UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

ist
Institut für Softwaretechnik

Entwicklung einer Manöverdatenstruktur und Planung von Wegen für Car-like Robots

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science
im Studiengang Informatik

vorgelegt von

Christian Tobias Eiserloh

Erstgutachter: Prof. Dr. Dieter Zöbel
Institut für Softwaretechnik

Zweitgutachter: Dipl.-Inform. Christian Weyand
Institut für Softwaretechnik

Koblenz, im Mai 2011

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Abstract

This bachelor thesis deals basically with the design and the implementation of a data structure describing maneuvers for car-like robots. This data structure affords planning feasible routes which admit a continuous modification of the steering angle. The focus of this thesis rests on turn maneuvers, which consist of two clothoid arcs and one circular arc. One of the key aspects is the design of a concept for creating a corridor for these maneuvers. A corridor shall enclose the area which the vehicle covers during the corresponding maneuver. It finally provides a basis to guarantee collision freedom. Furthermore, a planning module is designed and implemented within this bachelor thesis. Based on specified construction lines, the planning module creates feasible routes made up of maneuvers.

Zusammenfassung

Diese Bachelorarbeit beschäftigt sich im Wesentlichen mit dem Entwurf und der Implementierung einer Manöverdatenstruktur für Car-like Robots. Diese Datenstruktur soll die Planung von fahrbaren Wegen mit kontinuierlicher Lenkwinkeländerung ermöglichen. Dabei stehen entsprechende Kurvenmanöver, die aus zwei Klothoiden und einem Kreisbogen bestehen, im Vordergrund. Zudem liegt ein Schwerpunkt auf der Entwicklung eines Konzepts zur Berechnung einer Hülle für entsprechende Manöver. Diese Hülle umschließt die Fläche, die das Fahrzeug bei der Ausführung eines Manövers einnimmt. Die Hülle ermöglicht es, beim koordinierten Fahren mit mehreren Fahrzeugen Kollisionsfreiheit zu garantieren. Auf der Manöverdatenstruktur aufbauend wird im Rahmen dieser Bachelorarbeit zudem ein Planungsmodul entworfen und implementiert, das zu vorgegebenen groben Wegen fahrbare, aus Manövern zusammengesetzte Wege erzeugt.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Diese Bachelorarbeit ist in der Arbeitsgruppe Echtzeitsysteme im Fachbereich Informatik an der Universität Koblenz-Landau entstanden. Die Arbeitsgruppe Echtzeitsysteme beschäftigt sich unter anderem mit dem assistierten und autonomen Fahren von Serienfahrzeugen. Die Entwicklung von Datenstrukturen zur Beschreibung von Fahrzeugbewegungen ist daher ein zentraler Bereich der Forschung. Diese Datenstrukturen ermöglichen Simulationen und Tests mit Modell-Fahrzeugen. Dabei steht die Planung der Fahrzeugbewegung und deren Überwachung in Echtzeit im Vordergrund. Das zentrale Ziel dabei ist es, Fahrbarkeit und Kollisionsfreiheit zu garantieren.

1.1 Motivation

Bei der Planung von Fahrzeugbewegungen werden sogenannte Manöver verwendet. Diese beschreiben grundlegende Bewegungen und dienen als elementare Bausteine der Planung von zusammengesetzten Wegen ([Zöbel, 2003]). Bei der Planung von Manövern und daraus zusammengesetzten Wegen ist die Fahrbarkeit von zentraler Bedeutung. Das Manöver muss also so konstruiert sein, dass es unter Berücksichtigung der kinematischen Eigenschaften eines Fahrzeugs von diesem Fahrzeug gefahren werden kann. Bei dem von der Arbeitsgruppe Echtzeitsysteme vertretenen Konzept der Wegplanung ist eine sprunghafte Änderung des Lenkwinkels erlaubt. Beispielsweise wird bei einem Übergang von einer Geradeaus- zu einer Kurvenfahrt eine sprunghafte Lenkwinkeländerung verwendet. Wenn das Fahrzeug präzise fahren soll, ist es dabei

notwendig, anzuhalten, um den Lenkwinkel im Stand zu ändern. Um ein zwischenzeitliches Anhalten zu vermeiden, wäre es denkbar, eine sprunghafte Änderung des Lenkwinkels auszuschließen und stattdessen eine stetige Änderung vorauszusetzen. Diese hat den Vorteil, dass man den Lenkwinkel während der Fahrt bei konstanter Geschwindigkeit kontinuierlich ändern kann. Das Konzept der kontinuierlichen Lenkwinkeländerung wird durch die Planungsmethode von Fraichard und Scheuer ([Fraichard u. Scheuer, 2004], [Scheuer, 1999], [Scheuer u. Fraichard, 1997a], [Scheuer u. Fraichard, 1997b]) realisiert. Die Methode beschreibt entsprechende Kurvenmanöver für Car-like Robots. Diese Bachelorarbeit beschäftigt sich im Wesentlichen mit dieser Planungsmethode.

