Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Benedikt Johannsen

Hardware/Software-Codesign für ein SoC-basiertes Spurführungssystem

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Benedikt Johannsen

Hardware/Software-Codesign für ein SoC-basiertes Spurführungssystem

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

Betreuender Prüfer : Prof. Dr.-Ing. Bernd Schwarz
Zweitgutachter : Prof. Dr. Jürgen Reichardt

Abgegeben am 28. April 2013

Benedikt Johannsen

Thema

Hardware/Software-Codesign für ein SoC-basiertes Spurführungssystem

Stichworte

System-on-Chip, SoC, Spurführungsalgorithmus, Pure Pursuit, FAUST, FPGA, Xilinx System Generator, Projektive Transformation, Hough-Transformation, Parameterstudie, Sichtfeldanalyse Carolo Cup, Xilkernel, Microblaze

Zusammenfassung

Diese Masterarbeit dokumentiert die Optimierung und Erweiterung eines SoC-basierten Spurführungssystems gemäß des Carolo-Cup-Reglements. Die Fahrspurmarkierung wird mit einer CCD-Kamera erfasst und durch eine Hough-Transformation in eine Geradenapproximation überführt, die als Messgröße in die Spurführung auf Basis des Pure-Pursuit Algorithmus eingeht und den Lenkwinkel als Stellgröße des Systems berechnet. Das Spurführungssystem wird anhand seiner mathematischen Modelle hinsichtlich der Parametrisierung untersucht, ein geeigneter Parametersatz extrahiert und durch eine Analyse des Sichtfelds der Kamera mit einer passenden Perspektive unterstützt. Es wird ein Bluetooth-basiertes Datalogging-System entwickelt und zur messtechnischen Analyse und Auswertung von Testfahrten genutzt, um die Ergebnisse der Skalierungen des mathematischen Modelle zu validieren.

Benedikt Johannsen

Title of the paper

Hardware/Software-Codesign for a SoC-based path-following-system

Keywords

System-on-Chip, SoC, Path Following Algorithm, Pure Pursuit, FAUST, FPGA, Xilinx System Generator, Projective transformation, Hough-Transformation, Parameterstudie, Field of View Analysis, Carolo Cup, Xilkernel, Microblaze

Abstract

This master thesis documents the improvement and extension of a SoC-based path-following-system in according to the Carolo-Cup regulations. The lane is detected by a CCD camera and transfered to a lane approximation by a Hough Transformation. The result is the input variable to the path following section based on the Pure-Pursuit algorithm, which calculates the steering angle as manipulated variable of the system. The mathematical models of the path- following-system are examined for parametrization, a suitable set of parameters is extracted and supported by an analysis of the field of view of the camera with an appropriate perspective. A Bluetooth-based data logging system is developed and used for metrological analysis and evaluation of the results of the scaling of the mathematical models.

Inhaltsverzeichnis

1. Einleitung

Die Anforderungen an Energieverbrauch, Baugröße und Rechengeschwindigkeit sowie Datendurchsatz von computergestützten Anwendungen steigen stetig an. Schon heute ist der Großteil der eingesetzten Prozessoren in Embedded-Systemen zu finden, die dort durch eine gezielte Hard- und Softwareauswahl für einen konkreten Algorithmus optimiert worden sind [13]. Vor allem bei Anwendungen mit Echtzeitanforderungen finden daher Hardware-Beschleunigungs-Module Anwendung, um Algorithmen umzusetzen, die in einer Softwarelösung sehr viel Rechenzeit benötigen würden [8]. Die Entwicklung dieser Beschleuniger ist allerdings ebenso zeitintensiv und auch nicht für jeden Algorithmus geeignet. Somit stellt eine Symbiose aus Hard- und Software ein vielversprechendes Konzept zur Bewältigung solcher Aufgaben dar.

FPGAs eignen sich besonders als Basis für ein Hardware-Software-Codesign, da sie eine große Flexibilität in der Entwicklung bieten, indem sie sowohl Hardware-Beschleuniger-Module als auch Prozessoren zur Ausführung von Software umsetzen können [13][30]. Die Kombination von Prozessorsystemen und Hardware-Beschleunigern in einem Chip wird „System-on-Chip“ (SoC) genannt und wird in dieser Master-Arbeit mit einer praxisnahen Beispielanwendung durchdrungen.

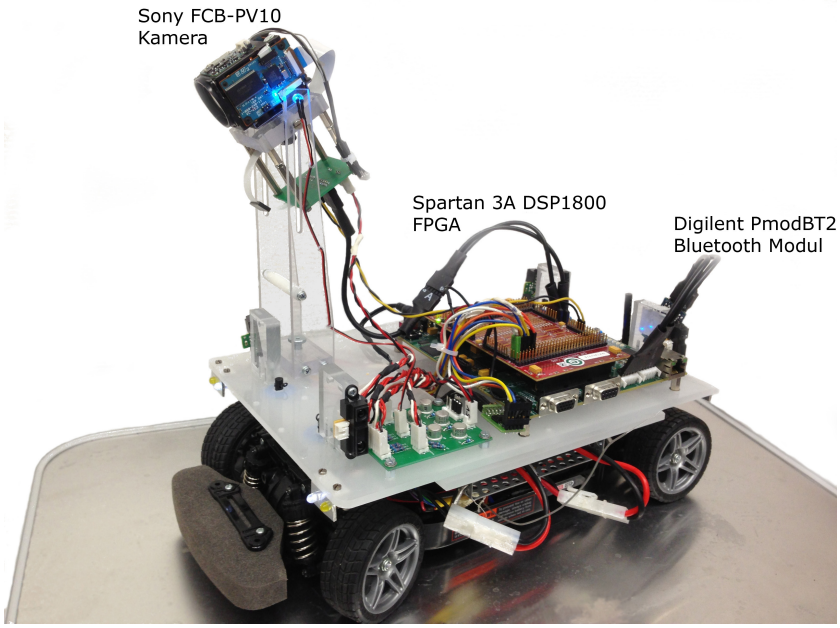

Abbildung 1.1.: Das System-on-Chip Autonomous Vehicle (SAV)

Das Forschungsprojekt FAUST (Fahrerassistenz- und Autonome Systeme) des Departments Informatik der Hochschule für angewandte Wissenschaften Hamburg entwickelt Prototypen für Fahrzeugplattformen für den autonomen Fahrbetrieb [9]. Die Entwicklung orientiert sich am Regelwerk des Carolo-Cups, einem studentischen Wettbewerb für autonome Fahrzeuge. Dementsprechend dienen Modellbauplattformen im Maßstab 1:10 als Basis um die gestellten Aufgaben wie das Befahren eines Rundkurses, Ausweichmanöver oder paralleles Einparken umzusetzen [6]. Der Arbeitskreis Hardware-Software-Codesign entwickelt dabei ein Fahrzeug auf FPGA-Basis, das sogenannte System-on-Chip Autonomous Vehicle (SAV).

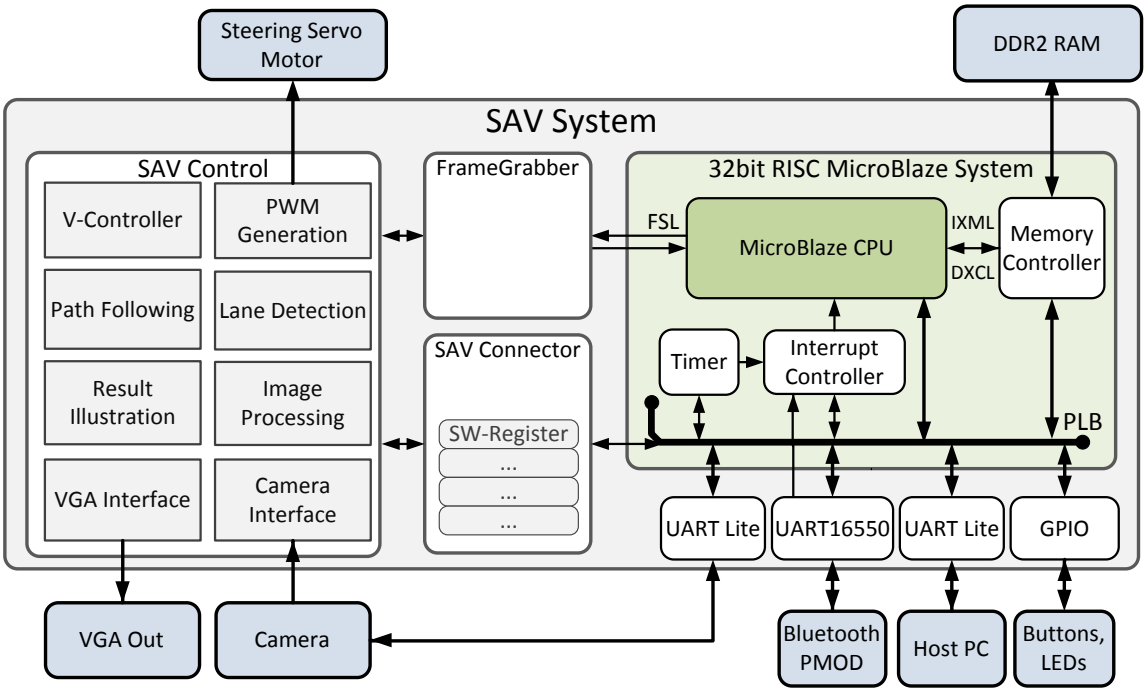

Abbildung 1.2.: Die SoC-Architektur des SAV Systems.

Das SAV ist Summe der Ergebnisse vorangegangener Bachelor- und Masterarbeiten mit den Schwerpunkten Bilddatenverarbeitung, Fahrspurführungsalgorithmen und Geschwindigkeitsregelung [11][12][14][21][22][31]. Diese Masterarbeit liefert Beiträge zur verbesserten Konsistenz der implementierten Algorithmen und damit zum systematischeren Verhalten des Spurführungssystems.

Im Schwerpunkt behandelt diese Masterarbeit die SoC-Implementierung einer Video-basierten Spurführung:

- Die Konsistenz der Festkommaarithmetik der Videostrom-Auswertung wird überprüft. Hierzu zählen die Bildvorverarbeitung (*IMAGE_PROCESSING*), sowie die Erkennung der Fahrspurmarkierung (*LANE_DETECTION*. Die dem System zugrunde liegenden mathematischen Modelle werden analysiert, um einen geeigneten Parametersatz zu bestimmen sowie das Fahrverhalten gezielt zu beeinflussen.

- Es wird das Sichtfeld der Kamera analysiert und anhand der geometrischen Gegebenheiten des Fahrzeugs und der Fahrbahn eine geeignete Kameraperspektive ermittelt.
- Die Mess- und Stellgrößen des Fahrbetriebs werden online übertragen und aufgezeichnet. Die hierfür eingesetzte Bluetooth-Schnittstelle wird ebenso dafür genutzt, eine Parametrisierung zur Laufzeit durchzuführen.
- Die Spurführungsalgorithmik (*PATH_FOLLOWING*) wird in Software portiert, da nach der Fahrspurerkennung keine hohen Datenraten vorliegen, die eine RTL-Lösung erfordern würden.

Aspekte wie das automatische Einparken oder Ausweichmanöver und die damit verbundene zusätzliche Sensorik werden nicht betrachtet.

Gliederung der Arbeit

In **Kapitel 2** werden eingesetzte Technologien und Algorithmen vorgestellt mit besonderem Augenmerk auf die Fahrspurapproximation und die Spurführungsalgorithmen. Die Komponenten des SAV, das Kamera-Interface, die Bilderverarbeitungspipeline, das Spurführungssystem sowie die Ansteuerung der Aktorik werden in **Kapitel 3** beschrieben.

Die mathematische Analyse und Skalierung der Fahrspurführung wird in **Kapitel 4** vorgenommen und die Parameter für den Spurführungsalgorithmus „Pure-Pursuit“ ermittelt, die optimale Kamera-Perspektive bestimmt sowie die Diskretisierung der Hough-Transformations-Ebene vorgenommen.

Kapitel 5 beschreibt die Umsetzung der Bluetooth-Kommunikationsschnittstelle für das Datalogging sowie die Integration der im vorangehenden Kapitel 4 bestimmten Parameter in das System des SAV. Die Erprobungsfahrten zur Validierung und Bestätigung werden in **Kapitel 6** dokumentiert und ausgewertet.

Kapitel 7 schließt diese Abschluss-Arbeit zusammenfassend ab.

2. Technologieübersicht

In diesem Kapitel wird eine Übersicht der verwendeten Technologien gegeben. Dies sind Spurführungsalgorithmen, insbesondere „Pure-Pursuit“, die Geradenapproximation der Fahrspurmarkierung durch eine Hough-Transformation sowie die verwendete FPGA-Platform des SoC des SAV.

2.1. Spurführungsalgorithmen

Die Verfahren zur Spurführung eines autonomen Fahrzeugs werden in die folgenden Kategorien eingeteilt:

Zielpunktverfahren

Diese Verfahren folgen der Spur, in dem sie einen sich kontinuierlich ändernden Zielpunkt, den „Look-Ahead-Point“ (LAP), ansteuern. Hierzu zählen die bekannten Algorithmen „Follow-The-Carrot“ (FTC) und „Pure-Pursuit“ (PP)[25].

Regelung auf einen Sollwert

Ein Regler bestimmt den Lenkwinkel. Die Regelgröße ist der Abstand zur Fahrspur, die als Ist- und Sollwert vorliegen muss.

Im Zuge der Entwicklung des SAV wurden die Zielpunktverfahren „Follow-the-Carrot“ und „Pure-Pursuit“ erprobt sowie deren Kombination mit einem PD-Regler. Die besten Ergebnisse ließen sich mit „Pure Pursuit“ erzielen. Daher wird im Rahmen dieser Arbeit mit diesem Verfahren gearbeitet [22].

Pure-Pursuit

Der Algorithmus Pure-Pursuit fährt den LAP auf einem Kreisbogen mit der Krümmung λ an. Dabei wird die Fahrzeugkinematik durch Einbeziehung des Achsabstands L berücksichtigt. Der Abstand des Fahrzeugs zum LAP wird als „*Look-Ahead-Distance*“ (LAD) bezeichnet (vgl. Abbildung 2.1) [25].

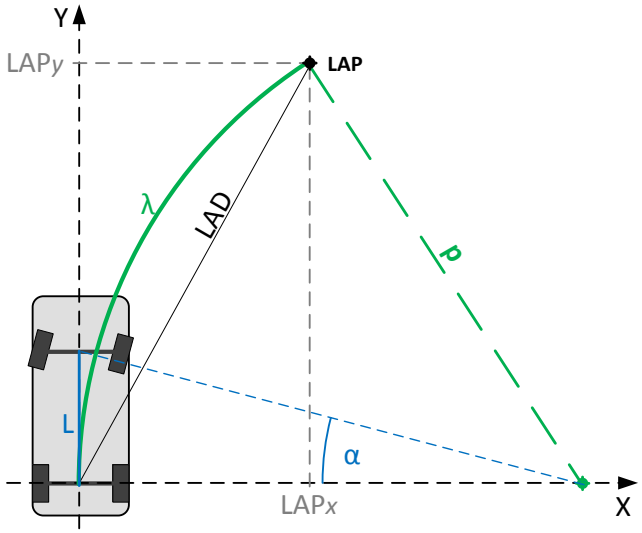

Abbildung 2.1.: Konstruktion des Pure-Pursuit-Spurführungsalgorithmus.

Die Krümmung λ eines Kreisbogens ist wie folgt definiert:

$$\lambda = \frac{1}{p} \tag{2.1}$$

Durch den Satz des Pythagoras wird über zwei rechtwinklige Dreiecke der Radius p mit den Parametern LAD und LAP bestimmt:

$$LAD^2 = LAP_y^2 + LAP_x^2 \tag{2.2}$$

$$p^2 = LAP_y^2 + (p - LAP_x)^2 \tag{2.3}$$

Der Radius des Kreisabschnitts p wird durch Gleichsetzen der Gleichungen 2.2 und 2.3 über LAP_y^2 bestimmt.

$$p = \frac{LAD^2}{2 * LAP_x} \tag{2.4}$$

Hieraus kann unter Berücksichtigung des Achsabstands des Fahrzeugs L der Lenkwinkel α errechnet werden:

$$\alpha = \tan^{-1}\left(\frac{L}{p}\right) = \tan^{-1}\left(\frac{2 * LAP_x * L}{LAD^2}\right) \tag{2.5}$$

Dieses Verfahren ist nur für den Fall geeignet, dass die Kreisradien p deutlich größer als der Achsabstand L sind, da die Gegenkathete des Winkels α nicht auf dem Kreisabschnitt liegt und somit eine Näherung darstellt.

2.2. Hough-Transformation zur Approximation der Fahrspur

Aus der Pixelmenge des Kamerabildes wird die Fahrspurmarkierung extrahiert. Um die Datenmenge zu reduzieren, wird die Fahrbahnmarkierung durch eine Geradenapproximation beschrieben.

Die lineare Geradengleichung der Form $y = m * x + b$ ist allerdings für die Anwendung in der Spurführung nicht nutzbar, da eine senkrechte Gerade mit der Steigung $m \to \infty$ nicht definiert ist. Für eine Spurführung würde dieser Fall jedoch das optimale Szenario darstellen, in dem das Fahrzeug parallel zur Fahrspurmarkierung ausgerichtet ist.

Eine andere Form der Geradenbeschreibung ist die Hessesche Normalform (HNF), die eine Gerade durch eine Lotlänge r zum Ursprung und dem Winkel ϕ zwischen der X-Achse und dem Lot eindeutig beschreibt (vgl. Abbildung 2.2).

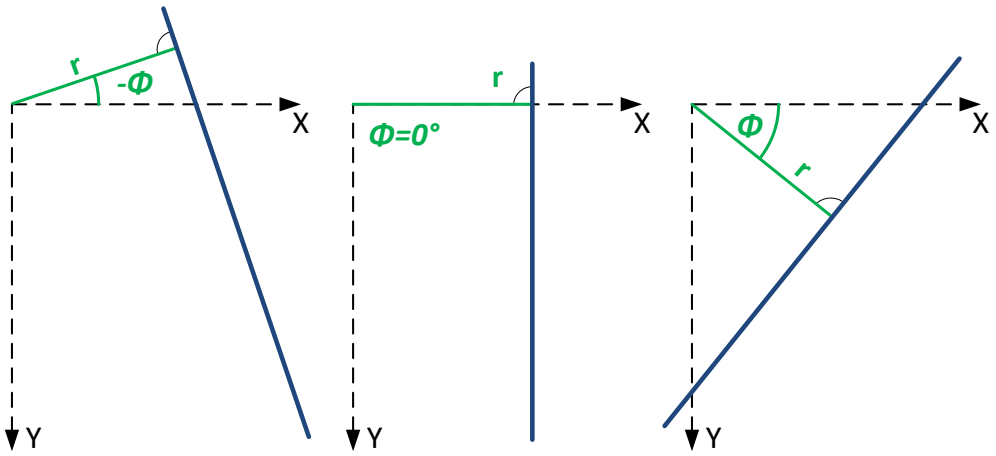

Abbildung 2.2.: Geradendarstellung durch die Hessesche Normalform. Eine Parallele zur Y-Achse ist definiert durch $\phi = 0$.

Ist $\phi = 0°$, so ergibt sich $r = x$, eine Parallele zur Y-Achse mit der Lotlänge x. Bei $\phi = 90°$ ist $r = y$, also eine Parallele zur X-Achse mit der Lotlänge y. Folglich lautet die Geradengleichung der HNF für einen Punkt $(x|y)$:

$$r = x * cos(\phi) + y * sin(\phi) \tag{2.6}$$

So lässt sich für jeden Punkt $P(x|y)$ im kartesischen Koordinatensystem eine Funktion $r = P(\phi)$ bilden. Schnittpunkte dieser Funktionen in der $(r|\phi)$-Ebene haben eine gemeinsame Gerade im kartesischen Koordinatensystem (vgl. Abbildungen 2.3 & 2.4).

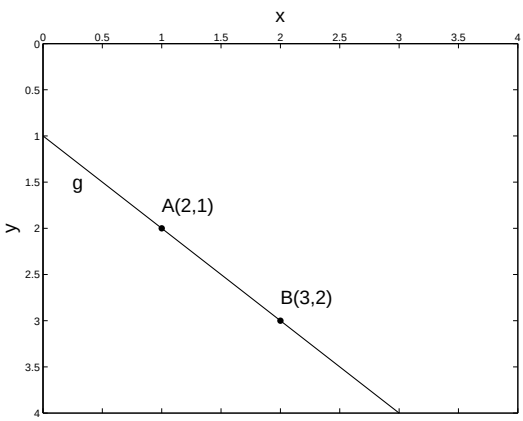

Abbildung 2.3.: Punkte A und B haben die gemeinsame Gerade g [14].

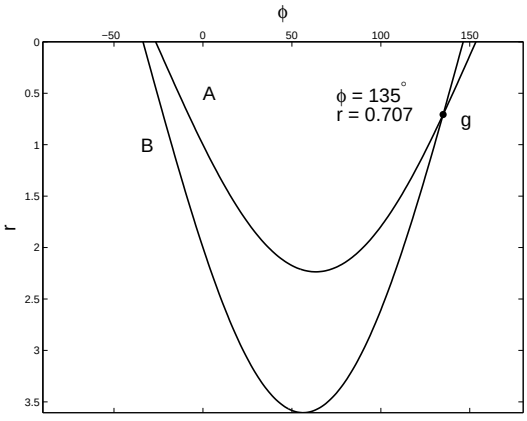

Abbildung 2.4.: In der Hough-Ebene wird g durch Schnittpunkte von A und B definiert [14].

Die Hough-Transformation (HT) ist ein robustes Verfahren zur Bestimmung einer Gerade in der Hesseschen Normalform aus einer Punktemenge. Sie wird genutzt um Geraden aus einem Kamerabild zu extrahieren, indem alle zur zu bestimmenden Gerade gehörenden Pixel als Eingangsparameter für die HNF dienen. Als Ergebnis erhält man eine Menge von Funktionen in der $(r|\phi)$-Ebene, deren Schnittpunkt-Maxima das Ergebnis der Hough-Transformation bildet [5].

2.3. SoC-Plattform des SAV

Das SAV basiert auf einem *Xilinx Spartan 3A DSP Board* mit einem *Xilinx Spartan 3A DSP1800* FPGA. Das Entwicklungsboard verfügt über 128*MB* DDR2 RAM, einen Oszillator mit 125*MHz* Taktfrequenz sowie folgende Peripherie-Elemente:

- **Schnittstellen** : VGA, RS232, Ethernet, JTAG, 2 *Digilent Pmod*-Header
- **Erweiterungs-Slots** : 2 168EXP Connectors mit jeweils 84 I/O Pins
- **Test-Sensoren/Aktoren** : 8 LEDs, 8 DIP Switches, 4 Push Buttons

Die Erweiterungsschnittstellen sind mit einem *AVNET EXP Bus Prototyping Board* belegt, das die IO Pins des FPGAs auf Pin-Header, legt an denen die Aktorik und Sensorik des SAV angeschlossen wird [3]. Die *Digilent Pmod* Header sind mit einem *Digilent PmodBT2* Bluetooth Modul für die drahtlose Datalogging Schnittstelle belegt (vgl. Abbildung 3.11) [7]. Die RS232-Schnittstelle wird zur Kommunikation mit dem Host-PC sowie zum Übertragen von aufgezeichneten Kamerabildern mit dem Frame-Grabber genutzt (vgl. Kap. 3.4). Über den VGA-Ausgang werden Kamerabilder ausgegeben um sie auf einem Monitor darzustellen.

Die komplette Übersicht über das System wird in Kapitel 5.1 auf Seite 38 gegeben.

3. Komponenten des SoC-basierten Spurführungssystems

Im Folgenden wird das Gesamtsystem des SAVs anhand seiner funktionellen Komponenten aufgezeigt. Sein Design basiert auf einer Pipeline-Architektur, die durch den eingehenden Pixelstrom einer CCD-Kamera betrieben wird. Die einzelnen Stufen der Pipeline sind dabei teilweise übertaktet um die entsprechende Funktionalität in der zur Verfügung stehenden Zeit von einem Pixeltakt durchführen zu können.

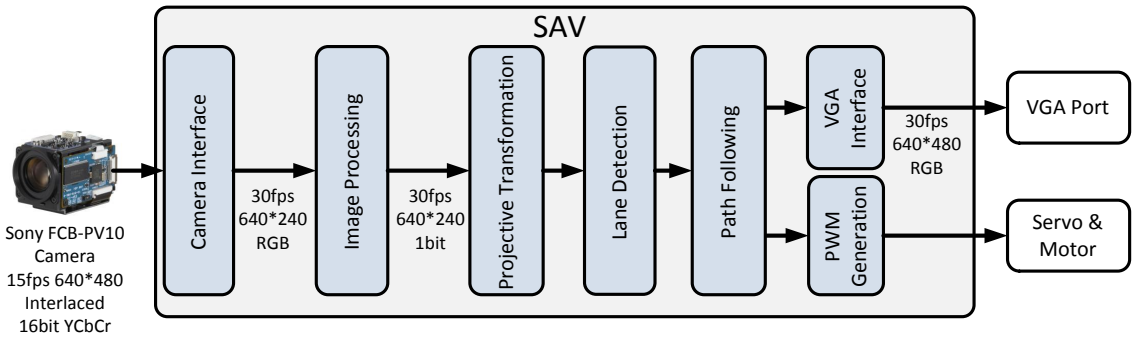

Abbildung 3.1.: Übersicht über die Funktionsmodule des SAV mit Darstellung des Bildformats des Pixelstroms zwischen den Pipelinestufen.

3.1. Bildverarbeitungspipeline zur Fahrspurerkennung

Als Kamera wird eine *Sony FCB-PV10* CCD-Kamera eingesetzt. Diese bietet eine VGA-Auflösung von $640 * 480$ Pixeln im *YCbCr*-Format und ist auf einen Interlacing Modus konfiguriert. Sie sendet somit pro Bild nur jede zweite Zeile, so dass die resultierenden Bilder nur noch die halbe vertikale Auflösung von $640 * 240$ Pixeln haben. Dies staucht die Perspektive des Bildes. Vorteil des Interlacing Modus ist die höhere Bildfrequenz von $30Hz$. Der resultierende Pixeltakt, der die Taktung der nachfolgenden Bildverarbeitungspipeline bestimmt, ist $13.5MHz$ [11] [23].

3.1.1. Kamera-Interface

Das Kamera-Interface ist als VHDL-Beschleuniger-Modul realisiert. Es verarbeitet den eingehenden Pixelstrom der Kamera und bereitet ihn für die nachfolgenden Bearbeitungsschritte auf. Es übernimmt die folgenden Aufgaben [11]:

Synchronisation und Generierung von Steuersignalen

Aus dem Pixelstrom werden die *FrameValid* (FVAL) und *LineValid* (LVAL) Steuersignale generiert. Diese erleichtern die Synchronisierung mit dem Pixelstrom für nachfolgende Module.

Konvertierung von YCrCb zu RGB

Die Kamera sendet im *YCrCb* Format im Verhältnis 4:2:2. Zur weiteren Verarbeitung sowie zur späteren Ausgabe auf einem VGA-Controller wird dieses in RGB Darstellung konvertiert.

3.1.2. Bildvorverarbeitung

Als Vorbereitung für die Gradenerkennung wird das Bild bearbeitet um Störfaktoren bei der Erkennung zu minimieren. Diese Beschleuniger-Module sind als *Xilinx System Generator* Blöcke implementiert. Sie dienen dazu den Pixelstrom für die nachfolgende Hough-Transformation aufzubereiten sowie die relevanten „Vordergrundpixel“ zu identifizieren und zu extrahieren (siehe Kap. 2.2).

Adaptive Binarisierung

Der eingehende Pixelstrom mit Grauwerten wird in ein binäres schwarz-weiß Bild konvertiert, um scharfe Kontraste und damit Kanten zu extrahieren sowie Hardware Ressourcen in den nachfolgenden Pipelinestufen zu sparen, da die benötigten Bitbreiten reduziert werden.
Um unabhängig von unterschiedlichen Beleuchtungssituationen zu sein wird der Schwellenwert für diese Binarisierung dynamisch angepasst. Hierzu wird das vorhergehende Bild analysiert und der maximale Grauwert ermittelt [31].

Erosionsfilter

Erosionsfilter dienen dazu feine Konturen im Bild zu eliminieren, um Störeinflüsse zu minimieren. Dies erleichtert die nachfolgende Hough-Transformation zur Fahrspurapproximierung . Der Filter arbeitet mit einer 3x3 Pixel Faltungsmatrix. Der Wert des zentralen Pixels wird im binären Pixelstrom aus der UND-Verknüpfung seiner 8 angrenzenden Nachbarn gebildet. Ist mindestens ein benachbartes Pixel kein „Vordergrundpixel“, so wird das zentrale Pixel angepasst und ist ebenfalls kein Vordergrundpixel (vgl. Abbildung 3.2).

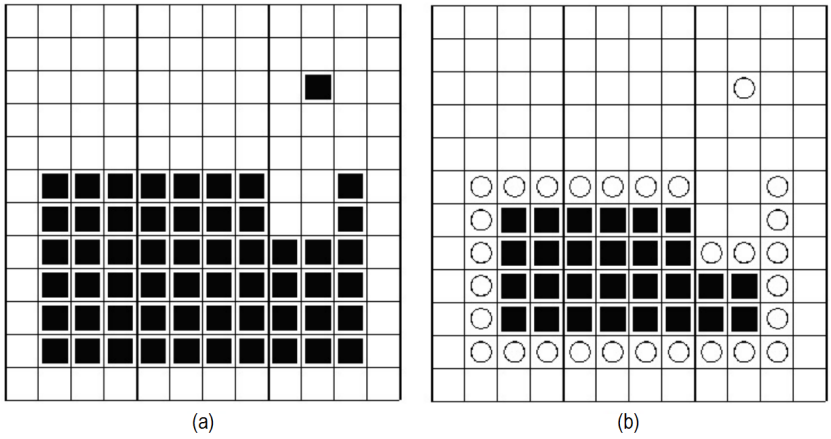

Abbildung 3.2.: Links das unbearbeitete Bild, rechts bearbeitet mit einem einfachen Erosionsfilter. (Quelle: [31])

3.1.3. Projektive Transformation

Die verwendeten Algorithmen und Verfahren zur Umsetzung der visuellen Fahrspurerkennung setzen voraus, dass das eingehende Bild eine Draufsicht auf die Fahrbahn zeigt. Da die hierfür nötige Kameraposition für ein autonomes Fahrzeug jedoch unpraktikabel ist, muss die vorhandene Perspektive durch eine projektive Transformation umgerechnet werden (vgl. Abbildung 3.3)

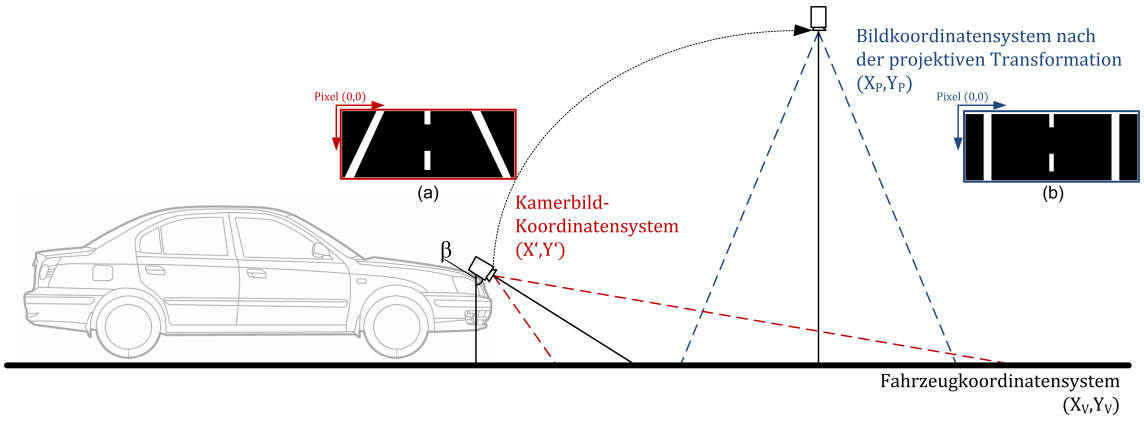

Abbildung 3.3.: Funktionsprinzip der Projektiven Transformation (Quelle: [12])

Es wird jedem Pixel $(x'|y')$ aus dem ursprünglichen Bild der Kamera eine korrigierte Position $(X_k|X_k)$ im Bild zugewiesen. Dies wird durch eine $3 * 3$ Transformations-Matrix B errechnet:

$$\begin{pmatrix} x_k \\ y_k \\ 1 \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} * \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \tag{3.1}$$

Aus dem Gleichungs-System 3.1 werden folgende Gleichungen zur Berechnung der korrigierten Pixelposition für die x und y Koordinaten extrahiert:

$$x_k = \frac{b_{11}x' + b_{12}y' + b_{13}}{b_{31}x' + b_{32}y' + 1} \qquad (3.2)$$

$$y_k = \frac{b_{21}x' + b_{22}y' + b_{23}}{b_{31}x' + b_{32}y' + 1} \qquad (3.3)$$

Die nachfolgenden Grafiken 3.4 und 3.5 zeigen das Kamerabild ohne projektive Transformation (links) und die gleiche Perspektive mit projektiver Transformation (rechts):

Abbildung 3.4.: Kamerabild ohne PT.

