Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Modellierung eines Einparkassistenten für ein autonomes Fahrzeug implementiert auf einer SoC-Plattform

Thorsten Alpers

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Modellierung eines Einparkassistenten für ein autonomes Fahrzeug implementiert auf einer SoC-Plattform

Thorsten Alpers

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

Betreuender Prüfer: Prof. Dr.-Ing. B. Schwarz
Zweitgutachter: Prof. Dr. rer. nat. H. Heitmann

Abgegeben am 14. Juli 2010

Thorsten Alpers

Thema der Bachelorarbeit

Modellierung eines Einparkassistenten für ein autonomes Fahrzeug implementiert auf einer SoC-Plattform

Stichworte

Einparkassistent, Bahnplanung, Virtuelle Deichsel, Kreistechnik, Matlab, Spurführung

Kurzzusammenfassung

Der Schwerpunkt liegt in der Bahnplanung des Einparkvorgangs. Es wurden fünf Einparkphasen identifiziert, für die jeweils eigene Bahnplanungsverfahren und Matlab-Modelle entwickelt wurden. Die fünf Einparkphasen wurden durch zwei Fahrtechniken realisiert. Eine ist die Kreistechnik, mit der Kreisbahnen zu einem Ziel geplant und gefahren werden. Die Bahnsegmente bestehen aus ein bis zwei Kreisbahnen, die aufgrund des maximalen Lenkwinkels eine maximale Krümmung und minimale Krümmung besitzen. Die Verbindung erfolgt über Anhaltepositionen an den Berührungspunkten. Es werden hiermit kürzeste Wege erzeugt, die in einem definierten Abstand zu den Hindernissen führen. Die Fahrtechnik Virtuelle Deichsel ist eine Lenkwinkelregelung. Über die Wahl der Regelungsparametern wird Einfluss auf das Fahrverhalten genommen. Ausgehend von einer unbekannten Position des Fahrzeugs zu einem Ziel, werden die geeignetsten Parameter gesucht. Hierfür wurden Intervalle von X-, Y- und Achswinkel Differenzen zum Ziel gebildet. In diesen Suchräumen wurden Parameter mit denen die durchschnittliche Abweichung am Ziel am geringsten ist.

Thorsten Alpers

Title of the paper

Modeling of a parking assistance system for an autonomous vehicle implemented on a SoC platform

Keywords

Parking assist, path planning, follow-the-carrot, circle method, Matlab, tracking

Abstract

The focus of this bachelor thesis is the path planning of a parking maneuver. Five parking phases were identified, each own path planning method and therefore were Matlab models developed. The five parking phases were realized by two path planning methods. One is the circle method, which planned orbits towards the goal. The path segments consist between one or two orbits, which have a maximum steering angle. It contains the maximum and minimum curvature. The stopping position is at the contact points of the circles. With this are shortest paths created leading to a defined distance around the obstacles. The follow-the-carrot is a steering angle control. The choice of control parameters influence its behaviour. The most appropriate parameters are searching from an unknown starting position to a goal. This have been formed by intervals of X, Y, and shaft angle differences to the goal. In these search spaces are choosen the parameters which have the minimal average deviations.

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich bei der Durchführung und Erstellung dieser Arbeit tatkräftig unterstützt haben. Mein Dank gilt Prof. Dr.-Ing. Bernd Schwarz und dem gesamten FAUST-Team.

Für die Unterstützung bei Fragen zur Robotik danke ich Prof. Dr. Meisel.

Für die Zusammenarbeit während meines Studium bedanke ich mich bei allen netten Kommilitonen.

Für die Korrektur und Durchsicht dieser Arbeit danke ich Prof. Dr.-Ing. Bernd Schwarz.

Für die finanzielle und moralische Unterstützung während meines Studiums danke ich meinen Eltern Ruth und Werner Alpers, meiner Oma Ruth Kleindienst, meinen Schwestern Iris, Tina und Silvia, meinem Bruder Andreas, meiner Nichte Emilia und meiner verstorbenen Tante Johanna Wilck.

Inhaltsverzeichnis

1 Einleitung

Der Schwerpunkt dieser Bachelorarbeit sind die Bewegungs- und Bahnplanung für einen Einparkassistenten. Die Arbeit ist in das Hochschulprojekt Fahrerlose Autonome Transportsysteme (FAUST) integriert. Der Schwerpunkt des Projektes ist die Entwicklung von autonomen Fahrzeugen. Die Digitale Signalverarbeitung in Fahrzeugsystemen erfordern eine hohe Rechengeschwindigkeiten der Komponenten zur Auswertung der Umgebungsinformationen und zur Berechnung des nächsten Bewegungsablaufs. FPGA-basierte „System on Chip“ Plattformen erfüllen diese Anforderung mit parallelen DSP-Funktionselementen.

Das autonome Bewegen von Fahrzeugen ist eine Technologie die einen großen Nutzen bringt: Im Straßenverkehr werden Unfälle verhindert und der Verkehrsfluss erhöht, in Fertigungsanlagen erhöhen Flurförderfahrzeuge die Abfertigungsgeschwindigkeit und damit die Produktivität. In für den Menschen nicht zugänglichen Zonen, wie in Teilen des Hamburger Hafens am Containerterminals Altenwerder [21], werden autonome Fahrzeuge bereits eingesetzt und verringern die Abfertigungsdauer.

Es gibt zur Zeit eine Vielzahl von Fahrerassistenzsystemen[16]: Einparkassistent, ABS, ESC, Verkehrszeichenbeobachter, Spurwechselassistent, Notbremsassistent, Spurhalteassistent, ... Sie machen das Fahren einfacher und sicherer. Einparkassistenzsysteme werden in vier Kategorien eingeteilt[20]:

1. **Informierende Einparkassistenzsysteme:**
   Sie sind am weitesten verbreitet, sie informieren den Fahrer über die Abstände zu Objekten seitlich, hinter und vor dem Fahrzeug.
2. **Geführte Einparkassistenzsysteme:**
   Sie geben implizite Handlungsanweisungen an den Fahrer: Soll-Lenkwinkel, Fahrtrichtung, Stopp-Punkte und das Ende des Einparkvorgangs.
3. **Semiautomatische Einparkassistenz:**
   Die Lenkung übernimmt das Assistenzsystem, der Fahrer übernimmt die Geschwindigkeitssteuerung.
4. **Vollautomatische Einparkassistenz:**
   Das System übernimmt vollständig das Einparken.

Das Ziel ist die Modellierung eines Einparkassistenten, das als aufbauende Arbeit in dem Hochschulprojekt FAUST Verwendung finden wird. Implementiert wird das Assistenzsystem auf einer System on Chip Plattform, die beim Carolo-Cup eingesetzt wird. Plattformunabhängige Werkzeuge wie UML und SysML gewährleisten die Wiederverwendbarkeit und verringern die Einarbeitungsdauer.

Schwerpunkte sind:

- Recherche zu Bahnplanungskonzepten und Modell-Simulation in Matlab
- Konzept zur Integration von HW-Komponenten
- Entwurf des Einparkassistenten

Vorgehensweise

Die Schritte zur Entwicklung eines Einparkassistenzsystems (siehe Abb. 1) werden hier behandelt. In einem aufbauenden Projekt muss ein Überblick geschaffen werden, was vorhanden ist und was vorab getan werden muss, um den Einparkassistenten realisieren zu können (vgl. Kapitel „Fahrzeugplattform“ und „Kinematische Einspurfahrzeugmodell“). Ein Überblick über ausgewählte Bahnplanungskonzepte aus der Literatur und Forschung, dient als Basiswissen zum Entwurf von eigenen Bahnplanungsverfahren (siehe Kapitel „Grundlagen der Bahnplanung“). Die zu entwickelnden Bahnplanungsverfahren müssen getestet werden. Dafür sind Matlab-Modelle der Verfahren notwendig. Mit ihnen können Fehler entdeckt werden und ein Vergleich von Bahnplanungsverfahren kann realisiert werden. Der Einparkassistent braucht Schnittstellen zum Bahnplanungsverfahren und weiteren Komponenten, die mit der Sensorik und Aktorik kommunizieren. Eine UML-Darstellung des HW/SW-Systems, aus deren Schnittstellen und Aktivitäten ist dafür hilfreich.

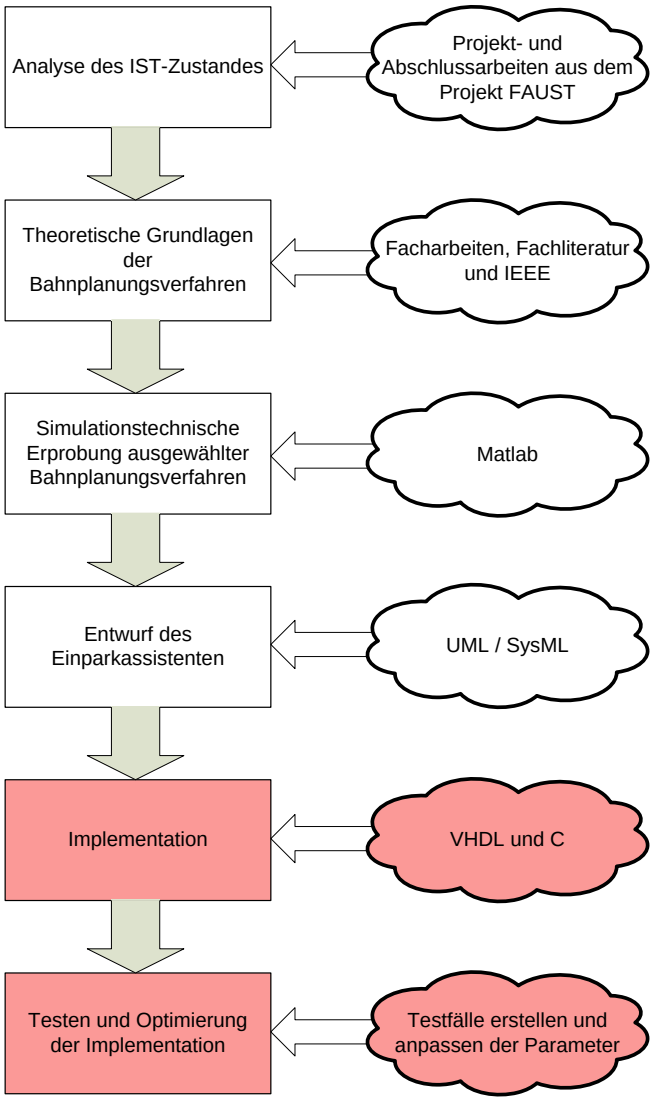


Abb. 1: *Vorgehensweise zur Entwicklung des Einparkassistenten, weiß = realisierte Schritte, rot = zukünftige Entwicklungsschritte*

Fünf Einparkphasen wurden identifiziert (vgl. Abb. 2), für die jeweils eigene Bahnplanungsverfahren und Matlab-Modelle entwickelt wurden. Realisiert wurden die Einparkphasen mit zwei Fahrtechniken.

Eine ist die Kreistechnik, mit der Kreisbahnen zu einem Ziel geplant und gefahren werden. Die Bahnsegmente bestehen aus ein bis zwei Kreisbahnen, die aufgrund des maximalen Lenkwinkels eine maximale Krümmung und minimale Krümmung besitzen. Die Verbindung erfolgt über Anhaltepositionen an den Berührungspunkten. Es werden hiermit kürzeste Wege erzeugt, die in einem definierten Abstand zu den Hindernissen führen.
Die Fahrtechnik Virtuelle Deichsel ist eine Lenkwinkelregelung. Über die Wahl der Regelungsparametern wird Einfluss auf das Fahrverhalten genommen. Ausgehend von einer unbekannten Position des Fahrzeugs zu einem Ziel, werden die geeignetsten Parameter gesucht. Hierfür wurden Intervalle von X-, Y- und Achswinkel Differenzen zum Ziel gebildet, in denen die

Parameter ausgewählt wurden, bei denen die durchschnittliche Simulationsabweichung am Ziel am geringsten ist.

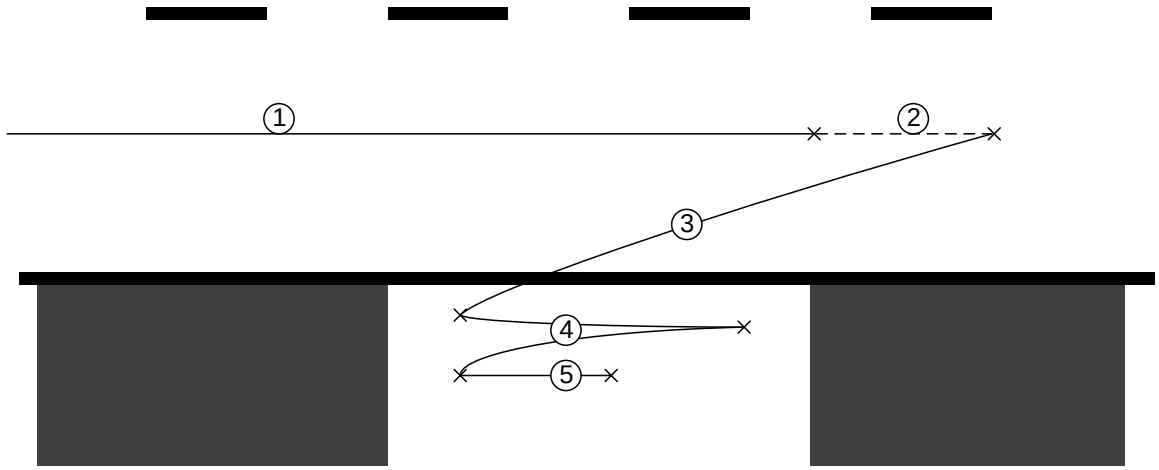


Abb. 2: *Einparkphasen des Einparkvorgangs: 1) Spurführung, 2) Positionierung, 3) erster Einparkschritt, 4) N-Einparkschritte, 5) Fahrzeug ausrichten*

Übersicht zu den Einparkphasen (vgl. Abb. 2)

1. **Spurführung**: Der Fahrspur in einem Achswinkel von 0 Grad folgen.
2. **Positionierung**: Zum Einparken eine exakte Position erreichen, von der aus eingeparkt wird.
3. **Erster Einparkschritt**: Das erste Einparkmanöver in die Parklücke hinein.
4. Weitere **N-Einparkschritte**: Durch Vor- und Zurücksetzen in der Parklücke eine ausreichende Tiefe erreichen.
5. Das **Fahrzeug ausrichten**: Die Einparktiefe ist erreicht, nur der Achswinkel liegt nicht im Bereich von 0 +/- 5 Grad.

Kapitelübersicht

Einleitend wird im Kapitel 2 eine Übersicht zu der Fahrzeugplattform gegeben. Inhalte sind die Fahrzeugmechanik, -aktorik und -sensorik, sowie die FPGA Hardware/Software-Plattform. Die Sensorik erfasst die Umgebung, wie die Fahrspur und die Parklücke. Eine Zuordnung von erfassten Sensorwerten zu einer relativen Lage des Fahrzeugs und die Lokalisierung des Fahrzeugs in der Umgebung, wird mit einem mathematischen Einspurfahrzeugmodell realisiert. Im Kapitel 3 wird es beschrieben und deren Simulation in Matlab.
Darauf aufbauend werden in Kapitel 4 ausgewählte Bahnplanungsverfahren aus der Forschung analysiert und eigene Verfahren für das Einparkmanöver entworfen. Es werden zwei Bahnplanungsverfahren die „Virtuelle Deichsel“ und die „Kreistechnik“ vorgestellt, mit denen alle 5 Einparkphasen realisiert werden.
Im Kapitel 5 wird die mathematische Realisierung der Bahnplanung für die Einparkphasen entworfen. Matlab-Modelle gewährleisten die Funktionalität der Verfahren. Zur Abrundung der Arbeit und zum Fortschritt des FAUST-Projektes, wird im Kapitel 6 ein Ausblick auf weitere Entwicklungsschritte des Einparkassistenzsystems gegeben.

2 Fahrzeugplattform

Im folgenden Kapitel wird die Plattform erläutert, die der Einparkassistent nutzen wird. Zuerst wird die Fahrzeugmechanik und -elektronik beschrieben, auf der die Steuerlogik aufbaut. Sie besteht aus einem FPGA Spartan3e und hat über die I/Os des Nexys2-Boards Kommunikationswege zu der Aktorik (Geschwindigkeit und Lenkwinkel) und Sensorik des Fahrzeugs.

2.1 Fahrzeugmechanik und Elektronik

Die Fahrzeug, des zu entwickelnden autonomen Einparkassistenzsystems, ist ein 1:10 Fahrzeugmodellauto. Es wird manuell über eine Fernsteuerung bedient. Die Kommunikationskanäle der Fahrzeugperipherie werden abgegriffen und an den FPGA Spartan3e angebunden und ausgewertet. Zum Treffen von autonomen Fahrentscheidungen werden Umgebungsinformationen gebraucht, die zusätzliche Sensoren liefern.

Fahrzeuggeometrie des Tamiya TT-01 1/10 Chassis Enzo Ferrari:

- Länge: 457 mm
- Spurweite, Abstand zwischen den Radmittelpunkten auf einer Achse: 160 mm
- Breite: 202 mm
- Radstand, Abstand zwischen Vorderachse zur Hinterachse: 257 mm
- Abstand Vorderachse zur vorderen Fahrzeugbegrenzung: 80 mm
- Abstand Hinterachse zur hinteren Fahrzeugbegrenzung: 55 mm
- maximaler Lenkwinkel: 20 Grad

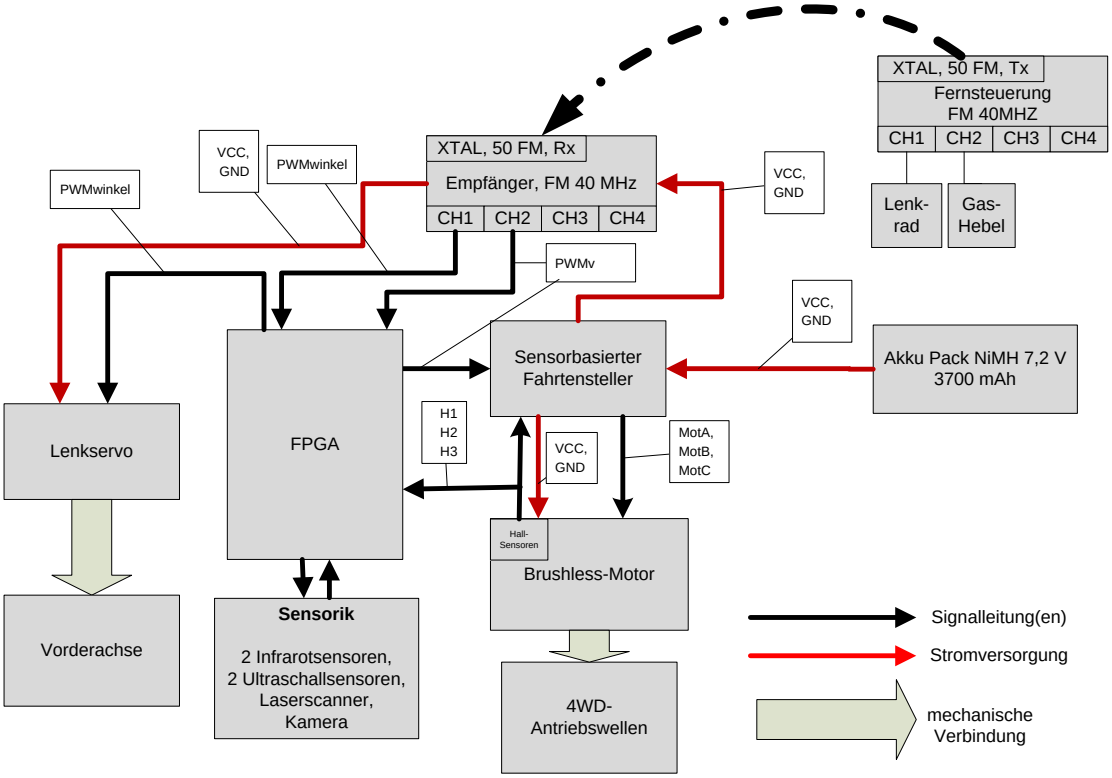


Abb. 3: *Aufbau und Wirkungsweise der Fahrzeugelektronik und -mechanik*

Der Aufbau der vorhandenen Fahrzeugmechanik und -elektronik ist in Abb. 3 dargestellt.

Die Kennwerte der Hardware Peripherie sind wie folgt:

Manuelle Steuerung

- **Fernsteuerung**: Acoms Hayabusa, 4-Kanal-FM-Sender 40 MHz, Kanal 50 bis 53.
- **Empfänger**: Acoms FR-4, 4-Kanal-FM-Empfänger 40 MHz, Kanal 50 bis 53.

Sensorik

- **Infrarotsensoren**: Entfernungssensor GP2D12, Messbereich von 8 cm bis 80 cm, sie erfassen punktuell Entfernungen. Die Entfernungen werden über die SPI-Schnittstelle[18] empfangen.
- **Ultraschallsensoren**: Ultraschall Entfernungsmesser SRF 10, Reichweite von 4 cm bis 6 m, sie erfassen die Umgebung über einer Kegelförmigen Ausbreitungswelle des Schalls
- **Kamera**: Sony PV10 VGA
- **Laserscanner**: HOKUYO URG-04LX, Scannbereich von 240 Grad

Lenkwinkelansteuerung

- **Servo**: Acoms AS-17, stellt über eine PWM einen Lenkwinkel ein

Geschwindigkeitsansteuerung

- **Fahrtensteller**: A.I. Brushless Reverse digital, stellt die Drehzahl am Motor ein und sorgt für die Spannungsverteilung. Zusätzlich wird der Fahrtensteller während der Fahrt automatisch kalibriert (A.I.), so wird der Geschwindigkeitspunkt getroffen[6]
- **Motor**: LRP Crawler Brushless 21,5 Turns , ist ein Sensorgesteuerter Bürstenloser Motor[7], der ohne Kontaktflächen den Motor antreibt. Über die Hallsensoren wird die Position des Rotors erfasst. Über die Rückführung können die Phasen je nach Lage des Rotors passend geschaltet werden um das volle Drehmoment zu bekommen.
- **Fahrwerk**: Die Funktionsweise [5] eines 4WD Allradantriebes ist, dass über Differentiale/Übersetzungen Drehzahlausgleich zwischen allen Rädern erfolgt. Ein Längsdifferential verteilt die Drehzahlen längs auf die Vorder- und Hinterachse und ein Achsdifferential verteilt die Drehzahl quer auf die Räder einer Achse. So wird sichergestellt, dass kein Rad durchdreht und das Fahrzeug nicht über- oder untersteuert. Alle Räder haben so unterschiedliche Drehzahlen und die Drehzahl ist nur bei Geradeausfahrt die des Motors. Laut Dokumentation[33] hat der TT-01 1/10 Enzo Ferrari nur 2 Achsdifferentiale, eine Angabe zu einem Längsdifferential und einer Differentialsperre fehlt.
- Zwei **Achsdifferentiale**[5] an Vorder- und Hinterachse, sie gleichen die Drehzahlen an den Rädern einer Achse aus. In einer Kurvenfahrt fahren die Räder unterschiedliche Radien und legen so einen unterschiedlich langen Weg zurück, die Differentiale gleichen das aus. Eine zentrale Kardanwelle überträgt die Drehzahl gleichmäßig auf die Differentiale. Während einer Geradeausfahrt ist das Verteilungsverhältnis des Drehmoments[4] 50:50, die Summe der Verhältnisse ist das Doppelte der Drehzahl an der Kardanwelle. Bleibt ein Rad stehen und das Fahrzeug besitzt keine Differentialsperre, so wird die Drehzahl des anderen Rades Doppelt so hoch übersetzt.

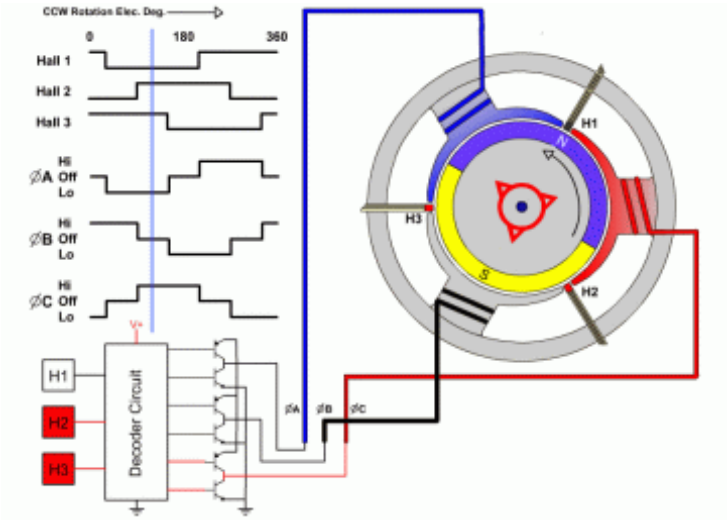


Abb. 4: *Funktionsweise eines 2-Poligen 3-Phasen DC Brushless-Motors[30]*

Die Funktionsweise eines DC Brushless Motors (Abb. 4) ist, dass je nach Lage des Dauermagneten Phasen geschaltet werden. Je nach Drehrichtung des Motors bewirken sie eine Abstoßung oder Anziehung zu dem Dauermagneten. Die dargestellte Vorschaltung sorgt für die zeitliche Schaltung und die Umpolung der Phasen, so dass lokal eine Wechselspannung erzeugt wird. Die dort angebrachten Hallsensoren erfassen den Magnetfluss und können so die Position der Dauermagneten erfassen und beim Beschleunigen aus dem Stillstand, das volle Drehmoment abrufen.

2.2 Autonomes Steuerungssystem

Das Fahrerassistenzsystem wird auf einer System on Chip Plattform implementiert. Es können sowohl Software wie auch Hardwarelogik genutzt werden. Anschlüsse des Boards werden zur Kommunikation mit der Aktorik und Sensorik verwendet.

Nexys2-Board
Das Nexys2-Board von Digilent besteht aus dem FPGA Spartan3e der Firma Xilinx und On-Board Peripherie.

Technische Daten des Nexys2 Boards sind:

- Clock 50 MHz
- 1200 K Gates
- 2168 CLBs
- Flash 16 MByte
- SDRAM 16 MByte
- RS232 Port, bidirektional
- 10 Signal VGA Port, mit 8 Bit Farben
- 4 Taster
- 8 Schalter
- 4 mal 12 Pins für Pmods

Entwickelt wird das System mit dem Xilinx Embedded Development Kit (EDK) und dem Software Development Kit (SDK). Das EDK verwendet die IBM Core-Connect Busarchitektur(siehe. Abb. 5), an die eigene Komponenten hinzugefügt werden und darüber Daten austauschen. Natürlich können Komponenten auch interne und externe Signalleitungen verwenden, ohne den Bus zu nutzen. Das EDK bietet hier unter anderem an, die Signale von Komponenten grafisch zu verbinden und Konfigurationen von Komponenten vorzunehmen.

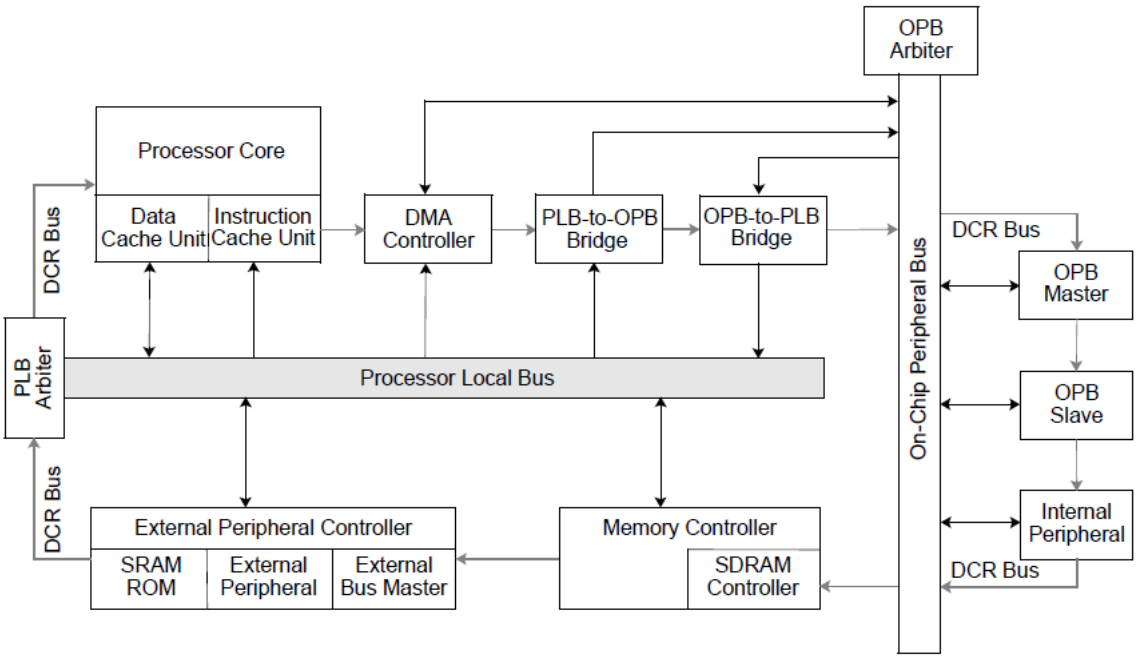


Abb. 5: *IBM CoreConnect Bus-Architektur: [2]*

An den Processor Local Bus (**PLB**) sind Peripherie und der MicroBlaze verbunden. Maximal können 16 Bus-Master und beliebig viele Bus-Slaves verwendet werden. Es unterstützt 16-, 32- und 64 Bit Transfers, die auf maximal 256 Bit erweiterbar sind.
Die langsamere Peripherie kann an den On Chip Peripheral Bus (**OPB**) angeschlossen werden. Über zwei Bridges wird die Kommunikation zwischen PLB und OPB hergestellt.
Eine direkte Verbindung mit dem Prozessor kann mit dem Fast System Link (**FSL**) realisiert werden. Co-Prozessoren nutzen ihn, um die ALU zu umgehen. Insgesamt sind 8 FSL-Master und 8 FSL-Slaves Interfaces erlaubt, die jeweils unidirektional sind. Sie tauschen Daten über eine FIFO aus, die eine Breite von 8,16 oder 32 Bit beträgt.

Eigene IP-Cores können somit über den FSL, OPB und FSL in das System eingebunden werden (vgl. IBM CoreConnect Manual[2]).
In der folgenden Abbildung6 ist die Struktur eines IP-Cores näher dargestellt.

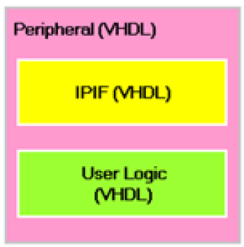


Abb. 6: *Struktur eines Ip-Cores : [3]*

Sie besteht aus der Top-Entity Peripheral.vhd, die den Namen der IP trägt. Instanziiert sind zwei Komponenten. Das ist zum einen die IP Bus-Interface IPIF.vhd, die eine Busschnittstelle implementiert, je nach dem was gewünscht ist wird ein PLB, FSL oder OPB Interface instanziiert.

In der User_logik.vhd kann der Anwender Hardwarebeschleuniger direkt über VHDL-Code oder Komponenten-Instanziierung erzeugen.

IP Cores des Fahrzeugsystems
Die vorhandenen und geplanten IP-Cores dieses Projektes sind wie folgt vorgesehen:

- Das Starten einer Disziplin, wie Einparken, Rundkurs und Rundkurs mit Hindernissen, wird während des Carolo Cups über Push Buttons erfolgen. Sie werden vom **Microblaze** ausgelesen.
- **IP Einparkassistent** beinhaltet den Steuerautomaten für das Einparken.
- Der Geschwindigkeitsregler **V-Regler_IP** ermittelt den gefahrenen Weg und die Ist-Geschwindigkeit. Er gibt an den Fahrtensteller die Soll-Geschwindigkeit über eine PI-Regelung in Form einer PWM weiter.
- **IP Lenkwinkelsteller** gibt den Soll-Lenkwinkel an den Lenkservo weiter.
- **IP LED-Ansteuerung** schaltet die LEDs an, aus oder blinkend ein.
- Der **IP HAW SPI-Master** wertet die Infrarotsensoren aus und liefert die exakten Abstände.
- Die **IP US-Sensorik** liefert die Ultraschallsensordaten.
- Der **NET 1 IP CORE** realisiert die Datenübertragung über WLAN, hauptsächlich zum Datenlogging während der Fahrt.
- Über die **UART**-Schnittstelle soll als Datenlogger dienen, solange das Fahrzeug steht.
- An den **Interrupt-Controller** können mehrere Interruptquellen angeschlossen werden auf die der MicroBlaze oder die IP-Cores reagieren können.
- **IP Fahrspurerkennung** erkennt die Fahrspurbegrenzung.

Im Anhang „Ergänzungen zur Plattform“ werden die vorhandenen IP-Cores als UML-Kompositionsdiagramm detailliert dargestellt und ein Detailplan des Aufbaus der Fahrzeugmechanik und -elektronik.

3 Kinematische Einspurfahrzeugmodell

Für die Lokalisierung des Fahrzeugs wird ein Mathematisches Fahrzeugmodell gebraucht, mit der von einem Anfangszustand die Änderungen (*theta*, x, y, s) erfasst werden. Das wird über kinematische Differentialgleichungen für ein Einspurfahrzeugmodell realisiert (siehe Kapitel 3.1).
Im Kapitel 3.2 werden Matlab-Modelle gezeigt, mit denen das Einspurfahrzeugmodell simuliert wird. In der folgenden Einleitung wird Basiswissen aus dem Bereich der Mobilen Robotik beschrieben. Sie ist die Grundlage des Einspurfahrzeugmodells und der Bahnplanung.

Das Fahrzeug kann sich nicht auf der Stelle drehen und benötigt dafür Zeit, deshalb kann eine Bahnplanung nicht nur die X/Y-Koordinaten als Grundlage verwenden, sondern muss immer den Achswinkel θ mit berücksichtigen. Weiterhin benötigt eine Bewegungsregelung Informationen über den momentanen Lenkwinkel α und die Geschwindigkeit vm. Um das Mathematisch zu erfassen werden hier folgende Mengenbegriffe eingeführt:

- Der **Zustand** des Fahrzeugs $z = \{\theta, x_p, y_p, s, v, \alpha\}$, $z \in X$ beschreibt den Zustand des Fahrzeugs zu einem Zeitpunkt, mit der X/Y-Koordinate der Hinterachse x_p, y_p, Achswinkel θ, Lenkwinkel α, Geschwindigkeit v und dem zurückgelegtem Weg s. Als Zustand wird in der folgenden Arbeit verkürzt als $z = \{\theta, x_p, y_p, s\}$ definiert.
- **Arbeitsraum**[24] $W \in \Re^2$: Die Ebene / Umgebung in der sich das Fahrzeug bewegen kann und Hindernisse den Fahrweg beeinflussen.
- **Konfigurationsraum**[24] C beinhaltet alle Konfigurationen des Fahrzeugs im Arbeitsraum mit dem Achswinkelintervall $[-180°, +180°)$.
- **kollisionsfreier Konfigurationsraum**[24] C_{free} : Konfigurationsraum des Fahrzeugs, in der keine Kollision statt findet.
- **Hindernis-Konfigurationsraum**[24] C_{obs} sind verbotene Konfigurationen des Fahrzeugs, die Kollisionen mit Hindernissen beinhalten. Verwendet wird der Raum C_{obst} zur Beschreibung von Eckpunkten eines Hinderniselements.
- Die **Konfiguration**[11] des Fahrzeugs $q = \{x_p, y_p, \theta\}, q \in C$ beschreibt die minimale und eindeutige Lokalisierung des Fahrzeugs in C_{free} und ein Hindernis in C_{obst}. Wobei x_p und y_p die Koordinaten der Hinterachse und θ der Achswinkel des Fahrzeugs ist. Die Hinterachse des Fahrzeugs ist nicht beweglich, deshalb ist der Lagewinkel der Hinterachse gleichzeitig der Achswinkel.
- **Zustandsraum**[34] X der Raum aller kollisionsfreien Konfigurationen des Fahrzeugs vereinigt mit dem Steuerungsaktionsraum und dem Raum des zurückgelegten Weges s in $\Re$, zusätzlich werden die X/Y-Koordinaten der Vorderachse berechnet.
- Der **Steuerungsaktionsraum** U beinhaltet Geschwindigkeiten v im Intervall $[v_{min}, v_{max}]$ und Lenkwinkel α im Intervall $[-\alpha_{max}, \alpha_{max}]$.
- Mit **Steuerungsaktionen**[34] u = $\{v, \alpha\}$, $u \in U$ wird auf die Konfiguration des Fahrzeugs Einfluss genommen. Steuerungsaktionen sind Geschwindigkeit v und Lenkwinkel α.

In der folgenden Arbeit wird die Konfiguration q zur Beschreibung von Fahrzeugpositionen (C_{free}) und Hindernispositionen C_{obst} verwendet. Eine Pfadplanung muss das Fahrzeug aus einer Startkonfiguration q_{start} in eine Zielkonfiguration q_{ziel} überführen und dabei sicherstellen dass der Weg kollisionsfrei (C_{free}) ist.

Das Fahrzeug ist in der Bewegungsfreiheit eingeschränkt, es kann sich nicht drehen oder seitlich bewegen. Sie werden in der Literatur als nicht holonome Fahrzeuge bezeichnet.

- **Holonome Fahrzeuge** haben keine Einschränkung in der Bewegungsfreiheit. Ein Beispiel ist ein Hovercraft Fahrzeug[34] mit 3 Steuerungsparametern $\{v_x, v_y, v_\theta\}$ und 3 Konfigurationsparametern $\{x, y, \theta\}$. Es kann sich drehen, sowie seitwärts und vorwärts/rückwärts bewegen.
- **Nicht holonome Fahrzeuge** sind in der Bewegungsfreiheit eingeschränkt, wie das beschriebene Fahrzeug dieser Arbeit, mit den 2 Steuerungsgrößen $\{v, \alpha\}$ und den 3 Konfigurationsgrößen $\{x, y, \theta\}$.

3.1 Kinematische Einspurfahrzeugmodell für nicht holonome Fahrzeuge

Für die Bahnplanung ist es notwendig zu wissen, wo sich das Fahrzeug im Arbeitsraum befindet. Die Konfiguration q beinhaltet die minimale Anzahl von Zustandsgrößen, die eine eindeutige Lage des Fahrzeugs beschreiben. Das nicht-linearen kinematische Einspurfahrzeugmodell(siehe Abb. 7) beschreibt die Änderung der Konfiguration q. Das Einspurfahrzeugmodell basiert auf einem Fahrzeugmodell mit langsamer Lenkung[34]. Es ist eine vereinfachte Beschreibung der Fahrzeugkinematik, indem die Vorder- und Hinterräder auf einer Achse zu einem Punkt zusammen geschoben werden. Außerdem wird der Lenkwinkel der Vorderräder zusammengefasst. Es ist ein kinematisches Modell, das die Querkräfte außen vor lässt. Das ist die Grundlage, die das Planen von Fahrzeugbewegungen vereinfacht.

Das Kinematische Einspurfahrzeugmodell wurde in[31] und [25] für ein Frontangetriebenes Fahrzeug entwickelt wurde und wird für ein Allradangetriebenes Fahrzeug angepasst.

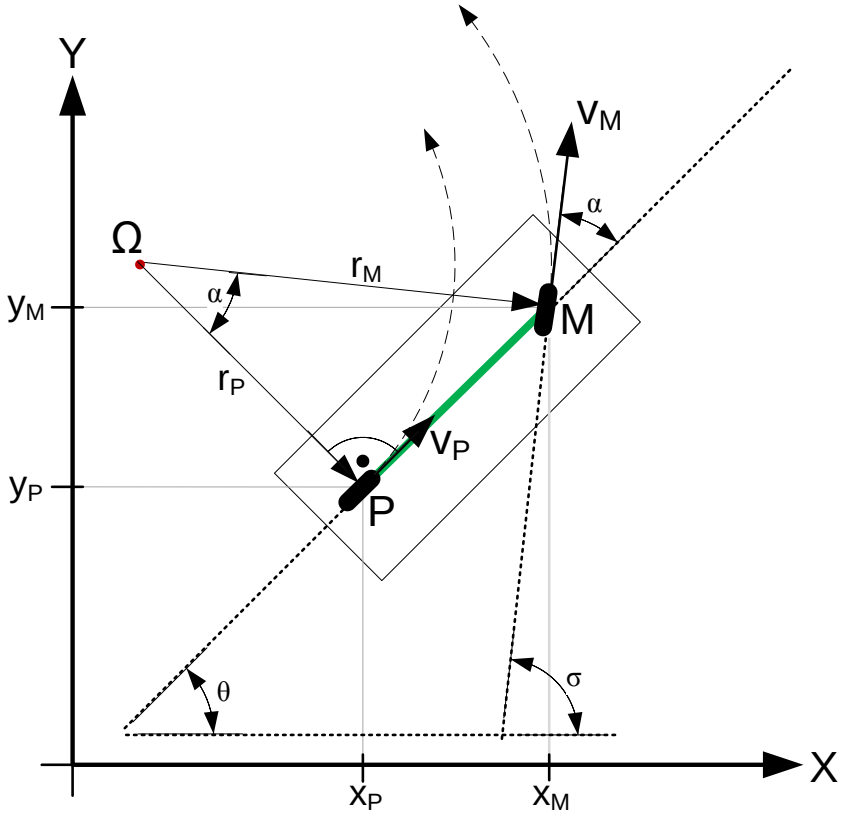


Abb. 7: *Kinematische Einspurfahrzeugmodell mit zusammengeschobenen Vorder- und Hinterräder*

Das Fahrzeug hat vier Räder, sie werden auf der Achse zu einem Rad zusammengeschoben. Es

ist der Mittelpunkt zwischen zwei Rädern auf einer Achse. In der folgenden Tabelle sind die verwendeten Symbole des Modells erläutert.

Symbol	Definition
M	Mittelpunkt des zusammengeschobenen Vorderrades
P	Mittelpunkt des zusammengeschobenen Hinterrades
Ω	Mittelpunkt der Wenderadien rm und rp
r_M	Wenderadius des zusammengeschobenen Vorderrades
r_P	Wenderadius des zusammengeschobenen Hinterrades
v_M	Geschwindigkeit des Hinterrades
v_P	Geschwindigkeit des Vorderrades
α	Lenkwinkel der Vorderachse, $\alpha < 0$ ist definiert als Linksfahrt
θ	Achswinkel des Fahrzeugs
σ	Zielwinkel, $\theta - \alpha$
L	Radstand, Entfernung zwischen Vorder- zur Hinterachse

Tabelle 1: *Größen des kinematischen Einspurfahrzeugmodells*

Der Wenderadius ist der Radius, den ein Fahrzeug mit einem konstanten Lenkwinkel abfährt. Der Mittelpunkt der Wenderadien rp und rm ist identisch. Zu beachten ist, dass der Lenkwinkel α bei einer Linksfahrt negativ und bei einer Rechtsfahrt positiv definiert ist.

Differentialgleichungssystem des Fahrzeugmodells

Das in diesem Projekt verwendete kontinuierliche Differentialgleichungssystem wurde in [25] und [31] für ein Fahrzeug aufgestellt (vgl. Gl. 1), in dem die Vorderradgeschwindigkeit einfließt. Eine zeitdiskrete Form des DGS wurde in [31] aufgestellt und in [26] in VHDL implementiert.

$$\begin{pmatrix} \dot{\theta}(t) \\ \dot{x_M}(t) \\ \dot{y_M}(t) \\ \dot{x_P}(t) \\ \dot{y_P}(t) \end{pmatrix} = \begin{pmatrix} \frac{vm(t)\cdot\sin[-\alpha(t)]}{L} \\ v_M(t)\cdot\cos[\theta(t)-\alpha(t)] \\ v_M(t)\cdot\sin[\theta(t)-\alpha(t)] \\ v_M(t)\cdot\cos[\alpha(t)]\cdot\cos[\theta(t)] \\ v_M(t)\cdot\cos[\alpha(t)]\cdot\sin[\theta(t)] \end{pmatrix} \tag{1}$$

In der Literatur[34] existieren zwei Schreibweisen des Fahrzeugmodells. Der Unterschied ist die eingehende Größe der Geschwindigkeit, entweder der Vorder- oder Hinterradgeschwindigkeit.

Umwandlung der Hinter- zur Vorderradgeschwindigkeit.
Die Geschwindigkeit wird kann bei einem 4WD-Fahrzeug nicht direkt aus der Drehzahl des Motors ermittelt werden, weil Differentiale die Drehzahl auf alle Räder unterschiedlich verteilen. Aus diesem Grund muss die Geschwindigkeit über einen Inkrementalgeber ermittelt werden. Das vorhandene DGS ist für ein Frontangetriebenes Fahrzeug implementiert. Eine Umformung der Geschwindigkeit ist eine Alternative zur Implementierung . Die Winkelgeschwindigkeit ist der Winkel, der pro Sekunde abgefahren wird. Er ist bei unterschiedlichen Entfernungen/Radien zum Mittelpunkt des Kreises konstant (vgl. Gl. 2).

$$\omega(t) = \frac{v_M(t)}{r_M(t)} = \frac{v_P(t)}{r_P(t)} \tag{2}$$

Aus der Umformung (siehe Gl. 2) lässt sich nun die Vorderradgeschwindigkeit wie folgt angeben:

$$\begin{aligned} v_M(t) &= \omega(t) \cdot r_M(t) \\ v_M(t) &= \frac{v_P(t) \cdot r_M(t)}{r_P(t)} \end{aligned}$$

Der Radstand L ist, dadurch ergibt sich die Beziehung von Hinterradradius zu Vorderradradius (siehe Abb. 7) über das Dreieck (vgl. Gl. 3) mit dem Lenkwinkel α.

$$\cos(\alpha(t)) = \frac{r_P(t)}{r_M(t)} \tag{3}$$

Über die Winkelgeschwindigkeit (vgl. Gl. 2) wird die Beziehung zwischen der Vorder- und Hinterradgeschwindigkeit zu den Radien aufgestellt:

$$\frac{r_P(t)}{r_M(t)} = \frac{v_P(t)}{v_M(t)}$$

Gleichung 4 in Gleichung 3 eingesetzt ergibt:

$$\frac{r_m(t)}{r_p(t)} = \frac{1}{\cos(\alpha(t))}$$

Umgeformt wird die Vorderradgeschwindigkeit über die Hinterradgeschwindigkeit und den Lenkwinkel angegeben:

$$v_M(t) = \frac{v_P(t)}{\cos(\alpha(t))}$$

Das Kinematischen Differentialgleichungen können in einander überführt werden. Zur Bahnplanung (vgl. Kapitel 4.2) wird nur die Konfiguration q verwendet, weil der Achswinkel orthogonal zum Kreismittelpunkt liegt (vgl. Abb. 7). Die Aufstellung des Differentialgleichungssystems 1. Ordnung mit der einfließenden Größe der Hinterradgeschwindigkeit:

$$(\dot{q}) = \begin{pmatrix} \dot{x_P}(t) \\ \dot{y_P}(t) \\ \dot{\theta}(t) \end{pmatrix} = \begin{pmatrix} v_P(t) \cdot \cos[\theta(t)] \\ v_P(t) \cdot \sin[\theta(t)] \\ \frac{1}{L} \cdot v_P(t) \cdot \tan(\alpha) \end{pmatrix} \tag{4}$$

Eine Implementierung des DGS für ein Frontangetriebenes Fahrzeug (siehe Gl. 1) steht dem FAUST Projekt zur Verfügung. Zur Nutzung einer Geschwindigkeit ist es präziser die Hinterradgeschwindigkeit des Einspurfahrzeugmodells zu verwenden, weil die Hinterachse starr ist. Über einen Inkrementalgeber kann die Geschwindigkeit eines Seitenrades gemessen und die Winkelgeschwindigkeit berechnet werden.

