Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Alexander Kant

Bildverarbeitungsmodul zur Fahrspurerkennung
für ein autonomes Fahrzeug

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Alexander Kant

Bildverarbeitungsmodul zur Fahrspurerkennung für ein autonomes Fahrzeug

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

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

Abgegeben am 24. September 2007

Alexander Kant

Thema der Bachelorarbeit

Bildverarbeitungsmodul zur Fahrspurerkennung für ein autonomes Fahrzeug

Stichworte

Fahrspurerkennung, LTI-Lib, Impresario, Canny-Operator, Hough-Raum

Kurzzusammenfassung

In dieser Arbeit wird ein Softwaremodul zur Fahrspurerkennung für ein autonomes Fahrzeug entwickelt. Es wird in der Programmiersprache C++ ein Ansatz vorgestellt, mit dem die gestellte Aufgabe gelöst werden soll. Die dafür notwendigen Begriffe, Werkzeuge sowie Bibliotheken werden ebenfalls behandelt.

Alexander Kant

Title of the paper

Module for image processing to lane detection for autonomous vehicle

Keywords

Lane Detection, LTI-Lib, Impresario, Canny Operator, Hough-Space

Abstract

In this paper a software-module for lane detection of an autonomous vehicle is developed. An approach to solve the given task, developed in the programming language C++, is presented. Also, the required terms, tools and libraries are discussed.

Alles, was ich denke, haben schon andere gedacht.
Alles, was ich schreibe, haben schon andere geschrieben.
Doch alles ist neu - durch die Verbindung

Karl-Josef Durwen

Danksagung

An dieser Stelle möchte ich meinen Dank denjenigen aussprechen, die mir bei der Realisierung dieser Arbeit zur Seite standen.

Als erstes möchte ich mich bei meinen beiden Betreuern, Prof. Dr.-Ing. Andreas Meisel sowie Prof. Dr. rer. nat. Stephan Pareigis, für ihre Unterstützung während meiner Bachelorarbeit bedanken. Ohne diese beiden Menschen wäre diese Arbeit nicht zu Stande gekommen.

Ebenso möchte ich mich bei folgenden intelliTruck-Teammitgliedern für ihre Unterstützung und Zusammenarbeit bedanken: Enrico Hensel, Sven Geibert, Kordian Kubert, Eike Jenning und Umut Nar.

Des Weiteren möchte ich mich bei einem weiteren hilfsbereiten Mensch bedanken: Herrn Bruno Carstensen.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Heute werden immer mehr verschiedene Assistenzsysteme für Fahrzeuge entwickelt und verwendet. Bereits Ende der 80er Jahre wurde von EUREKA (European Research Coordination Agency) ein Projekt Namens PROMETHEUS (Programme for a European Traffic with Highest Efficiency and Unprecedented Safety) gestartet, welcher dazu dienen sollte, Technologien im Bereich der Kommunikation und Information für Fahrzeuge und Straßeninfrastruktur zur Anwendungsreife fortzuentwickeln. Am Ende des Projektes im Jahr 1994 hat ein mit zahlreichen Hochleistungs-Kameras und Rechnern ausgestattetes Fahrzeug die 3-spurige Autoroute A1 in Paris über 1000km völlig autonom gefahren. Dabei hat das Fahrzeug bei den Geschwindigkeiten von bis zu 130 km/h Konvoifahrten mit automatischer Abstandshaltung sowie Überholmanöver selbständig durchgeführt. Das Fahrzeug konnte in seiner Umgebung bis zu 6 andere Fahrzeuge erfassen und auswerten.

Es gibt eine Reihe von marktreifen Assistenzsystemen, die bereits zur Standardausstattung gehören:

- ABS (Antiblockersystem)
- ESP (Elektronisches Stabilitätsprogramm)
- ASR (Antriebsschlupfregelung)
- ...

Alle diese Systeme arbeiten lediglich mit den internen Sensorinformationen des Fahrzeugs, wie z.B. der Umdrehungszahl der Räder, Lenkwinkel des Steuers, Querbeschleunigung des Fahrzeugs etc. Assistenzsysteme, die autonom ein Fahrzeug von einem Ort zum anderen zuverlässig steuern unter der Berücksichtigung aller Verkehrsregeln werden immer noch entwickelt und getestet.

1.2 Aufgabenstellung

In dieser Arbeit wird ein Softwaremodul zur Fahrspurerkennung für ein autonomes Fahrzeug entwickelt. Hierfür werden bekannte und erprobte Methoden zum Einsatz kommen. Das Modul benötigt ein Videosignal mit einer Fahrspurmarkierung als Input, woraus die Orientierung des Fahrzeugs berechnet wird.

1.3 Arbeitsfeld

Die Arbeit erfolgt größtenteils in den Räumlichkeiten der HAW-Hamburg. Das Modul wird mit Hilfe der Bibliotheken der LTI-Lib (Lehrstuhl für Technische Informatik) entwickelt. Als Entwicklungsumgebung wird MS Visual C++ verwendet und das Produkt „Impressario“, welches mit den Bibliotheken arbeiten kann, wird als Testumgebung verwendet. Des Weiteren ist es geplant, das entwickelte Modul in einen Testmodellfahrzeug (Abbildung 1.1) der HAW-Hamburg einzubinden um es unter den Realbedingungen testen zu können.

Abbildung 1.1: ein Testfahrzeug der HAW-Hamburg

1.4 Aufbau der Arbeit

Kapitel 1 dient zur Einführung. Nach einer kurzen Motivation wird erklärt was die Zielsetzung dieser Arbeit ist und in welchem Umfeld die Arbeit geschrieben wird.

In **Kapitel 2** wird die Einordnung des entwickelten Verfahrens dargestellt. Hier werden verschiedene Einsatzbereiche für autonome Fahrzeuge sowie deren Stand der Technik beschrieben.

Die zugrunde liegenden Bildverarbeitungsverfahren werden in **Kapitel 3** erklärt, damit im darauffolgenden **Kapitel 4** die Realisierung und Implementierung des Moduls nachvollzogen werden kann.

Anschließend werden in **Kapitel 5** die Testläufe, Ergebnisse und Erfahrungen beschrieben.

Zusammenfassung sowie ein Ausblick für weitere Arbeiten kommt abschließend in **Kapitel 6**.

2 Einordnung des entwickelten Verfahrens

2.1 Stand der Technik

Seit den frühen 60er wird versucht Maschinen visuelle Fähigkeiten zu verleihen. Erst in den 70ern gelang es dem Stanford Research Institute den allerersten mobilen Roboter (Abbildung 2.1) zu konstruieren, der mit einer Kamera ausgestattet war.

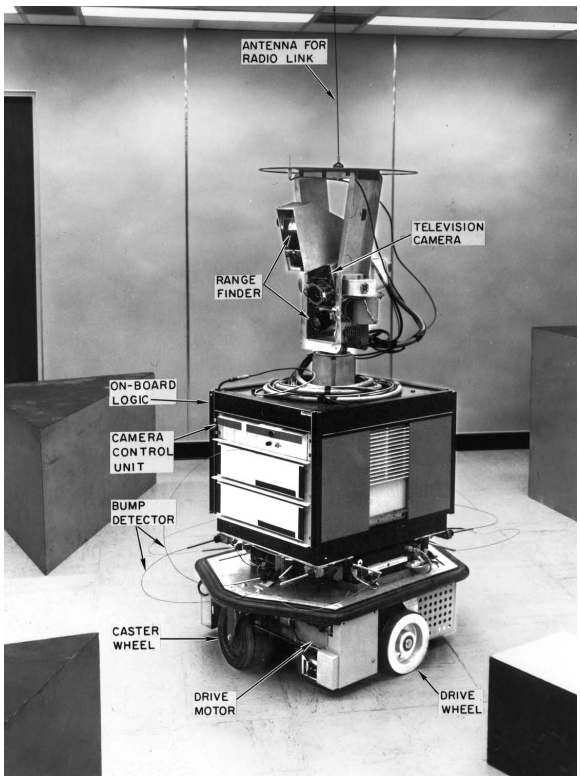

Abbildung 2.1: der erste mobile Roboter Namens „Shakey“ (www.frc.ri.cmu.edu)

Shakey wurde mit Hilfe eines Großrechners ferngesteuert und konnte an einem guten Tag Blöcke von einem Ort zu einem anderen Ort verschieben. Dazu wurde ein schwaches, kan-

tenbasiertes Algorithmus zusammen mit Lasersensoren für Entfernungsvermessung verwendet.

In den letzten 30 Jahren wurden im Bereich der Informationstechnologie rasante Fortschritte erzielt. Somit stehen heute deutlich leistungsfähigere Rechner zur Verfügung als in den 70er Jahren. Dank dieses Fortschrittes ist es möglich neue Ansätze zur Fahrspurerkennung zu entwickeln und zu realisieren. So wurde im Jahr 2006 im Rahmen des Forschungsverbundes des FOBIAS „Bioanaloge Sensomotorische Assistenz“ der Bayerischen Forschungsstiftung ein 4D-Ansatz zur schritthaltenden Bildfolgeverarbeitung in Zusammenarbeit mit dem Industriepartner Audi AG zur Fahrspurerkennung vorgestellt.

Dieser Ansatz vermisst und klassifiziert kontinuierlich die Fahrspurmarkierungen und nutzt die zeitliche Dimension um den weiteren Verlauf der Fahrspurmarkierung vorherzusagen (Abbildung 2.2).