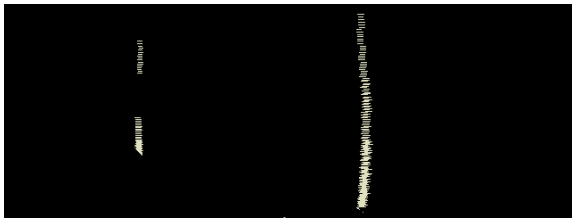

Abbildung 3.5.: Kamerabild mit PT.

Die Bestimmung von B wird einmalig nach der Wahl der Kameraperspektive durchgeführt. Hierfür werden 4 Referenzpunkte benötigt, typischerweise in den Bildecken. Für jeden Referenzpunkt werden die metrischen Koordinaten im Fahrzeugkoordinatensystem den $(x|y)$ Pixelkoordinaten im Kamerabild $(x'|y')$ zugeordnet. Der Vorgang zur Kalibrierung ist im Anhang A.3 detailliert beschrieben [10].

3.2. Fahrspurerkennung

Die Erkennung der Fahrspur basiert auf einer Approximation der Fahrspurmarkierung durch eine Gerade. Diese wird mit einer Hough-Transformation umgesetzt. (vgl. Kapitel 2.2) Damit die Hough-Ebene kleiner dimensioniert werden kann, wird nicht das komplette Bild betrachtet, sondern nur eine „Region of Interest“ (ROI) (siehe Abbildung 3.6). Aus den geometrischen Gegebenheiten der Fahrspur und der bekannten Kameraperspektive des Fahrzeugs kann ein Erwartungswert für die Position der Fahrspur abgeleitet werden. Diese Annahme dient als Grundlage für die Positionierung der ROI.

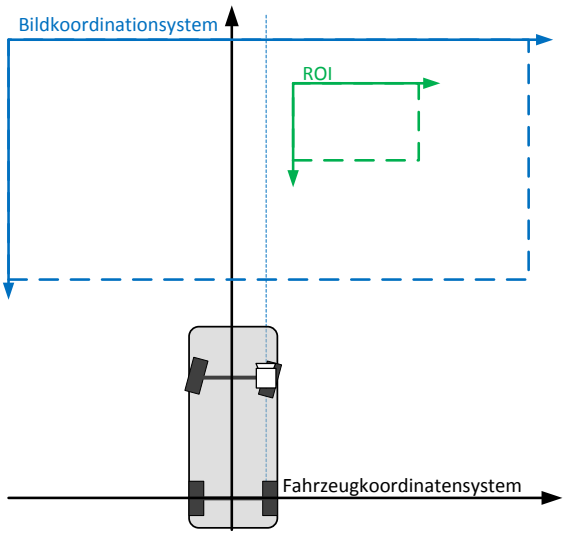

Abbildung 3.6.: Koordinatensystem der ROI im Kamerakoordinatensystem des SAV.

Dynamische horizontale ROI-Position

Durch die Lenkbewegung des Fahrzeugs und die auftretenden Kurvenradien ist der horizontale Spielraum der Position der Fahrspurmarkierung im Kamerabild deutlich größer als die ROI. Daher wird die ROI der Fahrspur dynamisch nachgeführt und nach jedem Bild horizontal neu positioniert [22]. Die nächste Position der ROI wird mit Gleichung 3.4 ermittelt:

$$ROI_{x_{neu}} = ROI_x - \frac{ROI_{width}}{2} + r \tag{3.4}$$

Durch die Berücksichtigung der Breite der ROI und der Lotlänge r aus der Hough-Transformation wird die ROI für das nächste Bild so positioniert, dass die erkannte Gerade in der Mitte der ROI liegen würde. Somit bietet die ROI den größtmöglichen Varianzbereich für die Gerade im nachfolgenden Bild.

Hough-Transformation

Die RTL-Implementierung der Hough-Transformation nutzt eine Pipelining-Struktur und ist als Multizyklusdatenpfad umgesetzt worden um durch Ressource-Sharing Ressourcen zu sparen [14].

Zur Umsetzung muss die Hough-Ebene diskretisiert werden um sie auf Hardware-Strukturen abzubilden. So entstehent ein zweidimensionale Feld mit den Dimensionen $\phi * r$, dessen Zellen die Funktion von Akkumulatoren übernehmen, die die

Häufigkeit des Auftretens eines Ergebnisses $(r|\phi)$ zählen (vgl. Abb. 3.7). Die Adresse eines Akkumulators entspricht dem Ergebnis $(r|\phi)$ dessen Häufigkeit er zählt.

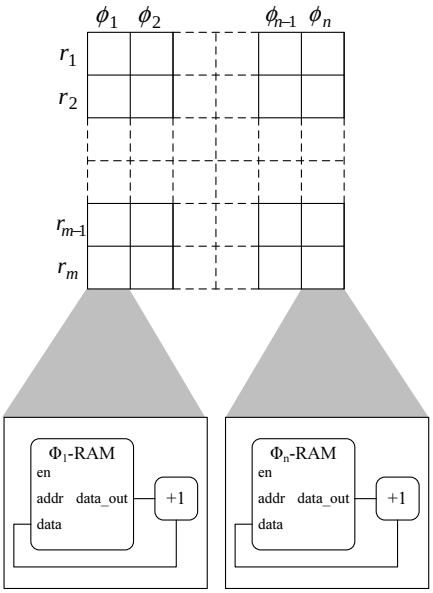

Abbildung 3.7.: Struktur der Hough-Ebene im RTL-Entwurf mit $\phi * r$ Akkumulatoren (Quelle:[14]).

Die Berechnung erfolgt in parallelen Transformatoren, die jeweils einen diskreten Winkel ϕ repräsentieren und für eingehende Vordergrundpixel $(X_p|Y_p)$ einen r-Wert errechnen. Das Ergebnis wird in einem lokalen RAM gespeichert und entspricht einer Zeile aus der Hough-Ebene (vgl. 3.7). Da die Ergebnismenge für r negative Werte beinhaltet, wird aus dem Wert eine Adresse generiert, die als Index für den RAM nutzbar ist (vgl. Abb. 3.8).

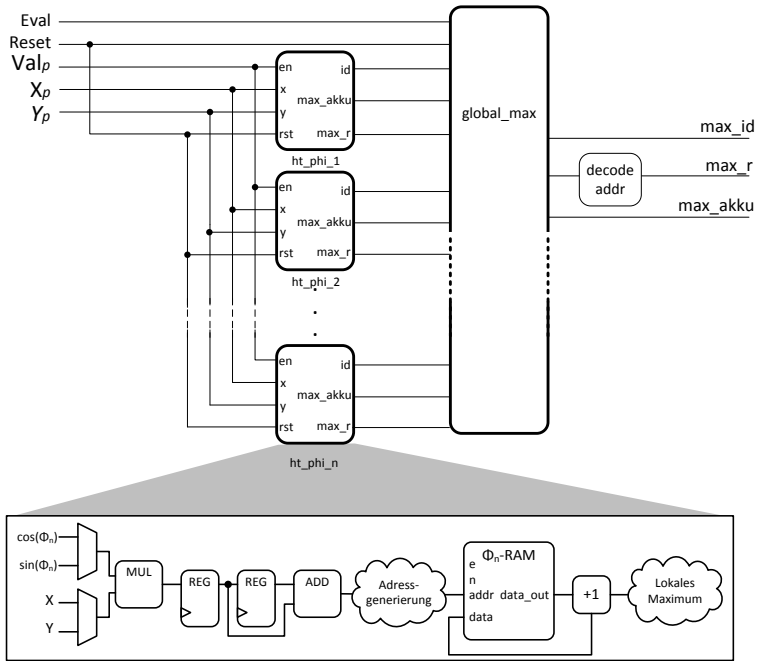

Abbildung 3.8.: Struktur des RTL-Entwurfs der Hough-Transformation (Quelle:[14]).

Jeder Transformator enthält einen Multiplizierer, den er für beide Multiplikationen verwendet und so durch Ressource-Sharing RTL-Ressourcen einspart (vgl. Gl. 2.6). Nach vollständiger Transformation der Menge der Vordergrundpixel wird das globale Maximum der Transformatoren bestimmt und als Ergebnis der Geradenapproximation ausgegeben.

3.3. Spurführungssystem

Die Spurführung ist in einem *Xilinx System Generator*-Modell implementiert worden und setzt die Algorithmen *Follow-the-Carrot* (FTC), *Pure-Pursuit* (PP) und eine Kombination von Pure-Pursuit mit einem PD-Regler um. Durch den Einsatz eines Multizyklusdatenpfads können RTL-Ressourcen wie beispielsweise Multiplizierer mehrfach verwendet werden [22].

Es wird zuerst der Zielpunkt LAP bestimmt und danach der eigentliche Spurführungsalgorithmus. Der LAP wird durch die Fahrspurapproximation der Hough-Transformation sowie den Sollabstand d zur Fahrspurmarkierung in Abhängigkeit von LAD definiert (vgl. Abbildung 3.9).

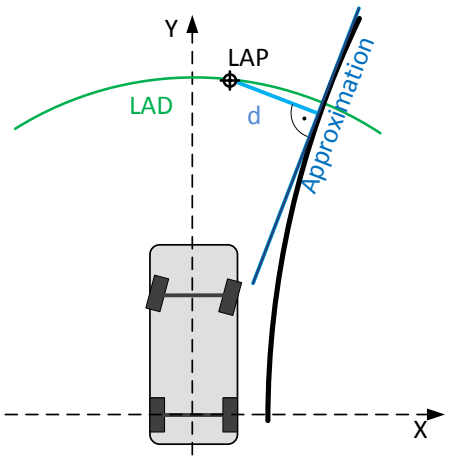

Abbildung 3.9.: Konstruktion des LAP

3.4. Frame-Grabber

Der Frame-Grabber ist ein Modul zur Analyse und Dokumentation des Bilddatenstroms, der Bilder aus dem Datenstrom aufzeichnet. Er besteht aus einem RTL-IP Block, der den Bilddatenstrom abgreift und über eine FSL-Busanbindung an den *Microblaze*-Prozessor überträgt sowie aus einer Software-Komponente, die den eingehenden Datenstrom empfängt, ihn im DDR2-RAM ablegt und die Daten im Bitmap-

Format über die serielle Schnittstelle zum Host-PC überträgt[11][21]. Abbildung 3.10 zeigt ein mit dem Frame-Grabber aufgezeichnetes Bild.

3.5. VGA Darstellung des Systemzustands

Das Kamerabild wird auf einem VGA-Ausgang ausgegeben um eine Visualisierung der Ergebnisse aus Fahrspurerkennung und Spurführung des Systems zu erhalten. Grundlage für diese Anzeige ist der Pixelstrom nach der Bildvorverarbeitung. In dieses Bild werden die Ergebnisse eingeblendet sowie Orientierungshilfen für den Bezug zum Fahrzeugkoordinatensystem dargestellt.

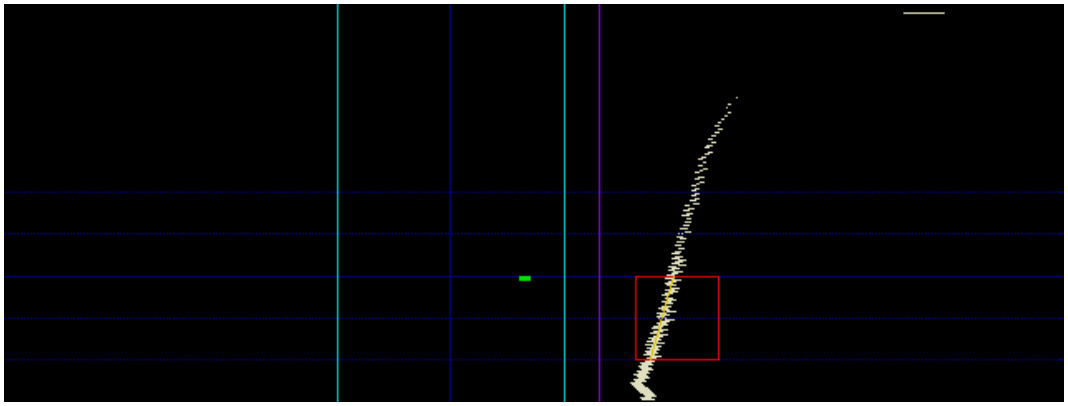

Abbildung 3.10.: Grafische Ausgabe des SAV mit Ergebnissen der Fahrspurerkennung. Aufgezeichnet mit dem Frame-Grabber. Es werden der LAP (*grün*), die ROI (*rot*), die Fahrspurapproximation (*orange*), das Fahrzeugkoordinatensystem (*blau*), der Sollabstand d (*violett*) sowie die Kanten des Fahrzeugs (*türkis*) dargestellt.

Das Bild hat weiterhin eine durch das Interlacing reduzierte Auflösung von $640 * 240$ Pixeln. (vgl. Kap. 3.1)
Um es auf einem VGA-Ausgang darstellen zu können, muss es wieder auf die VGA-kompatible Auflösung von $640 * 480$ Pixeln erweitert werden. Dies geschieht durch eine Verdopplung der Bildzeilen, die jede Zeile zwei mal in Folge ausgibt. Das resultierende Bild wird dadurch vertikal gestreckt, sodass die ursprünglichen Proportionen des Kamerabildes wiederhergestellt werden (vgl. Kap. 3.1)[11].

3.6. Ansteuerung der Aktorik

Die Aktorik des Fahrzeugs besteht aus einen Lenk-Servo sowie einem Brushless-Motor, dessen Hall-Sensoren zum Auslesen der Drehzahl genutzt werden (vgl. Abb. 3.11). Die Ansteuerung für beide Aktoren wird durch PWM-Signale realisiert, deren Timing in Tabelle 3.1 beschrieben ist:

Motor	Lenkservo	Duty Cycle
Vorwärts	rechts	10% (2ms)
Stillstand	geradeaus	7.5% (1.5ms)
Rückwärts	links	5% (1ms)

Tabelle 3.1.: Eigenschaften der PWM-Signale mit einer Periodendauer von 20ms und einem Pegel von 3.3V zur Ansteuerung der Aktorik des SAV.

Die Fahrzeuggeschwindigkeit wird mit einem VHDL-Modul mit einem PI-Regler kontrolliert um den Einfluss der Akku-Spannung auf die Geschwindigkeit zu reduzieren. Durch die Pulse der Hall-Sensoren des Brushless-Motors wird die Drehzahl des Motors gemessen und als Regelgröße genutzt [4][24].

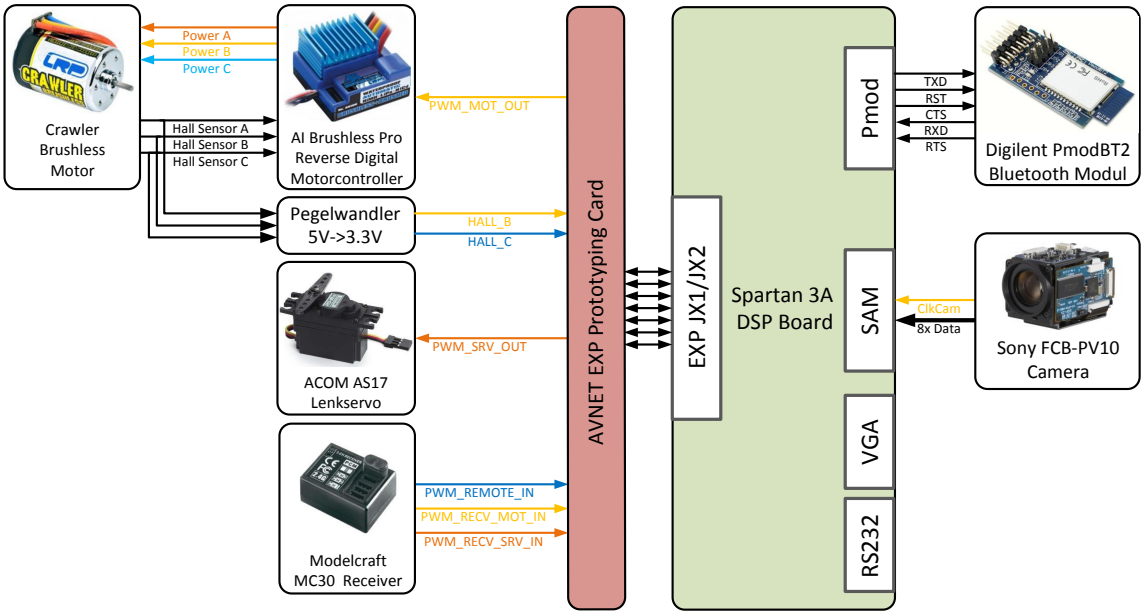

Abbildung 3.11.: Datenverbindungen zwischen dem FPGA-Board und der Peripherie des SAV.

Aus Sicherheitsgründen sieht das Carolo-Cup-Reglement vor, dass der autonome Fahrbetrieb des Fahrzeugs jederzeit per Fernbedienung unterbrochen werden kann. Diese Anforderung wird durch die Nutzung des dritten PWM-Kanals der Fernbedienung erfüllt. Wird kein Signal empfangen, werden die PWM-Signale der Handfernbedienung an Lenkung und Motorsteuerung genutzt und somit auf eine manuelle Steuerung des Fahrzeugs umgeschaltet (vgl. Abb. 3.11). Somit greift dieser Mechanismus auch im Fall einer unterbrochenen Funkverbindung zur Fernbedienung [6].

4. Skalierung des mathematischen Modells der Fahrspurführung

Die Komponenten des SAV basieren auf mathematischen Modellen, deren Parameter in diesem Kapitel identifiziert und analysiert werden, um ihren Einfluss auf das Gesamtsystem zu ermitteln sowie einen Parametersatz zu errechnen, der als Ausgangsbasis für Testfahrten dienen soll. Hierbei werden die Spurführungs-Regelung, die Kameraperspektive sowie die Dimensionierung der Hough-Transformation-Ebene betrachtet.

4.1. Spurführungs-Regelung

Die Spurführungs-Regelung basiert auf dem Pure-Pursuit-Algorithmus, der nach Kapitel 2.1 Gleichung 2.5, folgendes Stellgesetz liefert:

$$\alpha = \tan^{-1}\left(\frac{2 * LAP_X * L}{LAD^2}\right) \tag{4.1}$$

Der Achsabstand L ist durch die Fahrzeuggeometrie vorgegeben und LAD ein Parameter des Algorithmus. Die X-Komponente des LAP wird zur Laufzeit berechnet. Dies wird linear gelöst, indem das Fahrzeugkoordinatensystem $(X|Y)$ um den Winkel ϕ aus der Hough-Transformation rotiert wird damit man das Hilfskoordinatensystem $(X_h|Y_h)$ erhält, dessen Y-Achse parallel zur Approximation der Fahrspurmarkierung liegt (siehe Abbildung 4.1).

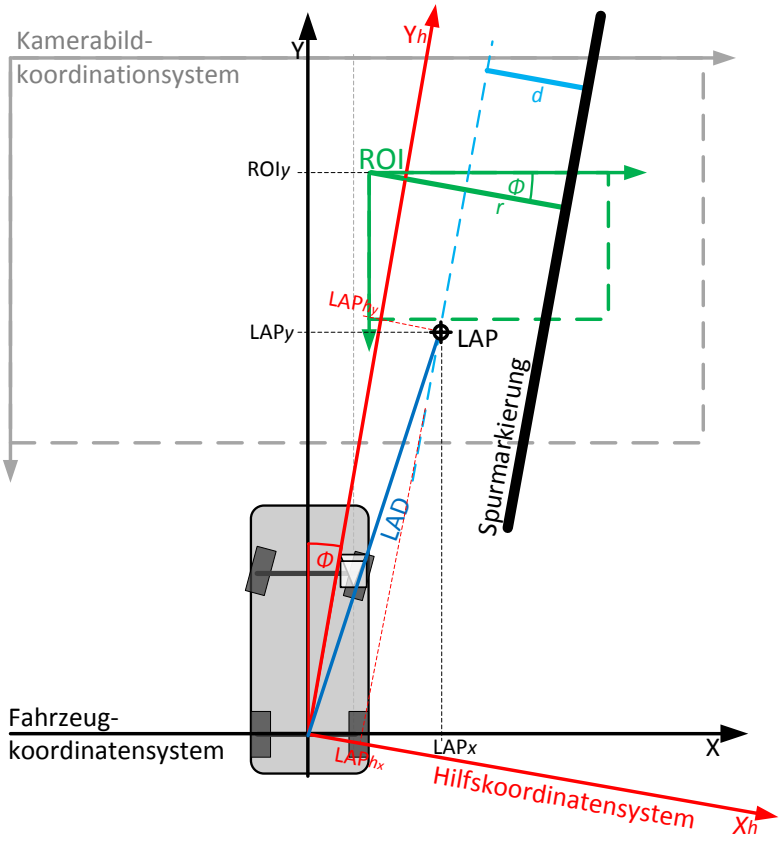

Abbildung 4.1.: Konstruktion des LAP durch Drehung des Fahrzeugkoordinatensystems $(X|Y)$ um ϕ in ein Hilfskoordinatensystem $(X_h|Y_h)$.

Der Rechenweg ist nachfolgend dargestellt. Der Ursprung der ROI wird in das Hilfskoordinatensystem überführt:

$$\begin{pmatrix} ROI_{X_h} \\ ROI_{Y_h} \end{pmatrix} = \begin{pmatrix} cos(\phi) & sin(\phi) \\ -sin(\phi) & cos(\phi) \end{pmatrix} * \begin{pmatrix} ROI_X \\ ROI_Y \end{pmatrix} \tag{4.2}$$

Im Hilfskoordinatensystem lassen sich beide LAP-Komponenten linear berechnen:

$$LAP_{X_h} = ROI_{X_h} + r - d \tag{4.3}$$

$$LAP_{Y_h} = \sqrt{LAD^2 - LAP_{X_h}^2} \tag{4.4}$$

Der LAP wird zurück in das Fahrzeugkoordinatensystem überführt, damit er für die Spurführungsalgorithmen verwendbar ist:

$$\begin{pmatrix} LAP_X \\ LAP_Y \end{pmatrix} = \begin{pmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{pmatrix} * \begin{pmatrix} LAP_{X_h} \\ LAP_{Y_h} \end{pmatrix} \tag{4.5}$$

In die Berechnung des Zielpunktes LAP fließen folgende zur Laufzeit ermittelte Größen ein:

- ROI_X : Die horizontale Position der ROI (vgl. Gleichung 4.2 und 4.3) wird zur Laufzeit dynamisch angepasst um die ROI an die Position der Fahrspurmarkierung im Kamerabild anzupassen (vgl. Kapitel 3.2, S.16).
- r : Die Lotlänge r aus der Hough-Transformation beschreibt die Fahrspurapproximation (vgl. Gleichung 4.3).

Zudem gehen folgende Parameter mit ein, deren Einfluss in einer Parameterstudie analysiert und geeignete Werte ermittelt werden:

- ROI_Y : Die vertikale Position der ROI (vgl. Gleichung 4.2 und 4.3).
- LAD : Die Distanz zum LAP, gemessen ab dem Ursprung des Fahrzeugkoordinatensystems.
- d : Der Sollabstand d des Drehpunktes des Fahrzeugs im Ursprung des Fahrzeugkoordinatensystems zur Fahrspurmarkierung.

4.1.1. Parameterstudien zu Pure-Pursuit-Gleichungen

Für eine Parameterstudie des Pure-Pursuit-Algorithmus werden die Wertebereiche der extrahierten Parameter festgelegt. Dafür werden folgende Annahmen getroffen:

- Die berechneten Kurven sind Kreisabschnitte mit einem Mittelpunkt auf der X-Achse des Fahrzeugkoordinatensystems. (vgl. Abbildung 4.2)
- Es werden nur Rechtskurven simuliert, da ein Unterschied zu Linkskurven nur durch die horizontale Begrenzung des Kamerabildes gegeben wäre, die in dieser Studie nicht berücksichtigt wird.
- Der Sollabstand d sowie der Ist-Abstand d_{ist} zur Fahrspurmarkierung beträgt jeweils $0cm$ vom Fahrzeugmittelpunkt.
- Der Wertebereich von LAD ist auf $45cm - 165cm$ festgelegt und deckt damit einen größeren vertikalen Bereich ab als die Kamera erfassen kann (vgl. Kap. 4.2).
- Der Wertebereich von ROI_y ist auf $45cm - 100cm$ festgelegt worden, da der minimale Kurvenradius mit $100cm$ definiert ist [6] und somit bei größeren ROI_y Werten die ROI oberhalb der Fahrspurmarkierung liegen und so nicht erkannt würde (vgl. Abbildung 4.3).

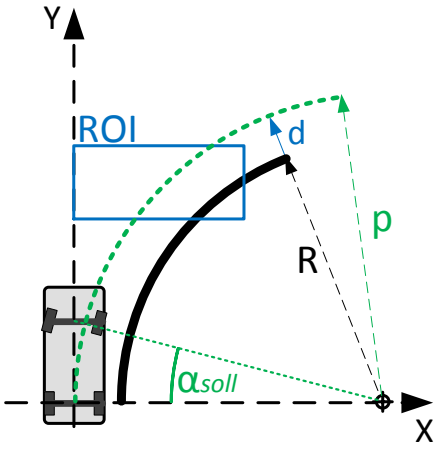

Abbildung 4.2.: Konstruktion der in der Parameterstudie verwendeten Fahrsituation.

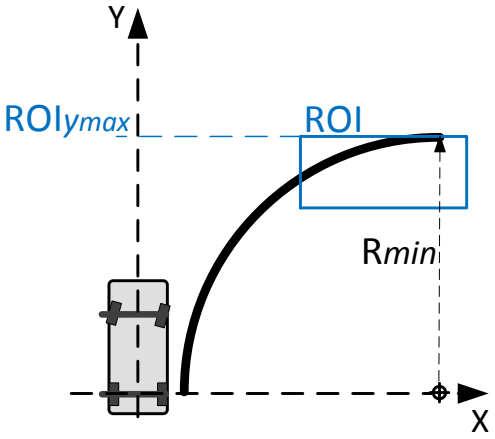

Abbildung 4.3.: Der minimale Kurvenradius $R_{min} = 100cm$ begrenzt die maximale ROI_y Position der ROI.

Für diese Fahrsituation mit $d = d_{ist}$ ist der gewünschte Lenkwinkel α_{soll} nach Pure-Pursuit die Krümmung eines Kreises mit dem Referenzradius R_{soll}, der im Abstand d zur Fahrspurmarkierung verläuft (vgl. Abbildung 4.2).

$$R_{soll} = R + d \tag{4.6}$$

Unter diesen Annahmen werden für alle Kombinationen von LAD- und ROI_y-Werten mit Vorgabe eines Kurvenradius R und einem Sollabstand d der Radius R_{pp} nach Pure-Pursuit berechnet:

$$R_{pp} = \frac{L}{tan(\alpha)} \tag{4.7}$$

Über alle Parameterpaare, die den Radius R_{pp} mit einer Genauigkeit von ±5% ergeben, wird der Durchschnitt ermittelt. Das Ergebnis wird als Funktion $R_{pp} = f(ROI_Y, LAD)$ in einem 3D-Plot dargestellt (siehe Abbildung 4.4).

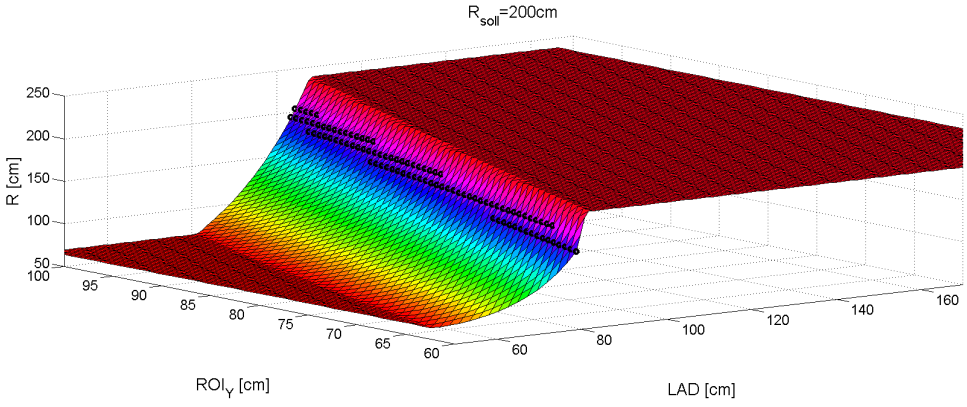

Abbildung 4.4.: Grafische Darstellung der Parameterstudie für den Kurvenradius $R = 200cm$. Die Parameterpaare (ROI_Y, LAD), deren Ergebnis in der Toleranz von R_{pp} liegen, sind schwarz markiert (siehe Anhang C.2)

Die Studie wurde in $10cm$ Schritten für Kurvenradien von $R = 100cm$ bis $R = 5000cm$ durchgeführt und die Ergebnisse in Abhängigkeit von R aufgetragen:

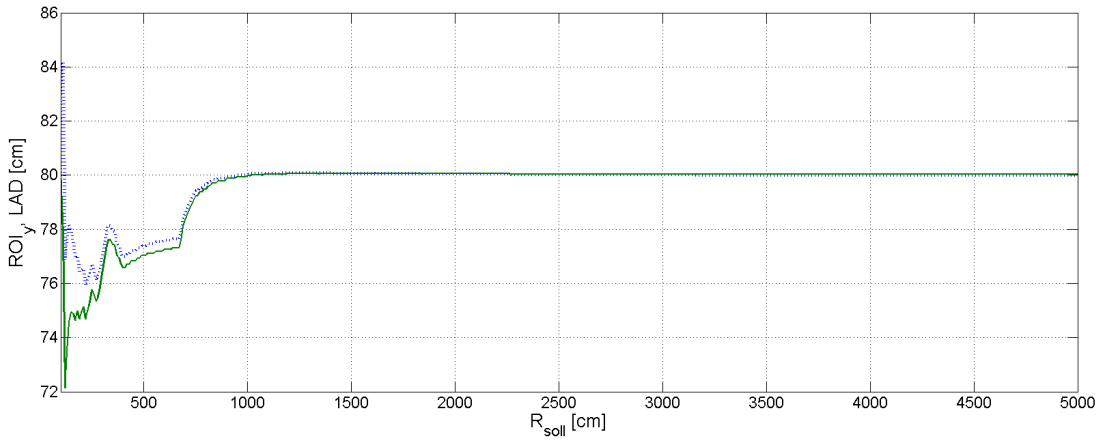

Abbildung 4.5.: ROI_Y (blau, gepunktet) und LAD (grün) als Ergebnis der einzelnen Parameterstudien in Abhängigkeit von R.

Das Ergebnis der Parameterstudie lässt folgende Aussagen zu:

- ROI_Y und LAD sollten im Verhältnis $1:1$ gewählt werden. Bei kleineren Radien R ist es vorteilhaft wenn LAD kleiner ROI_Y ist.
- Für größere Kurvenradien R konvergieren ROI_Y und LAD gegen $80cm$. Bei kleineren Radien sind kleinere Werte von bis zu $72cm$ zu wählen.

Entsprechend werden $ROI_Y = LAD = 80cm$ als Ausgangspunkt für die nachfolgende Bestimmung der Kameraperspektive angenommen.

Anmerkung: Zu einem späteren Zeitpunkt dieser Abschlussarbeit wurde eine fehlerhaften Parameterübergabe zwischen der Hough-Transformation und der Spurführungsalgorithmik festgestellt. Die Transformationsrichtung des Winkels ϕ wird in der Hough-Transformation im Uhrzeigersinn, in der Spurführung jedoch gegen den Uhrzeigersinn interpretiert. Entsprechend muss der Winkel ϕ der Hough-Transformation mit negativen Vorzeichen in die Spurführung eingehen. Die hier ermittelten Werte basieren auf einem Winkel ϕ mit positiven Vorzeichen und sind daher nicht korrekt. Im Anhang D werden die Auswirkungen und Korrekturen erläutert.

4.2. Bestimmung der Kameraperspektive

Die Wahl der Kameraperspektive bestimmt, welche Ausschnitte der Fahrspur vom SAV erfasst und bearbeitet werden. In der Parameterstudie des Spurführungsalgorithmus wurde ein ROI_y-Wert von $80cm$ als Stellgröße ermittelt, so dass die Perspektive so gewählt wird, dass das Kamerabild auf dieser Höhe möglichst breit ist um den ROI_x-Variationsbereich zu erhöhen (vgl. Kapitel 3.2). Zudem wird die Auswirkung der Perspektive auf die nutzbaren ROI_y-Parameter, sowie die Auflösung der Fahrspur im Kamerabild untersucht.

Die Kamera ist auf dem Fahrzeug so montiert, dass sie in ihrer Neigung β in vertikaler Richtung verstellbar ist. Wird die Kamera weniger geneigt, so dass sie weiter in die Ferne gerichtet ist, deckt ihr Sichtbereich eine größere Fläche ab. Dies hat allerdings den Nachteil eines größeren Abstands zwischen Kamerabild und Fahrzeug und daher einer geringeren Auflösung, da jedes Pixel einen größeren Bildbereich repräsentiert (siehe Abbildung 3.3).