Das Bahnplanungsverfahren Kreistechnik verwendet Bogenlängen mit der Kreise gefahren werden, aus dem zurückgelegten Weg (vgl. Gl. 5) kann das Fahrzeug einen an einem Kreiswinkel anhalten:

Zusammenfassend realisiert das Kinematische Einspurfahrzeugmodell die Lokalisierung des Fahrzeugs, es gibt auch Nachteile:

1. Fehler addieren sich über die Zeit auf

$$\dot{s_P}(t) = v_P(t) = \begin{cases} \frac{v_m(t) \cdot r_P(t)}{r_M(t)}, & \text{wenn } r_{m(t)} \neq 0 \\ v_m(t), & \text{wenn } r_{m(t)} \equiv 0 \end{cases} \tag{5}$$

Symbol	Definition	Wertebereich
x_P	X-Koordinate der Hinterachse	$\Re$
y_P	Y-Koordinate der Hinterachse	$\Re$
θ	Achswinkel	$(-\pi, +\pi]$
α	Ist-Lenkwinkel	$[-\alpha_{max}, \alpha_{max}]$
s	zurückgelegter Weg des Fahrzeugs	$\Re$
α_{soll}	Soll-Lenkwinkel	$[-\alpha_{max}, \alpha_{max}]$
vm	mittlere Geschwindigkeit der Vorderräder	$[v_{min}, v_{max}]$

Tabelle 2: *Symbolbeschreibungen und Wertebereich des DGS*

2. vereinfachte Einspurmodell, keine 100%-ige Genauigkeit
3. Querkräfte und Schlupf werden nicht behandelt
4. Verschleiß der Räder und mechanische Bauteile, die einen Einfluss auf die Fahreigenschaften des Autos haben werden nicht berücksichtigt

3.2 Simulationsumgebung in Matlab

Das Differentialgleichungssystem (siehe Gl. 4) ist nichtlinear und kann analytisch nicht eindeutig gelöst werden. Matlab bietet ein Numerisches Verfahren an, das diese Differentialgleichungen löst. Es ist eines der populärsten Mathematik-Programmen und besitzt eine eigene Skriptsprache, die Objektorientierte Techniken unterstützt. Zusätzlich sind viele Bibliothek-Funktionen vorhanden, unter anderen numerische Lösungsverfahren von Anfangswertproblemen. Ein Anfangswertproblem besteht aus einer Differentialgleichung die mit einem gegebenen Anfangswert gelöst werden soll. Das kann in der Regel nur mit einem Numerischen Verfahren angenähert werden.
Matlab bietet mit ODE45 ein Verfahren an, mit dem ein gewöhnliches Differentialgleichungssystem 1. Ordnung gelöst werden kann. ODE45 nutzt das Runge-Kutta-Verfahren 4. Ordnung, das die Euler-Methode und die Trapezintegration verwendet, indem zu jedem Abtastschritt diskrete Zwischenschritte bzw. Stützpunkte berechnet werden. Zusätzlich wird zu jedem Rechenschritt der Fehler angenähert. Um Rechenzeit zu sparen, versucht ODE die Abtastschritte groß zu wählen, die Schrittweite wird automatisch verringert, wenn der Fehler zu groß wird.

Es gibt in Matlab die Bibliotheks-Funktion ode45, sie erwartet als Parameter das DGS in Form einer Funktion mit Rückgabevektor die Lösung der Differentialgleichung zu einem Rechenschritt. Übergabeparameter sind: die Zeitpunkte zu denen ode45 ein Ergebnis berechnen soll, den Anfangswertvektor der Differentialgleichung, Simulationsoptionen und optionale Parameter für die DGS-Lösungsfunktion. Ode45 ruft die DGS-Funktion in jedem Rechenschritt auf um das Integral zu bilden.

Starten der Simulation mit ODE45:

```
[t, fhzg_state]=ode45(@fhzg_modell_fcn, tspan, fhzg_init, options, optVars);
```

Die Parameter folgend detailliert aufgeführt:

- **t**: Rückgabevektor der erfolgreiche berechneten Zeitpunkten
- **fhzg_state**: Die berechneten Integrale zu den Zeitpunkten t,
  Matrixdimension: length(t) X length(fhzg_init)
- **fhzg_modell_fcn**: Pointer auf die Funktion in der das DGS berechnet wird
- **tspan**: Zeitspanne als Spaltenvektor
- **fhzg_init**: Die Anfangswerte des zu lösenden Integrals
- **options**: Einstellungen für die Simulation
- **optVars**: optionale Parameterliste, die der Fahrzeugmodell-Funktion übergeben wird

Die DGS-Funktion berechnet die Differenzen zu einem Rechenschritt, der Rückgabevektor ist die berechnete Differenz.

Die Funktion zur Lösung des des DGS ist folgend als Codeausschnitt aufgeführt. Das DGS hat zusätzlich die Vorderradposition (xm, ym) und den Lenkwinkel α, der über ein Verzögerungsglied 1. Ordnung im Rahmen des FAUST Projektes erstellt wurde.

```
1  function fhzg_modell_dot = fhzg_modell(t, fhzg_state, L)
2
3  global vm alpha_soll            % globale Variablen
4  fhzg_modell_dot = zeros(7, 1);  % Vektor mit Nullen füllen
5
6  theta = fhzg_state(1,1);
7  alpha_ist = fhzg_state(6,1);
8  t_max = 0.29475;                % Median der Messwertreihe [s]
9  tau = t_max / 5;                % PT1-Glied
10
11 % Umrechnung Vorderradgeschwindigkeit nach Hinterradgeschwindigkeit
12 if(alpha_ist == 0)
13     vp = vm;
14 else
15     rm = L / sin(alpha_ist);
16     rp = L / tan(alpha_ist);
17     omega = vm / rm;
18     vp = rp * omega;
19 end
20
21 %%%%% Diffenrentialgleichungssystem 1. Ordnung %%%%%%
22 % 1. Achswinkel theta
23 fhzg_modell_dot(1,1) = 1/L*vm*sin(-alpha_ist);
24 % 2. Vorderrad xm
25 fhzg_modell_dot(2,1) = vm*cos(theta - alpha_ist);
26 % 3. Vorderrad ym
27 fhzg_modell_dot(3,1) = vm*sin(theta - alpha_ist);
28 % 4. Hinterrad xp
29 fhzg_modell_dot(4,1) = vm*cos(alpha_ist)*cos(theta);
30 % 5. Hinterrad yp
31 fhzg_modell_dot(5,1) = vm*cos(alpha_ist)*sin(theta);
32 % 6. Ist-Lenkwinkel alpha_ist
33 fhzg_modell_dot(6,1) = (alpha_soll - alpha_ist) / tau;
34 % 7. zurückgelegter Weg vom Hinterrad s_ist
35 fhzg_modell_dot(7,1) = abs(vp);
36
37 end
```

Der Rückgabevektor fhzg_modell_dot wird mit Nullen initialisiert und die Differentialgleichungen f°(x,t) werden berechnet. Die Variablen hier noch mal detailliert beschrieben:

- **t_ist**: Skalar von dem aktuellen Zeitpunkt
- **fhzg_state_last**: Spaltenvektor mit den letzten N-Ergebnissen
- **fhzg_modell_dot**: ist die Differenz zum letzten Rechenschrittergebnis und der Rückgabevektor der Funktion

ODE45 bietet eine Anpassung der Simulation über ODE-Options an. Mit denen kann auf die Simulation Einfluss genommen werden, wie z.B. Fehlertoleranz.
Hier nur eine Teilmenge von **ODE45 Optionen**, die für diese Simulation notwendig sind.

- **RelTol**: relative Fehlertoleranz zum vorhergehenden Zwischenwert
- **AbsTol**: absolute Fehlertoleranz
- **OutputFcn**: eine Funktion als Pointer übergeben, der nach jedem aktualisierten Integrationsschritt aufgerufen wird. Als Übergabeparameter enthält sie den Lösungsvektor und die Zeitpunkte tspan der Zwischenwerte. Aufgerufen wird sie jeden Integrationsschritt, der mit maxStep begrenzt werden muss
- **OutputSel**: definiert die Vektorgrenzen des Lösungsvektor, der als Parameter der OutputFcn übergeben wird
- **Events**: Funktionspointer, eine Funktion die Bedingungen beinhaltet und Bedingung aufzeichnet oder die Simulation abbricht
- **MaxStep**: maximale Integrationsschrittweite

Wenn keine Optionen verwendet werden, so muss eine leere Menge $\{\}$ als Parameter übergeben werden. Mit den Optionen kann die Simulation verbessert werden und sollte deshalb genutzt werden. Über die OutputFcn können weitere Funktionen eingebunden werden, wie z. B. die zeitdiskrete Trapezintegration[31].

Kontrollstrukturen:
Bedingungen können nicht direkt in der DGS-Funktion (siehe Code Fahrzeugmodell) eingebunden werden. Da sie nicht sequentiell aufgerufen werden, die Simulationszeitpunkte können somit wiederholt werden. Die Funktion wird in einer Schrittweite mehrfach aufgerufen, im worst case wird die alte Berechnung aufgrund eines Fehlers verworfen und wird nochmal ausgeführt. Oder die Schrittweite ist zu groß und die Bedingung wird niemals eintreten.

Aus diesen Gründen können Kontrollstrukturen nicht direkt in der DGS implementiert werden. Aber es gibt in Matlab dafür 2 Varianten zeitliche Bedingungen einzubinden.

1. **OutputFcn**: Eine Funktion die nach jedem Integrationsschritt aufgerufen wird und Kontrollstrukturen enthalten kann
2. **Events**: Bei einer Bedingung die Simulation abbrechen, Werte neu setzen und die Simulation mit neuen Werten initialisieren und starten

OutputFcn wird nach jedem Integrationsschritt ausgeführt. Als globale Variablen werden in der OutputFunction der Lenkwinkel α gesetzt. Das Fahrzeug hat nur alle 20 ms Zeit einen neuen Lenkwinkel einzustellen, aus diesem Grund ist eine Simulation der Virtuellen Deichsel näher an der Realität. Außerdem ist das Debuggen der Simulation schwierig, weil ODE bei Fehlern in der Zeit zurückspringen kann. Deshalb ist das zeichnen von Kurven und die Ausgabe von Debugdaten zur Fehlersuche wichtig. Um eine Kurve zu plotten werden mindestens 3 Zwischenwerte gebraucht. Dazu werden in einem Schieberegister die letzten 10 Zustandswerte

gespeichert und nach dem 10. Aufruf mit plot() gezeichnet. Um das Zeichnen zu verlangsamen wird mit pause(0.1) ein Delay von 0,1 Sekunden eingebracht.

```
function status = outputFunction_dchsl(t,fhzg_state,flag, Ziel, verst,
    max_winkel, gui)

global vm alpha_soll % globale Variablen
persistent xp_srg yp_srg xm_srg ym_srg count; % Statische Variablen

   if strcmp(flag, 'init') == 1
        % Initialisierung von Datenstrukturen
        % Initialisierung der Datenstrukturen
        xp_srg = zeros(1,10);
        yp_srg = zeros(1,10);
        xm_srg = zeros(1,10);
        ym_srg = zeros(1,10);

        count = 0;
        pause on;                % Pause-Funktionalität aktivieren
   elseif isempty(flag) == 1 %  ==> Integrationsschritte

        theta = fhzg_state(1,length(fhzg_state(1,:)));
        xm = fhzg_state(2,length(fhzg_state(1,:)));
        ym = fhzg_state(3,length(fhzg_state(1,:)));
        xp = fhzg_state(4,length(fhzg_state(1,:)));
        yp = fhzg_state(5,length(fhzg_state(1,:)));
        alpha_ist = fhzg_state(6,length(fhzg_state(1,:)));
        s_ist = fhzg_state(7,length(fhzg_state(1,:)));

        ydl = yp - Ziel.y;
        xdl = xp - Ziel.x;

        if(xdl > 0 || xdl < 0)  % Division durch Null
            sigma = atan(ydl*verst/xdl);
        else
            sigma = 0;
        end

        alpha_soll = sigma - theta;

        if(vm > 0)
            alpha_soll = -alpha_soll ;   % Vorwärtsfahrt
        else
            alpha_soll = alpha_soll;     % Rückwärtsfahrt
        end

        if(alpha_soll > pi)             % Überlauf-Wraparound
            alpha_soll = alpha_soll - 2 *pi;
        elseif(alpha_soll < -pi)
            alpha_soll = alpha_soll + 2 * pi;
        end;

        if(alpha_soll > max_winkel) % Lenkwinkel normieren
            alpha_soll =  max_winkel;
        elseif(alpha_soll < -max_winkel)
            alpha_soll = -max_winkel;
        end;

        xp_srg = [xp_srg(1,2:10) xp];
        yp_srg = [yp_srg(1,2:10) yp];
```

```
57          xm_srg = [xm_srg(1,2:10) xm];
58          ym_srg = [ym_srg(1,2:10) ym];
59
60          if(count == 9)              % Plotten
61              count = 0;
62              plot([xp_srg(1,:)],[yp_srg(1,:)],'r'); hold on;
63              plot([xm_srg(1,:)],[ym_srg(1,:)],'b'); hold on;
64              setGuiText(vm, s_ist, xp, yp, theta, alpha_soll, alpha_ist,
                    gui); hold on;
65              pause(0.01);
66          else
67              count = count + 1;
68          end;
69
70      elseif strcmp(flag, 'done') == 1
71          if(xp_srg(1,1) ~= 0)      % nur plotten wenn das (Rechts)
                Schieberegister auch komplett gefüllt ist
72              plot([xp_srg(1,:)],[yp_srg(1,:)],'r');hold on;      %
                    Hinterachse
73              plot([xm_srg(1,:)],[ym_srg(1,:)],'b'); hold on;     %
                    Vorderachse
74              pause(0.01);
75          end;
76      end
77
78      status = 0; %1 == Stop Simulation, 0 == Continue Simulation
79  end
```

Die Berechnung des Lenkwinkels ist im Kapitel erläutert. Die Outputfunktion dient in den Matlab-Modellen zum grafischen Debuggen von Konfigurationsgrößen, es werden nach 10 erfolgreichen Integrationsschritten die letzten Positionsdaten in Form einer Kurve gezeichnet und Debugdaten in eine GUI geschrieben.

Flag hat die Werte: init (nach der Initialisierung), leer [] (während der Simulationsphase) und done (wenn Simulation beendet wurde).
Y und t sind die bisherigen Lösungen.

Events
Aufgrund der Gleitkommaarithmetik nicht immer erreicht werden kann. Deshalb werden Events in Form von Nullstellen aufgestellt.
Der Benutzer hat die Wahl zwischen einer Aufzeichnung von Events und dem Abbruch der Simulation. Aufgezeichnete Events können nur im nach hinein ausgewertet werden. Der Abbruch der Simulation bewirkt, dass eine Aktion ausgeführt werden kann. Die Simulation kann hinterher mit den letzten Simulationszuständen neu gestartet werden.

Events werden über den Wert value ausgelöst. Sie sind Bedingungen, die bei einem Nullübergang ein Event auslösen. Folgend eine exemplarische Event gesteuerte Funktion event_dchsl.m:

```
1  function [value,isterminal,direction] = Event_dchsl( t, fhzg_state,
       next_position)
2
3  xp = fhzg_state(4,1);
4  yp = fhzg_state(5,1);
5
6  value(1)      = next_position.x - xp;
```

```
isterminal(1) =  1;
direction(1)  = 0;

value(2)      = next_position.y - yp;
isterminal(2) =  1;
direction(2)  = 0;

end
```

Es können mehrere Bedingungen aufgeführt werden, die Variablen sind Vektoren und müssen bei mehreren Bedingungen mit einem Index versehen werden.

- **value**: die Bedingung muss als Nullstelle aufgestellt werden
- **isterminal**: Simulationsanweisung, bestimmt ob Simulation abgebrochen wird
  0 = Simulation fortführen, 1 = Abbruch der Simulation
- **direction**: die Richtung in der die Nullstelle durchlaufen wurde
  -1 = vom Positivem ins Negative, 0 = jede Richtung, 1 = vom Negativen ins Positive

Mit den Event gesteuerten Funktionen kommen zusätzliche Rückgabeparameter des Funktionsaufrufes des ODE45 Solvers hinzu.

```
[t,fhzg_state,te,ye,ie]=ode45(@fhzg_modell_fcn,tspan,
                              fhzg_init,options,optVars)
```

- **te**: Zeitpunkt zu dem sich das Event ereignete
- **ye**: Lösungen zu dem Zeitpunkt te
- **ie**: Index des Bedingungsvektors, die ein Event auslöste

Folgend ein Code-Ausschnitt der Testbench einparkassistent.tb der Einparkphase Spurführung. Bekannt ist die Startkonfiguration (x,y, θ). Die Funktion Spurführung berechnet die nächste Zielposition und den Verstärkungsfaktor. Die Simulation wird mit ode45 gestartet, nach Beendigung sind die Rückgabevektoren: [t fhzg_state te ye ie] mit Ergebnissen gefüllt.

```
% Berechnung der nächsten Koordinate und des
[Ziel verst] = phase_Spurfuehrung(Start, fahrspurmitte);
    Verstärkungsfaktor

  vm = 0.1;                          % Vorwärts fahren
  next_position.x = Ziel.x;          % Event setzen
  next_position.y = -1;              % Event ausschalten
  next_position.weg = -1;
  t_max = -1;

  if(t_new ==0)                      % erste Simulation, Zeitvektor mit
     Null beginnen
     tspan = [0:dt:maxSteps*dt];
     fhzg_init = [Start.theta;Start.x+L*cos(Start.theta);Start.y+L*sin(
        Start.theta);Start.x;Start.y;0;0];          % [theta, xm, ym, xp
        , yp, alpha_ist, s]
  else
     tspan = [te(length(te)):dt:maxSteps*dt];              % die neue
        Simulation mit den Daten der letzten Simulation initialisieren
     last_index = length(fhzg_state(:,1));
     fhzg_init = [fhzg_state(last_index,1);fhzg_state(last_index,2);
        fhzg_state(last_index,3);fhzg_state(last_index,4);fhzg_state(
        last_index,5);fhzg_state(last_index,6);fhzg_state(last_index,7)
```

```
        ];          % Startzustandsvektor [theta, xm, ym, xp, yp,
        alpha_ist, s]
17  end
18  options = odeset('OutputFcn',@(t,fhzg_state,flag)outputFunction_dchsl
        (t,fhzg_state,flag, Ziel, verst, max_winkel, gui),'MaxStep', dt ,'
        Events', @(t, fhzg_state)Event_dchsl( t, fhzg_state, t_max,
        next_position),'Refine', 1) ;
19  [t fhzg_state te ye ie] = ode45(@(t, fhzg_state)fhzg_modell(t,
        fhzg_state, L), tspan, fhzg_init, options);
20
21  theta = fhzg_state(length(fhzg_state(:,1)),1);
22  xp = fhzg_state(length(fhzg_state(:,4)),4);
23  yp = fhzg_state(length(fhzg_state(:,5)),5);
24  alpha_ist = fhzg_state(length(fhzg_state(:,6)),6);
25  s_ist = fhzg_state(length(fhzg_state(:,7)),7);
```

Das Skriptfile berechnet mit der Funktion „phase_Spurfuehrung“ die Zielkoordinate und den Verstärkungsfaktor (vgl. Kapitel 4.3). Wenn die X-Koordinate des Ziels erreicht ist, dann wird die Simulation beendet (Zeile 4). In Zeile 11 und 15 wird der Anfangswertvektor mit dem Anfangs-Fahrzeugzustand initialisiert. Simulationsoptionen werden in Zeile 17 gesetzt. Das Starten der Simulation wird in Zeile 18 durchgeführt. Nach Beendigung der Simulation können über den Rückgabevektor $fhzg_state$ zu allen Integrationsschritten der Fahrzeugzustand ausgelesen werden. Das letzte Element (length(fhzg_state)) ist der Zustand des letzten erfolgreichen Integrationsschritts.

In Matlab gibt es Funktionen, die als Parameter einen Pointer auf eine Funktion verlangen (vgl. Zeile 17-18). Die Rückgabevektoren und Übergabevektoren müssen stimmen. Weitere Argumente müssten so über globale Variablen realisiert werden. Eine Methode zusätzliche Parameter zu übergeben ist es einen Pointer auf eine Anonyme Funktionen zu übergeben. Die als einzigste Aktion eine eigene Funktion aufruft, die eine längere Übergabe-Parameterliste und die gleichen Rückgabevektoren enthält. Das wird in der Matlab Dokumentation empfohlen und erhöht die Lokalität von Daten.

```
1 return_Wert = (Übergabe_Parameter) Ausdruck   %Anonyme Funktion
2 pointer = @(Übergabe_Parameter) Ausdruck      %Pointer auf Anonyme
      Funktion
3
4 odesolver = @old_odesolver
                              %Pointer auf eine Funktion mit dem Namen
      old_odesolver
5 odesolver = @(t, y) new_odesolver(t, y, var1, .., varN) %Odelöser mit
      mehreren Übergabeparameter var1,..varN
```

Die Numerische Lösung eines Differentialgleichungssystems kann präzise erfolgen. Das kann über die Simulationsoptionen absolute *absTol* und relative Fehlertoleranz *relTol* eingestellt werden. Die relative Fehlertoleranz ist auf den Standardwert 0,1% gesetzt.

4 Grundlagen der Bahnplanung

Dieses Kapitel behandelt grundlegende Verfahren die Pfade planen und das Fahrzeug entlang des optimalen Soll-Pfades regeln. Im Abschnitt 4.1 werden Bahnplanungsverfahren aus der Forschung analysiert. Sie dienen zur Übersicht zu Bahnplanungsverfahren die in der Forschung verwendet werden. Die weiteren Abschnitten Bahnplanungsverfahren Kreis-Technik (siehe 4.2) und das Bewegungs- und Bahnplanungsverfahren die Virtuelle Deichsel (siehe 4.3) behandeln die Verfahren, die zum Einparken Verwendung finden (siehe Kapitel 5).

4.1 Überblick Stand der Forschung

Das Thema Bahnplanung gehört zu dem Themengebiet „Mobile Robotik“. Es gibt zur Zeit kein Verfahren, das als Lösung für alle Fahrsituation verwendet wird. Vielmehr ist es ein offenes Themengebiet, auf dem zur Zeit stark geforscht wird. Die Verfahren gehen von Lösungen zur Bahnerzeugung von Kreisbahnen in der Ebene, bis hin zu Graphen die im mehrdimensionalen Raum Wege suchen. Im folgenden Abschnitten werden ausgesuchte Verfahren gezeigt:

Dubins Kurven

Folgender Abschnitt wird sinngemäß aus [24, LaValle 2006] zitiert. L. E. Dubins [14] hat 1957 eine Grundlage geschaffen, mit der der kürzeste Weg für Fahrzeuge beschrieben wird. Es ist kein Algorithmus, sondern eine Beschreibungssprache für Kreisfahrten von Fahrzeugen, die das Lenkverhalten vorgeben. Es gibt einige Einschränkungen / Besonderheiten die zur Vereinfachung dienen:

- das Fahrzeug kann nur vorwärts fahren
- der Lenkwinkel wird ohne Verzögerung umgestellt
- es gibt nur 3 Bewegungsrichtungen: Vorwärts, Links und Rechts
- der Lenkwinkel ist entweder 0 Grad (Geradeaus), minimaler (Linksfahrt) oder maximaler Lenkwinkel (Rechtsfahrt)
- die Geschwindigkeit des Fahrzeuges ist konstant
- der Bewegungsraum des Fahrzeuges ist kollisionsfrei

Der kürzeste Pfad besteht aus 3 Segmenten (vgl. Abb. 8), sie bestehen aus Kreisfahrten C und aus Graden S.

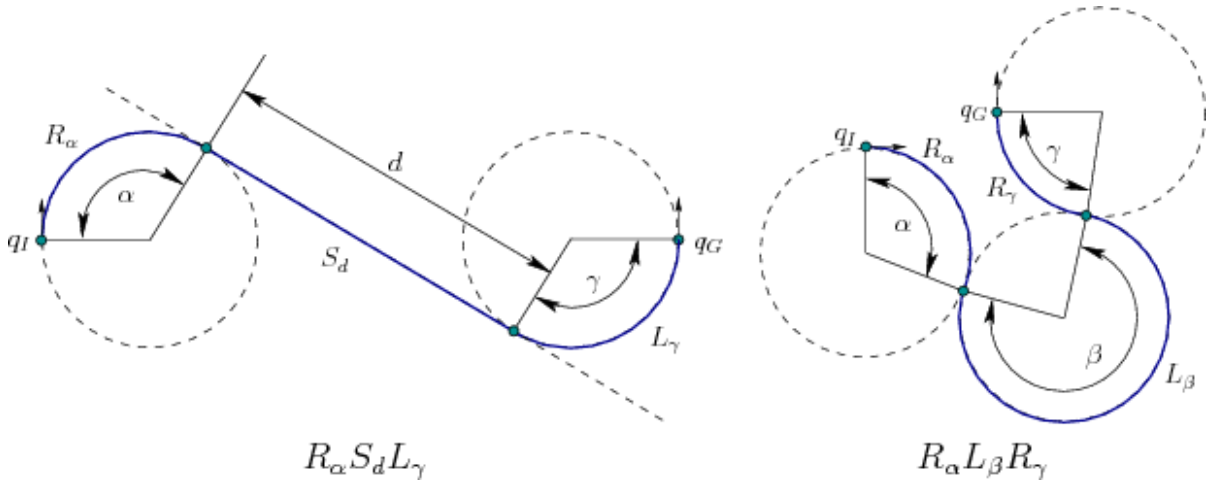


Abb. 8: *In der linken Abbildung ein CSC-Pfad, bestehend aus einer Rechtskurve, Gerade und einer Rechtskurve. Und Rechts eine CCC-Pfad, aus Rechts-, Links- und Rechtskurve [24, LaValle 2006]*

Der minimale Weg wird mit einer Folge von drei Lenkkommandos beschrieben. Lenkkommandos sind: L = Linksfahrt, S = Geradeausfahrt, R = Rechtsfahrt. Bei 3 Variablen mit der Länge 3 ergeben sich 27 Kommandoworte, um einen minimalen Weg zu beschreiben. Jedoch nur 6 Kommandos sind dafür ausreichend, da z.B. ein Wort LLL keinen Sinn machen würde.

$$\{LRL, RLR, LSL, LSR, RSL, RSR\} \tag{6}$$

Mit Angabe des zu fahrenden Winkels α, β, γ und Strecke d:

$$\{L_\alpha R_\beta L_\gamma, R_\alpha L_\beta R_\gamma, L_\alpha S_d L_\gamma, L_\alpha S_d R_\gamma, R_\alpha S_d L_\gamma, R_\alpha S_d R_\gamma\} \tag{7}$$

Außerdem wurde in [14] eine Einschränkung des Winkels α, β, γ und Strecke s bewiesen, so wird die Suche des kürzesten Pfad auf einen Wertebereich eingeschränkt:

$$\alpha, \gamma \in [0, 2\pi), \beta \in (\pi, 2\pi), d \geq 0 \tag{8}$$

Eine Vereinfachung ist die Ersetzung der Kommandos für Links- und Rechtsfahrt, mit einem Kommando C, für Kreisfahrt. Das erleichtert die Sicht und macht die Beschreibung abstrakter.

$$\{C_\alpha S_d C_\gamma, C_\alpha C_\beta C_\gamma\} \tag{9}$$

Der wesentliche Vorteil gegenüber anderen Techniken wie Splines ist, dass die Kreisbahnen immer eine maximale Krümmung besitzen (vgl. Gl. 11). Das kann mit Splines nur Abschnittweise erreicht werden. Damit kann mit der Weg-Bogenlängen-Technik der kürzeste Weg beschrieben werden und ist die Grundlage vieler Forschungsarbeiten. Ein wesentlicher Nachteil ist, dass Dubins-Kurven nur für Vorwärts fahrende Fahrzeuge den kürzesten Weg ermitteln.

Reeds and Shepps Kurven

1990 haben J. A. Reeds und L. A. Shepp[29] eine Erweiterung der Dubins Kurven mit der Rückwärtsfahrt veröffentlicht. Damit kann der kürzeste Weg eines Fahrzeugs beschrieben werden, das Vorwärts und Rückwärts fahren kann. Allerdings nur für den kürzesten Weg in einem kollisionsfreien Raum und mit der erneuten Einschränkung, dass der Lenkwinkel ohne Verzögerung umgestellt werden kann.

Die Notation der Lenkkommandos sind:
L^+ = Vorwärts- und Linksfahrt, L^- = Rückwärts- und Linksfahrt,
S^+ = Vorwärtsfahrt, S^- = Rückwärtsfahrt,
R^+ = Vorwärts- und Rechtsfahrt, R^- = Rückwärts- und Rechtsfahrt

Eine vereinfachte Schreibweise mit Angabe von Kreisbahnen:
C^+ = Vorwärts- und Kreisfahrt, C^- = Rückwärts- und Kreisfahrt,
S^+ = Vorwärtsfahrt, S^- = Rückwärtsfahrt

Und in einer kompakten Form:

C = Kreisfahrt, S = Vorwärtsfahrt, | = Richtungswechsel

Die Anzahl der Kommandoworte um den kürzesten Pfad zu beschreiben sind 48.

$$\{C \mid C \mid C,CC \mid C,C \mid CC,CSC,CC \mid CC,C \mid CC \mid C,C \mid C_{\pi/2}SC,CSC_{\pi/2} \mid C,C \mid C_{\pi/2} \mid C\} \quad (10)$$

Die Reeds and Shepps Kurven beschreiben den kürzesten Weges für Vorwärts- und Rückwärtsfahrt. Jedoch der kürzeste Pfad ist nicht immer der schnellste Pfad, wegen den Anhaltepunkten zwischen den Segmenten.

CC Technik

Kennzeichen der CC-Technik ist die Eliminierung von Anhaltepunkten zwischen der Start- und Endkonfiguration des Fahrzeugs auf einem Teilpfad. Es sind Bahnsegmente die eine kontinuierliche Krümmungsänderung besitzen. Über die Continuous Curvature Technik können Reeds und Shepps Kurven über ein Verbindungsstück verbunden werden. Die Verbindungssegmente stellen eine Alternative zum Anhalten dar. Dadurch ist der Pfad nicht mehr der kürzeste. Über die Kombination der Reeds and Shepps Kurven mit den CC Kurven wird ein Pfad erzeugt, der auf den kürzesten Weg aufbaut. Ein Klothoidenstück ist ein Pfad, der bei konstanter Geschwindigkeit gefahren und kontinuierlich umgelenkt wird. In der Abbildung 9 ist dargestellt,

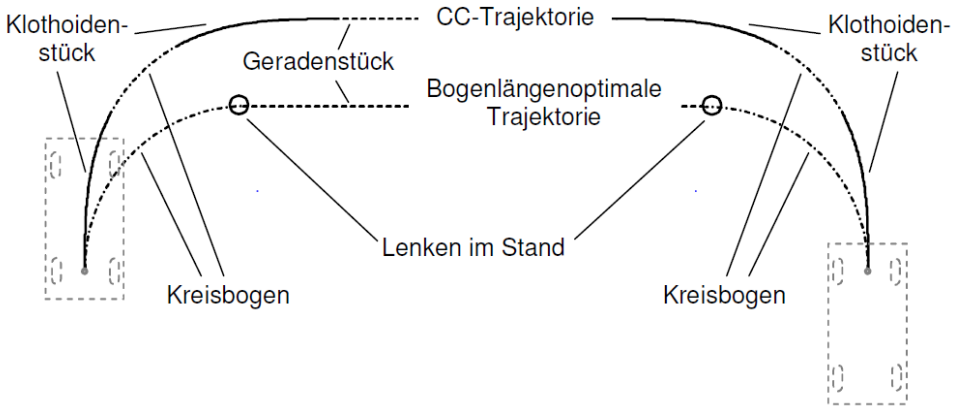


Abb. 9: *Vergleich von Kreisbögen mit mit CC-Kurven [10]*

dass der Weg mit einer CC-Trajektorie länger wird, jedoch die Anhaltepositionen zwischen den Kreisbahnen verschwinden. Ein CC-Bahnsegment 9 fährt meist geringer, weil das Anhalten und Umlenken bei der Verbindung von Kreisbahnen Zeit beansprucht. Die Größen Fahr- und Lenkwinkelgeschwindigkeit nehmen auf die Länge eines CC-Segments Einfluss. Die Berechnung von Klothoiden ist in der Arbeit Emese Szadeczky-Kardoss und Balint Kiss[15] beschrieben. Es wird gezeigt, wie Klothoiden-Segmente berechnet werden können. Den Fokus haben sie auf die Echtzeitberechnung der Klothoidensegmente gelegt.

In der Arbeit von Thierry Fraichard und Alexis Scheuer[19], wird eine Methode vorgestellt, die Reeds and Shepps Kurven mit CC-Turns erweitern.

Die Krümmung ist der Kehrwert des Wenderadius rp des Fahrzeugs:

$$k = \frac{1}{rp} \quad (11)$$

Ein CC-Turn ersetzt eine Kreisbahn mit maximaler Krümmung (vgl. Gl. 11), sie besteht aus einer Klothoide, eine Kreisfahrt mit maximalem Lenkwinkel und einer weiteren Klothoide. Sie kann als Verbindungsstück zwischen zwei Geraden verwendet werden. Der einzustellende Lenkwinkel kann nur 3 Werte annehmen: $-\alpha_{max}, 0, \alpha max$. Das umschalten von den drei Lenkwinkelwerten bewirkt in Abhängigkeit von der Lenkwinkel- und Fahrzeuggeschwindigkeit eine Klothoide. Eine Klothoide zwischen zwei Kreisbahnen mit positiver und negativer Krümmung wird Wendelinie genannt. Sie findet vor allem im Straßenbauwesen Verwendung.

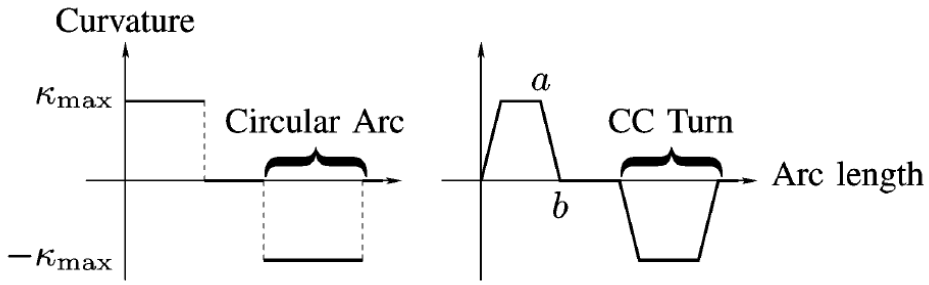


Abb. 10: *Krümmungsband einer Teilstrecke [19], CC-Bahn zwischen A und B, CC-Turn besteht aus zwei CC-Bahnen und einer Kreisbahn*

In der Abbildung 10 ist im Krümmungsband die Funktionsweise eines CC-Turns dargestellt. Die minimale und maximale Krümmung, wird über den minimalen und maximalen Lenkwinkel bestimmt (vgl. Gl. 11). In der linken Abbildung ist ein Reeds and Shepps Pfad dargestellt, der aus einer Kreisbahn, Gerade und entgegengerichtete Kreisbahn besteht. Zwischen den Segmenten muss das Fahrzeug anhalten und im Stehen umlenken. In der rechten Abbildung ist ein Pfad mit Klothoiden, die als Schrägen zu erkennen sind.

Ein Nachteil von CC-Trajektorien ist die aufwendige Berechnung von Koordinaten auf der Klothoide. Das muss vorab Offline durchgeführt werden. Außerdem sei hier noch auf eine weitere Arbeit von Doran K. Wilde[35] hingewiesen.

Weg-Bogenlängen Metriken

Zur Kollisionserkennung und -vermeidung werden Informationen über die Erreichbarkeit von Hindernissen gebraucht. Ein Hindernis seitlich zum Fahrzeug ist aufgrund des Achswinkels und der Lenkwinkelverzögerung schwerer zu erreichen, als ein Hindernis, das direkt vor dem Fahrzeug liegt. Die kürzeste Entfernung kann nicht einfach über den Satz des Pythagoras ermittelt werden. Denn die Entfernung (Weg) ist abhängig vom Achswinkel am Start und am Ziel. Um die kürzeste Entfernung zu berechnen, gibt es Shortest-Feasible-Path-Metriken (SFP) oder auch Weg-Bogenlängen Metriken.

Dubins Metrik.

In der Arbeit von Lumelsky und Shkel[32] wurde ein Verfahren entwickelt, mit dem CSC-Pfade erzeugt werden. Die CCC-Kurven (vgl. Gl. 9) stellen eine Ausnahme dar und können nur entstehen, wenn die Entfernung zwischen der Anfangs- und Endkonfiguration (x, y, θ) kleiner als das vierfache des minimalen Wenderadius ist (siehe Lumelsky und Shkel [32]). Es ist eine Shortest-Path-Metrik für diese Entfernungen.

In Abb. 11 wird der kürzeste Pfad dargestellt, der von Startkonfiguration **I** zu Endkonfiguration **F** führt. Außerdem noch ein Pfad in der Form CCC, der jedoch nicht kürzer ist, obwohl der

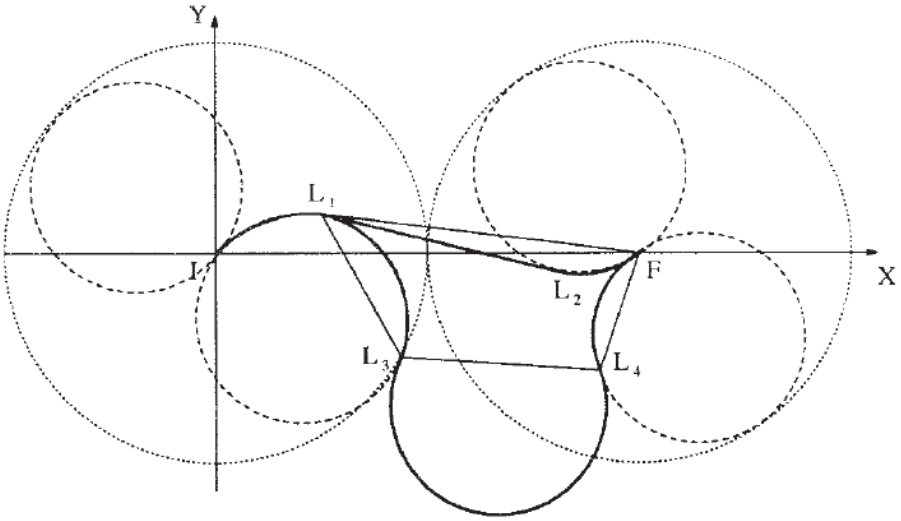


Abb. 11: *Kürzester Pfad RSL zwischen 2 Konfigurationen, CSC und CCC [32]*

Final Quadrant / Initial Quadrant	1	2	3	4
1	rsl	rsl or {rsr or lsr}	{rsr or lsr}	{rsr or lsr}
2	rsl or {lsl or lsr}	rsl or {rsr or lsl}	rsr	{rsr or rsl}
3	{lsl or lsr}	lsl	lsr or {lsl or rsr}	lsr or {rsr or rsl}
4	{lsr or rsl}	{lsl or rsl}	lsr or {lsl or rsl}	lsr

Abb. 12: *Tabelle mit CSR-Kurven, bei unterschiedlichen Winkeln [32]*

Abstand zwischen den beiden Konfigurationen kleiner als 4 mal minimaler Wenderadius ist. Die Konstruktion von CSC-Kurven erfolgt über das ziehen von zwei Kreisen, jeweils zur Anfangs- und zur Zielkonfiguration, damit hat man schon 2 C-Kurven. Was fehlt ist noch eine Gerade, die bei einer RSL oder LSR - Kurve einer Inneren Tangente entspricht. Und bei einer LSL und RSR-Kurve ist es eine Äußere Tangente. Somit gibt es acht Pfade, die mit Hilfe der Inneren und Äußeren Tangente ermittelt werden.

Die Tabelle 12 von [32] beschreiben Bahnen, bei denen die CSC-Kurven minimiert wurden. Sie zeigen die Kombinationen von Kreisbahnen, die den kürzesten Pfad beschreiben. Zu beachten ist, dass die Berechnung der Winkel nicht über die X-Achse als Winkelbezugsachse erfolgt, sondern über die Verbindungslinie zwischen den beiden Konfigurationen.

Weg-Bogenlängen Pfad-Planer

Einparken mit Entscheidungslogik

Die Arbeit von Ollero, Cuesta und Gomez-Bravo[17] beschreibt ein Bahnplanungsalgorithmus

mit Entscheidungslogik. Es werden mehrere kollisionsfreie Pfade generiert, wobei der Pfad aus SCC-Pfadsegmenten besteht. Sie bestehen aus einer Kreisbahn *C* mit maximaler Krümmung zur Einparkposition und mehreren Kreisbahnen *C* von der Startposition zu den Berührungspunkten. Eine Gerade *S* führt zur ersten Kreisbahn in Richtung Parklücke. Der Radius der Kreisbahn von der Startposition aus besitzt eine maximale Krümmung. Bei einer ausreichend großen Parklücke werden so viele Kreisbahnen generiert, der Berührungspunkt der beiden Kreisbahnen ist jeweils der Anhaltepunkt, an dem der Lenkwinkel im Stehen umgestellt wird. Eine Fuzzy-Logik entscheidet aufgrund von Erfahrungswerten, welche die beste ist.

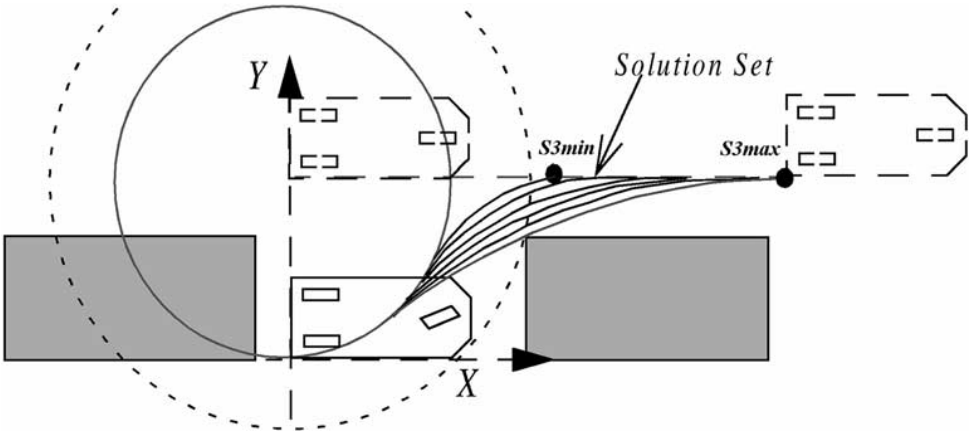


Abb. 13: *Generierte Pfade, die abfahrbar sind[17]*

Es ist eine Bahnplanung, mit der ein kurzer Pfad erzeugt wird und anschließend eine Entscheidung getroffen wird. Jedoch behandelt diese Arbeit nur den ersten Einparkschritt (siehe Kapitel 4.2) des Einparkmanövers.

Wegplanung über einen gerichteten Graphen Ein Konzept zur Wegplanung von A. Bicchi, G. Casalino und C. Santilli[9] verwendet zur Wegfindung einen Graphensuche.

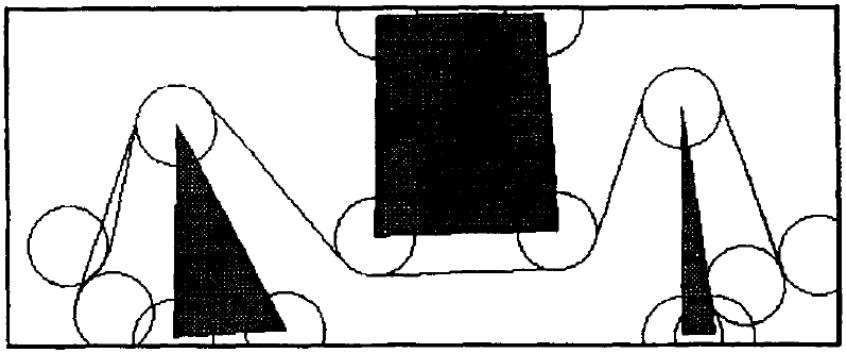


Abb. 14: *Erzeugter Pfad entlang den Eckpunkten der Hindernisse [9]*

Der Algorithmus enthält folgende Schritte:

1. Jeweils zwei Kreise um die Start- und Zielkonfiguration ziehen
2. an allen Hindernis-Eckpunkten einen Kreis ziehen mit dem Radius der das Maximum von minimalen Wenderadius und Mindestabstand zum Hindernis entspricht
3. alle Kreise mit (Inneren und Äußeren) Tangenten verbinden
4. alle Tangenten streichen, die eine Kollision verursachen
5. Start-, Zielkoordinate und alle Tangentenpunkte sind Knoten
6. die Kosten von Kanten entsprechen dem Weg
7. einen Algorithmus zur Wegfindung verwenden, wie z.B. Dijkstra

In einer Umgebung mit vielen Ecken, müssen viele Kreise gezogen werden und der Aufwand des Algorithmus ist dann groß. Zum Einparken kann das Verfahren nur verwendet werden, wenn die Parklückenlänge mindestens doppelt so breit wie der Wenderadius sein muss. Da die Kreise sich sonst überlagern.

Eine Verbesserung des Algorithmus ist, dass die Kreismittelpunkte nach Ablauf des Algorithmus in Richtung Hindernis verschoben werden. Da der minimale Wenderadius des Fahrzeuges meist viel größer ist, als der minimale Abstand des Fahrzeuges zum Hindernis (siehe Kollisionsradius Abb. 16).

Kombination von erprobten Verfahren. Die Arbeit von B. Müller, J. Deutscher und S. Grodde[10][11][12][13] umfasst mehrere Verfahren aus vielen Forschungsarbeiten, sie haben diese zusammen gebracht und zu eine Allzweck-Lösung für unterschiedlichste Umgebungen gemacht. Sie liefert beim Einparken in sehr tiefe Parklücken einen kurzen Weg. Da ist es besser im ersten Einparkschritt eine große Tiefe zu erreichen und danach den Achswinkel des Fahrzeug auszurichten.
Der Algorithmus besteht aus zwei Schritten: 1) kollisionsfreier Pfad zum Ziel generieren, ohne die Beschränkungen in der Bewegungsfreiheit zu beachten (holonome Fahrzeugpfad), 2) Umwandlung des Pfades in einen Pfad für nicht holonomische Fahrzeuge.

1. Zuerst wird zu allen Hinderniskoordinaten die kürzeste Bogenlänge über eine SFP-Metrik ermittelt. Die Hindernisse werden als als eine Menge von Hinderniselementen aufgefasst, die mit Angabe von einem Winkel nur verschoben sind. So können diese Abstandswerte Offline berechnet werden und je nach Hinderniselement und Abstand wird so der kürzeste Pfad über eine Lookup Tabelle bezogen. Daraus werden zwei Teilpfade erzeugt. Einer von der Startkonfiguration in Richtung Ziel und ein Teilpfad von der Zielkonfiguration zur Startkonfiguration. Die Teilpfade werden unter aufgrund von großen Entfernungen zu den Hindernissen, jedoch in Richtung des Ziels generiert. Damit ist sichergestellt, das ein Teilpfad wenige Fahrtrichtungswechsel beinhaltet.
2. Der kollisionsfreie Pfad wird in einen fahrbaren Pfad umgerechnet.

Das Konzept ist eine Komplettlösung, die dynamisch einen kurzen Pfad, mit wenigen Anhaltepunkten, generiert. Sie benutzen mehrere Pfadplaner, wobei der geeignetste Pfad ausgewählt wird. Es werden umfangreiche Kenntnisse zu den unterschiedlichsten Verfahren benötigt, um diese Komplettlösung zu übernehmen.

Wegfindung in Graphen

Ein weiteres Themengebiet sind Such-Algorithmen die einen Weg im Graphen suchen. Dazu wird ein Graph erzeugt der meist im höher dimensionierten Raum aufgespannt wird.

Der **RRT** [23] [22] Rapidly-exploring Random Tree- Algorithmus sucht im drei- oder mehrdimensionalem Zustandsraum (x,y,θ) oder auch (x,y,θ , v , ...) einen Pfad von der Startkonfiguration in Richtung Ziel und von der Zielkonfiguration in Richtung Startkonfiguration. Bei Dreidimensionalen Räumen wird von der Startkonfiguration des Fahrzeugs (x_{start} , y_{start}, θ_{start}) eine zufällige Konfiguration gewählt, von der bekannt ist dass sie im kollisionsfreien Raum liegt und erreichbar ist. Der Graph wird über die folgende Konfiguration oder weiteren erreichbaren Konfigurationen fortgeführt, bis beide Teilpfade verbunden sind. Die Kanten beinhalten die

Geschwindigkeit und den Lenkwinkel.

