Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminararbeit

Eike Jenning

Fahrspurerkennung in Videoechtzeit mit System-on-Chip

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Eike Jenning

Fahrspurerkennung in Videoechtzeit mit System-on-Chip

Seminararbeit eingereicht im Rahmen der Ringvorlesung
im Studiengang Master Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Professor : Prof. Dr.Ing. Bernd Schwarz

Abgegeben am 28. Februar 2008

Thema der Seminararbeit

Fahrspurerkennung in Videoechtzeit mit System-on-Chip

Stichworte

Fahrspurerkennung, Bildverarbeitung, FAUST, Echtzeit, System-on-Chip, HW/SW-Codesign, Deserialisierung, Pipeline

Kurzzusammenfassung

Im Rahmen der Teilnahme der HAW Hamburg am CaroloCup 2008 an der TU Braunschweig wurde im Seminar eine Recherche zu Fahrspurerkennungsalgorithmen durchgeführt. Zusätzlich wurden allgemeine Entwurfsmethoden zur Umsetzung von Bildverarbeitung auf einem System-on-Chip betrachtet. Im nachfolgenden Bericht werden eine Auswahl von Verfahren aus der Recherche, sowie die Entwurfsmethoden vorgestellt.

Title of the paper

Lane Detection in Video-Realtime with System-on-Chip

Keywords

lane detection, image processing, FAUST, realtime, System-on-Chip, hw/sw-codesign, deserialisation, pipeline

Abstract

In context of the participation of the UAS Hamburg at the CaroloCup 2008 at the TU Braunschweig a research regarding lane detection algorithms was performed. Additionally general design methods for the realisation of image processing on a system-on-chip were considered. The following paper introduces a selection of the researched algorithms as well as the design methods.

Inhaltsverzeichnis

1. Einleitung

Das Projekt FAUST, kurz für Fahrerassistenz- und Autonome Systeme, aus dem Department für Informatik der Hochschule für Angewandte Wissenschaften Hamburg verfolgt die Analyse und Synthese von verteilten, eingebetteten Echtzeitsystemen. Zu diesem Zweck werden (Modell-)Fahrzeuge mit Sensorik, Aktorik und Steuereinheiten ausgestattet um beispielsweise Ausweichassistenten, Stabilisierungsverfahren oder zeitgesteuerte Systeme zu entwickeln.

Das nachfolgend beschriebene Seminarthema gliedert sich in den FAUST-Kontext ein. Zur Teilnahme der HAW Hamburg am CaroloCup[1], einem Wettbewerb für autonome Modellfahrzeuge am Institut für Regelungstechnik der Technischen Universtität Braunschweig[2], müssen Anforderungen an die Fahrzeugführung auf einer Straße erfüllt werden(vgl. (1) Abschnitt 3). Im Seminar wird aus diesem Grund eine Recherche zu Fahrspurerkennungsalgorithmen durchgeführt. Zusätzlich erfolgt eine Einarbeitung in Entwurfsmethoden zur Bildverarbeitung in dedizierten, rekonfigurierbaren Hardwarebausteinen. Das Hardware-Software-Codesign wird aus Gründen der Regelungsgeschwindigkeit, der Energiebilanz des Fahrzeugs und der aktuellen Industrierelevanz eingeführt.

Die Seminararbeit präsentiert drei Fahrspurerkennungsverfahren aus der Recherche. Kapitel 2.2 beschreibt ein Verfahren zur Vektorisierung von Fahrbahnen(vgl. (16). Ein zeilenbasiertes Verfahren wird in Kapitel 2.3 vorgestellt(vgl. (14)). Das Verfahren in Kapitel 2.4 basiert auf verformbaren Fahrbahnmodellen(vgl. (11)). Durch die Auswahl sollen verschiedenartige Lösungsmechanismen vorgestellt werden.

Ziel der Seminarbeit ist die Einarbeitung in verschiedene Fahrspurerkennungsalgorithmen und in die Entwurfsmethodik zur Umsetzung der Algorithmen auf einem System-on-Chip.

Die Arbeit ist wie folgt aufgebaut. Zunächst werden in Kapitel 2 Verfahren zur Fahrspurerkennung vorgestellt. In Kapitel 3 wird untersucht, mit welchen Entwurfsmethoden Bildverarbeitung in einem SoC realisiert werden kann. Eine Zusammenfassung in Kapitel 4 und ein Ausblick auf Projekt- und Masterarbeit in Kapitel 5 schließen diese Arbeit ab.

[1] Link: http://www.carolo-cup.de
[2] Link: http://www.ifr.ing.tu-bs.de

2. Verfahren zur Fahrspurerkennung

Nachfolgend werden drei Verfahren aus der Recherche vorgestellt. Die notwendigen mathematischen Operationen der Verfahrensschritte werden in Anhang A zusammengefasst. Eine Übersicht weiterer Algorithmen zur Fahrspurerkennung befindet sich in Anhang C.

2.1. Herausforderung bei der Fahrspurerkennung

An Verfahren zur kamerabasierten Fahrspurerkennung sind hohe Anforderungen gestellt. Neben der Verarbeitungsgeschwindigkeit ist vor allem die Zuverlässigkeit der Fahrbahnerkennung wesentliches Kriterium für den Einsatz eines Verfahrens. Die Herausforderung liegt in der korrekten Verarbeitung der Fahrbahninformation unter ungünstigen Bedingungen. Schattige Fahrbahnabschnitte(vgl. Abbildung 2.1a) und Feuchtigkeit(vgl. Abbildung 2.1c) bewirken eine Kontrastveränderung zwischen Fahrbahnmarkierung und Straße. Vor allem in entfernten Bildbereichen lassen sich Markierungen dadurch kaum noch extrahieren. Fehlende oder unterbrochene Markierungen(vgl. Abbildung 2.1b) erschweren die Erkennung einer zusammenhängenden Fahrbahnbegrenzung.

(a) Schattige Fahrbahn (b) Fehlende Markierungen (c) Feuchtigkeit auf der Fahrbahn

Abb. 2.1.: Herausforderungen bei der Erkennung von Fahrspuren

2.2. TFALDA-Algorithmus

Die Funktionsweise von TFALDA basiert auf der einmaligen Erkennung und der anschließenden Verfolgung von Fahrbahnmarkierungen. Zur Beschreibung von Fahrbahnmarkierungen werden 3-dimensionale Vektoren mit den Vektorelementen

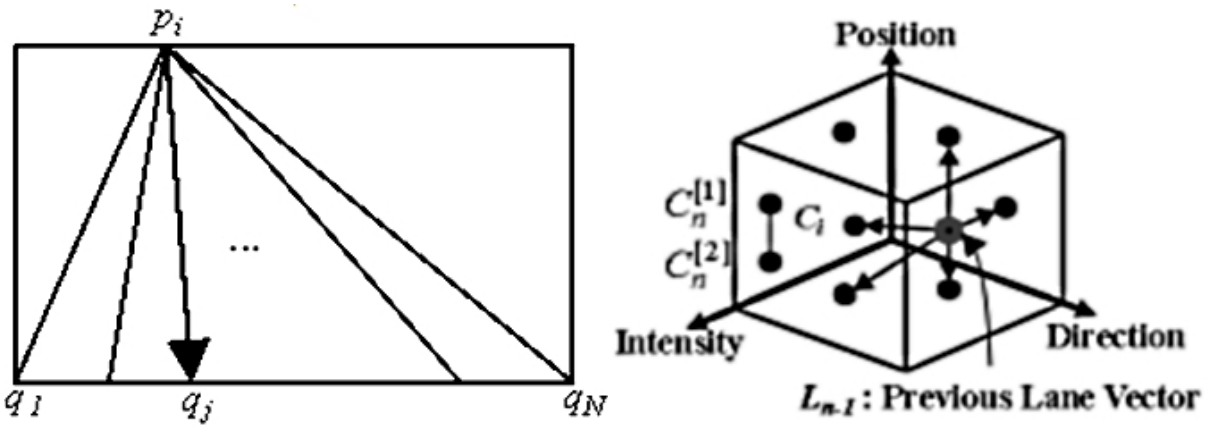
- P, der Startpunkt des Vektors im Bild
- D, die Richtung des Vektors
- I, die Grauwertintensität des Vektors

eingesetzt. Die Länge eines Vektors ist durch die Bildsegmentierung vorgegeben. Wie in Abbildung 2.2 skizziert, wird das Straßenbild in unbewegliche, horizontal ausgerichtete Bereiche voller Bildbreite geteilt, in denen wiederrum bewegliche Teilbereiche angeordnet sind. Jeder bewegliche Teilbereich(nachfolgend als ROI[1] bezeichnet) liegt über einer Fahrbahnmarkierung.