4.2.1. Einfluss des Neigungswinkels der Kamera auf das Sichtfeld

Die *Sony FCB PV10* Kamera hat einen horizontalen Sichtwinkel γ_h von 46° und ein Seitenverhältnis von 4 : 3 [23]. Sie ist in einer Höhe von $h = 28cm$ über der Vorderachse des Fahrzeugs montiert. Aus diesen Angaben wird der Einfluss der Neigung β auf die Breite des Sichtbereichs ermittelt:

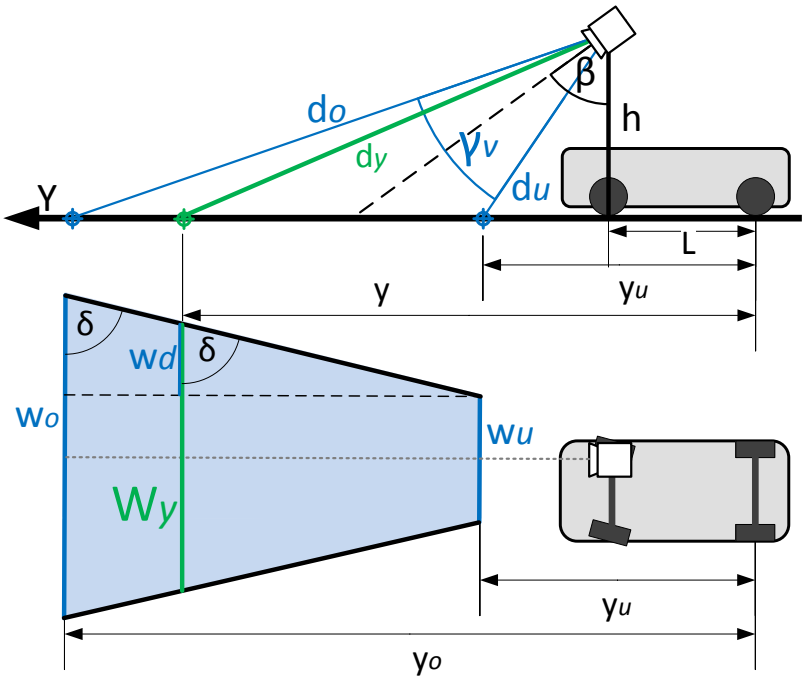

Abbildung 4.6.: Der Neigungswinkel β der Kamera bestimmt die Breite des Kamerabildes an der Ober- und Unterkante des Bildes.

Die Breite des Kamerabildes W_y bei $Y = y$ setzt sich zusammen aus der Breite der Unterkante w_u des Bildes sowie dem Anteil w_d:

$$W_y = w_u + 2 * w_d \tag{4.8}$$

w_d wird über den Tangens des Winkels δ bestimmt, der über das rechtwinklige Dreieck mit der Gegenkathete $d_o - d_u$ und der Ankathete $0.5 * (w_o - w_u)$ ermittelt wird:

$$w_d = \left(\frac{y - y_u}{tan(\delta)}\right) \tag{4.9}$$

$$\delta = tan^{-}1\left(\frac{y_o - y_u}{0.5 * (w_o - w_u)}\right) \tag{4.10}$$

Durch Einsetzen von Gleichung 4.10 in Gleichung 4.9 erhält man:

$$w_d = \left(\frac{(y - y_u)(0.5 * (w_o - w_u))}{y_o - y_u}\right) \tag{4.11}$$

Die Breite des Sichtbereichs an der Oberkante w_o und der Unterkante w_u werden mit den Entfernungen d_u und d_o zwischen der Kamera und den erfassten Punkten auf der Fahrbahn berechnet (vgl. Abbildung 4.7 und 4.6).

$$w_u = 2 * d_u * sin(0.5 * \gamma_h) \tag{4.12}$$

$$w_o = 2 * d_o * sin(0.5 * \gamma_h) \tag{4.13}$$

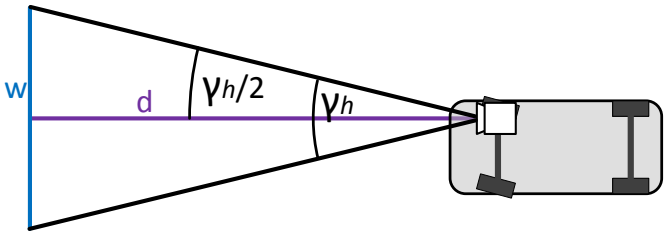

Abbildung 4.7.: Die Breite des Kamerabildes w in der Entfernung d kann durch den horizontalen Sichtwinkel γ_h bestimmt werden.

Die Distanzen d_u und d_o werden über rechtwinklige Dreiecke mit der Kameraneigung β in Kombination mit dem vertikalen Sichtwinkel γ_v bestimmt (vgl. Abbildung 4.6):

$$d_u = \frac{h}{cos(\beta - 0.5\gamma_v)} \tag{4.14}$$

$$d_o = \frac{h}{cos(\beta + 0.5\gamma_v)} \tag{4.15}$$

Der vertikale Sichtwinkel γ_h wird durch den horizontalen Sichtwinkel γ_h sowie das Seitenverhältnis bestimmt:

$$\gamma_v = \gamma_h * \frac{3}{4} \tag{4.16}$$

Die Strecken zwischen Kamera und den Bildkanten auf der Fahrbahn y_o und y_o in Y-Richtung wird wie folgt ermittelt (vgl. Abbildung 4.6):

$$y_u = tan(\beta - 0.5 * \gamma_v) * h + L \qquad (4.17) \qquad y_o = tan(\beta + 0.5 * \gamma_v) * h + L \qquad (4.18)$$

Der mathematische Zusammenhang zwischen der Breite W_y des Kamerabildes in der Entfernung y und dem Neigungswinkel der Kamera β wird als Funktion $W(\beta, y)$ durch sukzessives Einsetzen der Gleichungen 4.9 bis 4.18 in Gleichung 4.8 sowie durch Vereinfachen bestimmt:

$$W(\beta, y) = \left(\frac{2 * sin(0.5 * \gamma_h) * \Big(h * cos(\beta) - L * sin(\beta) + y * sin(\beta)\Big)}{cos(0.5 * \gamma_v)} \right) \qquad (4.19)$$

Der Einstellungsbereich für die Neigung β ist nach unten begrenzt auf $\beta_{min} = 23°$, da die Kamera bei kleineren Winkeln die Front des Fahrzeugs mit erfasst. Bei einem Winkel von $\beta_{max} = 90° - 0.5 * \gamma_v = 72°$ liegt d_o parallel zur Y-Achse, sodass y_o nicht mehr bestimmt werden kann und das Kamerabild den Horizont abbildet. Abbildung 4.8 zeigt die Ergebnisse der Untersuchung dieses Winkelbereichs auf W_y bei $y = 80cm$:

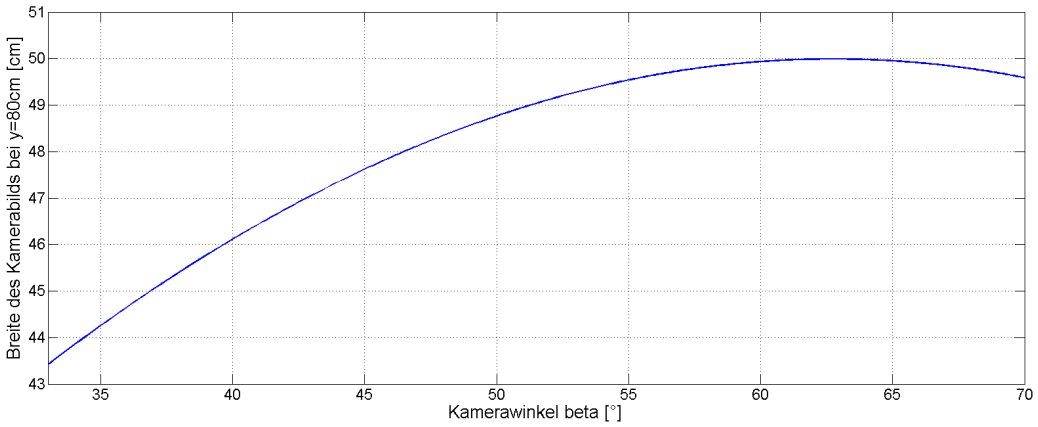

Abbildung 4.8.: Einfluss des Neigungswinkels β der Kamera auf die Breite des Sichtfeldes auf Höhe $y = 80cm$. Die maximale Breite des Sichtfeldes W_{max} wird bei $\beta = 62.7°$ erreicht.

Die Funktion hat einen Maximalwert W_{max}, dessen Abhängigkeit von der Entfernung y durch Gleichsetzung der Ableitung von Gleichung 4.19 $W' = 0$ bestimmt wird:

$$W(\beta, y)' = \left(\frac{-2 * sin(0.5 * \gamma_h) * \Big(L * cos(\beta) + h * sin(\beta) + y * cos(\beta)\Big)}{cos(0.5 * \gamma_v)} \right) \qquad (4.20)$$

Durch Gleichsetzen von $W' = 0$ wird die Distanz y bestimmt, an der W maximal ist:

$$y_{W_{max}} = h * tan(\beta) + L \tag{4.21}$$

Das Ergebnis zeigt, dass die maximale Bildbreite W_{max} immer auf der Höhe y erreicht wird, auf die die Mittelachse der Kamera ausgerichtet ist (vgl. Abbildung 4.6). Für die Distanz von $80cm$ bedeutet dies einen Neigungswinkel von $\beta = 62.7°$. Durch den Verlauf der Funktion um diesen Maximalwert herum ergeben sich zudem Intervalle, die den Winkelbereich erweitern und nur minimal an Bildbreite verlieren. Im folgenden Intervall I_w ist die Abweichung zu W_{max} kleiner als $2mm$:

$$I_w = \left\{\beta || W(\beta_{max}, 80cm) - W(\beta, 80cm)| < 2mm\right\} = [57.8°, 67.7°] \tag{4.22}$$

Die Perspektive wird innerhalb dieses Intervalls auf weitere Zusammenhänge und Einflüsse untersucht.

4.2.2. Intervall der vertikalen ROI-Position

Abhängig von der Kameraperspektive ergibt sich das Intervall der vertikalen ROI Positionen, bei denen die ROI komplett im Kamerabild liegt. Das Intervall I_w der in Kapitel 4.2.1 bestimmten Neigungen der Kamera wurde auf den Variationsbereich der ROI untersucht (vgl. Gleichung 4.22).

Der Ursprung des ROI-Koordinatensystems liegt in seiner oberen linken Ecke, so dass seine maximale Y-Lage bei $ROI_{y_{max}} = y_o$ liegt (vgl. Abb. 3.6 und Gl. 4.13). Die minimale Y-Position $ROI_{y_{min}}$ ergibt sich aus der unteren Bildkante y_u und der metrischen Höhe der ROI durch die Perspektive (vgl. Gleichung 4.23).

$$ROI_{y_{min}} = y_u + ROI_{height_{cm}} + L \tag{4.23}$$

Die metrische Höhe der ROI wird aus der Höhe im Pixelkoordinatensystem multipliziert mit dem Verhältnis von metrischer Bildhöhe zu Y-Pixelauflösung der Kamera errechnet:

$$ROI_{height_{cm}} = ROI_{height_{px}} * \left(\frac{y_o - y_u}{240px}\right) \tag{4.24}$$

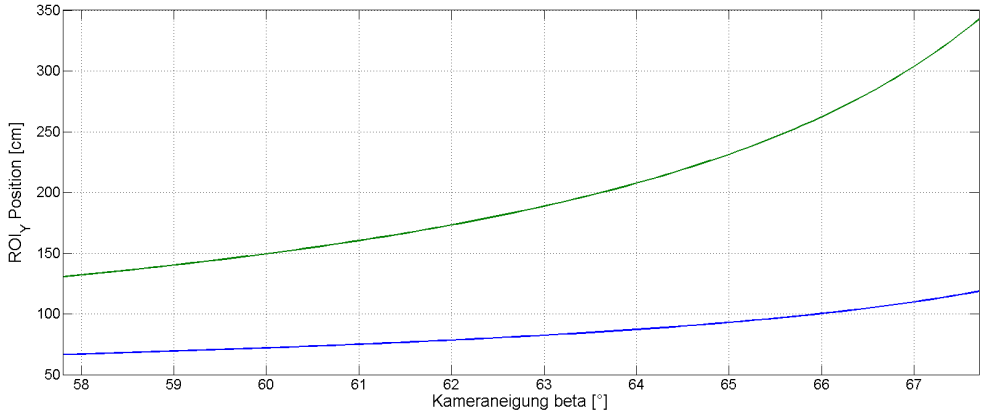

Abbildung 4.9.: Einfluss des Neigungswinkels β auf den vertikalen Variationsbereich der ROI. Die obere Grenze (grün) liegt an der oberen Kante des Kamerabildes, die untere Grenze (blau) liegt um die Höhe der ROI über der unteren Kante.

Ausgehend vom Ergebnis der Parameterstudie zum Pure-Pursuit-Spurführungsalgorithmus in Kapitel 4.1 mit $ROI_y = 80cm$ kann das Intervall I_w weiter begrenzt werden, da der ROI auf dieser Y-Position im Kamerabild abgebildet sein soll. Durch Einsetzen in Gleichung 4.23 und Auflösen nach β wird der Kamerawinkel β berechnet, der für $ROI_{y_{min}} = 80cm$ bei einer vertikalen ROI-Größe von $ROI_{height} = 50px$ ergibt:

$$\beta_{ROI_y} = 62.4° \tag{4.25}$$

Somit ergibt sich das Intervall I_{roi80} für den Kamerawinkel β.

$$I_{roi80} = \left\{\beta | \beta \in I_w \wedge ROIy_{min}(\beta) = 80cm\right\} = [57.8°, 62.4°] \tag{4.26}$$

In diesem Intervall liegen die ROI-Positionen $67cm$ bis $180cm$.

4.2.3. Auflösung der Fahrspurmarkierung

Der Neigungswinkel β der Kamera bestimmt die Größe des Sichtfelds und damit die Auflösung des Bildes bei konstanter Pixelzahl. Im Hinblick auf die Fahrspur-Approximation durch die Hough-Transformation ist die Anzahl der Vordergrundpixel zur Darstellung der $2cm$ breiten Fahrspurmarkierung in der ROI entscheidend, da bei zu breiter Darstellung mehrere Geraden in die Fahrspur passen und das Ergebnis nicht eindeutig ist [6] [22].

Die Breite der Fahrspur im Pixelkoordinatensystem P_{px} wird durch ihre Breite im Fahrzeugkoordinatensystem multipliziert mit dem Verhältnis der horizontalen Auflösung der Kamera zur Breite des Sichtfelds der Kamera w_o definiert (siehe Gleichung

4.13). Der Einfluss der Erosionsfilter (vgl. Kap. 3.1.2) wurde dabei nicht berücksichtigt. Für P_{px} ergibt sich daher:

$$P_{px} = 2cm * \frac{640px}{w_o} = 2cm * \frac{640px}{2 * sin(0.5 * \gamma_v) * \frac{h}{cos(\beta+0.5*\gamma_v)}} \tag{4.27}$$

Der Einfluss der Kameraneigung β wurde auf die Darstellungsbreite untersucht, das Ergebnis ist in Abbildung 4.10 dargestellt.

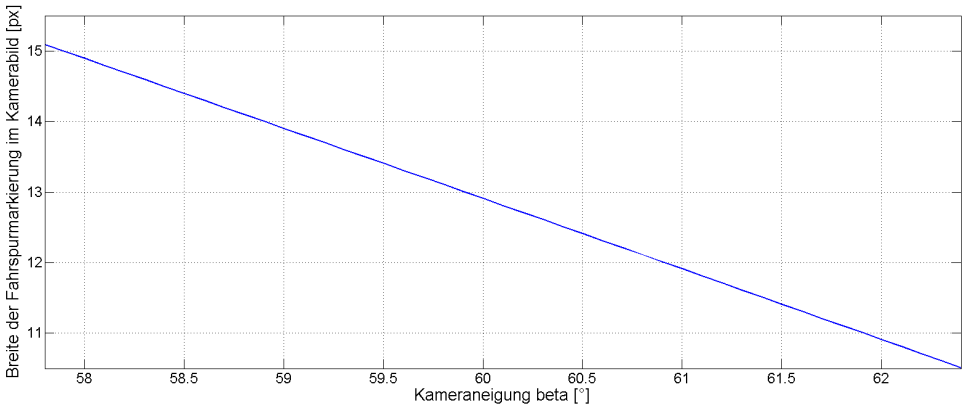

Abbildung 4.10.: Einfluss des Neigungswinkels β der Kamera auf die Darstellungsbreite der Fahrspurmarkierung im Kamerakoordinatensystem.

Die Breite der Fahrspur im Kamerabild verringert sich bei steigender Kameraneigung und verhält sich im betrachteten Intervall I_{roi80} annähernd linear. An der unteren Grenze des Intervalls mit $\beta = 57.8°$ ist die Fahrspur im Kamerabild $15px$ breit abgebildet und damit schmaler als die ROI mit $50px$ Breite. Durch eine Erhöhung der Neigung auf die obere Grenze $\beta = 62.4°$ wird die Fahrspurdarstellung auf $10px$ Breite reduziert.

4.2.4. Entscheidung der Perspektive

Perspektiven mit einem Kameraneigungswinkel β im Intervall I_{ROI80} lassen die Nutzung der vertikalen ROI-Position von $ROI_y = 80cm$ gemäß der Parameterstudie zur Spurführung zu (vgl. Kap. 4.1). Ebenso ist die Breite des Kamerabildes an dieser Stelle höchstens $2mm$ schmaler als der maximal mögliche Wert, so dass der horizontale Variationsbereich der ROI nur minimal eingeschränkt wird. Hinsichtlich der Darstellungsbreite der Fahrspurmarkierung im Kamerabild und der damit verbundenen Ungenauigkeiten der Fahrspurapproximation ist eine größere Neigung vorteilhaft, da hier die Fahrspur schmaler dargestellt wird (vgl. Kap. 4.2.3). Somit ist die obere Grenze des Intervalls I_{ROI80} als die bestmögliche Perspektive zu betrachten. Allerdings sind die Unterschiede zu den anderen Perspektiven innerhalb des Intervalls minimal.

Die Perspektive für die Kamera des SAV wurde daher mit dem Schwerpunkt hinsichtlich der Testfahrten so gewählt, das eine vertikale ROI-Position von $ROI_y = 70cm$ noch im Kamerabild liegt um den Spielraum für diesen Parameter zu erhöhen. Die entsprechende Kameraneigung β ergibt sich aus der gewünschten ROI_Y Position von $70cm$ sowie der Höhe der ROI in Abhängigkeit zur Kameraauflösung:

$$70cm = y_u + ROI_{height_{px}} * (\frac{y_o - y_u}{240px}) + L \tag{4.28}$$

Durch Einsetzen der Gleichungen 4.17 und 4.18 und Auflösen nach β erhält man eine Kameraneigung von $\beta = 59.2°$. Durch die grobe Justagemöglichkeit der Kamera wurde ein Winkel von 59.0° eingestellt. Die Abweichung zur Zielperspektive sind in Tabelle 4.1 festgehalten und zeigen keine messbaren Unterschiede:

Kriterium	$\beta = 62.4°$	$\beta = 59.2°$	$\beta = 59.0°$
W bei $80cm$	$50.0cm$	$49.9cm$	$49.9cm$
Spurbreite in px	$11px$	$14px$	$14px$
$ROI_{y_{min}}$	$80cm$	$70cm$	$70cm$

Tabelle 4.1.: Vergleich der bestmöglichen Perspektive mit der gewählten Perspektive zur Nutzung von $ROI_Y = 70cm$.

Abbildung 4.11 stellt die genutzte Perspektive mit $\beta = 59°$ dar:

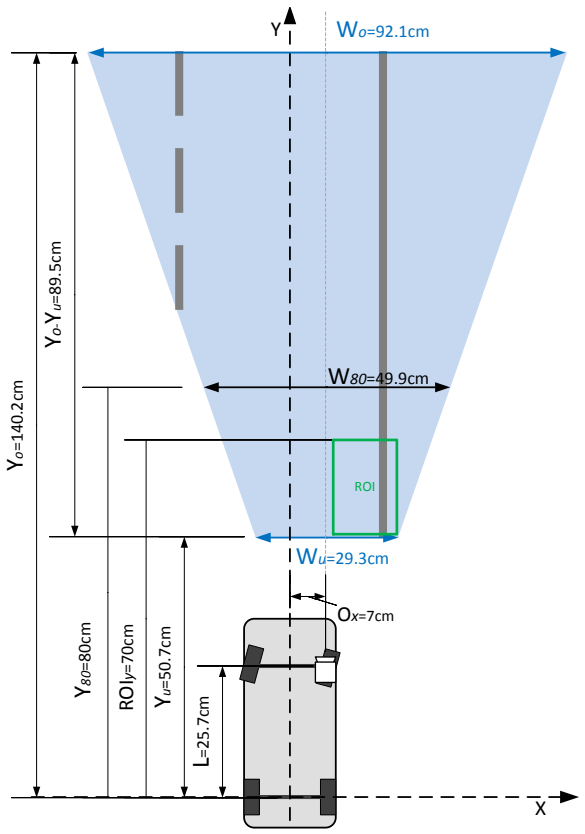

Abbildung 4.11.: Ergebnis der Kameraperspektive mit $\beta = 59.0°$. Die ROI liegt bei $ROI_y = 70cm$ an der unteren Bildkante.

4.3. Diskretisierung der Hough-Transformations-Ebene

Die Umsetzung der Fahrspurapproximation durch eine Hough-Transformation (HT) in RTL erfordert eine Diskretisierung der Hough-Ebene (vgl. Kap. 2.2). Die Implementierung der HT setzt parallel Transformatoren für jeden Winkel ϕ zur Berechnung von r für jedes Vordergrundpixel ein, so dass die Anzahl der Transformatoren direkt vom abgedeckten Winkelbereich sowie der Schrittweite zwischen den Winkel abhängt und dieser den Bedarf an RTL Ressourcen bestimmt (vgl. Kap.3.2).

4.3.1. Wertebereich des Winkelbereichs

Mit einer *Xilinx System Generator* Simulation wurde untersucht, wie sich die Hough-Transformation verhält, wenn der Winkel der betrachteten Fahrspur außerhalb des Winkelbereichs der Transformation liegt. Es wurde ein Kamerabild mit einer Geraden als Stimulus für das *LANE DETECTION* Modul verwendet und die Ergebnisse der Hough-Ebene aufgezeichnet um sie mit *Matlab* zu visualisieren (vgl. Abb. 4.12). Auf diese Weise wurde eine statische Analyse der RTL-Implementierung der Hough-Transformation durchgeführt:

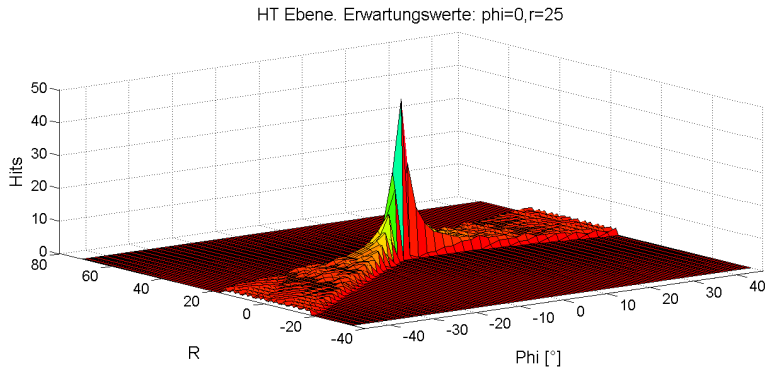

Abbildung 4.12.: Darstellung der Hough-Ebene für eine simulierte Gerade mit $\phi = 0°$ und $r = 25px$. Das globale Maximum hebt sich deutlich von den anderen Feldern der Ebene ab.

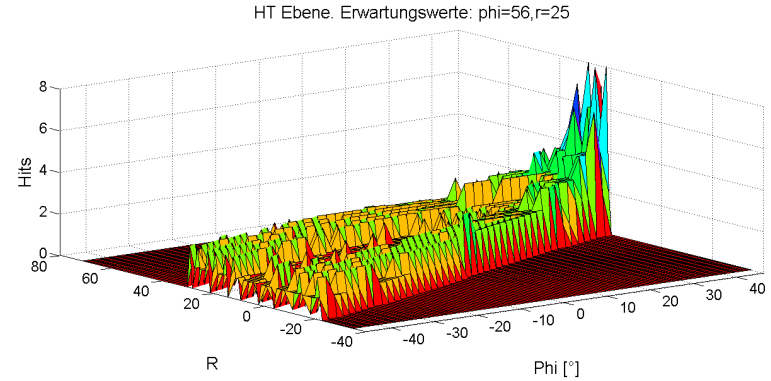

Abbildung 4.13.: Darstellung der Hough-Ebene für eine simulierte Gerade mit $\phi = 56°$ und $r = 25px$. Die Hough-Transformation ist nur bis 46° dimensioniert und das globale Maximum liegt an der oberen Grenze des abgedeckten Bereichs.

Die Hough-Transformation ergibt für Winkel, die außerhalb ihrer Dimensionierung liegen, als Ergebnis den nächstgelegenen Winkel an. Somit lässt sich der Winkelbereich errechnen ohne das Einflüsse der Lenkbewegung des Fahrzeugs berücksichtigt werden, da in diesen Fällen die HT ein gültiges Ergebnis ergibt.

Der maximale Winkel ϕ_{max} wurde in Abhängigkeit zur *ROI*-Position bestimmt:

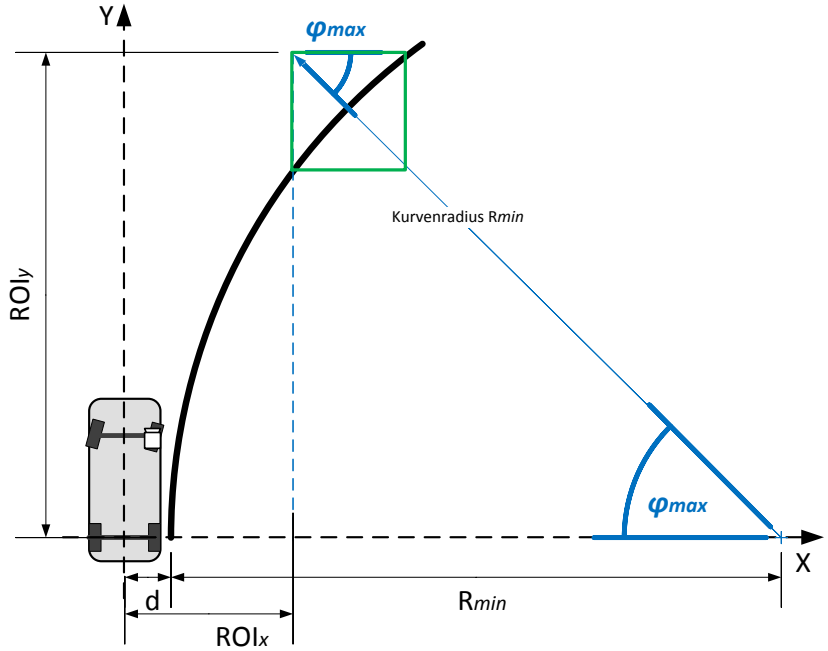

Abbildung 4.14.: Konstruktion von ϕ_{max} in Abhängigkeit von ROI_Y.

Der Winkel ϕ_{max} liegt in einem rechtwinklige Dreieck mit der Gegenkathete ROI_y und der Ankathete $R_{min} + d - ROI_y$ (vgl. Abb. 4.14).

$$\phi_{max} = tan^{-1}\left(\frac{ROI_y}{R_{min} + d - ROI_{x_{max}}}\right) \tag{4.29}$$

Die maximale ROI_x-Position $ROI_{x_{max}}$ wird durch die Breite des Kamerabilds auf der Höhe ROI_y abzüglich der Breite der ROI bestimmt (vgl. Abb. 4.11 & Gl. 4.19).

$$ROI_{x_{max}} = \frac{W(\beta, ROI_y)}{2} + o_x - ROI_{width_{cm}} \tag{4.30}$$

Die angestrebte ROI_y Position von $80cm$ mit einen Sollabstand $d = 5cm$ und einer Kameraperspektive mit der Neigung $\beta = 59°$ (vgl. Kap. 4.2.4) ergibt sich $\phi_{max} = 46.02°$:

$$ROI_{x_{max}} = \frac{W(59°, 80cm)}{2} + 7cm - 7.2cm = 27.8cm \tag{4.31}$$

$$\phi_{max} = tan^{-1}\left(\frac{80cm}{100cm + 5cm - 27.8cm}\right) = 46.02° \tag{4.32}$$

Die Hough-Ebene wurde für ϕ auf das Intervall $[-46°, +46°]$ beschränkt (vgl. Gl.4.32). Die Winkelauflösung $\Delta\phi$ wird anhand des resultierenden Bedarfs an RTL-Ressourcen diskutiert.

RTL Ressource	Transformator mit DSP	Transformator ohne DSP
Slices	210	290
LUTs	390	408
DSPs	1	0

Tabelle 4.2.: RTL-Ressourcenbedarf eines Hough-Transformators mit und ohne Nutzung eines DSP48-Slice für den Multiplizierer (vgl. Kap. 3.2).

Die Anzahl Transformatoren resultiert aus der Winkelauflösung $\Delta\phi$:

$\Delta\phi$	Transformatoren	Transformator ohne DSP			Transformator mit DSP		
		Slices	LUTs	DSP48s	Slices	LUTs	DSP48s
0.5°	185	53650	75480	0	38850	72150	185
1°	93	26970	37944	0	19530	36270	93
2°	47	13630	19176	0	9870	18330	47
3°	31	8990	12648	0	6510	12090	31
4°	24	6960	9792	0	5040	9360	24
5°	19	5510	7752	0	3990	7410	19

Tabelle 4.3.: RTL-Ressourcenbedarf der Hough-Transformation für einen Winkelbereich von ±46° in Abhängigkeit zur Winkelauflösung $\Delta\phi$. Werte in rot übersteigen die verfügbaren Ressourcen des Spartan 3A DSP1800 FPGAs (vgl. Kap. 2.3).

Das Gesamtsystem des SAV braucht ohne die Hough-Transformation ungefähr 6500 Slices, 11600 LUTs und 35 DSP48As. Folglich ist eine Winkelauflösung von $\Delta\phi = 2°$ im Hinblick auf die vorhandenen RTL-Ressourcen des Spartan 3A DSP1800 umsetzbar.

Mit sinkender Winkelauflösung $\Delta\phi$ deckt jeder Transformator einen größeren Winkelbereich ab und reduziert damit die Auflösung für die nachfolgende Spurführung. Die Abweichung wird definiert durch die prozentuale Abweichung des Ergebnisses des Spurführungsalgorithmus Pure-Pursuit mit dem Parameter ϕ zum Ergebnis mit dem Parameter $\phi + 0.5 * \Delta\phi$. Die Ergebnisse sind in Tabelle 4.4 aufgelistet.

Δ_ϕ	0.5°	1°	2°	3°	4°	5°
Abweichung	±0.34%	±0.68%	±1.38%	±2.12%	±2.88%	±3.67%

Tabelle 4.4.: Abweichung des Ergebnisses der Spurführung durch das gewählte Winkelintervall $\Delta\phi$.

Um die Abweichung der Fahrspurführung durch die Geradenapproximation zu minimieren wird eine Winkelauflösung von $\Delta\phi = 2°$ gewählt, da die hierfür benötigten RTL-Ressourcen zur Verfügung stehen.

4.3.2. Wertebereich der Lotlänge

Die Dimension von r ist von der Größe der ROI und dem Winkelbereich von ϕ abhängig und lässt sich wie folgt konstruieren:

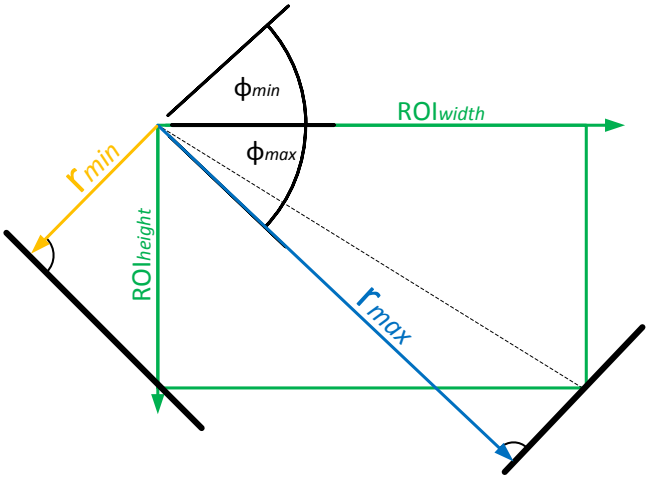

Abbildung 4.15.: Dimensionierung von r durch geometrische Konstruktion.

Die maximale Lotlänge r_{max} wurde über das rechtwinklige Dreieck bestimmt, das die Diagonale der ROI mit dem Lot im Winkel ϕ_{max} aufspannt. Die Diagonale wird durch den Satz von Pythagoras über die Höhe und Breite der ROI bestimmt:

$$r_{max} = cos(\phi_{max} - \left(tan^{-1}\left(\frac{ROI_{height}}{ROI_{width}}\right)\right)) * \sqrt{ROI^2_{height} + ROI^2_{width}} \tag{4.33}$$

Die minimale Lotlänge r_{min} wurde durch den Cosinus des Winkels $90° - \phi_{min}$ mit der Hypothenuse ROI_{height} bestimmt:

$$r_{min} = -cos(90° - |\phi_{min}|) * ROI_{height} \tag{4.34}$$

So ergeben sich für den in Kapitel 4.3.1 bestimmten Wertebereich von ϕ unter Annahme einer ROI-Größe von $50 * 50px$ folgende Grenzwerte für r:

$$r_{min} = -36px \tag{4.35}$$

$$r_{max} = 71px \tag{4.36}$$

5. Systemmodifikationen und Erweiterungen

Am Gesamtsystem des SAV wurden Modifikationen und Erweiterungen umgesetzt um die in Kapitel 4 gewonnenen Erkenntnisse zu integriert. In diesem Kapitel wird erst ein Überblick über das Gesamtsystem gegeben und anschließend im Detail auf einzelne Schwerpunkte eingegangen. So werden die Implementierung der Bluetooth-Schnittstelle für das Datalogging und die Systemparametrisierung, ein Softwareentwurf für die Spurführung sowie die Anpassungen der Fahrspurerkennung an die in Kapitel 4.3 ermittelten DiskretisierungsIntervalle beschrieben.