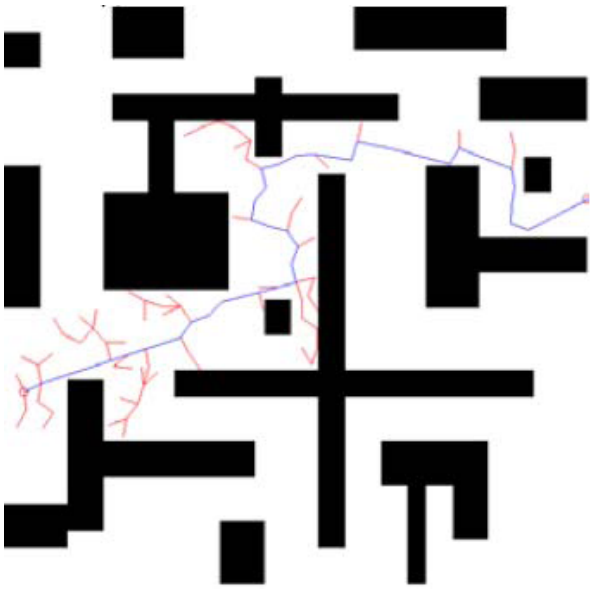


Abb. 15: *RRT Pfaderzeugung [28]*

Zu sehen ist in der Abbildung 15 eine erzeugte Baumstruktur, die sukzessiv zum Ziel aufgebaut wird. Der Algorithmus garantiert nicht die Qualität der Erreichbarkeit, Schnelligkeit und des kürzesten Pfades. Es ist ein Algorithmus der 1998 entwickelt wurde und seit dem Jahre 2000 wurden im IEEE 109 Facharbeiten veröffentlicht wurden (Stand 01.07.2010).

4.2 Kreistechnik

Folgender Abschnitt ist eine Einführung in die Kreistechnik, für eine detailliertere Beschreibung siehe Kapitel 5 Mathematischer Bahnplanungsentwurf der Einparkphasen, Abschnitt „Erster Einparkschritt“ und „N-Einparkschritte“. Die Kreistechnik ist ein Bahnplanungsverfahren, deren Pfade nur aus Kreissegmenten bestehen. Sie ist eine Vorstufe zu den Dubins Kurven, die auch Kreissegmente mit maximaler Krümmung besitzen. Jedoch ohne Geraden zwischen den Kreisen. Die Verbindungen zwischen den Kreissegmenten sind Anhalteposition, an denen das Fahrzeug im stehen auf den minimalen oder maximalen Lenkwinkel umlenkt.

Verwendung findet die Technik im ersten und n-ten Einparkschritt (siehe Kapitel 5 und 6).

Der erste Einparkschritt wird schon nach dem Erkennen der Parklücke geplant, indem eine Startkonfiguration berechnet wird, die entweder eine Bahn zur finalen Einparkkonfiguration oder eine maximale Tiefe erzielt. Kennzeichen des ersten Einparkschrittes sind:

- unbekannte Startkonfiguration
- bekannte Zielkonfiguration innerhalb der Mitte der Parklücke
- Hindernisse begrenzen die Parklücke

Zwei Szenarien sind zu unterscheiden:

1. Zielkonfiguration kann direkt ohne Kollisionen erreicht werden
2. Einparken in mehreren Einparkschritten, weil die Parklücke zu schmal ist

Bei Szenario 1. hat die Kreisbahn aus der Parklücke heraus, einen Berührungspunkt mit der finalen Einparkkonfiguration, ohne eine Kollision mit Hindernissen zu haben. Der Mittelpunkt des Kreises ist in Y-Richtung Fahrspur und hat die gleich X-Koordinate der finalen Einparkkonfiguration, da das Ziel mit einem Achswinkel von 0 Grad erreicht werden soll. Die andere Kreisbahn von der Fahrspur hat einen Berührungspunkt mit dem zuvor erzeugten Kreis und hat einen Berührungspunkt mit der Y-Koordinate des Fahrzeugs in einem Achswinkel von 0 Grad (vgl. Kapitel 5.3, Gl. 13 und Gl. 16). Das Fahrzeug muss zur Positionierung nur vor- oder zurücksetzen. In Szenario 2. ist die Parklücke zu schmal, der Kreis aus der Parklücke hat eine Kollision mit einem Hindernis. Es muss deshalb ein Weg in einem sicheren Abstand zu den Hindernissen konstruiert werden. Dazu werden die innere und äußere Kollisionsradien des Fahrzeugs verwendet (vgl. Abb. 16).

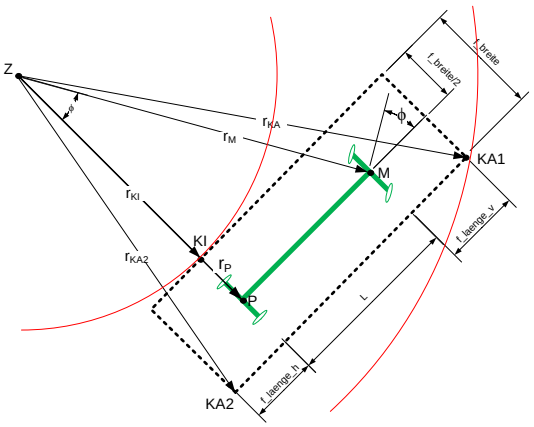


Abb. 16: *Kollisionsradien des Fahrzeugs, rot = innerer und äußerer Kollisionsradius*

Der Wenderadius des Fahrzeug wird wie folgt berechnet:

$$r_p = L/\tan(max_{winkel}) \tag{12}$$

Der äußere Kollisionsradius (vgl. Gl. 16) wird bei diesem Fahrzeug über den Radius r_{KA1} berechnet, da die Entfernung vom Hinterrad zum Heck f_laenge_h kleiner ist als die zur Fahrzeugfront $f\_laenge\_v + L$. Der äußere Kollisionsradius kann verringert werden, wenn die Fahrzeugfront gekürzt und / oder die Frontecken abgerundet werden.

$$r_{KI} = r_P - \text{f\_}breite/2 \tag{13}$$

$$r_{KA1} = \sqrt{(r_P + \text{f\_}breite/2)^2 + (L + f\_laenge\_v)^2} \tag{14}$$

$$r_{KA2} = \sqrt{(r_P + \text{f\_}breite/2)^2 + f\_laenge\_h^2} \tag{15}$$

$$r_{KA} = max(r_{KA1}, r_{KA2}) \tag{16}$$

Ein sicherer Weg beinhaltet ein Kreissegment mit einem Mindestabstand der Kollisionsradien zu dem Hindernis (vgl. Abb. 17).

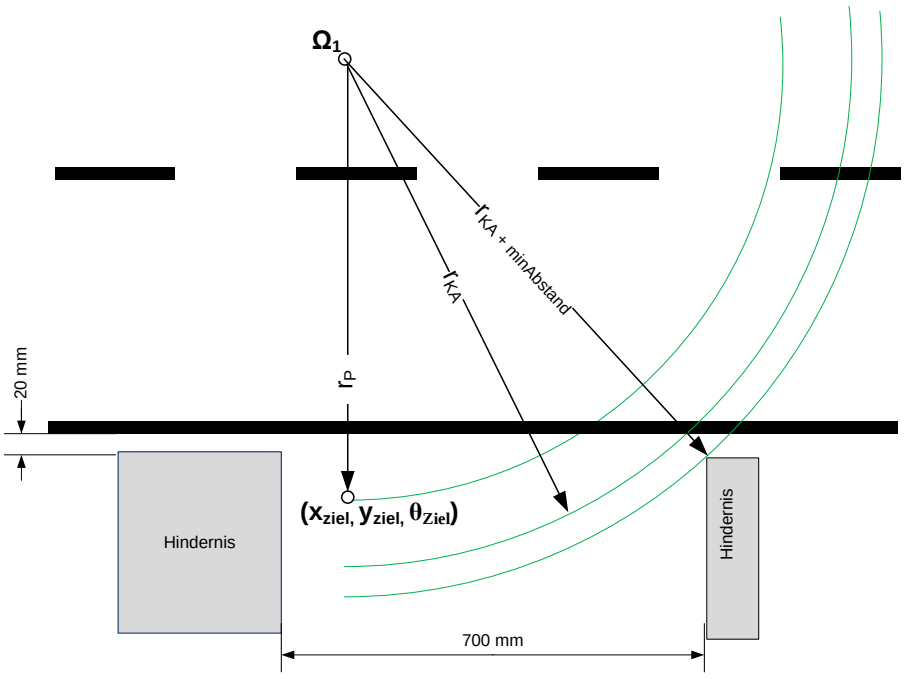


Abb. 17: *Szenario 2: Konstruktion des ersten Kreises aus der Parklücke heraus, mit der maximalen Tiefe und minimalem Abstand zum Hindernis*

Der erste zu konstruierende Kreis (siehe Gl. 12) führt aus der Parklücke heraus und hat besitzt eine X-Koordinate in einem Mindestabstand zum linken Hindernis. Der Kreis des Kollisionsradius plus Mindestabstand hat den gleichen Mittelpunkt, wie der des Wendekreises des Weges (vgl. Abb. 7). Die Y-Koordinate des Kreises hat einen Berührungspunkt mit dem rechten Hindernis.

Der weitere Kreis zur Fahrspur hat einen Berührungspunkt mit dem zuvor erzeugten Kreis und berührt entweder die Y-Koordinate der aktuellen Konfiguration des Fahrzeugs. Wenn das aufgrund einer Kollision nicht geht, hat der zweite Keis den Berührungspunkt, mit dem Kreis aus der Parklücke, in der Position die kollisionsfrei ist (vgl. Abb. 34).

Während des **N-ten Einparkschritts** befindet sich das Fahrzeug in der Parklücke zwischen den Hindernissen, entweder am linken oder am rechten Hindernis. Das Ziel besteht darin, dass das Fahrzeug am anderen Ende der Parklücke in einem Achswinkel von 0 Grad stehen bleibt und eine maximale Tiefe erreicht.
Kennzeichen des ersten N-ten Einparkschrittes sind:

- bekannte Startkonfiguration
- unbekannte Zielkonfiguration, jedoch in einem bekannten Abstand zum Hindernis
- Bahn zwischen den Hindernissen ist kollisionsfrei

Der Weg von einem Ende der Parklücke zum anderen Ende kann aus 3 Kombinationen von Kreissegmenten bestehen (vgl. Kap. 5.4):

1. Linkskurve
2. Rechtskurve
3. Links- und Rechtskurve

Diese Kombinationen werden je nach Fahrtrichtung unterschiedliche Gleichungen aufgestellt. Insgesamt gibt es deshalb 6 Szenarien in der Einparkphase N-ter Einparkschritt.

Kennwerte der Kreistechnik:

- + Minimale Wege
- + definierte Abstände zu Hindernissen
- + häufig verwendete Bahnplanungsverfahren in Forschungsarbeiten
- + Erweiterbarkeit hin zu einer vollwertigen Weg-Bogenlängen Technik mit CC-Bahnen und größeren Kreisen, als der Wenderadius
- - hohe Komplexität bei Optimierung über Erweiterungen, wie CC-Bahnsegmenten
- - der Aufwand zur Konstruktion vergrößert sich, wenn mehr als zwei Kreisbahnen zur Bahnplanung verwendet werden

4.3 Virtuelle Deichsel

Die Virtuelle Deichsel ist eine Lenkwinkelregelung, die bei der Wahl der Zielposition und des Verstärkungsfaktors zu einem Bahnplanungsverfahren wird. Sie wird in den Einparkphasen „Spurführung“, „Positionierung“ und „Fahrzeug ausrichten“ verwendet (siehe Kapitel 5). Sie wurde in einer früheren Arbeit[25] für einen Einparkassistenten genutzt, der vorwärts fahrend in eine Parklücke mit großen Entfernungen in einem Einparkschritt eingeparkt ist. Es wird hier das Regelungsverhalten der Virtuellen Deichsel analysiert und Verfahren zur Findung von Zielpositionen und Verstärkungsfaktoren erläutert.

Einzige Parameter zur Beeinflussung des Regelverhaltens sind die Zielposition und der Verstärkungsfaktor. Sie ist eine nichtlineare Lenkwinkelregelungsfunktion, die als Argumente zeitlich abhängige Variablen enthält. Sie berechnet den Lenkwinkel über die Differenz vom Zielwinkel σ zum Achswinkel θ (vgl. Gl. 17 und Abb. 7). Der Lenkwinkel α ist bei einer Linksfahrt als negativ und eine Rechtsfahrt als positiv festgelegt [25].

$$\alpha(t) = \begin{cases} -(\sigma(t) - \theta(t)), & \text{wenn } vm > 0 \\ (\sigma(t) - \theta(t)), & \text{wenn } vm < 0 \end{cases} \tag{17}$$

Ohne Verstärkung des Zielwinkels σ wird das Fahrzeug in jedem Rechenschritt einen Lenkwinkel direkt zum Ziel vorgeben. Eine proportionale Verschiebung *verst* des Ziels zu der Entfernung, erzielt eine Kreisfahrt zum Ziel und regelt das Fahrzeug auf einen Achswinkel von 0 Grad (vgl. Abb. 23). Die Regelung ist abhängig von der Entfernung des Fahrzeugs zum Ziel und dem Verstärkungsfaktor, der das Ziel bei einer Berechnung proportional zur Entfernung verschiebt. Je näher das Ziel, um so geringer die Y-Verschiebung und desto direkter der Kurs auf das Ziel.

$$\sigma(t) = \arctan\left(\frac{verst \cdot y_d(t)}{x_d(t)}\right) \tag{18}$$

4.3.1 Lenkwinkelregelung

Der folgende Abschnitt behandelt die Lenkwinkelregelung der Virtuellen Deichsel, insbesondere ihr Verhalten und die Winkelberechnung bei Rückwärtsfahrten (siehe Abb. 20).

Die Wirkungsweise des Verstärkungsfaktors ist in Abbildung 18 mit den Verstärkungsfaktoren 0.5, 1 und 2 dargestellt.

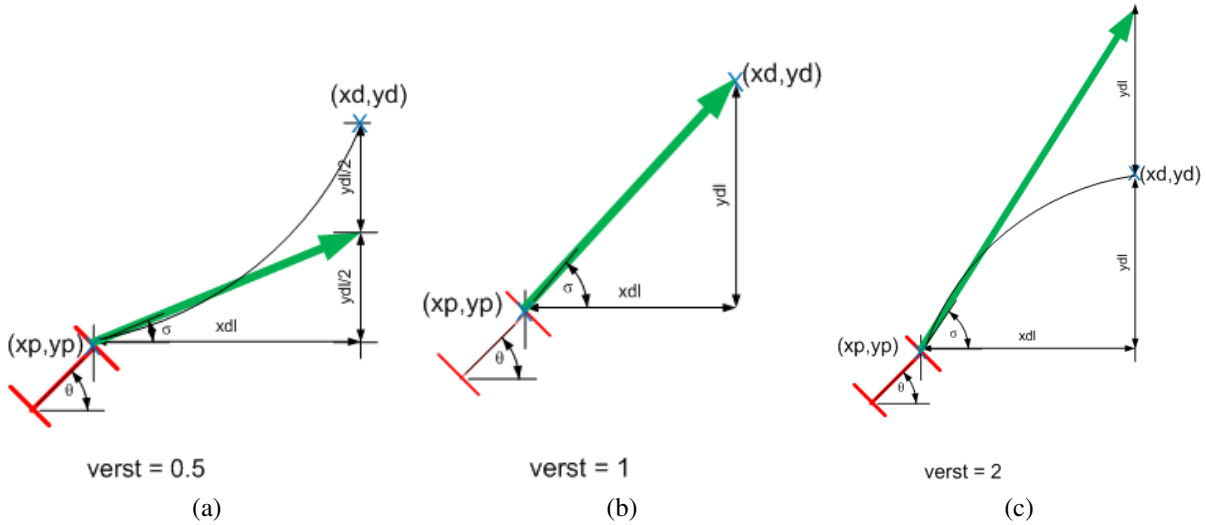


Abb. 18: *Virtuelle Deichsel - Verstärkung. Der Zielwinkel σ wird proportional zu der Entfernung verstärkt, Schwarze Kurve = abzufahrende Bahn, Grüner Pfeil = Winkel zum Ziel während einem Rechenschritt*

Ein großer Verstärkungsfaktor (vgl. Abb. 18c) bewirkt, dass zum Ziel nahezu horizontal gefahren wird. Dagegen bewirkt ein geringer Verstärkungsfaktor eine vertikale Fahrt zum Ziel. Mit abnehmender Entfernung wirkt sich der Verstärkungsfaktor immer geringer auf den Zielwinkel aus, das Fahrzeug wird immer direkter zum Ziel gelenkt. Deswegen kann mit der Virtuellen Deichsel eine S-Kurve gefahren werden.

Die Winkelberechnung zum Ziel σ kann wie bei der Vorwärtsfahrt nicht verwendet werden. Da der Wertebereich von atan() nur zwischen -90 bis +90 Grad liegt (vgl. Abb. 19).

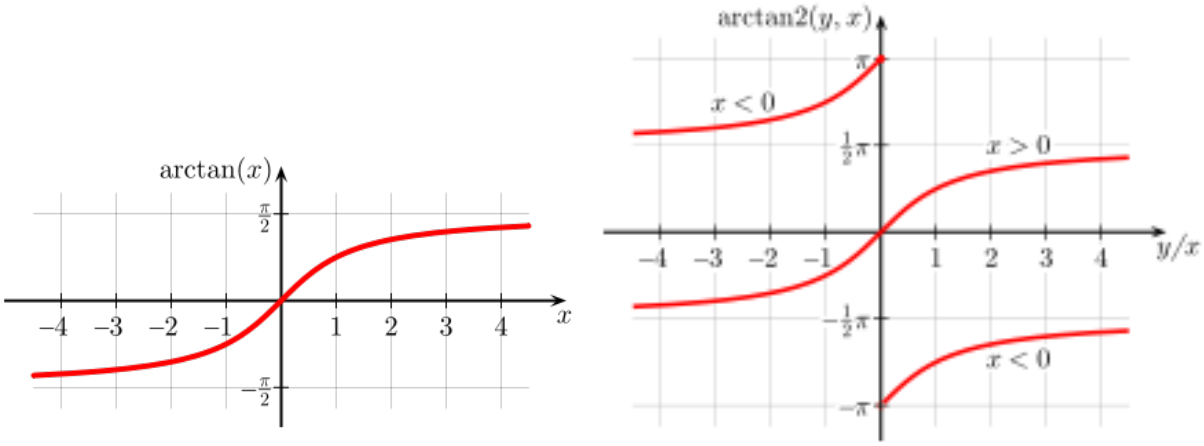


Abb. 19: *Wertebereich von atan[1] (+/-π/2) und atan2 [8] (+/-pi)*

Das Ziel liegt bei Rückwärtsfahrten hinter dem Fahrzeugheck (siehe Abb. 20). Somit kann atan() so nicht verwendet werden. Der Wertebereich muss erweitert werden auf Winkel die hinter dem Fahrzeug liegen.

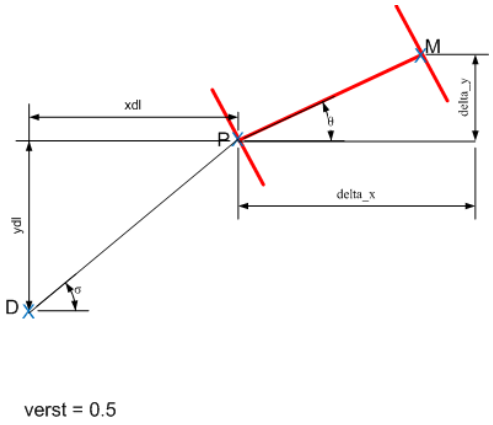


Abb. 20: *Berechnung des Zielwinkels zu einem Ziel D hinter dem Fahrzeugheck. Hinterrad P ist die Bezugskoordinate.*

Die Verwendung von atan2() erweitert den Wertebereich von 180 auf 360 Grad. Im folgenden werden beide Varianten näher beschrieben.

1. über atan2()
2. über atan() + π, zur Berechnung eine Verschiebung des Ziels um 180 Grad

Variante 1): Winkelberechnung über atan2()
Mit atan2() wird der Wertebereich verdoppelt. Der Winkel zum Ziel kann berechnet werden. Das Fahrzeug muss beim Rückwärtsfahren noch in die entgegengesetzte Richtung lenken, da die Geschwindigkeit einen negativen Wert hat (vgl. Gl. 17).

$$\alpha(t) = \begin{cases} (atan2(\left(\frac{verst \cdot y_d(t)}{x_d(t)}\right)) - \theta(t)), & \text{wenn } vm < 0 \\ -(atan2(\left(\frac{verst \cdot y_d(t)}{x_{dl}(t)}\right)) - \theta(t)), & \text{wenn } vm > 0 \end{cases} \tag{19}$$

Bei Vorwärtsfahrten und einem Ziel das vor sich vor dem Heck des Fahrzeug befindet, kann auch atan() verwendet werden. Der Vorteil von atan2() ist, dass sobald das Ziel in X- oder Y-Richtung überfahren wird, ändert sich der Winkel zum Ziel. Der Nachteil ist: der Lenkwinkel springt ohne Verzögerung auf den maximalen oder minimalen Lenkwinkel, sobald das Ziel überfahren wird (siehe Abb. 21).

Sobald die Hinterachse das Ziel überfahren hat, ändert sich ohne Verzögerung der Lenkwinkel (siehe Abb. 21). Die Richtung des Ausschlags hängt von dem Überfahren der X- und Y-Richtung ab. Für das Einparkmanöver ist es nicht notwendig, das Ziel zu umfahren. Außerdem ist es wünschenswert am Ziel gerade zum stehen zu kommen. Ein leichtes überfahren des Ziels sollte nur eine geringe Änderung auf den Achswinkel haben.

Variante 2): Winkelberechnung über atan() + π
Mit atan()+π wird das Ziel um 180 Grad verschoben, so dass das Ziel vor dem Fahrzeug liegt. Nach der Berechnung wird der Lenkwinkel α negiert, da das Fahrzeug rückwärts fährt.

Aus dem Simulationsergebnis Abb. 22 ist zu erkennen, dass nach dem Überfahren des Ziels mit der Hinterachse, das Fahrzeug nicht sofort umlenkt. Die Fahrtrichtung bleibt danach weiter stetig, weil das Ziel nur in einem Sichtbereich von +/- 90 Grad liegt.
Somit ist ein leichtes Überfahren des Ziels mit atan() toleranter gegenüber dem Verschieben

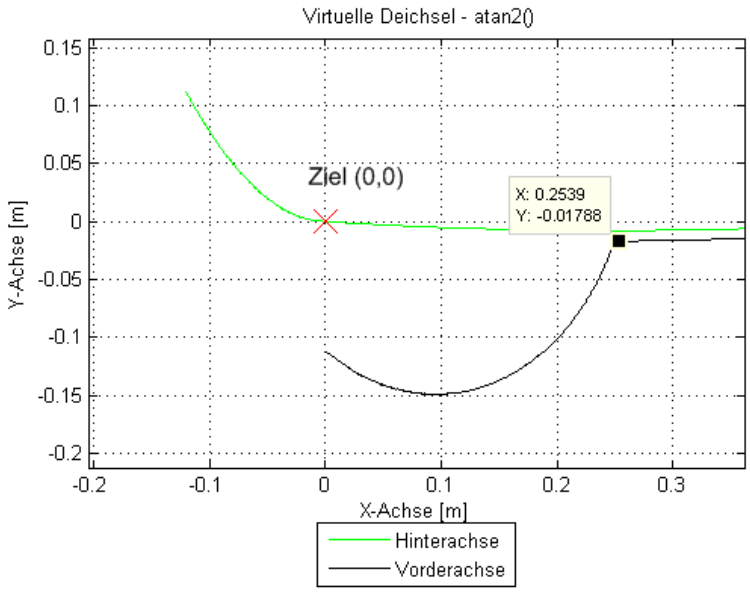


Abb. 21: *Start in ausreichender Entfernung (10,10) zum Ziel. Beim überfahren des Ziels (0,0) mit dem Hinterrad bewirkt mit atan2() ein sofortiges umlenken zurück zum Ziel.*

$$\alpha(t) = \begin{cases} (atan(\left(\frac{verst \cdot y_{dl}(t)}{x_{dl}(t)}\right)) + \pi - (\theta(t) + \pi)), & \text{wenn } vm < 0 \\ -(atan(\left(\frac{verst \cdot y_{dl}(t)}{x_{dl}(t)}\right)) - \theta(t)), & \text{wenn } vm > 0 \end{cases} \tag{20}$$

$$\alpha(t) = (atan(\left(\frac{verst \cdot y_{dl}(t)}{x_{dl}(t)}\right)) - \theta(t)), \text{wenn } vm < 0$$

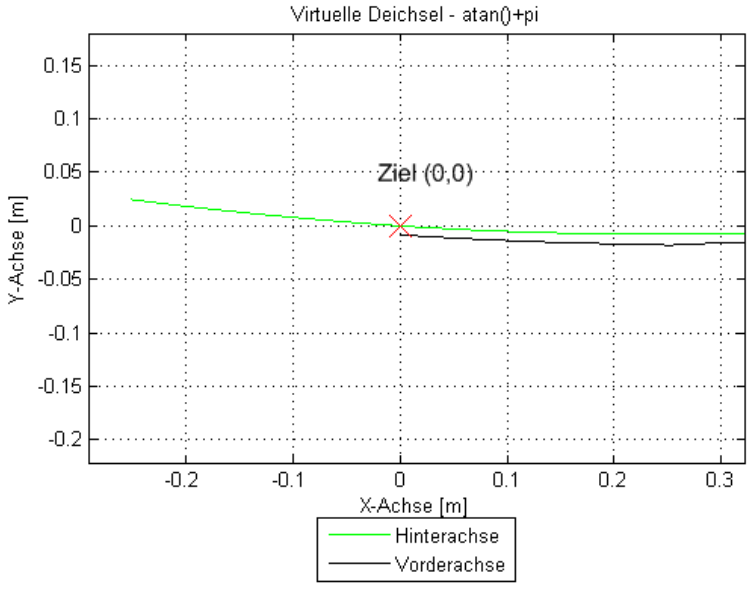


Abb. 22: *Start in ausreichender Entfernung (10,10) zum Ziel. Beim überfahren des Ziels (0,0) mit dem Hinterrad fährt das Fahrzeug geradeaus weiter.*

des Achswinkels.

Konsequenz
Atan2() bietet beim Einparken keine Vorteile gegenüber atan() + π und den wichtigen Nachteil, dass ein leichtes Überfahren des Ziels ein ohne Verzögerung Ausschlagen des Lenkwinkels zur Folge hat.

	Wertebereich [rad]	**Wertebereich [Grad]**	**nutzbarer Wertebereich [Grad] für Rückwärtsfahrt**
atan()	$[\ -\frac{\pi}{2}\ ,\ \frac{\pi}{2}\ ]$	[-90, 90]	nur -90 und 90
atan() $+\pi$	$[\ -\pi, -\frac{\pi}{2}\ ]$ und $[\ \frac{\pi}{2}\ ,\ \pi\ ]$	[90, 180] und [-90,-180]	[-90, -180] und [90, 180]
atan2()	$[-\pi, \pi\ ]$	[-180, 180]	[-90, -180 und [90, 180]]

Tabelle 3: *Wertebereiche zur Berechnung des Lenkwinkels über atan und atan2*

Aus der Tabelle 3 wird deutlich, dass atan() + π den notwendigen Wertebereich beim Rückwärtsfahren komplett ausnutzt und völlig ausreichend ist. Darüber hinaus noch die bessere Eigenschaft der stabileren Lenkung beim Einparkmanöver besitzt.

Eine Normierung des Lenkwinkels ist aufgrund der Beschränkung des Lenkwinkels vorzunehmen :

$$\alpha(t) = \begin{cases} minWinkel, & \text{wenn } \alpha(t) < minWinkel, \\ maxWinkel, & \text{wenn } \alpha(t) > maxWinkel, \\ \alpha(t), & \text{wenn } minWinkel <= \alpha(t) <= maxWinkel \end{cases} \tag{21}$$

Verhalten der Virtuellen Deichsel an deren Grenzen
Die Grenzen der Virtuellen Deichsel wird über einen sehr großen und einen sehr kleinen Verstärkungsfaktor simuliert. Die X/Y-Entfernungen zum Ziel sind sehr groß, damit das Verhalten sichtbar gemacht werden kann.

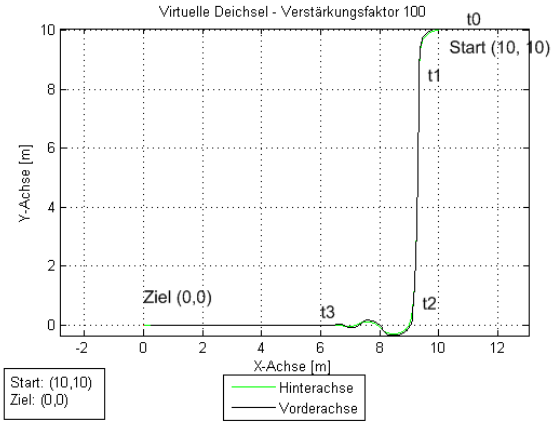


Abb. 23: *Regelung auf einen Achswinkel von 0 Grad. Verstärkungsfaktor 100, Startkonfiguration (x,y,theta)=(10 m,10 m,0 Grad) und Zielkonfiguration (0 m,0 m,0 Grad), mit negativer Geschwindigkeit von -0.1 m/s*

Zu erkennen ist in Abb. 24, dass ein sehr kleiner Verstärkungsfaktor von 0.01 eine vertikale Regelung auf das Ziel bewirkt, der 90 Grad besitzt. Ein sehr großer Verstärkungsfaktor verursacht eine horizontale Regelung auf den Zielwinkel 0 Grad (siehe Abb. 23).

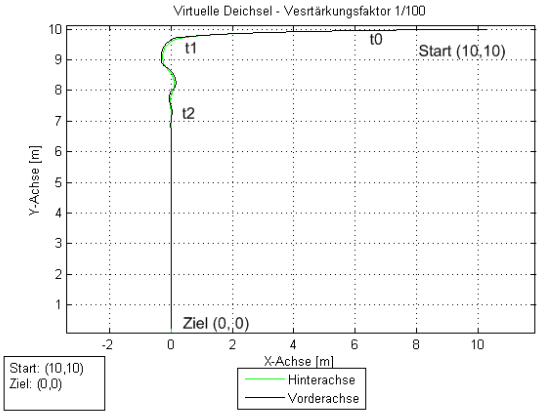


Abb. 24: *Regelung auf einen Achswinkel von 90 Grad. Verstärkungsfaktor 1/100, Startkonfiguration (x,y,theta)=(10 m,10 m,0 Grad) und Zielkonfiguration (0 m,0 m,0 Grad), negative Geschwindigkeit von -0.1 m/s*

Mathematische Erklärung Die Berechnung des Lenkwinkels wird über die Differenz vom Ziel- zum Achswinkel berechnet (vgl. Gl. 17). Der Grenzwert der Zielwinkelberechnung lässt das Verhalten sichtbar werden.

Das Verhalten der **Lenkwinkelregelung mit großem Verstärkungsfaktor** ist wie folgt zu erklären (vgl. Abb. 23):

$$\begin{aligned}\lim_{verst\to+\infty}\left[atan(verst\cdot ydl(t)/xdl(t))-\theta(t)\right] &= atan(\infty)-\theta(t)\\ &= \frac{\pi}{2}-\theta(t)\end{aligned}$$

Zum Zeitpunkt t0 der Ausgangslage der Simulation, war der Achswinkel 0 Grad und der Lenkwinkel wurde folglich auf 90 Grad eingestellt. Erst als der Achswinkel zum Zeitpunkt t1 90 Grad erreicht hat, wurde der Lenkwinkel auf 0 Grad gesetzt und die Fahrzeuglenkung auf 0 Grad gestellt. Erst bei erreichen der Y-Position des Ziels zum Zeitpunkt t2, geht ydl gegen 0 und damit der atan()-Ausdruck bei t3 gegen 0.

$$\begin{aligned}\alpha(t0) &= \frac{\pi}{2}-0=\frac{\pi}{2}\\ \alpha(t1) &= \frac{\pi}{2}-\frac{\pi}{2}=0\\ \alpha(t2) &= 0-\frac{\pi}{2}=-\frac{\pi}{2}\\ \alpha(t3) &= 0-0=0\end{aligned}$$

Ein sehr **kleiner Verstärkungsfaktor** hat folgendes Verhalten (siehe Abb. 24):

$$\lim_{verst \to 0} atan(verst \cdot ydl(t)/xdl(t)) - \theta(t)) = atan(0) - \theta(t) = -\theta(t)$$

Zum Zeitpunkt t0 der Ausgangslage der Simulation, war der Achswinkel 0 Grad und der Lenkwinkel wurde folglich auf 0 Grad eingestellt. Erst als die Zielposition in X-Richtung erreicht wird, geht der Ausdruck $ydl(t) * verst/xdl(t)$ bei einem kleineren Wert von xdl gegenüber verst, gegen ∞ und damit der atan()-Wert gegen $\pi/2$. Zum Zeitpunkt t2 hat der Achswinkel einen Wert von $\frac{\pi}{2}$ und das Fahrzeug fährt geradeaus weiter.

$$\begin{aligned} \alpha(t0) &= 0 - \theta(t) = 0 - 0 = 0 \\ \alpha(t1) &= 0 - \theta(t) = \frac{\pi}{2} - 0 = \frac{\pi}{2} \\ \alpha(t2) &= 0 - \frac{\pi}{2} - \frac{\pi}{2} = 0 \end{aligned}$$

Unangenehm sind bei sehr großen und sehr kleinen Verstärkungsfaktoren, dass sie Überschwingungen an den Eckpunkten erzeugen (vgl. Abb. 23 und Abb. 24). Sie entstehen beim Übergang von dem Zeitpunkt t1 auf t2. Die Überschwingungen haben zur Konsequenz, dass längere Wege gefahren werden müssen.

Eigenschaften der Regelung der Virtuellen Deichsel:

- + Regelung auf die Zielkonfiguration
- + Einfachheit in der Parametrisierung
- - die Regelung verfolgt nicht die vorab geplante Bahn, sondern die Zielkonfiguration. Damit ist eine kollisionsfreie Bahnplanung schwierig
- - sie ist eine Regelung, die nur den Lenkwinkels berücksichtigt. Eine Zusammenführung von Lenkwinkel- und Geschwindigkeitsregelung ist unter Umständen sehr aufwendig

4.3.2 Bahnplanung

Eine Bahn kann vorab offline durchgeführt werden. Sie über die Virtuelle Deichsel zu planen, kann nur über Simulationen durchgeführt werden. Denn sie ist eine nichtlineare Regelung, die alleine über die Parameter beeinflussbar ist. Die integrierte Regelung regelt auf eine Zielkonfiguration x, y, θ. Ein vorab geplanter Weg muss deshalb immer einen Toleranzabstand zu den Hindernissen haben.
Ein kurzer Weg zum Ziel, das von Hindernissen blockiert wird, ist ein Weg nahe an den Hindernissen vorbei. Das ist alleine über das Bahnplanungsverfahren der Virtuellen Deichsel nur in mehreren Schritten realisierbar und im Vergleich zur Kreistechnik nur mit einem nahezu doppelt so langen Weg verbunden. Aus diesem Grund werden sie nicht für den ersten und n-ten Einparkschritt verwendet (vgl. Kapitel 5.3 und 5.4).

Die Parameter die die Regelung und damit die Bahnplanung beeinflussen sind:

1. Zielkoordinate x_{Ziel}, y_{Ziel}
2. Verstärkungsfaktor *verst*
3. Winkel des Koordinatensystems

Die Zielkoordinate 1. ist ein Parameter, der mit bedacht gewählt werden muss. Da nicht jede Koordinate erreicht werden kann und evtl. nicht mit dem gewünschten Achswinkel. Der Verstärkungsfaktor 2. bewirkt eine Regelung auf 0 (vgl. Abb. Abb. 23) oder 90 Grad (vgl. Abb. Abb. 24) zum Winkel des Koordinatensystems. Über den Parameter 3. wird das Koordinatensystem gedreht, es kann so auf jeden Achswinkel θ am Ziel geregelt werden, wenn das Ziel ausreichend entfernt liegt.

Die Simulation der Virtuellen Deichsel bei Vorwärts- und Rückwärtsfahrt ergab (siehe Anhang C) C zusätzliche Simulationsergebnisse), ergab eine Symmetrie zur Hinterradkonfiguration (xp, yp, θ). Die Symmetrie gilt für relative Entfernungen von der Hinterradposition mit dem Achswinkel (xp, yp, θ) zu allen relativen Positionen in allen 4 Quadranten (z. B.: (xp,yp, θ) = (10,10, 0), ziel1 = (9,9, -10 Grad), ziel2 = (9,11, -10 Grad), ziel3 = (11,11, 10 Grad), ziel4 = (11,9, 10 Grad)). Somit deckt eine Simulation diese 4 Fälle ab, in dem das Fahrzeug zum Ziel gerichtet ist (vgl. Abb. 25). Es muss lediglich unterschieden werden, ob das Fahrzeug zum Ziel gerichtet oder entgegengerichtet ist (25). Um diese Unterscheidung in einer Achse einzubringen, wurde der Achswinkel θ mit einem Vorzeichen erweitert, der nur eine Aussage über die Richtung zum Ziel liefert. Diese neue Achse wird $\Delta\ \theta$ bezeichnet. Wenn das Fahrzeug zum Ziel gerichtet ist, wird es auf der Achse $\Delta\theta$ als positiv dargestellt und eine entgegengerichtete Konfiguration als negativ dargestellt.

Die Verfahren zur Parameterbestimmung sind Suchverfahren, die in einem Intervall bzw. Raum Verstärkungsfaktoren suchen. Das ist notwendig, weil während der Fahrt die relative Konfiguration vom Hinterrad zum Ziel sich ändert.

Je nach Verwendungszweck werden hier drei verschiedene Verfahren vorgestellt, die beim Einparkassistenten Verwendung finden.

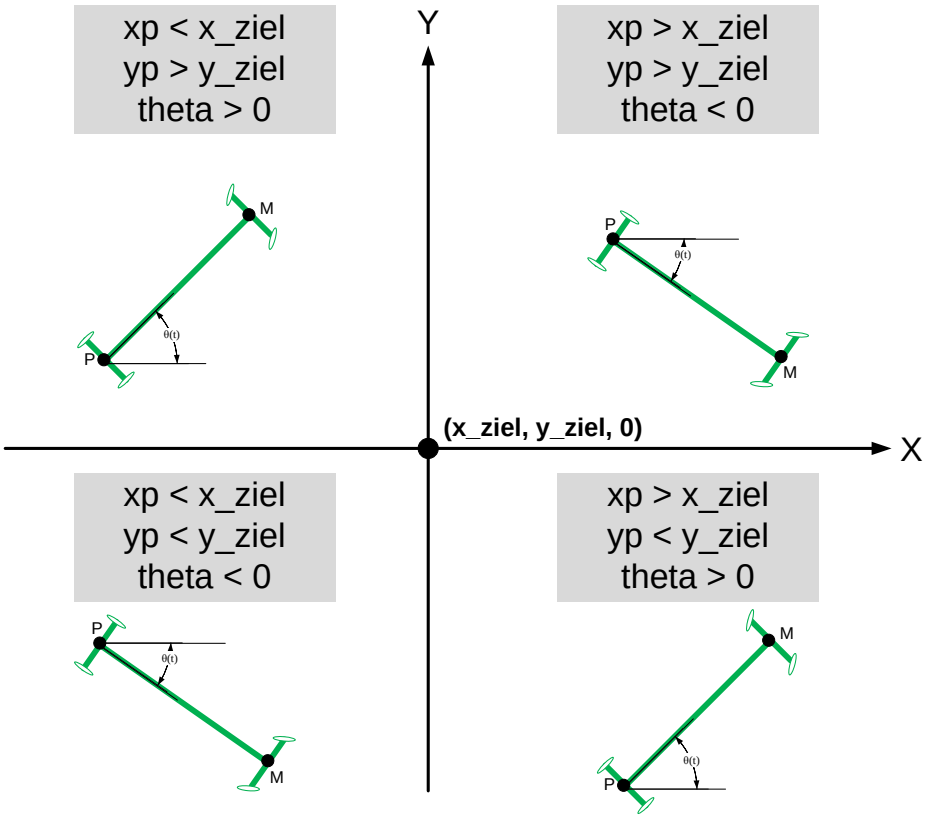


Abb. 25: *Fahrzeugkonfigurationen mit denen die Fahrzeugkonfiguration (x_p,y_p,θ) zur Zielkonfiguration entgegengerichtet ist. Das Ziel ist in der Mitte (0,0,0), es werden hier 4 Fälle gezeigt, in denen das Fahrzeug quer zum Ziel steht.*

Die Richtung wird im folgendem als direction bezeichnet, positiv bedeutet eine Konfiguration zum Ziel geneigt und negativ eine Konfiguration wie in Abbildung 25 quer zum Ziel (vgl. Gl. 22).

$$direction = \begin{cases} -1, \text{ wenn} & (xp < x_{ziel} \wedge yp > y_{ziel} \wedge \theta \geq 0) \\ & \vee (xp < x_{ziel} \wedge yp < y_{ziel} \wedge \theta < 0) \\ & \vee (xp \geq x_{ziel} \wedge yp > y_{ziel} \wedge \theta < 0) \\ & \vee (xp \geq x_{ziel} \wedge yp < y_{ziel} \wedge \theta \geq 0) \\ 1, \text{ wenn} & \neg[(xp < x_{ziel} \wedge yp > y_{ziel} \wedge \theta \geq 0) \\ & \vee (xp < x_{ziel} \wedge yp < y_{ziel} \wedge \theta < 0) \\ & \vee (xp \geq x_{ziel} \wedge yp > y_{ziel} \wedge \theta < 0) \\ & \vee (xp \geq x_{ziel} \wedge yp < y_{ziel} \wedge \theta \geq 0)] \end{cases} \tag{22}$$

Suchverfahren VD_1): Bahnplanung zu einem Ziel, bei dem nur die Einhaltung des Zielachswinkels von Bedeutung ist.

Dieses Verfahren dient nur dazu, das Fahrzeug auf diesen Achswinkel hin auszurichten. Bekannt ist während der Fahrt die Startposition (x,y) und die Zielkonfiguration (x,y,θ). Der Achswinkel am Ziel (θ) soll 0 Grad sein. Dazu wird ein Verstärkungsfaktor gesucht, der eine minimale Abweichung des Achswinkels am Ziel bewirkt.
Bei der Simulation ist nur der Achswinkel unbekannt. Damit während der Fahrt ein Verstärkungsfaktor gewählt werden kann, werden hier Intervalle von Achswinkeln verwendet. Jedes Intervall wird in ein Grad Schritten abgetastet und simuliert. Anschließend wird der Mittelwert

über die Abweichung der Abtastung gebildet. Der Betragsmäßig geringste Mittelwert in einem Intervall, ist der gesuchte Verstärkungsfaktor für dieses Intervall. Während der Fahrt muss das Fahrzeugsystem nur das Intervall kennen, in dem der momentane Achswinkel liegt und über eine Lookup-Tabelle den Verstärkungsfaktor wählen.

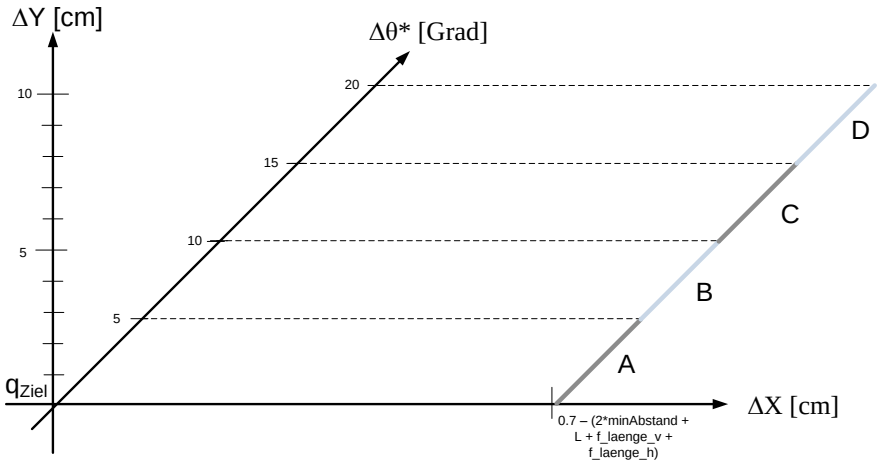


Abb. 26: *Suchverfahren VD_1), Suchräume A - D, nur der Lenkwinkel verändert sich in der Simulation, Zielkonfiguration (0,0,0)*

Die Suchräume des Achswinkels werden in jeweils 5 Grad Intervalle eingeteilt, die zwischen 0 bis 20 Grad liegen. Die Abtastung erfolgt in ein Grad Schritten (vgl. Abb. 26) Das Verfahren 1) findet Anwendung in der Einparkphase „Fahrzeug ausrichten“. Das Fahrzeug muss entweder bei Vorwärts- oder Rückwärtsfahrt den Achswinkel ausrichten. Aufgrund der Symmetrieeigenschaft der Virtuellen Deichsel deckt die Simulation das ab.

Suchverfahren VD_2): Bahnplanung zu einem Ziel mit einer vorgegebenen Y-Koordinate und Achswinkel am Ziel.

Das Fahrzeug soll eine vorgegebene Y-Koordinate in einem bestimmten Achswinkel erreichen. Die X-Entfernung ist nicht von Bedeutung und kann variabel sein.

Dieses Verfahren dient dazu, das Fahrzeug entlang einer Spur zu führen. An der Spur soll das Fahrzeug in einem Achswinkel von 0 Grad entlang fahren. Dazu wird ein Verstärkungsfaktor gesucht, der eine minimale Abweichung des Achswinkels und Y-Abweichung am Ziel bewirkt. Damit während der Fahrt ein Verstärkungsfaktor gewählt werden kann, werden hier Intervalle von Y- und Achswinkeldifferenzen zum Ziel gebildet. Zur Aufwandsminimierung wird hier nur zu zwei konstanten X-Entfernungen (50 und 100 cm) gesucht. Jedes Intervall wird in ein Grad oder 1 cm Schritten abgetastet und simuliert (siehe Abb. 27). Anschließend wird der Mittelwert über die Abweichung der Abtastung gebildet. Der Betragsmäßig geringste Mittelwert in einem Intervall, ist der gesuchte Verstärkungsfaktor für dieses Intervall.

Suchverfahren VD_3): Bahnplanung zu einem Ziel mit einer vorgegebenen Konfiguration (x,y,θ) am Ziel.

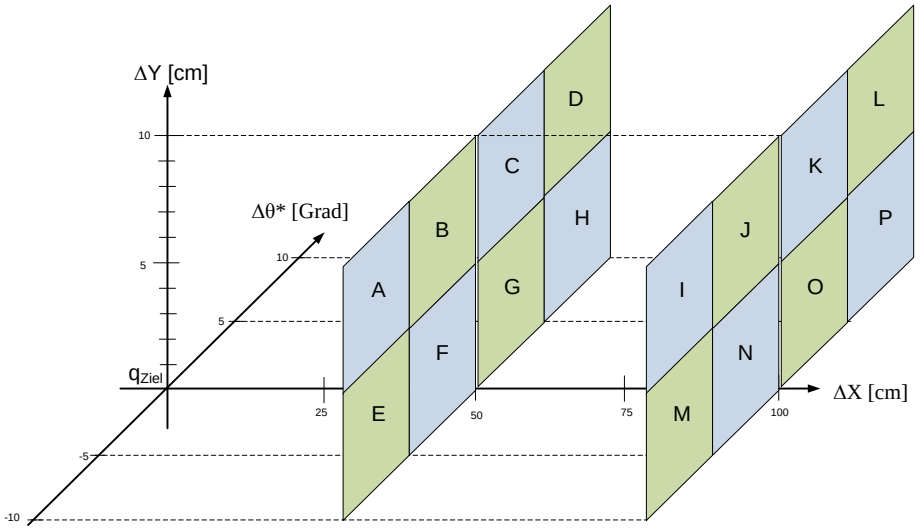


Abb. 27: *Suchverfahren VD_2), Suchräume A - P, der Lenkwinkel und Y-Entfernung zum Ziel werden in der Simulation modifiziert, Zielkonfiguration (0,0,0)*

Bei diesem Verfahren ist es wichtig, das Fahrzeug auf eine bestimmte Zielkonfiguration zu überführen. Bekannt ist nur der Zielachswinkel von 0 Grad. Gesucht ist der Verstärkungsfaktor der Überführung mit einer minimalen Abweichung erzielt.
Damit während der Fahrt ein Verstärkungsfaktor gewählt werden kann, werden hier Räume von X-, Y- und Achswinkeldifferenzen zum Ziel gebildet. Jedes Intervall wird in ein Grad oder 1 cm Schritten abgetastet und simuliert (siehe Abb. 28). Anschließend wird der Mittelwert über die Abweichung der Abtastung gebildet. Der Betragsmäßig geringste Mittelwert in einem Intervall, ist der gesuchte Verstärkungsfaktor für dieses Intervall.