[1] Region Of Interest

Abb. 2.2.: Beispielhafte Bildsegmentierung durch TFALDA Algortihmus

In einer ROI werden mit dem Sobeloperator Kanten extrahiert. Anschließend werden die sogenannten Kandidatenvektoren bestimmt. Beginnend beim ROI-Ursprung(oben links) werden zu jedem Punkt P_i auf der Obergrenze die Vektoren zu allen Punkten Q_j auf der Untergrenze der ROI gebildet(vgl. Abbildung 2.3a.). Beide Indizes laufen von 0 bis N, wobei N die Breite der



(a) Analyse nach Kandidatenvektoren in einer ROI (b) Vektorraum zur Bestimmung der Nachbarschaft von Kandidaten

Abb. 2.3.: Vektorbestimmung durch TFALDA

ROI in Pixeln ist. Entlang eines Vektors zwischen P_i und Q_j wird die Grauwertintensität durch Addition der Pixelgrauwerte berechnet. Zu dem Punkt P_i wird der Vektor mit der höchsten Grauwertintensität gespeichert. Der Winkel dieses Kandidaten bestimmt sich aus dem zu P_i und der Intensität gehörenden Q_j. Zu einer ROI sind nach der Analyse N Kandidatenvektoren gespeichert.

Die Verfolgung der Fahrbahnmarkierung wird durch einen gewichteten Vergleich zwischen zeitlich aufeinanderfolgenden Vektoren erzielt. Der Vergleich findet zwischen den N Kandidaten einer ROI des aktuellen Bildes und dem gewählten Vektor derselben ROI des vorherigen Bildes statt. Nachvollziehen lässt sich dieser Schritt durch die Darstellung in Abbildung 2.3b. Aus den gewichteten Vektorelementen wird ein Vektrorraum aufgebaut, in dem der am dichtesten am Vorvektor liegende Kandidat gesucht wird. Die Berechnung wird mit Formel 2.1 durchgeführt:

$$\lambda_{CV_i} = K_P * |P(CV_i) - P(PV)| + K_D * |D(CV_i) - D(PV)| + K_I * (I(PV) - I(CV_i)) \tag{2.1}$$

wobei CV_i die aktuellen Kandidatenvektoren und PV den Vorvektor einer ROI symbolisiert. Die Gewichtung der Elemente wird in (16) vorgegeben. Sie entstand durch die Anwendung eines evolutionären Algorithmus auf einen Trainingsdatensatz. Die Grauwertintensität wird ohne Betragsbildung ausgewertet, damit Vektoren mit hoher Intensität priorisiert werden. Ausgewählt wird der Kandidatenvektor mit dem niedrigsten λ.

Sobald für jede ROI der neue Vektor feststeht, wird eine Plausibilitätsprüfung durchgeführt. Unter anderem wird betrachtet, wie sich die, zwischen den Vektoren aufgespannte, Fahrbahn über die Zeit in ihrer Breite verändert. Bei einer sprunghaften Veränderung werden die neuen Daten gezielt verworfen und stattdessen die Information aus dem vorherigen Bild benutzt. Beide Aspekte, Vektornähe und Plausibilität, bestimmen in Kombination die Robustheit des TFALDA Verfahrens. Die Geschwindigkeit wird mit 30 Bildern pro Sekunde auf einem PII Prozessor mit 266Mhz angegeben.

2.3. Peak-finding-Algorithmus

Der Peak-finding-Algorithmus basiert auf der Extraktion von Maxima(Peaks) aus dem Grauwerthistogramm einer Bildzeile. Grundlage dafür sind folgende Annahmen über eine Fahrspurmarkierung:

- Helligkeit: Markierungen sind generell heller als die Fahrbahn
- Breite: Markierungen sind normalerweise hell und schmal
- Nachbarschaft: Markierungen mit zwei Enden befinden sich in der Nähe ihrer Nachbarn

Aus den Annahmen wird die Spezifikation eines Maximums abgeleitet, indem Grenzwerte für die maximale Breite und die minimale Höhe des Maximums festgelegt werden(vgl. Abbildung 2.4). Die Flanken eines Maximums verlaufen streng monoton und müssen jeweils eine bestimmte Höhe aufweisen. Die Berechnung der Grenzwerte wird in (14) nicht beschrieben. Damit

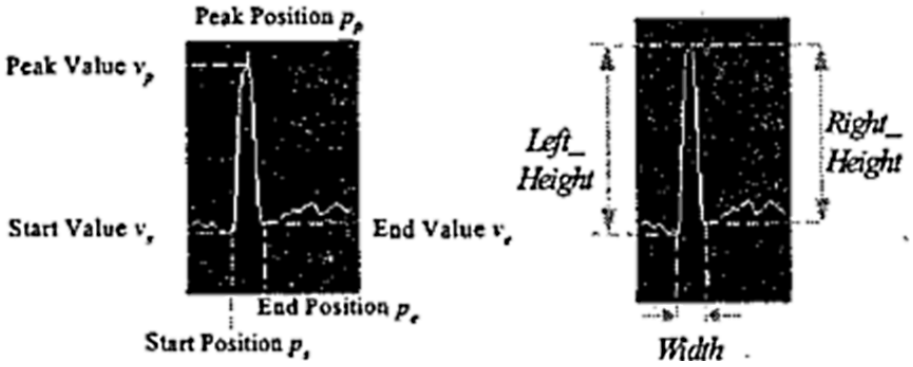


Abb. 2.4.: Spezifikation eines Maximums im Grauwerthistogramm einer Bildzeile

Bildstörungen keine Auswirkung auf den Flankenverlauf gültiger Maxima ausüben, wird eine Gauss-Filterung zur Glättung eingesetzt.