5.1. Systemüberblick

Das SAV basiert auf einem *Xilinx MicroBlaze* 32bit RISC Softcore Prozessor. Er führt die Software-Komponenten aus und bindet die Hardware-Beschleuniger über Bussysteme an. Es werden zuerst die beteiligten Hardware-Komponenten erläutert und anschließend die Software-Architektur dargestellt.

5.1.1. SoC Systemkomponenten

Das MicroBlaze System besteht aus folgenden IP-Cores:

- Der Xilinx **MicroBlaze** 32bit RISC Softcore Prozessor wird mit Floating Point Unit FPU instanziiert. Bedingt durch den auf dem Board verbauten 125MHz Oszillator läuft der Prozessor mit einer Taktfrequenz von 62.5MHz (vgl. [27]). Die Halbierung der Frequenz ergibt sich aus der Verwendung des DDR2-RAMs, der mit der doppelten Frequenz betrieben wird. Der MicroBlaze bietet 16kB BRAM für Instruktionen und Daten. Er hat nur einen Interrupt-Eingang. Werden mehrere Interrupt Quellen benötigt, muss ein externer Interrupt Controller eingesetzt werden.
- Ein Xilinx **XPS Timer** generiert Interrupts, die als Zeitbasis für die Scheduling und Software-Timer-Funktionalitäten des auf dem *MicroBlaze* eingesetzten *Xilkernel*-Betriebssystems dienen. Er ist über den PLB angeschlossen und bietet 2 32bit Timer.

- Der **Interrupt Controller** bündelt mehrere Interrupt Quellen und stellt diese priorisiert dem Interrupt Eingang des *MicroBlaze* zur Verfügung. Die Quelle des jeweiligen Interrupts wird über den PLB ausgelesen werden um den zugeordneten Interrupt-Handler in Software aufzurufen.
- Der verwendete **Memory Controller** bindet den externen DDR2 RAM an um diesen als Daten- und Instruktionenspeicher zu nutzen. Somit steht mehr Speicher als die internen 16*kB* BRAM des *MicroBlaze* zur Verfügung.

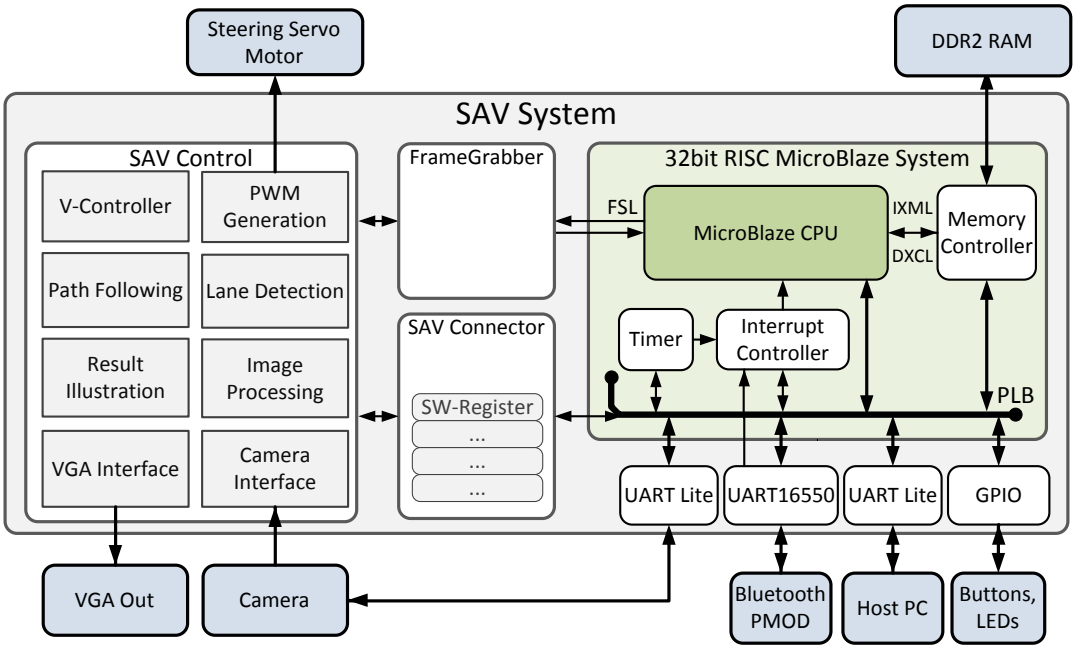

Abbildung 5.1.: Übersicht der SoC-Architektur des SAV Systems.

Weitere IPs der Peripheriekomponenten und Kommunikationsschnittstellen wurden eingesetzt:

- **XPS GPIO**: Zur Ansteuerung der Buttons, Switches und LEDs auf dem Entwicklungsboard.
- **XPS UART Lite**: Die RS232-Schnittstelle zum Host-PC wird als Standard-Out für die Konsolenausgaben der Software genutzt.
- **XPS UART 16550**: Zur Anbindung des Bluetooth Pmod.

Zur Unterstützung der Entwicklung sowie zum Testen wurden folgende IPs ins System eingebracht:

- **MicroBlaze Debug Module** (MDM): Zur Unterstützung von Software Debugging auf dem *MicroBlaze* über das JTAG Interface.
- **Xilinx ChipScope**: Zur Analyse von Signalen innerhalb des Systems wird ein *Integrated Logic Analyzer* (ILA) eingesetzt.

Die Hardware-Beschleuniger des SAV sind in zwei IP Cores strukturiert:

SAV_Control

Das *SAV_CONTROL* setzt das Kamera-Interface, die Fahrspurerkennung, die Spurführung und die Geschwindigkeitsregelung um. Es besteht sowohl aus VHDL-Blöcken als auch als *Xilinx System-Generator*-Blöcken. Diese sind in einem eigenständigen IP gebündelt, um die Integration in die SoC-Plattform des SAVs zu erleichtern. Er wird im nachfolgenden Kapitel 5.1.2 erläutert.

SAV_Connector

Das *SAV_CONTROL* hat keine Busanbindung um unabhängig vom Bussystem zu sein. Die Anbindung geschieht daher ausgelagert im *SAV_CONNECTOR*, der diese für den PLB umsetzt. Der IP bietet 32 je 32*bit* breite Software Register zum Datenaustausch zwischen dem *SAV_CONTROL*-IP und der auf dem Microblaze ausgeführten Software und synchronisiert den Zugriff auf diese zwischen den Taktdomänen des pixeltaktgetriebenen *SAV_CONTROL* mit 13.5*MHz* Pixeltakt und dem *MicroBlaze*-System mit 62.5*MHz* Taktfrequenz.

5.1.2. RTL-Modelle für Fahrspurerkennung und Spurführung

Der IP-Core enthält alle RTL-Modelle für die Fahrspurerkennung und die Spurführung und ist in fünf Module unterteilt:

Kamera Interface

Das Kamera Interface synchronisiert den Pixelstrom und konvertiert die Farbdarstellung von *YCrCb* in RGB-Graustufen (vgl. Kap. 3.1.1).

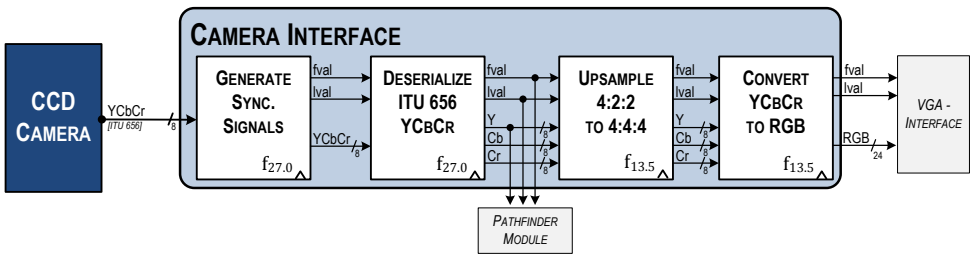

Abbildung 5.2.: Struktur des Kamera Interfaces [12].

Pathfinder Modul

Die Entity „*PATHFINDER*“ ist in einem System-Generator-Modell modelliert und wird als NGC-Netzliste in die Top-Level-Entity eingebunden:

- Die Bildvorverarbeitung findet im Modul „***IMAGE_PROCESSING***“ statt und beinhaltet die adaptive Binarisierung, zwei Erosionsfilter mit einer 3*3 Faltungsmatrix, zwei Erosionsfilter mit einer horizontalen 1*3 Faltungsmatrix sowie die projektive Transformation (vgl. Kap. 3.1.1).
- Die Hough-Transformation ist im Funktionsblock „***LANE_DETECTION***“ umgesetzt worden. Dieser enthält zudem die horizontale Korrektur der dynamischen ROI. Die Hough-Transformation wird von einem Zustandsautomaten gesteuert, der abhängig von der Y-Koordinate des Pixelstroms die HT zurücksetzt (*HT_RESET*), Vordergrundpixel aus der ROI an sie weitergibt und die abschließende Maximaberechnung anstößt (*HT_EVAL*) (vgl. Abb. 5.3). Anschließend signalisiert das Modul mit dem Signal „*HT_DONE*“, dass das Ergebnis der Hough-Transformation vorliegt. Das Ergebnis für den Winkel ϕ ist hierbei nicht in Grad angegeben sondern mit der ID des entsprechenden Transformators. Diese ID ist gleichzeitig der Index für die Look-Up-Tables der trigonometrischen Funktionen in der nachfolgenden Spurführung [22].

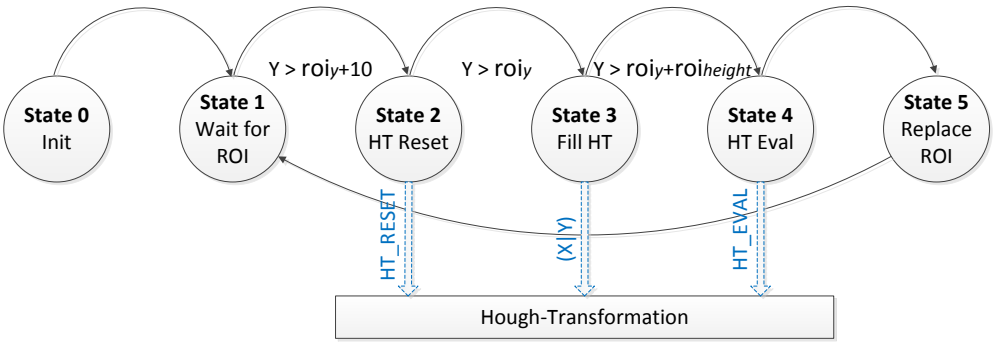

Abbildung 5.3.: Zustandsautomat zur Steuerung der Hough-Transformation in Abhängigkeit der Y-Koordinate des Pixelstroms.

- „***PATH_FOLLOWING***“ enthält den Multizyklusdatenpfad für die Spurführungsalgorithmen (vgl. Kap. 3.3). Es wurden *Pure Pursuit*, *Follow-the-Carrot* und deren Kombination mit einem PD-Regler implementiert und können zur Laufzeit ausgewählt werden. Die Aktualisierung des Lenkwinkels α wird durch das Signal *PF_DONE* angezeigt.
- Die Ergebnisdarstellung „***RESULT_ILLUSTRATION***“ besteht aus zwei Funktionsblöcken: Der Wiederherstellung der Pixelreihenfolge nach der projektiven Transformation und der Einblendung von Systeminformationen in das Kamerabild. Der Block *IMAGE_BUFFER* speichert in einem

Dual-Ported-RAM die eingehenden Pixel des Pixelstroms an ihren ursprünglichen Koordinaten vor der PT um es so, um ein Bild verzögert, in der korrekten Reihenfolge auszulesen. Anschließend wird im Block *RESULT_OVERLAY* das Kamerabild um die Anzeige von Ergebnissen der vorangegangen Hough-Transformation und der Spurführung erweitert (vgl. Kap. 3.5). Dies wird durch eine Konvertierung des Pixelstroms in 24bit RGB-Farbdarstellung erreicht, um die berechneten Pixel durch andere Farbwerte zu überschreiben.

- Die Berechnung des PWM-Signals übernimmt der Block „***PWM_GENERATION***“. Der Lenkwinkel α wird auf den Stellbereich des Lenkservos von ±20° begrenzt und in den Duty Cycle der PWM zur Ansteuerung des Servo umgerechnet (vgl. Kap. 3.6).

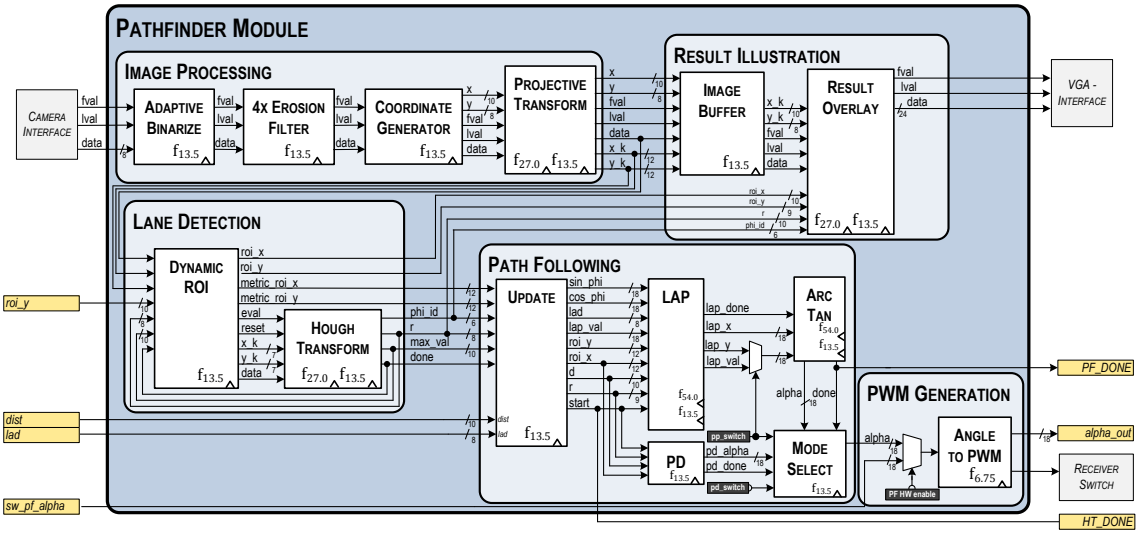

Abbildung 5.4.: *PATHFINDER* Modul mit Funktionsblöcken *IMAGE_PROCESSING, LANE_DETECTION, PATH_FOLLOWING, RESULT_ILLUSTRATION und PWM_GENERATION [12].*

Geschwindigkeitsregelung

Das Modul ***V_CONTROL*** ist als VHDL-Block umgesetzt worden und enthält die Geschwindigkeitsregelung des SAV. Die aktuelle Fahrzeuggeschwindigkeit wird durch die Pulse der Hall-Sensoren des Brushless Motors ermittelt. Durch den Durchmesser der Antriebsräder des Fahrzeugs kann, unter der Annahme, dass es keinen Schlupf gibt, die Fahrzeuggeschwindigkeit errechnet werden, da die Übersetzung zwischen Motor und Abtriebsrädern konstant ist [4].

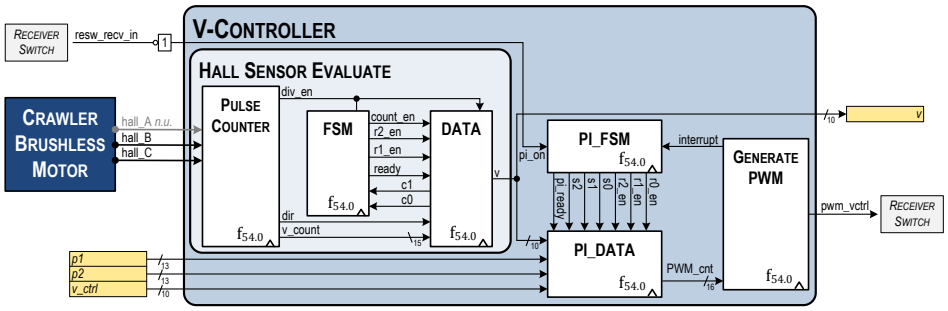

Abbildung 5.5.: Struktur des *V_CONTROL* Moduls [12].

Receiver-Controller

Zur Umsetzung der Anforderung aus dem Carolo-Cup-Reglement bezüglich der Unterbrechbarkeit des autonomen Fahrbetriebs überwacht das Modul ***RECV_CONTROL*** den korrekten Empfang des Signals der Fernbedienung (vgl. Kap. 3.6). Ist dies nicht der Fall, werden die PWM-Ausgänge zum Lenkservo und zum Motor unterbrochen. Damit das Fahrzeug in diesem Fall manuell gesteuert werden kann, werden die PWM-Steuersignale der Fernbedienung vom Funkempfänger der Fahrzeugplattform an Motor und Lenkservo weitergegeben.

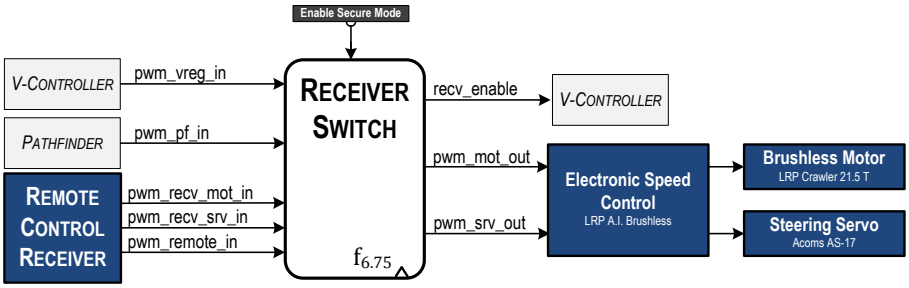

Abbildung 5.6.: Struktur des *RECEIVER_SWITCH* Moduls [12].

VGA Interface

Die Ausgabe des Kamerabildes über den VGA-Ausgang übernimmt das Modul ***VGA_INTERFACE***. Es generiert die Synchronisationssignale *HSYNC* und *VSYNC* aus den *FVAL* und *FVAL*-Steuersignalen des Pixelstroms und hebt die Auflösung auf die für den VGA-Standard nötigen $640 * 480px$ an, indem es die Bildzeilen verdoppelt und somit das Interlacing der Kamera ausgleicht.

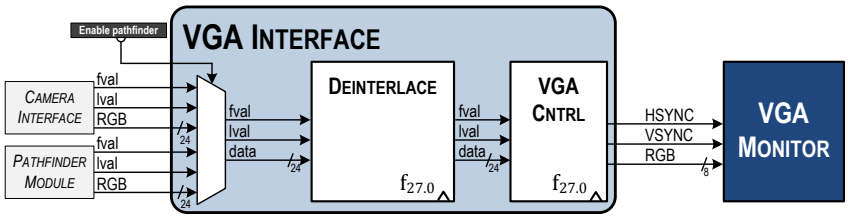

Abbildung 5.7.: Struktur des *VGA_INTERFACE* Moduls [12].

5.1.3. Schnittstellen zwischen RTL-Modellen und Software

Die Schnittstellen zwischen Bildverarbeitungspipeline und dem *MicroBlaze*-System besteht aus den Software-Registern des *SAV_CONNECTOR* IPs und Interrupt-Quellen aus den RTL-Modellen.

Interrupt Quellen der RTL-Modelle

Am Interrupt Controller IP sind vier Interrupt-Quellen angeschlossen, die Zustandsänderungen der RTL-Modelle signalisieren. Dies sind der Timer-Interrupt für die Ticks des *Xilkernel*, das Ende der Geradenerkennung durch die Hough-Transformation, das Ende der Spurführung sowie das Empfangen von Daten über das Bluetooth-Interface (siehe Tabelle 5.1). Die Priorisierung der Interrupt-Quellen greift nur, wenn zwei Interrupts zeitgleich anliegen. Ein niedrigpriorisierter Interrupt-Handler kann vom *Xilkernel* nicht durch einen höherpriorisierten unterbrochen werden.

Bit	Signal Name	Quell-IP	Beschreibung
3	Timer	XPS_TIMER	Timer Ticks für den Xilkernel
2	HT_DONE	SAV_CONTROL	Hough-Transformationsergebnis liegt vor
1	PF_DONE	SAV_CONTROL	Ergebnis der Spurführung liegt vor
0	T_RECEIVE	PMOD_UART	Daten vom Bluetooth-Interface liegen vor

Tabelle 5.1.: Belegung des Eingangsvektors des Interrupt Controllers.

Die höchste Priorität erhält der Interrupt des Timers, damit die Software-Timer und Scheduling-Funktionalitäten des Xilkernels gewährleistet sind. Das Signal für den Abschluss der Fahrspurerkennung *HT_DONE* erhält die zweithöchste Priorität, damit die Spurführung in Software direkt angestoßen wird. Der Abschluss der RTL-Spurführung *PF_DONE* ist niedriger priorisiert, da diese für den Fahrbetrieb keine

unmittelbare Relevanz hat und lediglich das Senden einer Datalogging-Nachricht mit den Ergebnissen der Spurführung anstößt.

Das Empfangen von Konfigurationsdaten über das Bluetooth-Interface erhält die niedrigste Priorität, da es zur Laufzeit nur sporadisch und unregelmäßig eingesetzt wird und somit keine Echtzeitanforderung besteht.

Software-Register Belegung

Der *SAV_CONTROL*-IP bezieht die folgenden Parameter aus Software-Registern:

Parameter	Format	Einheit	Beschreibung
OFFSET_X	S11.0	*px*	Verschiebung des Bildes in X
OFFSET_Y	11.0	*px*	Verschiebung des Bildes in Y
ROI_X	S10.0	*cm*	Initiale X Position der ROI
ROI_Y	U10.0	*cm*	Y Position der ROI
ROI_WIDTH	U10.0	*px*	Breite der ROI
ROI_HEIGHT	U10.0	*px*	Höhe der ROI
V_SOLL	S10.0	*cm/s*	Soll-Geschwindigkeit des Fahrzeugs
V_PARAM1	S13.8	-	Parameter der Geschwindigkeitsregelung
V_PARAM2	S13.8	-	Parameter der Geschwindigkeitsregelung
D_SOLL	U10.0	*cm*	Soll-Abstand zur Fahrspurmarkierung
LAD	U8.0	*cm*	„Look-Ahead-Distance“
LAD_VAL	U16.2	*cm*	Ergebnis von $\frac{LAD^2}{2*L}$.
THRESHOLD	U8.0	-	Schwellwert für die Binarisierung bei deaktivierten Adaption
HT_VALID_THRESHOLD	U10.0	-	Grenzwert für ein gültiges HT-Ergebnis
SW_STATUS_REGISTER	Status-Flags der Software		
SW_CONTROL_REGISTER	Flags zum Aktivieren von Hardware-Funktionen		

Tabelle 5.2.: Parameter des SAV_CONTROL-IPs mit Angabe des Q-Formats der Festkomma-Werte.

Besonderheit sind das *SW_CONTROL_REGISTER*, über dessen Wert einzelne Funktionsblöcke im RTL-Modell aktiviert werden, sowie das *SW_STATUS_REGISTER* das Zustände der Software überträgt. So wird beispielsweise durch das Flag „*SW_PF_DONE*“ signalisiert, dass die Berechnung der Spurführung in der Software abgeschlossen ist und die Ergebnisse in den entsprechenden Registern vorliegen,

so dass das RTL-Modell den Lenkwinkel aktualisieren kann. Eine vollständige Beschreibung dieser Register befindet sich im Anhang A.4.

Das *SAV_CONTROL* gibt folgende Werte aus:

Parameter	Format	Einheit	Beschreibung
LAP_X	S16.2	*cm*	Position des LAP in X
LAP_Y	11.0	*cm*	Position des LAP in Y
V_IST	S9.0	cm/s	Geschwindigkeit des Fahrzeugs
ALPHA_IST	S16.10	°	Lenkwinkel
ROI_X_IST	S16.7	*cm*	X Position der ROI
PHI_ID	U8.0	-	ID des HT Transformators
PHI	S9.0	°	Winkel ϕ aus der HT
R	S8.0	*px*	Lotlänge r aus der HT
HT_MAX_VAL	U16.0	-	Maxima Wert der HT
HT_MAX_VAL	U16.0	-	Maxima Wert der HT
COUNT	U16.0	*px*	Anzahl Vordergrundpixel in der ROI
HW_STATUS_REGISTER	Status Flags der Hardware		

Tabelle 5.3.: Ausgänge des SAV_CONTROL-IPs mit Angabe des Q-Formats der Festkomma-Werte.

5.1.4. Software-Architektur des SAV

Die Architektur der Software nutzt die POSIX-Unterstüzung des als Betriebssystem eingesetzten *Xilkernel* um jede Aufgabe in einem separaten Thread abzuarbeiten, der wiederum in einem festen Intervall ein Flag abfragt, ob seine Aktion ausgeführt werden soll [29]. Eine Synchronisierung der einzelnen Threads ist hierbei nicht nötig, da keine Überschneidungen zwischen ihnen bestehen. Auch der Zugriff auf die Flags muss nicht geschützt werden, da die Threads jeweils nur das Flag löschen und nicht schreiben.

Scheduling und Threads

Der Scheduler des *Xilkernel* bietet die Optionen mit Prioritäten oder zeitscheibenbasiert nach dem Round-Robin Verfahren zu arbeiten [29]. Im Fall der Priorisierung werden Threads gleicher Priorität unter sich mit Round-Robin betrieben. Im SAV wird der Scheduler mit Prioritäten eingesetzt, da die zu verwaltenden Threads eine Priorisierung hinsichtlich der Berechnung des Spurführungsalgorithmus aufweisen.

Main-Thread

Der Main-Thread fragt die Hardware-Taster in einem festen Intervall von 100*ms* ab und führt im Falle eines entsprechenden Tastendrucks die folgenden Aktionen aus:

- Button 1: Startet den Frame-Grabber (vgl. Kap. 3.4)
- Button 2: Deaktiviert das Bluetooth-Interface.
- Button 3: Aktiviert das Bluetooth-Interface.
- Button 4: Gibt die Werte aller Software-Register über die RS232 Schnittstelle aus.

Pathfinder Thread

Fragt das System Flag SAV_FLAG_DOPF in einem Intervall von 2*ms* ab und berechnet die Spurführung nach Pure-Pursuit (vgl. Abb. 5.8).

Remote Logging Thread

Formatiert und sendet die Datalogging-Nachrichten nachdem die Spurführungsalgorithmik für ein Bild abgeschlossen wurde. Der Thread pollt das System Flag *SAV_FLAG_LOGTOSEND* in einem Intervall von 3*ms*.

Remote Parsing Thread

Fragt in einem Intervall von 100*ms* den Füllstand des Empfangspuffers des Bluetooth-Interfaces ab und ruft bei vorliegenden Daten die Verarbeitungsmethode *remote_config_handleByte()* auf, die das Kommunikationsprotokoll abhandelt (vgl. Kap. 5.2.2).

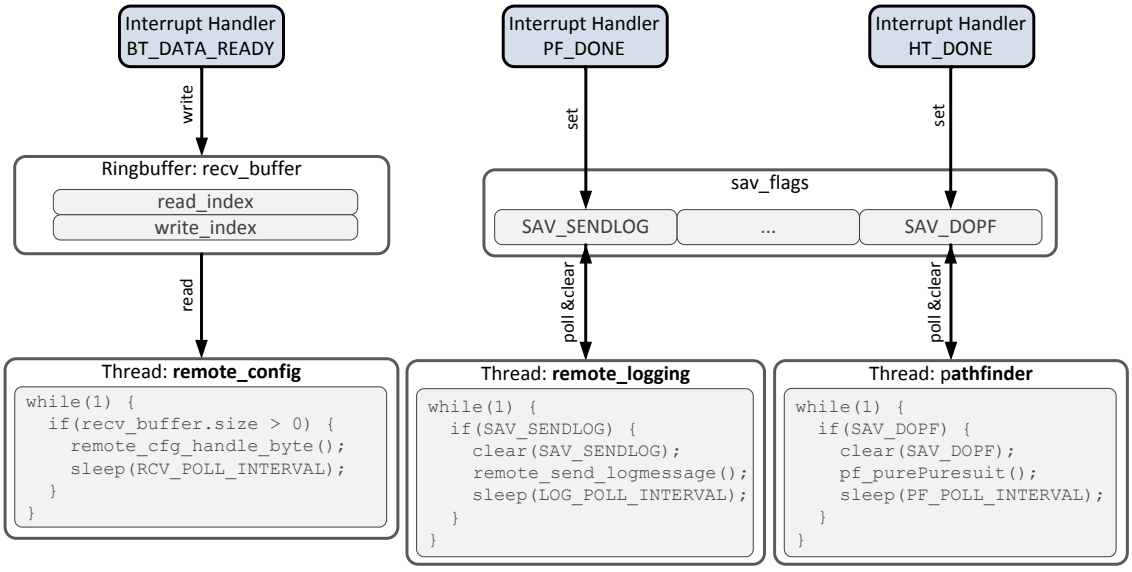

Abbildung 5.8.: Aufbau der Software mit parallelen *pthreads* und ihre Auslösebedingungen.

5.2. Datalogging und Online-Systemparametrisierung

Das Aufzeichnen von Mess- und Stellgrößen aus dem Fahrbetrieb stellt die Grundlage für eine Auswertung des Fahrverhaltens dar. Dies wird durch eine Bluetooth-Funkverbindung zu einem Host-Computer realisiert.

5.2.1. Bluetooth-Schnittstelle

Bluetooth ermöglicht durch die Verwendung eines „*Serial Port Profile*“ (SPP) die Emulation einer seriellen Schnittstelle. Somit kann die Funkverbindung wie eine serielle Schnittstelle angesprochen werden was die Programmierung vereinfacht. Es wird ein *Digilent PmodBT2* Bluetooth Modul, basierend auf einem *Roving Networks RN42* Chip, verwendet [7][19].

Die Anbindung an den *MicroBlaze* erfolgt über den PLB-Bus durch einen *XPS UART 16550* IP der es, im Vergleich zum *XPS UART Lite* IP, erlaubt die Parameter der seriellen Verbindung zur Laufzeit über den Software-Treiber anzupassen. Die Verbindung wird mit einer Geschwindigkeit von 115200*baud* bei einem Start-, einem Stopp- und ohne Paritätsbit betrieben.

Der Empfang von Daten erfolgt interruptbasiert gesteuert, der Interrupt-Handler schreibt die empfangenen Daten in einen Ringpuffer um die Laufzeit des Interrupt-Handlers zu reduzieren. Die Verarbeitung der Daten wird separat abgehandelt (vgl. Kap. 5.1.4).

5.2.2. Protokoll des Dataloggings und der Systemparametrisierung

Die Kommunikation findet über ASCII-Zeichenketten statt und orientiert sich am CSV-Format um eine Auswertung in Tabellenkalkulationsprogrammen zu erleichtern. Eine Nachricht setzt sich aus drei Elementen zusammen, getrennt von einem Semikolon:

- **Präambel** : Zur Erkennung des Anfangs einer neuen Log-Nachricht seitens des Empfängers ist die Präambel auf *LOG*.
- **Länge** : Die Gesamtlänge der Nachricht in Bytes. Das Element ist immer 3 Dezimalstellen lang, nicht genutzte Stellen werden mit vorangestellten Nullen aufgefüllt.
- **Daten** : Es können beliebig viele dezimale Datenelemente folgen. Die Reihenfolge und Anzahl der Datenfelder ist dabei auf Empfänger- und Senderseite implementiert, da sie vom Empfänger dazu genutzt wird, die Felder zuzuordnen.

Es werden die X- und Y-Komponenten des LAP, die horizontale Position der ROI ROI_X, die Fahrzeuggeschwindigkeit v, die Ergebnisse der Hough-Transformation r und ϕ sowie der Lenkwinkel α übertragen:

Präambel	Länge	Daten							
LOG	000	STATUS_REG	LAP_Y	LAP_X	ROI_X	v	ϕ	r	α

Tabelle 5.4.: Aufbau der Datalogging-Nachrichten.

Die Parametrisierungs-Nachrichten sind äquivalent zu den Datalogging-Nachrichten aufgebaut, mit dem Unterschied, dass die Präambel als „*CFG*“ definiert ist.

Präambel	Länge	Daten						
CFG	000	CONFIG_REG	d	ROI_X	ROI_Y	ROI_{width}	ROI_{height}	LAD

Tabelle 5.5.: Aufbau einer Parametrisierungs-Nachricht.

5.2.3. Java Software des Host-PCs

Eine grafische Java-Applikation empfängt die Logging-Nachrichten, zeichnet diese zur späteren Auswertung auf und sendet Parametrisierungs-Nachrichten an das SAV. Die Kommunikation über die Bluetooth-Schnittstelle wird durch die *RXTX*-Bibliothek zur Ansteuerung von seriellen und parallelen Schnittstellen unter Java [20] umgesetzt, da das Bluetooth-Interface im SPP betrieben wird und daher wie eine serielle Schnittstelle angesprochen wird (vgl. Kap. 5.2.1).