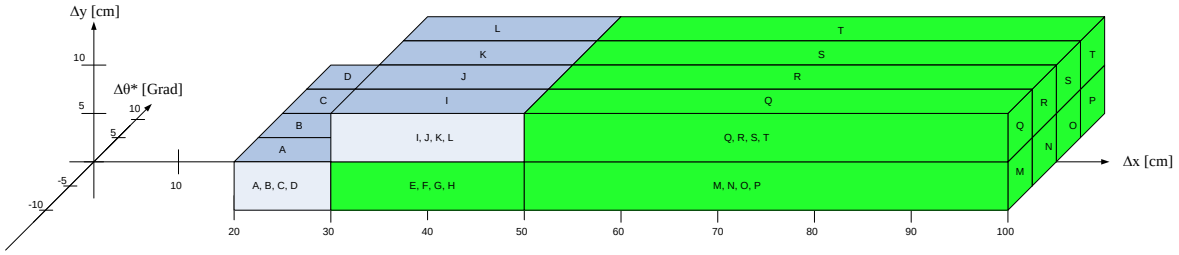


Abb. 28: *Suchverfahren VD_3), Suchräume A - T, in der Simulation werden Lenkwinkel, X- und Y-Entfernung verändert, Zielkonfiguration (0,0,0)*

Ein Suchraum wird durch ein dreidimensionalen Kasten repräsentiert (vgl. Abb. 28). Das Verfahren kann nur angewendet werden, wenn sich das Fahrzeug in einem Suchraum befindet. Eine größere X-Entfernungsdifferenz zum Ziel wird hier toleriert. Wenn sich das Fahrzeug nicht in einem der Suchräume befindet oder ein Suchraum zu große Simulationsabweichung besitzt, dann gibt es keinen Verstärkungsfaktor. Das Fahrzeug zurücksetzen und es in einer größeren Entfernung noch mal versuchen.

Kennwerte der Bahnplanung über die Virtuelle Deichsel:

- + geringer Aufwand zur Ermittlung der Bahnplanungsparameter mit den Verfahren VD_1, VD_2 und VD_3
- - nur im kollisionsfreien Raum sicher anwendbar, sonst muss ein hoher Sicherheitsabstand Kollisionsfreiheit gewährleisten

- - Simulation nur mit konstanter Geschwindigkeit, Geschwindigkeitsänderungen während der Fahrt verursachen neue Bahnen

5 Mathematischer Bahnplanungsentwurf der Einparkphasen

Zur Bahnplanung wird Das komplette Einparkmanöver wurde hier in 5 Einparkphasen eingeteilt (siehe Abb. 2), da sie unterschiedliche Anforderungen an die Bahnplanung haben. Die einzelnen Einparkphasen werden hier nacheinander in den Unterabschnitten erläutert.

Die Einparkphasen sind:.

1. **Spurführung**: Der Fahrspur in einem Achswinkel von 0 Grad folgen.
2. **Positionierung**: Zum Einparken eine exakte Position erreichen, von der aus eingeparkt wird.
3. **Erster Einparkschritt**: Das erste Einparkmanöver in die Parklücke hinein.
4. Weitere **N-Einparkschritte**: Durch Vor- und Zurücksetzen in der Parklücke eine ausreichende Tiefe erreichen.
5. Das **Fahrzeug ausrichten**: Die Einparktiefe ist erreicht, nur der Achswinkel liegt nicht im Bereich von 0 +/- 5 Grad.

Das Fahrzeug startet vor einer Startlinie und die Phase 1. Spurführung wird aktiv. In der Phase 1. verfolgt das Fahrzeug eine Virtuelle Spur. Über die Sensoren wird der rechte Fahrbahnrand nach geeignet großen Parklücken abgesucht. Wird eine Parklücke gefunden, so wird das Fahrzeug zunächst angehalten und Phase 3. geplant, indem der kürzeste Pfad aus der Parklücke berechnet wird. Zu der Startkonfiguration von Phase 3. wird das Fahrzeug über die Phase 2. dorthin positioniert. Die Positionierung wird wiederholt, falls die Abweichung zur Zielkonfiguration zu groß ist. Bei Erfolg wird der erste Einparkschritt ausgeführt. Wenn bereits nach Phase 3. eine ausreichende Tiefe der Parklücke erreicht ist und das Fahrzeug gerade steht, dann ist das Einparkmanöver beendet. Bei nicht ausreichender Tiefe wird in die Phase 4. übergegangen, bis die erforderliche Tiefe erreicht ist. Am Ende wird das Fahrzeug in Phase 5. noch ausgerichtet, falls der Achswinkel von $0 +/- 5$ Grad nicht eingehalten wird.

Die Einparkphasen Spurführung und erster Einparkschritt müssen miteinander verbunden werden. Dazu wird die Phase Positionierung verwendet. Ihre Dauer kann verkürzt werden, wenn die Y-Koordinate der Phase Spurführung mit der Y-Kooedinate des ersten Einparkschritts übereinstimmt. Es muss nur Vorwärts oder Rückwärts gefahren werden, um zur Startkonfiguration des ersten Einparkschritts zu gelangen.

Die Y-Startkoordinate der Phase Spurführung kann bei bekannten Maßen der Parklücke vorab berechnet werden. Die Startkoordinate liegt vor der Startlinie, die Y-Koordinate ist frei wählbar. Jedoch darf das Fahrzeug nur auf der rechten Fahrspur entlang fahren.

```
1 [Start_positionieren B Ziel] = phase_ersterEinparkschritt(
    hindernis_rechts, fahrspurmitte, finalC);
2 Start.x = 0;
3 Start.y = min(Start_positionieren.y, fahrspurmitte.y + 0.05); % 0.05 m
```

Die Auswahl der Y-Koordinate hängt davon ab, bei welcher Y-Koordinate die Startposition des späteren ersten Einparkschrittes sein wird. Sie ist von der Breite der Parklücke abhängig, diese Umgebungsinformationen der Parklücke sind vorab bekannt und kann deshalb berechnet werden. Die maximale Y-Koordinate liegt auf der Fahrspurmitte der rechten Fahrspur. In der Höhe ist noch ein Mindestabstand von 4.8 cm zur Straßenmitte vorhanden. Da die Fahrzeugbreite 20,2 cm und die Breite der Fahrspur 40 cm beträgt.

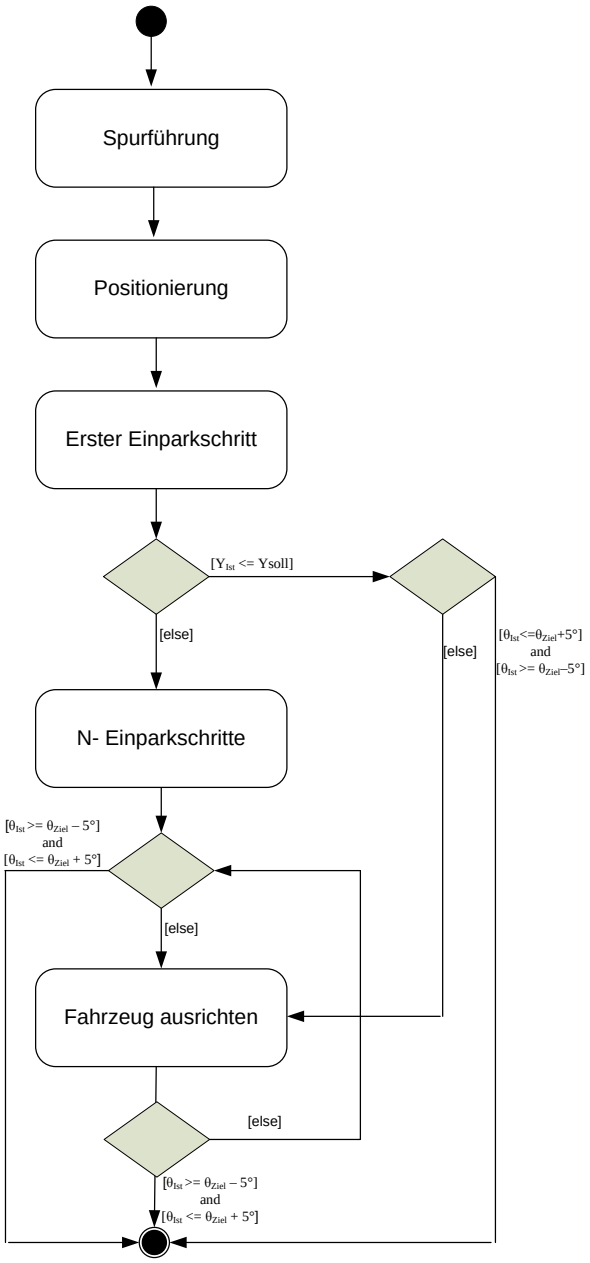


Abb. 29: *UML-Diagramm der Strategieeinheit, die Steuerung der Einparkphasen vornimmt*

Die Strategieeinheit wählt die Einparkphasen aus (siehe Abb. 29).

5.1 Spurführung

Das Ziel der Spurführung ist es, das Fahrzeug während der Parklückensuche entlang der Fahrspur zu führen. Das Konzept der Spurführung beruht auf der Lenkwinkelregelung der Virtuellen Deichsel. Genutzt wird das Bahnplanungsverfahren VD_2 (siehe Kapitel 4.3 und Abb. 27).

Zur Nutzung dieses Bahnplanungsverfahren wurden Suchräume aufgestellt (vgl. Tabelle 4)

Δ Konfigurationsraum	$\Delta x[cm]$	Δy **[cm]**	$\Delta\theta^*[Grad]$
A	50	(5,10]	(-10,-5]
B	50	(5,10]	(-5,0)
C	50	(5,10]	(0,5)
D	50	(5,10]	[5,10)
E	50	[0,5]	(-10,-5]
F	50	[0,5]	(-5,0)
G	50	[0,5]	(0,5)
H	50	[0,5]	[5,10)
I	100	(5,10]	(-10,-5]
J	100	(5,10]	(-5,0)
K	100	(5,10]	(0,5)
L	100	(5,10]	[5,10)
M	100	[0,5]	(-10,-5]
N	100	[0,5]	(-5,0)
O	100	[0,5]	(0,5)
P	100	[0,5]	[5,10)

Tabelle 4: *Suchräume in der Phase Spurführung, mit dem Verfahren VD_2, abgetastet in 1 cm und 1 Grad Schritten*

Simulations ergebnis		Kennwerte des Verstärkungsfaktors						
Delta Konfigurations-raum	Resultierender Verstärkungs-faktor	Ø Abweichung in Y-Richtung |yp - y_ziel| [m]	Ø Abweichung des Achswinkels |theta_ist- theta_ziel| [Grad]	Anzahl der Abweichungen in Y-Richtung > 0.05 m	Anzahl der Abweichungen des Achswinkels > 5 Grad	maximale Abweichung in Y-Richtung [m]	maximale Abweichung des Achswinkels [Grad]	Anzahl an Simulations-iterationen
A	2	0.01081	17.72783	0	25	0.01473	30.90273	25
B	3	0.02611	9.71275	0	16	0.03256	18.81369	20
C	3	0.02556	4.24664	0	8	0.03385	10.65681	20
D	3	0.02426	2.62087	0	2	0.03380	6.33497	25
E	4	0.01347	2.40738	0	5	0.02536	11.23357	30
F	3	0.00777	1.06918	0	0	0.01652	3.20773	24
G	2	0.00437	1.90864	0	0	0.00894	4.56409	24
H	1	0.00605	3.79899	0	10	0.01293	8.81748	30
I	5	0.01060	3.47288	0	6	0.01392	14.33133	25
J	5	0.00877	1.43716	0	0	0.01322	2.95171	20
K	4	0.00607	1.01642	0	0	0.00802	2.59935	20
L	4	0.00550	1.24795	0	0	0.00760	3.22131	25
M	5	0.00395	0.92413	0	0	0.00733	1.95219	30
N	4	0.00222	0.79966	0	0	0.00437	1.88416	24
O	4	0.00158	1.36507	0	0	0.00363	2.39161	24
P	2	0.00139	1.48594	0	0	0.00368	3.69388	30

Abb. 30: *Auswertung zur Ermittlung geeigneter Verstärkungsfaktors in der Phase Spurführung, Grün gekennzeichnet sind die Suchräume mit geringster Abweichung*

Während der Fahrt muss die X-Entfernung ausgewählt werden. Aus den Simulationsergebnissen (vgl. Abb. 30) geht hervor, dass bei einer Entfernung von 100 cm (Suchräume I bis P) die Abweichung vom Achswinkel und in Y-Richtung insgesamt geringer ist. Lediglich die Suchräume F und G haben eine maximale Abweichung von 5 Grad des Achswinkels und werden deshalb verwendet.

Die Einteilung in einen der Suchräume erfolgt nur über die Achswinkel- und Y-Abweichung. Hier werden die Suchräume F und G den Suchräumen N und O vorgezogen. Es können nicht beide verwendet werden, weil sie in den gleichen Y- und Achswinkelintervalle liegen.

$$\begin{pmatrix} x_{Ziel} \\ y_{Ziel} \end{pmatrix} = \begin{pmatrix} \begin{cases} x_p + 50, \text{ wenn} & q \in Suchraum\_F \cup Suchraum\_G \\ x_p + 100, \text{ wenn} & q \notin Suchraum\_F \cup Suchraum\_G \end{cases} \\ y_{Spur} \end{pmatrix} \tag{23}$$

Der Verstärkungsfaktor hängt von dem Suchraum ab, in dem sich das Fahrzeug mit einer Konfiguration q während der Fahrt befindet. Falls die Fahrzeugkonfiguration sich nicht in einem der Suchräume befindet, dann muss dennoch ein Ziel und ein Verstärkungsfaktor ausgewählt werden. Deshalb wird ein Standard-Verstärkungsfaktor von 1 und eine X-Entfernung zum Ziel von 100 cm verwendet.

$$verst = \begin{cases} 1, \text{ wenn} & q \notin Suchraum\_P \cup Suchraum\_G \cup Suchraum\_F \cup Suchraum\_K \\ & \cup Suchraum\_L \cup Suchraum\_I \cup Suchraum\_J \cup Suchraum\_M \\ 2, \text{ wenn} & q \in Suchraum\_P \cup Suchraum\_G \\ 3, \text{ wenn} & q \in Suchraum\_F \\ 4, \text{ wenn} & q \in Suchraum\_K \cup Suchraum\_L \\ 5, \text{ wenn} & q \in Suchraum\_I \cup Suchraum\_J \cup Suchraum\_M \end{cases} \tag{24}$$

Die Simulation der Einparkphase Spurführung ergab folgendes (siehe Abb. 31):

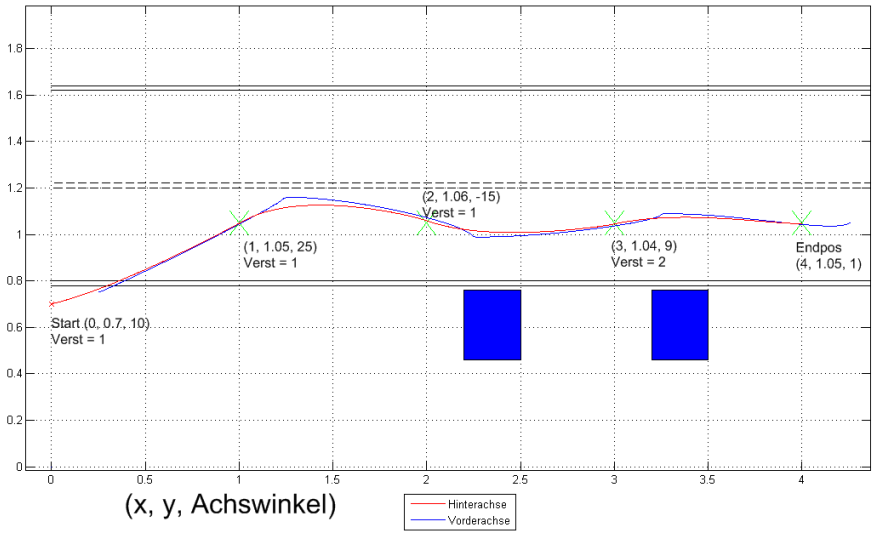


Abb. 31: *Simulationsergebnis der Einparkphase Spurführung, mit Angabe der Positionen (x,y,theta) zwischen Start (x,y,theta) und Endposition (x,y,theta)*

Bei einer Startkonfiguration mit großem Abstand zur Fahrspurmitte ergibt nach 4 Metern eine Parallelfahrt zum Fahrbahnrand.

5.2 Positionierung

Zur Positionierung soll das Ziel in einer Konfiguration mit einer geringen Achswinkelabweichung von +/- 5 Grad erreichen. Eine Abweichung in X- und Y-Richtung von 5 cm wird toleriert. Um diese Vorgaben zu erreichen, wird geprüft ob das Ziel mit einem Verstärkungsfaktor

zu erreichen ist. Für diese Einparkphase wird das Verfahren VD_3 verwendet ((siehe Kapitel 4.3 und Abb. 28)) . Die Konfigurationen C die das Fahrzeug haben kann werden in mehrere Suchräume eingeteilt.

Δ Konfigurationsraum	$\Delta x[cm]$	$\Delta y[cm]$	$\Delta\theta^*[Grad]$
A	[20,30)	(0,5]	(-10,-5]
B	[20,30)	(0,5]	(-5,0)
C	[20,30)	(0,5]	(0,5)
D	[20,30)	(0,5]	[5,10)
E	[30,50)	(0,5]	(-10,-5]
F	[30,50)	(0,5]	(-5,0)
G	[30,50)	(0,5]	(0,5)
H	[30,50)	(0,5]	[5,10)
I	[30,50)	(5,10]	(-10,-5]
J	[30,50)	(5,10]	(-5,0)
K	[30,50)	(5,10]	(0,5)
L	[30,50)	(5,10]	[5,10)
M	[50,100)	(0,5]	(-10,-5]
N	[50,100)	(0,5]	(-5,0)
O	[50,100)	(0,5]	(0,5)
P	[50,100)	(0,5]	[5,10)
Q	[50,100)	(5,10]	(-10,-5]
R	[50,100)	(5,10]	(-5,0)
S	[50,100)	(5,10]	(0,5)
T	[50,100)	(5,10]	[5,10)

Tabelle 5: *Suchräume in der Phase Positionierung, mit dem Verfahren VD_3, abgetastet in 1 cm und 1 Grad Schritten*

Die Suchräume werden in X- und Y-Richtung in 1 cm Schritten und der Achswinkel in 1 Grad Schritten abgetastet und simuliert.

Befindet sich das Fahrzeug innerhalb der Y- und Achswinkelintervalle der Suchräume Q bis P, dann wird die Fahrzeugkonfiguration q den Suchräume Q bis P zugeordnet. Denn alle Simulationsergebnisse ergaben, dass bei zunehmender X-Entfernung zum Ziel, die Abweichungen geringer werden.

Simulations ergebnis		Kennwerte des Verstärkungsfaktors						
Konfigurations-raum	Resultierender Verstärkungs-faktor	Ø Abweichung in Y-Richtung |yp - y_ziel| [m]	Ø Abweichung des Achswinkels |theta_ist- theta_ziel [Grad]|	Anzahl der Abweichungen in Y-Richtung > 0.05 m	Anzahl der Abweichungen des Achswinkels > 5 Grad	maximale Abweichung in Y-Richtung [m]	maximale Abweichung Achswinkels [Grad]	Anzahl an Simulations-iterationen
A	1	0.02989	8.80521	7	211	0.05608	14.43588	225
B	1	0.01950	9.82710	0	164	0.03900	14.88869	180
C	1	0.00971	9.19442	0	141	0.02646	15.34532	180
D	1	0.00611	7.36873	0	134	0.01789	15.95202	225
E	4	0.01178	10.08694	0	332	0.03754	23.68400	475
F	4	0.01142	5.26685	0	129	0.02344	22.02378	380
G	3	0.00853	3.39405	0	76	0.01677	14.94998	380
H	1	0.00522	4.03690	0	148	0.01356	12.99511	475
I	1	0.03892	20.38772	85	475	0.08754	25.60336	475
J	2	0.01378	20.22694	6	380	0.06064	31.10540	380
K	2	0.01099	15.71582	0	374	0.03251	30.67436	380
L	2	0.01381	11.28985	0	402	0.02012	31.18199	475
M	2	0.00624	3.04592	0	271	0.01083	12.77615	1225
N	2	0.00470	1.85745	0	33	0.00920	8.21776	980
O	2	0.00308	2.34924	0	0	0.00894	4.31500	980
P	1	0.00380	2.78949	0	175	0.01125	7.35619	1225
Q	2	0.01357	5.48756	0	478	0.02019	30.90273	1225
R	2	0.01209	3.94127	0	243	0.01870	17.39796	980
S	2	0.01029	3.50072	0	182	0.01856	11.48044	980
T	2	0.00869	4.22075	0	563	0.01974	8.36314	1225

Abb. 32: *Auswertung der Ermittlung des geeignetsten Verstärkungsfaktors*

Das Ergebnis ist (vgl. Abb. 32), dass die Suchräume G, H und M bis T eine durchschnittliche Abweichung des Achswinkels unter 5 Grad haben. Eine maximale Abweichung über 5 Grad ist in den Räumen enthalten, wird allerdings zunächst nicht berücksichtigt. Denn nach der Ausführung der Einparkphase Positionierung, wird bei einer Abweichung des Achswinkels von mehr als 5 Grad die Positionierung wiederholt.

$$verst = \begin{cases} 1, \text{ wenn} & q \notin Suchraum\_M \cup Suchraum\_N \cup Suchraum\_O \cup Suchraum\_Q \\ & \cup Suchraum\_R \cup Suchraum\_S \cup Suchraum\_T \cup Suchraum\_G \\ 2, \text{ wenn} & q \in Suchraum\_M \cup Suchraum\_N \cup Suchraum\_O \cup Suchraum\_Q \\ & \cup Suchraum\_R \cup Suchraum\_S \cup Suchraum\_T \\ 3, \text{ wenn} & q \in Suchraum\_G \end{cases} \quad (25)$$

Die Zielposition ist die Startposition des ersten Einparkschrittes.

$$\begin{pmatrix} x_{Ziel} \\ y_{Ziel} \end{pmatrix} = \left(\begin{cases} x_{Ziel}, \text{ wenn} & q \in Suchraum\_M \cup Suchraum\_N \cup Suchraum\_O \\ & \cup Suchraum\_Q \cup Suchraum\_R \cup Suchraum\_S \\ & \cup Suchraum\_T \cup Suchraum\_G \\ xp + 100cm, \text{ wenn} & (q \notin Suchraum\_M \cup Suchraum\_N \cup Suchraum\_O \\ & \cup Suchraum\_Q \cup Suchraum\_R \cup Suchraum\_S \\ & \cup Suchraum\_T \cup Suchraum\_G) \\ & \wedge xp > x_{Start\_1Einparkschritt} \\ xp - 100cm, \text{ wenn} & (q \notin Suchraum\_M \cup Suchraum\_N \cup Suchraum\_O \\ & \cup Suchraum\_Q \cup Suchraum\_R \cup Suchraum\_S \\ & \cup Suchraum\_T \cup Suchraum\_G) \\ & \wedge xp < x_{Start\_1Einparkschritt} \\ & y_{Start_{positionieren}} \end{cases} \right) \tag{26}$$

Und wenn sich die Fahrzeugkonfiguration q nicht innerhalb der Suchräume befindet, so wird das Fahrzeug in die entgegengesetzte Richtung fahren, damit beim nächsten Anlauf eine Positionierung erfolgreicher verlaufen kann. Dazu muss das Fahrzeug eine neue Zielposition und einen neuen Verstärkungsfaktor wählen. Das Verfahren VD_2 und deren Simulationsergebnisse werden verwendet. Nur die neue X-Position liegt in der entgegengesetzten Richtung zum Ziel.

Die Simulation der Einparkphase Positionierung ergab folgendes Ergebnis (siehe Abb. 33):

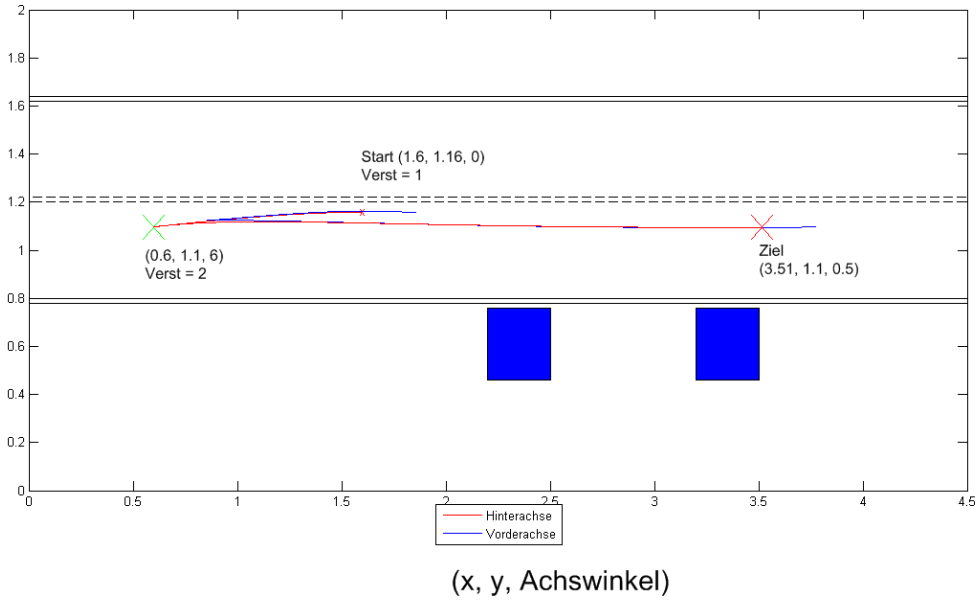


Abb. 33: *Simulationsergebnis der Einparkphase Positionierung, mit Angabe der Positionen (x,y,theta) zwischen Start (x,y,theta) und Endposition (x,y,theta) und dem Verstärkungsfaktor zu Beginn einer Bahn*

In einem Richtungswechsel wird die Zielkonfiguration erreicht.

5.3 Erster Einparkschritt

Der erste Einparkschritt wird realisiert, indem zu einer Zielposition innerhalb der Parklücke mit einem Achswinkel von 0 Grad gefahren wird. Zur Konstruktion der Kreisbahn Ω_1 aus der Parklücke heraus, gibt es zwei Szenarien:

1. Die Parklücke ist ausreichend breit, der Kreis Ω_1 hat einen Berührungspunkt mit der Zielposition in der Parklücke $final_C$.
2. Die Parklücke ist nicht ausreichend lang, der Kreis Ω_1 hat im äußeren Kollisionsradius r_{KA} einen Berührungspunkt mit dem rechten Hindernis (x_{hR}, y_{hR}) und einen Berührungspunkt in einer Zielkonfiguration *Ziel* in einem Geraden Achswinkel.

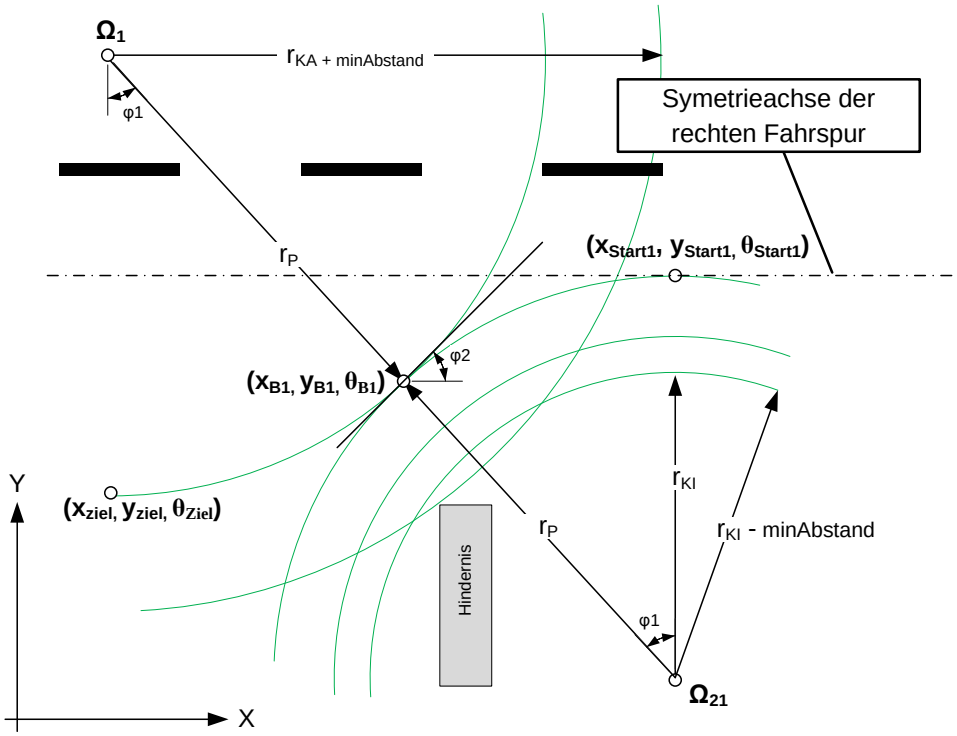


Abb. 34: *Szenario 1: Kreisbahn mit Berührungspunkt mit der Fahrspurmitte und keine Kollision des äußeren Kollisionsradius mit dem Hindernis*

Die finale Zielkonfiguration $final_C$ wird vorab berechnet, sie gibt Auskunft an die Entscheidungseinheit, ob noch weitere Einparkschritte durchgeführt werden müssen.

$$final_C = \begin{pmatrix} x_{final_C} \\ y_{final_C} \end{pmatrix} = \begin{pmatrix} y_{hL} + abst\_hindernis\_zu\_linie - f\_breite/2 - toleranz \\ x_{hL} + f\_laenge\_h + minAbstand; \end{pmatrix} \tag{27}$$

Die finale Zielkonfiguration berechnet sich zum Abstand des linken Hindernis (x_{hL}, y_{hL}). Die Toleranz ist ein Erfahrungswert, in der Arbeit hat die Toleranz einen Wert von 5 cm.

$$r^2 = (x - x_\Omega)^2 + (y - y_\Omega)^2 \tag{28}$$

$$y = y_\Omega + r \cdot \sin(\varphi) \tag{29}$$

$$x = x_\Omega + r \cdot \cos(\varphi) \tag{30}$$

Der kürzeste Weg, der von einem Hindernis blockiert wird, hat einen minimalen Abstand zum Hindernis. In die Koordinatengleichung eingesetzt (siehe Abb. 34): $x_\Omega = x_{final_C}$, r = r_{KA} +

minAbstand, für x und y die Koordinate des rechten Hindernisses (x_{hR}, y_{hR}) x = x_{hR} und y = y_{hR} ergibt folgende Mittelpunktkoordinate des Mittelpunktes Ω_1:

$$\Omega_1 = \begin{pmatrix} x_{\Omega_1} \\ y_{\Omega_1} \end{pmatrix} = \left(\begin{cases} x_{final_C} \\ y_{final_C} + r_P, \\ \text{wenn } \sqrt{(x_{hR} - x_{\Omega_1})^2 + ((y_{final_C} + r_P) - y_{hR})^2} \geq (r_{KA} + minAbstand) \\ y_{hR} + \sqrt{(r_{KA} + minAbstand)^2 - (x_{hR} - x_{final_C})^2}, \\ \text{wenn } \sqrt{(x_{hR} - x_{\Omega_1})^2 + ((y_{ziel} + r_P) - y_{hR})^2} < (r_{KA} + minAbstand) \end{cases} \right) \tag{31}$$

Das Fahrzeug kann in einem Einparkschritt die finale Zielposition erreichen, wenn Bedingung 1 erfüllt wird. Der Kreismittelpunkt liegt in einem Abstand von rp über der Zielkonfiguration. Im zweiten Fall liegt der Kreismittelpunkt Ω_1 (vgl. Gl. 31) in einem minimalen Abstand zum rechten Hindernis. Damit ist eine maximale Einparktiefe gewährleistet.

Der zweite Kreis Ω_2 zur Fahrspur hat mehr Spielraum, es gibt einen optimale Kreisbahn Ω_{21}, der die Fahrspur in einem Achswinkel von 0° berührt und eine Kreisbahn Ω_{22} mit kurzem Abstand zum Hindernis. Der optimale Kreis kann eine Kollision verursachen, wenn dessen Berührungspunkt vor dem Berührungspunkt der kurzen Kreisbahnen liegen. Dazu werden beide Berührungspunkte berechnet und falls der Berührungspunkt von Ω_{21} vor dem von Ω_{22} liegt, wird die optimale Kreisbahn Ω_{21} benutzt, sonst die kurze und sicher kollisionsfreie Kreisbahn Ω_{22}.

Szenario 1: Ausreichend große Parklücke. Die zweite Kreisbahn Ω_{21} (vgl. Gl. 32) kann nur konstruiert werden, wenn sie kollisionsfrei ist. Dazu wird im ersten Schritt die optimale Kreisbahn Ω_{21} berechnet und im zweiten Schritt die minimale Kreisbahn Ω_{22}. Wenn die optimale Kreisbahn weiter vom Hindernis entfernt liegt, dann wird die optimale Kreisbahn genommen. Die zweite Kreisbahn hat einen Berührungspunkt mit der Kreisbahn Ω_1 und einen Berührungspunkt mit der rechten Fahrspurmitte in einem Winkel von 0 Grad.

$$\Omega_{21} = \begin{pmatrix} x_{\Omega_{21}} \\ y_{\Omega_{21}} \end{pmatrix} = \begin{pmatrix} x_{Omega_1} + \sqrt{(2 \cdot r_P)^2 - (y_{Omega_1} - (y_{spurmitte} - rp))^2} \\ y_{spurmitte} - r_P \end{pmatrix} \tag{32}$$

Der Berührungspunkt liegt auf der Hälfte der Strecke zwischen der Start- und Zielkonfiguration, die einen Achswinkel von 0° besitzen. Die Winkel φ (vgl. Gl. 33), die das Fahrzeug zum Berührungspunkt fahren muss, sind somit gleich groß.

$$\varphi_1 = \arctan\left(\frac{x_{\Omega_{21}} - x_{\Omega_1}}{y_{\Omega_1} - y_{\Omega_{21}}} \right) \tag{33}$$

Der Berührungspunkt berechnet sich über die Parameterform des Kreises (vgl. Gl. 30):

$$B_1 = \begin{pmatrix} x_{B_1} \\ y_{B_1} \\ \theta_{B1} \end{pmatrix} = \begin{pmatrix} x_{\Omega_1} + r_p \cdot \sin(\varphi_1) \\ y_{\Omega_1} - r_p \cdot \cos(\varphi_1) \\ \varphi_1 \end{pmatrix} \tag{34}$$

Und die Startkonfiguration $start_1$ (vgl. Gl. 35) die das Fahrzeug in der Positionierungsphase erreichen muss und die Konfiguration, ab der der erste Einparkschritt startet.

$$Start_1 = \begin{pmatrix} x_{Start_1} \\ y_{Start_1} \\ \theta_{Start_1} \end{pmatrix} = \begin{pmatrix} x_{\Omega_{21}} \\ y_{\Omega_{21}} + r_p \\ 0 \end{pmatrix} \tag{35}$$

Die Simulation der Einparkphase erster Einparkschritt ergab für Szenario 1 folgendes Ergebnis (siehe Abb. 35):

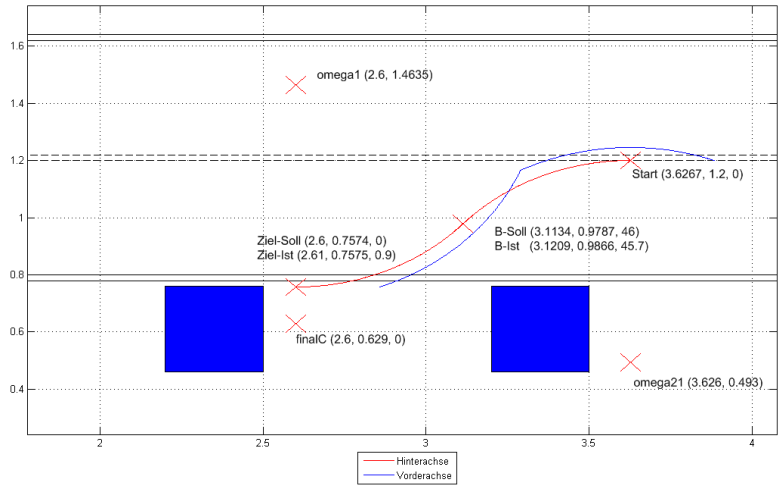


Abb. 35: *Simulationsergebnis der Einparkphase erster Einparkschritt, mit Angabe der Berührungskonfiguration (x,y,theta) zwischen Start (x,y,theta) und Ziel (x,y,theta) und den Mittelpunkten der Kreisbahnen omega1 und omega2*

Die Zielposition kann in Y-Richtung nicht erreicht werden (vgl. Abb. 35). Deshalb wird das Ziel aus der Parklücke verschoben. Die Achswinkelabweichung am Ziel beträgt 0,9 Grad.

Szenario 2: In die Parklücke kann nicht in einem Einparkschritt eingeparkt werden.
Der Kreis Ω_{22} (siehe Abb. 36) mit dem minimalem Abstand zum Hindernis wird mit einem Berührungspunkt, des Kreises aus der Parklücke, mit definierten Mindestabstand konstruiert:

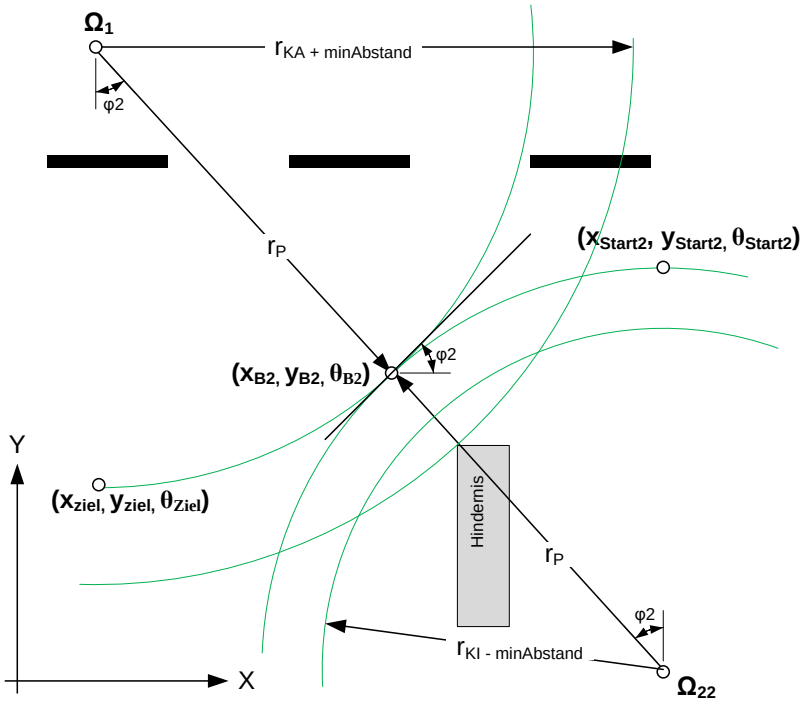


Abb. 36: *sicher kollisionsfreie Kreisbahn zur Fahrspur mit minimalem Abstand zum Hindernis*

Der erste Kreis Ω_1 (siehe Gl. 31) ist sicher kollisionsfrei, wenn die Kreisbahn in Höhe des Hindernisses beginnt. Der zweite Kreis Ω_{22} ist auch kollisionsfrei, weil die Differenz von dem äußeren Kollisionsradius zum Hinterradradius größer ist, als die Differenz von dem inneren Kollisionsradius zum Hinterradradius. Die Kreisbahn Ω_{22} endet somit in Höhe des minimalen Abstandes vom Kreis Ω_1 zum Hindernis.

Der Winkel φ_2 wird über dem neuen Kreismittelpunkt Ω_{22} berechnet:

$$\varphi_2 = \arctan\left(\frac{x_{hR} - x_{\Omega_1}}{y_{\Omega_1} - y_{hR}}\right) \tag{36}$$

φ_2 ist der Winkel vom Kreismittelpunkt Ω_1 zum Hindernis (x_{hR}, y_{hR}).
Der Kreismittelpunkt Ω_{22} liegt in einer Entfernung von 2 mal Wenderadius zu dem Kreis Ω_1.

$$\Omega_{22} = \begin{pmatrix} x_{\Omega_{22}} \\ y_{\Omega_{22}} \end{pmatrix} = \begin{pmatrix} x_{\Omega_1} + \sin(\varphi_2) \cdot 2 \cdot r_P \\ y_{\Omega_1} - \cos(\varphi_2) \cdot 2 \cdot r_P \end{pmatrix} \tag{37}$$

Und die neue Berührungskoordinate liegt im gleichen Winkel φ_2:
Die Startkonfiguration $start_2$ mit der das Fahrzeug diese Phase erster Einparkschritt beginnt:

$$B_2 = \begin{pmatrix} x_{B_2} \\ y_{B_2} \\ \theta_{B2} \end{pmatrix} = \begin{pmatrix} x_{\Omega_1} + r_p \cdot \sin(\varphi_2) \\ y_{\Omega_1} - r_p \cdot \cos(\varphi_2) \\ \varphi_2 \end{pmatrix} \tag{38}$$

$$Start_2 = \begin{pmatrix} x_{Start_2} \\ y_{Start_2} \\ \theta_{Start_2} \end{pmatrix} = \begin{pmatrix} x_{\Omega_{22}} \\ y_{\Omega_{22}} + r_p \\ 0 \end{pmatrix} \tag{39}$$

Die Kreisbahn Ω_{21} kann nur verwendet werden, wenn sie vor dem Berührungspunkt der minimalen Kreisbahn Ω_{22} endet. Bzw. wenn der Berührungspunkt der Kreisbahn B_1 weiter außerhalb der Parklücke liegt:

$$\begin{pmatrix} Start \\ B \\ \varphi \end{pmatrix} = \begin{cases} \begin{pmatrix} Start_1 \\ B_1 \\ \varphi_1 \end{pmatrix}, & \text{wenn } y_{B_1} \geq y_{B_2} \\ \begin{pmatrix} Start_2 \\ B_2 \\ \varphi_2 \end{pmatrix}, & \text{wenn } y_{B_1} < y_{B_2} \end{cases} \tag{40}$$

Die Simulation der Einparkphase erster Einparkschritt ergab für Szenario 2 folgendes Ergebnis (siehe Abb. 37):

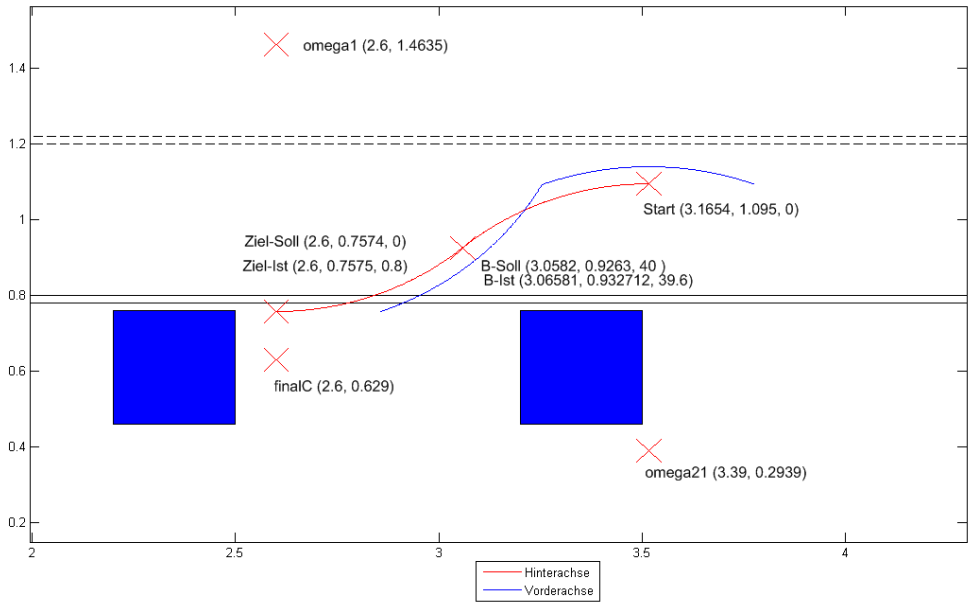


Abb. 37: *Simulationsergebnis der Einparkphase erster Einparkschritt, mit Angabe des Berührungskonfiguration (x,y,theta) zwischen Start (x,y,theta) und Ziel (x,y,theta) und den Mittelpunkten der Kreisbahnen omega1 und omega2*

Die Zielposition kann erneut nicht in Y-Richtung erreicht werden. Deshalb wird das Ziel aus der Parklücke verschoben. Die Achswinkelabweichung am Ziel beträgt 0,6 Grad.

Die Zielkonfiguration *Ziel* die das Fahrzeug in dem Einparkschritt erreichen wird ist:

$$Ziel = \begin{pmatrix} x_{Ziel} \\ y_{Ziel} \\ \theta_{Ziel} \end{pmatrix} = \begin{pmatrix} x_{finalC} \\ y_{\Omega_1} - rp \\ 0 \end{pmatrix} \tag{41}$$

Da der Achswinkel an der Startposition und an der Zielposition o Grad beträgt, wird der Weg über die Bogenlänge des Winkels ϕ_1 berechnet (siehe Abb. 36 und 34 und Gl. 42).

$$\begin{pmatrix} s_{\Omega_1} \\ s_{\Omega_2} \end{pmatrix} = \begin{pmatrix} rp \cdot \phi_1 \\ rp \cdot \phi_1 \end{pmatrix} \tag{42}$$

5.4 N-Einparkschritte

In der Einparkphase „N-Einparkschritte“ wird das Fahrzeug mit maximalen Lenkwinkel vor- und zurücksetzen, um so eine maximale Tiefe pro Einparkschritt zu erzielen. Im Idealfall startet das Fahrzeug aus einer Startkonfiguration mit einem Achswinkel parallel zur Parklücke und wird die Zielkonfiguration auch parallel erreichen. Die Bahn besteht aus zwei entgegengerichteten Kreisbahnen. Es gibt jedoch Ausnahmesituation in der nur eine Kreisbahn fahrbar ist, um eine maximale Tiefe und geraden Achswinkel zu erreichen.

Zur Bestimmung der Einparkschrittrichtung (0 = Rückwärts, von Rechts nach Links und 1 = Vorwärts, von Links nach Rechts) wird die Lage des Fahrzeugs in der Parklücke ermittelt. Wenn der Fahrzeugmittelpunkt größer oder gleich dem Parklückenmittelpunkt ist (Richtung = 0), dann wird das Fahrzeug von Rechts nach Links rückwärts fahren. Ansonsten (Richtung = 1) wird der Einparkschritt von Links nach Rechts berechnet.

$$Richtung = \begin{cases} 0, & \text{wenn } xp + L/2 \geq (x_{hR} - (x_{hR} - x_{hL})/2) \\ 1, & \text{wenn } xp + L/2 < (x_{hR} - (x_{hR} - x_{hL})/2) \end{cases} \tag{43}$$

Je Einparkrichtung gibt es je drei verschiedene Arten von Bahnen: Linkskurve, Rechtskurve und Links/Rechtskurve. Insgesamt sind es 6 unterschiedliche Szenarien, die differenziert betrachtet werden müssen.

- **Szenario 1)** Vorwärtsfahrt, Weg zum Ziel besteht aus 2 Kreisbahnen
- **Szenario 2)** Vorwärtsfahrt, Weg zum Ziel besteht aus einer Links-Kurve
- **Szenario 3)** Vorwärtsfahrt, Weg zum Ziel besteht aus einer Rechts-Kurve
- **Szenario 4)** Rückwärtsfahrt, Weg zum Ziel besteht aus 2 Kreisbahnen
- **Szenario 5)** Rückwärtsfahrt, Weg zum Ziel besteht aus einer Rechts-Kurve
- **Szenario 6)** Rückwärtsfahrt, Weg zum Ziel besteht aus einer Links-Kurve

Die Startkoordinate ist immer die aktuelle Position des Hinterrades xp und yp.

$$\begin{pmatrix} x_{start} \\ y_{start} \end{pmatrix} = \begin{pmatrix} xp \\ yp \end{pmatrix} \tag{44}$$

Der erste Kreis von der Startkonfiguration wird je nach dem Vorzeichen des Achswinkels unterschiedlich berechnet, sie wird über die Dreiecke in den Abbildungen vom Szenario 1) und 4) konstruiert.

$$\begin{pmatrix} x_{\Omega_1} \\ y_{\Omega_1} \end{pmatrix} = \begin{pmatrix} x_{start} + \sin(\theta) \cdot rp \\ y_{start} - \cos(-\theta) \cdot rp \end{pmatrix} \tag{45}$$

Die Herleitung vom Kreis Ω_1 kann in Abbildung von Szenario 1) und 3) nachgesehen werden.

Zur Simulation wird der Weg zum Berührungspunkt abgefahren. Der Winkel in [Rad] ist definiert als: $winkel = Weg/Radius$.