Das zweite zentrale Ziel bei der Planung von Manövern und Wegen in der Arbeitsgruppe Echtzeitsysteme ist die Kollisionsfreiheit. Dazu wird die Fläche betrachtet, die ein Fahrzeug bei einem Manöver einnimmt. Das Manöver wird mit einer Hülle versehen, die der durch das Fahrzeug belegten Fläche entspricht oder diese umschließt. Diese Hülle ermöglicht es, Kollisionsfreiheit zu gewährleisten. Eine Betrachtung der Fläche fehlt bei der Planungsmethode von Fraichard und Scheuer. Daher soll die Planungsmethode im Rahmen dieser Bachelorarbeit um ein entsprechendes Konzept erweitert werden.

1.2 Ziele der Arbeit

Im Rahmen dieser Bachelorarbeit soll eine Manöverdatenstruktur, basierend auf der Planungsmethode von Fraichard und Scheuer, entworfen und implementiert werden. Die Datenstruktur soll mit dem Ziel des Kollisionsausschlusses um ein Hüllenkonzept erweitert werden. Darauf aufbauend soll die vorhandene Anwendung *EZLeitstand* um ein Planungsmodul erweitert werden. Dieses Planungsmodul soll dem Benutzer die Möglichkeit bieten, grobe Wege vorzugeben, zu denen das Planungsmodul fahrbare, aus Manövern zusammengesetzte Wege erzeugt. Bei der Entwicklung dieser Software soll schrittweise vorgegangen werden:

- Zunächst ist eine Einarbeitung in die problemspezifische Literatur notwendig. Dazu gehört einerseits die Planungsmethode von Fraichard und Scheuer, andererseits die Manöverproblematik der Arbeitsgruppe Echtzeitsysteme.
- Anschließend sollen die Anforderungen an die oben genannte Software definiert werden. Diese sollen in einem Pflichtenheft festgehalten werden. Darin soll auch der zeitliche Ablauf der weiteren Arbeitsschritte festgelegt werden.
- Im Folgenden soll die Planungsmethode von Fraichard und Scheuer um das eigene Hüllenkonzept erweitert werden. Außerdem soll die Methode derart erweitert werden, dass damit auch Wege, die sich aus Manövern zusammensetzen, geplant werden können.
- Als erster Teil der Software soll die Manöverdatenstruktur für Car-like Robots entworfen und implementiert werden. Die Datenstruktur soll auf der Planungsmethode von Fraichard und Scheuer sowie auf der oben genannten Erweiterung basieren.
- Darauf aufbauend soll die Anwendung *EZLeitstand* um das erwähnte Planungsmodul erweitert werden.

1.3 Einordnung in den wissenschaftlichen Kontext

Wie am Anfang des Kapitels beschrieben, beschäftigt sich die Arbeitsgruppe Echtzeitsysteme unter anderem mit dem assistierten und autonomen Fahren von Serienfahrzeugen ([Zöbel, 2008], [Zöbel, 2003]). Im Bezug auf das autonome Fahren soll primär ein koordiniertes Fahren mit mehreren Fahrzeugen auf einem abgeschlossenen Gelände ermöglicht werden. Im Bereich des Gütertransports ist es beispielsweise denkbar, Fahrzeuge, die auch im normalen Straßenverkehr eingesetzt werden, auf dem Werksgelände autonom fahren zu lassen. Dies würde einen zeitlich effizienteren Ablauf beim Warentransport ermöglichen und Personalkosten senken (vgl. [Weyand u. a., 2010]). Beim assistierten Fahren steht die Unterstützung des Fahrers, insbe-