Die Auswertung durch zeilenweise Abtastung des Bildes nach Maxima extrahiert vertikale Kanten. Sobald ein Maximum die Spezifikation erfüllt, werden dessen Bildkoordinaten in ein Zielbild übertragen. Auf diese Weise entsteht das sogenannte Peak-Point-Image(vgl. Abbildung 2.5). Anschließend erfolgt eine Gruppierung von Maxima zu Geraden. Mit Hilfe einer Methode der kleinsten Fehlerquadrate wird die maximal zulässige Entfernung eines Punktes zur Geraden festgelegt. Maxima ohne Geradenzugehörigkeit werden aus dem Bild entfernt. Gemäß der anfangs genannten Nachbarschaft werden die Geraden durch eine Verlängerung der Endpunkte zusammengeführt(vgl. Abbildung 2.6). Des Endergebnis des Peak-finding-Algorithmus ist ebenfalls in Abbildung 2.6 dargestellt. Die Geschwindigkeit des Verfahrens liegt bei etwa 15

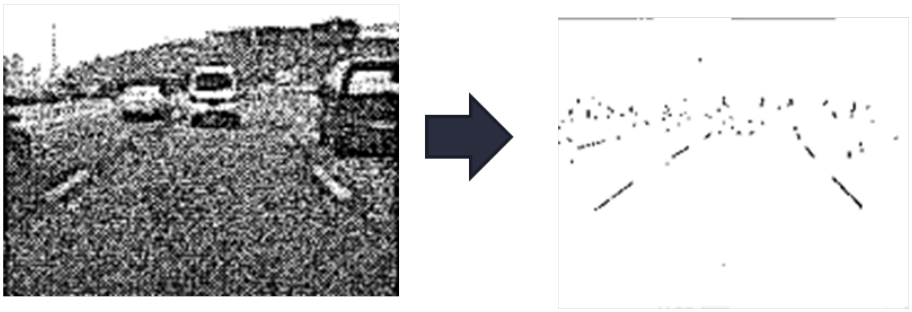


Abb. 2.5.: Resultierendes Peak-Point-Image durch Maxima-Übertragung

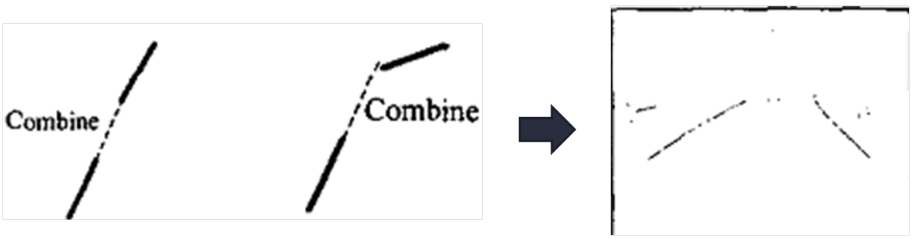


Abb. 2.6.: Kombination gruppierter Maxima und Endergebnis

Millisekunden pro Bild auf einem PC mit 1,8GHz(vgl. (14), Tabelle 5.3). Über die Robustheit des Verfahrens wird keine Auskunft gegeben.

2.4. LOIS-Deformable-Template-Algorithmus

Im Gegensatz zu den vorangehenden Verfahren analysiert der LOIS-Deformable-Template-Algorithmus[2] die Passgenauigkeit eines mathematischen Fahrbahnmodells auf das aktuelle Straßenbild(vgl. Abbildung 2.8).

Der Aufbau von LOIS ist in drei Kernkomponenten strukturiert:

- Das mathematische Fahrbahnmodell
- Eine Wahrscheinlichkeitsfunktion zur Bestimmung der Passgenauigkeit des Modells
- Eine Optimierungsfunktion zur Bestimmung des Maximums der Wahrscheinlichkeitsfunktion

Als mathematisches Modell wird eine Beschreibung von Fahrbahnmarkierungen durch Kurven eingesetzt(vgl. (11) Abschnitt 3.1). Dazu werden drei Parameter definiert(vereinfacht dargestellt):

- Der Bogen der Kurve
- Die Orientierung der Kurve
- Der Offset der Kurve

Da sich der Bogen und die Orientierung der Kurven für die linke und die rechte Fahrbahnmarkierung nur für sehr kleine Kurvenradien stärker unterscheiden, werden die Parameter für beide Kurven gleichgesetzt. Der resultiernde Parametersatz für beide Fahrbahnmarkierungen besteht aus dem Bogen und der Orientierung beider Markierungen, sowie dem Offset der linken Markierung und dem Offset der rechten Markierung.

Die Wahrscheinlichkeitsfunktion bestimmt für einen Parametersatz des mathematischen Modells die Passgenauigkeit in Form eines ganzzahligen Wertes. Je höher der Wert ist, umso besser

[2] LOIS - Likelihood of image shape.

passt das Modell mit dem eingestellten Parametersatz auf das aktuelle Bild. Der Wert berechnet sich als die Summe der Passwahrscheinlichkeiten aller Pixel im Bild. Zur Untersuchung der Pixel werden mit dem Prewitt-Operator einmalig die Gradienteninformationen des aktuellen Bilds erzeugt[3]. Für die Berechnung der Passwahrscheinlichkeit erfolgt eine Gewichtung der Nähe eines Pixels zur Modellfahrbahnkante durch den Gradientenbetrag dieses Pixels. Die Passwahrscheinlichkeit steigt zusätzlich, wenn die Gradientenrichtung des Pixels nahe dem Lot der Tangente der angesetzten Fahrbahnkante liegt(vgl. Abbildung 2.7). Die Passwahrscheinlichkeit ist durch die genannte Berechnung bei einer Überlagerung von Modellfahrbahnkante und Bildfahrbahnkante am größten.

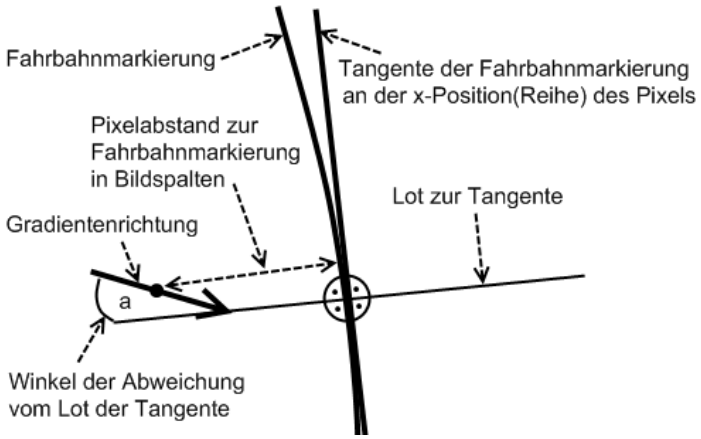


Abb. 2.7.: Berechnung der Passwahrscheinlichkeit eines Pixels anhand seines Gradienten

Ziel der Optimierungsfunktion ist die Maximierung des Wertes der Wahrscheinlichkeitsfunktion. In einem iterativen Verfahren wird der Parametersatz des mathematischen Modells mit Zufallswerten aus einer definierten Nachbarschaft der Parametereinstellungen belegt und mit der Wahrscheinlichkeitsfunktion analysiert. Neue Parametereinstellungen werden im Falle eines höheren Wahrscheinlichkeitswertes übernommen. Abbildung 2.8 stellt exemplarisch das Ergebnis einiger Iterationsschritte mit den zugehörigen Wahrscheinlichkeitswerten dar. Die Qualität

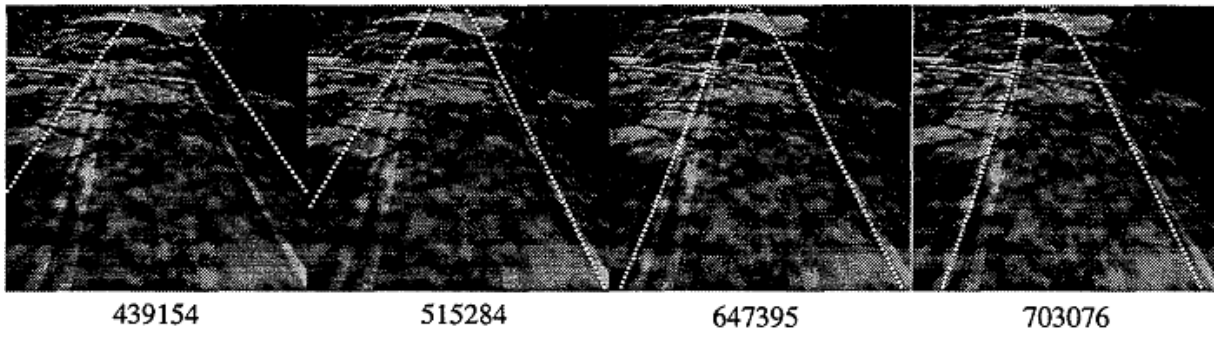


Abb. 2.8.: Ergebnis der Wahrscheinlichkeitsfunktion mit unterschiedlichen Modellparametern

der Optimierungsfunktion ist eng an dessen Parameter gebunden. Je nach Anzahl der Iterationen oder der Saat für den Zufallszahlengenerator besteht die Möglichkeit, dass die Optimierungsfunktion nicht das Maximum der Wahrscheinlichkeitsfunktion erreicht. In dem Fall bildet das Modell die tatsächliche Fahrbahn ungenau ab.