Szenario 1) und 4) treten nur ein, wenn die anderen Fälle ausgeschlossen werden können. Die Herleitung der Fallunterscheidung für Szenario 1) ist in der Szenario 2) und 3) erklärt und für Szenario 4) in Szenario 5) und 6). Eine detaillierte Beschreibung der Szenarios wird in folgenden Unterabschnitten beschrieben.

$$Szenario = \begin{cases} 1, & \text{wenn } richtung == 1 \\ & \wedge(x_{\Omega_1} \geq x_{hR} - minAbstand - L - f\_laenge\_v) \\ & \wedge(x_{\Omega_1} + sin(-\theta) \cdot 2 \cdot rp \leq x_{hR} - minAbstand - L - f\_laenge\_v) \\ 2, & \text{wenn } richtung == 1 \\ & \wedge(x_{\Omega_1} + sin(-\theta) \cdot 2 \cdot rp > x_{hR} - minAbstand - L - f\_laenge\_v) \\ 3, & \text{wenn } richtung == 1 \\ & \wedge(x_{\Omega_1} > x_{hR} - minAbstand - L - f\_laenge\_v) \\ 4, & \text{wenn } richtung == 0 \\ & \wedge(x_{\Omega_1} - sin(\theta) \cdot 2 \cdot rp \geq x_{hL} + minAbstand + f\_laenge\_h) \\ & \wedge(x_{\Omega_1} \geq x_{hL} + minAbstand + f\_laenge\_h) \\ 5, & \text{wenn } richtung == 0 \\ & \wedge(x_{\Omega_1} - sin(\theta) \cdot 2 \cdot rp < x_{hL} + minAbstand + f\_laenge\_h) \\ 6, & \text{wenn } richtung == 0 \\ & \wedge(x_{\Omega_1} < x_{hL} + minAbstand + f\_laenge\_h) \end{cases} \tag{46}$$

$$s1 = rp \cdot \varphi_1 \tag{47}$$

$$s2 = rp \cdot \varphi_2 \tag{48}$$

Szenario 1: Einparkschritt Vorwärts

Im ersten Szenario besteht der Weg zum Zwischenziel aus zwei Kreisbahnen. Einer Links und einer Rechtskurve, die vorwärts gefahren wird.

Der Kreiskoordinate x_{Ω_2} liegt über der Zielkonfiguration, da der Achswinkel dort 0 Grad beträgt. Die Y-Koordinate wird über den Satz des Pythagoras berechnet.

Der Hilfswinkel β ist gleichzeitig der abzufahrende Winkel vom zweiten Kreis, er berechnet sich über das abgebildete Dreieck:

Zur Berechnung des Berührungspunktes b gibt es mehrere Wege, wie z.B. Satz des Pythagoras, hier über den Hilfswinkel:

Der abzufahrende Winkel φ_1 wird über den Hilfswinkel β und den negativen Achswinkel θ berechnet. $\varphi_1 = \pi - (\pi - \beta) - (-\theta)$ und aufgelöst wie folgt:

Die Simulation der Einparkphase N-Einparkschritte ergab für Szenario 1 folgendes Ergebnis (siehe Abb. 39):

Das Fahrzeug wird in diesem Szenario 1 grade gezogen und die Y-Tiefe wird erhöht.

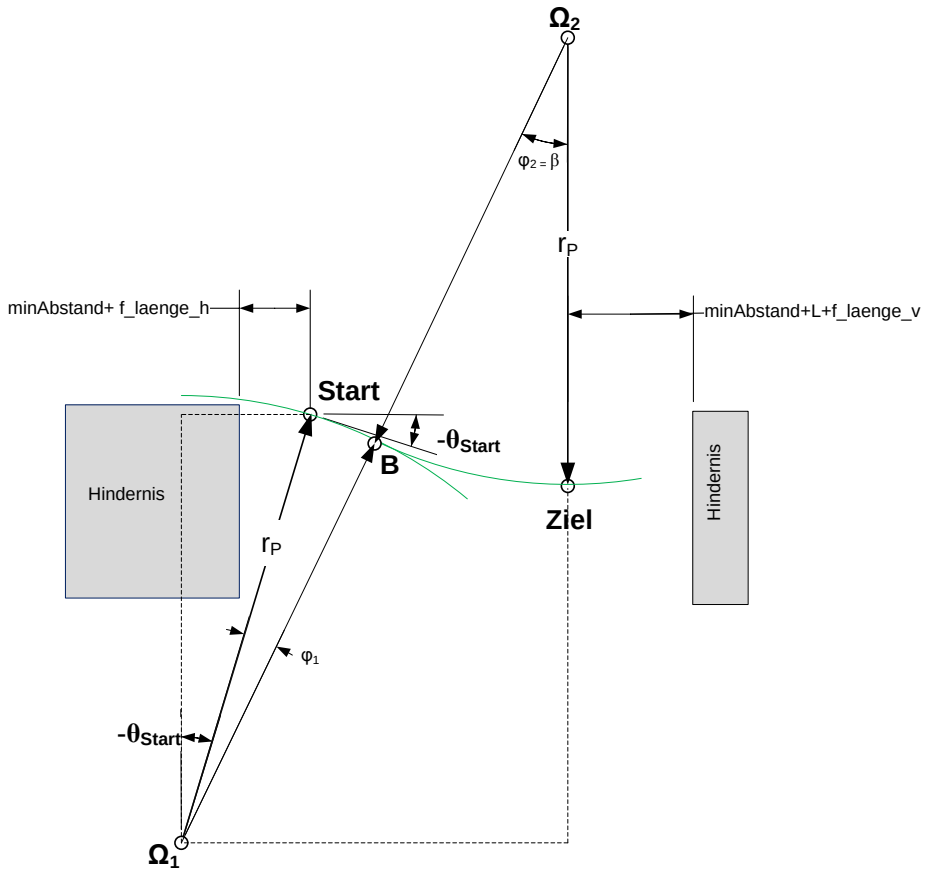


Abb. 38: *Einparkschritt vorwärts, bestehend aus 2 Kreisbahnen*

$$\begin{pmatrix} x_{\Omega_2} \\ y_{\Omega_2} \end{pmatrix} = \begin{pmatrix} x_{hR} - minAbstand - L - f\_laenge\_v \\ y_{\Omega_1} + \sqrt{(2 \cdot rp)^2 - (x_{\Omega_2} - x_{\Omega_1})^2} \end{pmatrix} \tag{49}$$

$$\begin{pmatrix} x_{Ziel} \\ y_{Ziel} \\ \theta_{Ziel} \end{pmatrix} = \begin{pmatrix} x_{\Omega_2} \\ y_{\Omega_2} - r_P \\ 0 \end{pmatrix} \tag{50}$$

$$\beta = \arctan((x_{\Omega_2} - x_{\Omega_1})/(y_{\Omega_2} - y_{\Omega_1})) \tag{51}$$

$$\begin{pmatrix} x_B \\ y_B \\ \theta_B \end{pmatrix} = \begin{pmatrix} x_{\Omega_1} + r_P \cdot \sin(\beta) \\ y_{\Omega_1} + r_P \cdot \cos(\beta) \\ -\beta \end{pmatrix} \tag{52}$$

Szenario 2: Einparkschritt Vorwärts

Im Einparkschritt Szenario 2) hat das Fahrzeug einen großen negativen Achswinkel und kann die Zielkonfiguration nicht in einem parallelen Achswinkel erreichen. Der Weg besteht somit

$$\begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} \beta + \theta \\ \beta \end{pmatrix} \tag{53}$$

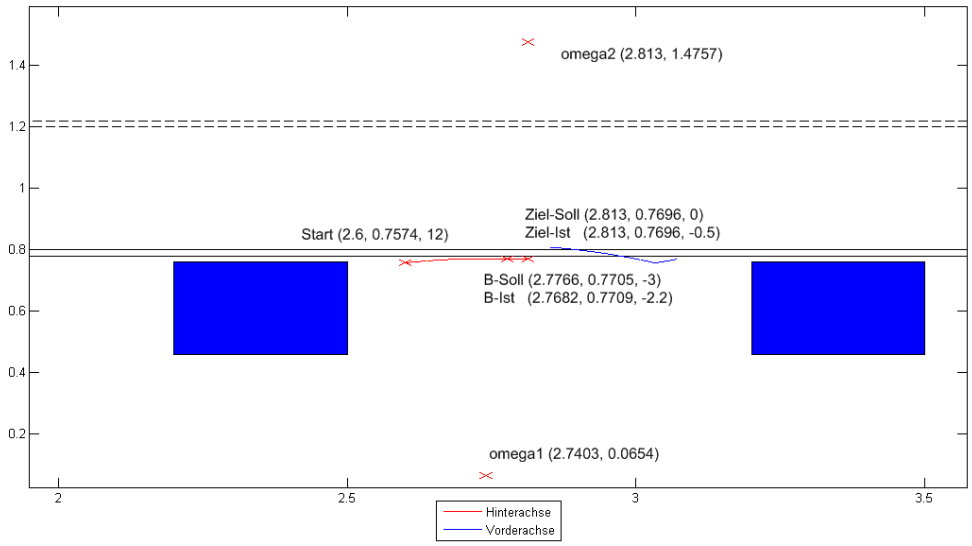


Abb. 39: *Simulationsergebnis der Einparkphase N Einparkschritte, mit Angabe der Startkonfiguration (x,y,theta) und des Soll-Ziel, des Ist-Ziels(x,y,theta), der Berürungskonfiguration und der Mittelpunkte der Kreisbahnen*

nur aus einer Linkskurve, die vorwärts gefahren wird. Szenario 2) tritt unter folgender Bedingung ein:

$$richtung == 1 \wedge (x_{\Omega_1} + sin(-\theta) \cdot 2 \cdot rp > x_{hR} - minAbstand - L - f\_laenge\_v) \tag{54}$$

Die Bedingung ist erfüllt, wenn der zweite Kreismittelpunkt Ω_2 in X-Richtung hinter dem Ziel liegt. Das bedeutet dass die Zielkonfiguration auch einen negativen Achswinkel besitzt und somit ist nur eine Kreisbahn notwendig.

Der zweite Kreis Ω_2 wird über das dargestellte Dreieck über den Kathetensatz berechnet.

Der Hilfswinkel β wird aus dem Dreieck von Ω_2 und *Ziel* konstruiert.

Die Y-Koordinate des Ziels x_{Ziel} wird dann nur noch über den Hilfswinkel β berechnet.

In der Abbildung bestehen die Winkel aus Der abzufahrende Winkel φ_2 wird wie folgt berechnet:

Die Simulation der Einparkphase N-Einparkschritte ergab für Szenario 2 folgendes Ergebnis (siehe Abb. 41):

Das Fahrzeug steht schief in der Parklücke. Das Ziel dieses Szenarios der Einparkphase N-Einparkschritte ist, dass der Achswinkel des Fahrzeugs auf 0 Grad gezogen wird.

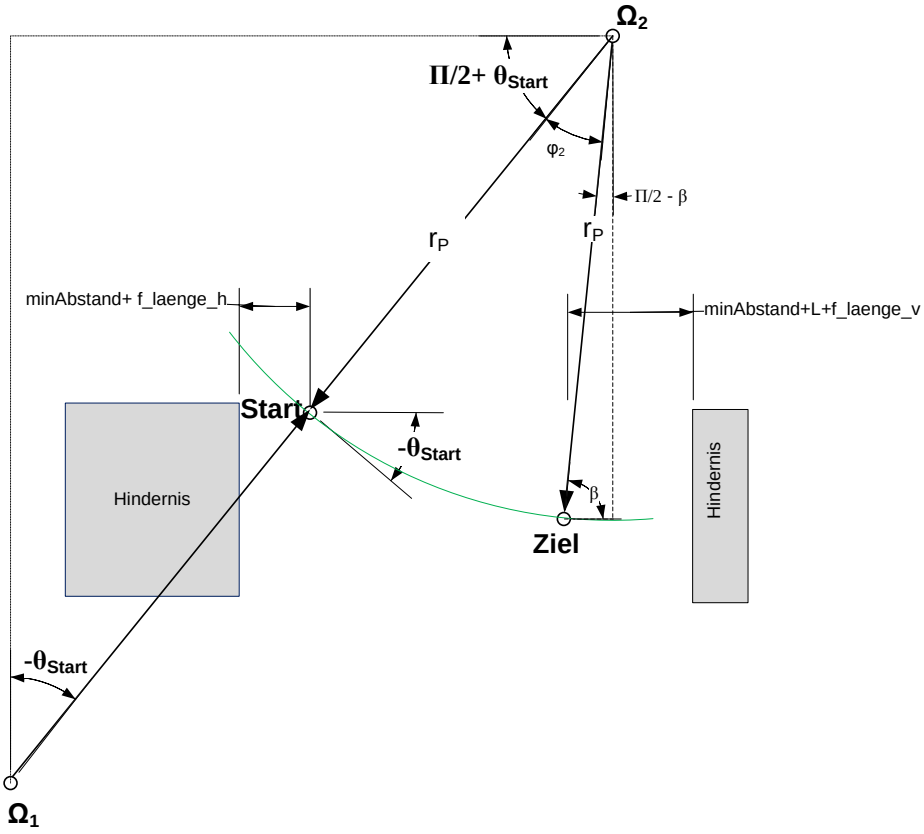


Abb. 40: *Einparkschritt vorwärts, bestehend aus 2 Kreisbahnen*

$$\begin{pmatrix} x_{\Omega_2} \\ y_{\Omega_2} \end{pmatrix} = \begin{pmatrix} x_{\Omega_1} + 2 \cdot rp \cdot \sin(-\theta) \\ y_{\Omega_1} + 2 \cdot rp \cdot \cos(-\theta) \end{pmatrix} \tag{55}$$

$$\beta = \arccos\left(\frac{x_{\Omega_2} - (x_{hR} - minAbstand - f\_laenge\_v - L)}{rp} \right) \tag{56}$$

$$\begin{pmatrix} x_{Ziel} \\ y_{Ziel} \\ \theta_{Ziel} \end{pmatrix} = \begin{pmatrix} x_{hR} - minAbstand - f\_laenge\_v - L \\ y_{\Omega_2} - rp \cdot \sin(\beta) \\ -(\pi/2 - \beta) \end{pmatrix} \tag{57}$$

$$\begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} 0 \\ \pi/2 - (\pi/2 - \beta) - (\pi/2 + \theta) = \beta - \theta - \pi/2 \end{pmatrix} \tag{58}$$

Szenario 3: Einparkschritt Vorwärts

Im Einparkschritt Szenario 3) hat das Fahrzeug einen großen positiven Achswinkel und kann die Zielkonfiguration nicht in parallelen Achswinkel erreichen. Der Weg besteht somit nur aus einer Rechtskurve. Szenario 3) tritt unter folgender Bedingung ein:

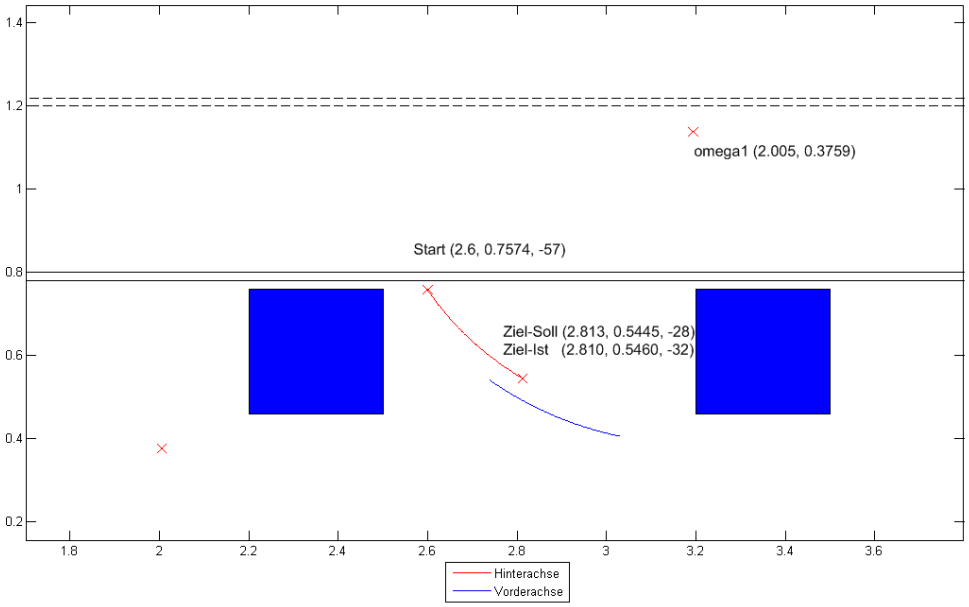


Abb. 41: *Simulationsergebnis der Einparkphase N Einparkschritte, mit Angabe der Startkonfiguration (x,y,theta) und des Soll-Ziel, des Ist-Ziels(x,y,theta)*

$$richtung == 1 \wedge (x_{\Omega_1} > x_{hR} - minAbstand - L - f\_laenge\_v) \tag{59}$$

Die Bedingung ist erfüllt, wenn der Kreismittelpunkt Ω_1 in X-Richtung hinter dem Ziel liegt. Das bedeutet dass die Zielkonfiguration auch einen positiven Achswinkel besitzen muss und somit ist nur eine Kreisbahn notwendig.

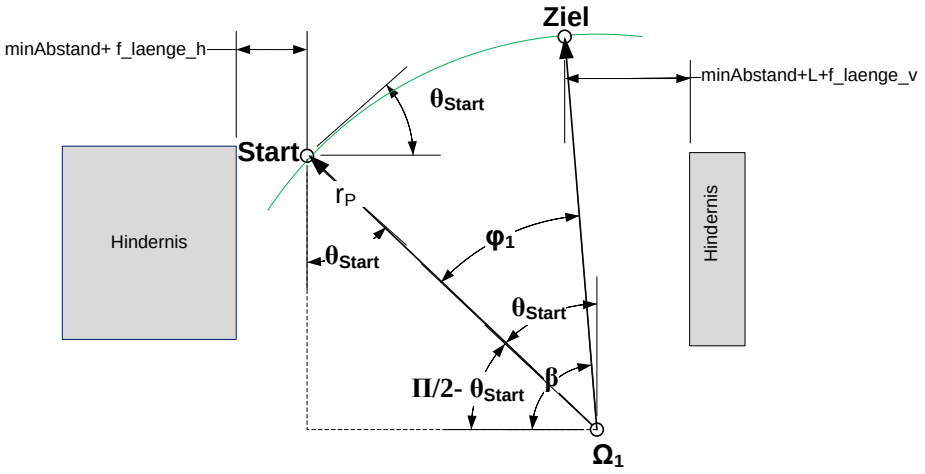


Abb. 42: *Einparkschritt vorwärts, bestehend aus 2 Kreisbahnen*

Der Hilfswinkel β wird aus dem Dreieck von Ω_1 und *Ziel* konstruiert.

Die Y-Koordinate des Ziels x_{Ziel} wird dann nur noch über den Hilfswinkel β berechnet.

$$\beta = \arccos\left(\frac{x_{\Omega_1} - (x_{hR} - minAbstand - f\_laenge\_v - L)}{rp}\right) \tag{60}$$

$$\begin{pmatrix} x_{Ziel} \\ y_{Ziel} \\ \theta_{Ziel} \end{pmatrix} = \begin{pmatrix} x_{hR} - minAbstand - f\_laenge\_v - L \\ y_{\Omega_1} + rp \cdot \sin(\beta) \\ \pi/2 - \beta \end{pmatrix} \tag{61}$$

Die Herleitung des Winkels φ_1 ist wie folgt: $\varphi_1 = \pi/2 - (\pi/2 - \theta) - (\pi/2 - \beta)$

$$\begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} \beta + \theta - \pi/2 \\ 0 \end{pmatrix} \tag{62}$$

Die Simulation der Einparkphase N-Einparkschritte ergab für Szenario 3 folgendes Ergebnis (siehe Abb. 43):

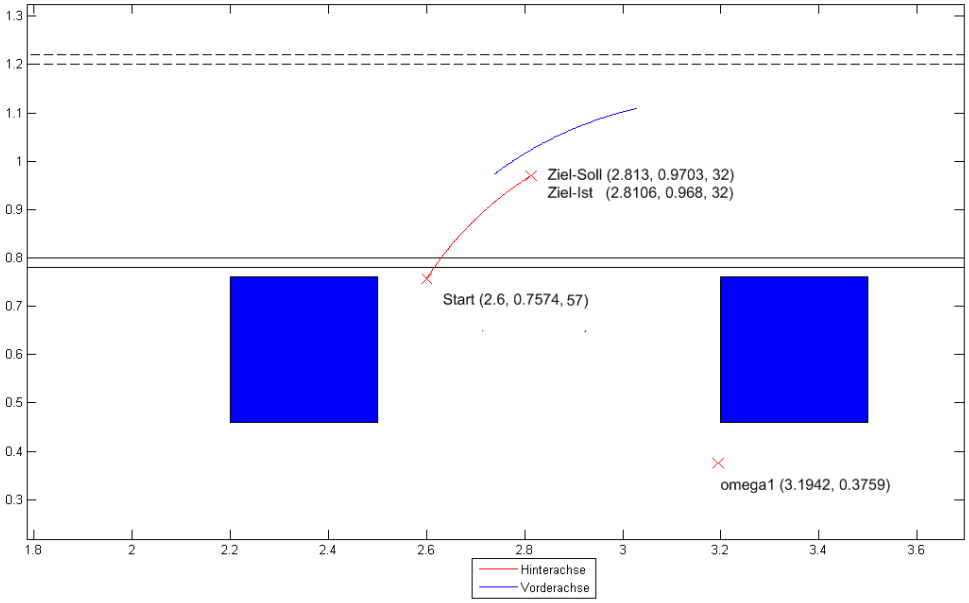


Abb. 43: *Simulationsergebnis der Einparkphase N Einparkschritte, mit Angabe der Startkonfiguration (x,y,theta) und des Soll-Ziel, des Ist-Ziels(x,y,theta), sowie der Angabe der Mittelpunkte der beiden Kreissegmente*

Mit einem Achswinkel von 57 Grad beginnt das Fahrzeug dieses Einparkphase. Es wird hier das Ausrichten des Achswinkels am Ziel erreicht, damit in folgenden Einparkschritten die Szenarien 1 und 4 verwendet werden können.

Szenario 4: Einparkschritt Rückwärts

Im Szenario 4) besteht der Weg zum Zwischenziel aus einer Rechts- und einer Linkskurve. Das Fahrzeug muss dafür rückwärts fahren.

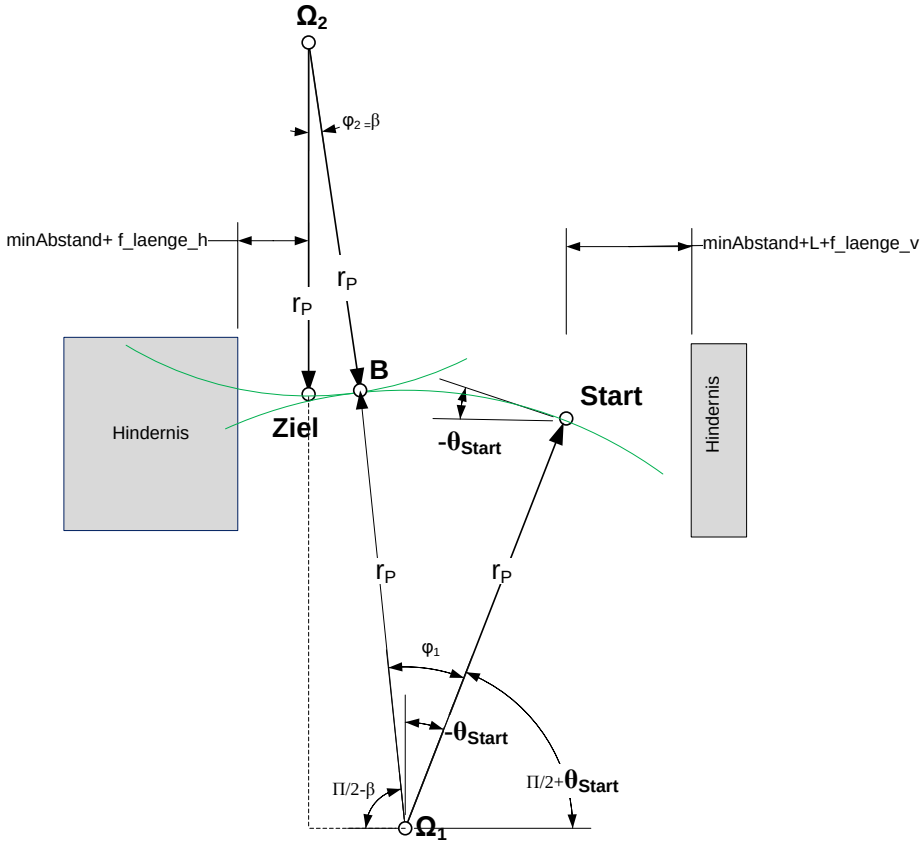


Abb. 44: *Einparkschritt Rückwärts, mit 2 Kreisbahnen*

Die Y-Koordinate des Kreises y_{Ω_2} wird über den Satz des Pythagoras hergeleitet:

$$\begin{pmatrix} x_{\Omega_2} \\ y_{\Omega_2} \end{pmatrix} = \begin{pmatrix} x_{hL} + minAbstand + f\_laenge\_h \\ y_{\Omega_1} + \sqrt{(2 \cdot rp)^2 - (x_{\Omega_2} - x_{\Omega_1})^2} \end{pmatrix} \tag{63}$$

Die Zielkoordinaten sind in einem geradem Achswinkel zu erreichen:

$$\begin{pmatrix} x_{Ziel} \\ y_{Ziel} \end{pmatrix} = \begin{pmatrix} x_{\Omega_2} \\ y_{\Omega_2} - r_P \end{pmatrix} \tag{64}$$

Und der Hilfswinkel β wie folgt:

$$\beta = \arctan((x_{\Omega_1} - x_{\Omega_2})/(y_{\Omega_2} - y_{\Omega_1})) \tag{65}$$

Der Berührungspunkt B wird über den Hilfswinkel errechnet:

$$\begin{pmatrix} x_B \\ y_B \\ \theta_B \end{pmatrix} = \begin{pmatrix} x_{\Omega_2} + r_P \cdot \sin(\beta) \\ y_{\Omega_2} - r_P \cdot \cos(\beta) \\ \beta \end{pmatrix} \tag{66}$$

Der abzufahrende Winkel φ_1 wird über Hilfswinkel konstruiert, wie in der Abbildung dargestellt.

$$\begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} \pi - (\pi/2 - \beta) - (\pi/2 + \theta) = \beta - \theta \\ \beta \end{pmatrix} \tag{67}$$

Die Simulation der Einparkphase N-Einparkschritte ergab für Szenario 4 folgendes Ergebnis (siehe Abb. 45):

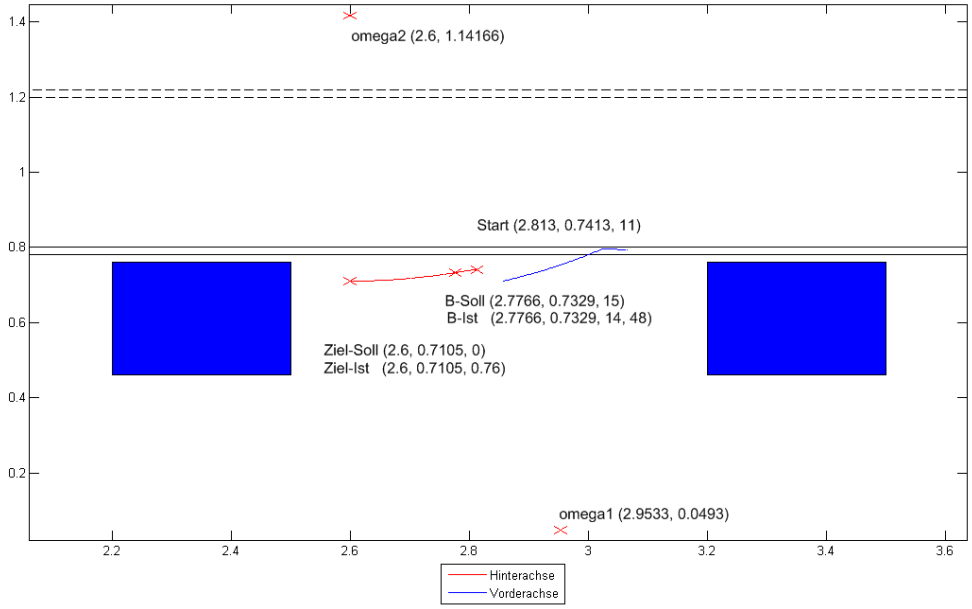


Abb. 45: *Simulationsergebnis der Einparkphase N Einparkschritte, mit Angabe der Startkonfiguration (x,y,theta) und des Soll-Ziel, des Ist-Ziels(x,y,theta) und des Berührungspunktes, sowie der Angabe der Mittelpunkte der beiden Kreissegmente*

Eine geringe Abweichung des Achswinkels am Ziel, bewirkt beim nächsten Einparkschritt eine größere Tiefe zu erreichen.

Szenario 5: Einparkschritt Rückwärts

Im Einparkschritt Szenario 5) hat das Fahrzeug einen großen positiven Achswinkel und kann die Zielkonfiguration nicht in einem Graden Achswinkel erreichen. Der Weg besteht somit nur aus einer Linkskurve. Szenario 5) tritt unter folgender Bedingung ein:

$$richtung == 0 \wedge (x_{\Omega_1} - sin(\theta) \cdot 2 \cdot rp < x_{hL} + minAbstand + f\_laenge\_h) \tag{68}$$

Die Bedingung ist erfüllt, wenn der zweite Kreismittelpunkt Ω_2 in X-Richtung hinter dem Ziel liegt. Das bedeutet dass die Zielkonfiguration auch einen positiven Achswinkel besitzt und somit ist nur eine Kreisbahn notwendig.

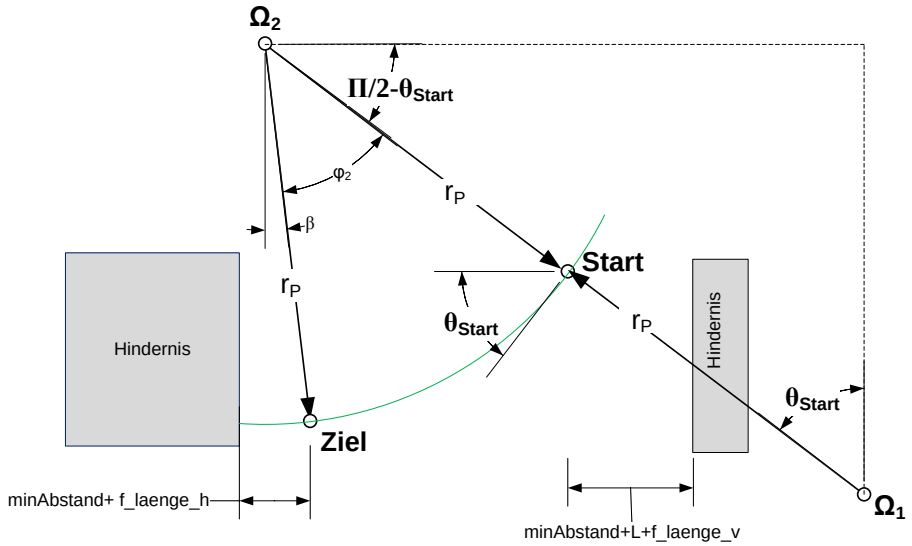


Abb. 46: *Einparkschritt Rückwärts, bestehend aus einer Linkskurve*

Der zweite Kreis Ω_2 wird über das dargestellte Dreieck über den Kathetensatz berechnet.

$$\begin{pmatrix} x_{\Omega_2} \\ y_{\Omega_2} \end{pmatrix} = \begin{pmatrix} x_{\Omega_1} - 2 \cdot rp \cdot \sin(\theta) \\ y_{\Omega_1} + 2 \cdot rp \cdot \cos(\theta) \end{pmatrix} \tag{69}$$

Der Hilfswinkel β wird aus dem Dreieck von Ω_2 und *Ziel* konstruiert.

$$\beta = \arcsin\left(\frac{(x_{hL} + minAbstand + f_laenge_h) - x_{\Omega_2}}{rp}\right) \tag{70}$$

Die Y-Koordinate des Ziels x_{Ziel} wird dann nur noch über den Hilfswinkel β berechnet.

$$\begin{pmatrix} x_{Ziel} \\ y_{Ziel} \\ \theta_{Ziel} \end{pmatrix} = \begin{pmatrix} x_{hL} + minAbstand + f\_laenge\_h \\ y_{\Omega_2} - rp \cdot \cos(\beta) \\ \beta \end{pmatrix} \tag{71}$$

In der Abbildung bestehen die Winkel aus Der abzufahrende Winkel φ_2 wird wie folgt berechnet: $\varphi_2 = \pi/2 - \beta - (\pi/2 - \theta)$ und aufgelöst wie folgt:

$$\begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} 0 \\ \theta - \beta \end{pmatrix} \tag{72}$$

Die Simulation der Einparkphase N-Einparkschritte ergab für Szenario 5 folgendes Ergebnis (siehe Abb. 47):

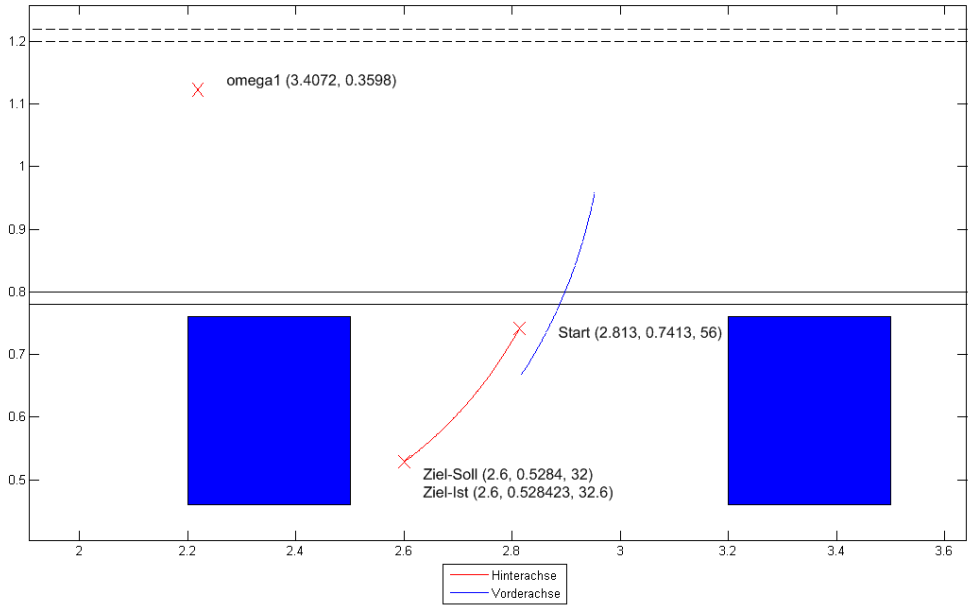


Abb. 47: *Simulationsergebnis der Einparkphase N Einparkschritte, mit Angabe der Startkonfiguration (x,y,theta) und des Soll-Ziel und des Ist-Ziels(x,y,theta), sowie der Angabe des Mittelpunktes des Kreissegments*

Das Fahrzeug startet in der Parklücke in einem Achswinkel von 56 Grad. Das Fahrzeug wird zunächst grade gezogen, um danach die Szenarien 1 und 4 durchführen zu können.

Szenario 6: Einparkschritt Rückwärts

Im Einparkschritt Szenario 6) hat das Fahrzeug einen großen negativen Achswinkel und kann die Zielkonfiguration nicht in einem Graden Achswinkel erreichen. Der Weg besteht somit nur aus einer Rechtskurve. Szenario 6) tritt unter folgender Bedingung ein:

$$richtung == 0 \wedge (x_{\Omega_1} < x_{hL} + minAbstand + f\_laenge\_h) \tag{73}$$

Die Bedingung ist erfüllt, wenn der zweite Kreismittelpunkt Ω_1 in X-Richtung hinter dem Ziel liegt. Das bedeutet dass die Zielkonfiguration auch einen negativen Achswinkel besitzt und somit ist nur eine Kreisbahn notwendig.

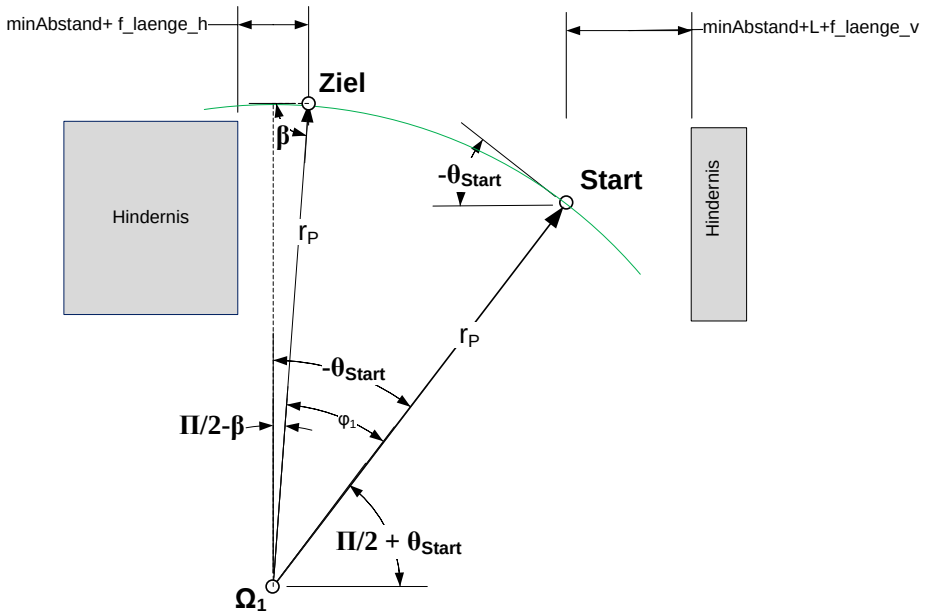


Abb. 48: *Einparkschritt Rückwärts, bestehend aus einer Rechtskurve*

Der Hilfswinkel β wird aus dem Dreieck von Ω_1 und *Ziel* konstruiert.

$$\beta = \arccos\left(\frac{(x_{hL} + minAbstand + f_laenge_h) - x_{\Omega_1}}{rp}\right) \tag{74}$$

Die Y-Koordinate des Ziels x_{Ziel} wird dann nur noch über den Hilfswinkel β berechnet.

$$\begin{pmatrix} x_{Ziel} \\ y_{Ziel} \\ \theta_{Ziel} \end{pmatrix} = \begin{pmatrix} x_{hL} + minAbstand + f\_laenge\_h \\ y_{\Omega_1} + rp \cdot \sin(\beta) \\ -(\pi/2 - \beta) \end{pmatrix} \tag{75}$$

In der Abbildung bestehen die Winkel aus Der abzufahrende Winkel φ_1 wird wie folgt berechnet: $\varphi_2 = \pi/2 - (\pi/2 - \beta) - (\pi/2 + \theta)$ und aufgelöst wie folgt:

$$\begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} \beta - \theta - \pi/2 \\ 0 \end{pmatrix} \tag{76}$$

Die Simulation der Einparkphase N-Einparkschritte ergab für Szenario 6 folgendes Ergebnis (siehe Abb. 49):

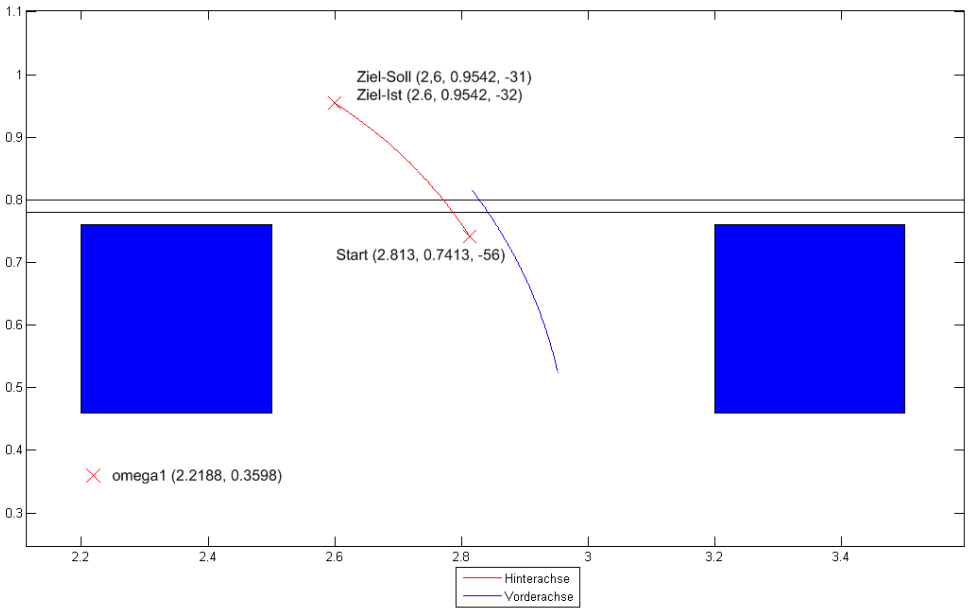


Abb. 49: *Simulationsergebnis der Einparkphase N Einparkschritte, mit Angabe der Startkonfiguration (x,y,theta) und des Soll-Ziel und des Ist-Ziels(x,y,theta), sowie der Angabe des Mittelpunktes des Kreissegments*

Das Fahrzeug startet in der Parklücke in einem Achswinkel von -56 Grad. Um eine größere Y-Tiefe in der Parklücke zu erreichen, wird das Fahrzeug zunächst grade gezogen, um danach die Szenarien 1 und 4 durchführen zu können.

5.5 Fahrzeug ausrichten

Diese Phase wird ausgeführt, wenn das Fahrzeug in Y-Richtung weit genug in der Parklücke steht, jedoch noch nicht einen Achswinkel von 0 +/-5 Grad hat. Somit muss eine Bahn gefunden werden, mit der das Fahrzeug ausgerichtet wird. Die Ausgangsposition x_p,y_p des Fahrzeugs ist in der Parklücke die Position Start. Entweder wird das Fahrzeug an die nächste Position Ziel vorwärts oder rückwärts heran fahren. Als Verfahren zur Ermittlung des Verstärkungsfaktors wird VD_1 verwendet (siehe Kapitel Bahnplanung, Abschnitt Virtuelle Deichsel und Abb. 26).

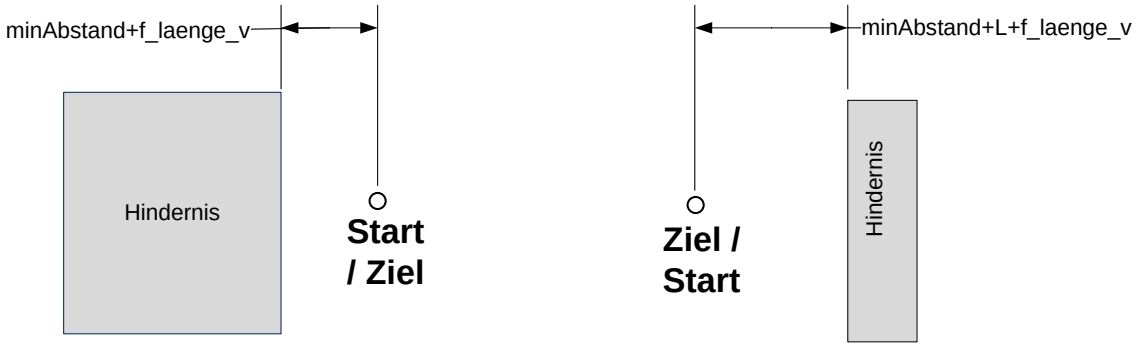


Abb. 50: *Ausganssituation zu begin der Einparkphase*

Die Suchräume werden wie folgt eingeteilt:

Δ **Konfigurationsraum**	$\Delta\theta[Grad]$
A	[0,5)
B	[5,10)
C	[10,15)
D	[-15,-20)

Tabelle 6: *Suchräume in der Phase Fahrzeug ausrichten, mit dem Verfahren VD_1, abgetastet in 1 Grad Schritten*

Simulations ergebnis		Kennwerte des Verstärkungsfaktors						
Delta Konfigurations-raum	Resultierender Verstärkungs-faktor	Ø Abweichung in Y-Richtung |yp - y_ziel| [m]	Ø Abweichung des Achswinkels |theta_ist- theta_ziel| [Grad]	Anzahl der Abweichungen in Y-Richtung > 0.05 m	Anzahl der Abweichungen des Achswinkels > 5 Grad	maximale Abweichung in Y-Richtung [m]	maximale Abweichung des Achswinkels [Grad]	Anzahl an Simulations-iterationen
A	12	0.00161	0.77393	0	0	0.00346	2.13987	5
B	1	0.01300	2.30848	0	0	0.01690	2.37750	5
C	1	0.02319	1.53320	0	0	0.02758	2.07802	5
D	1	0.03528	0.91814	0	0	0.04110	2.11026	5

Abb. 51: *Auswertung der Ermittlung des geeignetsten Verstärkungsfaktors*

Zur Ermittlung des geeignetsten Verstärkungsfaktors, wurden die Suchräume in 1 Grad Schritten abgetastet. Es wird der Verstärkungsfaktor gewählt, der die geringste durchschnittliche Abweichung aufweist.

$$verst = \begin{cases} 1, \text{ wenn} & q \notin Suchraum\_A \\ 12, \text{ wenn} & q \in Suchraum\_A \end{cases} \tag{77}$$

Wenn die Fahrzeugkonfiguration keinem Suchraum zugeordnet werden kann, dann wird der Verstärkungsfaktor auf 1 gesetzt.

Am Ende der Phase kann das Fahrzeug durch dieses Manöver wieder aus der Y-Parklückentiefe heraus kommen. Die Entscheidungseinheit wird darauf reagieren und ggf. Phase 4. „N Einparkschritte“ wiederholen.
Die Simulation der Einparkphase Fahrzeug ausrichten ergab folgendes Ergebnis (siehe Abb. 52):

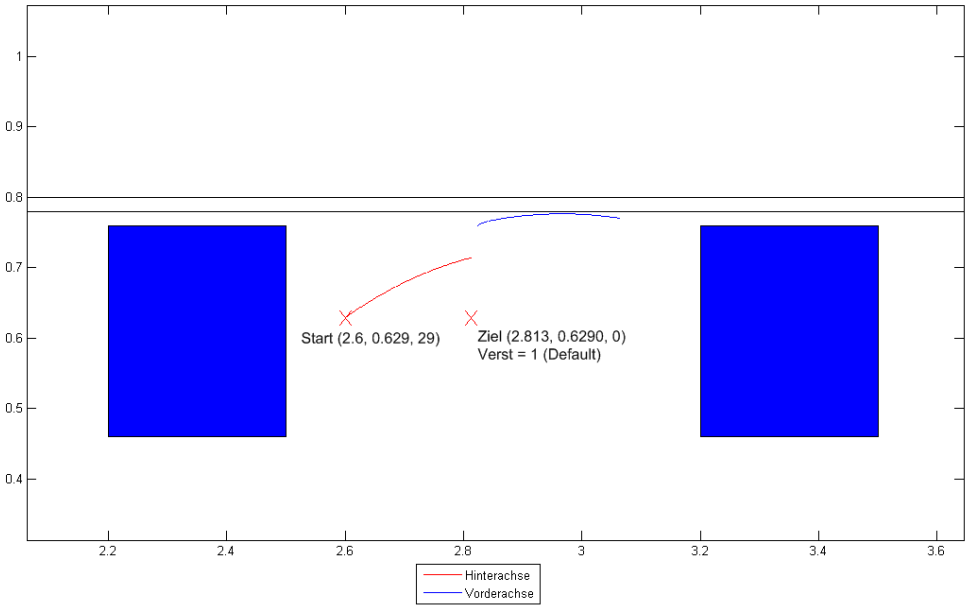


Abb. 52: *Simulationsergebnis der Einparkphase Fahrzeug ausrichten, mit Angabe des Verstärkungsfaktors, der Start- und Zielposition*

Für die Konfiguration des Fahrzeugs existiert kein Suchraum, deshalb wird der Standard-Wert für den Verstärkungsfaktor 1 gewählt.

