für autonomes Fahren *Carolo-Cup* orientiert. Um die Funktionalität der Fahrbahnerkennung quantitativ auszuwerten, wurden die Positionsabweichungen zwischen den vom Verfahren dieser Arbeit und den von Menschen erkannten Fahrbahnmarkierungen berechnet und ausgewertet. Daraus ergab sich, dass die in dieser Arbeit vorgestellte Ansatzvariante zur Fahrbahnerkennung mit einer Richtig-Postiv-Rate zwischen 97,71% und 98,08% sehr gute Ergebnisse liefert. Darüber hinaus zeigten die Ergebnisse, dass die implementierte Fahrbahnerkennung sowohl robust gegen Ausreißer als auch gegen unregelmäßige Lichtbedingungen ist. Außerdem wird mit dieser Ansatzvariante nicht nur eine zufriedenstellende Genauigkeit der Modellierung der Fahrbahnmarkierungen bei Geraden erreicht, sondern auch bei gekrümmten Fahrstrecken. Allerdings beziehen sich diese Ergebnisse auf einer einfachen Testfahrbahn, die lediglich aus einer Gerade und einer Kurve besteht. Aus diesem Grund ist als Ausblick auf diese Bachelorarbeit wünschenswert, die Fahrbahnerkennung auf komplexeren Fahrstrecken zu testen. Beispielsweise wäre es noch interessant, die Antwort des Algorithmus auf eine S-Kurve oder auf unterbrochene Fahrbahnmarkierungen zu untersuchen.

Bei den Testfahrten auf der Testfahrbahn stellte sich heraus, dass das Verfahren dieser Arbeit an den folgenden Stellen Verbesserungs- bzw. Optimierungsbedarf aufzeigt. Zum einen erfüllt zwar das Verfahren mit einer gesamten Rechenzeit von 41,58ms pro Bild die Echtzeitanforderung, aber die Rechenzeit kann bei der Modellierung der Fahrbahnmarkierungen noch reduziert werden. Zum anderen soll die implementierte Fahrbahnführung, also die Fahrzeugregelung, verbessert werden, da diese simpel ist und Probleme bei gekrümmten Fahrstrecken aufzeigt. Aus diesem Grund wurde am Ende des Kapitels 4 auf verschiedene Verbesserungsvorschläge eingegangen.

Ausgehend von den präsentierten Ergebnissen ist das in dieser Arbeit vorgestellte Verfahren zur Fahrbahnerkennung und automatischen Fahrbahnführung vielversprechend. Diese Bachelorarbeit und das vorgestellte Verfahren lassen sich somit als Informationsgrundlage für die Teilnahme an Wettweberben für autonomes Fahren, wie Carolo-Cup, oder für interne Wettbewerbe und Projekte an der TU-Darmstadt weiterverwenden. Darüber hinaus wurde das implementierte Verfahren mit ROS integriert und wird als ein ROS-Node bereitgestellt, damit weitere Studenten diese benutzen und erweitern können. Wünschenswerte Erweiterungen und mögliche nachfolgende Projekte, die sich an diese Arbeit anknüpfen lassen, sind beispielsweise Verfahren zur Kreuzungs-, Hindernis- und Parkmarkierungserkennung.

Literatur

[Aly08] Aly, Mohamed: Real time Detection of Lane Markers in Urban Streets. In: *IEEE Intelligent Vehicles Symposium*. Eindhoven, The Netherlands, 2008

[Aly11a] Aly, Mohamed: *Caltech Lane Detection Software*. http://vision.caltech.edu/malaa/software/research/caltech-lane-detection/. Version: 2011. – Stand: 15.06.2016

[Aly11b] Aly, Mohamed: *CaltechCV Image Labeler*. http://vision.caltech.edu/malaa/software/research/ccvLabeler/. Version: 2011. – Stand: 15.06.2016

[BK08] Bradski, Gary ; Kaehler, Adrian: *Learning OpenCV: Computer Vision with the OpenCV Library*. 1. O'Reilly & Associates, 2008

[Car14] *Carolo-Cup Regelwerk 2015*. Technische Universität Braunschweig. https://wiki.ifr.ing.tu-bs.de/carolocup/system/files/Hauptwettbewerb2015.pdf. Version: Juni 2014. – Stand: 10.08.2016

[Ehm16] Ehmes, Sebastian: *Erkunden von unbekanntem Terrain mit einem autonomen Modellfahrzeug*, Technische Universität Darmstadt, Diplomarbeit, 2016

[FB81] Fischler, Martin A. ; Bolles, Robert C.: Random Sample Consensus: A Paradigm for Model Fitting with Apphcatlons to Image Analysis and Automated Cartography. In: *Communications of the ACM* 24 (1981), Juni, Nr. 6

[Han15] Handau, Daniel: *Kamerabasierte Fahrbahnerkennung mit einem autonomen Modellfahrzeug*. Darmstadt, Deutschland, Technische Universität Darmstadt, Diplomarbeit, April 2015