[3] Anhang B liefert eine grafische Veranschaulichung von Gradienteninformationen

3. Entwurfsmethoden zur Bildverarbeitung in Hardware

Der schematische Aufbau des geplanten System-on-Chip wird in Anhang D dargestellt. Die in diesem Kapitel besprochene Entwurfsmethodik steckt in dem ersten Modul der „Image Processing Pipeline“.

Wesentliches Merkmal der Bildverarbeitung in Hardware ist die Anwendung der Algorithmen auf den Datenstrom. Motiviert wird diese Methodik zum einen durch den begrenzten Speicherplatz auf dem Hardwarebaustein. Bereits ein Bild mit 1280x1024 Pixeln und 8 Bit Farbtiefe benötigt etwa 1,3MB Speicher. Zum anderen verlangsamen sich durch den Zugriff auf externe Speicherbausteine die Algorithmen. Der Vorteil der parallelen Datenverarbeitung wird durch den begrenzten Adress- und Datenbus zum Speicherbaustein verringert.

Die Anwendung der Datenstrom-Methodik erfolgt am Beispiel der Vorverarbeitungsschritte aus den, in Kapitel 2, vorgestellten Fahrspurerkennungsalgorithmen. Das TFALDA-Verfahren und das LOIS-Deformable-Template-Verfahren setzten den Sobel- bzw. den Prewitt-Operator zur Vorverarbeitung ein. Beide Operatoren basieren auf der Anwendung von 3x3 Faltungsmasken(vgl. (3) Kapitel 3.2.6 und (4) Kapitel 8.7.3). Im Peak-finding-Algorithmus kommt ein Gaussfilter zur Bildglättung zum Einsatz. Eine Vereinfachung ist der Binomialfilter, der sich bei großen Faltungsmasken der Gaussfilterung annähert(vgl. (8) Kapitel 11.4). Die genannten Vorverarbeitungsstufen lassen sich somit durch die Anwendung von Faltungsmasken auf den Datenstrom realisieren.

Die Deserialisierung des Datenstroms dient zur Aufbereitung der Daten für die Verarbeitung. Die Anwendung einer 3x3 Faltungsmaske erfordert beispielsweise den gleichzeitigen Zugriff auf 9 zusammenliegende Bildpunkte. Durch das zeilenweise Auslesen der Daten aus dem Bildsensor ergibt sich die Notwendigkeit einer Zwischenspeicherung von zwei Bildzeilen und drei Pixeln der dritten Bildzeile in Schieberegistern(vgl. Anhang E Abbildungen E.1 und E.2). Sobald die Maske erstmalig vollständig über Bildpunkten liegt, wird mit jedem neu ausgelesenen Pixel die Maske verschoben und sofort der Filterwert berechnet.

Die Berechnung von 3x3 Faltungsmasken erfolgt parallel in einer getakteten Pipelinestruktur(vgl. Abbildung 3.1). Ziel ist es mit jedem Takt alle Ergebnisse einer Verarbeitungsstufe an die nächste zu übergeben. Um eine hohe Verarbeitungsgeschwindigkeit zu erreichen, werden pro Pipelinestufe zur Bildung der Zwischenergebnisse die maximale Anzahl an Additionen parallel durchgeführt. Für eine 3x3 Faltungsmaske sind das in der ersten Stufe 4 Additionen. Nicht verrechnete Werte einer Stufe müssen ohne Änderung in die nachfolgende Stufe überführt werden, sonst würden sie mit dem nächsten Takt durch nachrückende Werte überschrieben werden. Die vor jeder Addition durchgeführte Mutliplikation wird für ganzzahlige Faltungswerte durch Schiebeoperationen implementiert. Die Berechnung reeller Zahlen(z.B. Gauss-Filter) ist ebenfalls möglich, jedoch langsamer.

Jeder Verarbeitungsschritt einer Fahrspurerkennung benötigt eine geeignete Deserialisierung und gegebenenfalls eine Pipeline zur parallelen Berechnung. Der Datenstrom aus der 3x3