5.6 Simulationsergebnis des kompletten Einparkmanövers

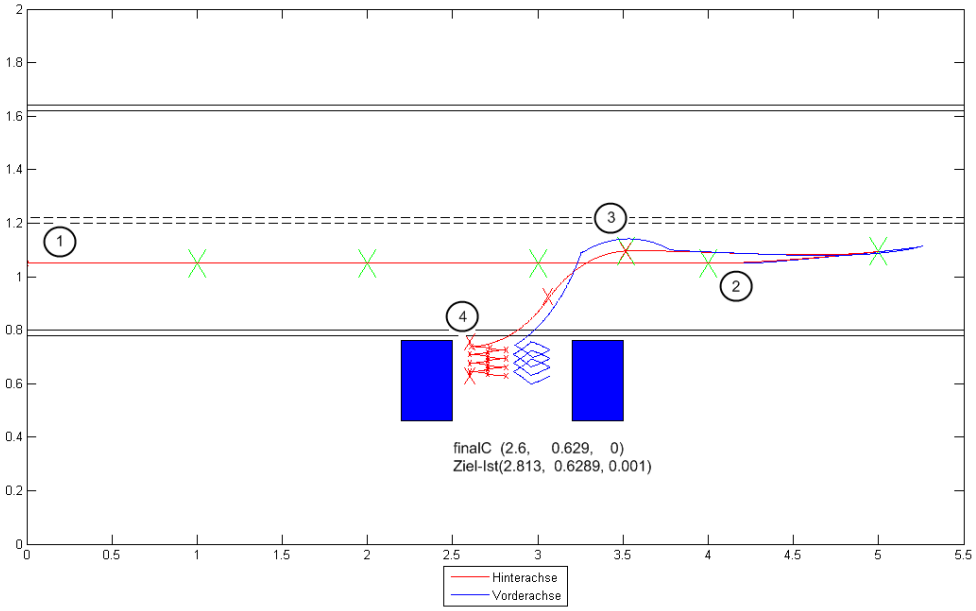


Abb. 53: *Das komplette Einparkmanöver von der Startlinie aus zum Ziel innerhalb der Parklücke. Die Zielposition finalC (x,y,theta) wurde in 4 Phasen erreicht. 1. Spurführung, 2. Positionierung, 3. Erster Einparkschritt, 4. N-te Einparkschritte*

Das Simulationsergebnis (siehe Abb. 53) des Einparkmanöver von der Fahrspurmitte der rechten Fahrbahn zur finalen Einparkpostion $finalC$, ergab bei konstanter Geschwindigkeit von 0,1 m/s folgende Kennwerte:

- Einparkphasen: 1. Spurführung (5 mal), 2. Positionierung (2 mal), 3. erster Einparkschritt (1 mal), 4. N-te Einparkschritte (7 mal)
- Anhaltepositionen: 18
- gefahrener Weg: 8,9147 m
- Simulationszeit: 96,04 s
- Achswinkelabweichung am Ziel: 0,001 Grad
- Y-Abweichung am Ziel: 0,0001 m

6 Weitere Entwicklungsschritte des Fahrzeuges

Der Einparkassistent wird ein System eingebunden, in der drei Assistenzsysteme implementiert werden.

- Einparkassistent
- Assistent zum autonomen Fahren von Rundstrecken ohne Hindernisse
- Assistent zum autonomen Fahren von Rundstrecken mit Hindernissen

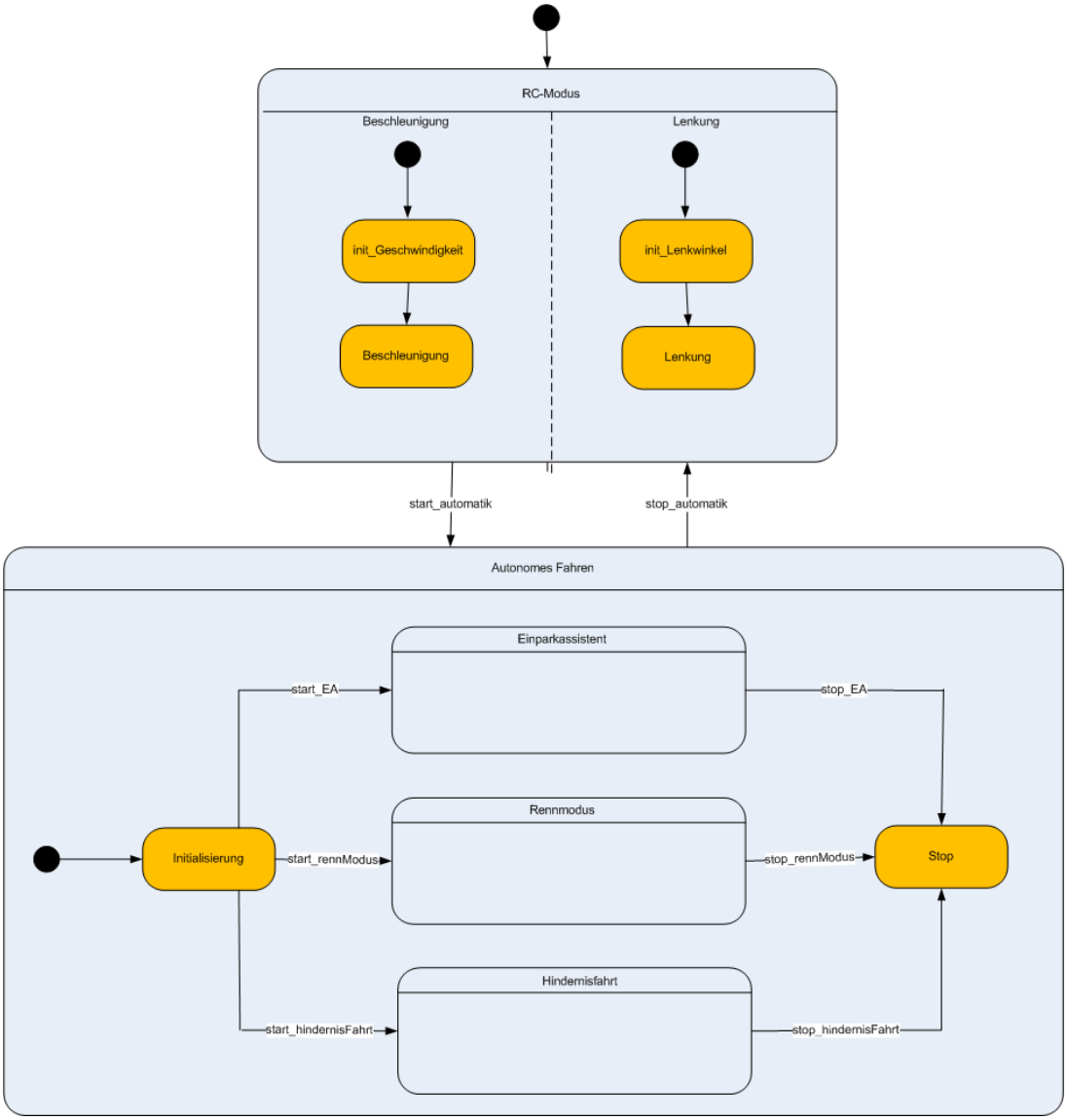


Abb. 54: *UML Hierarchisches Zustandsdiagramm. Hauptautomaten des Fahrzeugsystems sind RC-Modus und autonomes Fahren*

Die beiden Hauptzustände in der obersten Hierarchieebene sind (vgl. Abb. 54):

- RC-Modus: manuelle Steuerung des Fahrzeugs über eine Funkverbindung
- Autonomes Fahren: Es besteht aus drei weiteren Subzuständen: Einparkassistent, Hindernisfahrt und Rennmodus. Das sind die 3 Disziplinen die beim Carolo Cup Verwendung finden

Der Subautomat Einparkassistent (vgl. Abb. 54) besteht aus mehreren Bahnplanungsverfahren, die sequentiell ausgeführt werden (siehe Kapitel 5).

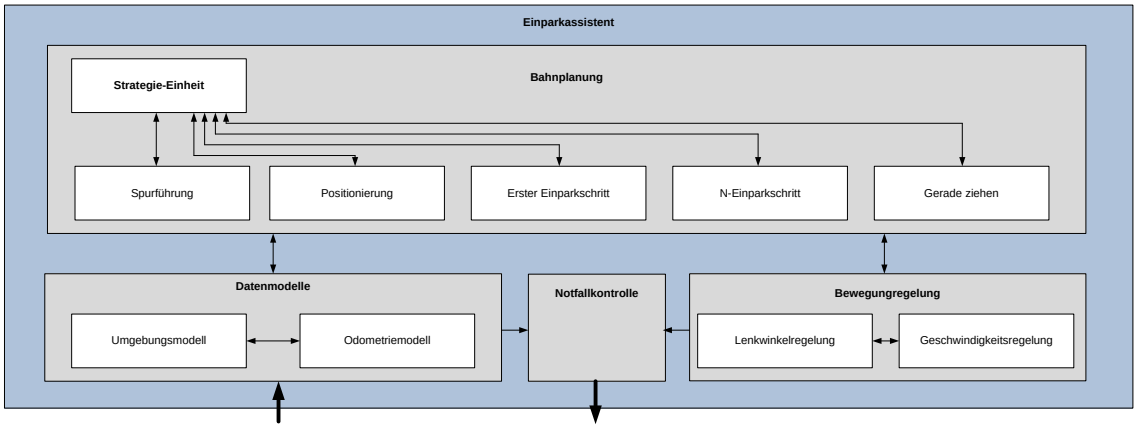


Abb. 55: *Kompositionsdiagramm des Einparkassistenten*

Die Strategieeinheit (siehe Kapitel 5, Abb. 29) trifft die Auswahl des Bahnplanungsverfahrens, das je nach Einparkphase und Fahrsituation verwendet wird. Die Bahnplanungsverfahren des „ersten Einparkschritts“ und „N-Einparkschritte“ berechnen Wege und Zielpositionen. Diese müssen an eine Lenkwinkelregelung und Geschwindigkeitsregelung weiter geleitet werden. Die Einparkphasen der Virtuellen Deichsel „Spurführung“, „Positionierung“ und „Fahrzeug ausrichten“ berechnen Zielpositionen und den Verstärkungsfaktor der Lenkwinkelregelung. Eine Geschwindigkeitsregelung muss das Fahrzeug auch bei den Verfahren „Positionierung“ und „Fahrzeug ausrichten“ eine Regelung auf eine Zielposition beinhalten.

Zur Vereinfachung des Einparkassistenten wird vorgeschlagen, eine Komponente „Datenmodelle“ zu erstellen. Sie kommuniziert mit den Komponenten, die Umgebungs- und Odometriedaten abholen, auswerten und ggf. korrigieren. Sie bereiten die Daten vor und leiten sie an die Komponente Bahnplanung weiter.

Eine Notfallkontrolle muss im Notfall die Kontrolle über die Aktorik entweder an die manuelle Steuerung übergeben oder das Fahrzeug autonom eine Fahrentscheidung treffen.

Chassis

Mit einem geringeren Kollisionsradius des Fahrzeugs können die Abstände zu den Hindernissen verkleinert werden. In einem Einparkschritt kann so eine größere Tiefe erzielt werden.

Maßnahmen:

- kleinere Spurweite L
- größeren maximalen Lenkwinkel α
- eine Abrundung der Eckpunkte des Fahrzeugs verkleinert den äußeren Kollisionsradius (siehe Kollisionsradius Abb. 16)

Zusätzlich zur Einparkphase Spurführung muss die Parklücke gesucht werden (siehe Abb. 56).

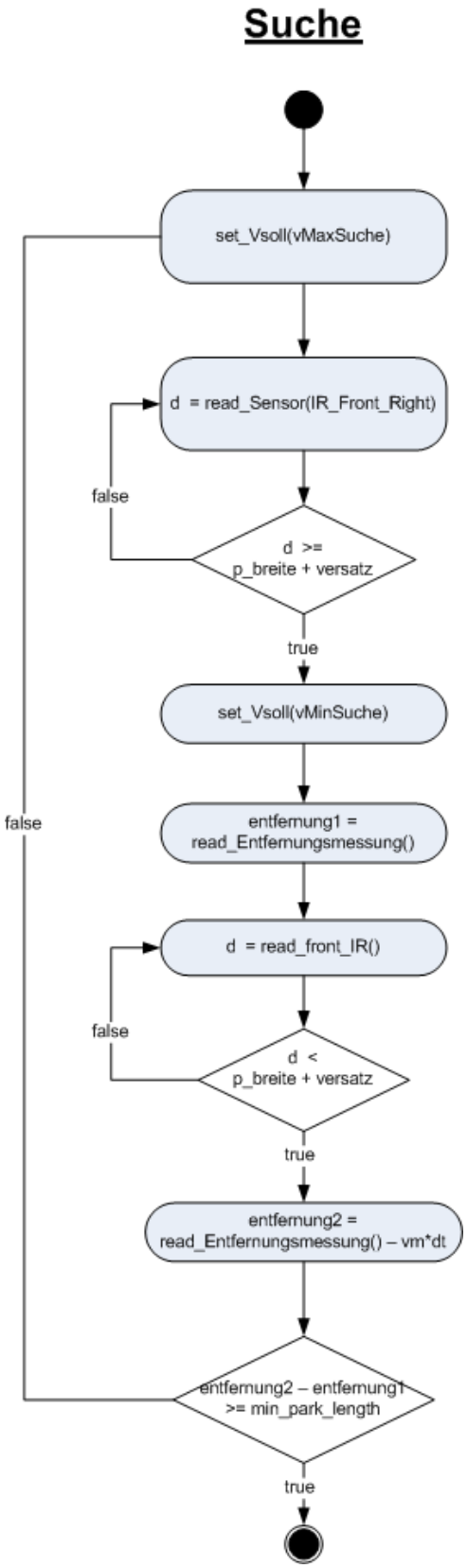


Abb. 56: *UML-Aktivitätsdiagramm Parklückensuche*

Dazu müssen noch die Anforderungen des Carolo Cup Regelwerks umgesetzt werden, wie die manuelle Steuerung (siehe Abb. 54) und die LED-Blinkersteuerung.

Virtuelle Deichsel

Die Anzahl der Suchräume sollte insgesamt vergrößert werden. Zur Abtastung der Suchräume verbessern kleinere Schritte die Suche nach Parametern, dass z. B. die Abtastung des Achswinkelintervall nicht in ein Grad-Schritten , sondern in 0,1 Grad Schritten erfolgt. Die große Anzahl von Konfigurationsräumen und deren Zuordnung zu einem Verstärkungsfaktor müsste dann automatisch erfolgen, weil der Aufwand der Implementierung sonst zu groß wäre. Das heißt es wird eine Konvertierungsfunktion gebraucht, die eine Fahrzeugkonfiguration einem Suchraum zuordnet und deren Parameter Zielposition und Verstärkungsfaktor liefert.

Die Geschwindigkeit innerhalb der Suchräume beträgt immer 0.1 m/s. Wenn die Suchräume um die Dimension Geschwindigkeit erweitert würden, dann könnten mehrere Geschwindigkeiten bei der Virtuellen Deichsel verwendet werden.

Zur Spurführung Rundkursen kann die Virtuelle Deichsel verwendet werden. In einem Abstand von 50 oder 100 cm kann der Achswinkel zu einer Kurve ermittelt werden. Bei Drehung des Koordinatensystems auf diesen Achswinkel kann eine Regelung erfolgen. Mit dem Bahnplanungsverfahren VD_2 wird das Fahrzeug auf einen Zielachswinkel von 0 Grad geregelt. Es müssen somit Stützstellen aus der Fahrspur genommen werden, die Abschnittweise als Zielkoordinate der Virtuellen Deichsel dienen.

Weg-Bogenlängen Technik

Eine Erweiterung der Kreistechnik hin zu einer vollwertigen Weg-Bogenlängen Technik. Es umfasst folgende Arbeiten:

- CC-Bahnsegmente zwischen den minimalen Kreisen, die aus Wendeklothoiden konstruiert werden
- Zeitoptimierte Bahnen, die aus Bahnsegmenten mit variablen Wenderadien bestehen
- Bewegungsregelung der Weg-Bogenlängen Technik

Der Vorteil der Weg-Bogenlängen Technik ist die Freiheit eine Bahn zu planen, die nicht auf Geometrischen Methoden, sondern auf einfachen Lenkmanövern basiert. Mit ihnen können kürzeste Wege erstellt werden.

Erster Einparkschritt

Als erster Einparkschritt wurde ein Verfahren gewählt, mit dem Zwischenziele in einem Achswinkel von 0 Grad erreicht werden. Eine Bahnplanungsverfahren sollte entwickelt werden, das im ersten Schritt direkt auf das Ziel zufährt und danach die N-Einparkschritte durchführt.

Odometrie

Zur Geschwindigkeitserfassung bietet sich die Hinterradgeschwindigkeit an, da die Hinterachse starr ist und so einfacher zu ermitteln ist. Dies kann über ein Inkrementalgeber realisiert werden. Die Odometrie sollte dann die Gleichungen des Kinematischen Einspurfahrzeugmodells für heckangetriebene Fahrzeuge verwenden, da die Gleichungen aus weniger Thermen besteht. Die Hinterradposition ist ausreichend, somit wären weniger Gleichungen zu implementieren.

Kollisionserkennung

In der Parklücke ist der Platz begrenzt. Die Sensoren haben einen toten Bereich in dem sie keine Abstandswerte ermitteln. Das Fahrzeug muss deshalb aus Sicherheitsgründen anhalten. Innerhalb dieses toten Bereichs könnte das Fahrzeug trotzdem weiterfahren, wenn der Mindestabstand eingehalten wird. Die Hindernisumrandungen könnten als Funktionen abgespeichert werden, die eine Kollisionserkennung nutzt. Der Mindestabstand wäre so nicht mehr über die Sensoren eingeschränkt.

Zur Hinderniserkennung sind beim Einparkmanöver nur wenige Kollisionspunkte und -flächen zu berücksichtigen. Das sind die Fahrzeugaußenflächen, die mit den Eckpunkten des Hindernis kollidieren könnten (siehe Abb. 57).

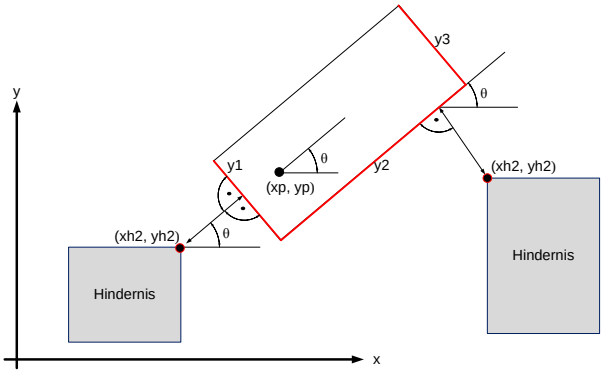


Abb. 57: *Szenario 1: Kollision des Fahrzeugs mit den Hinderniseckpunkten*

Und es sind die Eckpunkte des Fahrzeugs, die mit den Innenflächen der Hindernisse zusammen treffen könnten (siehe Abb. 58).

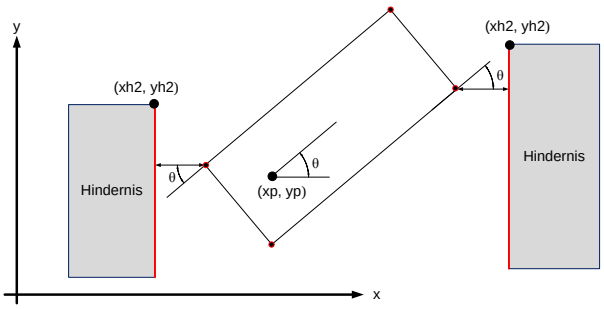


Abb. 58: *Szenario 2: Kollision des Fahrzeugs mit den Hindernisfla"chen*

Die Flächen können als Gerade und die Eckpunkte als x-y-Positionen abgespeichert werden.

7 Zusammenfassung

Es ist ein Einparkassistent für ein autonomes Fahrzeug entwickelt worden. Der Schwerpunkt liegt in der Bahnplanung des Einparkmanövers von einer Startposition bis hin zu einem Ziel innerhalb der Parklücke. Es wurden fünf Einparkphasen identifiziert, für die jeweils eigene Bahnplanungsverfahren und Matlab-Modelle entwickelt wurden. Realisiert wurden die fünf Einparkphasen durch zwei Fahrtechniken.
Eine ist die Kreistechnik, mit der Kreisbahnen zu einem Ziel geplant und gefahren werden. Die Bahnsegmente bestehen aus zwei Kreisbahnen, die aufgrund des maximalen Lenkwinkels eine maximale Krümmung und minimale Krümmung besitzen. Die Verbindung erfolgt über Anhaltepositionen an den Berührungspunkten. Es werden hiermit kürzeste Wege erzeugt, die in einem definierten Abstand zu den Hindernissen führen. Realisiert wurde es, indem Pfade zu Zwischenziele erzeugt wurden, die einen Achswinkel von 0 Grad besitzen.
Die Fahrtechnik Virtuelle Deichsel ist eine Lenkwinkelregelung. Über die Wahl der Regelungsparametern wird Einfluss auf das Fahrverhalten genommen. Ausgehend von einer unbekannten Position des Fahrzeugs zu einem Ziel, werden die geeignetsten Parameter gesucht. Hierfür wurden Intervalle von X-, Y- und Achswinkel Differenzen zum Ziel gebildet, in denen die Parameter ausgewählt wurden, bei denen die durchschnittliche Simulationsabweichung am Ziel am geringsten ist.

Die zwei Fahrtechniken werden in fünf Einparkphasen verwendet. Die erste Einparkphase ist die „Spurführung“, die das Fahrzeug entlang der Fahrspurmitte führt und in der die Suche einer Parklücke erfolgt. Das Bahnplanungsverfahren VD_2 realisiert eine geregelte Fahrt entlang der Fahrspurmitte und führt das Fahrzeug parallel zum Fahrbahnrand. Ist eine Parklücke mit ausreichender Breite gefunden, wird in der Einparkphase „Positionierung“ das Fahrzeug zu einer geeigneten Ausgangsposition für das erste Einparkmanöver geführt. Das Verfahren VD_3 positioniert das Fahrzeug, mit einer Achswinkeldifferenz von +/- 5 Grad, dort hin. Der „erste Einparkschritt“ wird das Fahrzeug in eine maximale Tiefe innerhalb der Parklücke führen. Die Realisation erfolgte über die Kreistechnik, aufgrund der Ausnutzung des maximalen Lenkwinkels. Weitere „N-Einparkschritte“ sind für das Erreichen weiterer Einparktiefe erforderlich. Verwendet wurde die Kreistechnik, wegen der maximalen Einparktiefe in einem Einparkschritt. Eine Regelung ist aufgrund der kurzen Distanz zwischen vorderer und hinterer Parklückenbegrenzung nicht notwendig. Beim Erreichen des Ziels wird das „Fahrzeug ausgerichtet“, um eine maximale Abweichung des Achswinkels von 0 +/- 5 Grad zu gewährleisten. Realisiert wurde es mit dem Verfahren VD_1.

Die fünf entwickelten Bahnplanungsverfahren der Einparkphasen decken alle zu erwartenden Situationen während des Einparkvorgangs ab.

Glossar

Trajektorie	Ein Weg oder Bahn die das Fahrzeug mit einer Geschwindigkeit fahren kann
Bahnsegment	Ein Abschnitt einer Bahn, gekennzeichnet durch gleichbleibende Eigenschaften wie z. B. eine Kurve mit gleichbleibender Krümmung
Konfiguration	Eindeutige und minimale Beschreibung der Lage des Fahrzeugs x_p, y_p, θ
Konfigurationsraum	Menge aller Konfigurationsvarianten innerhalb der Dimensionsgrenzen
Arbeitsraum	Raum/Ebene in dem sich das Fahrzeug bewegen kann
Shortest Path	Kürzester Pfad in einem kollisionsfreien Arbeitsraum
Graph	Verkettung von Knoten (Konfigurationen) über Kanten (Steuerungsgrößen)
System on Chip	Mikrocontrollersystem auf einem Chip integriert
Kinematik	Lehre der Bewegung von Körpern, ohne Berücksichtigung von einwirkenden Kräften
DGS 1. Ordnung	Mehrere Funktionen, die in Form einer ersten Ableitung notiert sind
Einparkschritt	Ein Fahrmanöver zur Parklücke, in Vorwärts- oder Rückwärtsfahrt
UML	Graphische Beschreibungssprache von Softwaresystemen
SysML	Graphische Beschreibungssprache von Hardware- und Softwaresystemen
Weg-Bogenlängen Technik	Graphische Beschreibungssprache von Softwaresystemen
Kreistechnik	Wegkurven entlang von Kreisen planen
CC Technik	Fahrmanöver bei der der Lenkwinkel bei konstanter Geschwindigkeit geändert wird
Virtuelle Deichsel	Bahnplanungsverfahren mit integrierter Lenkwinkelregelung auf die Zielkonfiguration
Odometrie	Lagebestimmung des Fahrzeugs über ein Differentialgleichungssystem
Chassis	Der tragende Rahmen des Fahrzeugs
Einspurfahrzeugmodell	Vereinfachtes Kinematische Modell des Fahrzeugs, bei dem die Räder einer Achse zusammengeschoben werden
Aktorik	Wandeln elektrische Signale in mechanische Energie um
Sensorik	Wandeln physikalische Größen in digitale Größen um
Suchraum	Ein abgetasteter Raum aus Startkonfigurationen in dem ein Bahnplanungsverfahren simuliert und ausgewertet wird

Intellectual Property	**IP:** Eine Digitale Funktionseinheit, die einmal spezifiziert und mehrfach instanziiert werden kann. Sie kann mit anderen IPs über interne Bussysteme oder Signalleitungen kommunizieren
Linearität	Eine Änderung des Eingangs bewirkt eine proportionale Veränderung des Ausgangs
Numerische Lösung	angenäherte Lösung über Näherungsverfahren
Analytische Lösung	eindeutige Lösung über Umformung
Bogenlänge	Länge eines Kreisbogens
Klothoide	Bahnsegment, bei der der Lenkwinkel umgeschlagen wird und das bei konstanter Geschwindigkeit
Continuous Curvature	Bahnsegment mit den Kennwerten einer kontinuierlichen Krümmungsänderung
Bahnsegment	Ein Abschnitt der Trajektorie

Symbolverzeichnis

ABS **A**nti**b**lockier**s**ystem

CC **C**ontinuous-**C**urvature

DC **D**irect **C**urrent: Gleichstrom

DGS **D**ifferential**g**leichungs**s**ystem

DSP **D**igital **S**ignal **P**rocessor

EDK **E**mbedded **D**evelopment **K**it

ESC **E**lectronic **S**tability **C**ontrol

FPGA **F**ield **P**rogrammable **G**ate **A**rray

FSL **F**ast **S**ystem **L**ink

HW **H**ard**w**are

I/O **I**nput/**O**utput

IP **I**ntellectual **P**roperty

LED **L**ight **E**mitting **D**iode

ODE **O**rdinary **d**ifferential **e**quation

OPB **O**n Chip **P**eripheral **B**us

PLB **P**rocessor **L**ocal **B**us

RRT **R**apidly-exploring **R**andom **T**ree

SDK **S**oftware **D**evelopment **K**it

SFP **S**hortest **f**easible **p**ath

SW **S**oft**w**are

SysML **Sys**tems **M**odeling **L**anguage

UML **U**nified **M**odeling **L**anguage

VD_N Bahnplanungsverfahren **V**irtuelle **D**eichsel N

VHDL **V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Academic dictionaries and encyclopedias. http://de.academic.ru/dic.nsf/dewiki/98330. abgerufen am 10.03.2010.

[2] *Manual: 128-Bit Processor Local Bus Architecture Specifications*, ibm, version 4.7 edition.

[3] *Manual: EDK Concepts, Tools, and Techniques*, xilinx, edk 11 edition.

[4] Modellbau wiki. http://www.modellbau-wiki.de/wiki/Differentialgetriebe. abgerufen am 10.05.2010.

[5] Rc modellbau lexikon. http://www.modell-bau.org/achs-differential/. abgerufen am 10.03.2010.

[6] *User Manual A.I. Brushless Reverse digital, LRP*.

[7] Wiki rc-networks. http://wiki.rc-network.de/Brushless-Motor. abgerufen am 10.05.2010.

[8] Wikipedia - atan2. http://upload.wikimedia.org/wikipedia/commons/thumb/a/a8/Arctangent2.svg/220px-Arctangent2.svg.png. abgerufen am 10.05.2010.

[9] G. C. A. Bicchi and C. Santilli. *Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles*, volume 16 of *Journal of Intelligent and Robotic Systems*. Springer, 1996. 387–405 pp.

[10] J. D. B. Müller. Zweistufige trajektorienplanung für das automatische einparken. *AUTOREG 2006, VDI-Berichte Nr. 1931*, 2006. abgerufen am 01.03.2010.

[11] J. D. B. Müller. *Two-step Trajectory Planning for Automatic Parking*. PhD thesis, Technische Fakultät der Universität Erlangen, http://www.rt.e-technik.uni-erlangen.de/FGnls/papers/diss_Mueller_2009.pdf, 2009. abgerufen am 01.03.2010.

[12] S. G. B. Müller, J. Deutscher. Trajectory generation and feedforward control for parking a car. *International Symposium on Intelligent Control IEEE*, pages 163 – 168, Oktober 2006.

[13] S. G. B. Müller, J. Deutscher. Continuous curvature trajectory design and feedforward control for parking a car. *Control Systems Technology, IEEE Transactions on*, 15:541 – 553, May 2007.

[14] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, July 1957.

[15] B. K. E. Szadeczky-Kardoss. Continuous-curvature paths for mobile robots. Technical report, Dept. of Control Engineering and Information Technology, Budapest University of Technology and Economics,, http://mycite.omikk.bme.hu/doc/54630.pdf, June 2008. abgerufen am 01.03.2010.

[16] D. V. e.V. Die besten beifahrer. http://www.bester-beifahrer.de/die_besten_beifahrer.html. abgerufen am 03.03.2010.

[17] A. O. F. Cuesta, F. Gomez-Bravo. Parking maneuvers of industrial-like electrical vehicles with and without trailer. *Industrial Electronics, IEEE Transactions on*, 51:257 – 269, April 2004.

[18] S. Farshbaf-Masalehdan. Abstandsregelung für ein autonomes fahrzeug implementiert auf einer fpga basierten soc plattform. Bachelorarbeit.

[19] T. Fraichard and A. Scheuer. From reeds and shepp's to continuous-curvature paths. *IEEE Transaction on Robotics*, 20(6):367–393, Dezember 2004.

[20] G. W. H. Winner, S. Hakuli. *Handbuch Fahrerassistenzsysteme*. Vieweg, 2009. 471–476 pp.

[21] HHLA. Containerterminal altenwerder. http://www.container-terminal-altenwerder.de/. abgerufen am 10.05.2010.

[22] J.-P. Laumond. *Robot Motion Planning and Control*. Springer, http://homepages.laas.fr/jpl/book.html, lectures notes in control and information edition, 1998. abgerufen am 01.03.2010.

[23] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Department of Computer Science Iowa State University, http://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.pdf, 1998. abgerufen am 01.03.2010.

[24] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[25] N. Liu. Ein automatischer parkassistent auf basis einer laserscanner-abstandserfassung für ein fahrerloses transportsystem. Masterarbeit.

[26] D. Mellert and C. Eskikaya. Hsc odometrie. Projektarbeit SS09.

[27] L. Papula. *Mathematische Formelsammlung*, volume 9. Vieweg, 2006. 139 pp.

[28] Q. d. Zhu, Y. b. Wu, G. q. Wu, X. Wang . An improved anytime rrts algorithm. In *Artificial Intelligence and Computational Intelligence, International Conference on*, pages 268 – 272, November 2009.

[29] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[30] Scantec. Brushless gleichstrom motor. http://www.scantec.de/modules.php?name=News&file=article&sid=346. abgerufen am 03.03.2010.

[31] D. Schetler. Automatischer ausweichassistent mit einer laserscanner - basierten abstandsregelung für ein fahrerloses transportsystem. Masterarbeit.

[32] A. M. Shkel and V. J. Lumelsky. On calculation of optimal paths with constrained curvature: The case of long paths. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 4, pages 3578 – 3583, April 1996.

[33] Tamiya. 1/10 r/c enzo ferrari. http://www.tamiya.com/english/products/58302enzo/index.htm. abgerufen am 01.03.2010.

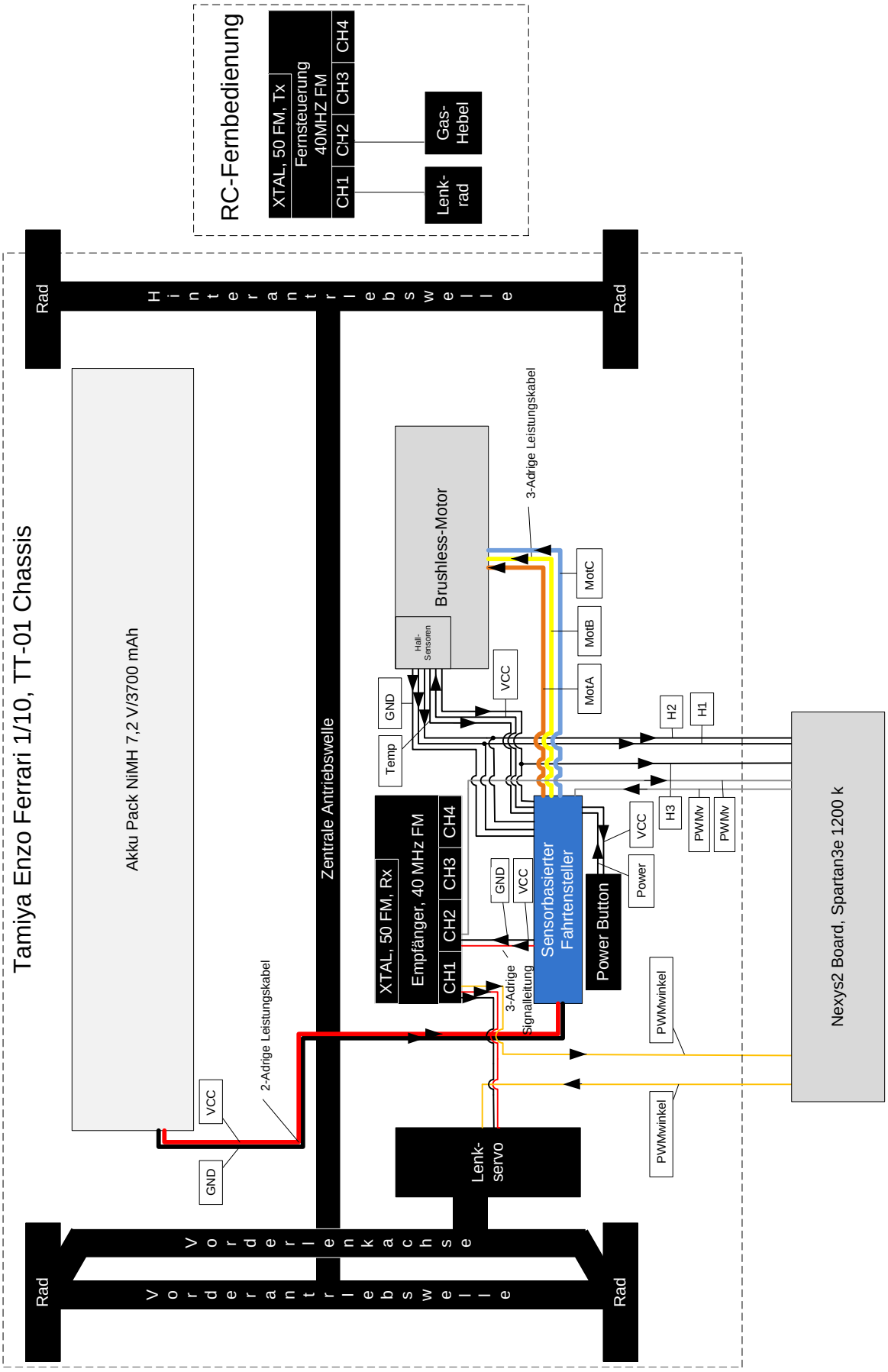
[34] B. Triggs. Motion planning for nonholonomic vehicles: An introduction.

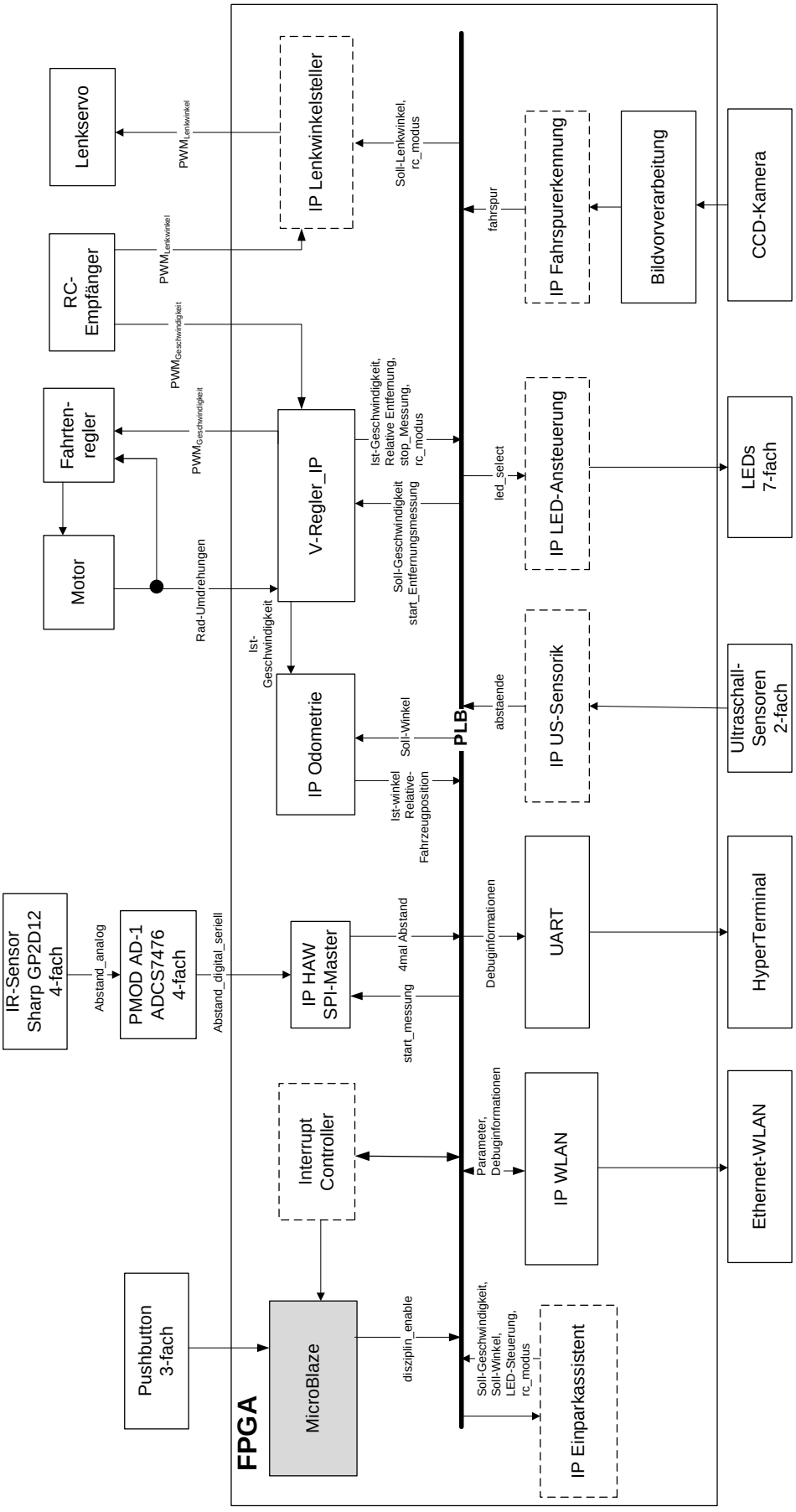
[35] D. Wilde. Computing clothoid segments for trajectory generation. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2440 – 2445, Dezember 2009.

A Ergänzungen zur Plattform

Arbeitsplanung
für den Aufbau, Entwurf und die Implementation
eines SoC-Einparkassistenten

Kennung	*Aufgabenname*	*Kommentare zur Aufgabe*
1	**Allgemeine Vorbereitung**	
2	Vorüberlegungen Stromversorgung	Benötigte Leistung berechnen
3	Vorüberlegungen EMV	Motorsteller und Motor erzeugen hohe el. Magn. Felder, wie abschirmen / entstören (Materalien, Kondensatoren, Lage)
4	Skizze SoC-IPs	Allgemeine Aufbau des SoC mit Anschlüsse der IP´s, das Zusammenwirken grob erfassen
5	Skizze Elektrotechnischen-Komponenten	Allgemeine Verkabelung der wichtigsten Komponenten grob Skizzieren
6	**Grob-Entwurf Einparkassistent**	
7	Recherche zu den Bahnplanungsalgorithmen	Vergleich der vorhandenen Lösungen
8	Simulation der Bahnplanungsalgorithmen	Simulation ausgewählter Techniken
9	Vorüberlegungen der Phasen vom Startzustand bis zum Einparken	Aktivitätsdiagramm der einzelnen Phasen
10	Auswahl der Fahrtechnik zu jeder Einparkphase	Simulation jeder einzelnen Phase mit Matlab
11	grober Entwurf des Einparkassistenten	Gesamt-Simulation mit Matlab über einen Zustandsautomaten
12	grober Entwurf des SoC-Gesamtsystems	Zeichnung aller IP´s (Komponentendiagramm) und Ablaufdiagramm (Sequenzdiagramm) wie Komponenten zusammenwirken
13	**Elektronik integrieren**	
14	Detailzeichnung der Elektrotechnischen Komponenten	Zeichnung aller el. Komponenten mit Nexys-Board, mit allen Anschlüssen
15	Einkaufsliste erstellen	Sensoren, Elektronik, ……, alles was benötigt wird kaufen
16	Verkabeln, Löten	zusammenbauen
17	Gesamttechnische Analyse	Funktionieren alle Bauteile korrekt, Testen der Anschlüsse
18	**IP Implementation und Testen**	
19	IP Lenkwinkelsteller	Analyse des Servos, IP implementieren, Zeitverhalten ermitteln
20	IP Odometrie	IP implementieren, Zeitverhalten ermitteln
21	IP-LED-Ansteuerung	LEDs verkablen, IP implementieren
22	IP Geschwindigkeitsregler	Analyse des V-Reglers, IP implementieren, Zeitverhalten ermitteln
23	IP SPI (US)	Analyse des US-Sensoren, IP implementieren, Zeitverhalten ermitteln
24	IP SPI (IR)	Analyse des IR-Sensoren, IP implementieren, Zeitverhalten ermitteln
25	IP Spurführung (LenkwinkelRegler)	Analyse der Kamera, Bilder filtern, Regler entwickeln, IP implementieren, Zeitverhalten ermitteln
26	IP-Gesamtanalyse	Funktionieren alle IP`s noch korrekt, Verhalten des Gesamtsystems korrekt, Fehlersuche
27	**Fein-Entwurf Einparkassistent**	
28	Sicherheitsfunktionen integrieren	Sicherheitsabstand und Kollisionserkennung in IP implementieren, Zeitverhalten ermitteln
29	Implementation Einparkassistent	IP implementieren, Zeitverhalten ermitteln
30	HW/SW-Entwurf Einparkassistent	Aktivierung/Deaktivierung des Einparkassistenten über Fernbedienung, ein oder mehrere kleine IP´s entwickeln, Wiederverwendbarkeit erhöhen (Abhängigkeiten verringern)
31	H/SW-Implementation Einparkassistent	Instanzieren, testen und korrekte Funktionsweise nachprüfen
32	**Optimierung des Einparkassistenten**	
33	Continous Curvatures als Fahrtechnik	Die Reeds and Shepps Technik mit der CC-Technik erweitern und testen
34	Regelung der Fahrmanöver	Eine verbesserte Regelung für die Fahrmanöver entwickeln und testen


RC-Fernbedienung
XTAL, 50 FM, Tx
Fernsteuerung 40MHZ FM
CH1
CH2
CH3
CH4
Lenk-rad
Gas-Hebel
Tamiya Enzo Ferrari 1/10, TT-01 Chassis
Rad
Hinterradantriebswelle
Rad
Akku Pack NiMH 7,2 V/3700 mAh
Zentrale Antriebswelle
Brushless-Motor
Hall-Sensoren
3-Adrige Leistungskabel
MotC
MotB
MotA
VCC
GND
Temp
H2
H1
H3
PWMv
PWMv
XTAL, 50 FM, Rx
Empfänger, 40 MHz FM
CH1
CH2
CH3
CH4
GND
VCC
Sensorbasierter Fahrtensteller
VCC
Power
Power Button
3-Adrige Signalleitung
2-Adrige Leistungskabel
VCC
GND
Lenk-servo
PWMwinkel
PWMwinkel
Nexys2 Board, Spartan3e 1200 k
Rad
Vorderradantriebswelle
Rad


FPGA
PLB
MicroBlaze
Interrupt Controller
IP HAW SPI-Master
IP Odometrie
V-Regler_IP
IP Lenkwinkelsteller
IP LED-Ansteuerung
IP US-Sensorik
IP WLAN
IP Einparkassistent
IP Fahrspurerkennung
Bildvorverarbeitung
UART
Pushbutton 3-fach
IR-Sensor Sharp GP2D12 4-fach
PMOD AD-1 ADCS7476 4-fach
Lenkservo
RC-Empfänger
Fahrten-regler
Motor
CCD-Kamera
LEDs 7-fach
Ultraschall-Sensoren 2-fach
HyperTerminal
Ethernet-WLAN
Abstand_analog
Abstand_digital_seriell
4mal Abstand
start_messung
Debuginformationen
Parameter, Debuginformationen
disziplin_enable
Soll-Geschwindigkeit, Soll-Winkel, LED-Steuerung, rc_modus
Rad-Umdrehungen
Ist-Geschwindigkeit
Soll-Winkel
Ist-winkel Relative-Fahrzeugposition
abstaende
Soll-Geschwindigkeit start_Entfernungsmessung
Ist-Geschwindigkeit, Relative Entfernung, stop_Messung, rc_modus
led_select
fahrspur
Soll-Lenkwinkel, rc_modus
PWMLenkwinkel
PWMLenkwinkel
PWMGeschwindigkeit
PWMGeschwindigkeit

B Fahrtechniken

B.1 Polynome

Im folgenden Abschnitt wird eine Technik gezeigt, wie mit Funktionen eine Bahn erzeugt werden kann. Funktionen bzw. Polynome werden meist im Zusammenhang mit Splines verwendet. Splines sind eine verbreitete Methode zur Bahnplanung, sie bestehen Abschnittsweise aus Polynomen n-ten Grades. Splines werden über Stützstellen erzeugt, der Abschnitt zwischen den Stützstellen sind Polynome. Eine kubische Funktion hat folgende Notation:

$$f_{(x)} = a * x^3 + b * x^2 + c * x + d \tag{78}$$

Die unbekannten Parameter a, b, c und d werden über vier Randbedingungen ermittelt.

1. Y-Startkoordinate ist eine Funktion $f(X - Startkoordinate)$
2. Y-Zielkoordinate ist eine Funktion $f(X - Zielkoordinate)$
3. Die Ableitung von $f(X - Startkoordinate)$ ist der Achswinkel an der Startkoordinate
4. Die Ableitung von $f(X - Zielkoordinate)$ ist der Achswinkel an der Zielkoordinate

$$y_{start} = a * x_{start}^3 + b * x_{start}^2 + c * x_{start} + d \tag{79}$$
$$y_{ziel} = a * x_{ziel}^3 + b * x_{ziel}^2 + c * x_{ziel} + d \tag{80}$$
$$f'_{(x_{start})} = \theta_{start} \tag{81}$$
$$f'_{(x_{ziel})} = \theta_{ziel} \tag{82}$$

Wichtig ist die Überprüfung ob die Funktion für das Fahrzeug, unter Berücksichtigung des maximalen Lenkwinkels, befahrbar ist. Wenn nicht muss die Zielkoordinate verändert werden. Der Radius den ein Fahrzeug bei konstantem Lenkwinkel fährt ist $r_P(t) = \frac{L}{\tan(\alpha(t))}$ und die Krümmung k ist $k = \frac{1}{r_P(t)}$. Durch den begrenzten Lenkwinkel ergibt sich ein maximale Krümmung der Kurve, die eingehalten werden muss, für ein Polynom ist die Krümmung wie folgt definiert[27]:

$$\frac{1}{r(t)} = k = \frac{y''}{[1+(y')^2]^{3/2}}$$

Der Wenderadius $r_P(t)$ ergibt sich aus der Umformung:

$$r_p(t) \leq r(t) = \left| \frac{(1+{f'_{(x)}}^2)^{3/2}}{f''_{(x)}} \right|$$

Der einzustellende Lenkwinkel zu jeder Koordinate ergibt:

$$\alpha(t) = arctan(\frac{L}{r_P(t)})$$

Die Funktion muss damit den folgenden Lenkwinkel einhalten:

$$\alpha_{max} \leq |\alpha(t)| = |arctan(\frac{L}{r_P(t)})|$$

Etwas aufwendiger ist das zeitlich korrekte setzen des Lenkwinkels und der Geschwindigkeit unter Berücksichtigung der Lenkwinkel- und Geschwindigkeitsbeschleunigung.