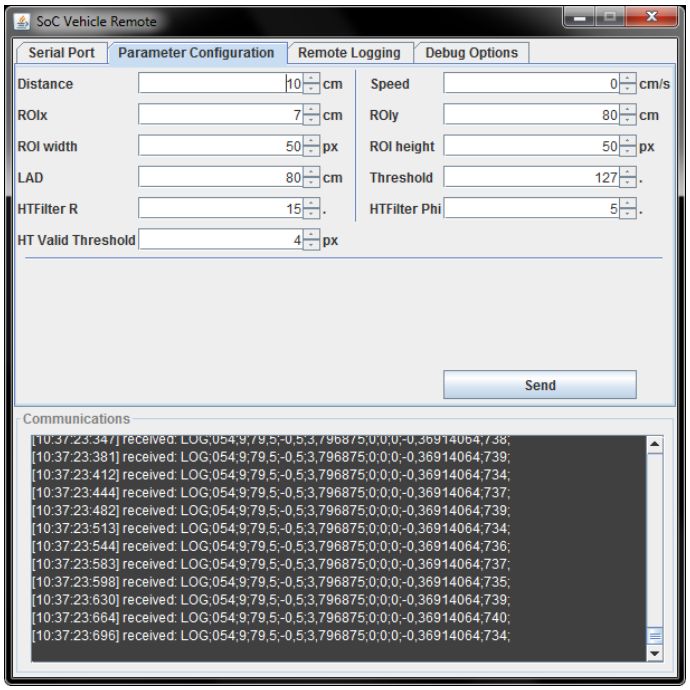

Abbildung 5.9.: Screenshot des SAV Remote Tools zum Aufzeichnen von Logging-Daten sowie zur Online Systemparametrisierung.

Die Software besteht aus folgenden Klassen (vgl. Abb. 5.10):

- *RemoteController*: Hält die Instanzen auf die weiteren Komponenten und verwaltet die Zustände der Software.
- *RemoteUI*: Die Swing-basierte, grafische Benutzeroberfläche mit Ereignisbehandlung für alle Eingabe-Elemente.
- *SerialPortHandler*: Anbindung der seriellen Schnittstelle des Bluetooth Interfaces.
- *SerialPortListener*: Event-Listener für das Empfangen von Logging-Daten über die serielle Schnittstelle.
- *ConfigParser*: Liest die Informationen über die Nachrichtenformate aus der XML-Konfigurationsdatei aus.
- *LogFileWriter*: Schreibt empfangene Logging-Daten in eine Datei.
- *RemoteLoggingFrame*: Repräsentiert eine Logging-Nachricht und prüft das Nachrichtenformat der empfangenen Daten.
- *RemoteConfigFrame*: Repräsentiert eine Konfigurations-Nachricht.

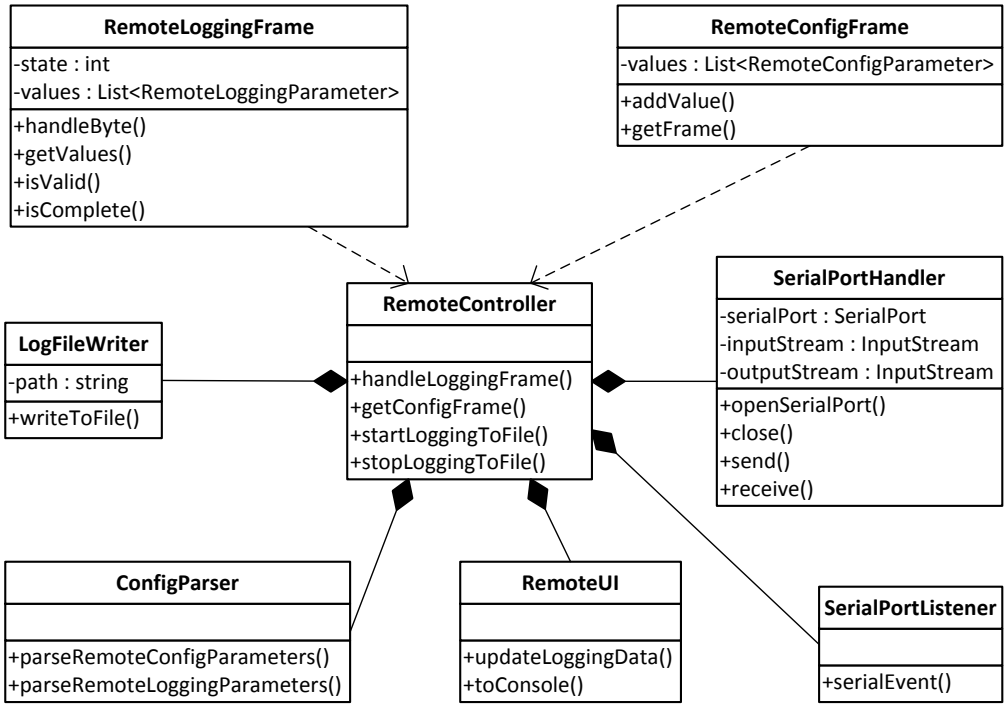

Abbildung 5.10.: Klassendiagramm des Java Remote Tools.

Die Belegung der Datenfelder der Datalogging- und Parametrisierungsnachrichten wird in einer XML-Konfigurationsdatei konfiguriert und kann so dynamisch angepasst werden, ohne dass die Anwendung erneut kompiliert werden muss.

```
11:24:49:221;21.0;99.5;-0.0;4.953125;70.0;44.0;91.0;0.9277344;2.0
11:24:49:251;1.0;98.5;-14.5;4.953125;-70.0;44.0;14.0;-1.25390624;92.0
11:24:49:282;1.0;98.5;-13.5;4.953125;-71.0;46.0;25.0;-4.1484375;223.0
11:24:49:318;1.0;99.0;-13.0;4.953125;72.0;44.0;29.0;-3.5449219;172.0
11:24:49:351;1.0;99.0;-11.0;4.953125;71.0;46.0;37.0;-3.32617188;345.0
```

Listing 5.1: Ausschnitt aus einem gespeicherten Logfile. Es werden die Log-Nachrichten ohne Präambel und Länge gespeichert sowie eine Zeitmarke für den Zeitpunkt des Empfangs vorangestellt.

Die Logfiles entsprechen dem CSV-Format und können daher direkt zur Auswertung in Tabellenkalkulationsprogramme importiert werden.

5.3. Anpassung am RTL-Modell

Die Änderungen in den RTL-Modellen der Bildverarbeitungspipeline umfassen Anpassungen an der Bildvorverarbeitung, der Fahrspurerkennung und der Spurführung.

- **Bildvorverarbeitung**

  Das Modul *IMAGE_PROCESSING* wurde um einen zusätzlichen Erosionsfilter und drei horizontale Erosionsfilter ergänzt, um die Anzahl Vordergrundpixel zur Darstellung der Fahrspurmarkierung und damit die Anzahl möglicher Geradenapproximationen zu reduzieren (vgl. Kap. 4.2.3).

  Die Transformationsmatrix B der Projektiven Transformation wurde nach dem im Anhang A.3 beschriebenen Verfahren zur Kalibrierung auf die in Kapitel 4.2.4 ermittelte Kameraperspektive mit der Neigung $\beta = 59°$ bestimmt:

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & 1 \end{pmatrix} = \begin{pmatrix} 1.0000 & 3.0329 & 0.0000 \\ 0.0000 & 3.2501 & 5.0526 \\ 0.0000 & 0.0095 & 1 \end{pmatrix} \tag{5.1}$$

- **Fahrspurerkennung**

  Gemäß den Ergebnissen der Diskretisierung der Hough-Ebene in Kapitel 4.3 wurde die Anzahl der Hough-Transformatoren auf 47 erhöht und decken einen Winkelbereich von ±46° in 2°-Schritten ab.

  Die Übergabe der Y-Pixelkoordinaten der Vordergrundpixel an die Hough-Transformation wurde korrigiert um die vertikale Stauchung des Kamerabildes durch das Interlacing der Kamera auszugleichen (vgl. Kap. 3.1.1). Hierzu

werden die Y-Koordinaten mit dem Faktor 2 multipliziert um die ursprünglichen Verhältnisse wiederherzustellen:

```
x = xfix({xlUnsigned, 10, 0}, x_in);
y = xfix({xlUnsigned, 10, 0}, (y_in*2) ); % Ausgleich des
    interlacing
```

Listing 5.2: Übergabe der Pixelkoordinaten an die Hough-Transformation

- **Spurführung**

  Das RTL-Modell der Spurführung arbeitet auf metrischen Einheiten und erhält die Lotlänge r der Hough-Transformation in Pixeln. Der Umrechnungsfaktor hängt von der Kameraperspektive und den daraus resultierenden Verhältnissen von Pixeln zu abgebildeter Fläche ab. r wird von der Spurführung im Hilfskoordinatensystem genutzt, so dass r immer parallel zur X-Achse liegt.

$$r_{metric} = r_{px} * \frac{w_o}{P_x} \tag{5.2}$$

  Entsprechend erfolgt die Umrechnung nur in X-Richtung durch die Multiplikation mit dem Verhältnis von metrischer Bildbreite des Kamerabildes w_o (vgl. Kap. 4.2.1) und der Anzahl Pixel der Kamera in X-Richtung (vgl. Gl. 5.2).

5.3.1. Verbrauch von RTL-Ressourcen

Der Bedarf an RTL-Ressourcen des Modells ist insbesondere durch die Vergrößerung des Winkelbereichs der Hough-Ebene gestiegen. Um diesen Bedarf zu kompensieren wurde der zweite *MicroBlaze*-Prozessor aus dem System entfernt [12]. Die Multiplizierer der Hough-Transformatoren wurden voranging mit DSP48A-Slices umgesetzt um den Verbrauch an LUTs zu reduzieren (vgl. Kap. 4.3.1). Fünf Multiplizierer mussten in Verhaltenslogik umgesetzt werden, da nicht mehr genug DSP-Splices zur Verfügung standen.

RTL Ressource	Verfügbar	Verbraucht	relativer Verbrauch	Änderung
Slices	33280	26134	79%	+18%
4 Input LUTs	33280	31668	95%	+29%
Occupied Slices	16640	16555	99%	+16%
DCMs	8	3	37%	±0%
DSP48As	84	84	100%	+70%
BRAM	84	53	63%	−22%

Tabelle 5.6.: RTL-Ressourcen-Bedarf des SAV.

Eine Auflistung der Ressourcen der einzelnen Module und IPs befindet sich im Anhang A.6.

5.4. Software-Entwurf des Spurführungsalgorithmus

Die Spurführung wurde als Software-Entwurf in der Programmiersprache C entwickelt und wird durch den Interrupt nach Abschluss der Fahrspurapproximation angestoßen (vgl. Kap. 5.1.4). Als Datentyp wurde *float* gewählt, da die Floating Point Unit des *MicroBlaze* Prozessors mit „Single Precision“ arbeitet [28]:

```
int r = (int)platform_getR(); // r in px
int d = platform_getD(); // d in cm
int phi = platform_getPhi(); // phi in degree
float roi_x = platform_getROIX(); // roi_x in cm
int roi_y = platform_getROIY(); // roi_y in cm
int lad = platform_getLAD(); // lad in cm

float roi_x_h = 0;
float lap_x_h = 0, lap_y_h = 0;
float lap_x = 0, lap_y = 0;
float alpha = 0;
float r_factor = (float)SAV_CAM_VISIBLE_X/(float)SAV_CAM_RESOLUTION_X;

// precalculations
float phi_rad = phi*(PI/180);
float sin_phi = sin(phi_rad);
float cos_phi = cos(phi_rad);
float lad_pwr2 = lad*lad;
float r_metric = (float)r)*((float)r_factor;

//1. rotate ROI CW by phi (Hilfskoordinatensystem)
roi_x_h = ( cos_phi*roi_x + sin_phi*roi_y );

//2. lap_h
lap_x_h = roi_x_h + r_metric - d;
lap_y_h = sqrt( lad_pwr2 - lap_x_h*lap_x_h );

//3. rotate LAP CCW by phi (to Fahrzeugkoordinatensystem)
lap_x = ( cos_phi*lap_x_h + -sin_phi*lap_y_h );
lap_y = ( sin_phi*lap_x_h + cos_phi*lap_y_h );

//4. PP algorithm
alpha = atan( (2*lap_x*SAV_CONSTANT_L)/lad_pwr2 );
```

Listing 5.3: Software-Entwurf der Spurführung

Die Umsetzung wird im Folgenden auf Laufzeitverhalten untersucht und mit der RTL-Lösung verglichen.

5.4.1. Laufzeitverhalten

Die Gesamtlatenz vom Puls des Interrupts bis zum Ende der Berechnung des Spurführungsalgorithmus setzt sich aus vier Abschnitten zusammen:

- Die **Latenz zum Aufruf des Interrupt Handlers**: 6.4μs (vgl. Abb. B.1)
- Die **Laufzeit des Interrupt Handlers**: 5.7μs (vgl. Abb. B.2)
- Das Polling-Intervall: 2ms (vgl. Kap. 5.1.4)
- Die **Laufzeit des Spurführungsalgorithmus**.

Die Berechnungszeit des Algorithmus wird hinsichtlich der Nutzung der Floating-Point-Unit (FPU) und der Compileroptimierung untersucht und mit einem Oszilloskop an GPIO-Pins gemessen:

Optimierung	O0	O2	Os
ohne FPU	1.61ms	1.31ms	1.31ms
mit FPU	1.45ms	1.21ms	1.21ms

Tabelle 5.7.: Berechnungszeit von Pure-Pursuit in Abhängigkeit der Compileroptimierung und der Nutzung der FPU.

Die Verwendung der FPU mit Nutzung der Optimierung $O2$ resultiert in der schnellsten Berechnungszeit von 1.21ms. Somit beträgt die maximale Gesamtlaufzeit:

$$6.4\mu s + 5.7\mu s + 2000\mu s + 1210\mu s = 3222.1\mu s \tag{5.3}$$

Die Laufzeit der RTL-Lösung wurde mit dem Xilinx Tool *ChipScope* gemessen und beträgt 206 Takte bei 62.5MHz, beziehungsweise 3.296μs, und ist dabei um den Faktor 1000 schneller als die Software-Variante:

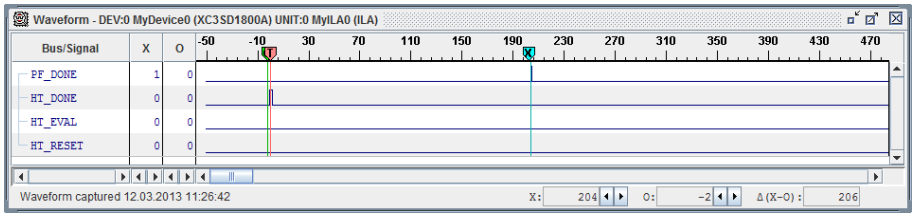

Abbildung 5.11.: Laufzeit der Berechnung der RTL-Lösung des Spurführungsalgorithmus „Pure-Pursuit“ (Abtastfrequenz 62.5MHz)

Bei der Bildwiederholungsrate von 30Hz stehen pro Kamerabild 33.3ms zur Verfügung (vgl. Kap. 3.1.1). Somit sind beide Varianten schnell genug um die Spurführung abzuschließen bevor das Ergebnis der Fahrspurapproximation des nachfolgenden Kamerabildes abgeschlossen ist.

5.4.2. Genauigkeit der Berechnung

Zur Bestimmung der Genauigkeit werden sowohl die Softwarelösung als auch die RTL-Variante mit einer *Matlab*-Implementierung als Referenz verglichen. Für die Vergleichsmessung wurden LAD und ROI_y mit $80cm$, d mit $10cm$ und ROI_x mit $7cm$ angenommen. Die Ergebnisse sind in Tabelle 5.8 festgehalten:

HT-Ergebnis	SW	SW (12bit)	RTL	Matlab
$\phi = 0°, r = 20px$	$-0.043°$	-0.254 (490%)	$\alpha = -0.566°$ (1216%)	$\alpha = -0.043°$
$\phi = 20°, r = 46px$	$1.609°$	1.596 (1.1%)	$\alpha = 1.506°$ (6.4%)	$\alpha = 1.609°$
$\phi = 44°, r = 84px$	$4.943°$	4.912 (0.6%)	$\alpha = 4.709°$ (4.7%)	$\alpha = 4.944°$

Tabelle 5.8.: Vergleich der Ergebnisse von Software- und Hardware-Spurführung. Die Ergebnisse der Software-Lösung sind mit 32bit-Auflösung angegeben.

Die Ergebnisse der Softwarelösung mit $32bit$-Auflösung stimmen mit den Matlab-Referenzergebnissen überein. Die Genauigkeit wird jedoch durch die verwendete Bitbreite von $12bit$ für den Lenkwinkel α im Software-Register eingeschränkt, bleibt aber dennoch höher als die der RTL-Lösung.

In Testfahrten wurden der Einfluss der längeren Berechnungszeit der Software-Lösung und der höheren Genauigkeit untersucht (vgl. Kap. 6.2).

6. Messtechnische Analyse

Zur Validierung der in Kapitel 4 bestimmten Parameter sowie zur Erprobung der in Kapitel 5 beschriebenen Anpassungen des SAV werden Testfahrten durchgeführt. Im folgenden werden die Bewertung der Testdaten, die Teststecke sowie die Ergebnisse erläutert.

Die Qualität einer Testfahrt wird durch die durchschnittliche Abweichung des Abstands des Fahrzeugs, bezogen auf seinen Drehpunkt im Ursprung des Fahrzeugkoordinatensystems, zur Fahrbahnmarkierung d_{real} beschrieben. Die Sensorik des SAV ist allerdings auf das Kamerabild beschränkt und bietet keine Möglichkeit diesen Abstand direkt zu messen.

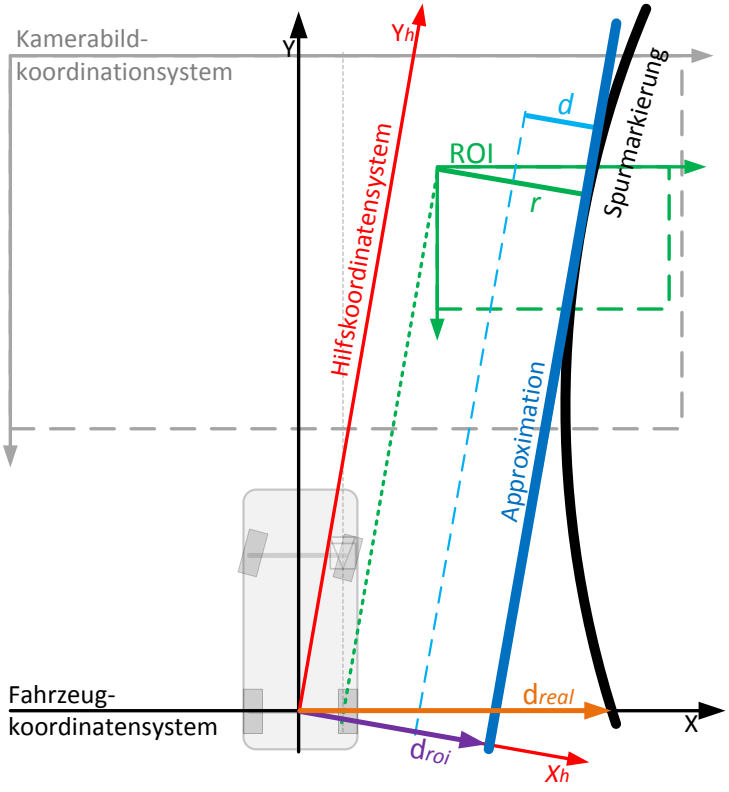

Abbildung 6.1.: Konstruktion des tatsächlichen Abstands zur Fahrspurmarkierung d_{real} und der Ersatzgröße d_{roi}.

Als Näherung wurde der Abstand d_{roi} zur Fahrspurmarkierung gewählt, der auf Höhe der ROI gemessen wird. Er setzt sich zusammen aus der Lotlänge r der Hough Transformation zum Ursprung des ROI-Koordinatensystems sowie der X-Position

der ROI im Hilfskoordinatensystem h (vgl. Kap. 4.1). d_{roi} weicht bei Kurvenfahrten von d_{real} ab, da die Approximation der Fahrspurmarkierung durch eine Gerade nur auf dem Kamerabild in der ROI basiert (siehe Abb. 6.1). Daher ist d_{roi} nicht als absolute Größe zu interpretieren und daher nur als relative Bezugsgröße für den Vergleich mehrerer Testfahrten auf der gleichen Strecke untereinander geeignet.

$$d_{roi} = ROI_{X_h} + r \tag{6.1}$$

Die X-Koordinate der ROI im Hilfskoordinatensystem ist in Gleichung 4.2 definiert und kann entsprechend umgestellt werden:

$$ROI_{X_h} = cos(\phi) * ROI_X + sin(\phi) * ROI_Y \tag{6.2}$$

Die Differenz f zwischen dem Sollabstand d_{roi} und d wurde als Güte der Fahrsituation definiert.

$$f = |d - d_{roi}| \tag{6.3}$$

Der Durchschnitt aller Werte einer Testfahrt wurde als Güte der Testfahrt genutzt. Je kleiner diese Abweichung ist, desto spurtreuer war die Fahrt. Für den Fall, dass die Fahrspur in einem Bild nicht erkannt werden sollte, wurde die vorherige Güte genommen und um einen Malus von $1cm$ erhöht. Dadurch führen diese Aussetzer zu einer negativen Bewertung, die um so stärker ins Gewicht fällt, je länger die Unterbrechung andauert.

6.1. Testumgebung

Als Teststrecke wurde ein Abschnitt der Carolo-Cup-Strecke von 2009 [6] gewählt, da diese keine Unterbrechung der Fahrspurmarkierung aufweist und mit zwei Linkskurven, eine Rechtskurve sowie drei Geraden alle Kurventypen besitzt (vgl. Abbildung 6.2).

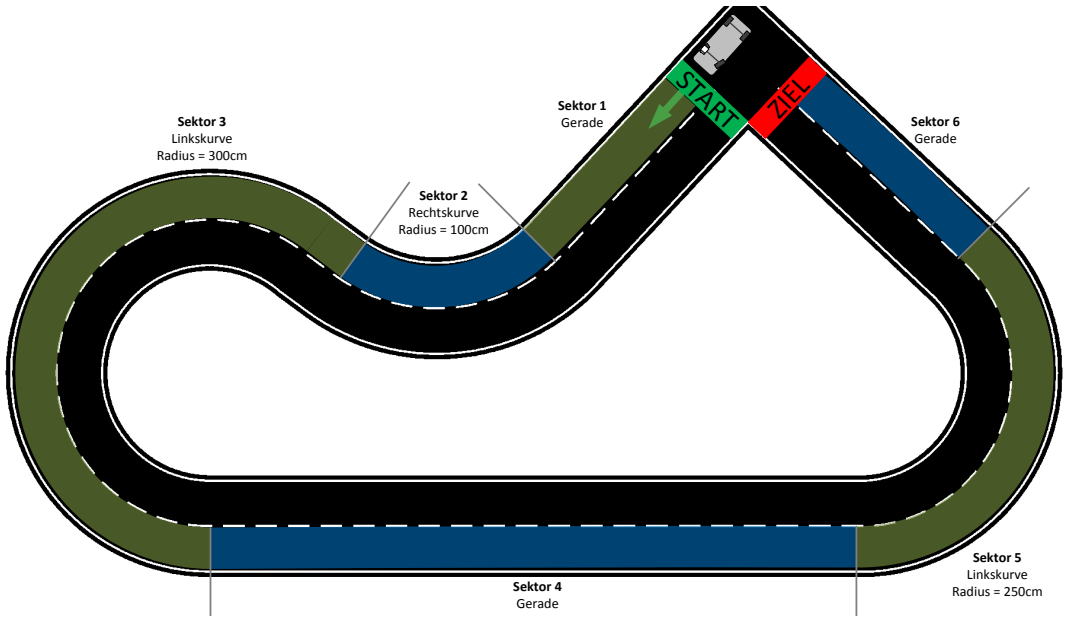

Abbildung 6.2.: Die Teststrecke wird in 6 Sektoren unterteilt.

Die Strecke wird in sechs Sektoren eingeteilt um eine Zuordnung zur Fahrsituation auf der Strecke zu gewährleisten. Die nachfolgende Grafik 6.3 zeigt die Aufzeichnung einer Testfahrt über die komplette Strecke mit Darstellung der einzelnen Sektoren. Die Winkel α und ϕ sind in Grad angegeben, ein positives α steht für einen Lenkeinschlag nach rechts. Die horizontale Position der ROI ist in Zentimetern angegeben, die Lotlänge r in Pixeln.

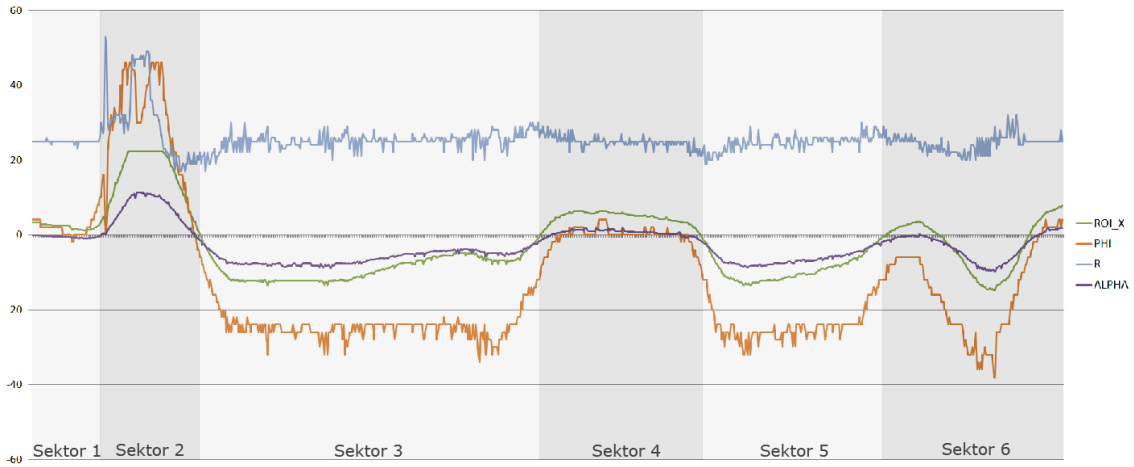

Abbildung 6.3.: Grafische Darstellung der Logdaten einer Fahrt auf der Teststrecke mit $ROI_Y = 80cm$, $LAD = 80cm$, $v = 60cm/s$ und $d = 7cm$.

Durch die dynamische horizontale Positionierung der ROI pendelt die Lotlänge r der Hough-Transformation um $25px$, der Hälfte der Breite der ROI (vgl. Kap. 3.2). Im Sektor 2, einer Rechtskurve, ist zu erkennen, dass die Ergebnisse der Geradenapproximation stark schwanken und die Position der ROI unverändert bei $+22cm$ liegt. Dies bedeutet, dass die Fahrspurmarkierung am rechten Bildrand des Kamerabildes

liegt und der ROI ihr nicht mehr folgen kann. Dadurch schwanken die Ergebnisse der Hough-Transformation.

Die Strecke hat eine Gesamtbreite von 82cm und jede Fahrspur ist 40cm breit (vgl. [6]), sos das ein Sollabstand d von 20cm zwischen der Fahrzeugmitte zur Fahrspurmarkierung gewünscht wäre. Dies ist allerdings durch die Kameraperspektive nicht umsetzbar. Daher wurden die Testfahrten mit kleineren Abständen durchgeführt um den Einfluss der Perspektive zu reduzieren und sich auf die eigentliche Spurführungsalgorithmik zu konzentrieren. (vgl. Kap. 4.2).

Eine Auflistung aller dokumentierten Testfahrten befindet sich im Anhang B.1.

6.2. Auswertung der Testfahrten

Für die Bewertung der Fahrten wurden die Größen ROI_X, ϕ und r sowie die Parameter ROI_Y und d benötigt um die Bewertungsgröße d_{roi} zu konstruieren (vgl. Kap. 6). Zusätzlich wurden die Koordinaten des Zielpunkts LAP_X und LAP_Y, die Fahrzeuggeschwindigkeit v und der Lenkwinkel α aufgezeichnet. Es wurde pro Kamerabild ein Datensatz übertragen.

Folgende Aussagen wurden untersucht:

- Vertikale Position der ROI
- Verhältnis von ROI_y zu LAD
- Einfluss der Fahrgeschwindigkeit
- Vergleich der Spurführungsimplementierungen in RTL und Software

Vertikale Positionierung der ROI

In der Parameterstudie zu Pure Pursuit in Kapitel 4.1.1 wurde eine vertikale ROI-Position von 80cm errechnet. Dieser Wert wurde messtechnisch untersucht indem das Fahrverhalten mit dem kleineren und einer größeren Distanz verglichen wird. Dabei wurde LAD jeweils identisch zu ROI_Y gewählt, Fahrzeuggeschwindigkeit auf $60\frac{cm}{s}$ sowie der Sollabstand d auf 10cm gesetzt. Das Ergebnis der Testfahrten ist in Tabelle 6.1 festgehalten:

Testfahrt	1.7	1.4	1.1
ROI_Y	70cm	80cm	90cm
Güte f	43.1	36.9	40.8

Tabelle 6.1.: Auswertung verschiedener ROI_Y Größen mit $d = 10cm$.

Eine vertikale ROI-Position von 70*cm* ist mit der gewählten Kameraperspektive nicht fahrtauglich, da das Kamerabild hier zu schmal ist und so bei Rechtskurven die Fahrspurmarkierung das Kamerabild verlässt. Die Daten einer solchen instabilen Testfahrt sind in der folgenden Grafik 6.4 dargestellt.

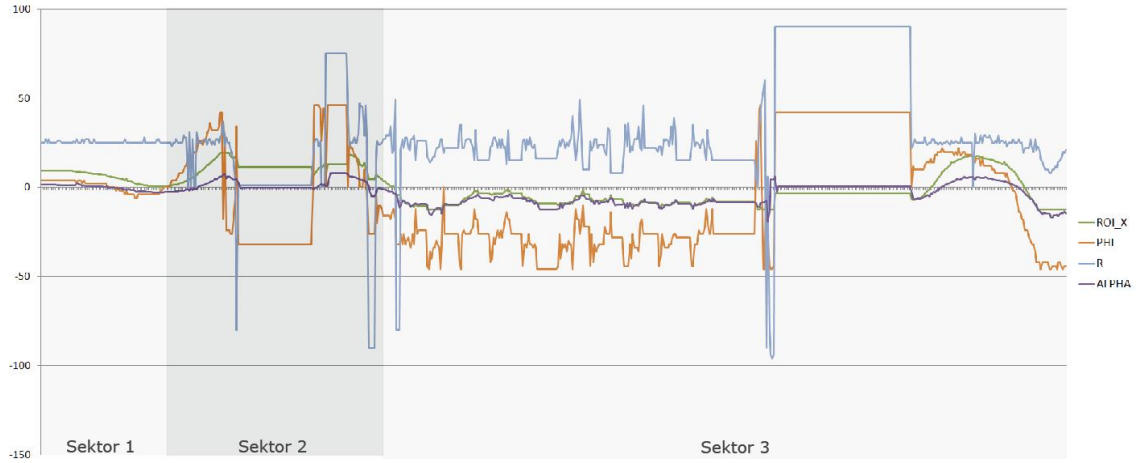

Abbildung 6.4.: Testfahrt mit $ROI_Y = 70cm$ und $d = 10cm$. In Sektor 2 kann die Fahrspur nicht mehr erkannt werden da sie den Bereich des Kamerabildes verlässt. Danach ist keine stabile Fahrt mehr möglich.

Um den Einfluss der Kameraperspektive zu minimieren wurden die Testfahrten mit einem kleineren Sollabstand $d = 7cm$ wiederholt. Dabei schneidet das Fahrzeug zwar die Fahrspurmarkierung, dies ist jedoch unabhängig von der Bewertung der Güte des Parameters.

Testfahrt	1.12	1.10	1.11
ROI_Y	70*cm*	80*cm*	90*cm*
Güte f	28.5	31.8	57.9

Tabelle 6.2.: Auswertung verschiedener ROI_Y Größen mit $d = 7cm$.

Je näher die ROI am Fahrzeug positioniert wird, desto besser wird die Spurtreue. Allerdings kann so bedingt durch die Kameraperspektive der Abstand zur Fahrspurmarkierung nicht eingehalten werden, da dann die Fahrspur verloren geht.

Verhältnis ROI_y **zu** LAD

Die Parameterstudie zum Algorithmus Pure-Pursuit ergab, dass die vertikale Position der ROI annähernd gleich der Entfernung zum Zielpunkt LAD sein sollte. Diese Aussage wurde durch Variation der LAD relativ zur ROI_y Position um 10*cm* untersucht:

ROI_Y	$LAD = ROI_Y - 10cm$	$LAD = ROI_Y$	$LAD = ROI_Y + 10cm$
$80cm$	51.4	36.9	58.7
$90cm$	47.8	40.8	99.2

Tabelle 6.3.: Auswertung des Verhältnisses zwischen ROI_Y und LAD.

Ist LAD größer ROI_y, führte dies zu Übersteuern, ist LAD kleiner ROI_y, trat Untersteuern auf. Somit konnten die Ergebnisse der Parameterstudie im Hinblick auf das Verhältnis zwischen ROI_y und LAD messtechnisch bestätigt werden (vgl. Kap. 4.1.1).

Fahrgeschwindigkeit v

Die Fahrzeuggeschwindigkeiten v wurde variiert:

Geschwindigkeit v	$40\frac{cm}{s}$	$60\frac{cm}{s}$	$80\frac{cm}{s}$	$100\frac{cm}{s}$
Güte f	36.4	36.9	37.4	54.8

Tabelle 6.4.: Auswertung des Einflusses der Fahrzeuggeschwindigkeit v auf die Abweichung der Fahrspurführung.