[HB09] Hanke-Bourgeois, Prof. Dr. M.: *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. 3. Vieweg+Teubner, 2009

[Jen08] Jenning, Eike: *Systemidentifikation eines autonomen Fahrzeugs mit einer robusten, kamerabasierten Fahrspurerkennung in Echtzeit*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2008

[JMK13] Jung, Heechul ; Min, Junggon ; Kim, Junmo: An Efficient Lane Detection Algorithm For Lane Departure Detection. In: *IEEE Intelligent Vehicles Symposium (IV)*. Australia, Juni 2013

[Kal60] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. In: *ASME-Journal of Basic Engineering* (1960)

[Kr"15] Krüger, Till-Julius: *Fahrspurmodellierung mit Punktvalidierung bei autonomen Modellfahrzeugen*. Berlin, Deutschland, Freie Universität Berlin, Diplomarbeit, September 2015

[KS15] Kumar, Ammu M. ; Simon, Philomina: Review of Lane Detection and Tracking Algorithms in Advanced Driver Assistance System. In: *International Journal of Computer Science & Information Technology (IJCSIT)* 7 (2015), August, Nr. 4

[Kuh55] Kuhn, H. W.: The Hungarian method for the assignment problem. In: *Naval Research Logistics Quarterly 2* (1955), S. 83–97

[Lä15] Lässig, Conrad: *Fahrspurerkennung mit RANSAC bei autonomen Modellfahrzeugen*. Berlin, Deutschland, Freie Universität Berlin, Diplomarbeit, Mai 2015

[LFW12] Li, Hua ; Feng, Mingyue ; Wang, Xiao: Inverse Perspective Mapping Based Urban Road Markings Detection. In: *Proceedings of IEEE CCIS2012*, 2012

[LSB+08] Lipski, C. ; Scholz, B. ; Berger, K. ; Linz, C. ; Stich, T. ; Magnor, M.: A Fast and Robust Approach to Lane Marking Detection and Lane Tracking. In: *Image Analysis and Interpretation, 2008. SSIAI 2008. IEEE Southwest Symposium on* (2008)

[MRP16] MRPT: *Discover MRPT*. http://www.mrpt.org/. Version: 2016. – Stand: 15.08.2016

[Ope16] OpenCV: *OpenCV - About*. http://opencv.org/about.html. Version: 2016. – Stand: 20.08.2016

[Pow11] Powers, David M. W.: Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. In: *Journal of Machine Learning Technologies* 2 (2011), S. 37–63

[ROS14] ROS: *ROS - Introduction*. http://wiki.ros.org/ROS/Introduction. Version: 2014. – Stand: 20.08.2016

[SDJM10] S.Tuohy ; D.O'Cualain ; Jones, E. ; M.Glavi: Distance determination for an automobile environment using Inverse Perspective Mapping in OpenCV. In: *Signals and Systems Conference (ISSC)*. Irland, Juni 2010

[TFB05] Thrun, Sebastian ; Fox, Dieter ; Burgard, Wolfram: *Probabilistic Robotics*. The Mit Press, 2005

[Yan10] Yaniv, Ziv: Random Sample Consensus (RANSAC) Algorithm, A Generic Implementation. In: *The Insight Journal* (2010), Oktober

[YYD13] Yenikaya, Sibel ; Yenikaya, Gökhan ; Düven, Ekrem: Keeping the Vehicle on the Road – A Survey on On-Road Lane Detection Systems. In: *ACM Computing Surveys* 46 (2013)

A Anhang

A.1 Standardmäßige Einstellung der Parameter

Tabelle A.1: Standardmäßige Einstellung der Parameter

Parameter	**Bezeichnung**	**Standardwert**
a	Höhe des IPM-Bildes	320 Pixel
b	Breite des IPM-Bildes	240 Pixel
f_u	u-Wert der Brennweite der Kamera	270.077
f_v	v-Wert der Brennweite der Kamera	300.836
c_u	u-Wert des Bildmittelpunktes der Kamera	325.679
c_v	v-Wert des Bildmittelpunktes der Kamera	250.211
h	Höhe der Kamera (bzgl. Boden)	325mm
α	Pitch-Winkel der Kamera	23,4 Grad
β	Yaw-Winkel der Kamera	0,0 Grad
σ_x	Erwartete Breite der Fahrbahnmarkierungen	20
σ_y	Gewünschte Länge der Fahrbahnmarkierungen im IPM-Bild	500
r	Länge der Faltungsmatrix für die Kaltenfilterung in vertikaler Richtung	5
w	Länge der Faltungsmatrix für die Kaltenfilterung in horizontaler Richtung	7
φ	p-Quantil für die Thresholding-Operation nach Kantelfilterung	0,9875
τ	Schwellenwert für die Modellvalidierung bzw. Bestimmung der zu befahrenden Fahrspur	250