Eine Regelung ist beim abfahren von Funktionen notwendig, da über die Berechnung Genauigkeitsfehler entstehen, die Regelung stellt sich im ersten Ansatz als nicht einfach dar.

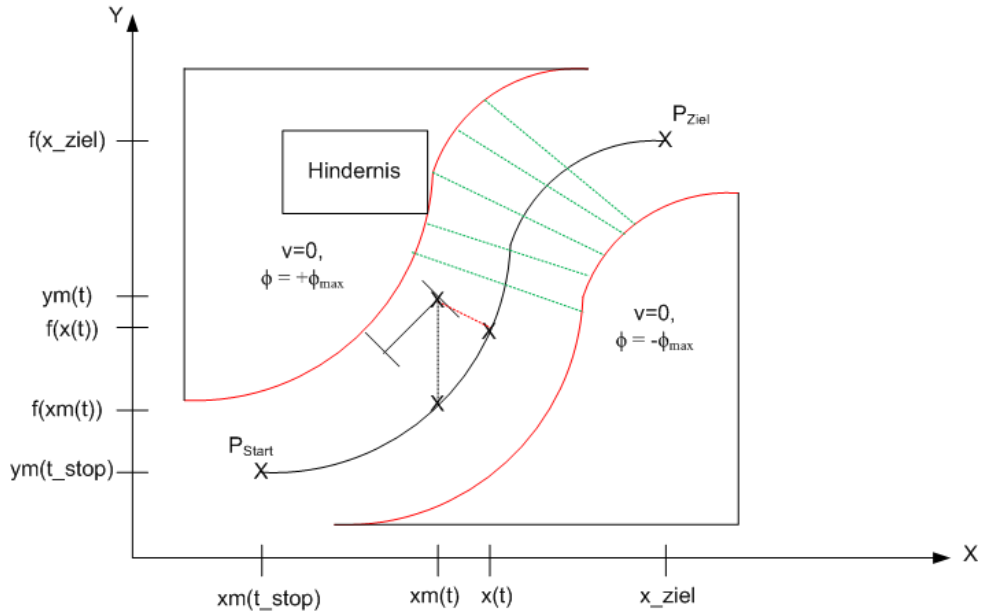


Abb. 59: *Eine kubische Funktion zur Bahnerzeugung, Problem der Regelung bei Regelabweichung*

Eine Abweichung der Soll-Kurve soll hier im Bezug zum Funktionskoordinate mit dem kürzesten Abstand zum Fahrzeug zur Regelung genutzt werden. Der kürzeste Abstand kann nur über eine Suche realisiert werden, indem zum Istwert der X-Koordinate der Fahrzeugposition der Y-Sollwert berechnet wird. Die Suchrichtung zum kürzesten Abstand kann über die benachbarten X-Istwerte und den daraus berechneten Y-Sollwerte berechnet werden. Über den Satz des Pythagoras wird der Abstand berechnet, wird der Abstand zu steigenden X-Istwerten kleiner, so wird zu steigenden X-Werten weiter gesucht. Eine Verbesserung ist eine Kombination von Funktionen und Virtueller Deichsel mit dem Verstärkungsfaktor 1, entlang der Funktion werden Stützpunkte für Deichselbezugspunkte gesucht, die das Fahrzeug immer von Stützstelle zu Stützstelle abfährt.

Ein wesentlicher Nachteil von Funktionen ist, dass die Krümmung der Kurve nicht durchgehend maximal ist und somit nicht der minimale Raum genutzt wird um einzuparken.

C Zusätzliche Simulationsergebnisse

Das folgende Simulationsergebnis zeigt die Symmetrieeigenschaft der Virtuellen Deichsel, bei gleich großer positiver und negativer Geschwindigkeit und identischen Entfernungen zum Ziel.

Symetrieeigenschaften der Virtuellen Deichsel bei Vorwärts- und Rückwärtsfahrt

Vorwärtsfahrt

Deichsel-bezugspunkt	Verstärkungs-Faktor	PStart [xp, yp]	Ziel	xp	yp	Theta [Grad]
[xp,yp]	2,00	[0,0]	[10,10]	10,0000	10,0002	-3,97426
[xp,yp]	4,00	[0,0]	[10,10]	10,0000	10,0000	-0,06338
[xp,yp]	6,00	[0,0]	[10,10]	10,0000	10,0000	-0,00475
[xp,yp]	8,00	[0,0]	[10,10]	10,0000	10,0000	-0,00103
[xp,yp]	10,00	[0,0]	[10,10]	10,0000	10,0000	0,00007
[xp,yp]	12,00	[0,0]	[10,10]	10,0000	10,0000	0,00026
[xp,yp]	14,00	[0,0]	[10,10]	10,0000	10,0000	-0,00022
[xp,yp]	2,00	[0,0]	[2,1]	2,0000	1,0103	-13,14829
[xp,yp]	4,00	[0,0]	[2,1]	2,0000	0,9692	-3,38809
[xp,yp]	6,00	[0,0]	[2,1]	2,0000	0,9474	-28,32848
[xp,yp]	8,00	[0,0]	[2,1]	2,0000	0,9838	-45,52235
[xp,yp]	10,00	[0,0]	[2,1]	2,0000	1,0327	-48,89520
[xp,yp]	12,00	[0,0]	[2,1]	2,0000	1,0714	-48,87907
[xp,yp]	14,00	[0,0]	[2,1]	2,0000	1,1022	-48,87930

Rückwärtsfahrt

Deichsel-bezugspunkt	Verstärkungs-Faktor	PStart [xp, yp]	Ziel	xp	yp	Theta [Grad]
[xp,yp]	2,00	[10,10]	[0,0]	0,0000	-0,0002	-3,97426
[xp,yp]	4,00	[10,10]	[0,0]	0,0000	0,0000	-0,06338
[xp,yp]	6,00	[10,10]	[0,0]	0,0000	0,0000	-0,00475
[xp,yp]	8,00	[10,10]	[0,0]	0,0000	0,0000	-0,00103
[xp,yp]	10,00	[10,10]	[0,0]	0,0000	0,0000	0,00007
[xp,yp]	12,00	[10,10]	[0,0]	0,0000	0,0000	0,00026
[xp,yp]	14,00	[10,10]	[0,0]	0,0000	0,0000	-0,00022
[xp,yp]	2,00	[2,1]	[0,0]	0,0000	-0,0103	-13,14908
[xp,yp]	4,00	[2,1]	[0,0]	0,0000	0,0308	-3,38809
[xp,yp]	6,00	[2,1]	[0,0]	0,0000	0,0526	-28,32848
[xp,yp]	8,00	[2,1]	[0,0]	0,0000	0,0162	-45,52235
[xp,yp]	10,00	[2,1]	[0,0]	0,0000	-0,0327	-48,89520
[xp,yp]	12,00	[2,1]	[0,0]	0,0000	-0,0714	-48,87907
[xp,yp]	14,00	[2,1]	[0,0]	0,0000	-0,1022	-48,87930

Abb. 60: *Symetrieeigenschaft der Virtuellen Deichsel bei Vorwärts- und Rückwärtsfahrt, Achswinkel am Start und am Ziel beträgt 0 Grad*

D Matlab Codes

D.1 Einparkphasen

D.1.1 Testbench Einparkassistent

```
% Skript-File zum Einparkassistenten
clear;
global fahrzeug_breite Weg
global L max_winkel rp rm x_Ziel   y_Start  alpha_soll vm t_max
global Start hindernis_rechts hindernis_links gui
global maxSteps dt verst
global abst_hindernis_zu_linie f_breite toleranz f_laenge_h minAbstand
    r_KA f_laenge_v
global szenario

% *************** Konfigurationsparameter *********************
maxSteps = 100000;                          % maximale Anzahl von
    Abtastschritten
dt = 0.02;                                  % Abtastperiode in s
L = 0.257;                                  % Achsabstand SOC-Fahrzeug [m]
max_winkel_grad = 20;                       % Maximaler Lenkwinkel [Grad]
max_winkel = max_winkel_grad/180*pi;        % Maximaler Lenkwinkel [rad]
rp = L / tan(max_winkel);                   % Wenderadius der Hinterachse
rm = L / sin(max_winkel);                   % Wenderadius der Vorderachse
abst_hindernis_zu_linie = 0.02;
toleranz = 0.05;
minAbstand = 0.05;
f_breite = 0.202;                  % anpassen!!!! [m] (symetrisch)
f_laenge_v = 0.08;                 % Fahrzeuglaenge nach vorne, gemessen
    ab der Vorderachse
f_laenge_h = 0.05;                 % Fahrzeuglaenge nach hinten, gemessen
    ab der Hinterachse
alpha_soll = 0;
r_KI = rp - f_breite/2;
r_KA = sqrt((L+f_laenge_v)^2 + (rp + f_breite/2)^2);
parkluecke_breite = 0.7;                    %0.5 0.6
axisSize = struct('xl',0,'xh',4.5,'yl',0,'yh',2);   % Breite und Hoehe
    der darzustellenden Achsen

fahrspurmitte = struct('y',1);
hindernis_links = struct('x',2.5,'y',fahrspurmitte.y-0.24);
hindernis_rechts = struct('x',hindernis_links.x+parkluecke_breite,'y',
    fahrspurmitte.y-0.24);


% *************** Datenstrukturen *********************
finalC = struct('x',[],'y',[]);
Start = struct('x',[],'y',[],'theta',[]);
Start1 = struct('x',[],'y',[],'theta',[]);
Start2 = struct('x',[],'y',[],'theta',[]);
Ziel = struct('x',[],'y',[],'theta',[]);
B1 = struct('x',[],'y',[],'theta',[]);
B2 = struct('x',[],'y',[],'theta',[]);
B = struct('x',[],'y',[],'theta',[]);
omega_1 = struct('x',[],'y',[]);
omega_2 = struct('x',[],'y',[]);
omega_21 = struct('x',[],'y',[]);
omega_22 = struct('x',[],'y',[]);
next_position = struct('x',-1,'y',-1,'theta',-1,'weg',-1);
```

```
49 Weg = struct('k1',[],'k2',[]);
50
51 % ************** GUI ********************
52 gui = guidata(GUI);                    % Singleton GUI
53 axes(gui.axes1);
54 cla(gui.axes1,'reset');                % Plot Inhalt Loeschen
55 set(gui.axes1,'XMinorTick','on');
56 axis([axisSize.xl axisSize.xh axisSize.yl axisSize.yh]);     %skaliert
       die Achsen
57 grid on;
58 plot(0,0,'r');hold on;      %Hinterachse
59 plot(0, 0,'b'); hold on;     %Vorderachse
60 legend('Hinterachse','Vorderachse','Location','SouthOutside'); hold on;
61
62 % Strasse plotten
63 plot([axisSize.xl, axisSize.xh], [fahrspurmitte.y+0.62, fahrspurmitte.y
       +0.62],'k-'); hold on;
64 plot([axisSize.xl, axisSize.xh], [fahrspurmitte.y+0.64, fahrspurmitte.y
       +0.64],'k-'); hold on;
65 plot([axisSize.xl, axisSize.xh], [fahrspurmitte.y+0.2, fahrspurmitte.y
       +0.2],'k--'); hold on;
66 plot([axisSize.xl, axisSize.xh], [fahrspurmitte.y+0.22, fahrspurmitte.y
       +0.22],'k--'); hold on;
67 plot([axisSize.xl, axisSize.xh], [fahrspurmitte.y-0.2, fahrspurmitte.y
       -0.2],'k-'); hold on;
68 plot([axisSize.xl, axisSize.xh], [fahrspurmitte.y-0.22, fahrspurmitte.y
       -0.22],'k-'); hold on;
69
70 % Hindernisse plotten
71 kreuz_breite_halbe = 0.01;
72 fill([hindernis_links.x hindernis_links.x-(0.3) hindernis_links.x-(0.3)
       hindernis_links.x],[hindernis_links.y hindernis_links.y
       hindernis_links.y-0.3 hindernis_links.y-0.3],'b'); hold on;
73 fill([hindernis_rechts.x hindernis_rechts.x+(0.3) hindernis_rechts.x
       +(0.3) hindernis_rechts.x],[hindernis_rechts.y hindernis_rechts.y
       hindernis_rechts.y-0.3 hindernis_rechts.y-0.3],'b'); hold on;
74
75 % ************* GUI ************************
76
77
78 % GUI-Felder auslesen und in struktur speichern, Format: (X,Y,Winkel)
79 % TODO
80 %Start = struct('x',str2double(get(gui.edit13,'String')),'y',str2double(
       get(gui.edit14,'String')),'theta',str2double(get(gui.edit15,'String')
       ));
81 %linke_Hindernis = struct('x',str2double(get(gui.edit1,'String')),'y',
       str2double(get(gui.edit8,'String')),'Winkel',str2double(get(gui.edit9
       ,'String')));
82 %rechte_Hindernis =
83 %struct('x',str2double(get(gui.edit10,'String')),'y',str2double(get(gui.
       edit11,'String')),'Winkel',str2double(get(gui.edit12,'String')))
84
85 axis([axisSize.xl axisSize.xh axisSize.yl axisSize.yh]);%skaliert die
       Achsen
86
87 t_max = 0;
88 t_new = 0;
89 fhzg_state_new =0;
90 eingeparkt = 0;
```

```

finalC.y = hindernis_links.y + abst_hindernis_zu_linie - f_breite/2 - toleranz; % Endposition in der Parklücke berechnen
finalC.x = hindernis_links.x + f_laenge_h + minAbstand;
% Berechnung der Startposition für den ersten Einparkschritt
[Start_positionieren B Ziel] = phase_ersterEinparkschritt(hindernis_rechts, fahrspurmitte, finalC);

Start.x = 0;
Start.y = min(Start_positionieren.y, fahrspurmitte.y+0.05);
fahrspurmitte.y = Start.y;          % neue virtuelle Fahrspurmitte für die Spurführung ist die Y-Startkoordinate
Start.theta = 0*pi/180;

%%%%%%%%%%%%%%%%%%%%%%%%%% AUSWAHL VON EINZELNEN SIMULATIONEN DER EINPARKPHASEN
%%%%%%%%%%%%%%%%%%%%%%%%%% HIER AUSWÄHLEN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
first_call = 0;             % fallss einzelne Einparkphasen getestet werden sollen
sim_phase = 4;              % zur Simulation einzelner Einparkphasen (ohne vorherige Phasen zu simulieren / plotten)
nterSchritt_szenario = 1;   % nur bei sim_phase 3 setzbar
ersterSchrittSzenario = 1;
switch sim_phase
    case 0,                 %Spurführung
        Start.x = 0;
        Start.y = 0.7;
        Start.theta = 0.2;
        phase = 0;
    case 1,                 %Positionierung
        phase = 1;
        Start.x = 1.6000;
        Start.y = 1.1574;
        Start.theta = 0.0;
        fhzg_init = [Start.theta;Start.x + cos(Start.theta)*L;Start.y + sin(Start.theta) * L;Start.x;Start.y;-0.3491;0];

        tspan = [0:dt:maxSteps*dt];
    case 2,                 %Erster Einparkschritt

        if(ersterSchrittSzenario==1)    %Szenario 2
           % finalC.y = finalC.y + 0.2;
           % hindernis_rechts.x = hindernis_rechts.x + 0.1;
            fahrspurmitte.y = fahrspurmitte.y +0.2;
           [Start B Ziel] = phase_ersterEinparkschritt(hindernis_rechts, fahrspurmitte, finalC);
            phase = 2;
            fhzg_init = [0;Start.x+L;Start.y;Start.x;Start.y;0;0];
            tspan = [0:dt:maxSteps*dt];
        else
            [Start B Ziel] = phase_ersterEinparkschritt(hindernis_rechts, fahrspurmitte, finalC);

        phase = 2;
        fhzg_init = [0;Start.x+L;Start.y;Start.x;Start.y;0;0];
        tspan = [0:dt:maxSteps*dt];
        end
    case 3,                 %NterEinparkschritt
```

```
140         phase = 3;
141         first_call = 1;
142         tspan = [0:dt:maxSteps*dt];
143         switch nterSchritt_szenario
144             case 1,         % szenario 1 ==> Vorwärts, 2 Kurven
145                 Start.x = 2.6000;
146                 Start.y = 0.7574;
147                 Start.theta = 0.2;
148                 fhzg_init = [Start.theta;Start.x + cos(Start.theta)*L;
                        Start.y + sin(Start.theta) * L;Start.x;Start.y;0;0];
149             case 2,         % szenario 2 ==> Vorwärts, 1 Linkskurve
150                 Start.x = 2.6000;
151                 Start.y = 0.7574;
152                 Start.theta = -1;
153                 fhzg_init = [Start.theta;Start.x + cos(Start.theta)*L;
                        Start.y + sin(Start.theta) * L;Start.x;Start.y
                        ;-0.3491;0];
154
155             case 3,
156                 Start.x = 2.6000;
157                 Start.y = 0.7574;
158                 Start.theta = 1;
159                 fhzg_init = [Start.theta;Start.x + cos(Start.theta)*L;
                        Start.y + sin(Start.theta) * L;Start.x;Start.y
                        ;-0.3491;0];
160             case 4,     % szenario 4 ==> Rückwärts, 2 Kurven
161                 Start.x = 2.8130;
162                 Start.y = 0.7413;
163                 Start.theta = 0.2;
164                 fhzg_init = [Start.theta;Start.x + cos(Start.theta)*L;
                        Start.y + sin(Start.theta) * L;Start.x;Start.y
                        ;-0.3491;0];
165             case 5,     % szenario 5 ==> Rückwärts, 1 Rechtskurve
166                 Start.x = 2.8130;
167                 Start.y = 0.7413;
168                 Start.theta = 1;
169                 fhzg_init = [Start.theta;Start.x + cos(Start.theta)*L;
                        Start.y + sin(Start.theta) * L;Start.x;Start.y
                        ;-0.3491;0];
170             case 6,     % szenario 6 ==> Rückwärts, 1 Linkskurve
171                 Start.x = 2.8130;
172                 Start.y = 0.7413;
173                 Start.theta = -1;
174                 fhzg_init = [Start.theta;Start.x + cos(Start.theta)*L;
                        Start.y + sin(Start.theta) * L;Start.x;Start.y
                        ;-0.3491;0];
175         end
176     case 4,                 % Fahrzeug ausrichten
177         phase = 4;
178         first_call = 1;
179         Start.x = finalC.x;
180         Start.y = finalC.y;
181         Start.theta = 30*pi / 180;
182         fhzg_init = [Start.theta;Start.x + cos(Start.theta)*L;Start.y +
              sin(Start.theta) * L;Start.x;Start.y;-0.3491;0];
183         tspan = [0:dt:maxSteps*dt];
184 end
185
186 kreuz_breite_halbe = 0.01;
```

```
187 plot([Start.x-kreuz_breite_halbe, Start.x+kreuz_breite_halbe], [Start.y-
    kreuz_breite_halbe, Start.y+kreuz_breite_halbe],'r'); hold on;
188 plot([Start.x-kreuz_breite_halbe, Start.x+kreuz_breite_halbe], [Start.y+
    kreuz_breite_halbe, Start.y-kreuz_breite_halbe],'r'); hold on;
189
190 while eingeparkt == 0
191     t_old = t_new;
192     fhzg_state_old = fhzg_state_new;
193     switch phase
194         case 0
195             set(gui.text46,'String','Spurführung');
196             [Ziel verst] = phase_Spurfuehrung(Start, fahrspurmitte);   %
                 Berechnen der nächsten Koordinate und dem
                Verstärkungsfaktor
197
198             vm = 0.1;                                                  %
                 Vorwärts fahren
199             next_position.x = Ziel.x;        % Event setzen
200             next_position.y = -1;            % Event ausschalten
201             next_position.weg = -1;
202             t_max = -1;
203
204             if(t_new ==0)                                              %
                 das erste Mal das Simulationsmodell initialisieren
205                 tspan = [0:dt:maxSteps*dt];
206                 fhzg_init = [Start.theta;Start.x+L*cos(Start.theta);
                    Start.y+L*sin(Start.theta);Start.x;Start.y;0;0];
                            % [theta, xm, ym, xp, yp, alpha_ist, s]
207             else
208                 tspan = [te(length(te)):dt:maxSteps*dt];
                    % die neue Simulation mit den Daten der letzten
                    Simulation initialisieren
209                 last_index = length(fhzg_state(:,1));
210                 fhzg_init = [fhzg_state(last_index,1);fhzg_state(
                    last_index,2);fhzg_state(last_index,3);fhzg_state(
                    last_index,4);fhzg_state(last_index,5);fhzg_state(
                    last_index,6);fhzg_state(last_index,7)];           %
                    Startzustandsvektor [theta, xm, ym, xp, yp, alpha_ist
                    , s]
211             end
212             options = odeset ('OutputFcn',@(t,fhzg_state,flag)
                    outputFunction_dchsl(t,fhzg_state,flag, Ziel, verst,
                    max_winkel, gui),'MaxStep', dt , 'Events' , @(t,
                    fhzg_state)Event_dchsl( t, fhzg_state, t_max,
                    next_position),'Refine', 1) ;
213             [t fhzg_state te ye ie] = ode45(@(t, fhzg_state)fhzg_modell(
                    t, fhzg_state, L), tspan, fhzg_init, options);
214
215             theta = fhzg_state(length(fhzg_state(:,1)),1);
216                     xp = fhzg_state(length(fhzg_state(:,4)),4);
217                     yp = fhzg_state(length(fhzg_state(:,5)),5);
218             alpha_ist = fhzg_state(length(fhzg_state(:,6)),6);
219             s_ist = fhzg_state(length(fhzg_state(:,7)),7);
220             setGuiText(vm, s_ist, xp, yp, theta, alpha_soll, alpha_ist,
                    gui);   % Zwischenwerte plotten
221
222             Start.x = xp;
                                                                           %
```

```
            letzte Koordinaten sind die neuen Koordinaten der
            nächsten Simulation
223             Start.y = yp;
224             Start.theta = theta;
225
226             if(Start.x >= hindernis_rechts.x) % irgendwo hinter dem
                    rechten Hindernis anhalten
227                 phase = phase + 1;
228                 vm = 0;
229             end
230
231         case 1
232
233             set(gui.text46,'String','Positionierung');
234             finalC.y = hindernis_links.y + abst_hindernis_zu_linie -
                    f_breite/2 - toleranz; % Endposition berechnen
235             finalC.x = hindernis_links.x + f_laenge_h + minAbstand;
236             [Start_positionieren B Ziel] = phase_ersterEinparkschritt(
                    hindernis_rechts, fahrspurmitte, finalC); % Berechnung
                    der Startposition
237             [Ziel verst] = phase_Positionierung(Start,
                    Start_positionieren); % Wegplanung mit dem Ziel
                    Startposition des ersten Einparkschrittes
238
239             if(Start.x < Ziel.x)
240                 vm = 0.1;
241             else
242                 vm = -0.1;
243             end
244
245             next_position.x = Ziel.x;       % Event setzen
246             next_position.y = -1;           % Event ausschalten
247             next_position.weg = -1;
248             t_max = -1;
249             if(sim_phase ~= 1)
250                 tspan = [te(length(te)):dt:maxSteps*dt];
251                 last_index = length(fhzg_state(:,1));
252                 fhzg_init = [fhzg_state(last_index,1);fhzg_state(
                        last_index,2);fhzg_state(last_index,3);fhzg_state(
                        last_index,4);fhzg_state(last_index,5);fhzg_state(
                        last_index,6);fhzg_state(last_index,7)];          %
                        Startzustandsvektor [theta, xm, ym, xp, yp, alpha_ist
                        , s]
253             end
254             if(sim_phase == 1)  %Simulationsauswahl
255                 sim_phase = 0;
256             end
257             options = odeset ('OutputFcn',@(t,fhzg_state,flag)
                    outputFunction_dchsl(t,fhzg_state,flag, Ziel, verst,
                    max_winkel, gui),'MaxStep', dt , 'Events' , @(t,
                    fhzg_state)Event_dchsl( t, fhzg_state, t_max,
                    next_position),'Refine', 1) ;
258             [t fhzg_state te ye ie] = ode45(@(t, fhzg_state)fhzg_modell(
                    t, fhzg_state, L), tspan, fhzg_init, options);
259
260             theta = fhzg_state(length(fhzg_state(:,1)),1);
261                         xp = fhzg_state(length(fhzg_state(:,4)),4);
262                         yp = fhzg_state(length(fhzg_state(:,5)),5);
263             alpha_ist = fhzg_state(length(fhzg_state(:,6)),6);
```

```
264             s_ist = fhzg_state(length(fhzg_state(:,7)),7);
265             setGuiText(vm, s_ist, xp, yp, theta, alpha_soll, alpha_ist,
                    gui); % GUI-Daten plotten
266
267             delta_x = abs(xp - Start_positionieren.x);
268             delta_y = abs(yp - Start_positionieren.y);
269             delta_theta = abs(theta - Start_positionieren.theta)*180/pi;
270             if(delta_x <= 0.05 && delta_y <= 0.05 && delta_theta <= 5 )
                    % Positionieren beenden wenn Konfiguration innerhalb
                    einer Toleranz liegt
271                 phase = phase + 1;
272                 vm = 0;
273             end
274             Start.x = xp;
275             Start.y = yp;
276             Start.theta = theta;
277
278         case 2
279
280             set(gui.text46,'String','Erster Einparkschritt');
281             [Start B Ziel] = phase_ersterEinparkschritt(hindernis_rechts
                    , fahrspurmitte, finalC);
282
283             phase_step = 0;
284                         while(phase_step <= 3)
285                 switch phase_step       % 4 Schritte für den ersten
                        Einparkschritt
286                     case 0  % Anhalten und umlenken
287                         set(gui.text48,'String','anhalten und umlenken'
                                );
288                         alpha_soll = max_winkel;
289                         if(sim_phase ~= 2)
290                             t_max = te(length(te)) + 0.29475;  % Events
                                    setzen
291                         else
292                             t_max = 0.29475;
293                         end
294                         next_position.x = -1;
295                         next_position.y = -1;
296                         next_position.weg = -1;
297                     case 1  % 1. Kurve
298                         set(gui.text48,'String','Rechtskurve');
299                         alpha_soll = max_winkel;
300                         next_position.x = B.x; % Events setzen
301                         next_position.y = B.y;
302                         next_position.theta = B.theta;
303                         next_position.weg = fhzg_state(length(
                                fhzg_state(:,7)),7) + Weg.k1;
304                         t_max = -1;
305                     case 2  % Anhalten und umlenken
306                         set(gui.text48,'String','anhalten und umlenken'
                                );
307                         alpha_soll = -max_winkel;
308                         t_max = te(length(te)) + 0.3;  % Events setzen
309                         next_position.x = -1;
310                         next_position.y = -1;
311                         next_position.weg = -1;
312                      case 3  %2. Kurve
313                         set(gui.text48,'String','Linkskurve');
```

```
314                         alpha_soll = -max_winkel;
315                         next_position.x = Ziel.x;  % Events setzen
316                         next_position.y = Ziel.y;
317                         next_position.weg = fhzg_state(length(
                                fhzg_state(:,7)),7) + Weg.k2;
318                         t_max = -1;
319                     end
320
321                     if(phase_step == 0 || phase_step == 2)      % Anhalten
                            und umlenken
322                         vm = 0;
323                     else
324                         vm = -0.1;
325                     end
326
327                     if ~(sim_phase == 2 && phase_step == 0)
328                         tspan = [te(length(te)):dt:maxSteps*dt];
329                         last_index = length(fhzg_state(:,1));
330                         fhzg_init = [fhzg_state(last_index,1);fhzg_state(
                                last_index,2);fhzg_state(last_index,3);fhzg_state
                                (last_index,4);fhzg_state(last_index,5);
                                fhzg_state(last_index,6);fhzg_state(last_index,7)
                                ];
331                     end
332
333                     options = odeset ('OutputFcn',@(t,fhzg_state,flag)
                            outputFunction(t,fhzg_state,flag, vm, alpha_soll, gui
                            ),'MaxStep', dt , 'Events' , @(t, fhzg_state)
                            Event_kreistechnik( t, fhzg_state, t_max,
                            next_position),'Refine', 1) ;
334                     [t fhzg_state te ye ie] = ode45(@(t, fhzg_state)
                            fhzg_modell(t, fhzg_state, L), tspan, fhzg_init,
                            options);
335
336                      phase_step = phase_step +1;
337                      if(phase_step == 4)
338                          alpha_soll = 0;
339                          phase = phase + 1;
340                          yp = fhzg_state(length(fhzg_state(:,5)),5);
341                          if(yp <= finalC.y)
342                              eingeparkt = 1;      % Ende
343                          end
344                      end
345                 end
346                 theta = fhzg_state(length(fhzg_state(:,1)),1);
347                 xp = fhzg_state(length(fhzg_state(:,4)),4);
348                 yp = fhzg_state(length(fhzg_state(:,5)),5);
349                 alpha_ist = fhzg_state(length(fhzg_state(:,6)),6);
350                 s_ist = fhzg_state(length(fhzg_state(:,7)),7);
351                 setGuiText(vm, s_ist, xp, yp, theta, alpha_soll, alpha_ist,
                        gui);% GUI-Daten plotten
352
353             case 3
354                 set(gui.text46,'String',' Nter Einparkschritt');
355
356                  if( first_call == 0 )
357                      Start.x = fhzg_state(length(fhzg_state(:,4)),4);
358                      Start.y = fhzg_state(length(fhzg_state(:,5)),5);
359                      Start.theta = fhzg_state(length(fhzg_state(:,1)),1);
```

```
360             end
361             [B Ziel] = phase_NterEinparkschritt(Start, hindernis_rechts
                    , hindernis_links); % nächsten Einparkschritt berechnen
362             phase_ende = 0;
363
364             if(B.x == 0)
365                 phase_step = 2;
366             else
367                 phase_step = 0;
368             end
369             while(phase_step <= 3)
370
371                 if(phase_step == 0 || phase_step == 2)
372                    vm = 0;
373                else
374                     if(Start.x < Ziel.x)
375                         vm = 0.1;
376                     else
377                         vm = -0.1;
378                     end
379                end
380
381                 switch phase_step
382                     case 0  % Anhalten und umlenken
383                         set(gui.text48,'String','anhalten und umlenken')
                                ;
384                         alpha_soll = max_winkel;
385                         if(first_call == 0)
386                              t_max = te(length(te)) + 0.29475;  % Events
                                     setzen
387                         else
388                             t_max = 0.29475;
389                         end
390                         next_position.x = -1;
391                         next_position.y = -1;
392                         next_position.weg = -1;
393                     case 1  % 1. Kurve, Rechtskurve
394                         set(gui.text48,'String','Rechtskurve');
395                         s_ist = fhzg_state(length(fhzg_state(:,7)),7);
396                         alpha_soll = max_winkel;
397                         next_position.x = B.x;
398                         next_position.y = B.y;
399                         next_position.weg = Weg.k1+s_ist;
400                         t_max = -1;
401                     case 2  % Anhalten und umlenken
402                         set(gui.text48,'String','anhalten und umlenken')
                                ;
403
404                         if(first_call == 0)
405                             t_max = te(length(te)) + 0.29475;
406                         else
407                             t_max = 0.29475;
408                         end
409                         if(szenario == 2 || szenario == 5 || szenario ==
                                1 || szenario == 4)
410                             alpha_soll = -max_winkel;
411                         else
412                             alpha_soll = max_winkel;
413                         end
```

```
414                         next_position.x = -1;
415                         next_position.y = -1;
416                         next_position.weg = -1;
417                     case 3  %2. Kurve, Linkskurve
418                         set(gui.text48,'String','Linkskurve');
419                         s_ist = fhzg_state(length(fhzg_state(:,7)),7);
420                         next_position.weg = Weg.k2+s_ist;
421                         next_position.x = Ziel.x;
422                         next_position.y = Ziel.y;
423                         t_max = -1;
424                 end
425
426
427                  if (first_call == 0)
428                     tspan = [te(length(te)):dt:maxSteps*dt];
429                     last_index = length(fhzg_state(:,1));
430                     fhzg_init = [fhzg_state(last_index,1);fhzg_state(
                            last_index,2);fhzg_state(last_index,3);fhzg_state
                            (last_index,4);fhzg_state(last_index,5);
                            fhzg_state(last_index,6);fhzg_state(last_index,7)
                            ];
431                  else
432                      first_call = 0;
433                  end
434                 options = odeset ('OutputFcn',@(t,fhzg_state,flag)
                        outputFunction(t,fhzg_state,flag, vm, alpha_soll, gui
                        ),'MaxStep', dt , 'Events' , @(t, fhzg_state)
                        Event_kreistechnik( t, fhzg_state, t_max,
                        next_position),'Refine', 1) ;
435                 [t fhzg_state te ye ie] = ode45(@(t, fhzg_state)
                        fhzg_modell(t, fhzg_state, L), tspan, fhzg_init,
                        options);
436
437                 theta = fhzg_state(length(fhzg_state(:,1)),1);
438                 phase_step = phase_step +1;
439                 if(phase_step == 4)
440                      alpha_soll = 0;
441                      yp = fhzg_state(length(fhzg_state(:,5)),5);
442                      if(yp <= finalC.y)
443                          phase = phase + 1;      % Ziel-Tiefe in der
                                Parklücke erreicht
444                          theta = fhzg_state(length(fhzg_state(:,1)),1);
445                          if(abs(theta*180/pi) <= 5)
446                              eingeparkt = 1;     % Zielkonfiguration
                                    erreicht
447                          end
448                     end
449                 end
450             end
451
452             theta = fhzg_state(length(fhzg_state(:,1)),1);
453             xp = fhzg_state(length(fhzg_state(:,4)),4);
454             yp = fhzg_state(length(fhzg_state(:,5)),5);
455             alpha_ist = fhzg_state(length(fhzg_state(:,6)),6);
456             s_ist = fhzg_state(length(fhzg_state(:,7)),7);
457             setGuiText(vm, s_ist, xp, yp, theta, alpha_soll, alpha_ist,
                    gui);  % GUI-Daten plotten
458
459         case 4
```

```
460             set(gui.text46,'String','Fahrzeug␣ausrichten');
461             if(first_call == 0)
462                 Start.x = fhzg_state(length(fhzg_state(:,4)),4);
463                 Start.y = fhzg_state(length(fhzg_state(:,5)),5);
464                 Start.theta = fhzg_state(length(fhzg_state(:,1)),1);
465                 tspan = [te(length(te)):dt:maxSteps*dt];
466                 last_index = length(fhzg_state(:,1));
467                 fhzg_init = [fhzg_state(last_index,1);fhzg_state(
                        last_index,2);fhzg_state(last_index,3);fhzg_state(
                        last_index,4);fhzg_state(last_index,5);fhzg_state(
                        last_index,6);fhzg_state(last_index,7)];          % [
                        theta, xm, ym, xp, yp, alpha_ist, s]
468             else
469                 first_call = 0;
470             end
471             [Ziel verst] = phase_ausrichten(Start);          % Parameter
                    zum Ausrichten berechnen
472
473             if(Start.x < Ziel.x)
474                 vm = 0.1;
475             else
476                 vm = -0.1;
477             end
478             next_position.x = Ziel.x;   % Events
479             next_position.weg = -1;
480             next_position.y = -1;
481             t_max = -1;
482
483
484             options = odeset ('OutputFcn',@(t,fhzg_state,flag)
                    outputFunction_dchsl(t,fhzg_state,flag, Ziel, verst,
                    max_winkel, gui),'MaxStep', dt , 'Events' , @(t,
                    fhzg_state)Event_dchsl( t, fhzg_state, t_max,
                    next_position),'Refine', 1) ;
485             [t fhzg_state te ye ie] = ode45(@(t, fhzg_state)fhzg_modell(
                    t, fhzg_state, L), tspan, fhzg_init, options);
486
487             theta = fhzg_state(length(fhzg_state(:,1)),1);
488             xp = fhzg_state(length(fhzg_state(:,4)),4);
489             yp = fhzg_state(length(fhzg_state(:,5)),5);
490             alpha_ist = fhzg_state(length(fhzg_state(:,6)),6);
491             s_ist = fhzg_state(length(fhzg_state(:,7)),7);
492             setGuiText(vm, s_ist, xp, yp, theta, alpha_soll, alpha_ist,
                    gui);       % GUI-Daten plotten
493
494             delta_theta = abs(theta)*180/pi;
495             delta_y = abs(yp - finalC.y);
496             if(delta_theta <= 5 && delta_y <=0.05) % mindestens 1/2
                    Meter hinter dem rechten Hindernis anhalten
497                 eingeparkt = 1;
498                 vm = 0;
499             end
500             if(delta_y > 0.05)
501                 phase = phase -1;
502                 first_call = 0; %Simulationsparameter der einzelnen
                        Einparkphasen löschen
503             end
504
```

```
505             Start.x = xp;       % letzte Fahrzeugkonfiguration ist die
                    Neue im nächsten Schritt
506             Start.y = yp;
507             Start.theta = theta;
508     end
509
510     % Aufzeichnung aller Fahrzeugparameter
511     if(t_old == 0)
512         t_new = [t];
513         fhzg_state_new = [fhzg_state];
514     else
515         t_new = [t_old;t];
516         fhzg_state_new = [fhzg_state_old; fhzg_state];
517     end
518
519 end
520
521
522 % theta = fhzg_state_new(:,1);
523 % xp = fhzg_state_new(:,4);
524 % yp = fhzg_state_new(:,5);
525 % alpha = fhzg_state_new(:,6);
526 % alpha_grad = alpha .* 180 ./ pi;
527 %
528 % %Die Fahrzeugbegrenzung
529 % hinterachse_links_x = -sin(theta).*(-fahrzeug_breite/2)+hinterachse_x;
530 % hinterachse_rechts_x = -sin(theta).*(fahrzeug_breite/2)+hinterachse_x;
531 % hinterachse_links_y = cos(theta).*(-fahrzeug_breite/2)+hinterachse_y;
532 % hinterachse_rechts_y = cos(theta).*(fahrzeug_breite/2)+hinterachse_y;
533 % vorderachse_links_x = -sin(theta).*(-fahrzeug_breite/2)+vorderachse_x;
534 % vorderachse_rechts_x = -sin(theta).*(fahrzeug_breite/2)+vorderachse_x;
535 % vorderachse_links_y = cos(theta).*(-fahrzeug_breite/2)+vorderachse_y;
536 % vorderachse_rechts_y = cos(theta).*(fahrzeug_breite/2)+vorderachse_y;
```

D.1.2 Eventsteuerung der Virtuellen Deichsel

```
1
2 function [value,isterminal,direction] = Event_dchsl( t, fhzg_state,
      t_max, next_position)
3
4 xp = fhzg_state(4,1);
5 yp = fhzg_state(5,1);
6 s_ist = fhzg_state(7,1);
7
8 % *** value => der Wert, der Null werden soll ***
9 % *** isterminal => Simulationsreaktion auf das Finden einer Nullstelle
      ***
10 % 0 = Simulation fortführen
11 % 1 = Abbruch der Simulation
12 % *** direction => Richtung der Nullstellenüberquerung ***
13 % -1= vom Positivem ins Negative
14 % 0 = alle Nullstellen jeder Richtung
15 % 1 = vom Negativen ins Positive
16
17
18 value(1)      = next_position.x - xp;
19 isterminal(1) =  1;
20 direction(1)  = 0;              % Überschreitung in jede Richtung
21
22 value(2)      = next_position.y - yp;
```

```
23 isterminal(2) =  1;
24 direction(2)  = 0;              % Überschreitung in jede Richtung
```

D.1.3 Eventsteuerung der Kreistechnik

```
1  function [value,isterminal,direction] = Event_kreistechnik( t,
       fhzg_state, t_max, next_position)
2
3 xp = fhzg_state(4,1);
4 yp = fhzg_state(5,1);
5 s_ist = fhzg_state(7,1);
6
7 % *** value => der Wert, der Null werden soll ***
8 % *** isterminal => Simulationsreaktion auf das Finden einer Nullstelle
      ***
9 % 0 = Simulation fortführen
10 % 1 = Abbruch der Simulation
11 % *** direction => Richtung der Nullstellenüberquerung ***
12 % -1= vom Positivem ins Negative
13 % 0 = alle Nullstellen jeder Richtung
14 % 1 = vom Negativen ins Positive
15
16 value(1) = next_position.weg - s_ist;
17 isterminal(1) = 1;
18 direction(1) = 0;
19
20 value(2) = t_max - t;             % if (t >= t_max)
21 isterminal(2) = 1;
22 direction(2) = -1;
```

D.1.4 Einparkphase - Spurführung

```
1  function [Ziel verst] = phase_Spurfuehrung(Start, fahrspurmitte)
2
3    Ziel.theta = 0;
4
5    if((Start.y > fahrspurmitte.y && Start.theta > 0) || (Start.y <
         fahrspurmitte.y && Start.theta < 0))
6        direction = -1;
7    else
8        direction = 1;
9    end;
10   dy = abs(Start.y - fahrspurmitte.y);
11   dtheta = abs(Start.theta - 0) * 180 / pi * direction;
12
13
14   Ziel.x = Start.x + 1;        %default
15   verst = 1;                   %default
16   Ziel.y = fahrspurmitte.y;    %default
17   if( dy >= 0 && dy <= 5 )     % Suchräume: F, G, M, P
18
19       if(  dtheta > -10 && dtheta <= -5  )%M
20           Ziel.x = Start.x + 1;
21           verst = 5;
22       elseif (dtheta > -5 && dtheta < 0) %F
23           Ziel.x = Start.x + 0.5;
24           verst = 3;
25       elseif (dtheta > 0 && dtheta < 5) %G
26           Ziel.x = Start.x + 0.5;
27           verst = 2;
```

```
28          elseif (dtheta >= 5 && dtheta < 10) %P
29              Ziel.x = Start.x + 1;
30              verst = 2;
31          end
32
33      else                    % Suchräume: I, J, K, L
34          if(  dtheta > -10 && dtheta <= -5  )%I
35              Ziel.x = Start.x + 1;
36              verst = 5;
37          elseif (dtheta > -5 && dtheta < 0) %J
38              Ziel.x = Start.x + 1;
39              verst = 5;
40          elseif (dtheta > 0 && dtheta < 5) %K
41              Ziel.x = Start.x + 1;
42              verst = 4;
43          elseif (dtheta >= 5 && dtheta < 10) %L
44              Ziel.x = Start.x + 1;
45              verst = 4;
46          end
47      end
```

D.1.5 Einparkphase - Positionierung

```
1  function [Ziel verst] = phase_Positionierung(Start, Start_positionieren)
2
3
4  Dx = abs(Start.x - Start_positionieren.x);
5  Dy = abs(Start.y - Start_positionieren.y);
6
7          if((Start.x > Start_positionieren.x && Start.y >
               Start_positionieren.y && Start.theta < 0) ||
8                          (Start.x > Start_positionieren.x && Start.y <
                               Start_positionieren.y && Start.theta > 0) ||
9                          (Start.x < Start_positionieren.x && Start.y >
                               Start_positionieren.y && Start.theta > 0) ||
10                         (Start.x < Start_positionieren.x && Start.y <
                               Start_positionieren.y && Start.theta < 0))
11             direction = -1;
12         else
13             direction = 1;
14         end;
15
16 Dtheta = abs(Start.theta*180/pi) * direction;
17
18 % H = (Dx >=30 && Dx <50) && (Dy > 0 && Dy <= 5) && (Dtheta >=5 &&
       Dtheta < 10)
19 % P = (Dx >=50) && (Dy > 0 && Dy <= 5) && (Dtheta >=5 && Dtheta < 10)
20 % G = (Dx >=30 && Dx <50) && (Dy > 0 && Dy <= 5) && (Dtheta >0 && Dtheta
       < 5)
21 % M = (Dx >=50) && (Dy > 0 && Dy <= 5) && (Dtheta >-10 && Dtheta <= -5)
22 % N = (Dx >=50) && (Dy > 0 && Dy <= 5) && (Dtheta >-5 && Dtheta < 0)
23 % O = (Dx >=50) && (Dy > 0 && Dy <= 5) && (Dtheta >0 && Dtheta < 5)
24 % Q = (Dx >=50) && (Dy > 5 && Dy <= 10) && (Dtheta >-10 && Dtheta <= -5)
25 % R = (Dx >=50) && (Dy > 5 && Dy <= 10) && (Dtheta >-5 && Dtheta < 0)
26 % S = (Dx >=50) && (Dy > 5 && Dy <= 10) && (Dtheta >0 && Dtheta < 5)
27 % T = (Dx >=50) && (Dy > 5 && Dy <= 10) && (Dtheta >=5 && Dtheta < 10)
28 if((Dx >=0.30 && Dx <0.50) && (Dy > 0 && Dy <= 0.5) && (Dtheta >=5 &&
       Dtheta < 10)|| (Dx >=0.50) && (Dy > 0 && Dy <= 0.5) && (Dtheta >=5 &&
        Dtheta < 10))
29         verst = 1;
```

```
30      Ziel.x = Start_positionieren.x;
31 elseif( ((Dx >=0.50) && (Dy > 0 && Dy <= 0.5) && (Dtheta >-10 && Dtheta
       <= -5)) || ((Dx >=0.50) && (Dy > 0 && Dy <= 0.5) && (Dtheta >-5 &&
       Dtheta < 0)) || ((Dx >=0.50) && (Dy > 0 && Dy <= 0.5) && (Dtheta >0
       && Dtheta < 5)) || ((Dx >=0.50) && (Dy > 0.5 && Dy <= 0.10) && (
       Dtheta >-10 && Dtheta <= -5)) || ((Dx >=0.50) && (Dy > 0.5 && Dy <=
       0.10) && (Dtheta >-5 && Dtheta < 0)) || ((Dx >=0.50) && (Dy > 0.5 &&
       Dy <= 0.10) && (Dtheta >0 && Dtheta < 5)) || ((Dx >=0.50) && (Dy >
       0.5 && Dy <= 0.10) && (Dtheta >=5 && Dtheta < 10)) )
32      verst = 2;
33      Ziel.x = Start_positionieren.x;
34 elseif((Dx >=0.30 && Dx <0.50) && (Dy > 0 && Dy <= 0.5) && (Dtheta >0 &&
       Dtheta < 0.5))
35      verst = 3;
36      Ziel.x = Start_positionieren.x;
37 else
38      verst = 1;
39      if(Start.x < Start_positionieren.x)
40          Ziel.x = Start.x - 1;
41      else
42          Ziel.x = Start.x + 1;
43      end
44 end
45
46 Ziel.y = Start_positionieren.y;
47 Ziel.theta = 0;
```

D.1.6 Einparkphase - Erster Einparkschritt

```
1
2 function [Start B Ziel] = phase_ersterEinparkschritt(hindernis_rechts,
      fahrspurmitte, finalC)
3
4      global minAbstand rp r_KA Weg
5
6      omega_1.x = finalC.x;
7      omega_1.y = finalC.y + rp - 0.05;
8
9      if(sqrt((hindernis_rechts.x-omega_1.x)^2 + (omega_1.y -
           hindernis_rechts.y)^2) < (r_KA+minAbstand))
10         omega_1.y = hindernis_rechts.y + sqrt(((r_KA+minAbstand))^2-((
               hindernis_rechts.x-omega_1.x)^2));
11     end
12
13     omega_21.y = fahrspurmitte.y - rp;
14     omega_21.x = omega_1.x + sqrt((2*rp)^2 - (omega_1.y - omega_21.y)^2)
           ;
15
16     phi_1 = atan((omega_21.x-omega_1.x)/(omega_1.y - omega_21.y));
17     B1.x = omega_1.x + rp*sin(phi_1);
18     B1.y =  omega_1.y - rp*cos(phi_1);
19     Start1.x = omega_21.x;
20     Start1.y = fahrspurmitte.y;
21
22     phi_2 = atan((hindernis_rechts.x-omega_1.x)/(omega_1.y -
           hindernis_rechts.y));
23     B2.x = omega_1.x + rp*sin(phi_2);
24     B2.y =  omega_1.y - rp*cos(phi_2);
25     omega_22.x = omega_1.x + sin(phi_2) * 2 * rp;
26     omega_22.y = omega_1.y - cos(phi_2) * 2 * rp;
```

```
27      Start2.x = omega_22.x;
28      Start2.y = omega_22.y + rp;
29
30      Ziel.x = finalC.x;
31      Ziel.y = omega_1.y - rp;
32
33      if(B1.y > B2.y)
34          Start.x = Start1.x;
35          Start.y = Start1.y;
36          B.theta = phi_1;
37          B.x = B1.x;
38          B.y = B1.y;
39          Weg.k1 = rp * phi_1;
40          Weg.k2 = rp * phi_1;
41      else
42          Start.x = Start2.x;
43          Start.y = Start2.y;
44          B.theta = phi_2;
45          B.x = B2.x;
46          B.y = B2.y;
47          Weg.k1 = rp * phi_2;
48          Weg.k2 = rp * phi_2;
49      end
50
51      Start.theta = 0;
52      Ziel.theta = 0;
53
54  end
```