Mit steigender Fahrgeschwindigkeit des Fahrzeugs wurde die Spurführung ungenauer, da die Abtastrate bezogen auf die Fahrstrecke abnimmt. Bei $80\frac{cm}{s}$ kam es vereinzelt zu Ausbrüchen, ab einer Geschwindigkeit von $100\frac{cm}{s}$ konnte kein stabiler Fahrbetrieb mehr realisiert werden, da das SAV zu oft die Spur verlor, ausbrach und die Runde abgebrochen werden musste.

Ein Vergleich zweier Testfahrten mit einer niedrigen Geschwindigkeit von $40\frac{cm}{s}$ und einer Fahrt mit $100\frac{cm}{s}$ zeigte deutliche Unterschiede auf (vgl. Abbildungen 6.5 und 6.6).

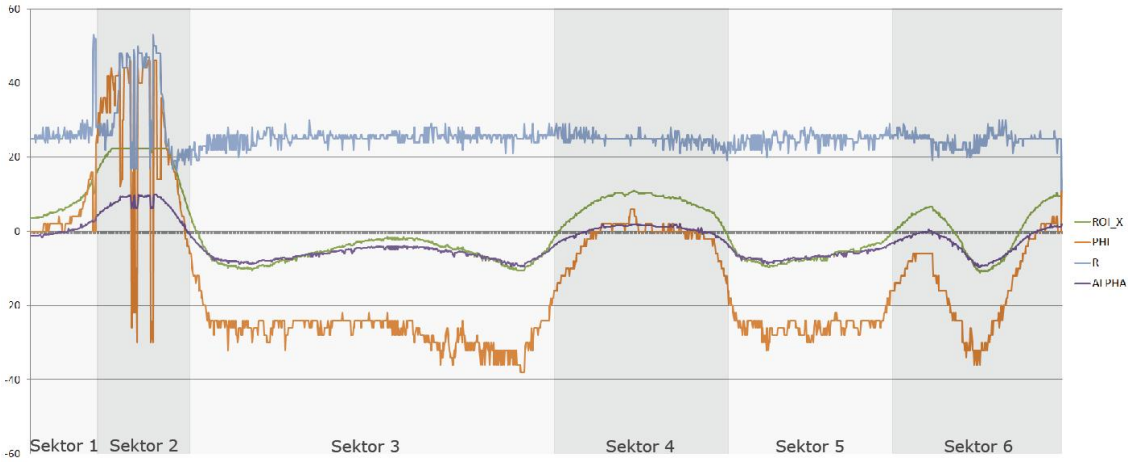

Abbildung 6.5.: Testfahrt mit Fahrzeuggeschwindigkeit $v = 40cm/s$.

Eine höhere Geschwindigkeit führte zu größeren Lenkausschlägen und damit dazu, dass die Fahrspurmarkierung nicht in der ROI des Folgebildes lag und die Spur mehr nicht erkannt wurde.

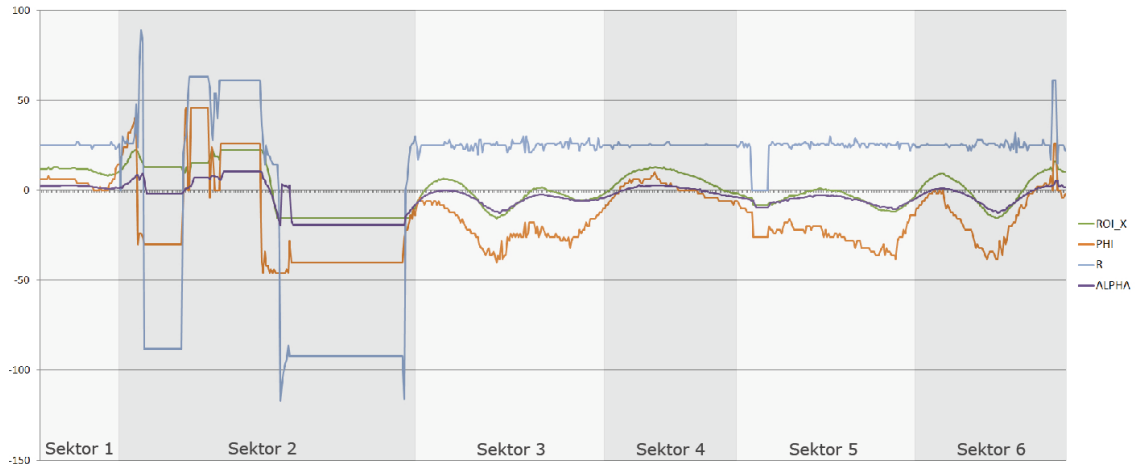

Abbildung 6.6.: Testfahrt mit Fahrzeuggeschwindigkeit $v = 100cm/s$.

Mit einer Bildrate von 30 Bildern pro Sekunde ist die Geschwindigkeit des SAV auf maximal $80\frac{cm}{s}$ begrenzt. Für eine höhere Fahrgeschwindigkeit wird eine Kamera mit höherer Bildwiederholungsrate benötigt.

Vergleich von RTL zu Software Lösung

Abschließend wurden die Implementierung der Spurführung in RTL und in Software verglichen (siehe Kap. 5.4).

v	RTL	SW
$40\frac{m}{s}$	36.4	34.8
$80\frac{m}{s}$	37.4	34.6

Tabelle 6.5.: Vergleich der RTL- und Software-Implementierung der Spurführung.

Die Softwarelösung führt zu leicht besseren Ergebnissen, die höherer Genauigkeit der Berechnung schlug sich deutlicher im Ergebnis nieder als die längere Totzeit.

6.3. Ergebnisse

Die Ergebnisse der Testfahrten lassen folgende Schlüsse zu:

- **Maximale Geschwindigkeit** v_{max} :

  Die maximale Fahrzeuggeschwindigkeit für eine erfolgreiche Spurführung liegt bei $v_{max} = 80cm/s$. Begrenzender Faktor ist die Abtastfrequenz von $30Hz$ die durch die Bildfrequenz der Kamera vorgegeben ist. Für höhere Geschwindigkeiten müsste eine Kamera mit höherer Frequenz eingesetzt werden.

- **Vertikale ROI Position** ROI_Y :

  Die besten Ergebnisse wurden mit einer vertikalen ROI-Position von $ROI_Y = 80cm$ erzielt, gemäß den Ergebnissen der Parameterstudien zu Pure-Pursuit (vgl. Kap. 4.1.1). Bei kürzeren Distanzen war das Kamerabild zu schmal, so dass die Fahrspurmarkierung verloren wurde. Bei längeren Distanzen wurde die Spurführung ungenauer und es trat Untersteuern auf (vgl. Kap. 6.2).

- **Verhältnis** ROI_Y **zu** LAD :

  Das in den Parameterstudie bestimmte Verhältnis vpn 1 : 1 zwischen der vertikalen ROI-Position und der LAD wurde bestätigt. Eine LAD kleiner ROI_Y führt zu Übersteuern, eine größere LAD zu Untersteuern (vgl. 6.2).

- **Vergleich zwischen RTL- und Software-Lösung**

  Die Softwarelösung erwies sich als minimal besser als die RTL-Lösung. Somit schlägt sich die höhere Genauigkeit deutlich auf die Fahreigenschaften aus als die längere Totzeit durch die längere Berechnungszeit (vgl. Kap. 5.4).

Einschränkungen

Die Testfahrten zeigten, dass die Spurführung des SAV bei Rechtskurven mit kleinem Radius, quer zur Fahrbahn verlaufenden Haltelinien und Unterbrechungen der Fahrspurmarkierung nur unzureichende Ergebnisse lieferte. Diese Punkte wurden abschließend untersucht und Lösungsansätze erarbeitet.

Rechtskurven mit geringem Radius

In Rechtskurven mit geringem Radius R kommt die ROI an den Rand des Kamerabildes und kann der Fahrspur nicht weiter folgen. Zieht die Fahrspurmarkierung noch weiter nach rechts, verlässt sie teilweise den sichtbaren Bereich, so dass die Anzahl der Vordergrundpixel in der ROI abnimmt und das Ergebnis der Fahrspurapproximation ungenauer wird und schließlich falsche Werte ergibt. Diese führen zu einer

fehlerhaften horizontalen Repositionierung der ROI nach links und damit weg von der Fahrspur. Die nachfolgend dargestellte Testfahrt zeigt genau dieses Verhalten:

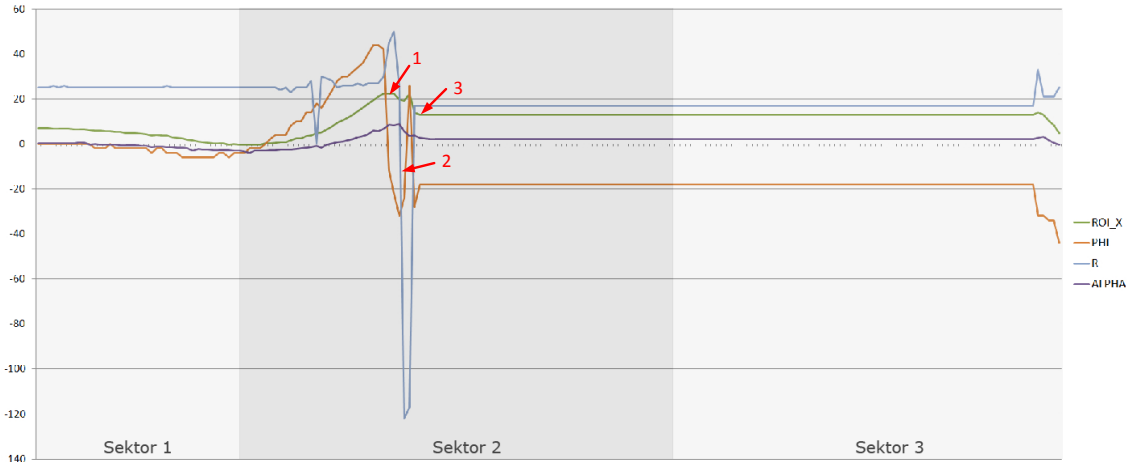

Abbildung 6.7.: Ausbruch in einer Rechtskurve. In Sektor 2 wird der ROI bis an den rechten Bildrand verschoben (1), doch die Fahrbahnmarkierung rückt noch weiter nach rechts, so dass die Ergebnisse der HT ungenau werden (2) und den ROI nach links verschieben (3).

In Sektor 2 wurde die Fahrspurmarkierung verloren kurz nachdem der ROI bis an den rechten Bildrand verschoben wurde (1). Daraufhin wurden die Ergebnisse der Fahrspurapproximation sprunghaft und ungenau (2) und verschoben dadurch den ROI nach links (3), wodurch die Fahrspurmarkierung nicht mehr erkannt wurde. Durch Zufall wurde die Fahrspur zu einem späteren Zeitpunkt erneut erkannt, so dass die Fahrt fortgesetzt werden konnte.

Abhilfe lässt sich hier durch eine Kamera mit einer Weitwinkeloptik schaffen, die einen breiteren Bereich abdeckt.

Haltelinien

Querlinien auf der Fahrbahn, die Haltelinien an Kreuzungen oder die Startlinie markieren, führten zu einem Ausbruch nach links (siehe [6]). Die Haltelinie erhöht die Anzahl der Vordergrundpixel in der ROI stark und verfälscht das Ergebnis der Fahrspurapproximation nach links.

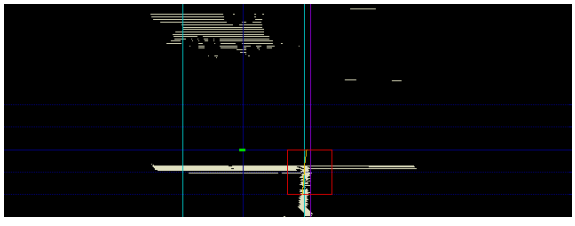

Abbildung 6.8.: ROI (*rot*) bei der Fahrt auf die Haltelinie der Kreuzung.

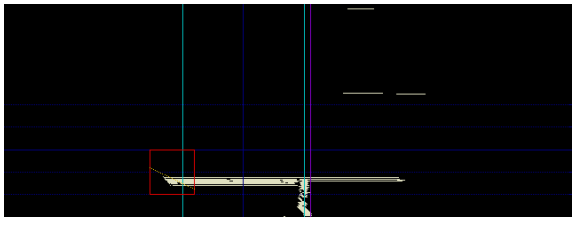

Abbildung 6.9.: Die ROI (*rot*) wird durch die Haltelinie nach links verschoben.

Als Lösungansatz ist es denkbar, die Ergebnisse der Hough-Transformation zu filtern und ihre Veränderung zu überwachen. Ist diese schlagartig, wird das Ergebnis ignoriert und der letzte gültige Wert genommen.

Unterbrechungen der Fahrbahnmarkierungen

Wird die Fahrbahnmarkierung unterbrochen und keine Fahrspur erkannt, fährt das Fahrzeug mit dem letzten errechneten Lenkwinkel α und der letzten horizontalen ROI-Position weiter. Dies führte an Kreuzungen dazu, dass die Fahrspurmarkierung danach nicht zuverlässig wiedergefunden wurde, da sie dann außerhalb der ROI lag. Ein Ansatz ist die horizontale ROI- Position zur Laufzeit zu variieren, um den abgedeckten Bereich zum Erkennen der Fahrspurerkennung zu vergrößern.

7. Zusammenfassung und Ausblick

Im Rahmen dieser Masterarbeit wurde das System-on-Chip basierte Spurführungssystem des SAV in den Aspekten Parameterbestimmung, Kalibrierung, Auswertbarkeit und Konsistenz weiterentwickelt und erweitert. Die Spurführung wurde anhand ihrer mathematischen Modelle analysiert (vgl. Kap. 4) und optimiert, sowie eine bluetoothbasierte Datalogging-Schnittstelle implementiert um Testfahrten systematisch auszuwerten (vgl. Kap. 5 und Kap. 6).

Das Spurführungssystem wurde in seinen Teilbereichen Kameraperspektive, Fahrspurapproximation und Lenkwinkelregelung analysiert. Die Kameraperspektive wurde auf Auswirkungen auf das Fahrverhalten analysiert und unter Berücksichtigung der Fahrbahneigenschaften nach dem Carolo-Cup-Reglement eine geeignete Perspektive bestimmt (siehe Kap. 4.2). Die Fahrspurapproximierung durch eine Hough-Transformation wurde hinsichtlich ihrer Diskretisierung untersucht und Parameter ermittelt, die alle auftretenden Fahrsituationen abdecken ohne zuviele RTL-Ressourcen zu belegen. In einer Parameterstudie wurde der Spurführungsalgorithmus „Pure-Pursuit“ ausgewertet und unter Einbezug des kinetischen Fahrzeugsmodells geeignete Parameter bestimmt. Ergänzend wurde der Spurführungsalgorithmus als Softwarelösung implementiert und hinsichtlich Genauigkeit und Totzeit mit der RTL-Lösung verglichen (vgl. Kap. 4.1).

Das Datalogging wurde über eine Bluetooth-Funkverbindung umgesetzt und Kommunikationsprotokolle entwickelt über die für jedes Bild ein Datensatz mit allen Stell- und Messgrößen des Spurführungssystems an den Host-PC übertragen und dort für eine spätere Auswertung aufgezeichnet werden. So konnten Testfahrten zur Validierung der ermittelten Parameter durchgeführt wurden, systematisch ausgewertet und die ermittelten Parameter messtechnisch bestätigt werden (vgl. Kap. 6).

Ergebnis dieser Arbeit ist ein einsatzfähiges, autonomes Spurführungssystem mit stabilem und reproduzierbarem Fahrverhalten. Somit ist die Basis zur Integration der weiteren Funktionalitäten wie Hinderniserkennung durch einen Laserscanner und Ausweichmanöver oder Einparkassistenten gelegt. Für zukünftige Weiterentwicklungen und Verbesserung der Spurführung wäre der Einsatz einer Kamera mit breiterem Sichtbereich und höherer Bildfrequenz denkbar, um die Einschränkungen durch Kurven außerhalb des Sichtbereichs und der Fahrgeschwindigkeit zu minimieren. Ebenso sollte eine Verhaltenslogik entwickelt werden, damit das Fahrzeug auch Unterbrechungen der Fahrspurmarkierung, wie sie beispielsweise an Straßenkreuzungen auftreten, bewältigen kann.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[1] Andresen, E. *Parallele Programmierung von Eingebetten SMP-Plattformen am Beispiel von Spurführungs-Algorithmen*. AW2 Report, Hochschule für Angewandte Wissenschaften - Hamburg, 2011.

[2] Andresen, E. *ARM-basierter MPSoC unter Linux für eine Fahrspurführung mit Bildverarbeitung*. Master Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2013.

[3] AVNET. Exp expansion connector - specification. http://www.em.avnet.com/en-us/design/linecard/Documents/Xilinx/exp_specification_v1_4.pdf, 2007.

[4] Bordasch, H. *VHDL-Modellierung einer Geschwindigkeitsregelung für ein autonomes Fahrzeug implementiert auf einer SoC-Plattform*. Bachelor Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2009.

[5] Burger, W., and Burge, M. J. Detektion einfacher kurven. In *Digitale Bildverarbeitung*, eXamen.press. Springer Berlin Heidelberg, 2005.

[6] Carolo Cup. Studenten entwickeln autonome Fahrzeuge, Regelwerk. http://www.carolo-cup.de/, 10 2012.

[7] Digilent, Inc. Pmodbt2 reference manual. http://www.digilentinc.com/Data/Products/PMOD-BT2/PmodBT2_rm.pdf.

[8] Guo, Z., Najjar, W., Vahid, F., and Vissers, K. A quantitative analysis of the speedup factors of fpgas over processors. In *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays* (New York, NY, USA, 2004), FPGA '04, ACM, pp. 162–170.

[9] HAW Hamburg, Department Informatik. Faust fahrerassistenz- und autonome systeme. http://www.informatik.haw-hamburg.de/faust.html.

[10] Jestel, A. *Kalibrierung einer Bildverarbeitungsplattform für eine automatische Fahrspurführung*. PJ2 Report, Hochschule für Angewandte Wissenschaften - Hamburg, 2012.

[11] Kirschke, M. *Echtzeitbildverarbeitung mit einer FPGA-basierten Prozessorelementkette in einem Fahrspurerkennungssystem*. Bachelor Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2012.

[12] Kirschke, M. *FPGA-basierte MPSoC-Plattform zur Integration eines Antikollisionssystems in die Fahrspurführung eines autonomen Fahrzeugs*. Master Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2012.

[13] Machanick, P. Smp-soc is the answer if you ask the right questions. In *Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries* (Republic of South Africa, 2006), SAICSIT '06, South African Institute for Computer Scientists and Information Technologists, pp. 12–21.

[14] Mellert, D. *Modellierung einer Video-basierten Fahrspurerkennung mit dem Xilinx System Generator für eine SoC-Plattform*. Bachelor Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2010.

[15] Moussali, R., Ghanem, N., and Saghir, M. A. R. Supporting multithreading in configurable soft processor cores. In *Proceedings of the 2007 international conference on Compilers, architecture, and synthesis for embedded systems* (New York, NY, USA, 2007), CASES '07, ACM, pp. 155–159.

[16] Nikolov, I. *Maschinelles Lernen zur Optimierung einer autonomen Fahrspurführung*. Master Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2011.

[17] Peters, F. *FPGA-basierte Bildverarbeitungspipeline zur Fahrspurerkennung eines autonomen Fahrzeugs*. Bachelor Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2009.

[18] Reichardt, J., and Schwarz, B. *VHDL-Synthese: Entwurf digitaler Schaltungen und Systeme*. Oldenbourg, 2007.

[19] Roving Networks, Inc. Rn42 advanced user manual.

[20] RXTX Project. Rxtx project wiki website. http://rxtx.qbang.org.

[21] Salathe, J. *Integration einer CMOS-Kamera mit parallelem Interface in ein System-on-Chip basiertes Spurführungssystem*. Bachelor Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2012.

[22] Schneider, C. *Ein System-on-Chip-basiertes Fahrspurführungssystem*. Bachelor Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2011.

[23] Sony Inc. *FCB-PV10 Color Camera Modul - Technical Manual*, 1.4 ed., 2006.

[24] Timoschenko, A. *Implementierung einer Geschwindigkeitsregelung als Prozessor-Element auf einer SoC-Plattform für ein autonomes Fahrzeug*. Bachelor Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2010.

[25] Wit, J. S., Wit, J. S., Crane, D. C., Duffy, D. J., Mason, D. P., Schueller, D. J., and Antonio, D. Vector pursuit path tracking for autonomous ground vehicles, 2000.

[26] Xilinx Inc. Xilinx device drivers documentation. http://www.xilinx.com/ise/embedded/edk6_2docs/xilinx_drivers.pdf, Jun. 2004.

[27] Xilinx Inc. Spartan-3a dsp starter platform user guide. http://www.xilinx.com/support/documentation/boards_and_kits/ug454_sp3a_dsp_start_ug.pdf, Jun. 2009.

[28] Xilinx Inc. Microblaze processor reference guide. http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_3/mb_ref_guide.pdf, Apr. 2012.

[29] Xilinx Inc. Os and libraries document collection. http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_3/oslib_rm.pdf, Oct. 2012.

[30] Yang, W., Chung, M.-K., and Kyung, C.-M. Current status and challenges of soc verification for embedded systems market. In *SOC Conference, 2003. Proceedings. IEEE International [Systems-on-Chip]* (2003), pp. 213–216.

[31] Özgür Nurettin Püskül. *SoC-basierte Bildverarbeitungspipeline für eine Fahrspurerkennung und- visualisierung*. Bachelor Thesis, Hochschule für Angewandte Wissenschaften - Hamburg, 2010.

Abkürzungsverzeichnis

ASCII American Standard Code for Information Interchange

BRAM Block RAM

BSP Board Support Package

BT Bluetooth

CCD Charge Coupled Device

CSV Comma Separated Value

DCM Digital Clock Manager

DDR Double Data Rate

DSP Digitaler Signal-Prozessor

FAUST Fahrerassistenz- und Autonome Systeme

FG Frame Grabber

FPGA Field Programmable Gate Array

fps Frames per Second

FPU Floating Point Unit

FSL Fast Simplex Link

FSM Finite State Machine

FTC Follow the Carrot

FVAL Frame Valid

GPIO General Purpose Input Output

HNF Hessesche Normalform

HT Hough Transformation

ILA Integrated Logic Analyzer

IOB Input Output Block

IP Intelligent Peripheral

ISR Interrupt Service Routine

JTAG Joint Test Action Group

LAD Look Ahead Distance

LAP Look Ahead Point

LMB Local Memory Bus

LUT Look Up Table

LVAL Line Valid

MDM Microblaze Debug Module

MPSoC Multi-Processor System-on-Chip

NGC Netlist Constraints File

PF Path Following

PLB Processor Local Bus

POSIX Portable Operating System Interface for UNIX

PP Pure Pursuit

PT Projektive Transformation

PWM Puls Width Modulation

RAM Random Access Memory

RAM Read Only Memory

RC Remote Control

RGB Red Green Blue

RISC Reduced Instruction Set Computer

ROI Region of Interest

RT Real Time

RTL Register Transfer Level

RTOS Realtime Operating System

SAV System-on-Chip Autonomous Vehicle

SoC System on Chip

SPP Serial Port Profile

UART Universal Asynchronous Receiver Transmitter

UCF User Constraint File

VGA Video Graphics Array

VHDL Very High Speed Integrated Circuit Hardware Description Language

XCL Xilinx Cache

XPS Xilinx Platform Studio

A. Systembeschreibung des SAV

Die Eigenschaften des SAV werden unter konzeptionellen Aspekten sowie ihrer Implementierung zusammengefasst.

A.1. Bezeichner im SAV

Konstanten

Bezeichner	Bedeutung	Wert
β	Neigungswinkel der Kamera (vgl. Abbildung 3.3)	59.0°
γ_v	Vertikaler Sichtwinkel der Kamera (vgl. Kap. 4.2)	34.5°
γ_h	Vertikaler Sichtwinkel der Kamera (vgl. Kap. 4.2)	46°
h	Höhe der Kamera über der Fahrbahn	$28cm$
H	Höhe des Kamerabilds	$98.5cm$
L	Achsabstand des Fahrzeugs	$25.7cm$
o_x	Kamera Offset zur Y-Achse des Fzg.-Koordinatensystems	$7cm$
P_x	Anzahl Pixel des Kamerabildes in X Richtung	$640px$
P_y	Anzahl Pixel des Kamerabildes in Y Richtung	$240px$
ϕ_{max}	Obere Grenze des Wertebereichs von ϕ (vgl. Kap. 4.3.1)	+46°
ϕ_{min}	Untere Grenze des Wertebereichs von ϕ (vgl. Kap. 4.3.1)	−46°
r_{max}	Obere Grenze des Wertebereichs von r (vgl. Kap. 4.3.2)	$71px$
r_{min}	Untere Grenze des Wertebereichs von r (vgl. Kap. 4.3.2)	$-36px$
R_{min}	Minimaler Kurvenradius R nach Carolo Cup Reglement	$100cm$
ROI_{height}	Höhe der ROI in Pixeln	$50px$
ROI_{width}	Breite der ROI in Pixeln	$50px$
W	Breite des Kamerabildes (vgl. Gleichung 4.8)	$92.1cm$
w_o	Breite des Kamerabildes an der Oberkante (vgl. Gl. 4.13)	$92.1cm$
w_u	Breite des Kamerabildes an der Unterkante (vgl. Gl. 4.12)	$29.3cm$

Tabelle A.1.: Bezeichnungen der konstanten Parameter des SAV.

Mess- und Stellgrößen

Bezeichner	Bedeutung
α	Lenkwinkel
d	Sollabstand zur Fahrspurmarkierung
λ	Kreisbogen im Pure-Pursuit
ϕ	Winkel als Ergebnis der HT
r	Lotlänge als Ergebnis der HT
R	Kurvenradius der Fahrspurmarkierung
v	Fahrzeuggeschwindigkeit in m/s

Tabelle A.2.: Bezeichnungen der Mess- und Stellgrößen des SAV.

A.2. Koordinatensysteme des SAV

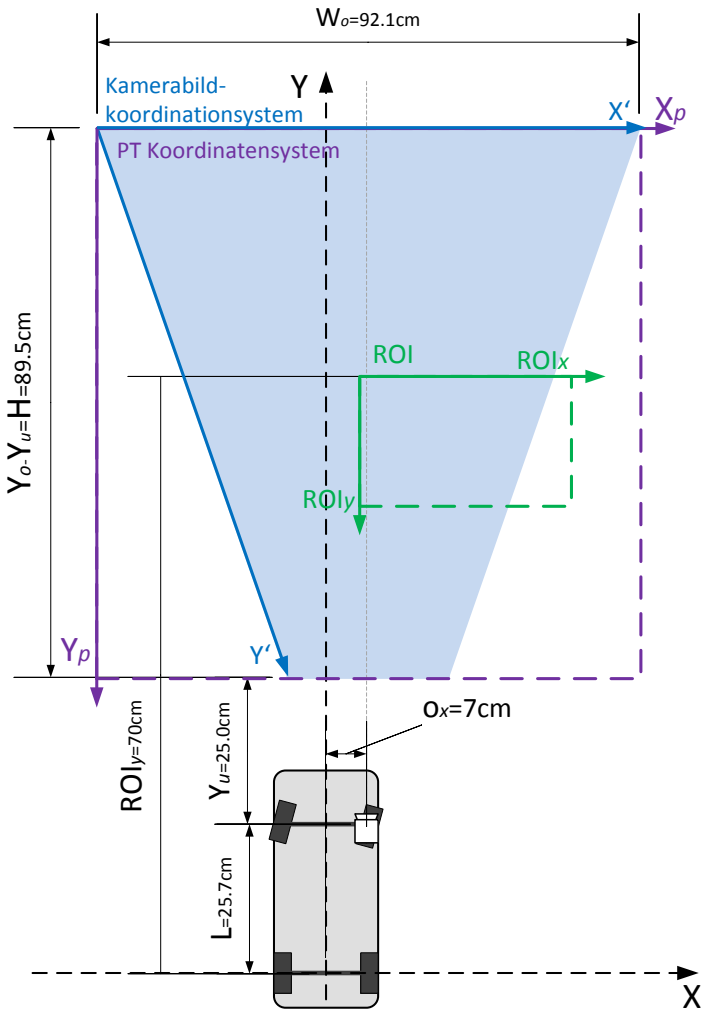

Abbildung A.1.: Konstruktion der Koordinatensysteme des SAV.

Bezeichner	Einheit	Bedeutung
$(X\|Y)$	metrisch	Fahrzeugkoordinatensystem
$(X'\|Y')$	pixel	Kamerabild Koordinatesystem
$(X_p\|Y_p)$	pixel	Transformierte Pixelkoordinaten nach der PT
$(X_h\|Y_h)$	metrisch	Hilfskoordinatensystem zur Konstruktion des LAP (vgl. Kap. 4.1))
$(X_{roi}\|Y_{roi})$	pixel	Koordinatensystem der ROI

Tabelle A.3.: Koordinatensysteme im SAV (vgl. Tab. A.1).

Umrechnungen zwischen Koordinatensystemen

- PT-Koordinaten in das Fahrzeugkoordinatensystem

$$X = (x_p - \left(\frac{P_x}{2}\right)) * \left(\frac{W}{P_x}\right) + o_x \qquad \text{(A.1)}$$

$$Y = (x_p - P_y) * (\frac{H}{P_y}) + L \qquad \text{(A.2)}$$

- Fahrzeugkoordinatensystem in PT-Koordinaten

$$x_p = (X - o_x) * \frac{P_x}{W} + \frac{P_x}{2} \qquad \text{(A.3)}$$

$$y_p = (L - Y) * \frac{P_y}{H} + P_y \qquad \text{(A.4)}$$

- ROI-Koordinaten in PT-Koordinaten

$$x_p = x_{roi} + ROI_{X_{px}} \qquad \text{(A.5)}$$

$$y_p = y_{roi} + ROI_{Y_{px}} \qquad \text{(A.6)}$$

- PT-Koordinaten in ROI-Koordinaten

$$x_{roi} = x_p - ROI_{X_{px}} \qquad \text{(A.7)}$$

$$y_{roi} = y_p - ROI_{X_{px}} \qquad \text{(A.8)}$$

A.3. Kalibrierung der Projektiven Transformation

Die in Kapitel 3.1.3 vorgestellte projektive Transformation beruht auf der Transformationsmatrix B deren Bestimmung in diesem Kapitel aufgezeigt wird. Die Koeffizienten der Matrix B werden durch das folgende Gleichungssystem A.9 beschrieben:

$$\begin{pmatrix} x'_1 & y'_1 & 1 & 0 & 0 & 0 & -x'_1 * x'_{k_1} & -y'_1 * x'_{k_1} \\ 0 & 0 & 0 & x'_1 & y'_1 & 1 & -x'_1 * y'_{k_1} & -y'_1 * y'_{k_1} \\ x'_2 & y'_2 & 1 & 0 & 0 & 0 & -x'_2 * x'_{k_2} & -y'_2 * x'_{k_2} \\ 0 & 0 & 0 & x'_2 & y'_2 & 1 & -x'_2 * y'_{k_2} & -y'_2 * y'_{k_2} \\ x'_3 & y'_3 & 1 & 0 & 0 & 0 & -x'_3 * x'_{k_3} & -y'_3 * x'_{k_3} \\ 0 & 0 & 0 & x'_3 & y'_3 & 1 & -x'_3 * y'_{k_3} & -y'_3 * y'_{k_3} \\ x'_4 & y'_4 & 1 & 0 & 0 & 0 & -x'_4 * x'_{k_4} & -y'_4 * x'_{k_4} \\ 0 & 0 & 0 & x'_4 & y'_4 & 1 & -x'_4 * y'_{k_4} & -y'_4 * y'_{k_4} \end{pmatrix} * \begin{pmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{21} \\ b_{22} \\ b_{23} \\ b_{31} \\ b_{32} \end{pmatrix} = \begin{pmatrix} x'_{k_1} \\ y'_{k_1} \\ x'_{k_2} \\ y'_{k_2} \\ x'_{k_3} \\ y'_{k_3} \\ x'_{k_4} \\ y'_{k_4} \end{pmatrix} \tag{A.9}$$

Es werden vier Referenzpunkte P im Kamerabild sowohl im metrischen Kalibrierungskoordinatensystem $(X_k|Y_k)$ als auch ihre Abbildung im Pixelkoordinatensystem P' bestimmt. Das Kalibrierungskoordinatensystem ist in seinem Ursprung um y_u in Y-Richtung und um o_x in X-Richtung zum Fahrzeugkoordinatensystem verschoben und liegt in der Mitte der unteren Bildkante.

Es bietet sich an, die vier Bildecken als Referenzpunkte zu wählen, da ihre entsprechenden Pixelkoordinaten bereits durch die Auflösung der Kamera vorgegeben sind und diese nicht separat gemessen werden müssen.