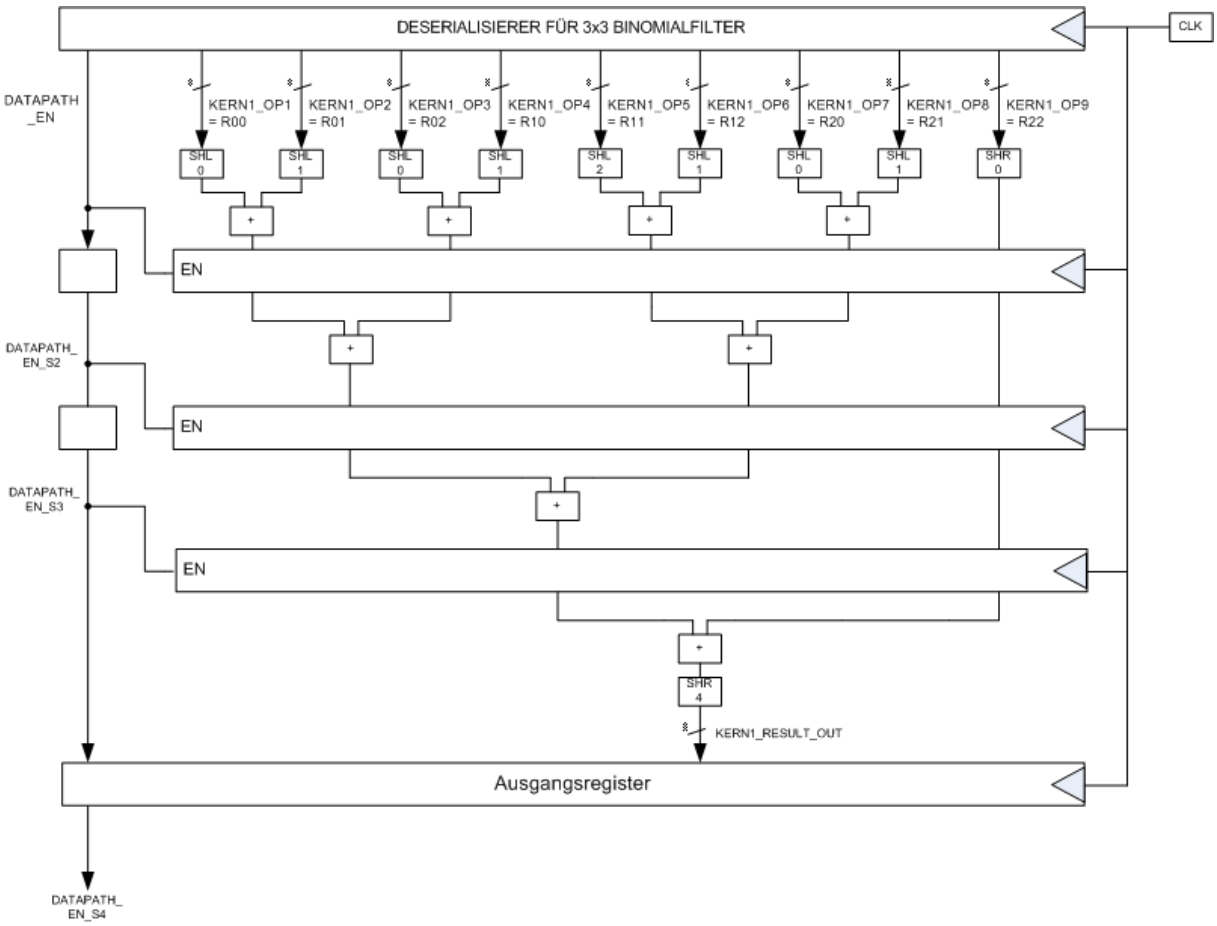


Abb. 3.1.: Pipeline-Struktur zur parallelen Berechnung eines 3x3 Binomalfilters

Binomialfilter-Pipeline(vgl. Abbildung 3.1) wird in der nächsten Verarbeitungsstufe zur Suche nach Maxima gemäß Peak-finding-Algorithmus deserialisiert(vgl. Abbildung 3.2). Die gefilterten Werte werden in ein Schieberegister zur parallelen Überprüfung der Monotonie der Werte gespeichert. Die Algorithmusstufe würde für das Beispiel in Abbildung 3.2 Informationen über Pixel 3 an die nächste Verarbeitungsstufe weitergeben.

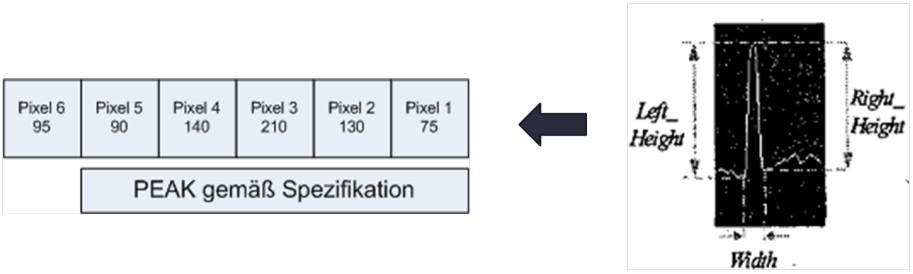


Abb. 3.2.: Deserialisierung für die Maxima-Analyse nach Peak-finding-Algorithmus

Laufzeitverzögerungen durch Deserialisierung und Pipelinestruktur entstehen für jedes auszuwertende Bild. Zunächst muss die Deserialisierungsstufe die Schieberegister auffüllen, bevor die erste Berechnung möglich ist. Zu dieser Verzögerung addiert sich die Durchlaufzeit der ersten Berechnungswerte durch die Pipelinestruktur. Sobald die Strukturen gefüllt sind, wird mit jedem Takt eine Berechnung durchgeführt.

Die Entwurfsmethoden sind aus der Recherche auf das gewählte Beispiel des 3x3 Binomialfilters übertragen wurden(vgl. (9), (10), (12)).

4. Zusammenfassung der Recherchen

Das Seminar diente zur Recherche von kamerabasierten Fahrspurerkennungsalgorithmen im Rahmen der Teilnahme der HAW Hamburg am CaroloCup 2008. Eine Einarbeitung in die Entwurfsmethoden zur Realisierung von Bildverarbeitung in Hardware dient zusätzlich als Grundlage für die spätere Masterarbeit. Eine Übersicht weiterer Fahrspurerkennungsverfahren liefert Anhang C.

Das TFALDA-Verfahren basiert auf einer Bildsegmentierung mit anschließender Vektorisierung der Fahrbahnmarkierungen(vgl. Kapitel 2.2). Die Robustheit des Verfahrens wird durch die Anwendung eines gewichteten Vergleichs von Vektoren über die Zeit gesteigert. Eine Verarbeitungsstufe zur Prüfung der Plausibilität der Vektoren, beispielsweise in Bezug auf die resultierende Fahrbahnbreite, steigert weiterhin die Robustheit durch gezielten Vektoraustausch. Durch die Bildsegmentierung werden Filteroperationen lediglich auf Bildausschnitte angewendet, sodass eine hohe Verarbeitungsgeschwindigkeit von 33 Millisekunden pro Bild auf einem PII Prozessor mit 266MHz erreicht wird.

Das Peak-finding-Verfahren analysiert das Bild zeilenweise zur Extraktion von Maxima(vgl. Kapitel 2.3). Ein Maximum muss in seiner Breite und Höhe eine Fahrbahnmarkierung definieren. Zur Erzeugung streng monoton verlaufender Flanken wird das Bild mit einem Gauss-Filter geglättet. Die Koordinaten eines Maximums werden in das sogenannte Peak-Point-Image übertragen. Dort werden die relevanten Punkte mit einer Methode der kleinsten Fehlerquadrate zu Geraden gruppiert und diese miteinander verbunden. Zur Steigerung der Robustheit werden keine Verfahren genannt. Die Verarbeitungsgeschwindigkeit liegt bei 15 Millisekunden pro Bild auf einem P4 Prozessor mit 1,8GHz.

Das LOIS-Deformable-Template-Verfahren untersucht die Passgenauigkeit eines mathematischen Fahrbahnmodells auf das aktuelle Straßenbild. Das Modell besitzt für zwei Fahrbahnmarkierungen vier unterschiedliche Parameter. Eine Wahrscheinlichkeitsfunktion berechnet für eine Einstellung des Parametersatzes mithilfe von Gradienteninformationen einen Kennwert der Wahrscheinlichkeit. Je näher und lotrechter ein Pixel mit hohem Gradientenbetrag zur angesetzten Fahrbahnmarkierung liegt, desto höher wird der Kennwert. Eine Optimierungsfunktion bestimmt iterativ die optimale Parametereinstellung durch die Suche nach dem Maximum der Wahrscheinlichkeitsfunktion. Die Parametereinstellungen werden zufällig aus einer definierten Nachbarschaft gewählt. Die Robustheit des Verfahrens hängt vom Erfolg der Optimierungsfunktion ab. Die Geschwindigkeit des Verfahrens wird nicht angegeben, ist aber aufgrund der Verarbeitung der gesamten Bildinformation als hoch einzuschätzen.

Die vorgestellten Entwurfsmethoden für die Bildverarbeitung in Hardware basieren auf den Prinzipien Deserialisierung und Parallelisierung. Es wurde die Umsetzung von 3x3 Faltungsmasken zur Implementierung der Vorverarbeitungsschritte der genannten Verfahren in Hardware aufgezeigt. Der Datenstrom aus dem Bildsensor wird in Schieberegistern zwischengespeichert und anschließend in einer Pipelinestruktur parallel verarbeitet. Durch die Zwischenpeicherung der Daten und die mehrstufige, parallele Berechnung enstehen Verzögerungszeiten bei der Verarbeitung jedes Bildes. Diese sind zu berücksichtigen und zu minimieren.

5. Ausblick auf Projekt und Masterarbeit

Das in diesem Bericht besprochene Thema der Fahrspurerkennung mit zusätzlichem Blick auf eine Realisierung in rekonfigurierbarer Hardware wird in das Projekt und die Masterarbeit übergehen. Das Seminar dient in diesem Zusammenhang zur Einarbeitung in Verfahren und Entwurfsmethoden.

Für die präsentierten Fahrspurerkennungsalgorithmen besteht zu diesem Zeitpunkt ein grundsätzliches Verständis. Eine Implementierung der Verfahren in Software soll zunächst die Besonderheiten und Eigenschaften im Detail klären. Anhand der Erfahrungswerte wird anschließend eine Analyse für Vereinfachungs- bzw. Verbesserungsmöglichkeiten durchgeführt und ein Verfahren für die Umsetzung auf einem System-on-Chip ausgewählt. Die Strukturierung des gewählten Verfahrens in Hardware- und Softwaremodule schließt die Vorbereitung der Implementierung in rekonfigurierbarer Hardware ab. Die Entwurfsmethoden für Bildverarbeitung mit Hardwaremodulen müssen in einer Analyse auf die gewählten Verfahrensschritte angewendet werden.