D.1.7 Einparkphase - N-Einparkschritte

```
1
2   function [B Ziel] = phase_NterEinparkschritt(Start, hindernis_rechts,
        hindernis_links)
3
4       global f_laenge_h minAbstand rp L f_laenge_v szenario Weg
5           B.x = 0;
6                   B.y = 0;
7       B.theta = 0;
8       richtung = 0;
9
10                  if( Start.x < (hindernis_rechts.x - (hindernis_rechts.x
                        - hindernis_links.x)/2 - L/2))
11                                  richtung = 1;   %von Links nach Rechts
                                        einparken
12      end
13
14                  omega1.x = Start.x + sin(Start.theta)*rp;
15                  omega1.y = Start.y - cos(-Start.theta)*rp;
16
17                  if(richtung == 0)   % Einparkschritt von Rechts nach
                        Links
18                                  if(omega1.x < hindernis_links.x +
                                        minAbstand + f_laenge_h)
19                                          % N2_Schritt3 ==> nur omega1
20                                           szenario = 6;
21                                  elseif(omega1.x - sin(Start.theta)*2*rp
                                        < hindernis_links.x + minAbstand +
                                        f_laenge_h)
22                                           % N2-Schritt2 ==> nur omega2
```

```
23                                          szenario = 5;
24                                  else
25                                          szenario = 4;
26                                  end
27                  else                % Einparkschritt von Links nach
                        Rechts
28                                  if(omega1.x > hindernis_rechts.x -
                                        minAbstand - L - f_laenge_v)
29                                          % N_Schritt3 ==> nur omega1
30                                          szenario = 3;
31                                  elseif(omega1.x + sin(-Start.theta)*2*rp
                                        > hindernis_rechts.x - minAbstand -
                                        L - f_laenge_v)
32                                          % N-Schritt2 ==> nur omega2
33                                          szenario = 2;
34                                  else
35                                          szenario = 1;
36                                  end
37      end
38
39          if(szenario == 1)
40          omega2.x = hindernis_rechts.x - minAbstand - L - f_laenge_v;
41                                  omega2.y = omega1.y + sqrt((2*rp)^2 - (
                                        omega2.x-omega1.x)^2);
42                                  Ziel.x = omega2.x;
43                                  Ziel.y = omega2.y - rp;
44                                  Ziel.theta = 0;
45          beta = atan((omega2.x-omega1.x)/(omega2.y - omega1.y));
46                                  B.x = omega2.x - rp*sin(beta);
47                                  B.y = omega2.y - rp*cos(beta);
48          B.theta = -beta;
49          phi1 = beta + Start.theta;
50          phi2 = beta;
51          Weg.k1 = rp * phi1;
52                                  Weg.k2 = rp * phi2;
53
54
55          elseif(szenario == 2)
56                                  omega2.x = omega1.x + 2*rp*sin(-Start.
                                        theta);
57                                  omega2.y = omega1.y+2*rp*cos(-Start.
                                        theta);
58
59                                  Ziel.x =  hindernis_rechts.x -
                                        minAbstand - f_laenge_v - L;
60                                  beta =  acos((omega2.x - Ziel.x)/rp);
61
62                                  Ziel.y = omega2.y - rp * sin(beta);
63          Ziel.theta = -(pi/2 - beta);
64                                  phi1 = 0;
65                                  phi2 = beta - Start.theta - pi/2;
66
67          Weg.k1 = rp * phi1;
68                                  Weg.k2 = rp * phi2;
69
70          elseif(szenario == 3)
71                                  Ziel.x =  hindernis_rechts.x -
                                        minAbstand - f_laenge_v - L;
72                                  beta =  acos((omega1.x - Ziel.x)/rp);
```

```
73
74                                  Ziel.y = omega1.y + rp * sin(beta);
75          Ziel.theta = pi/2 - beta;
76          phi1 = 0;
77                                  phi2 = beta + Start.theta - pi/2;
78
79          Weg.k1 = rp * phi1;
80                                  Weg.k2 = rp * phi2;
81          elseif(szenario == 4)
82                                  omega2.x = hindernis_links.x +
                                        minAbstand + f_laenge_h;
83                                  omega2.y = omega1.y + sqrt((2*rp)^2 - (
                                        omega2.x-omega1.x)^2);
84
85                                  Ziel.x = omega2.x;
86                                  Ziel.y = omega2.y - rp;
87                                  Ziel.theta = 0;
88
89                                  beta = atan((omega1.x-omega2.x)/(omega2.
                                        y - omega1.y));
90                                  B.x = omega2.x + rp*sin(beta);
91                                  B.y = omega2.y - rp*cos(beta);
92          B.theta = beta;
93
94                  phi1 = beta - Start.theta;
95          phi2 = beta;
96          Weg.k1 = rp * phi1;
97                                  Weg.k2 = rp * phi2;
98          elseif(szenario == 5)
99                                  omega2.x = omega1.x - 2*rp*sin(Start.
                                        theta);
100                                 omega2.y = omega1.y+2*rp*cos(Start.theta
                                        );
101
102                                 Ziel.x =  hindernis_links.x + minAbstand
                                         + f_laenge_h;
103                                 beta =  asin((Ziel.x - omega2.x)/rp);
104
105                                 Ziel.y = omega2.y - rp * cos(beta);
106                                                 Ziel.theta = beta;
107                                 phi1 = 0;
108                                 phi2 = Start.theta - beta;
109                                                 Weg.k1 = rp * phi1;
110                                 Weg.k2 = rp * phi2;
111         elseif(szenario == 6)
112                                 Ziel.x =  hindernis_links.x + minAbstand
                                        + f_laenge_h;
113                                 beta =  acos((Ziel.x - omega1.x)/rp);
114
115                                 Ziel.y = omega1.y + rp * cos(beta);
116                                 Ziel.theta = -(pi/2-beta);
117                                 phi1 = 0;
118                                 phi2 = beta - Start.theta - pi/2;
119                                 Weg.k1 = rp * phi1;
120                                 Weg.k2 = rp * phi2;
121     end
122 end
```

D.1.8 Einparkphase - Fahrzeug ausrichten

```
function [Ziel verst] = phase_ausrichten(Start)

global hindernis_links hindernis_rechts f_laenge_h f_laenge_v minAbstand
    L

if (Start.x+L/2) >= (hindernis_rechts.x-(hindernis_rechts.x-
    hindernis_links.x)/2)
    Ziel.x = hindernis_links.x + minAbstand + f_laenge_h;
else
    Ziel.x = hindernis_rechts.x - minAbstand - f_laenge_v - L;
end
Dtheta = abs(Start.theta*180/pi);

if(Dtheta >= 0 && Dtheta < 5)
    verst = 12;
else
    verst = 1;
end
Ziel.y = Start.y;
Ziel.theta = 0;
```

D.1.9 Funktion - Virtuelle Deichsel Lenkwinkelregelung

```

function status = outputFunction_dchsl(t,fhzg_state,flag, Ziel, verst,
    max_winkel, gui)

        global vm alpha_soll

        persistent xp_srg yp_srg xm_srg ym_srg count;

   if strcmp(flag, 'init') == 1
       % Initialisierung von Datenstrukturen
   elseif isempty(flag) == 1 % ==> Integrationsschritte

        theta = fhzg_state(1,length(fhzg_state(1,:)));
        xm = fhzg_state(2,length(fhzg_state(1,:)));
        ym = fhzg_state(3,length(fhzg_state(1,:)));
        xp = fhzg_state(4,length(fhzg_state(1,:)));
        yp = fhzg_state(5,length(fhzg_state(1,:)));
        alpha_ist = fhzg_state(6,length(fhzg_state(1,:)));
        s_ist = fhzg_state(7,length(fhzg_state(1,:)));

        ydl = yp - Ziel.y;
        xdl = xp - Ziel.x;

        if(xdl > 0 || xdl < 0) % Division durch Null
            sigma = atan(ydl*verst/xdl);
        else
            sigma = 0;
        end

        alpha_soll = sigma - theta;

        if(vm > 0)
            alpha_soll = -alpha_soll ;   % Vorwärtsfahrt
        else
            alpha_soll = alpha_soll;     % Rückwärtsfahrt
        end

```

```
37          if(alpha_soll > pi)              % Überlauf-Wraparound
38              alpha_soll = alpha_soll - 2 *pi;
39          elseif(alpha_soll < -pi)
40              alpha_soll = alpha_soll + 2 * pi;
41          end;
42
43          if(alpha_soll > max_winkel) % Lenkwinkel normieren
44              alpha_soll =  max_winkel;
45          elseif(alpha_soll < -max_winkel)
46              alpha_soll = -max_winkel;
47          end;
48      elseif strcmp(flag, 'done') == 1
49                                  %Nothing to do
50      end
51      status = 0; %1 == Stop Simulation, 0 == Continue Simulation
52  end
```

D.2 Bahnplanungsverfahren der Virtuellen Deichsel

D.2.1 Testbench - VD_1

```
1
2  global L max_winkel Start vm alpha_soll
3
4  maxSteps = 20000;                          % maximale Anzahl von
       Abtastschritten
5  dt = 0.017;                                % Abtastperiode in s
6  L = 0.257;                                 % Achsabstand SOC-Fahrzeug [m]
7  max_winkel_grad = 20;                      % Maximaler Lenkwinkel [Grad]
8  max_winkel = max_winkel_grad/180*pi;       % Maximaler Lenkwinkel [rad]
9  alpha_soll = 0;
10 minAbstand = 0.05;
11 f_breite = 0.202;                  % anpassen!!!! [m] (symetrisch)
12 f_laenge_v = 0.08;                 % Fahrzeuglaenge nach vorne, gemessen
       ab der Vorderachse
13 f_laenge_h = 0.05;                 % Fahrzeuglaenge nach hinten, gemessen
       ab der Hinterachse
14
15 rp = L / tan(max_winkel);                  % Wenderadius der Hinterachse
16
17 Start = struct('x',[],'y',[],'theta',[]); % Start- und
       Zielkonfigurationen
18 Ziel = struct('x',[],'y',[],'theta',[]);
19
20 Start.x = 0;
21 Start.y = 0;
22
23 Ziel.x = 0.7 - (2*minAbstand + L + f_laenge_v + f_laenge_h);
24 Ziel.y = 0;
25 verst_min = 1;
26 verst_max = 12;
27
28 vm = 0.1;
29 % Differenz der Konfigurationsräume
30 % 0=A, 1=B, 2=C, 3=D, 4=E, 5=F, 6=G, 7=H,
31 sim_raum_max = 7;
32
33 first_write = 1;
34
35 ergebnis = zeros(sim_raum_max+1,8);
```

```
36 loop_sim_raum = 1;
37 sim_raum = 0;
38
39
40 while loop_sim_raum == 1
41     switch (sim_raum)
42             case 0, % Raum A
43                 theta_sim_min = -19;
44                 theta_sim_max = -15;
45             case 1, % Raum B
46                 theta_sim_min = -14;
47                 theta_sim_max = -10;
48             case 2, % Raum C
49                 theta_sim_min = -9;
50                 theta_sim_max = -5;
51             case 3, % Raum D
52                 theta_sim_min = -4;
53                 theta_sim_max = -1;
54             case 4, % Raum E
55                 theta_sim_min = 0;
56                 theta_sim_max = 4;
57             case 5, % Raum F
58                 theta_sim_min = 5;
59                 theta_sim_max = 9;
60             case 6, % Raum G
61                 theta_sim_min = 10;
62                 theta_sim_max = 14;
63             case 7, % Raum H
64                 theta_sim_min = 15;
65                 theta_sim_max = 19;
66     end
67
68     loop_verst = 1;
69     verst = verst_min;
70
71     messreihe = zeros(verst_max,8);   %13 Zeilen, 8 Spalten
72
73     while loop_verst == 1
74         counter = 0;
75         loop_theta = 1;
76         theta_sim = theta_sim_min/180*pi;
77
78         while loop_theta == 1
79
80             alpha_soll = 0;
81             counter = counter + 1;
82
83             tspan = [0:dt:maxSteps*dt];
84             fhzg_init = [theta_sim;Start.x+L*cos(theta_sim);Start.y+L*
                   sin(theta_sim);Start.x;Start.y;0;0];
85
86             options = odeset ('maxStep',0.02,'OutputFcn', @(t,fhzg_state
                   ,flag)outputFunction_dchsl(t,fhzg_state,flag, Ziel, verst
                   , max_winkel),'Events' , @(t, fhzg_state)Event_dchsl(t,
                   fhzg_state, Ziel)) ;
87             [t fhzg_state te ye ie] = ode45(@(t, fhzg_state)
                   fhzg_modell_dchsl(t, fhzg_state, L, Ziel, verst,
                   max_winkel), tspan, fhzg_init, options);
88
```

```
89              index = length(fhzg_state(:,1));
90              theta = fhzg_state(index,1);
91              xm = fhzg_state(index,2);
92              ym = fhzg_state(index,3);
93              xp = fhzg_state(index,4);
94              yp = fhzg_state(index,5);
95              alpha_ist = fhzg_state(index,6);
96              s_ist = fhzg_state(index,7);
97              theta_grad = theta.* 180 ./ pi;
98
99              %messreihe
100             delta_theta = abs(theta)*180/pi;
101             delta_yp = abs(yp-Ziel.y);
102
103             messreihe(verst,1) = sim_raum;
104             messreihe(verst,2) = verst;
105             messreihe(verst,3) = messreihe(verst,3) + delta_yp;
106             messreihe(verst,4) = messreihe(verst,4) + delta_theta;
107
108             if(delta_yp > 0.05)
109                 messreihe(verst,5) = messreihe(verst,5) + 1;
110             end
111
112             if(delta_theta > 5)
113                 messreihe(verst,6) = messreihe(verst,6) + 1;
114             end
115
116             if(messreihe(verst,7) <= delta_yp)
117                 messreihe(verst,7) = delta_yp;
118             end
119
120             if(messreihe(verst,8) <= delta_theta)
121                 messreihe(verst,8) = delta_theta;
122             end
123
124             theta_sim = theta_sim + 1/180*pi;
125             if(theta_sim*180/pi > theta_sim_max + 0.5)
126                 loop_theta = 0;
127             end
128         end
129
130        messreihe(verst,3:4) = messreihe(verst,3:4)./counter; %
              Durchschnitt berechnen
131         verst = verst + 1;
132         if(verst > verst_max)
133             loop_verst = 0;
134         end
135     end
136
137     min_temp = 999;
138     ergebnis(sim_raum+1,1) = sim_raum;
139     for (k=verst_min: verst_max),
140         temp = messreihe(k,3)*100 + messreihe(k,4);
141         if(temp <= min_temp)
142             min_temp = temp;
143             ergebnis(sim_raum+1,2) = k;
144             ergebnis(sim_raum+1,3) = messreihe(k,3);
145             ergebnis(sim_raum+1,4) = messreihe(k,4);
146             ergebnis(sim_raum+1,5) = messreihe(k,5);
```

```
147             ergebnis(sim_raum+1,6) = messreihe(k,6);
148             ergebnis(sim_raum+1,7) = messreihe(k,7);
149             ergebnis(sim_raum+1,8) = messreihe(k,8);
150         end
151     end
152
153     if(first_write == 1)
154         first_write = 0;
155         fid = fopen('messreihe.txt','wt+');                    %
                Datei erstellen und zum schreiben öffnen
156         fprintf(fid,'<Simraum>\t<Verstärkung>\t\t<Durchschnitt delta y>\
                t\t<durchschnitt delta theta>\t\t<anzahl delta y > 0.05 m>\t\
                t<Anzahl delta theta >10 >\t< max delta yp >\t< max delta
                theta >\t<Anzahl Iterationen> \n');      %Überschrift
157     else
158         fid = fopen('messreihe.txt','at+');
159     end
160     fprintf(fid,'%c\t',char('A'+sim_raum));
161     fprintf(fid,'%2.0f\t',ergebnis(sim_raum+1,2));
162     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,3));
163     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,4));
164     fprintf(fid,'%3.0f\t',ergebnis(sim_raum+1,5));
165     fprintf(fid,'%3.0f\t',ergebnis(sim_raum+1,6));
166     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,7));
167     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,8));
168     fprintf(fid,'%3.0f\t',counter);
169     fprintf(fid,'\n');
170     fclose(fid);
171
172     sim_raum = sim_raum + 1;
173     if(sim_raum > sim_raum_max)
174         loop_sim_raum = 0;
175     end
176 end
```

D.2.2 Testbench - VD_2

```
1
2
3 global L max_winkel Start vm alpha_soll
4
5 maxSteps = 20000;                          % maximale Anzahl von
    Abtastschritten
6 dt = 0.017;                                % Abtastperiode in s
7 L = 0.257;                                 % Achsabstand SOC-Fahrzeug [m]
8 max_winkel_grad = 20;                      % Maximaler Lenkwinkel [Grad]
9 max_winkel = max_winkel_grad/180*pi;       % Maximaler Lenkwinkel [rad]
10 alpha_soll = 0;
11
12 rp = L / tan(max_winkel);                 % Wenderadius der Hinterachse
13
14 Start = struct('x',[],'y',[],'theta',[]); % Start- und
    Zielkonfigurationen
15 Ziel = struct('x',[],'y',[],'theta',[]);
16
17 Start.x = 0;
18 Start.y =0;
19
20 Ziel.x = 0.5; %l
21
```

```
22 verst_min = 1;
23 verst_max = 12;
24
25 vm = 0.1;
26 % Differenz der Konfigurationsräume
27 % 0=A, 1=B, 2=C, 3=D, 4=E, 5=F, 6=G, 7=H,
28 sim_raum_max = 7;
29
30 first_write = 1;
31
32 ergebnis = zeros(sim_raum_max+1,8);
33 loop_sim_raum = 1;
34 sim_raum = 0;
35
36 while loop_sim_raum == 1
37     switch (sim_raum)
38             case 0, % Raum A
39                 ziel_y_min = 0.06;
40                 ziel_y_max = 0.10;
41                 theta_sim_min = -9;
42                 theta_sim_max = -5;
43             case 1, % Raum B
44                 ziel_y_min = 0.06;
45                 ziel_y_max = 0.10;
46                 theta_sim_min = -4;
47                 theta_sim_max = -1;
48             case 2, % Raum C
49                 ziel_y_min = 0.06;
50                 ziel_y_max = 0.10;
51                 theta_sim_min = 1;
52                 theta_sim_max = 4;
53             case 3, % Raum D
54                 ziel_y_min = 0.06;
55                 ziel_y_max = 0.10;
56                 theta_sim_min = 5;
57                 theta_sim_max = 9;
58             case 4, % Raum E
59                 ziel_y_min = 0.00;
60                 ziel_y_max = 0.05;
61                 theta_sim_min = -9;
62                 theta_sim_max = -5;
63             case 5, % Raum F
64                 ziel_y_min = 0.00;
65                 ziel_y_max = 0.05;
66                 theta_sim_min = -4;
67                 theta_sim_max = -1;
68             case 6, % Raum G
69                 ziel_y_min = 0.00;
70                 ziel_y_max = 0.05;
71                 theta_sim_min = 1;
72                 theta_sim_max = 4;
73             case 7, % Raum H
74                 ziel_y_min = 0.00;
75                 ziel_y_max = 0.05;
76                 theta_sim_min = 5;
77                 theta_sim_max = 9;
78     end
79
80     loop_verst = 1;
```

```
81      verst = verst_min;
82
83      messreihe = zeros(verst_max+1,8);   %13 Zeilen, 8 Spalten
84
85      while loop_verst == 1
86          counter = 0;
87          loop_dy = 1;
88          Ziel.y = ziel_y_min;
89
90          while loop_dy == 1
91
92              loop_theta = 1;
93              theta_sim = theta_sim_min/180*pi;
94
95              while loop_theta == 1
96
97
98                  counter = counter + 1;
99
100                 tspan = [0:dt:maxSteps*dt];
101                 fhzg_init = [theta_sim;Start.x+L*cos(theta_sim);Start.y+
                        L*sin(theta_sim);Start.x;Start.y;0;0];
102
103                 options = odeset ('maxStep',0.02,'OutputFcn', @(t,
                        fhzg_state,flag)outputFunction_dchsl(t,fhzg_state,
                        flag, Ziel, verst, max_winkel),'Events' , @(t,
                        fhzg_state)Event_dchsl(t, fhzg_state, Ziel)) ;
104                 [t fhzg_state te ye ie] = ode45(@(t, fhzg_state)
                        fhzg_modell_dchsl(t, fhzg_state, L, Ziel, verst,
                        max_winkel), tspan, fhzg_init, options);
105
106                 index = length(fhzg_state(:,1));
107                 theta = fhzg_state(index,1);
108                 xm = fhzg_state(index,2);
109                 ym = fhzg_state(index,3);
110                 xp = fhzg_state(index,4);
111                 yp = fhzg_state(index,5);
112                 alpha_ist = fhzg_state(index,6);
113                 s_ist = fhzg_state(index,7);
114                 theta_grad = theta.* 180 ./ pi;
115
116                 %messreihe
117                 delta_theta = abs(theta)*180/pi;
118                 delta_yp = abs(yp-Ziel.y);
119
120                 messreihe(verst,1) = sim_raum;
121                 messreihe(verst,2) = verst;
122                 messreihe(verst,3) = messreihe(verst,3) + delta_yp;
123                 messreihe(verst,4) = messreihe(verst,4) + delta_theta;
124
125                 if(delta_yp > 0.05)
126                     messreihe(verst,5) = messreihe(verst,5) + 1;
127                 end
128
129                 if(delta_theta > 5)
130                     messreihe(verst,6) = messreihe(verst,6) + 1;
131                 end
132
133                 if(messreihe(verst,7) <= delta_yp)
```

```
134                     messreihe(verst,7) = delta_yp;
135                 end
136
137                 if(messreihe(verst,8) <= delta_theta)
138                     messreihe(verst,8) = delta_theta;
139                 end
140
141                 theta_sim = theta_sim + 1/180*pi;
142                 if(theta_sim*180/pi > theta_sim_max)
143                     loop_theta = 0;
144                 end
145             end
146             Ziel.y = Ziel.y + 0.01; % +1 cm
147             if(Ziel.y > ziel_y_max)
148                 loop_dy = 0;
149             end
150         end
151         messreihe(verst,3:4) = messreihe(verst,3:4)./counter; %
                Durchschnitt berechnen
152         verst = verst + 1;
153         if(verst > verst_max)
154             loop_verst = 0;
155         end
156     end
157     min_temp = 999;
158     ergebnis(sim_raum+1,1) = sim_raum;
159     for (k=verst_min: verst_max),
160         temp = messreihe(k,3)*100 + messreihe(k,4);
161         if(temp <= min_temp)
162             min_temp = temp;
163             ergebnis(sim_raum+1,2) = k;
164             ergebnis(sim_raum+1,3) = messreihe(k,3);
165             ergebnis(sim_raum+1,4) = messreihe(k,4);
166             ergebnis(sim_raum+1,5) = messreihe(k,5);
167             ergebnis(sim_raum+1,6) = messreihe(k,6);
168             ergebnis(sim_raum+1,7) = messreihe(k,7);
169             ergebnis(sim_raum+1,8) = messreihe(k,8);
170         end
171     end
172
173     if(first_write == 1)
174         first_write = 0;
175         fid = fopen('messreihe.txt','wt+');                    %
                Datei erstellen und zum schreiben öffnen
176         fprintf(fid,'<Simraum>\t<Verstärkung>\t\t<Durchschnitt␣delta␣y>\
                t\t<durchschnitt␣delta␣theta>\t\t<anzahl␣delta␣y␣>␣0.05␣m>\t\
                t<Anzahl␣delta␣theta␣>10␣>\t<␣max␣delta␣yp␣>\t<␣max␣delta␣
                theta␣>\t<Anzahl␣Iterationen>␣\n');      %Überschrift
177     else
178         fid = fopen('messreihe.txt','at+');
179     end
180     fprintf(fid,'%c\t',char('A'+sim_raum));
181     fprintf(fid,'%2.0f\t',ergebnis(sim_raum+1,2));
182     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,3));
183     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,4));
184     fprintf(fid,'%3.0f\t',ergebnis(sim_raum+1,5));
185     fprintf(fid,'%3.0f\t',ergebnis(sim_raum+1,6));
186     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,7));
187     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,8));
```

```
188     fprintf(fid,'%3.0f\t',counter);
189     fprintf(fid,'\n');
190     fclose(fid);
191
192     sim_raum = sim_raum + 1;
193     if(sim_raum > sim_raum_max)
194         loop_sim_raum = 0;
195     end
196 end
```

D.2.3 Testbench - VD_3

```
1
2  global L max_winkel Start vm alpha_soll
3
4  maxSteps = 20000;                          % maximale Anzahl von
       Abtastschritten
5  dt = 0.017;                                % Abtastperiode in s
6  L = 0.257;                                 % Achsabstand SOC-Fahrzeug [m]
7  max_winkel_grad = 20;                      % Maximaler Lenkwinkel [Grad]
8  max_winkel = max_winkel_grad/180*pi;       % Maximaler Lenkwinkel [rad]
9  alpha_soll = 0;
10
11 Start = struct('x',[],'y',[],'theta',[]); % Start- und
       Zielkonfigurationen
12 Ziel = struct('x',[],'y',[],'theta',[]);
13
14 Start.x = 0;
15 Start.y =0;
16
17 verst_min = 1;
18 verst_max = 12;
19
20 vm = 0.1;
21 % Differenz der Konfigurationsräume
22 % 1=A, 2=B, 3=C, 4=D, 5=E, 6=F, 7=G, 8=H, 9=I, 10=J, 11=K, 12=L, 13=M,
23 % 14=N, 15=O, 16=P, 17=Q, 18=R, 19=S, 20=T
24 sim_raum_max = 19;
25
26 first_write = 1;
27
28 ergebnis = zeros(sim_raum_max+1,8);
29 loop_sim_raum = 1;
30 sim_raum = 0;
31
32 while loop_sim_raum == 1
33     switch (sim_raum)
34             case 0, % Raum A
35                 ziel_x_min = 0.20;
36                 ziel_x_max = 0.29;
37                 ziel_y_min = 0.01;
38                 ziel_y_max = 0.05;
39                 theta_sim_min = -9;
40                 theta_sim_max = -5;
41             case 1, % Raum B
42                 ziel_x_min = 0.20;
43                 ziel_x_max = 0.29;
44                 ziel_y_min = 0.01;
45                 ziel_y_max = 0.05;
46                 theta_sim_min = -4;
```

```
47                  theta_sim_max = -1;
48              case 2, % Raum C
49                  ziel_x_min = 0.20;
50                  ziel_x_max = 0.29;
51                  ziel_y_min = 0.01;
52                  ziel_y_max = 0.05;
53                  theta_sim_min = 1;
54                  theta_sim_max = 4;
55              case 3, % Raum D
56                  ziel_x_min = 0.20;
57                  ziel_x_max = 0.29;
58                  ziel_y_min = 0.01;
59                  ziel_y_max = 0.05;
60                  theta_sim_min = 5;
61                  theta_sim_max = 9;
62              case 4, % Raum E
63                  ziel_x_min = 0.30;
64                  ziel_x_max = 0.49;
65                  ziel_y_min = 0.01;
66                  ziel_y_max = 0.05;
67                  theta_sim_min = -9;
68                  theta_sim_max = -5;
69              case 5, % Raum F
70                  ziel_x_min = 0.30;
71                  ziel_x_max = 0.49;
72                  ziel_y_min = 0.01;
73                  ziel_y_max = 0.05;
74                  theta_sim_min = -4;
75                  theta_sim_max = -1;
76              case 6, % Raum G
77                  ziel_x_min = 0.30;
78                  ziel_x_max = 0.49;
79                  ziel_y_min = 0.01;
80                  ziel_y_max = 0.05;
81                  theta_sim_min = 1;
82                  theta_sim_max = 4;
83              case 7, % Raum H
84                  ziel_x_min = 0.30;
85                  ziel_x_max = 0.49;
86                  ziel_y_min = 0.01;
87                  ziel_y_max = 0.05;
88                  theta_sim_min = 5;
89                  theta_sim_max = 9;
90              case 8, % Raum I
91                  ziel_x_min = 0.30;
92                  ziel_x_max = 0.49;
93                  ziel_y_min = 0.06;
94                  ziel_y_max = 0.10;
95                  theta_sim_min = -9;
96                  theta_sim_max = -5;
97              case 9, % Raum J
98                  ziel_x_min = 0.30;
99                  ziel_x_max = 0.49;
100                 ziel_y_min = 0.06;
101                 ziel_y_max = 0.10;
102                 theta_sim_min = -4;
103                 theta_sim_max = -1;
104             case 10, % Raum K
105                 ziel_x_min = 0.30;
```

```
106                 ziel_x_max = 0.49;
107                 ziel_y_min = 0.06;
108                 ziel_y_max = 0.10;
109                 theta_sim_min = 1;
110                 theta_sim_max = 4;
111             case 11, % Raum L
112                 ziel_x_min = 0.30;
113                 ziel_x_max = 0.49;
114                 ziel_y_min = 0.06;
115                 ziel_y_max = 0.10;
116                 theta_sim_min = 5;
117                 theta_sim_max = 9;
118             case 12, % Raum M
119                 ziel_x_min = 0.50;
120                 ziel_x_max = 0.99;
121                 ziel_y_min = 0.01;
122                 ziel_y_max = 0.05;
123                 theta_sim_min = -9;
124                 theta_sim_max = -5;
125             case 13, % Raum N
126                 ziel_x_min = 0.50;
127                 ziel_x_max = 0.99;
128                 ziel_y_min = 0.01;
129                 ziel_y_max = 0.05;
130                 theta_sim_min = -4;
131                 theta_sim_max = -1;
132             case 14, % Raum O
133                 ziel_x_min = 0.50;
134                 ziel_x_max = 0.99;
135                 ziel_y_min = 0.01;
136                 ziel_y_max = 0.05;
137                 theta_sim_min = 1;
138                 theta_sim_max = 4;
139             case 15, % Raum P
140                 ziel_x_min = 0.50;
141                 ziel_x_max = 0.99;
142                 ziel_y_min = 0.01;
143                 ziel_y_max = 0.05;
144                 theta_sim_min = 5;
145                 theta_sim_max = 9;
146             case 16, % Raum Q
147                 ziel_x_min = 0.50;
148                 ziel_x_max = 0.99;
149                 ziel_y_min = 0.06;
150                 ziel_y_max = 0.10;
151                 theta_sim_min = -9;
152                 theta_sim_max = -5;
153             case 17, % Raum R
154                 ziel_x_min = 0.50;
155                 ziel_x_max = 0.99;
156                 ziel_y_min = 0.06;
157                 ziel_y_max = 0.10;
158                 theta_sim_min = -4;
159                 theta_sim_max = -1;
160             case 18, % Raum S
161                 ziel_x_min = 0.50;
162                 ziel_x_max = 0.99;
163                 ziel_y_min = 0.06;
164                 ziel_y_max = 0.10;
```

```
165                 theta_sim_min = 1;
166                 theta_sim_max = 4;
167             case 19, % Raum T
168                 ziel_x_min = 0.50;
169                 ziel_x_max = 0.99;
170                 ziel_y_min = 0.06;
171                 ziel_y_max = 0.10;
172                 theta_sim_min = 5;
173                 theta_sim_max = 9;
174     end
175
176     loop_verst = 1;
177     verst = verst_min;
178
179     auswertung = zeros(verst_max+1,8);   %13 Zeilen, 8 Spalten
180
181     while loop_verst == 1
182         counter = 0;
183         loop_dx = 1;
184         Ziel.x = ziel_x_min;
185
186
187         while loop_dx == 1
188
189             loop_dy = 1;
190             Ziel.y = ziel_y_min;
191
192             while loop_dy == 1
193
194                 loop_theta = 1;
195                 theta_sim = theta_sim_min/180*pi;
196
197                 while loop_theta == 1
198
199                     counter = counter + 1;
200
201                     tspan = [0:dt:maxSteps*dt];
202                     fhzg_init = [theta_sim;Start.x+L*cos(theta_sim);
                            Start.y+L*sin(theta_sim);Start.x;Start.y;0;0];
203
204                     options = odeset ('maxStep',0.02,'OutputFcn', @(t,
                            fhzg_state,flag)outputFunction_dchsl(t,fhzg_state
                            ,flag, Ziel, verst, max_winkel),'Events' , @(t,
                            fhzg_state)Event_dchsl(t, fhzg_state, Ziel)) ;
205                     [t fhzg_state te ye ie] = ode45(@(t, fhzg_state)
                            fhzg_modell_dchsl(t, fhzg_state, L, Ziel, verst,
                            max_winkel), tspan, fhzg_init, options);
206
207                     index = length(fhzg_state(:,1));
208                     theta = fhzg_state(index,1);
209                     xm = fhzg_state(index,2);
210                     ym = fhzg_state(index,3);
211                     xp = fhzg_state(index,4);
212                     yp = fhzg_state(index,5);
213                     alpha_ist = fhzg_state(index,6);
214                     s_ist = fhzg_state(index,7);
215                     theta_grad = theta.* 180 ./ pi;
216
217                     %Auswertung
```

```
218                     delta_theta = abs(theta)*180/pi;
219                     delta_yp = abs(yp-Ziel.y);
220
221                     auswertung(verst,1) = sim_raum;
222                     auswertung(verst,2) = verst;
223                     auswertung(verst,3) = auswertung(verst,3) + delta_yp
                            ;
224                     auswertung(verst,4) = auswertung(verst,4) +
                            delta_theta;
225
226                     if(delta_yp > 0.05)
227                         auswertung(verst,5) = auswertung(verst,5) + 1;
228                     end
229
230                     if(delta_theta > 5)
231                         auswertung(verst,6) = auswertung(verst,6) + 1;
232                     end
233
234                     if(auswertung(verst,7) <= delta_yp)
235                         auswertung(verst,7) = delta_yp;
236                     end
237
238                     if(auswertung(verst,8) <= delta_theta)
239                         auswertung(verst,8) = delta_theta;
240                     end
241
242                     theta_sim = theta_sim + 1/180*pi;
243                     if(theta_sim*180/pi > theta_sim_max)
244                         loop_theta = 0;
245                     end
246                 end
247                 Ziel.y = Ziel.y + 0.01; % +1 cm
248                 if(Ziel.y > ziel_y_max)
249                     loop_dy = 0;
250                 end
251             end
252             Ziel.x = Ziel.x + 0.01;     % +1 cm
253             if(Ziel.x > ziel_x_max)
254                 loop_dx = 0;
255             end
256         end
257         auswertung(verst,3:4) = auswertung(verst,3:4)./counter; %
                Durchschnitt berechnen
258         verst = verst + 1;
259         if(verst > verst_max)
260             loop_verst = 0;
261         end
262     end
263     min_temp = 999;
264     ergebnis(sim_raum+1,1) = sim_raum;
265     for (k=verst_min: verst_max),
266         temp = auswertung(k,3)*100 + auswertung(k,4);
267         if(temp <= min_temp)
268             min_temp = temp;
269             ergebnis(sim_raum+1,2) = k;
270             ergebnis(sim_raum+1,3) = auswertung(k,3);
271             ergebnis(sim_raum+1,4) = auswertung(k,4);
272             ergebnis(sim_raum+1,5) = auswertung(k,5);
273             ergebnis(sim_raum+1,6) = auswertung(k,6);
```

```
274             ergebnis(sim_raum+1,7) = auswertung(k,7);
275             ergebnis(sim_raum+1,8) = auswertung(k,8);
276         end
277     end
278
279     if(first_write == 1)
280         first_write = 0;
281         fid = fopen('auswertung.txt','wt+');                    %
                Datei erstellen und zum schreiben öffnen
282         fprintf(fid,'<Simraum>\t<Verstärkung>\t\t<Durchschnitt delta y>\
                t\t<durchschnitt delta theta>\t\t<anzahl delta y > 0.05 m>\t\
                t<Anzahl delta theta >10 >\t< max delta yp >\t< max delta
                theta >\t<Anzahl Iterationen> \n');      %Überschrift
283     else
284         fid = fopen('auswertung.txt','at+');
285     end
286     fprintf(fid,'%c\t',char('A'+sim_raum));
287     fprintf(fid,'%2.0f\t',ergebnis(sim_raum+1,2));
288     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,3));
289     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,4));
290     fprintf(fid,'%3.0f\t',ergebnis(sim_raum+1,5));
291     fprintf(fid,'%3.0f\t',ergebnis(sim_raum+1,6));
292     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,7));
293     fprintf(fid,'%3.5f\t',ergebnis(sim_raum+1,8));
294     fprintf(fid,'%3.0f\t',counter);
295     fprintf(fid,'\n');
296     fclose(fid);
297
298     sim_raum = sim_raum + 1;
299     if(sim_raum > sim_raum_max)
300         loop_sim_raum = 0;
301     end
302 end
```

E Anforderungen an das Fahrzeugsystem

Im folgendem Abschnitt werde ich den Einsatzzweck des von mir verwendeten autonomen Fahrzeugs erläutern, das im Rahmen des FAUST-Projektes an der Hochschule für Angewandte Wissenschaften Hamburg entwickelt wird. Zu den Randbedingungen beim Einparkmanöver werde ich besonders eingehen.

E.1 Carolo-Cup

Der Carolo-Cup ist ein Hochschulübergreifender Wettbewerb, der jährlich von der TU-Braunschweig veranstaltet wird.
Hochschulen bekommen hier eine Plattform, um Fahrkonzepte für autonome Fahrzeuge zu präsentieren und unter Wettbewerbsbedingungen vorzuführen.

Der Rahmen des Wettbewerbs ist realitätsnah gestaltet, die Maße der Straße sowie der Modellfahrzeuge sind im Maßstab 1:10 festgelegt.
Je nach Disziplin gibt es unterschiedliche Hindernisse, beim Einparkmanöver sind es statische Hindernisse am Fahrbahnrand, die zufällig angeordnet sind und parkende Autos nachbilden. Beim Rundkurs sind es dynamische Hindernisse die fahrende Autos darstellen, sowie zufällig angeordnete fehlende Fahrbahnmarkierung machen die Wettbewerbsbedingungen realistisch.

Die Fahrdisziplinen und die zugehörige maximal zu erreichenden Punkte sind wie folgt vorgegeben:
Die Punkte werden Stufenweise von 0 bis maximal vergeben, das erfolgreichste Team bekommt

Statische Disziplinen	
Präsentation der Herstellkosten und der Energiebilanz	50 Punkte
Präsentation des Einparkkonzeptes	150 Punkte
Präsentation des Spurführungskonzeptes mit Ausweichmanövern	150 Punkte
Dynamische Disziplinen	
Einparken parallel	200 Punkte
Rundstrecke ohne Hindernisse	200 Punkte
Rundstrecke mit Hindernissen	250 Punkte
Maximal mögliche Gesamtpunktzahl	**1000 Punkte**

Abb. 61: *Carolocup Disziplinen und Punktevergabe: ca.* $1/3$ *der zu erreichenden Punkte können über ein gutes Konzept erreicht werden und ca.* $2/3$ *über das Gewinnen der Disziplinen*

die maximale Punktzahl und die Anderen erhalten anteilig Punkte.
Die Präsentationen dauern jeweils 10 Minuten, eine Fachjury bewertet die Präsentationen

E.2 Anforderungen an das Fahrzeug

Aus dem Regelwerk 2010 des Carolo-Cups sind folgende zwingende Anforderungen aufgeführt:

Die hier aufgeführten Anforderungen von Seiten des Carolo-Cups sind unbedingt einzuhalten.

ID	Kategorie	Text
F1	Fahrzeugmaße	Das Fahrzeugmodell muss im Maßstab 1:10 sein
F2	Fahrzeugmaße	Die maximale Höhe des Fahrzeugs muss 400 mm betragen
F3	Fahrzeugmaße	Die Spurweite muss mindestens 160 mm betragen
F4	Fahrzeugmaße	Der Radstand muss mindestens 200 mm betragen
F5	Karosserie	Die Fahrzeugverkleidung muss die Elektronik vor Spritzwasser schützen
F6	Karosserie	Die Fahrzeugverkleidung muss isolierend gegenüber Stromschlägen sein
F7	Karosserie	Die Fahrzeugverkleidung muss jederzeit schnell abbaubar sein
F8	Motor	Der Fahrzeugantrieb muss über einen elektrischen Motor erfolgen
F9	Räder	Die Anzahl der Räder muss vier sein
F10	Räder	Die Räder müssen auf zwei Achsen gelagert sein
F11	Räder	Die Anzahl der gelagerten Rädern muss auf einer Achse zwei betragen
F12	Lenkung	Die Lenkung des Autos muss über eine Zweiradlenkung der Vorderachse erfolgen
F13	Steuerung	Das Fahrzeug muss während einer dynamischen Disziplin autonom fahren
F14	Steuerung	Datenübertragungen sind während einer dynamischen Disziplin im Notfall allein über die Fernbedienung erlaubt
F15	Steuerung	Die Aktivierung des RC-Modus muss über die Fernbedienung erfolgen
F16	Steuerung	Die Deaktivierung des RC-Modus muss über die Fernbedienung erfolgen
F17 F18	Steuerung	Das Aktivieren des RC-Modus während eines Wettbewerbs darf nur in Notsituationen erfolgen
F19	RC-Modus	Die maximale Geschwindigkeit des Fahrzeugs darf im RC-Modus maximal 0.3 m/s betragen
F20	RC-Modus	Das Fahrzeug muss nach sofort nach dem aktivieren des RC-Modus anhalten
F21	RC-Modus	Das Fahrzeug darf im RC-Modus nur gelenkt und vorwärts oder rückwärts bewegt werden
F22	LEDs	Der aktive RC-Modus muss mit einer rundum sichtbaren blauen, blinkenden LED-Signalleuchte am höchsten Punkt des Fahrzeuges signalisiert werden
F23 F24	LEDs	Die Blinkfrequenz der LED-Signalleuchte muss 1Hz und ein Tastverhältnis von 50% aufweisen
F25	LEDs	Drei erkennbare Bremslichter müssen am Heck des Fahrzeuges montiert sein
F26 F27	LEDs	Negative Beschleunigungen von einer Dauer von mehr als 500 ms müssen signalisiert werden
F28	LEDs	Ein Überholvorgang muss durch seitliches blinken signalisiert werden
F29	LEDs	Das Abbiegen muss durch seitliches blinken signalisiert werden
F30	LEDs	Das Einparken muss durch seitliches blinken signalisiert werden
F31 F32	LEDs	Die LED-Fahrtrichtungsanzeiger müssen eine Blinkfrequenz von 1 Hz mit Tastverhätnis von 50 % haben
F33	LEDs	Die LED-Fahrtrichtungsanzeiger müssen Gelb oder Orange leuchten
F34 F35	LEDs	Die LED-Fahrtrichtungsanzeiger müssen an jeder Seite, jeweils vorne und hinten angebracht sein
F36	Sensoren	Die Sensoren müssen Sicherheitsanforderungen zur Nutzung in der Öffentlichkeit genügen, insbesondere Schädigung Dritter
F37	Sensoren	Ein Lasersensor darf maximal die Laser Klasse 2 besitzen
F38	Energieversorgung	Die Energieversorgung muss über Akkus erfolgen
F39	Allgemein	Alle Disziplinen sind mit dem gleichen Fahrzeug durchzuführen
F40	Allgemein	Es dürfen keine fertigen Lösungen von professionellen Ingenieuren oder anderen Dienstleistern übernommen werden

Tabelle 7: *Anforderungen an das Fahrzeug aus dem Regelwerk Carolo-Cup 2010. Ein Verstoß einer der aufgeführten Anforderungen hat den Ausschluss vom Wettbewerb oder einen Punktabzug zur Folge*

E.3 Anforderungen an den Einparkassistenten

ID	Kategorie	Text
E1	Start	Das Fahrzeug startet auf der Straße an einer weißen 40 mm breiten Startline
E2	Suche	Das Fahrzeug muss die rechte Fahrspur der Straße zur Parklückensuche entlang fahren
E3	Einparken	Das Fahrzeug muss parallel zur Fahrbahn und innerhalb der weißen Linien einparken
E4	Einparken	Das Fahrzeug muss in der Parklücke mit einer maximal zulässigen Winkelabweichung von 5 Grad stoppen
E5	Einparken	Der Abstand zum vorderen und hinteren Hindernis muss jeweils mindestens 10mm betragen
E6	Einparken	Die Hindernisse dürfen nicht berührt werden
E7	Einparken	Die äußere Parklückenbegrenzungslinie darf nicht überfahren werden
E8	Steuerung	Das starten des Einparkassistenten muss über ein Taster am Fahrzeug erfolgen
E9	Steuerung	Die Verwendung des RC-Modus ist während des Einparkens nicht erlaubt
E10	LEDs	Die Blinker müssen das Einparken signalisieren
E11	LEDs	Das Ende des Einparkmanövers muss durch Aufleuchten aller Fahrtrichtungsanzeiger angezeigt werden

Tabelle 8: *Anforderungen an den Einparkassistenten und den Einparkvorgang aus dem Regelwerk Carolo-Cup 2010. Ein Verstoß einer der aufgeführten Anforderungen hat den Ausschluss vom Wettbewerb oder einen Punktabzug zur Folge*

E.4 Steuerung des Fahrzeugsystems

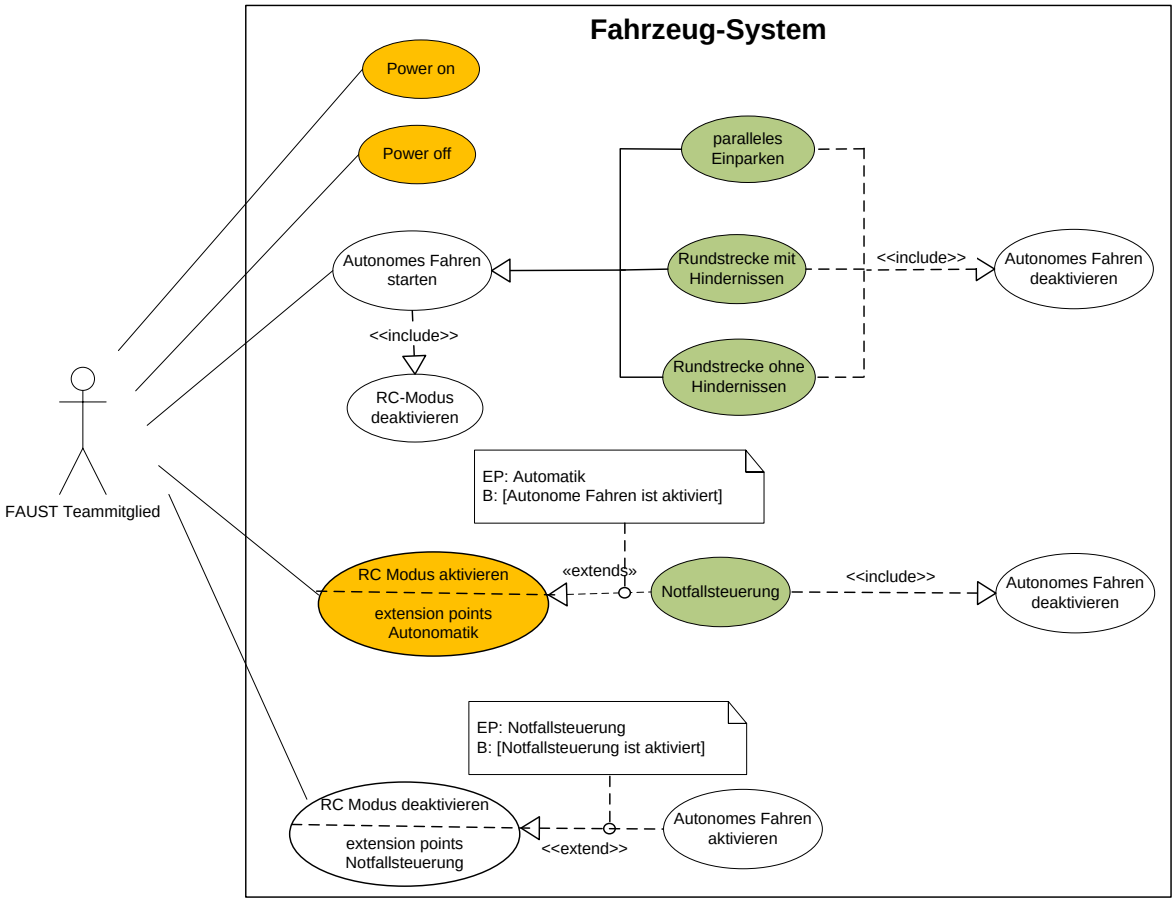


Abb. 62: *Anwendungsfalldiagramm zur Steuerung des Fahrzeugs*

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 14.07.2010