sondere beim Rückwärtsfahren mit Anhänger, im Vordergrund. Die Arbeitsgruppe Echtzeitsysteme legt ihren Schwerpunkt in beiden Anwendungsbereichen auf Fahrzeuggespanne, bestehend aus Zugfahrzeug und einachsigem Anhänger. Dabei wird mit dem in Abschnitt 1.1 beschriebenen Manöverkonzept gearbeitet, das zum einen die Fahrbarkeit voraussetzt, zum anderen die Fläche betrachtet, die das Fahrzeug bei einem Manöver einnimmt. Wie bereits beschrieben, behandelt die Planungsmethode von Fraichard und Scheuer Kurvenmanöver für Car-like Robots ([Fraichard u. Scheuer, 2004]). Car-like Robots haben vier Räder, wobei die Vorderräder lenkbar sind. Das Hüllenkonzept, das im Rahmen dieser Bachelorarbeit auf die Kurvenmanöver angewendet werden soll, wird demnach speziell für Car-like Robots angepasst. Die Entwicklung dieses Konzepts soll letztlich auch die Generizität des Manöverkonzepts verdeutlichen.

Die Planungsmethode von Fraichard und Scheuer realisiert das Konzept der kontinuierlichen Lenkwinkeländerung. Sie behebt damit die Schwachstelle anderer Planungsmethoden mit sprunghafter Lenkwinkeländerung (siehe Abschnitt 1.1). Diese Schwachstelle war letztlich die Motivation für Fraichard und Scheuer, eine eigene Planungsmethode zu entwickeln. In ihren Ausführungen wird daher auch auf entsprechende andere Planungsmethoden hingewiesen. Dazu gehören beispielsweise die Planungsmethode von Dubins ([Dubins, 1957]), bei der nur die Vorwärtsfahrt vorgesehen ist, oder die Planungsmethode von Reeds und Shepp ([Reeds u. Shepp, 1990]), die zusätzlich die Fahrtrichtung rückwärts ermöglicht. Beide Verfahren basieren auf elementaren Bewegungen, allerdings findet auch bei beiden eine nicht-kontinuierliche Lenkwinkeländerung statt. Die Methode von Fraichard und Scheuer kann als Weiterentwicklung der Methode von Reeds und Shepp betrachtet werden. Die Erweiterung der Planungsmethode um das Manöverkonzept im Rahmen dieser Bachelorarbeit führt somit den Entwicklungsverlauf fort.

1.4 Aufbau der Arbeit

In Kapitel 2 werden zunächst einige Grundlagen behandelt. Dazu gehören unter anderem die Planungsmethode von Fraichard und Scheuer sowie die für die Bachelorarbeit relevante Software der Arbeitsgruppe Echtzeitsysteme. In Kapitel 3 werden die Anforderungen an die Software definiert, die im Rahmen dieser Bachelorarbeit entwickelt werden soll. Anschließend wird die Entwicklung dieser Software vorgestellt. Kapitel 4 behandelt die Manöverdatenstruktur, die die Planungsmethode von Fraichard und Scheuer realisieren soll. Kapitel 5 behandelt das Planungsmodul, das die Planung von Wegen mit Hilfe der Manöverdatenstruktur ermöglichen soll. In Kapitel 6 werden Ergebnisse der Manöverberechnung und der Wegplanung verglichen und bewertet. In Kapitel 7 folgt eine abschließende Bewertung sowie ein Ausblick.

2 Grundlagen

An dieser Stelle wird zunächst in die Terminologien der für die Arbeit wichtigen Themengebiete eingeführt. Dabei werden die im Weiteren verwendeten Begrifflichkeiten definiert. Anschließend werden zunächst die mathematischen Grundlagen für die Berechnung der Kurvenmanöver mit stetiger Lenkwinkeländerung (*CC Turns*) vorgestellt. Diese werden in der Planungsmethode von Fraichard und Scheuer ([Fraichard u. Scheuer, 2004], [Scheuer, 1999], [Scheuer u. Fraichard, 1997a], [Scheuer u. Fraichard, 1997b]) behandelt. Damit sind sie zentraler Bestandteil der Manöverdatenstruktur, die zur Realisierung dieses Konzepts im Rahmen dieser Arbeit entwickelt werden soll. Außerdem wird die Softwarebibliothek *EZauto* vorgestellt, auf deren Funktionalitäten die im Rahmen dieser Arbeit erstellte Software zurückgreift. Dazu wird insbesondere auch auf die Anwendung *EZLeitstand* eingegangen, in die diese Software integriert werden soll.