Im Projekt wird das TFALDA-Verfahren(vgl. Kapitel 2.2) in Software zur Teilnahme am CaroloCup 2008 implementiert. Die Wahl wurde aufgrund der angegebenen Verarbeitungsgeschwindigkeit getroffen(vgl. (16) Abschnitt V.).

Auf das Projekt folgend, werden in der Masterarbeit die Verfahren Peak-finding(vgl. Kapitel 2.3) und LOIS-Deformable-Template(vgl. Kapitel 2.4) in Software implementiert. Anschließend wird mit dem oben beschriebenen Analyseteil fortgefahren.

Literaturverzeichnis

[1] *Carolo-Cup Regelwerk 2008*, Januar 2008. http://www.carolo-cup.de/uploads/media/20080130_Carolo-Cup_Regelwerk.pdf.

[2] AKIHIRO WATANABE und MAKOTO NISHIDA: *Lane detection for a Steering Assistance System*, 2005. http://ieeexplore.ieee.org/iel5/10053/32246/01505095.pdf?tp=&arnumber=1505095&isnumber=32246.

[3] AL BOVIK: *Handbook of Image and Video Processing, Second Edition*. Elsevier Academic Press, 2005. ISBN: 0-12-119792-1.

[4] ALEXANDER HORNBERG: *Handbook of Machine Vision*. Wiley-VCH, 2006. ISBN: 3-527-40584-4.

[5] ANDREAS MEISEL: *Skript zum Informatik-Wahlpflichtmodul Robot Vision*. Hochschule für Angewandte Wissenschaften Hamburg, 2005.

[6] ARATA TAKAHASHI, YOSHIKI NINOMIYA, MITSUHIKO OHTA und KOICHI TANGE: *A Robust Lane Detection using Real-time Voting Processor*, 1999. http://ieeexplore.ieee.org/iel5/6644/17743/00821123.pdf?tp=&arnumber=821123&isnumber=17743.

[7] AXEL GERN, RAINER MOEBUS und UWE FRANKE: *Vision-based Lane Recognition under Adverse Weather Conditions Using Optical Flow*, 2002. http://ieeexplore.ieee.org/iel5/8453/26633/01188025.pdf?tp=&arnumber=1188025&isnumber=26633.

[8] BERND JÄHNE: *Digitale Bildverarbeitung, 6., überarbeitete und erweiterte Auflage*. Springer, 2005. ISBN: 3-540-24999-0.

[9] F. PAULO: *Bildvorverarbeitungsmodule für eine FPGA-basierte CCD-Kamera mit Optimierung einer 700 MBit Schnittstelle*. Bachelorarbeit, Hochschule für Angewandte Wissenschaften Hamburg, 2007.

[10] IMMO BIRNBAUM: *Erweiterung eines FPGA-basierten CCD-Kamera-Prototypen mit Steuerungs- und Bildverarbeitungsmodulen*. Bachelorarbeit, Hochschule für Angewandte Wissenschaften Hamburg, 2007.

[11] KARL KLUGE und SRIDHAR LAKSHMANAN: *A Deformable-Template Approach to Lane Detection*, 1995. http://ieeexplore.ieee.org/iel3/4019/11537/00528257.pdf?tp=&arnumber=528257&isnumber=11537.

[12] LE ZHANG: *FPGA based CCD Camera with Bayer Filter Interpolation*. Masterarbeit, Hochschule für Angewandte Wissenschaften Hamburg, 2006.

[13] PEI-YUNG HSIAO, HSIEN-CHEIN CHENG, CHUN-WEI YEH, SHIH-SHINH HUANG und LI-CHEN FU: *Automobile Lane Detection System-on-Chip Integrated with Mixed Signal Mode CMOS Image Sensor*, 2005. http://ieeexplore.ieee.org/iel5/10028/32172/01502395.pdf?tp=&arnumber=1502395&isnumber=32172.

[14] SHIH-SHINH HUANG, CHUNG-JEN CHEN, PEI-YUNG HSIAO und LI-CHEN FU (Herausgeber): *On-Board Vision System for Lane Recognition and Front-Vehicle Detection to Enhance Driver's Awareness*, IEEE International Conference on Robotics and Automation, 2004. http://ieeexplore.ieee.org/iel5/9126/29022/01307429.pdf?tp=&arnumber=1307429&isnumber=29022.

[15] SUNGHOON KIM, JEONGHO PARK, SEONG IK CHO, SOONYOUNG PARK und K.H. CHOI: *A Framework for the Development of Robust Lane Recognition Systems*, 2007. http://ieeexplore.ieee.org/iel5/4357618/4357619/04357760.pdf?tp=&arnumber=4357760&isnumber=4357619.

[16] YOUNG UK YIM und SE-YOUNG OH: *Three-Feature Based Automatic Lane Detection Algorithm (TFALDA) for Autonomous Driving*. IEEE Transactions On Intelligent Transportation Systems, 4(4):219–225, Dezember 2003. http://ieeexplore.ieee.org/iel5/6979/28168/01260588.pdf?tp=&arnumber=1260588&isnumber=28168.

Abbildungsverzeichnis

A. Mathematische Operationen der vorgestellten Verfahren

Die nachfolgenden Tabellen dienen als erster Eindruck für eine Implementierbarkeit der Verfahren in Hardware. Eine vertiefende Analyse ist noch zu leisten und erfolgt im Rahmen der Softwareimplementierung der Verfahren.

A.1. TFALDA

Abbildung A.1 nennt die mathematischen Operationen der Verfahrensschritte des TFALDA-Algorithmus. Die Winkelberechnung mit Arkuskosinus lässt sich durch eine Subtraktion der

Verfahrensschritt	Mathematische Operationen
Resampling	z.B. bilineare Interpolation: Addition und Multiplikation
Sobeloperator	3x3 Faltung mit Addition und Multiplikation
Vektorkandidatensuche	Intensität: Addition und Vergleich Winkel: Arkuskosinus
Nachbarschaftsbestimmung	Addition und Multiplikation
Lane Inference System	Nicht dokumentiert und ungeklärt. Vermutlich Addition und Vergleich.

Abb. A.1.: Mathematische Operationen der Verfahrenschritte des TFALDA-Algorithmus

Positionen der Vektorenden P_i und Q_j substituieren(vgl. Abbildung A.2). Dieser Schritt ist möglich, da die Vektorenden immer den gleichen vertikalen Abstand zueinander haben.

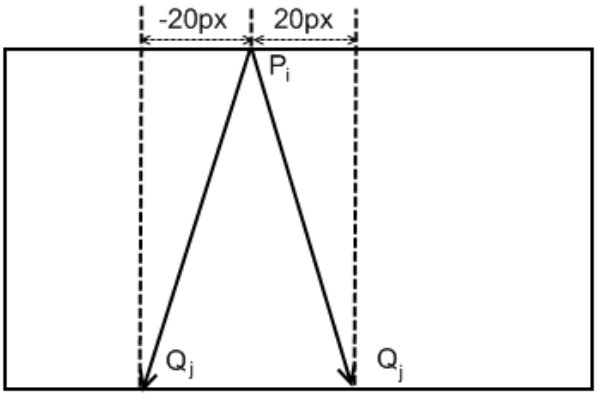


Abb. A.2.: Substitution des Winkels durch den horizontalen Pixelabstand der Vektorendpunkte

A.2. Peak-finding

Abbildung A.3 nennt die mathematischen Operationen der Verfahrensschritte des Peak-finding-Algorithmus.