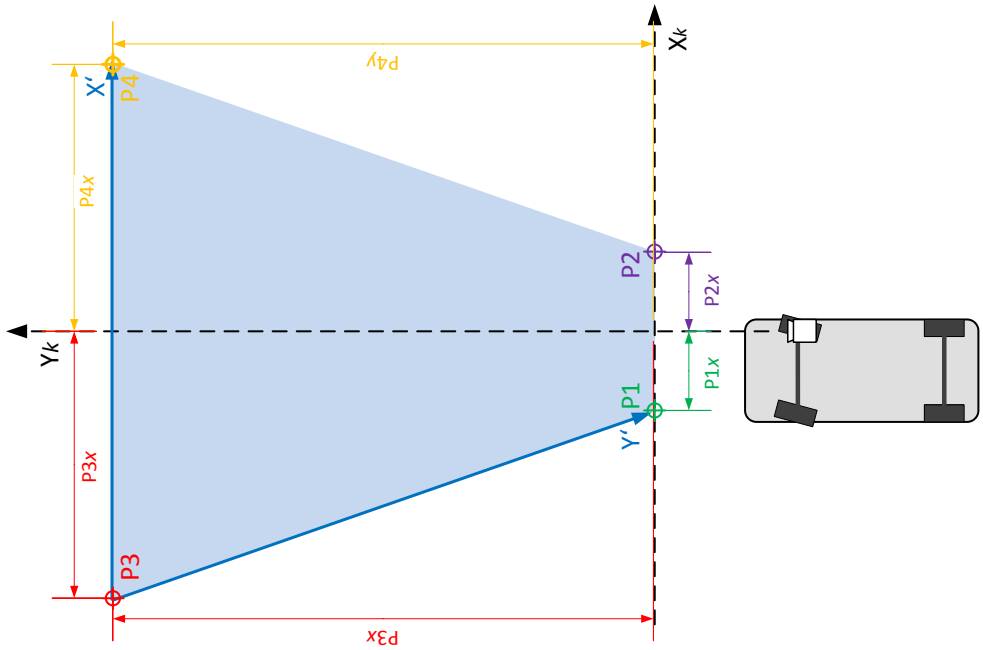

Abbildung A.2.: Kalibrierung der projektiven Transformation durch vier Referenzpunkte in den Bildecken.

Es werden die metrischen Koordinaten $(X_k|Y_k)$ der Referenzpunkte $P1$ bis $P4$ messtechnisch bestimmt und in das Pixelkoordinatensystem der Kamera $(X'|Y')$ umgerechnet (vgl. Gl. A.10 & A.11).

Fahrzeugkoordinatensystem in PT-Koordinaten:

$$X'_k = X_k * \frac{P_x}{W} + \frac{P_x}{2} \qquad \text{(A.10)} \qquad Y'_k = -Y_k * \frac{P_y}{H} + P_y \qquad \text{(A.11)}$$

Jedem Referenzpunkt P im untransformierten Kamerabild $(X'|Y')$ werden so die transformierten Koordinaten $(X'_k|Y'_k)$ zugeordnet, die durch Umrechnung der metrischen Punkte ins Kamerakoordinatensystem ermittelt werden:

Referenzpunkt	X_k	Y_k	X'_k	Y'_k	X'	Y'
P1	$-14.2cm$	$0.0cm$	$222px$	$240px$	$0px$	$240px$
P2	$14.2cm$	$0.0cm$	$418px$	$240px$	$640px$	$240px$
P3	$-46.5cm$	$93.0cm$	$0px$	$5px$	$0px$	$0px$
P3	$46.5cm$	$93.0cm$	$640px$	$5px$	$640px$	$0px$

Tabelle A.4.: Referenzpunkte zur Kalibrierung der Transformations-Matrix der projektiven Transformation.

Durch Einsetzen der 8 Punkte in das Gleichungssystem A.9 wird die Transformationsmatrix B bestimmt:

$$\begin{pmatrix} 0px & 240px & 1 & 0 & 0 & 0 & -0px*222px & -240px*222px \\ 0 & 0 & 0 & 0px & 240px & 1 & -0px*240px & -240px*240px \\ 640px & 240px & 1 & 0 & 0 & 0 & -640px*418px & -240px*418px \\ 0 & 0 & 0 & 640px & 240px & 1 & -640px*240px & -240px*240px \\ 0px & 0px & 1 & 0 & 0 & 0 & -0px*0px & -0px*0px \\ 0 & 0 & 0 & 0px & 0px & 1 & -0px*5px & -0px*5px \\ 640px & 0px & 1 & 0 & 0 & 0 & -640px*640px & -0px*640px \\ 0 & 0 & 0 & 640px & 0px & 1 & -640px*5px & -0px*5px \end{pmatrix} * \begin{pmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{21} \\ b_{22} \\ b_{23} \\ b_{31} \\ b_{32} \end{pmatrix} = \begin{pmatrix} 222px \\ 240px \\ 418px \\ 240px \\ 0px \\ 5px \\ 640px \\ 5px \end{pmatrix} \qquad \text{(A.12)}$$

Als Lösung des Gleichungssystems erhält man die Matrix B:

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & 1 \end{pmatrix} = \begin{pmatrix} 1.0000 & 3.0329 & 0.0000 \\ 0.0000 & 3.2501 & 5.0526 \\ 0.0000 & 0.0095 & 1 \end{pmatrix} \qquad \text{(A.13)}$$

Diese Berechnung ist im *Matlab*-Skript in Anhang C.1 umgesetzt worden.

Bestimmung mit der Neigung β

Alternativ kann die Matrix B über die Kameraneigung β bestimmt werden (vgl. Kapitel 4.2) sofern folgende Angaben bekannt sind:

- Die Höhe der Kamera über der Fahrbahn (h)
- Horizontaler Sichtwinkel der Kamera (γ_h)
- Vertikaler Sichtwinkel der Kamera (γ_v)

Aus diesen Angaben können die Referenzpunkte $P1$ bis $P4$ in den Bildecken nach den Gleichungen 4.12, 4.13, 4.17 und 4.18 in Abhängigkeit der Kameraneigung β errechnet werden:

Referenzpunkt	X_k	Y_k
P1	$-0.5 * w_u$	$0.0cm$
P2	$0.5 * w_u$	$0.0cm$
P3	$-0.5 * w_o$	$y_o - y_u$
P3	$0.5 * w_o$	$y_o - y_u$

Tabelle A.5.: Referenzpunkte zur Kalibrierung der Transformations-Matrix der projektiven Transformation.

Mit diesen Werten kann das im Anhang A.3 erläuterte Verfahren zur Bestimmung der Transformationsmatrix B genutzt werden.

A.4. Belegung der Software-Register

Die Software-Register des *SAV_Connector*-IP sind wie folgt belegt (vgl. Kap. 5.1.1):

Register	Zugriff	bits 31-16	bits 15-0
REG00	IN	OFFSET_X	OFFSET_Y
REG01	IN	ROI_X	ROI_Y
REG02	IN	ROI_WIDTH	ROI_HEIGHT
REG03	IN	V_SOLL	unused
REG04	IN	V_PARAM1	D_SOLL
REG05	IN	V_PARAM2	LAD_VAL
REG06	IN	LAD	THRESHOLD
REG07	IN	SW_PF_ALPHA	
REG08	IN	SW_PF_LAP_X	
REG09	IN	SW_PF_LAP_Y	
REG10	IN	HT_FILTER_PHI	HT_FILTER_R
REG11	IN	unused	HT_VALID_THRESHOLD
REG12	IN	unused	
REG13	IN	unused	
REG14	IN	SW_STATUS_REGISTER	
REG15	IN	SW_CONTROL_REGISTER	
REG16	OUT	LAP_X (S16.2)	LAP_Y (S16.2)
REG17	OUT	V_IST (S9.0)	ALPHA_IST (S16.10)
REG18	OUT	ROI_X_IST (S16.7)	PHI_ID
REG19	OUT	PHI (S9.0)	R (S8.0)
REG20	OUT	HT_MAX_VAL	COUNT
REG21	OUT	unused	
...	...	...	
REG30	OUT	unused	
REG31	OUT	HW_STATUS_REGISTER	

Tabelle A.6.: Belegung der Software-Register. Die Richtung des Zugriffs ist aus Sicht der Hardware definiert. Das Format ist in Klammern angegeben und bezieht sich auf das Q-Format des Wertes.

Belegung SAV_CONTROL_REGISTER

Die Bits sind high-aktiv. Das Register wird standardmäßig so initialisiert, dass alle Funktionen angeschaltet sind.

Bit	Name	Beschreibung
0	EROSION 1x	Schaltet einen Erosionsfilter hinzu
1	EROSION 2x	Schaltet einen Erosionsfilter hinzu
2	PT	Schaltet die Projektive Transformation ein
3	DYNAMIC ROI	Schaltet die dynamische ROI ein
4	SHOW ROI	Zeigt die ROI im RESULT_OVERLAY
5	SHOW LAP	Zeigt den LAP im RESULT_OVERLAY
6	SHOW HT RESULT	Zeigt die Fahrspurapproximation im RESULT_OVERLAY
7	SHOW GRID	Zeigt das metrische Koordinatensystem im RESULT_OVERLAY
8	SHOW VEHICLE	Zeigt die Fahrzeugbreite im RESULT_OVERLAY
9	SHOW D	Zeigt den Sollabstand d im RESULT_OVERLAY
10	PF in HW	Schaltet die Spurführung zwischen HW und SW Variante um. (1=HW)
11	SECURE MODE	Schaltet den *RECEIVER_SWITCH* ein. Abschalten ermöglicht Betrieb ohne Fernbedienung
12	PIXEL PIPELINE	Aktiviert die Pixelpipeline. Abschalten zeigt das ursprüngliche Kamerabild
13	PT SWITCH OFF	Abschalten zeigt nur den Chromianz-Anteil des Bildes in grün an
14	PP (else FTC)	Schaltet den Algorithmus der HW-Spurführung um (1=PP)
15	NO PD	Deaktiviert den PD Regler der HW-Spurführung
16	HOR. EROSION 1x	Schaltet einen horizontalen Erosionsfilter zu
17	HOR. EROSION 2x	Schaltet einen horizontalen Erosionsfilter zu
18	HOR. EROSION 4x	Schaltet einen horizontalen Erosionsfilter zu
19	unused	
20	unused	
21	ADAPTIVE BINARIZE	Aktiviert die adaptive Binarisierung
22	FILTER HT PHI	Aktiviert die Filterung der HT Ergebnisses für ϕ
23	FILTER HT R	Aktiviert die Filterung der HT Ergebnisses für r

Tabelle A.7.: Belegung des *SAV_CONTROL_REGISTER*s.

Belegung HW_STATUS_REGISTER

Bit	Name	Beschreibung
0	HT_DONE	Hough-Transformation abgeschlossen.
1	PF_DONE	Spurführung abgeschlossen.
2	HT_INVALID	Ergebnis der HT ist ungültig.
3	RECEIVER_ENABLED	Manueller Betrieb an Fernbedienung aktiviert.
4	HT_FILTER_ACTIVE	Filterung der HT-Ergebnisse hat gegriffen.

Tabelle A.8.: Belegung des *HW_STATUS_REGISTER*s.

A.5. Pinbelegung des Spartan 3A DSP Boards

In Tabelle A.9 sind alle zum Anschluss von externer Peripherie verwendeten Pins aufgeführt.

Signal		FPGA Pin	Connector	Pin	Beschreibung
PmodBT_RST	OUT	K18	J6 (Pmod 1)	1	
PmodBT_RTS	OUT	L18		1	
PmodBT_RxD	IN	L17	J7 (Pmod 2)	2	Digilent PmodBT
PmodBT_TxD	OUT	E17		3	
PmodBT_CTS	IN	F23		4	
DataIn(0)	IN	AA23		D0	
DataIn(1)	IN	V21		D1	
DataIn(2)	IN	U20		D2	
DataIn(3)	IN	AA24		D3	
DataIn(4)	IN	AA25	J8 (SAM)	D4	Kamera Interface
DataIn(5)	IN	U19		D5	
DataIn(6)	IN	U18		D6	
DataIn(7)	IN	Y22		D7	
ClkCam	IN	AA14		CLK	
HALL_B	IN	G19		JX1_3	EXP-Board: J2-6
HALL_C	IN	G21		JX1_8	EXP-Board: J2-7
PWM_REMOTE_IN	IN	E21		JX1_7	EXP-Board: J2-8
PWM_RECV_MOT_IN	IN	B21	EXP1	JX1_10	EXP-Board: J2-9
PWM_MOT_OUT	OUT	D23		JX1_9	EXP-Board: J2-10
PWM_RECV_SRV_IN	IN	C20		JX1_14	EXP-Board: J2-13
PWM_SRV_OUT	OUT	B23		JX1_13	EXP-Board: J2-14

Tabelle A.9.: Pin Belegung des SAV.

A.6. Nutzung der RTL Ressourcen des Spartan 3A DSP1800 FPGA

Modul	IP	Slices	LUTs	BRAM	DSP48A
SAV_Controller	*SAV_CONTROL*	16711	22404	10	76
MicroBlaze	*MICROBLAZE*	3579	4534	10	8
Memory Controller	*MPMC*	2496	1575	13	0
SAV_Connector	*SAV_CNCTR*	936	968	0	0
PmodBT_UART	*XPS_UART16550*	421	503	0	0
Timer	*XPS_TIMER*	350	351	0	0
PLB	*PLB_V46*	279	281	0	0
ChipScope ILA	*CHIPSCOPE_ILA*	210	186	3	0
Interrupt Controller	*XPS_INTC*	161	100	0	0
CAM UART	*XPS_UARTLITE*	152	128	0	0
RS232_UART	*XPS_UARTLITE*	141	117	0	0
Debug Module	*MDM*	141	139	0	0
PmodBT_Interface	*UTIL_IO_MUX*	134	89	0	0
LEDs	*XPS_GPIO*	108	65	0	0
Push_Buttons	*XPS_GPIO*	88	52	0	0
FSL FG_to_MB	*FSL_V20*	67	74	1	0
ChipScope ICON	*CHIPSCOPE_ICON*	55	44	0	0
System_Reset	*PROC_SYS_RESET*	40	22	0	0
Frame-Grabber	*FRAMEGRABBER*	29	9	0	0
FSL MB_to_FG	*FSL_V20*	20	21	0	0
DLMB_Controller	*LMB_BRAM_IF_CNTLR*	7	5	0	0
Clock Generator	*CLOCK_GENERATOR*	4	0	0	0
ILMB_Controller	*LMB_BRAM_IF_CNTLR*	2	0	0	0
DLMB Local Memory Bus	*LMB_V10*	1	0	0	0
ILMB Local Memory Bus	*LMB_V10*	1	0	0	0
Boot BRAM	*BRAM Block*	0	0	16	0
Summe		26134	31668	53	84

Tabelle A.10.: RTL-Ressourcen Bedarf der IPs im SAV.

Verwendete RTL-Ressourcen des SAV_CONTROL-IP

Modul	Slices	LUTs	BRAM	DSP48A
PATHFINDER	15028	20069	10	67
VGA-Interface	836	1533	0	0
VREGLER	335	496	0	1
Kamera-Interface	191	227	0	8
RECV_SWITCH	28	52	0	0
CLKMGR	5	1	0	0
Summe	16711	22404	10	78

Tabelle A.11.: RTL-Ressourcen Bedarf des *SAV_CONTROL* IPs.

Verwendete RTL-Ressourcen des Pathfinder-Moduls

Modul	Slices	LUTs	BRAM	DSP48A
IMAGE_PROCESSING	938	1264	0	4
RESULT_ILLUSTRATION	584	734	10	15
PHIID_TO_DEGREE	4	4	0	1
PATH_FOLLOWING	1992	2436	0	0
PWM_GENERATION	182	238	0	0
LANE_DETECTION	11328	15393	0	45
Summe	15028	20069	10	65

Tabelle A.12.: RTL-Ressourcen Bedarf des Pathfinder Moduls.

A.7. Protokoll der Anpassung an der Implementierung des SAV

- **Pathfinder** (System Generator Modul)
  - **Image_Processing**
    - Zusätzlichen Erosionsfilter und drei horizontale Erosionsfilter ergänzt.
  - **Lane_Detection**
    - Übergabe der Y-Pixelkoordinaten der Vordergrundpixel an die HT mit 2 multipliziert um die Stauchung des Kamerabilds durch das Interlacing auszugleichen.
    - Anzahl Transformatoren auf 47 erhöht, Winkelbereich $-46° - 46°$, Schrittweite 2° (vgl. Kap. 4.3) (vorher: $-19° - 19°$)
    - Intervall für Lotlänge r auf $1px$ reduziert (vorher $2px$).
  - **Path-Following**
    - Umrechnungsfaktor für r von Pixeln in Zentimeter korrigiert.
  - **Result_Illustration**
    - Anzeige erweitert
  - Umrechnung ins Fahrzeugkoordinatensystem durch globale Parametrisierung im Initialisierungs-Skript vereinheitlicht.
  - Abschalt-Funktion für Funktionsblöcke durch *CONTROL*-Bits ergänzt (vgl. Tab. A.7).
  - *HT_DONE* und *PF_DONE* als Ausgänge definiert als Interrupt-Quellen.
- **SDK Projekt**
  - **Board-Support-Package (BSP)**
    - MicroBlaze Taktfrequenz korrigier: $62.5MHz$ (vorher $100MHz$)
    - Scheduler Time Slices auf $2ms$ reduziert (vorher $10ms$)
    - Scheduler Policy auf *PRIO* geändert (vorher Round Robin) (vgl. Kap. 5.1.4)
  - Datalogging implementiert.
  - Remote Configuration implementiert mit Interrupt-basiertem Empfangen.

- **XPS Projekt**
  - ChipScope IPs ILA und ICON hinzugefügt.
  - Microblaze_1 entfernt um RTL-Ressourcen zu sparen zu Gunsten der größeren HT-Transformatoren Anzahl
- **SAV_Connector IP**
  - Anzahl der Software Register auf 32 erhöht (vorher 10)

Entwicklungsumgebung

Es wurden folgende Versionen der Entwicklungsumgebungen eingesetzt:

Software	Version
Matlab	2012a
Xilinx ISE	14.3

Tabelle A.13.: Versionsbezeichnungen der verwendeten Entwicklungs-Tools.

B. Messdaten

B.1. Liste der Testfahrten

Liste aller in Kapitel 6 Messtechnische Analyse verwendeten Testfahrten mit Angabe der verwendeten Parameter:

Bezeichnung	ROI_Y	LAD	d	v_{soll}	RTL vs. SW	Video
1.1.A	$90cm$	$90cm$	$10cm$	$60\frac{cm}{s}$	RTL	Ja
1.1.B	$90cm$	$90cm$	$10cm$	$60\frac{cm}{s}$	RTL	Nein
1.1.C	$90cm$	$90cm$	$10cm$	$60\frac{cm}{s}$	RTL	Nein
1.2.A	$90cm$	$90cm$	$12cm$	$60\frac{cm}{s}$	RTL	Ja
1.2.B	$90cm$	$90cm$	$12cm$	$60\frac{cm}{s}$	RTL	Nein
1.2.C	$90cm$	$90cm$	$12cm$	$60\frac{cm}{s}$	RTL	Nein
1.3.A	$90cm$	$90cm$	$15cm$	$60\frac{cm}{s}$	RTL	Ja
1.3.B	$90cm$	$90cm$	$15cm$	$60\frac{cm}{s}$	RTL	Nein
1.3.C	$90cm$	$90cm$	$15cm$	$60\frac{cm}{s}$	RTL	Nein
1.4.A	$80cm$	$80cm$	$10cm$	$60\frac{cm}{s}$	RTL	Ja
1.4.B	$80cm$	$80cm$	$10cm$	$60\frac{cm}{s}$	RTL	Nein
1.4.C	$80cm$	$80cm$	$10cm$	$60\frac{cm}{s}$	RTL	Nein
1.5.A	$80cm$	$80cm$	$12cm$	$60\frac{cm}{s}$	RTL	Ja
1.5.B	$80cm$	$80cm$	$12cm$	$60\frac{cm}{s}$	RTL	Nein
1.5.C	$80cm$	$80cm$	$12cm$	$60\frac{cm}{s}$	RTL	Nein
1.7.A	$70cm$	$70cm$	$10cm$	$60\frac{cm}{s}$	RTL	Ja
1.7.B	$70cm$	$70cm$	$10cm$	$60\frac{cm}{s}$	RTL	Nein
1.7.C	$70cm$	$70cm$	$10cm$	$60\frac{cm}{s}$	RTL	Nein
1.10.A	$80cm$	$80cm$	$7cm$	$60\frac{cm}{s}$	RTL	Ja
1.10.B	$80cm$	$80cm$	$7cm$	$60\frac{cm}{s}$	RTL	Nein
1.10.C	$80cm$	$80cm$	$7cm$	$60\frac{cm}{s}$	RTL	Nein
1.11.A	$90cm$	$90cm$	$7cm$	$60\frac{cm}{s}$	RTL	Ja
1.11.B	$90cm$	$90cm$	$7cm$	$60\frac{cm}{s}$	RTL	Nein
1.11.C	$90cm$	$90cm$	$7cm$	$60\frac{cm}{s}$	RTL	Nein
1.12.A	$70cm$	$70cm$	$7cm$	$60\frac{cm}{s}$	RTL	Ja
1.12.B	$70cm$	$70cm$	$7cm$	$60\frac{cm}{s}$	RTL	Nein

1.12.C	70cm	70cm	7cm	60$\frac{cm}{s}$	RTL	Nein
2.1.A	90cm	100cm	10cm	60$\frac{cm}{s}$	RTL	Ja
2.1.B	90cm	100cm	10cm	60$\frac{cm}{s}$	RTL	Nein
2.1.C	90cm	100cm	10cm	60$\frac{cm}{s}$	RTL	Nein
2.2.A	90cm	80cm	10cm	60$\frac{cm}{s}$	RTL	Ja
2.2.B	90cm	80cm	10cm	60$\frac{cm}{s}$	RTL	Nein
2.2.C	90cm	80cm	10cm	60$\frac{cm}{s}$	RTL	Nein
2.3.A	80cm	90cm	10cm	60$\frac{cm}{s}$	RTL	Ja
2.3.B	80cm	90cm	10cm	60$\frac{cm}{s}$	RTL	Nein
2.3.C	80cm	90cm	10cm	60$\frac{cm}{s}$	RTL	Nein
2.4.A	80cm	70cm	10cm	60$\frac{cm}{s}$	RTL	Ja
2.4.B	80cm	70cm	10cm	60$\frac{cm}{s}$	RTL	Nein
2.4.C	80cm	70cm	10cm	60$\frac{cm}{s}$	RTL	Nein
3.1.A	80cm	80cm	10cm	40$\frac{cm}{s}$	RTL	Ja
3.1.B	80cm	80cm	10cm	40$\frac{cm}{s}$	RTL	Nein
3.1.C	80cm	80cm	10cm	40$\frac{cm}{s}$	RTL	Nein
3.2.A	80cm	80cm	10cm	80$\frac{cm}{s}$	RTL	Ja
3.2.B	80cm	80cm	10cm	80$\frac{cm}{s}$	RTL	Nein
3.2.C	80cm	80cm	10cm	80$\frac{cm}{s}$	RTL	Nein
3.3.A	80cm	80cm	10cm	100$\frac{cm}{s}$	RTL	Ja
3.3.B	80cm	80cm	10cm	100$\frac{cm}{s}$	RTL	Nein
3.3.C	80cm	80cm	10cm	100$\frac{cm}{s}$	RTL	Nein
3.4.A	90cm	90cm	10cm	40$\frac{cm}{s}$	RTL	Ja
3.4.B	90cm	90cm	10cm	40$\frac{cm}{s}$	RTL	Nein
3.4.C	90cm	90cm	10cm	40$\frac{cm}{s}$	RTL	Nein
3.5.A	90cm	90cm	10cm	80$\frac{cm}{s}$	RTL	Ja
3.5.B	90cm	90cm	10cm	80$\frac{cm}{s}$	RTL	Nein
3.5.C	90cm	90cm	10cm	80$\frac{cm}{s}$	RTL	Nein
3.6.A	90cm	90cm	10cm	100$\frac{cm}{s}$	RTL	Ja
3.6.B	90cm	90cm	10cm	100$\frac{cm}{s}$	RTL	Nein
3.6.C	90cm	90cm	10cm	100$\frac{cm}{s}$	RTL	Nein
4.1.A	80cm	80cm	10cm	80$\frac{cm}{s}$	RTL	Ja
4.1.B	80cm	80cm	10cm	80$\frac{cm}{s}$	RTL	Nein
4.1.C	80cm	80cm	10cm	80$\frac{cm}{s}$	RTL	Nein
4.2.A	80cm	80cm	10cm	80$\frac{cm}{s}$	SW	Ja
4.2.B	80cm	80cm	10cm	80$\frac{cm}{s}$	SW	Nein
4.2.C	80cm	80cm	10cm	80$\frac{cm}{s}$	SW	Nein
4.3.A	80cm	80cm	10cm	40$\frac{cm}{s}$	SW	Ja
4.3.B	80cm	80cm	10cm	40$\frac{cm}{s}$	SW	Nein
4.3.C	80cm	80cm	10cm	40$\frac{cm}{s}$	SW	Nein

B.2. Laufzeiten der Software-Komponenten

Die Laufzeiten der Threads und Interrupt-Handler wurden mit einem *PicoScope* USB-Oszilloskop an GPIO-Pins gemessen.

Latenz des Aufrufs eines Interrupt Handlers

Die Latenz zwischen dem Puls und dem Aufruf des entsprechenden Interrupt Handlers beträgt 6.4μs (vgl. Abb. B.1).

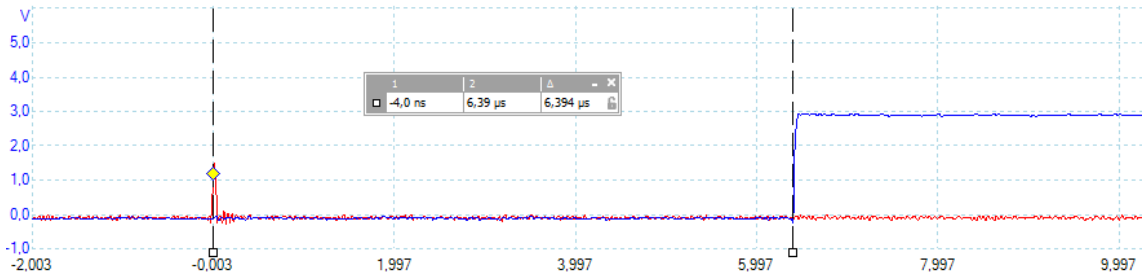

Abbildung B.1.: Die Latenz vom Puls des Interrupts (rot) bis zur Ausführung des Interrupt Handlers (blau) beträgt 6.4μs.

Laufzeiten der Interrupt Handler

Per Oszilloskop wurde die Laufzeiten der Interrupt Handler des SAV gemessen:

Interrupt	Dauer
HT_DONE	5.7μs
PF_DONE	5.6μs
BT_RECEIVE	23.2μs

Tabelle B.1.: Laufzeiten der Interrupt Handler des SAV.

Abbildung B.2.: Die Laufzeit des Interrupt Handlers *HT_DONE* beträgt 5.7μs.

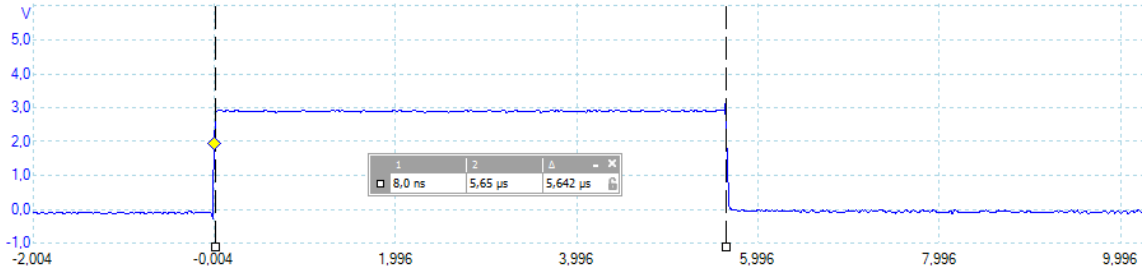

Abbildung B.3.: Die Laufzeit des Interrupt Handlers *PF_DONE* beträgt 5.6μs.)

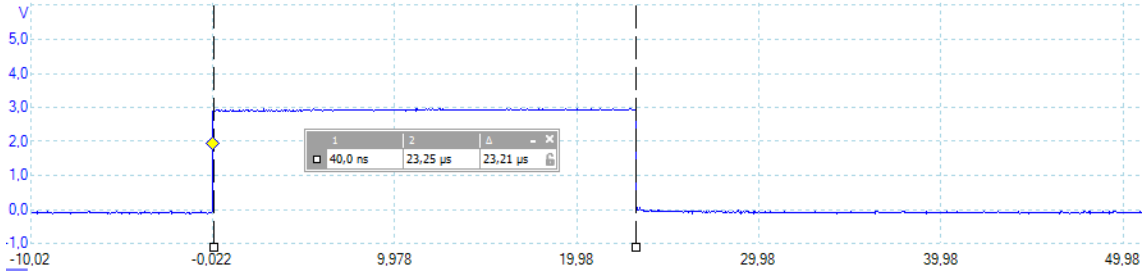

Abbildung B.4.: Die Laufzeit des Interrupt Handlers *BT_RECEIVE* beträgt 23.2μs.)

Laufzeit der Threads

Die Dauer einer Berechnung der Software-Threads des SAV wurde gemessen (vgl. Kap. 5.1.4)

Thread	Dauer
REMOTE_LOGGING	4.7*ms*
REMOTE_PARSING	12.2*ms*
PATHFINDER	1.21*ms*

Tabelle B.2.: Laufzeiten der Threads des SAV.

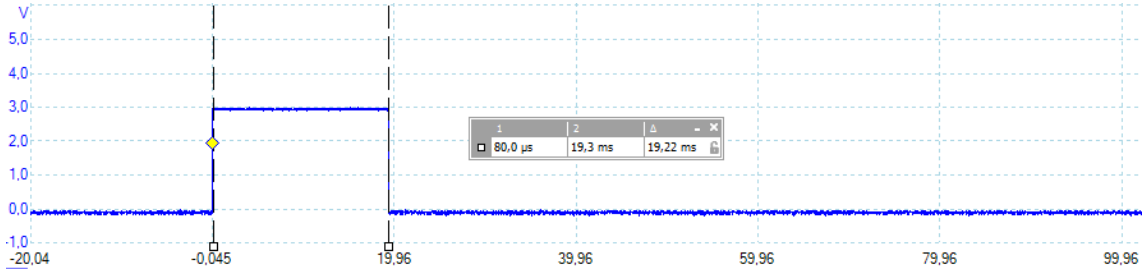

Abbildung B.5.: Empfangen einer Nachricht über das Bluetooth-Interface sowie deren Verarbeitung im Thread *REMOTE_PARSING*: 12.2*ms*.)

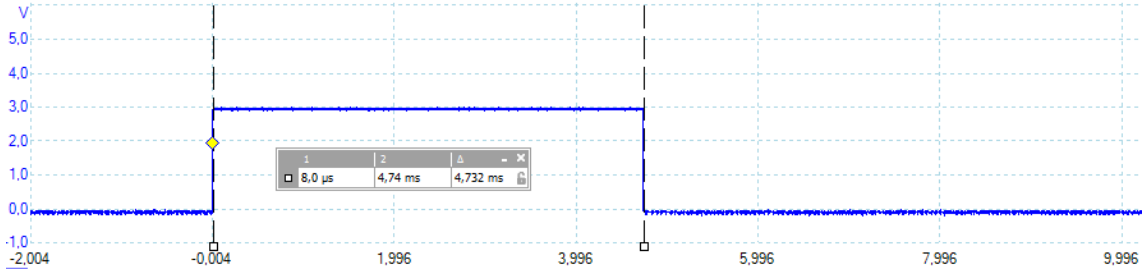

Abbildung B.6.: Formatieren und Senden einer Logging-Nachricht im Thread *REMOTE_LOGGING*: 4.7*ms*.)

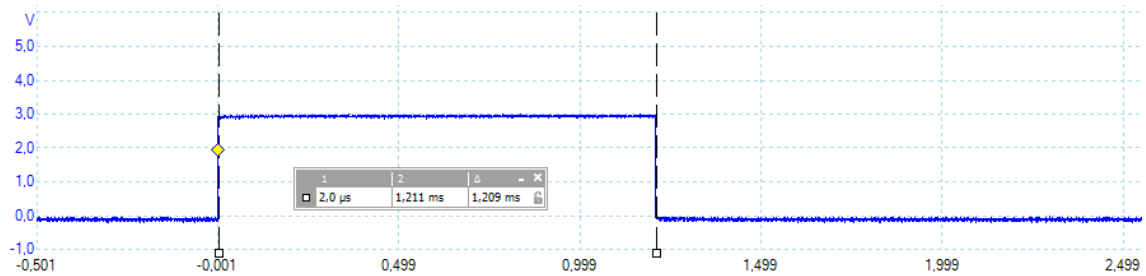

Abbildung B.7.: Berechnung des Spurführungsalgorithmus im Thread *PATHFINDER* bei Verwendung der FPU: 1.21*ms* (vgl. Kap. 5.4.1).)