2.1 Terminologie

Die Bewegung eines Fahrzeugs stellt eine Änderungen seiner **Konfiguration** dar. Die Konfiguration $q \in \mathcal{Q}$ setzt sich dabei aus bestimmten Eigenschaften zusammen wie zum Beispiel der aktuellen **Position** und **Ausrichtung**, dem aktuellen **Lenkwinkel** oder dem aktuellen Einknickwinkel bei Fahrzeugen mit einachsigem Anhänger. Der **Konfigurationsraum** $\mathcal{Q}$ vereinigt alle möglichen Konfigurationen des Fahrzeugs. Die Änderung der Konfiguration kann beschrieben werden durch sogenannte **Manöver**. Der Manövertyp bestimmt dabei, welche Eigenschaften sich wie ändern.

Ein **Kurvenmanöver** beispielsweise beschreibt eine bestimmte Änderung der Ausrichtung des Fahrzeugs, eine **Geradeausfahrt** hingegen eine Änderung der Position ohne Ausrichtungsänderung.

Ein Manöver besteht aus einem Pfad und einer Hülle. Der **Pfad** ist eine Abbildung $\Pi\colon [0, S] \to \mathcal{Q}$ mit $S \in \mathbb{R}^+$. Er liefert zu jeder Stelle $s \in [0, S]$ des Manövers eine Konfiguration $q \in \mathcal{Q}$ im Konfigurationsraum. Der Anfang des Manövers wird dabei durch $s = 0$, das Ende durch $s = S$ beschrieben. Im Rahmen dieser Bachelorarbeit gehören neben der Position $(x, y) \in \mathcal{P} \subseteq \mathbb{R}^2$ des Fahrzeugs auch die Ausrichtung $\theta \in \mathcal{A} \subseteq [0, 2\pi]$ sowie der Lenkwinkel $\alpha \in \mathcal{L} \subseteq [-\alpha_{max}, \alpha_{max}]$ zur Konfiguration. Die Konfiguration ist also $q = (x, y, \theta, \alpha)$, der Konfigurationsraum $\mathcal{Q} = \mathcal{P} \times \mathcal{A} \times \mathcal{L}$. Außerdem wird die Definitionsmenge der Abbildung Π beschränkt auf das Intervall $[0, 1]$, d.h. $S = 1$. Die Abbildung Π beschreibt eine stetige Kurve, die ein Bezugspunkt des Fahrzeugs beim zugehörigen Manöver abfährt. Insbesondere beschreibt die Projektion in $\mathcal{P}$ die Fahrkurve im Koordinatensystem. Im Weiteren wird der Begriff „Pfad“ synonym für die Abbildung Π sowie für die Fahrkurve verwendet. Der Pfad wird für ein Fahrzeug als **fahrbar** bezeichnet, wenn er von dem Fahrzeug, unter Berücksichtigung dessen kinematischer Eigenschaften, abgefahren werden kann.

Die **Hülle** ist eine Fläche, innerhalb der sich das Fahrzeug während des Manövers befindet. Sie kann durch ein Rechteck, einen Ring, ein bestimmtes Polygon oder eine andere beliebige Fläche realisiert werden. Diese muss nicht minimal sein. Entscheidend ist nur, dass sich das Fahrzeug bei diesem Manöver jederzeit innerhalb dieser Fläche aufhält. Die Fläche dient letztlich dazu, Kollisionsfreiheit zu gewährleisten.

Manöver lassen sich als elementare Bausteine zu einem **Weg** zusammensetzen. Umgekehrt betrachtet ist ein Weg eine Aneinanderreihung mehrerer Manöver.

Kurvenmanöver definieren sich wie bereits erwähnt im Wesentlichen über die damit erzielte Ausrichtungsänderung. Allerdings gibt es verschiedene Ansätze dafür, wie man dieses Manöver definiert und konstruiert. Die gewünschte Ausrichtungsänderung

kann dabei auf verschiedene Weise erreicht werden.