Verfahrensschritt	Mathematische Operationen
Bildglättung	Gauss-Filter: Multiplikation, Wurzel- und Exponentialrechnung Binomialfilter: 3x3 Faltung mit Addition und Multiplikation
Maxima-Suche	Vergleich
Maxima-Gruppierung	Methode der kleinsten Fehlerquadrate: Multiplikation und Addition
Segmentverknüpfung	Addition

Abb. A.3.: Mathematische Operationen der Verfahrenschritte des Peak-finding-Algorithmus

A.3. LOIS-Deformable-Template

Abbildung A.4 nennt die mathematischen Operationen der Verfahrensschritte des LOIS-Deformable-Template-Algorithmus.

Verfahrensschritt	Mathematische Operationen
Prewittoperator	3x3 Faltungs mit Addition und Multiplikation
Gradientenbetrag berechnen	Addition, Multiplikation, Wurzel
Gradientenrichtung berechnen	Arkustangens
Wahrscheinlichkeitsfunktion	Arkustangens, Kosinus, Multiplikation, Division, Addition
Optimierungsfunktion	Exponentialrechnung, Multiplikation, Division, Addition

Abb. A.4.: Mathematische Operationen der Verfahrenschritte des LOIS-Algorithmus

B. Veranschaulichung von Gradienteninformationen

Die nachfolgenden Grafiken dienen zur Veranschaulichung von Gradienten. Abbildung B.1 stellt die Gradientenrichtung mit blauen Pfeilen und den höchsten Gradientenbetrag in schwarz dar. In erster Linie dient die Darstellung für das Verständnis der Gradientenrichtung zu einer hellen, vertikalen Linie im Bild. Abbildung B.2 zeigt den Betrag der Gradienten in der Umgebung

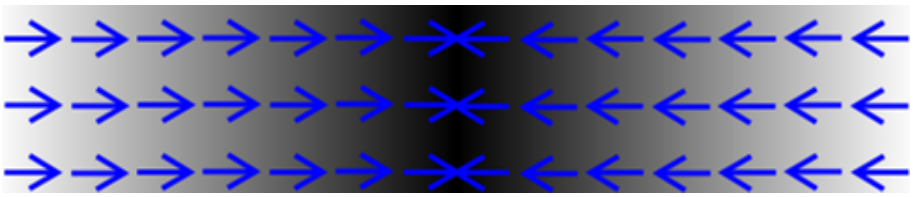


Abb. B.1.: Betrag und Richtung von Gradienten in der Umgebung einer vertikalen Gerade

eines hellen Bildpunkts auf der linken Seite. Die rechte Seite stellt die Gradientenrichtungen in der Draufsicht auf den Bildpunkt dar.

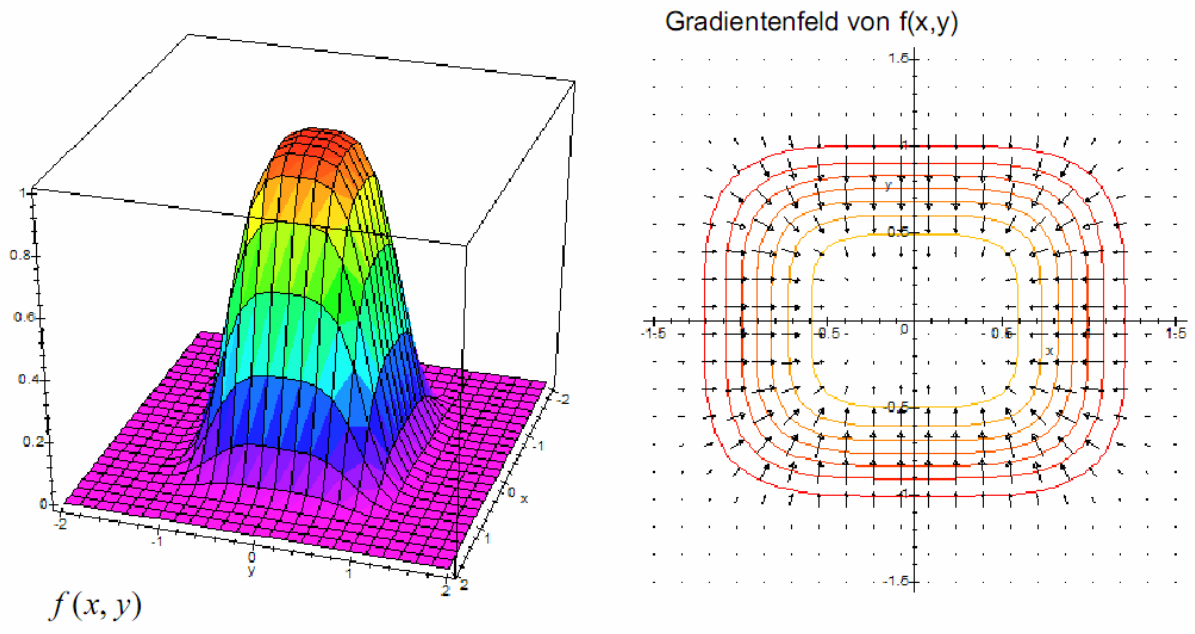


Abb. B.2.: Betrag und Richtung von Gradienten in der Umgebung eines hellen Punktes(vgl. (5))

C. Übersicht weiterer Fahrspurerkennungsverfahren

TFALDA(vgl. Kapitel 2.2) wird zur Implementierung eines Frameworks für robuste Fahrspurerkennungssysteme eingesetzt(vgl. (15)). Dort wird ein anderer Filter als Ersatz für den Sobeloperator zur Extraktion von Kanten eingesetzt. Zusätzlich wird das System um eine Separation von gelben und weißen Fahrbahnmarkierungen erweitert.

Der Peak-finding-Algorithmus(vgl. Kapitel 2.3) wird als Teilschritt in einem Steurungsassistenzsystem eingesetzt(vgl. (2)). Dort werden Anpassungen an der Gruppierung der extrahierten Maxima zu Linien vorgenommen. Des Weiteren existiert eine Umsetzung des Algorithmus als System-on-Chip(vgl. (13)). Dort wird auf die Grauwertdarstellung verzichtet und stattdessen eine Maximaextraktion in den Pixelströmen durchgeführt. Die Realisierung wird direkt in den CMOS-Bildsensor integriert.

Zur Verbesserung der Wahrscheinlichkeitsoptimierung wurde für den LOIS-Deformable-Template-Algorithmus(vgl. Kapitel 2.4) ein Echtzeitwahlprozessor entwickelt(vgl. (6)). Der Prozessor hat die Aufgabe Abschnitte aus dem Parameterraum für die Optimierung zu selektieren. Die Wahl erfolgt duch die Einordnung von extrahierten Kantenpunkten in die Abschnitte(vgl. (6) Fig.3). Das Abschnittspaar mit den meisten Kantenpunkten, das gleichzeitig eine korrekte Fahrbahn aufspannt(Breite,Konzentrizität), wird als Parameterraum selektiert.

Verfahren zur Analyse des optischen Flusses im Bild eignen sich insbesondere für ungünstige Wetterbedingungen(vgl. (7)). Über den zeitlichen Verlauf wird der Fluchtpunkt von parallel zur Straße liegenden Objekte(z.B. eine Leitplanke) bestimmt. Mit Hilfe eines Kalman-Filters werden Fahrbahninformationen und der Fluchtpunkt zusammengeführt.