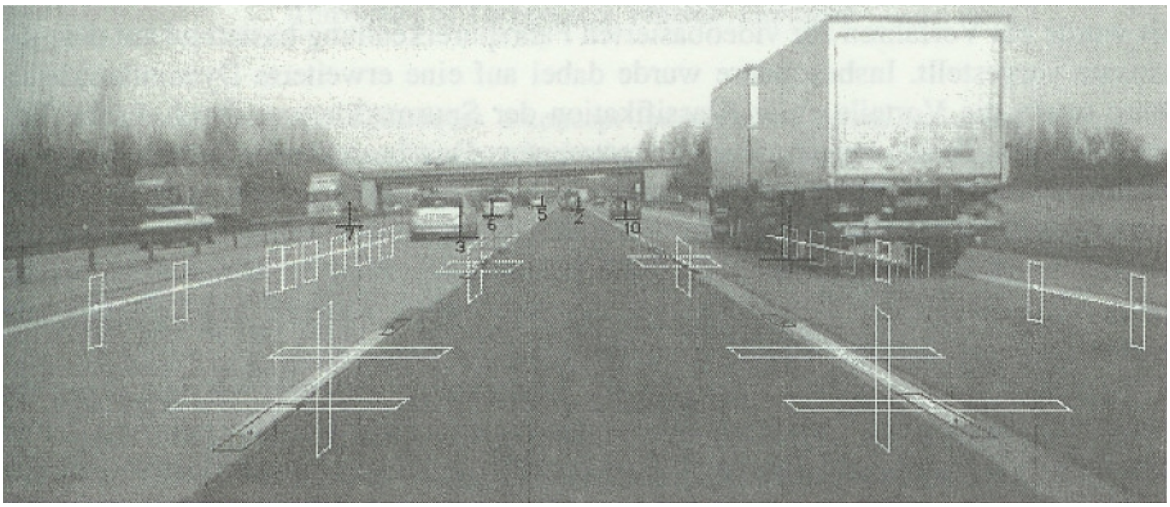

Abbildung 2.2: Fahrspurerkennung mit dem 4D-Ansatz während einer Autobahnfahrt (Quelle: Levi u. a. (2005))

2.2 Designfestlegungen

Die Software zur Fahrspurerkennung kann als eine selbständige Klasse realisiert werden. Es ist allerdings sinnvoll die Software in unterschiedliche Module aufzuteilen, die sich alleine um ihre Aufgabe kümmern. Dadurch wird die Software übersichtlicher und lässt sich in der Simulationsumgebung Impressario besser analysieren. In der Abbildung 2.3 wird der Aufbau der Software Abstrakt dargestellt.

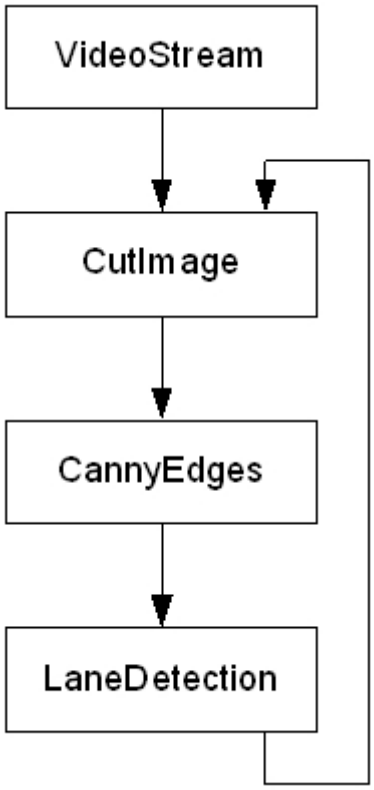

Abbildung 2.3: Abstrakte Darstellung des Systems

Das Modul `VideoStream` liest von einem Speichermedium eine Videodatei, Bildsequenzen oder greift direkt auf eine Videokamera zu und wandelt das vorliegende Datenmaterial nach `lti::channel8`[1], welches an das Modul `CutImage` zur weiteren Verarbeitung weitergereicht wird. Die Aufgabe des `CutImage`-Moduls ist es, aus einem übergebenen Bild eine ROI auszuschneiden, damit der `CannyEdges`-Modul nicht das ganze Bild, sondern nur die ROI bearbeitet und das binärisierte Bild weiter an das letzte `LaneDetection`-Modul reicht. Das `LaneDetection`-Modul erkennt den Verlauf der Fahrspurmarkierun-

[1] lti::channel8 ist ein von der LTI Lib zur Verfügung gestellter 8bit Datentyp für Grauwertbilder

gen, berechnet die Orientierung des Fahrzeugs sowie die Position und Größe der nächsten ROI für das `CutImage`-Modul.

Um eine Fahrspurmarkierung erkennen zu können muss diese zuerst definiert werden. Dabei müssen folgende Randbedingungen bedacht und festgelegt werden:

- Form der Markierung
- Farbe der Markierung

Form ist das wichtigste Kriterium, um eine Markierung zu erkennen. Die Form der Markierung hat den größten Einfluss auf die Performance des Systems. Je komplexer die Form ist, desto rechenintensiver ist deren Erkennung. Ein Kreis ist eine mögliche Form. Der Vorteil dieser Form ist, dass Kreise in der Natur selten vorkommen. Dies erhöht die Robustheit des Systems, da die Wahrscheinlichkeit hoch ist, dass die erkannten Kreise tatsächlich die Markierungen darstellen. Der Nachteil dieser Form ist, dass es rechenintensiv ist diese Form zu erkennen. Es müssen für die Erkennung der kreisförmigen Fahrspurmarkierung drei Parameter bestimmt werden:

- Radius
- X-Koordinate
- Y-Koordinate

Es ist ebenfalls notwendig mindestens zwei Fahrspurmarkierungen zu erkennen, um eine Aussage über deren Verlauf treffen zu können. Die Blickperspektive findet in den meisten Fällen aus einem Winkel <90° statt, weshalb die Markierungen nicht mehr kreisförmig erscheinen. Aus diesem Grund ist es noch rechenintensiver diese zu erkennen. Die Abbildung 2.4 verdeutlicht dies.

Abbildung 2.4: kreisförmige Fahrspurmarkierungen

Eine weitere mögliche Form der Markierung, ist ein Rechteck. Der Vorteil dieser Form ist, dass es anhand der längsten Seite der Markierung möglich ist eine Aussage über den weiteren Verlauf zu treffen. Da eine Gerade durch zwei Parameter dargestellt werden kann:

- R
- Winkel[2]

ist das Erkennen von Geraden im Raum mit weniger Rechenaufwand verbunden als das Erkennen von Kreisen. Der Nachteil dieser Form ist, dass es in der Natur deutlich mehr Geraden vorkommen als Kreise.

Um eine Fahrspurmarkierung sicherer Erkennen zu können, muss eine weitere Eigenschaft herangezogen werden: die Farbe. Nach der Form der Markierung ist die Farbe das zweitwichtigste Kriterium damit eine Fahrspurmarkierung zuverlässiger erkennt werden soll. Die Wahl der Farbe ist enorm wichtig, da es den Einfluss darauf hat, ob eine Fahrspurmarkierung überhaupt erkannt wird oder nicht. Erst wenn die Markierung sich von der Umgebung hervorhebt, ist eine Erkennung denkbar. Eine helle Farbe ist somit eine gute Wahl, da in der Natur dunkle Farben öfter vorkommen als die hellen Farben.

Für diese Arbeit wurden weiße Aluminiumstreifen verwendet. Weiß ist eine Farbe, welche eine Form aus ihrer Umgebung deutlich hervorhebt. Die Länge eines Streifens beträgt 60cm und deren Breite 5cm (Abbildung 2.5).

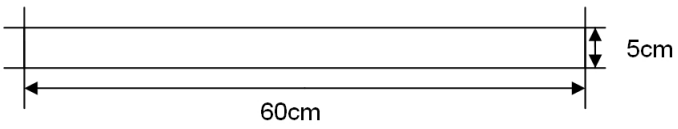

Abbildung 2.5: Skizze einer Fahrspurmarkierung

Aluminiumstreifen sind leicht und somit für einen Transport gut geeignet. Zugleich sind diese schwer und robust genug um nicht durch Witterungsverhältnisse wie Wind und Regen in ihrer Funktionalität beeinträchtigt zu werden. Die Länge von 60cm steigert die Robustheit des Algorithmus gegenüber möglichen Störungen bei der Kantenerkennung und hilft dem Algorithmus den Verlauf der Fahrspurmarkierungen genauer zu bestimmen.

[2] Die Hessesche Normalform verwendet zur Darstellung einer Geraden zwei Parameter: Die Länge R der Normalen vom Ursprung sowie deren Winkel zu der X-Achse

3 Zugrunde liegende Bildverarbeitungsverfahren

Die meisten digitalen Bilder liegen in einem RGB-Farbschema vor. Da jeder Kanal aus einem 8-bit Wert besteht, ist es möglich mit allen drei Kanälen (R, G, B) $256^3 = 16.777.216$ Farben darzustellen. Der in dieser Arbeit verwendete Operator zur Kantenextraktion setzt Grauwertbilder voraus. Aus diesem Grund ist es notwendig RGB-Bilder in Grauwertbilder umzuwandeln.

3.1 Farbbildverarbeitung

Die Umwandlung eines Bildes, welches in einem RGB Format (Abbildung 3.1) vorliegt in ein Grauwertbild (Abbildung 3.2) geschieht mittels der Formel $Grauwert = \frac{(R+G+B)}{3}$. Eine Gewichtung der einzelnen Farbkanäle ist nicht notwendig, da dies nur für die menschliche Wahrnehmung von Interesse ist.