C. Matlab-Skripte zum mathematischen Modell des SAV

C.1. Transformationsmatrix der PT

```
w_o = 93; %cm
H = 95; %cm
px_x = 640; %px
px_y = 240; %px
offset_x = px_x/2; %px;
offset_y = px_y; %px
x_px_per_cm = px_x/w_o;
y_px_per_cm = px_y/H;

%punkte metrisch gemessen (X_k)
p1m = [ -14.2 0 ]; %cm, X Y
p2m = [ 14.2 0 ];
p3m = [ -46.5 93 ];
p4m = [ 46.5 93 ];

%korrespondierende pixel aus originalbild (X')
p1 = [ 0   240 ]; %px, X Y
p2 = [ 640 240 ];
p3 = [ 0   0 ];
p4 = [ 640 0 ];

% px_pt koordinaten errechnet (X_k')
p1pt = [ (p1m(1)*x_px_per_cm+offset_x) (-p1m(2)*y_px_per_cm+offset_y) ];
p2pt = [ (p2m(1)*x_px_per_cm+offset_x) (-p2m(2)*y_px_per_cm+offset_y) ];
p3pt = [ (p3m(1)*x_px_per_cm+offset_x) (-p3m(2)*y_px_per_cm+offset_y) ];
p4pt = [ (p4m(1)*x_px_per_cm+offset_x) (-p4m(2)*y_px_per_cm+offset_y) ];

% Transformationsmatrix
T= [ p1(1) p1(2) 1 0 0 0 -(p1(1)*p1pt(1)) -(p1(2)*p1pt(1));
     0 0 0 p1(1) p1(2) 1 -(p1(1)*p1pt(2)) -(p1(2)*p1pt(2));
     p2(1) p2(2) 1 0 0 0 -(p2(1)*p2pt(1)) -(p2(2)*p2pt(1));
     0 0 0 p2(1) p2(2) 1 -(p2(1)*p2pt(2)) -(p2(2)*p2pt(2));
     p3(1) p3(2) 1 0 0 0 -(p3(1)*p3pt(1)) -(p3(2)*p3pt(1));
     0 0 0 p3(1) p3(y) 1 -(p3(1)*p3pt(2)) -(p3(2)*p3pt(2));
     p4(1) p4(2) 1 0 0 0 -(p4(1)*p4pt(1)) -(p4(2)*p4pt(1));
     0 0 0 p4(1) p4(2) 1 -(p4(1)*p4pt(2)) -(p4(2)*p4pt(2));];
```

```
X= [ p1pt(1); p1pt(2); p2pt(1); p2pt(2); p3pt(1); p3pt(2); p4pt(1); p4pt
    (2) ];

% Ergebnisse
B = T\X;
```

C.2. Parameterstudie zu Pure-Pursuit

Zur Parameterstudie in Kapitel 4.1.1 wurde folgendes Matlab-Skript entwickelt:

```
function varargout = parameterstudien_r(varargin)
%%% Eigenschaften des Fahrzeugs
    L = 25.7;                   % Achsabstand [cm]
    ALPHA_MAX = 20.00;          % Max. Lenkwinkel [°]
    VISUAL_RANGE_X = 120.00;    % Größe des Y Sichtbereichs [cm]
    VISUAL_RANGE_Y = 93.00;     % Größe des X Sichtbereichs [cm]
    VISUAL_OFFSET_Y = 20.00;    % Distanz Kamera->Sichtbereich [cm]
    RESOLUTION_X = 640;         % Kameraauflösung X [px]

%%% Eigenschaften der Parameterstudie
    STEPWIDTH = 1;              % Schrittweite von ROI_Y und LAD [cm]
    R_STEPWIDTH = 10;           % Schrittweite von R [cm]
    R_MIN = 100;                % minimaler Kurvenradius [cm]
    R_MAX = 5000;               % max. Kurvenradius [cm]
    R_TOLERANCE = 1;            % max. Abweichung um als korrekt zu gelten
    HT_R = 0;                   % r der HT simulation [px]
    D = 0;                      % Sollabstand [cm]
    OUTRULING = 0;              % Parameter die nicht Referenzradius
        ergeben
                                % werden nicht weiter berücksichtigt
    R_PP_MAX = 120;             % max. zulässiges Ergebnis für R in Prozent

    x_px_per_cm = RESOLUTION_X/VISUAL_RANGE_Y; %px pro cm
    visual_range_y_min = VISUAL_OFFSET_Y + L;
    visual_range_y_max = visual_range_y_min + VISUAL_RANGE_X;
    ht_r_metric = HT_R/x_px_per_cm;

    R_range = R_MIN:R_STEPWIDTH:R_MAX;
    lad_range = visual_range_y_min:STEPWIDTH:visual_range_y_max;
    roi_range = visual_range_y_min:STEPWIDTH:R_MIN;

    num_of_values = length(roi_range)*length(lad_range);
    VOTINGS =  zeros( num_of_values,1);
    lads =  zeros( num_of_values,1);
    rois =  zeros( num_of_values,1);
    INPOOL = ones( num_of_values,1);
    % Zur Darstellung von ROI_Y/LAD von R
```

```
R_index = 1;
avg_roi = zeros( size( R_range,1) );
avg_lad = zeros( size( R_range,1) );

ROI_SUMME = 0;
ROI_COUNT = 0;
LAD_SUMME = 0;
LAD_COUNT = 0;

for R=R_range % Für alle R
    R_soll = R+D;
    R_epsilon = R_soll/100*R_TOLERANCE;
    R_max = R_soll/100*R_PP_MAX;

    radien =  zeros( num_of_values,1);
    bestvalues = zeros( num_of_values,1);
    index = 1;

    disp(R);
    R_pp = R_max;

    for lad=lad_range
        for roi_y=roi_range
            if INPOOL(index)== 1
                ht_phi = ht_simulation( R, roi_y );
                roi_x = R+D-cosd(ht_phi)*R;
                alpha = purePursuit( lad, roi_y, roi_x, L, -ht_phi,
                    ht_r_metric, D );

                %Begrenzung von Alpha auf +/-ALPHA_MAX
                if( alpha > ALPHA_MAX )
                    alpha = ALPHA_MAX;
                end
                if( ~isreal(alpha) )
                    alpha = 0;
                end
                if( abs(alpha) < 0.1 || alpha < 0 )
                    R_pp = R_max;
                else
                    R_pp = L / tand(alpha); % Kurvenradius des
                        Fahrzeugs bestimmen
                end

                if( R_pp > R_max )
                    R_pp = R_max;
                end

                if( (R_pp - R_epsilon) < R_soll && (R_pp+R_epsilon) >
                    R_soll )
                    VOTINGS( index ) = VOTINGS( index ) +1;
                    bestvalues(index) = 1;
                else
```

```
                    if OUTRULING == 1
                        INPOOL(index) = 0;
                    end
                end
            end % in pool END
            radien( index ) = R_pp;
            rois( index ) = roi_y;
            lads( index ) = lad;
            index = index+1;
        end
    end

    roi_summe = 0;
    roi_count = 0;
    lad_summe = 0;
    lad_count = 0;

    for index = 1:num_of_values
        if( bestvalues( index ) == 1 )
            roi_summe = roi_summe + rois(index);
            roi_count = roi_count + 1;
            lad_summe = lad_summe + lads(index);
            lad_count = lad_count + 1;
        end
    end

    ROI_SUMME = ROI_SUMME + roi_summe;
    ROI_COUNT = ROI_COUNT + roi_count;
    LAD_SUMME = LAD_SUMME + lad_summe;
    LAD_COUNT = LAD_COUNT + lad_count;

    avg_lad(R_index) = lad_summe/lad_count;
    avg_roi(R_index) = roi_summe/roi_count;

    R_index=R_index+1;
end % for R=R_range

figure;
colormap hsv;
plot(R_range, avg_lad, R_range, avg_roi );
xlhand = get(gca,'xlabel')
set(xlhand,'string','X','fontsize',20)
xlabel('R_soll [cm]');
ylabel('ROI, LAD [cm]');

[X,Y] = meshgrid( lad_range, roi_range );%X=LAD, Y=ROI, Z=VOTES
Z=griddata(lads,rois,VOTINGS, X,Y,'cubic');
figure;
colormap hsv;
surf(X,Y,Z);
title( 'Votings' );
xlabel('LAD [cm]'); ylabel('ROI_Y [cm]'); zlabel('Votes');
```

```
    disp('================================');
    disp('OVER ALL:');
    disp( strcat('LAD average: ', num2str((LAD_SUMME/LAD_COUNT)) ) );
    disp( strcat('ROI_Y average: ', num2str((ROI_SUMME /ROI_COUNT)) ) );
    disp( strcat('Verhältnis LAD/ROI_Y: ', num2str((LAD_SUMME/LAD_COUNT)
       /(ROI_SUMME/ROI_COUNT))));
    disp('================================');
end

function phi = ht_simulation( R, roiy )
    phi = (asind( roiy / R ));
end


function y = rotateRight(x, phi)
    R = [cosd(phi), sind(phi);...
        -sind(phi), cosd(phi)];
    y = R*x;
end

function y = rotateLeft(x, phi)
    R = [cosd(phi), -sind(phi);...
       sind(phi), cosd(phi)];
    y = R*x;
end

function alpha = purePursuit( lad, roi_y, roi_x, L, phi, r, d )
    ROI = [ roi_x ; roi_y ];
    ROI_H = rotateRight(ROI, phi);
    LAP_H_X = ROI_H(1) + r - d;

    if (abs(LAP_H_X)  >= lad )
        LAP_H_Y = 0;
    else
        LAP_H_Y = sqrt( lad^2 - LAP_H_X^2 );
    end

    LAP_H = [ LAP_H_X; LAP_H_Y ];
    LAP_V = rotateLeft( LAP_H, phi );
    alpha = atand( L / ( lad^2 / ( 2 * LAP_V(1) ) ) );
end
```

C.3. Breite des Sichtbereichs in Abhängigkeit der Kameraneigung

```
function varargout = study_picturewidth(varargin)

    y = 60; %cm
    CAM_HEIGHT = 28; %cm Höhe der Kamera

    %Eigenschaften der Kamera (Sony FCB PV10)
    CAM_VIEW_ANGLE_H = 46.0; % degree
    CAM_VIEW_RATIO = (4/3);
    CAM_OFFSET_Y = 25.7; %cm, L
    CAM_VIEW_ANGLE_V = CAM_VIEW_ANGLE_H * (1/CAM_VIEW_RATIO);

    beta_range = 30:0.1:50;
    w =  zeros( length(beta_range),1);
    w_max = -1;
    w_max_beta = -1;
    index = 1;

    for beta=beta_range

        gamma_h = CAM_VIEW_ANGLE_H;
        gamma_v = CAM_VIEW_ANGLE_V;
        h = CAM_HEIGHT;
        L = CAM_OFFSET_Y;

        % berechnen der entfernungen der Bildkanten im Fzg-
            Koordinatensystem
        y_u = tand( beta-0.5*gamma_v ) * h; %approved
        y_o = tand( beta+0.5*gamma_v ) * h; %approved

        % distanz zwischen kamera und oberer bildkante
        d_u = sqrt( y_u^2 + h^2 );
        d_o = sqrt( y_o^2 + h^2 );

        % breite des kamerabilds an der unteren kante
        w_u = sind(0.5*gamma_h)*2*d_u;
        w_o = sind(0.5*gamma_h)*2*d_o;

        delta = atand( (y_o-y_u)/(0.5*(w_o-w_u)));
        w_x = (y-y_u-L)/tand(delta);
        w(index) = w_u + 2*w_x;

        if( w(index) > w_max )
            w_max = w(index);
            w_max_beta = beta;
        end
        index=index+1;
    end
```

```
    figure;
    colormap hsv;
    plot(beta_range, w);
    xlhand = get(gca,'xlabel')
    set(xlhand,'string','X','fontsize',20)
    xlabel('Kamerawinkel beta [°]');
    ylabel('Breite des Kamerabilds bei y=80cm [cm]');
    hold on;
end
```

D. Korrektur des mathematischen Modells des SAV

Zu einem späten Zeitpunkt dieser Abschlussarbeit wurde ein Fehler in der Parameterübergabe des Winkels ϕ zwischen der Geradenapproximation der Hough-Transformation und der Spurführungsalgorithmik festgestellt. Ursache ist eine unterschiedliche Interpretation der Drehrichtung des Winkels ϕ, den die Hough-Transformation positiv im Uhrzeigersinn definiert, die mathematische Drehrichtung der Rotationsmatrizen der Konstruktion des LAP der Spurführung jedoch gegen den Uhrzeigersinn gerichtet ist (vgl. Abbildung 4.1).

Die korrigierten Gleichungen für die Transformation von Koordinaten vom Fahrzeug- ins Hilfskoordinatensystem lautet:

$$\begin{pmatrix} ROI_{X_h} \\ ROI_{Y_h} \end{pmatrix} = \begin{pmatrix} cos(-\phi) & sin(-\phi) \\ -sin(-\phi) & cos(-\phi) \end{pmatrix} * \begin{pmatrix} ROI_X \\ ROI_Y \end{pmatrix} \tag{D.1}$$

Für die Rücktransformation von Hilfs- zu Fahrzeugkoordinatensystem gilt:

$$\begin{pmatrix} LAP_X \\ LAP_Y \end{pmatrix} = \begin{pmatrix} cos(-\phi) & -sin(-\phi) \\ sin(-\phi) & cos(-\phi) \end{pmatrix} * \begin{pmatrix} LAP_{X_h} \\ LAP_{Y_h} \end{pmatrix} \tag{D.2}$$

Die Auswirkungen auf die Konstruktion des LAP (vgl. Gl. 4.2 u. 4.3) wurden durch einen Vergleich der Berechnungen mit korrekter und fehlerhafter Rotationsrichtung dargestellt. Hierzu wurde eine Fahrsituation nachgestellt, bei der der Sollabstand d kleiner dem Ist-Abstand d_{roi} ist und somit eine Rechtskurve gefahren wird (vgl. Kap.6 u. Gl. 6.1):

Parameter	LAD	ROI_X	ROI_Y	d	r
Wert	$70cm$	$14.7cm$	$70cm$	$10cm$	$25px$

Tabelle D.1.: Parameter der untersuchten Fahrsituation mit d_{roi} d.

Zur linearen Konstruktion des LAP wird der Ursprung der ROI ($ROI_X|ROI_Y$) vom Fahrzeugkoordinatensystem in das Hilfskoordinatensystem ($ROI_{X_h}|ROI_{Y_h}$) überführt. Dazu wird korrekterweise der Punkt ($ROI_X|ROI_Y$) nach Gleichung D.1 für

positive Winkel ϕ im Uhrzeigersinn rotiert. Die Fehlerhafte Rotationsrichtung war gegen den Uhrzeigersinn gerichtet (vgl. Gl.4.2):

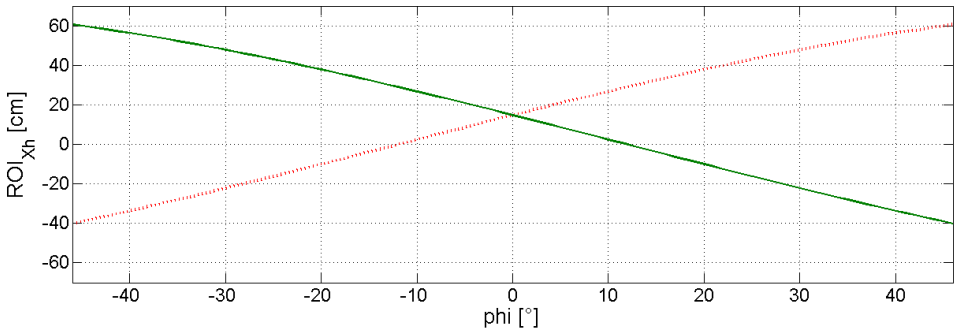

Abbildung D.1.: Fehlerhafte (Gl. 4.2, rot, gepunktet) und korrekte (Gl. D.1, grün) Konstruktion von ROI_{X_h}.

Die fehlerhafte Rotationsrichtung führt bei steigendem ϕ zu einem größeren ROI_{X_h} und damit auch zu einem größeren LAP_{X_h} (vgl. Gl. 4.3). Entsprechend sinkt LAP_{Y_h}, da nach dem Satz von Pythagoras bei konstanter Hypotenuse LAD und einer steigenden Kathete LAP_{X_h} die Länge zweite Kathete sinkt (vgl. Gl. 4.4 u. Abb. D.2).

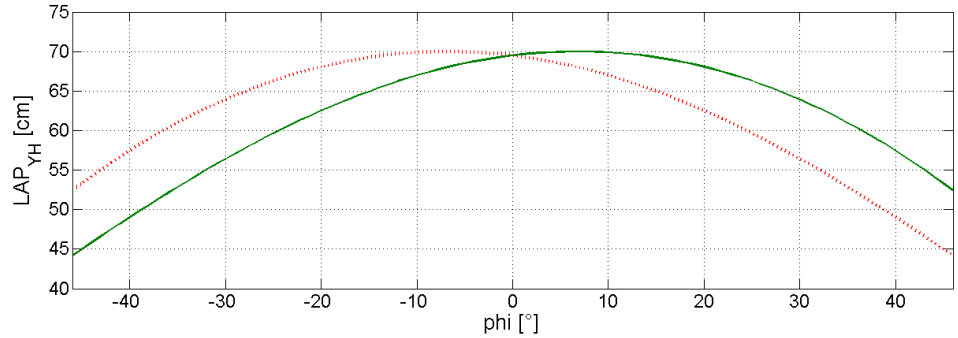

Abbildung D.2.: Konstruktion von LAP_{Y_h} nach Gleichung 4.4 aufbauend auf der fehlerhafte (rot, gepunktet) und korrekte (grün) Transformationsrichtung (vgl. Abb. D.1).

Die Drehung des LAP_h ins Fahrzeugkoordinatensystem zu $(LAP_x|LAP_y)$ erfolgt durch Gleichung 4.5 (vgl. Abb. D.3):

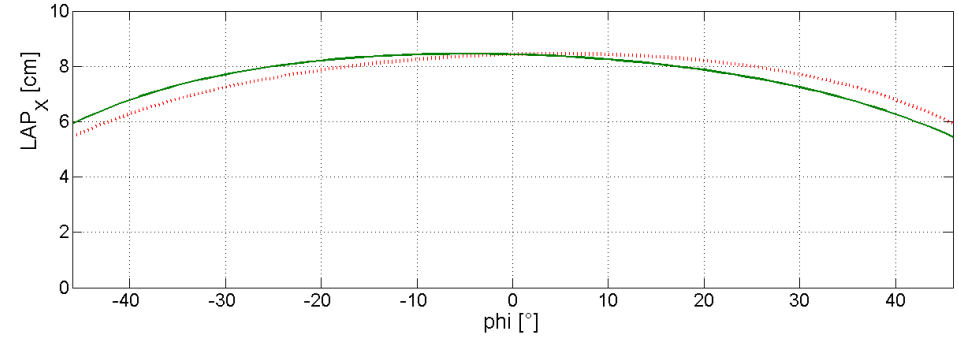

Abbildung D.3.: Konstruktion von LAP_X mit der fehlerhaften (Gl. 4.5, rot, gepunktet) und der korrekten (Gl. D.2, grün) Transformationsrichtung.

Der Lenkwinkel α wird durch das Stellgesetz von Pure-Pursuit berechnet (vgl. 4.1):

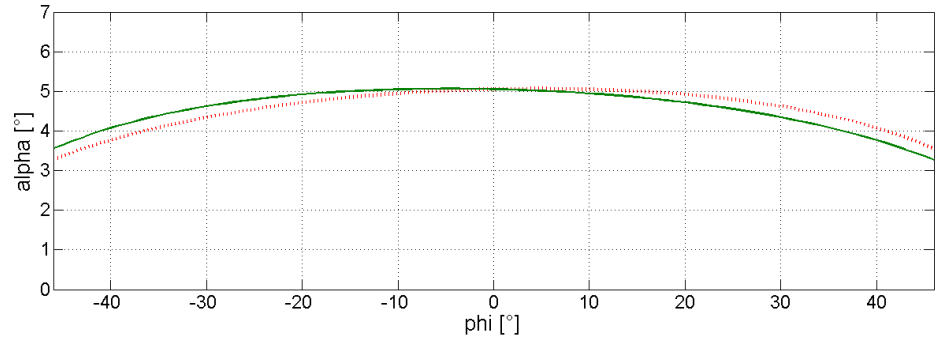

Abbildung D.4.: Lenkwinkel α mit korrekter (grün) und falscher (rot, gepunktet) Rotationsrichtung zur Konstruktion des LAP.

Die fehlerhafte Rotationsrichtung gegen den Uhrzeigersinn führt zu einem Übersteuern in Rechtskurven, da der Lenkwinkel α mit steigendem ϕ größer ist, als mit korrekter Rotationsrichtung im Uhrzeigersinn (vgl. Abb. D.4).

Das Verhalten für Linkskurven wird durch eine vertikale ROI-Position von $ROI_X = -2cm$ untersucht der zu Ist-Abstand zur Fahrspurmarkierung d_{roi} führt der kleiner dem Sollabstand d ist(vgl. Abb. D.5.

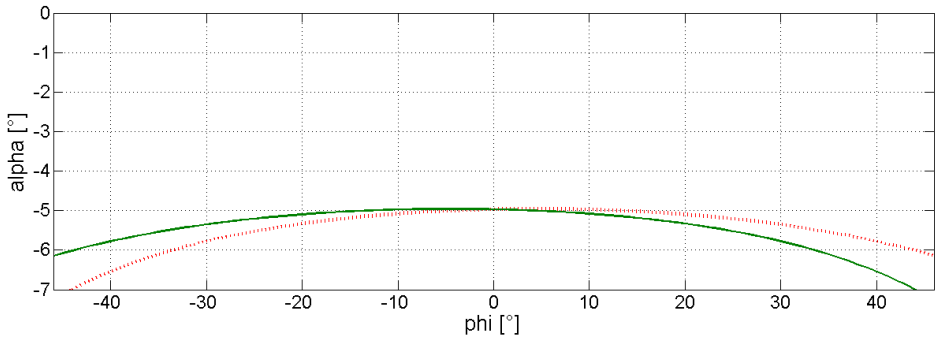

Abbildung D.5.: Lenkwinkel α mit korrekter (grün) und falscher (rot,gepunktet) Rotationsrichtung zur Konstruktion des LAP bei Linkskurven mit $ROI_X = -2cm$.

Auch bei Linkskurven führt die fehlerhafte Transformationsrichtung gegen den Uhrzeigersinn zu einem Übersteuern. Der Lenkwinkel α der fehlerhaften Berechnung ist bei negativem ϕ kleiner als die korrekte Berechnung und führt so zu einem stärkeren Lenken nach links.

Korrektur der Ergebnisse aus Kapitel 4

Die Parameterstudie zu Pure-Pursuit mit der nicht korrekten Transformationsrichtung dienten als Ausgangspunkte zur Auslegung der Kameraperspektive sowie

zur Dimensionierung der Hough-Ebene der Fahrspurapproximation. Die Parameterstudie wurde mit korrigierter Berechnung wiederholt und der Einfluss auf die gezogenen Schlüsse dargestellt

Parameterstudie zu Pure-Pursuit : Die Parameterstudie zu Pure-Pursuit wurde mit korrigierter Rotationsrichtung im Uhrzeigersinn erneut durchgeführt ein (vgl. Kap. 4.1.1).

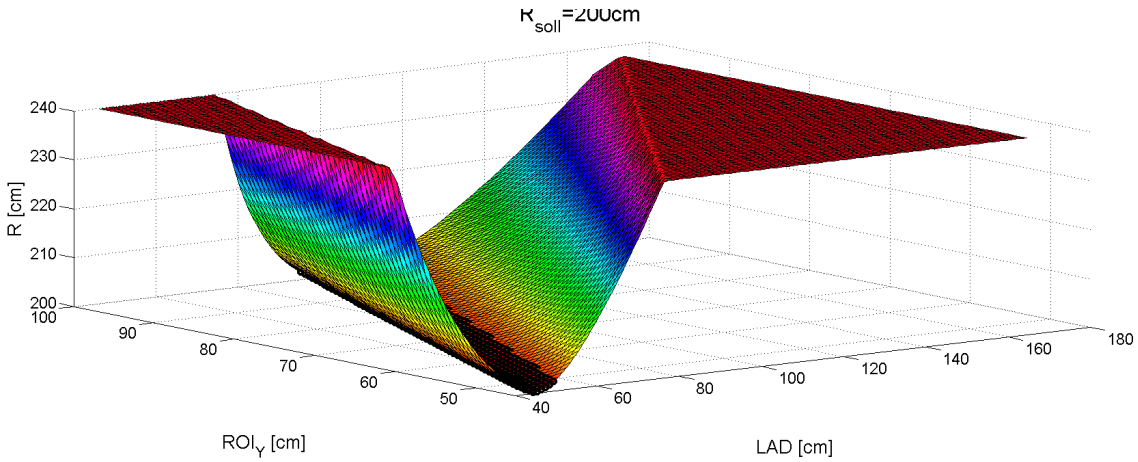

Abbildung D.6.: Parameterstudie für den Kurvenradius $R = 200cm$. Die Parameterpaare (ROI_Y, LAD), deren Ergebnis in der Toleranz des Referenzradius R_{soll} liegen, sind schwarz markiert (vgl. Kap.4.1.1).

Es zeigt sich, das die Parameter LAD und ROI_Y den Referenzradius R_{soll} ergeben, wenn sie in einem annähernd linearen Verhältnis zueinander liegen. Zu große Abweichungen von diesem Verhältnis resultieren in einem zu großen Radius, beziehungsweise in einem zu kleinen Lenkwinkel α. Die absoluten Werte der Parameter beeinflussen das Ergebnis nur minimal.

Um diese Aussage zu bestätigen wurden die Mittelwerte aller Parameterpaare, die den Referenzradius R_{soll} ergaben für alle Kurvenradien im Intervall $100cm - 5000cm$ bestimmt (vgl. Abb. D.7).

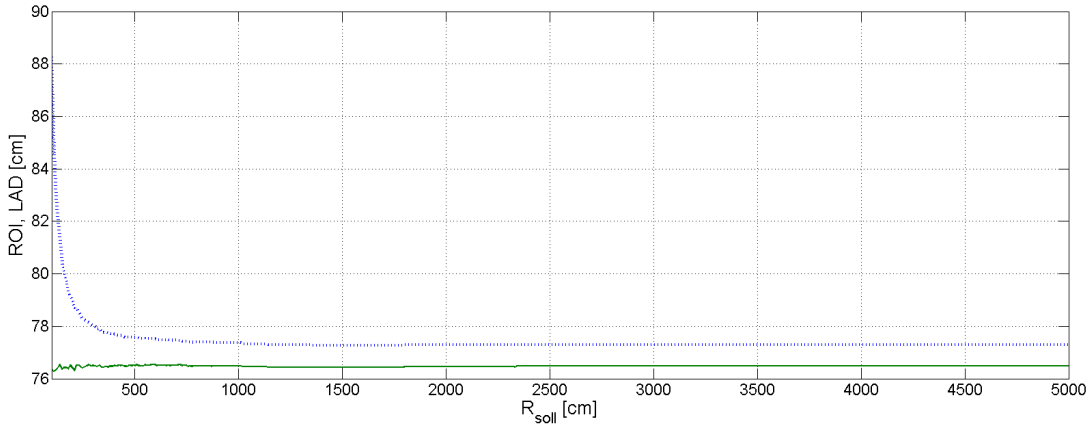

Abbildung D.7.: ROI_Y (blau, gepunktet) und LAD (grün) als Ergebnis der einzelnen Parameterstudien in Abhängigkeit des angestrebten Referenzradius R_{soll} (vgl. Abb. 4.5).

Entsprechend diesen Ergebnissen wurden die in Kapitel getätigten Aussagen korrigiert:

- ROI_Y konvergiert mit steigendem Radius R gegen $77.4cm$ (vorher $80.1cm$). Das Verhältnis von ROI_y zu LAD beträgt $1 : 1.01$ (vorher:$1 : 1$).
- Bei kleinen Radien ist ein größerer ROI_Y-Wert mit bis zu $88cm$ vorteilhaft wobei LAD konstant bleibt. Das Verhältnis verändert sich bis zu $1 : 1.15$.

Im Folgenden werden die Auswirkungen dieser korrigierten Ergebnisse auf die Wahl der Kameraperspektive sowie die Dimensionierung der Hough-Ebene aufgezeigt.

Wahl der Kameraperspektive :

Die Entscheidung für die Kameraperspektive orientierte sich an den verfälschten Ergebnissen der Parameterstudie und wurde ausgehend von einem ROI_Y-Wert von $80cm$ diskutiert (vgl. Kap. 4.2). Die gewählte Perspektive mit der Kameraneigung $\beta = 59°$ wird mit der Perspektive verglichen, die auf $Y = 77cm$ ausgerichtet ist ($\beta = 61.4°$, vgl. Gl. 4.21):

Kriterium	$\beta = 61.4°$	$\beta = 59°$
Bildbreite W bei $77cm$	$47.8cm$	$47.8cm$
Spurbreite	$12px$	$14px$
$ROI_{y_{min}}$	$76cm$	$70cm$
$ROI_{y_{mnax}}$	$165cm$	$141cm$

Tabelle D.2.: Vergleich der gewählten Perspektive ($\beta = 59°$) mit der korrigierten Ausrichtung auf $77cm$ ($beta = 61.4°$).

Die gewählte Kameraperspektive ist auch für die korrigierten Ergebnisse der Parameterstudie geeignet, da sie hinsichtlich der Bildbreite bei $y = 77cm$ keine Nachteile gegenüber einer auf diesen Wert ausgerichteten Perspektive hat. Die Darstellungsbreite der Fahrspurmarkierung würde um zwei Pixel schmaler und würde die Genauigkeit der Fahrspurapproximation begünstigen. Andererseits wäre mit der korrigierten Perspektive lediglich eine minimale, vertikale ROI-Position von $76cm$ möglich, wohingegen die ursprüngliche Perspektive bis zu $70cm$ zulässt.

Die Kameraperspektive bedarf folglich keiner Korrektur, da die Einflüsse durch die Korrektur der Rotationsrichtung zu vernachlässigen sind.

Dimensionierung der der Hough-Ebene :

Ausgehend von einer vertikalen ROI-Position von $ROI_y = 77cm$ ergibt sich der maximale Winkel ϕ für die Hough-Transformation (vgl. Kap. 4.3.1). Dieser wird nach Gleichung 4.32 erneut berechnet:

$$\phi_{max} = tan^{-1}\left(\frac{77cm}{100cm + 5cm - 27.8cm}\right) = 44.93° \tag{D.3}$$

Die ursprüngliche Dimensionierung ging von einer vertikalen ROI-Position von $ROI_y = 80cm$ aus und betrug 46.02° (vgl. Gl. 4.32). Die Abweichung zur korrigierten Auslegung auf $ROI_y = 77cm$ beträgt $delta\phi = 1.1°$ und ist damit kleiner als das gewählte Intervall zur Diskretisierung des Winkelbereichs ϕ der Hough-Ebene von 2°. Somit kann die bestehende Dimensionierung mit einem Winkelbereich von ±46° beibehalten werden ohne das RTL-Ressourcen für nicht benötigte Transformatoren verbraucht werden (vgl. Kap. 4.3.1).

Messtechnische Analyse des Fahrverhaltens des SAV :

Es wurden Testfahrten mit den korrigierten Transformationsgleichungen durchgeführt, um die unmittelbaren Auswirkungen auf das Fahrverhalten zu analysieren. Zur Vergleichbarkeit wurden die selben Parameter von Testfahrt 1.4 aus Kapitel 6 verwendet ($LAD = 80cm$, $ROI_y = 80cm$, $v = 60cm/s$, $d = 10cm$):

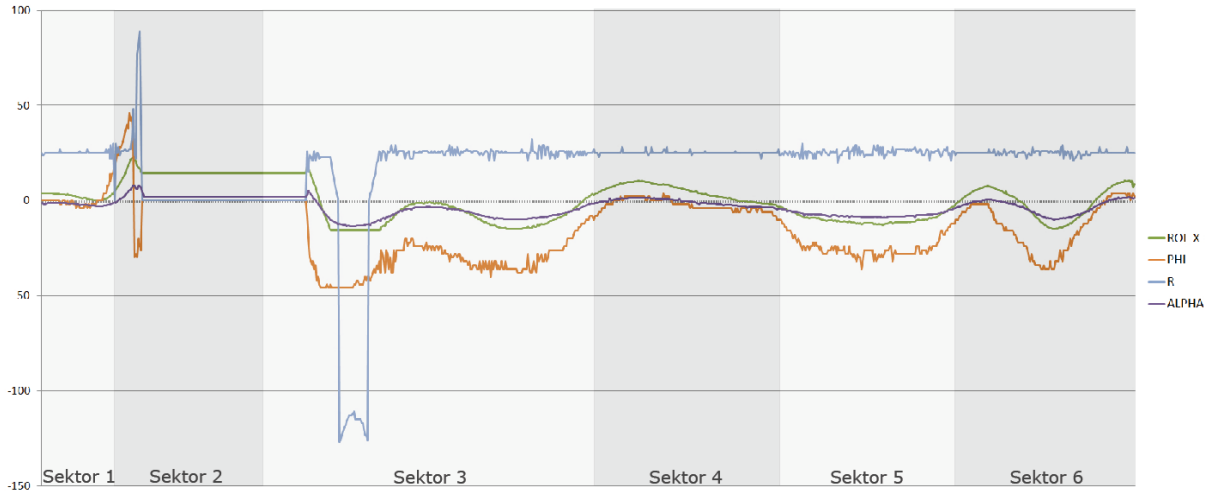

Abbildung D.8.: Testfahrt mit korrigierter LAP-Konstruktion ($ROI_y = 80cm, LAD = 80cm, v = 60cm/s$, $d = 10cm$).

Die Auswertung ergibt eine durchschnittliche Abweichung vom Sollabstand d von $f = 44.6$ (vgl. Kap. 6) und ist damit schlechter als die ursprünglichen Testfahrten mit fehlerhaften Rotationsrichtung mit $f = 36.0$ (vgl. Kap. 6).

E. DVD Inhaltsverzeichnis

- **Matlab-Skripte**
  - **Parameterstudien**
  - **HT-Simulation** : Simulation der HT mit Visualisierung der Hough-Ebene (vgl. Kap. 4.3)
  - **PT Kalibrierung** : Skript zur Berechnung der Transformationsmatrix *B* (vgl. Kap. 3.1.3)
- **Projekte**
  - **Pathfinder** : *Xilinx System Generator* Projekt
  - **XPS** : *Xilinx Platform Studio* Projekt des SAV
  - **Java Remote Tool** : Java Projekt des Remote Tools
- **Testfahrten**
  - **Videos**
  - **Logdaten**
- **Quellen**
  - **Datenblätter**
  - **Referenzen**

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, 28. April 2013	
Ort, Datum	Unterschrift