D. Schematische Architektur des SoC

Zur Implementierung der Fahrspurerkennung wird ein FPGA-basiertes[1], rekonfigurierbares System-on-Chip eingesetzt. Der modulare Aufbau so eines System-on-Chip unterstützt den Entwickler bei der Umsetzung verschiedenster Funktionalitäten. Analog zur Softwareentwicklung werden vorkonfektionierte Hardwaremodule unter Open-Source-Lizenz zur Verfügung gestellt[2]. Diese können dazu genutzt werden, eine Ethernet-Schnittstelle oder eine CameraLink-Schnittstelle in das System-on-Chip einzubinden. Einzelne Schritte der Bildverarbeitung werden ebenfalls modularisiert. Eine Vorverarbeitungsstufe bereitet den Pixelstrom durch Glättung oder Kantenextraktion für die Fahrspurerkennungsstufe auf(vgl. Abbildung D.1). Anschließend

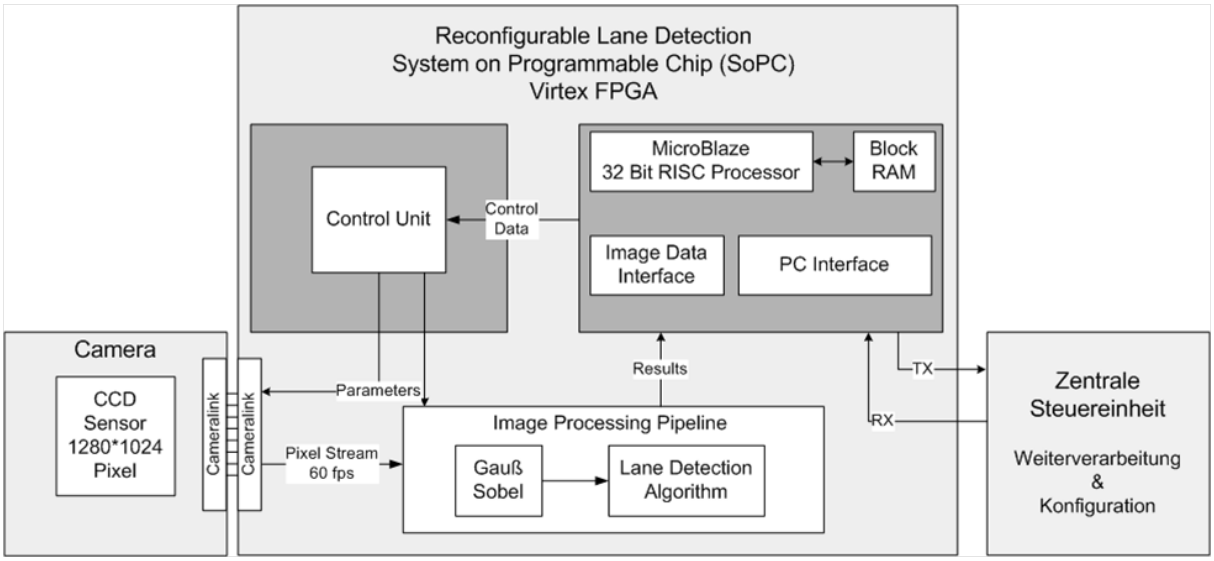


Abb. D.1.: Modularer Aufbau eines System-on-Chip zur Fahrspurerkennung

werden die Fahrbahndaten an die Kommunikationsschnittstelle zur Weiterverarbeitung übergeben. Bei Bedarf lässt sich auf dem Hardwarebaustein ein zusätzlicher RISC-Prozessor integrieren und durch eine Linux-Portierung betreiben. Einsatzgebiete wären unter anderem die Aufbereitung der Fahrbahndaten in Software oder die Steuerungskontrolle über das System-on-Chip(z.B. Kameraparameter).

[1] FPGA - Field Programmable Grid Array
[2] Eine Auswahl liefert: http://www.opencores.org/

E. Deserialisierung des Datenstroms in Schieberegister

Zur Deserialisierung des Datenstroms für eine 3x3 Faltungsmaske werden zwei Schieberegister voller Bildbreite, sowie ein Schieberegister der Länge 3 zur Speicherung der Daten benötigt. Sobald die Struktur komplett gefüllt ist(vgl. Abbildung E.2), wird erstmalig der Ergebniswert der Faltung berechnet. Mit jedem weiteren Pixel wird die Maske verschoben und sofort der nächste Ergebniswert bestimmt.

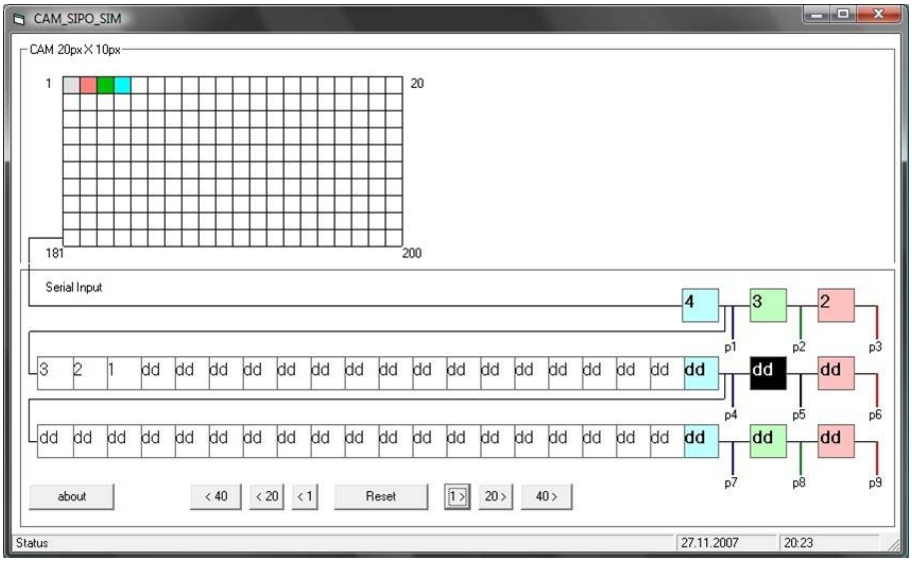


Abb. E.1.: Zustand der Schieberegister beim Auslesen der ersten Bildzeile

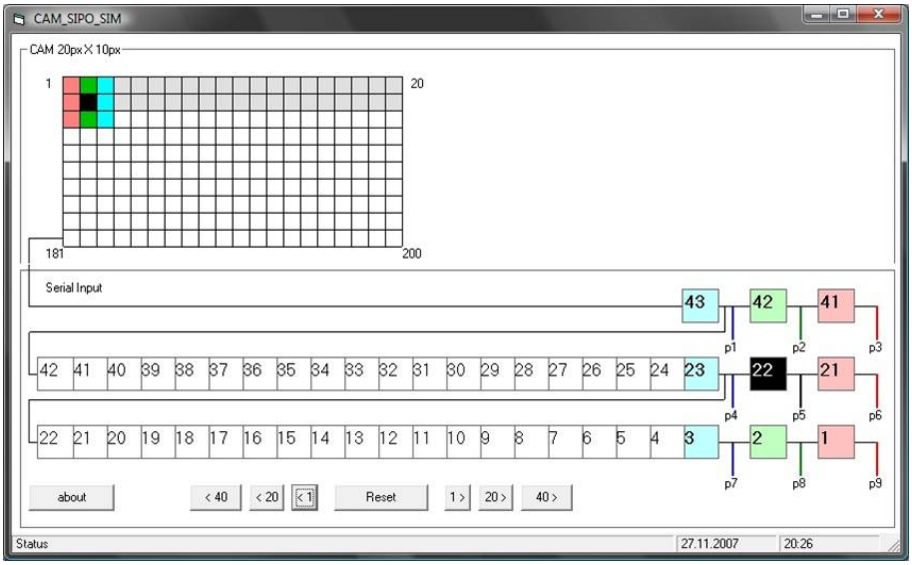


Abb. E.2.: Zustand der Schieberegister beim Auslesen der dritten Bildzeile