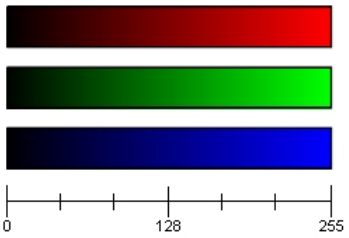

Abbildung 3.1: RGB (3-Kanal) Farbschema

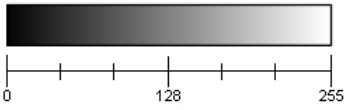

Abbildung 3.2: Grauwert (1-Kanal) Farbschema

3.2 Kantenerkennung mit dem Canny-Operator

In Grauwertbildern werden Kanten durch starke Helligkeitsunterschiede zwischen zwei benachbarten Pixels dargestellt. Da solche Helligkeitsunterschiede auch bei einem Bildrauschen auftreten, muss das Grauwertbild geglättet werden. Hierfür verwendet der Canny-Algorithmus eine Glättungsmaske (Abbildung 3.3), die einer Gaußschen Normalverteilung ähnelt.

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Abbildung 3.3: Glättungsmaske

Wenn ein Bild mit einer Maske bearbeitet wird, so spricht man auch von einer Bildfaltung. Die Bildfaltung mit einem Faltungskern (Abbildung 3.4) kann mittels der Formel durchgeführt werden: $f_{out}(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) * f(x + s, y + t)$.

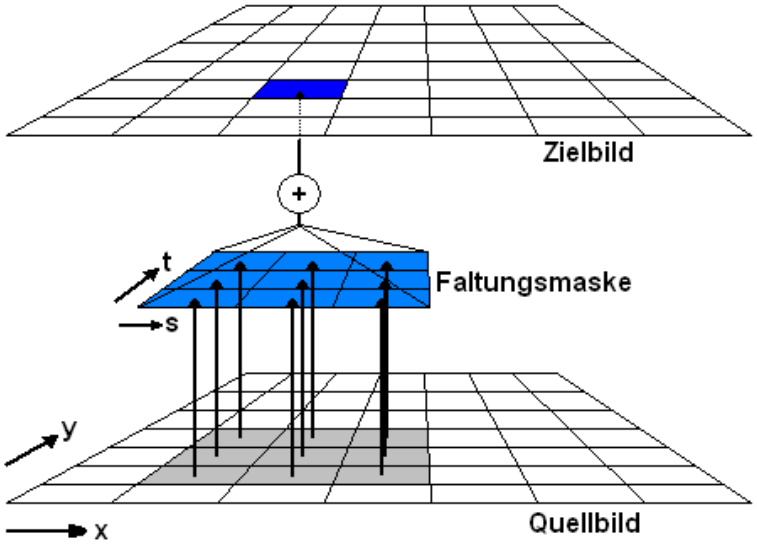

Abbildung 3.4: Bildfaltung mit einem Faltungskern

Je größer die Glättungsmaske ist, desto stärker wird das Rauschen unterdrückt. Es muss bedacht werden, dass mit der steigenden Größe einer Glättungsmaske Bildinformationen und somit die Kanten verloren gehen. In der Abbildung 3.5 wird dies durch die Anwendung verschiedener Größen einer Glättungsmaske veranschaulicht.

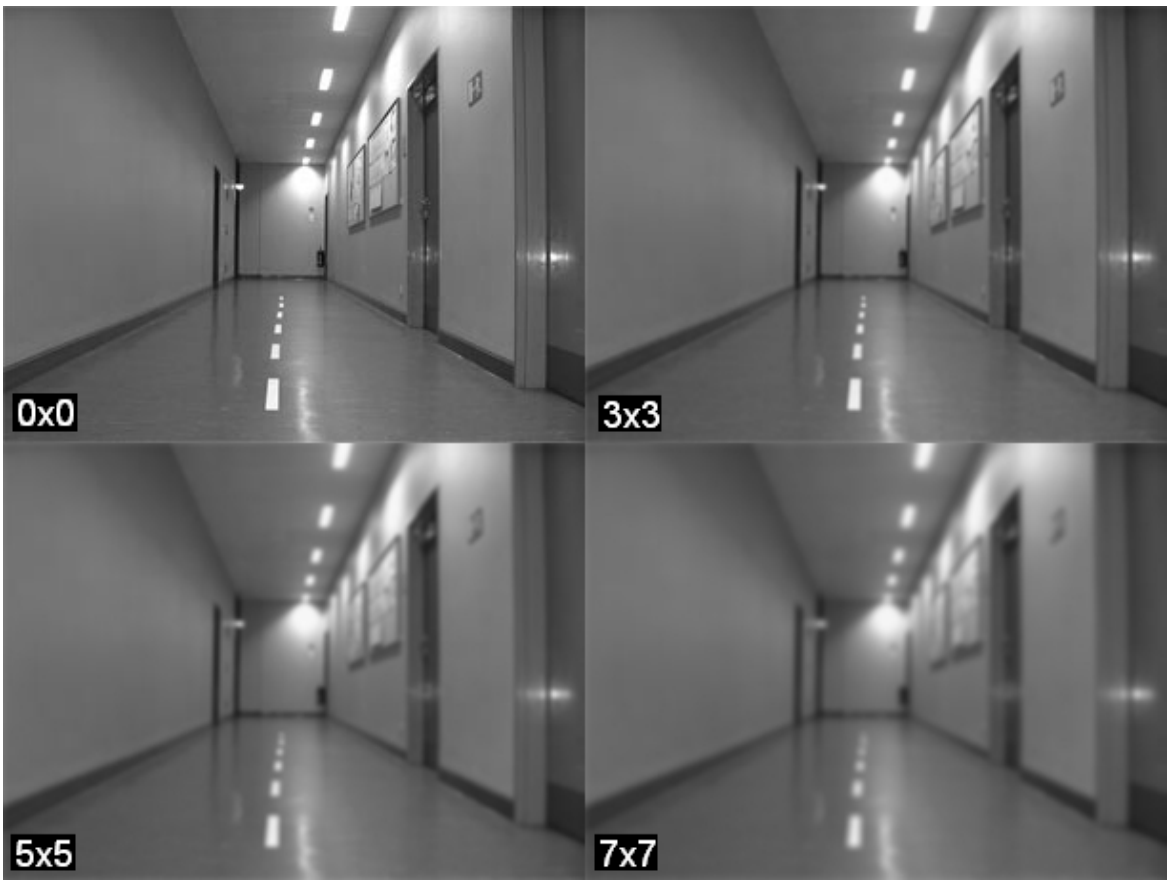

Abbildung 3.5: Faltungsergebnisse eines Grauwertbildes mit verschiedenen Glättungsmasken

Nach der Anwendung des Canny-Operators auf die unterschiedlich geglätteten Bilder werden, wie in der Abbildung 3.6 dargestellt, folgende Ergebnisse erzielt:

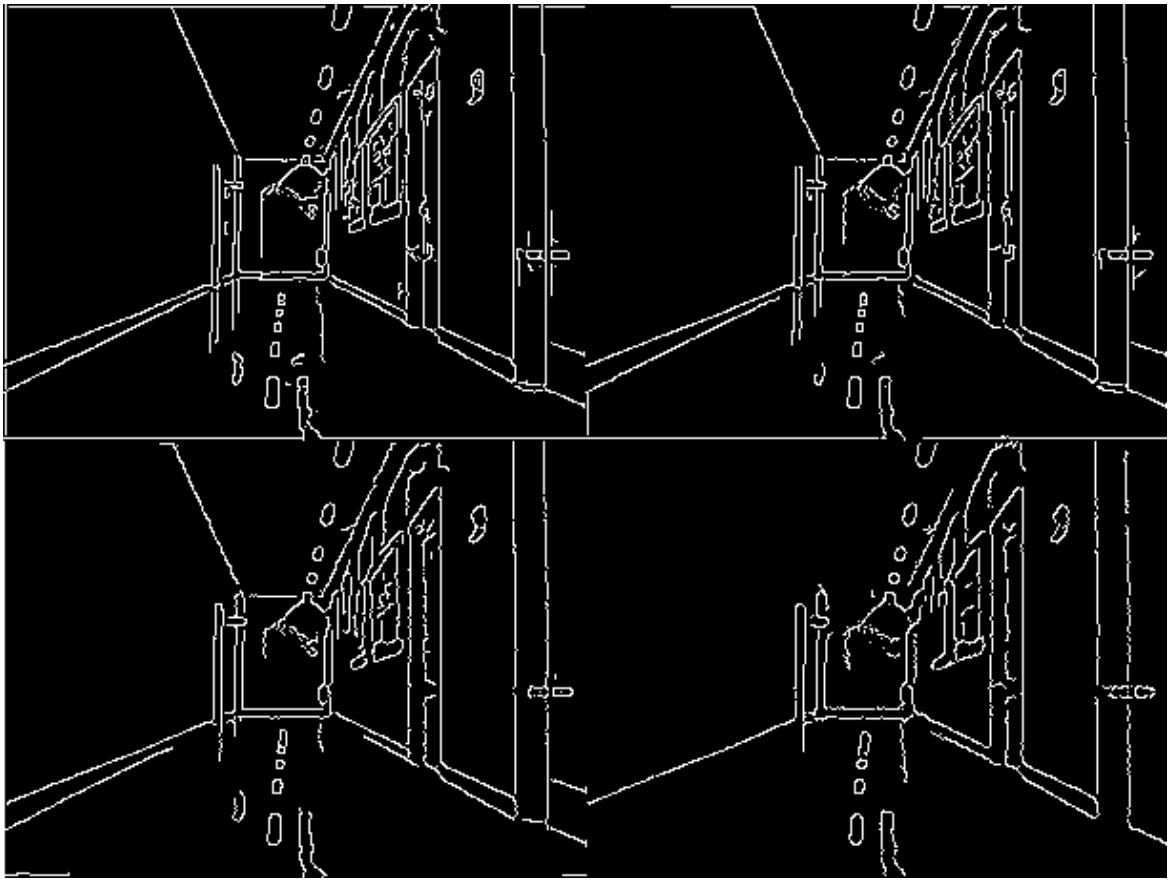

Abbildung 3.6: Ergebnisse der Anwendung eines Canny-Algorithmus

3.3 Extraktion der Geradenparameter

Für die Extraktion der Geradenparameter wird die Hough-Transformation verwendet. Diese ist für die Erkennung beliebiger geometrischen Figuren in einem binären Bild gut geeignet. Hierfür wird ein Hough-Raum[3] (auch als Parameterraum bekannt) erschaffen, in welchen die Parameter aller möglichen geometrischen Figuren, für alle Punkte im Binärbild welche üblicherweise auf einer Kante liegen, eingetragen werden. Auf dieser Weise wird ein Raum erschaffen, in welchem ein Punkt eine geometrische Figur in Binärbild repräsentiert.

[3]Hough-Raum ist eine n-dimensionale Matrix, wobei n=Anzahl der Parameter, die notwendig sind um eine geometrische Figur darzustellen.

Eine Gerade wird durch 2 Parameter repräsentiert. Die bekannteste Form der Geradengleichung ist $y = mx + b$. Diese Form ist für die Lösung des Problems nicht geeignet, da bei einer vertikalen Geraden $m = \infty$ ist. Aus diesem Grund wird für die Geradendarstellung die Hessesche Normalform verwendet. In der Abbildung 3.7 wird eine Gerade mit der Geradengleichung $y = -x + 5$ dargestellt.

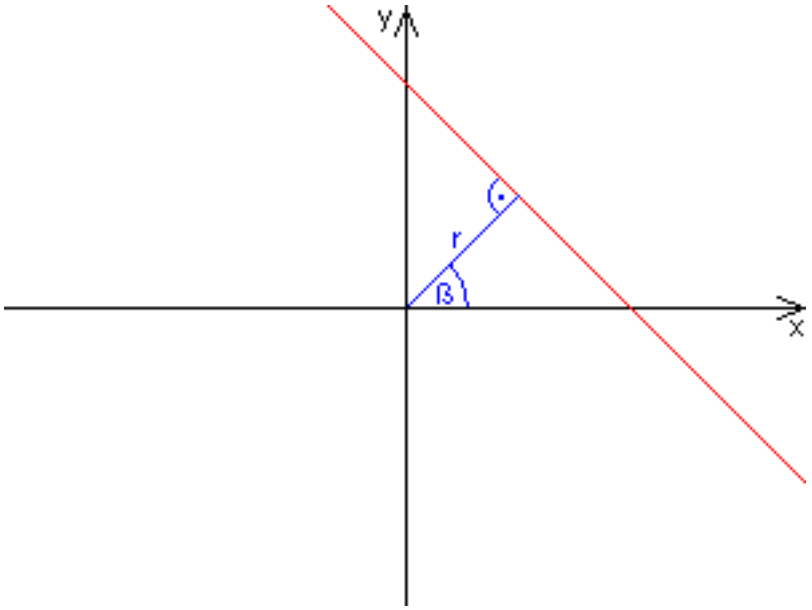

Abbildung 3.7: Geradendarstellung in Hessescher Normalform

Dieselbe Gerade wird in der Hessescher Normalform mit $\beta = 45°$ und $r = 3,54$ angegeben.

Für die Erkennung der Geraden im Binärbild wird ein zweidimensionaler Hough-Raum verwendet. Die erste Dimension repräsentiert den Winkel β, die zweite Dimension die Länge der Normalen r vom Ursprung. Das Binärbild wird pixelweise untersucht. Dabei werden für jeden weißen Pixel alle möglichen Geraden mittels der Gleichung $r = x*cos(\beta)+y*sin(\beta)$ bestimmt und in den Hough-Raum eingetragen. Jeder untersuchter, weißer Pixel erzeugt im Hough-Raum eine Parabel. Nach der vollständigen Untersuchung des Binärbildes wird im erstellten Hough-Raum nach Häufungspunkten gesucht, welche durch Überschneidungen mehrerer Parabeln erzeugt werden. Diese repräsentieren Geraden im Binärbild.

Die Abbildung 3.8 zeigt einen Hough-Raum mit mehreren Häufungspunkten wobei die Abbildung 3.9 das Ergebnis einer Kantenerkennung zeigt.

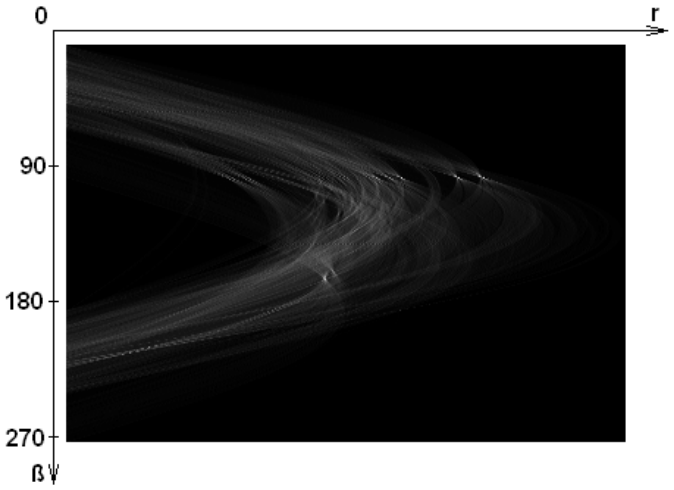

Abbildung 3.8: visuelle Darstellung eines Hough-Raums

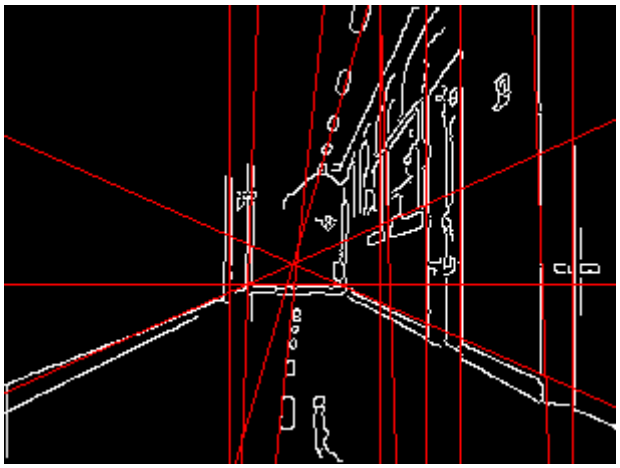

Abbildung 3.9: visualisiertes Ergebnis einer Kantenerkennung

4 Realisierung

In dieser Arbeit wird für die Realisierung eines Bildverarbeitungsmoduls zur Fahrspurerkennung ein kantenbasiertes Verfahren eingesetzt. Als Programmiersprache wird C++ und die Entwicklungsumgebung „Microsoft Visual Studio .NET 2003“ verwendet. Die für die Realisierung verwendete Bibliothek (LTI-Lib) setzt diese Entwicklungsumgebung voraus.

Als Markierung werden, wie in dem Abschnitt 2.2 beschrieben, gefärbte Aluminiumstreifen verwendet (Abbildung 4.1).

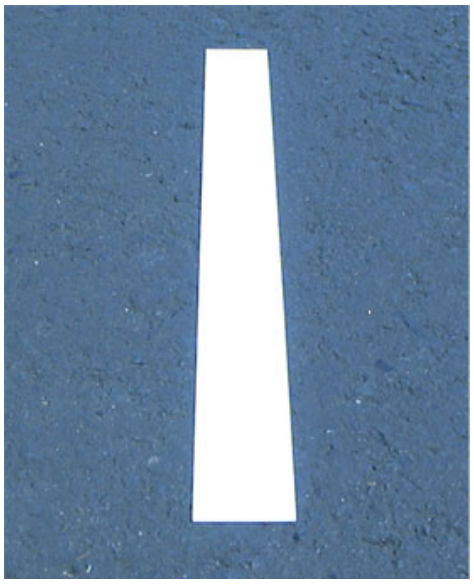

Abbildung 4.1: Fahrspurmarkierung

Das zu entwickelnde Modul benötigt, wie im Kapitel 1.2 erwähnt, ein Videosignal als Eingang. Hierfür wird eine von der HAW-Hamburg zur Verfügung gestellte BLIZZARD-60-U2 Kamera von Photonfocus verwendet (Abbildung 4.2).

Abbildung 4.2: BLIZZARD-60-U2 Kamera (www.photonfocus.com)

Technische Daten:

- Schwarz-Weiß CMOS Sensor
- 750x400 Pixel
- 60 Bilder/Sekunde
- Weitere Daten können der Herstellerseite www.photonfocus.com entnommen werden

Als Simulationsumgebung wird Impressario verwendet. Diese Software bietet die Möglichkeit an, Objektparameter eines beliebigen Moduls zur Laufzeit zu ändern und somit den Ablauf des Systems zu beeinflussen. Des Weiteren ist es möglich, Module miteinander nach Datenfluss-Prinzip zu koppeln, um so ein Gesamtsystem erstellen zu können. Das Gesamtsystem des Bildverarbeitungsmoduls zur Fahrspurerkennung ist in der Abbildung 4.3 zu sehen.

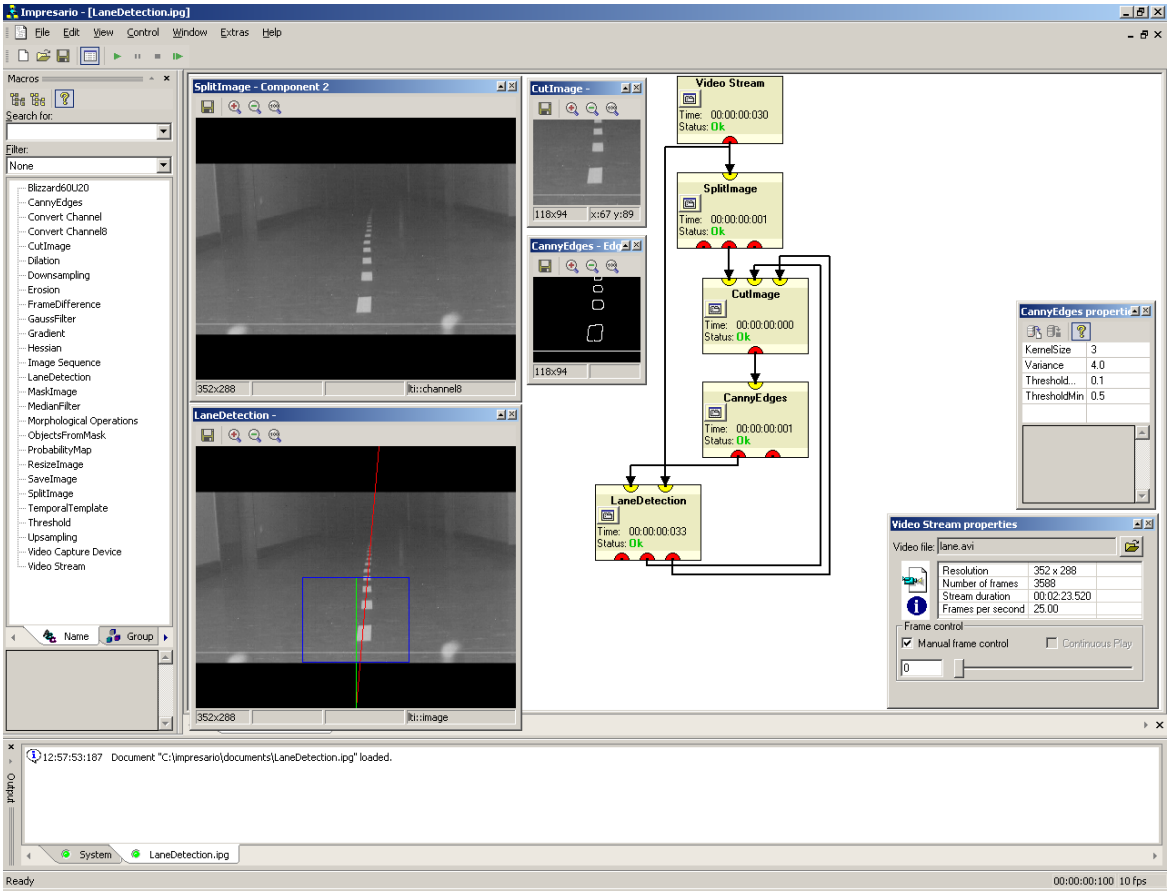

Abbildung 4.3: Bildverarbeitungsmodul zur Fahrspurerkennung in Impressario

4.1 Das Erkennen der Fahrspurmarkierung

Bevor die Fahrspurerkennung starten kann, müssen folgende Vorbedingungen erfüllt werden:

- das Fahrzeug muss sich auf der Spurmarkierung befinden,
- das Fahrzeug muss in die Fahrtrichtung ausgerichtet sein.

Da der Videostrom aus der Kamera (BLIZZARD-60-U2) bereits aus Grauwertbildern besteht, kann der Canny-Operator direkt angewendet werden. Die ansonsten notwendige Bildumwandlung zu Grauwertbildern wird vermieden, die Rechenzeit eingespart. Um die Performance weiter zu steigern, wird eine ROI (Region of Interest) eingeführt. Da es zu Anfang nicht bekannt ist, wo die Fahrspurmarkierung genau verläuft, hat die ROI folgende Größe und Position:

- P1($\frac{1}{3} Bildbreite$, $\frac{1}{2} Bildhöhe$)
- P2($\frac{2}{3} Bildbreite$, $Bildhöhe$)

Die Abbildung 4.4 zeigt die Sicht auf eine Fahrbahn aus der Fahrzeugperspektive. Die Initial-ROI wird hier durch ein blaues Rechteck dargestellt.

Abbildung 4.4: initial Region of Interest (ROI)

Diese Initial-ROI wird ebenfalls ausgewählt, sobald das System die Fahrspurmarkierung nicht mehr erkennt oder diese die ROI verlässt.

Der aus der LTI-Lib verwendete Canny-Operator kann nur auf ein komplettes Bild angewandt werden. Aus diesem Grund muss die ROI aus dem Quellbild ausgeschnitten und als neues Bild dem Canny-Operator zugeführt werden. In der Abbildung 4.5 links ist die ausgeschnittene ROI zu sehen. Das Bild in der Mitte zeigt das Ergebnis des Canny-Operators sowie die erkannten vertikale Kanten der Fahrspurmarkierung im rechten Bild.

Abbildung 4.5: Ergebnis einer Bildverarbeitung der ROI

Der hierfür erstellte Hough-Raum ist in der Abbildung 4.6 (linkes Bild) zu sehen.

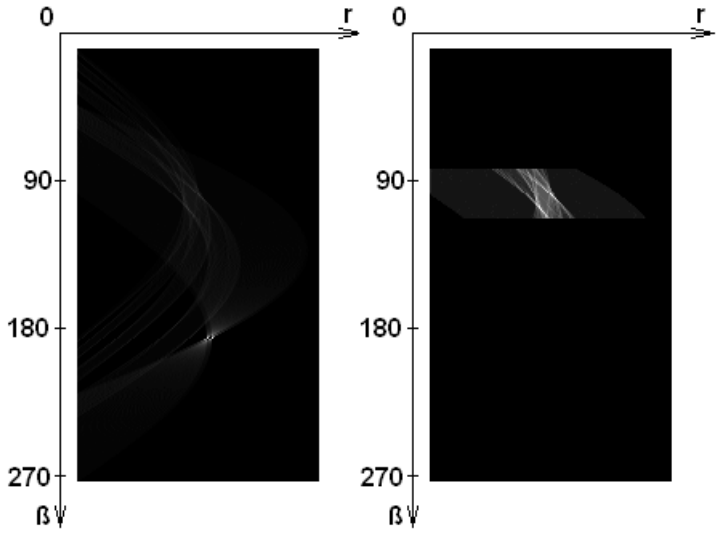

Abbildung 4.6: erstellter Hough-Raum der ROI

Da ein Fahrzeug immer einer Spur entlang fährt, ist die Fahrspurmarkierung in den entsprechenden Bildsequenzen vertikal angeordnet. Dies hat zu Folge, dass das System nur noch vertikale Linien erkennen muss. Alle horizontalen Linien können somit ignoriert werden. Die relative Position sowie der Winkel der Spur zum Fahrzeug beträgt nicht immer 90°. Deshalb wird hier von einem relativen Fehler von 5% ausgegangen (15°). Aus diesem Grund muss der Hough-Raum nicht komplett, sondern im Bereich von 90° +/- 15° aufgebaut werden (Abbildung 4.6, rechtes Bild).

Hierdurch wird die Rechenzeit weiter eingespart und die Performance erhöht. Die Abbildung 4.7 zeigt einen Performancevergleich zwischen einem komplett aufgebauten Hough-Raum (270°) und einem reduzierten Hough-Raum (30°).

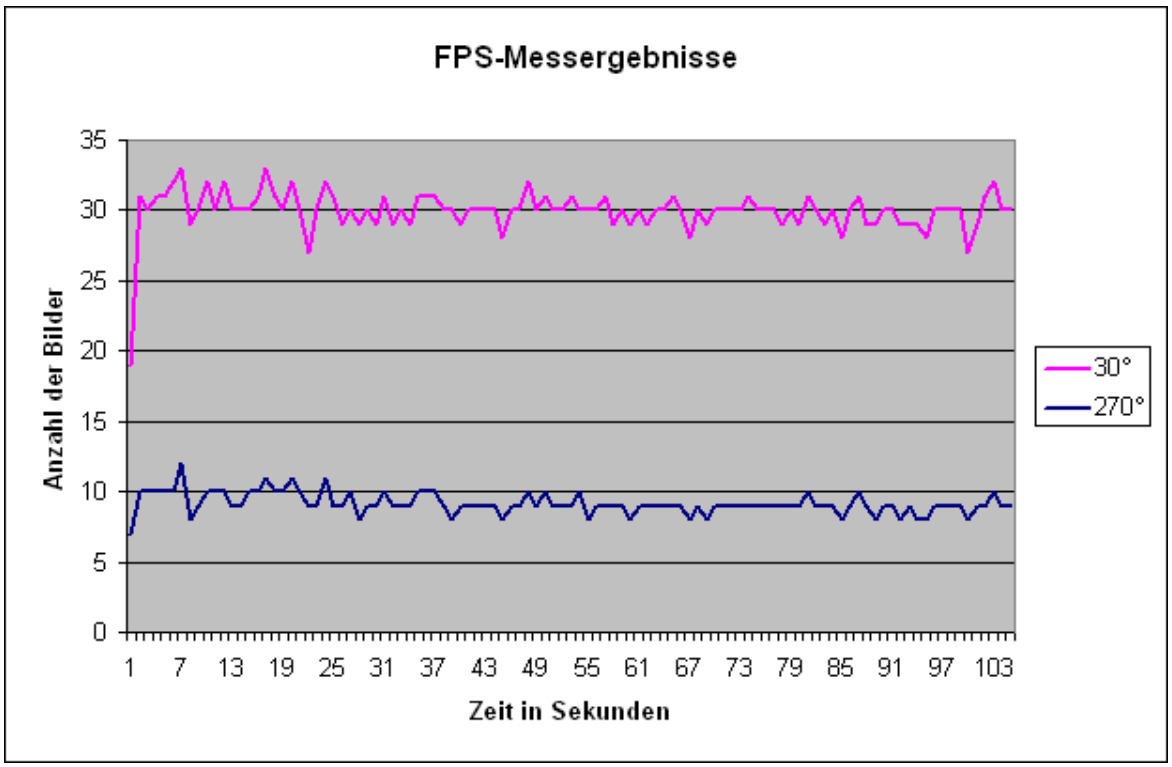

Abbildung 4.7: FPS-Messergebnisse

Nach der Erkennung der beiden vertikalen Kanten der Fahrspurmarkierung wird eine Mittellinie errechnet. Die Position und Orientierung dieser entspricht dem Verlauf der Fahrspurmarkierungen innerhalb der ROI. Nach der erfolgreichen Erkennung wird die ROI neu berechnet um die Performance zu steigern.

4.2 Die entwickelten Klassen

In dieser Arbeit wurden 2 Klassen entwickelt:

- CCutImage
- CLaneDetection

4.2.1 CCutImage - Klasse

Die Aufgabe der `CCutImage`-Klasse ist die Erstellung eines neuen Bildes aus einem Quellbild in Abhängigkeit von der ROI. Diese wird durch das Übergeben zweier Punkte definiert. Die Abbildung 4.8 zeigt ein Klassendiagramm der `CCutImage`-Klasse.

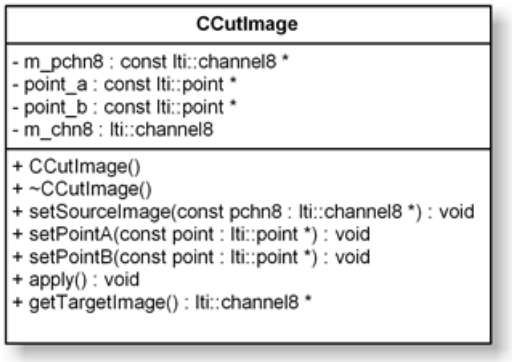

Abbildung 4.8: Klassendiagramm der CCutImage-Klasse

Methodenname	Funktionsbeschreibung
`setSourceImage()`	Ist eine Setter-Methode. Diese wird aufgerufen, um ein Quellbild der Klasse zu übergeben. Aus diesem wird die Spurverlaufinformation gewonnen.
`setPointA()`	Ist eine Setter-Methode. Dieser wird ein Punkt übergeben, welches die obere, linke Ecke der ROI markiert.
`setPointB()`	Ist eine Setter-Methode. Dieser wird ein Punkt übergeben, welches die untere, rechte Ecke der ROI markiert.
`apply()`	Diese Methode wird aufgerufen, um den Schneideprozess zu initiieren. Sollte die ROI zu diesem Zeitpunkt, wie es beim allerersten Mal der Fall ist, nicht bekannt sein, wird wie in Kapitel 4.1 beschrieben, von der initial-ROI ausgegangen
`getTargetImage()`	Ist eine Getter-Methode. Diese Methode liefert das ausgeschnittene Bild zurück.

Tabelle 4.1: Methodenbeschreibung der CCutImage-Klasse

4.2.2 CLaneDetection - Klasse

Die Klasse CLaneDetection ist der Kern des gesamten Systems. Die Aufgaben dieser Klasse sind:

- Erstellung des Hough-Raums
- Bestimmung der linken, sowie der rechten Kante der Fahrspurmarkierung
- Berechnung der Mittellinie
- Berechnung der ROI für das nächste Bild

Das Klassendiagramm der CLaneDetection-Klasse ist in der Abbildung 4.9 zu sehen.

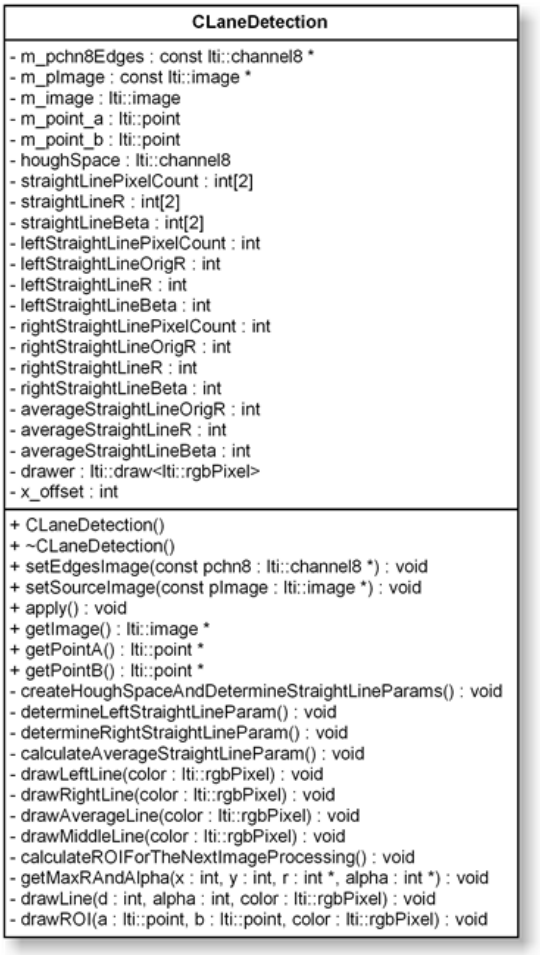

Abbildung 4.9: Klassendiagramm der CLaneDetection-Klasse

Methodenname	Funktionsbeschreibung
`setEdgesImage()`	Ist eine Setter-Methode. Ein durch den Canny-Operator verarbeitetes Bild wird dieser Methode übergeben.
`setSourceImage()`	Ist eine Setter-Methode. Dieser wird das Originalbild übergeben um in diesem die ROI sowie die erkannte Fahrspurrichtung visualisieren zu können.
`apply()`	Der Erkennungsprozess wird mit dieser Methode initiiert.
`getImage()`	Das Originalbild mit der eingetragenen ROI sowie möglichen Fahrspurrichtung wird durch diese Methode abgeholt.
`getPointA()`	Der linke, obere Punkt der neu errechneten ROI wird durch diese Methode abgeholt.
`getPointB()`	Der rechte, untere Punkt der neu errechneten ROI wird durch diese Methode abgeholt.
`createHoughSpace AndDetermine StraightLine Params()`	Diese Methode erstellt ein Hough-Raum. Aus Performancegründen werden dabei die beiden vertikalen Geraden bestimmt, die den äußeren Kanten der Fahrspurmarkierungen entsprechen.
`determineLeft StraightLine Param()`	Diese Methode errechnet die Geradenparameter[4] der linken Gerade für das Originalbild.
`determineRight StraightLine Param()`	Diese Methode errechnet die Geradenparameter der rechten Gerade für das Originalbild.
`calculateAverage StraightLine Param()`	Errechnet die Mittellinie aus den beiden zuvor erkannten Geraden.
`calculateROIFor TheNextImage Processing()`	Benutzt die Mittellinie um die ROI für das nächste Bild neu zu bestimmen.

Tabelle 4.2: Methodenbeschreibung der CLaneDetection-Klasse

[4] Die Neuerrechnung der Geradenparameter ist notwendig, da die alten Parameter nur in der ROI gültig sind. Die Abbildung 4.10 verdeutlicht dies.

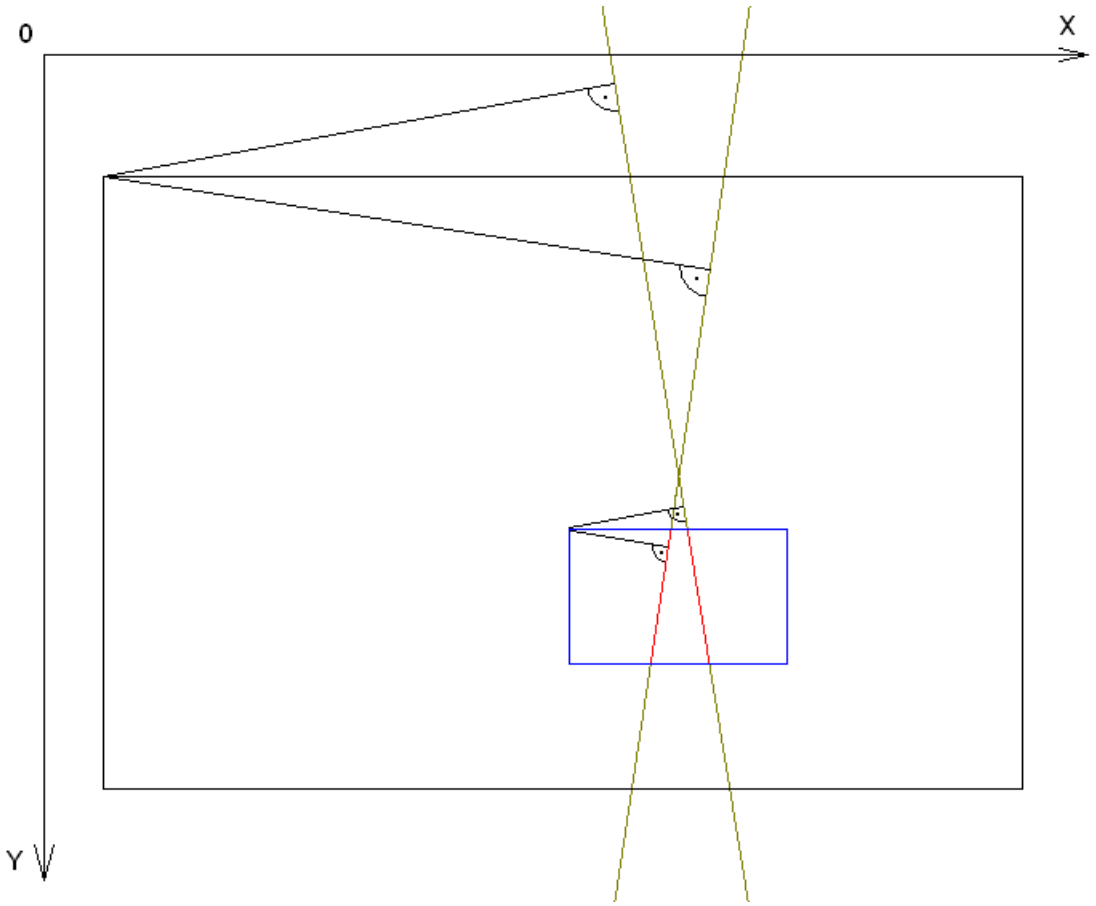

Abbildung 4.10: Sub-Koordinatensystem (ROI) im Gesamtbild

5 Testläufe und Ergebnisse

Zum Testen des Moduls wurden mehrere Aufnahmen bei verschiedenen Testfahrten gemacht. Diese zeigen realistische Situationen, mit denen der Algorithmus später konfrontiert sein wird.

5.1 Fahrspurerkennung

Bei dem ersten Außentest wurde die Kamera in einem Neigungswinkel von 0° angeordnet, wie in der Abbildung 5.1 zu sehen ist.

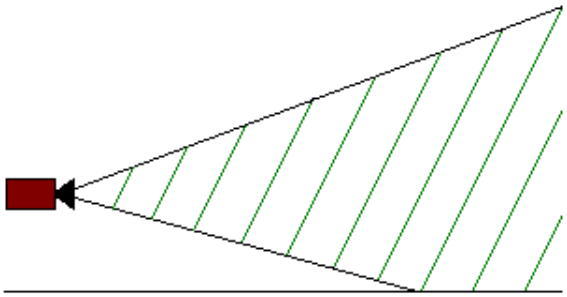

Abbildung 5.1: Ausrichtung der Kamera bei einem Neigungswinkel von 0°

Hierbei wurden die ersten Schwächen des Algorithmus sichtbar. Bei einer ungünstigen Ausrichtung der Kamera werden alle helleren Objekte im Blickwinkel des Fahrzeugs als Fahrspurmarkierungen missinterpretiert. Die Abbildungen 5.2 sowie 5.2 veranschaulichen solche Situationen.

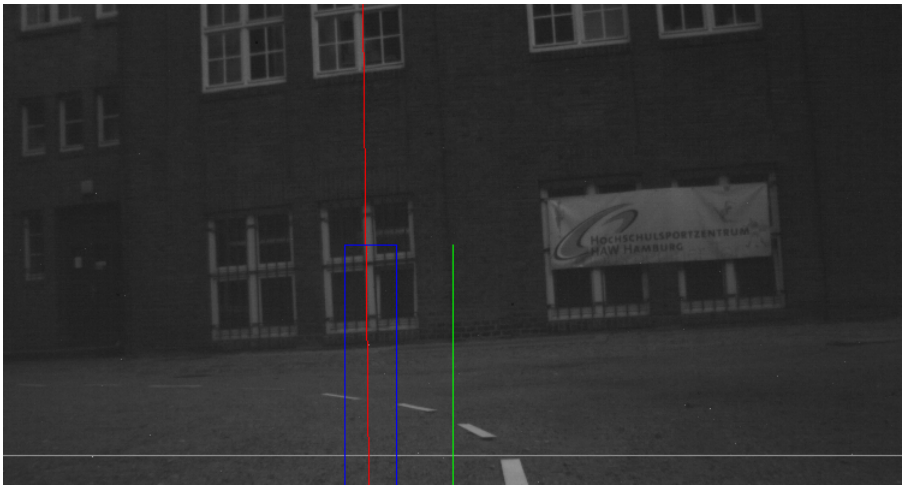

Abbildung 5.2: weißer Fensterrahmen wird als Fahrspurmarkierung missinterpretiert

Abbildung 5.3: Durchgang zwischen 2 Gebäuden wird als Fahrspurmarkierung missinterpretiert

Des Weiteren werden die wichtigsten Fahrspurmarkierungen, an denen der Fahrspurverlauf genauer bestimmt werden kann, übersehen. Die Fahrspurmarkierungen unmittelbar vor dem Fahrzeug können durch ihre höhere Auflösung besser erkannt werden als die weiter entfernten Fahrspurmarkierungen.

Zu diesem Zweck wurde die Fahrzeugkamera, wie in der Abbildung 5.4 zu sehen ist, um etwa 10° geneigt.

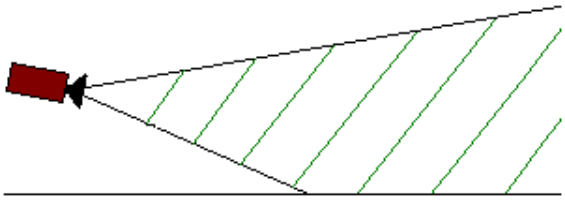

Abbildung 5.4: Ausrichtung der Kamera bei einem Neigungswinkel von etwa 10°

Die unter dieser Kameraeinstellung erzeugte Aufnahmen lieferten erwartungsgemäß einen höheren Erkennungsgrad. Die Abbildungen 5.5 sowie 5.6 zeigen erfolgreich erkannten Fahrspurmarkierungen bei einer Links- und Rechtskurve.

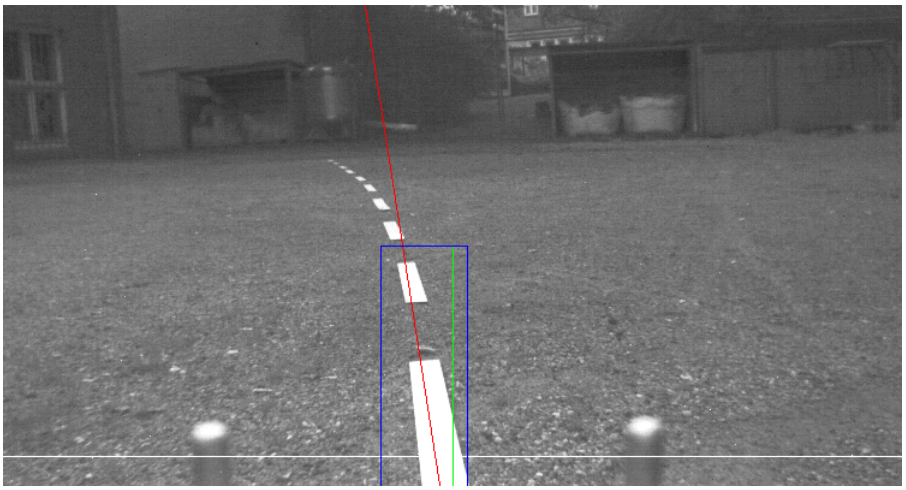

Abbildung 5.5: erfolgreiche Fahrspurerkennung bei einer Linkskurve

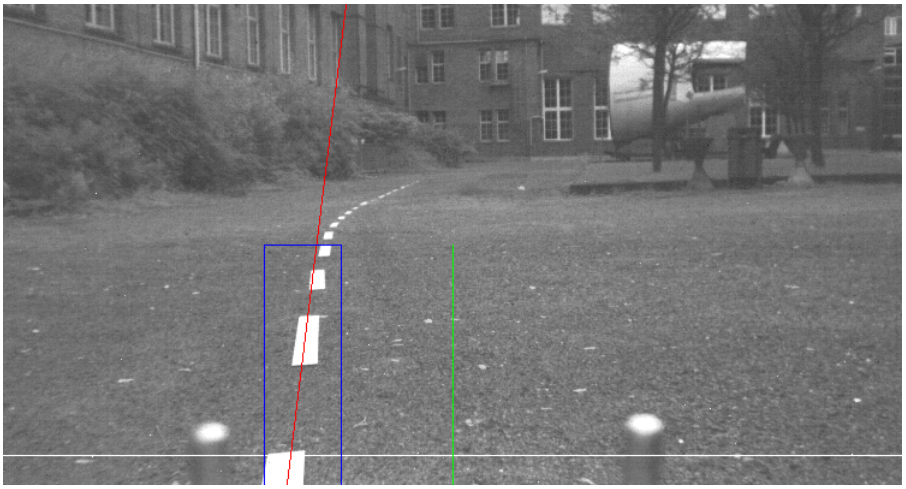

Abbildung 5.6: erfolgreiche Fahrspurerkennung bei einer Rechtskurve

Wie in den Abbildungen zu sehen ist, werden die unmittelbar vor dem Fahrzeug liegenden Fahrspurmarkierungen mit einer höheren Auflösung mitberücksichtigt. Die Neigung der Kamera sorgt ebenfalls dafür, dass die ROI die Fahrspurmarkierungen jederzeit am Boden sucht, wo diese sich erwartungsgemäß befinden.

5.2 Performance

Die mit der BLIZZARD-60-U2 Kamera von Photonfocus aufgezeichneten Videosequenzen haben eine Auflösung von 750x400 Pixel. Da die Fahrspurerkennung im Nahbereich arbeitet, ist eine so hohe Auflösung nicht notwendig. Diese wirkt sich sogar nachteilig auf das Gesamtverhalten aus. Bei einer hohen Auflösung ist jede kleine Veränderung sichtbar, worauf der Algorithmus umgehend reagiert. Des Weiteren müssen zwangsläufig mehr Pixel als notwendig untersucht werden, was zu mehr Rechenaufwand führt.

Um zu sehen, wie der Algorithmus sich bei einer kleineren Auflösung verhält, wurde die Auflösung der aufgezeichneten Videosequenzen um etwa 50% reduziert. Hierdurch konnte eine ca. 100% Performancesteigerung gemessen werden. Hierfür wurde der Algorithmus mit einer 27 Sekunden langen Videosequenz bei verschiedenen Auflösungen konfrontiert. Die Ergebnisse der beiden Testläufe können der Abbildung 5.7 entnommen werden.

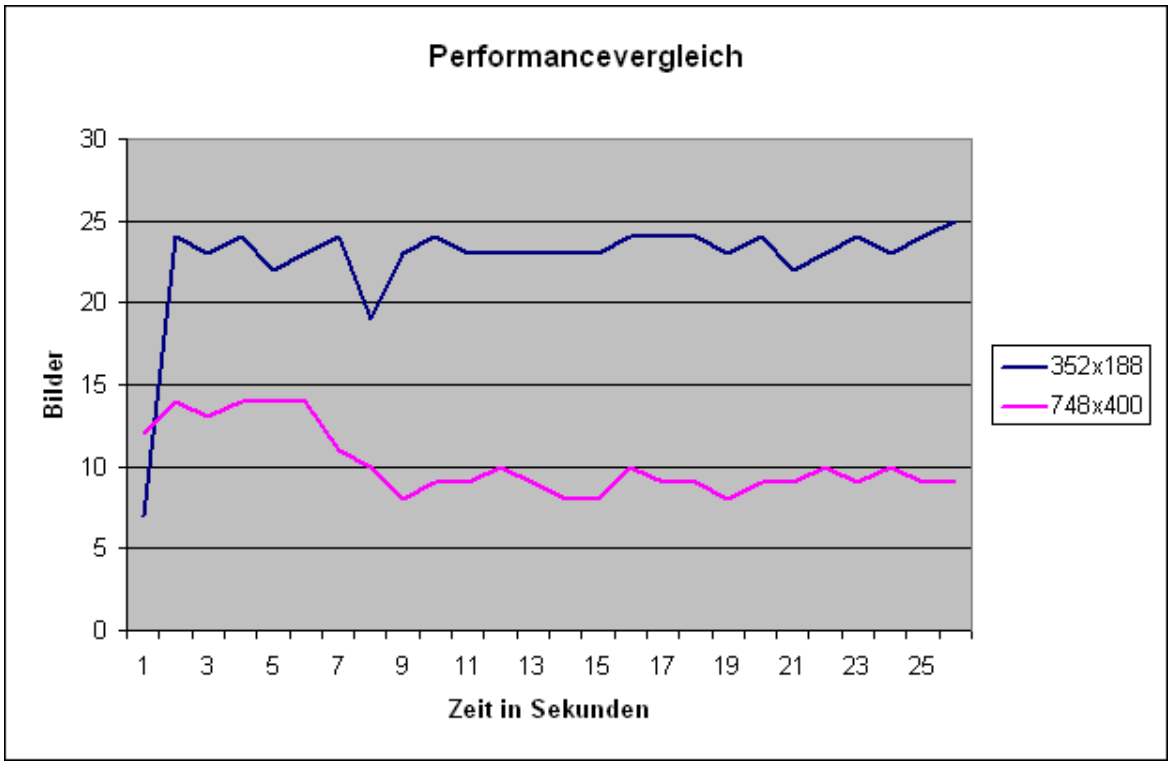

Abbildung 5.7: Performancevergleich

Damit solche Performance kontinuierlich zur Verfügung steht, wurde versucht die hochauflösende Quellbilder in Echtzeit zu verkleinern. Dieser Vorgang erwies sich als rechenaufwendig, da hierdurch ergänzter Algorithmus lediglich 7 Bilder/Sekunde bearbeiten konnte.

6 Zusammenfassung und Ausblick

6.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein videobasiertes Softwaremodul zur Fahrspurerkennung entwickelt. Das System ist in der Lage weiße, rechteckige Fahrspurmarkierungen zu entdecken und zu verfolgen. Hierfür verwendet der Algorithmus die bekannten Verfahren wie den Canny-Operator zur Kantenextraktion sowie die Hough-Transformation zur Extraktion der Geradenparameter. Durch die Einführung der ROI, die Reduzierung des Hough-Raums von 270° auf 30° sowie der Auflösung um ca. 50% wurde die Performance erheblich verbessert. Da die Reduzierung der Auflösung softwaretechnisch sehr viel Rechenzeit in Anspruch nimmt, sollte eine Kamera verwendet werden, welche bereits eine kleinere Auflösung liefert.

6.2 Ausblick

Da der Algorithmus relativ anfällig gegenüber anderen geraden Kanten ist, wäre es sinnvoll weitere Eigenschaften der Fahrspurmarkierungen für die Erkennung zu nutzen. Eine explizite Vermessung sowie Klassifizierung der Fahrspurmarkierungen würde die Robustheit des Algorithmus steigern. Es ist ebenfalls denkbar im Rahmen weiterer Arbeiten das System modular aufzubauen um eine Fahrbahn anhand anderer Kriterien zu erkennen.

Literaturverzeichnis

[Baessmann und W.Besslich 1989] BAESSMANN, Henning ; W.BESSLICH, Philipp: *Konturorientierte Verfahren in der digitalen Bildverarbeitung*. Springer, 1989. – ISBN 3-540-50772-8

[Broggi und Fascioli 2002] BROGGI, Alberto ; FASCIOLI, Alessandra: *Artificial Vision in Extreme Environments for Snowcat Tracks Detection*. 2002. – URL http://ieeexplore.ieee.org/iel5/6979/22198/01033759.pdf

[Cheng u. a. 2006] CHENG, Hsu-Yung ; JENG, Bor-Shenn ; TSENG, Pei-Ting ; FAN, Kuo-Chin: *Lane Detection With Moving Vehicles in the Traffic Scenes*. 2006. – URL http://ieeexplore.ieee.org/iel5/6979/4019425/04019429.pdf

[Fritzsch 1991] FRITZSCH, Klaus: *Maschinelles Sehen: Algorithmen, Systeme, Anwendungen*. Akademie Verlag, 1991. – ISBN 3-05-001785-6

[Hufnagl 2003] HUFNAGL, Peter: *Kantendetection*. 2003. – URL http://pathoweb.charite.de/fhtw/V6_Kantendetektion.pdf

[Jaehne u. a. 1995] JAEHNE, Bernd ; MASSEN, Robert ; NICKOLAY, Bertram ; SCHARFENBERG, Harald: *Technische Bildverarbeitung - Maschninelles Sehen*. Springer, 1995. – ISBN 3-540-58641-5

[Kuan u. a. 1988] KUAN, Darwin ; PHIPPS, Gary ; HSEUH, A.-Chuan: *Autonomous robotic vehicle road following*. 1988. – URL http://ieeexplore.ieee.org/iel1/34/351/00006773.pdf

[Levi u. a. 2005] LEVI, Paul ; SCHANZ, Michael ; LAFRENZ, Reinhard: Videobasierte Fahrspurerkennung zur Umfelderfassung bei Straßenfahrzeugen. In: *Autonome Mobile Systeme 2005* (2005), S. 173–178

[Li u. a. 2004] LI, Qing ; ZHENG, Nanning ; CHENG, Hong: *Springrobot: A Prototype Autonomous Vehicle and Its Algorithms for Lane Detection*. 2004. – URL http://ieeexplore.ieee.org/iel5/6979/29884/01364006.pdf

[Nischwitz u. a. 2004] NISCHWITZ, Alfred ; FISCHER, Max ; HABERÄCKER, Peter: *Masterkurs Computergrafik und Bildverarbeitung: alles für Studium und Praxis - Bildverarbeitungswerkzeuge, Beispiel-Software und interaktive Vorlesungen online verfügbar*. Vieweg, 2004. – ISBN 3-528-05874-9

[Wang u. a. 2005] WANG, Chun-Che ; HUANG, Shih-Shinh ; FU, Li-Chen: *Driver Assistance System for Lane Detection and Vehicle Recognition with Night Vision*. 2005. – URL http://ieeexplore.ieee.org/iel5/10375/32977/01545482.pdf

[Wang u. a. 1999] WANG, Yue ; TEOH, Eam K. ; SHEN, Dinggang: *Lane detection using B-snake*. 1999. – URL http://ieeexplore.ieee.org/iel5/6567/17541/00810313.pdf

[Yim und Oh 2003] YIM, Young U. ; OH, Se-Young: *Three-feature based automatic lane detection algorithm (TFALDA) for autonomous driving*. 2003. – URL http://ieeexplore.ieee.org/iel5/6979/28168/01260588.pdf

Glossar

ABS Antiblockersystem

ASR Antriebsschlupfregelung

ESP Elektronisches Stabilitätsprogramm

EUREKA European Research Coordination Agency

FOBIAS Forschungsverbund bioanaloge sensomotorische Assistenz

HAW-Hamburg Hochschule für Angewandte Wissenschaften Hamburg

LTI Lehrstuhl für Technische Informatik

PROMETHEUS Programme for a European Traffic with Highest Efficiency and Unprecedented Safety

ROI Region of Interest

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 24. September 2007	
Ort, Datum	Unterschrift