Ein naheliegender Ansatz hierzu ist die Realisierung als Fahrt mit konstantem Lenkwinkel bis zum Erreichen der entsprechenden Ausrichtungsänderung. Man würde also eine Kurve mit konstanter **Krümmung** abfahren, einen Kreisbogen. Die Krümmung entspricht dabei dem Kehrwert des Radius dieses Kreisbogens. Bei hintereinander geplanten Manövern ergibt sich hierbei ein Nachteil. Die Übergänge zweier hintereinander liegender Manöver sind in der Regel nicht glatt. Möchte man zum Beispiel ein solches Kurvenmanöver mit bestimmtem Lenkwinkel an eine Geradeausfahrt anschließen, hat die Fahrkurve am Übergang einen Knick. So ist man – wenn man präzise fahren will – gezwungen, nach der Geradeausfahrt anzuhalten, um den Lenkwinkel im Stand für die Kreisbogenfahrt entsprechend zu ändern. Der Lenkwinkel und damit auch die Krümmung der Fahrkurve werden also genau in diesem Berührungspunkt **sprunghaft** geändert. Diese nicht-kontinuierliche Krümmungsänderung ist im linken Teil von Abbildung 2.1 dargestellt.

Genau an diesem Punkt setzt die Planungsmethode von Fraichard und Scheuer ([Fraichard u. Scheuer, 2004]) an. Die Methode zielt darauf ab, Pfade zu entwerfen, deren Krümmung entweder konstant bleibt oder sich **stetig** ändert, wie im rechten Teil von Abbildung 2.1 dargestellt. Dies gibt einem Fahrzeug die Möglichkeit, den Pfad mit konstanter Geschwindigkeit abzufahren und den Lenkwinkel während der Fahrt kontinuierlich zu ändern.

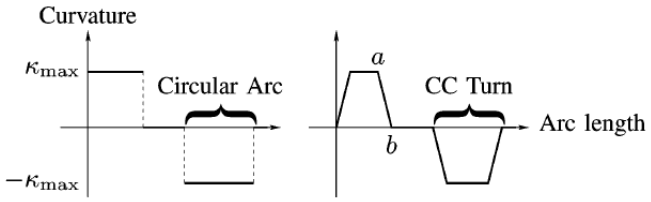

Abbildung 2.1: Vergleich nicht-kontinuierlicher mit kontinuierlicher Krümmungsänderung aus [Fraichard u. Scheuer, 2004].

Dazu werden als Kurvenmanöver sogenannte ***Continuous Curvature Turns*** (kurz: ***CC Turn***) verwendet. Wie in Abbildung 2.1 zu sehen, ist die Krümmung κ der Fahr-

kurve zunächst 0 und wächst dann linear mit der Bogenlänge auf κ_{max}. Eine solche Kurve mit stetiger Krümmungsänderung nennt man **Klothoide**. An diese schließt sich ein Kreisbogen mit eben dieser Krümmung κ_{max} und dem entsprechenden Radius κ_{max}^{-1} an. Darauf folgt ein zweites Klothoidenstück, das symmetrisch zum ersten ist, dessen Krümmung sich aber von κ_{max} zu 0, also genau umgekehrt, verändert.

Eine Klothoide wird definiert über ihre **Schärfe** σ. Sei $\kappa(0)$ die Anfangskrümmung. Dann lässt sich die Krümmung in Abhängigkeit von der Länge l nach [Fraichard u. Scheuer, 2004] durch folgende Formel ausdrücken:

$$\kappa(l) = \sigma \cdot l + \kappa(0) \tag{2.1}$$

Umgekehrt lässt sich die Länge l einer Klothoide, die mit $\kappa(0) = 0$ beginnt und mit $\kappa(l) = \kappa_{max}$ enden soll, bei gegebener Schärfe σ folgendermaßen berechnen:

$$l = \frac{\kappa(l)}{\sigma} = \frac{\kappa_{max}}{\sigma} \tag{2.2}$$

Ein *CC Turn* besteht also aus einem Kreisbogen und zwei Klothoiden, die einen „glatten Übergang“ zwischen dem Kreisbogen und den davor- und dahinterliegenden Geradenstücken realisieren.

In den folgenden Abschnitten werden weitere Begrifflichkeiten eingeführt und an entsprechender Stelle auch erläutert.

2.2 Kurvenmanöver mit stetiger Lenkwinkeländerung

Wie im vorherigen Abschnitt erläutert, basiert die Planungsmethode von Fraichard und Scheuer auf Kurvenmanövern mit stetiger Lenkwinkeländerung (*CC Turns*). Dabei wird von einem bestimmten Fahrzeug-Modell ausgegangen. Im Folgenden wird daher zunächst dieses Fahrzeug-Modell näher betrachtet. Anschließend werden die

mathematischen Grundlagen zur Berechnung der Kurvenmanöver vorgestellt.

2.2.1 Fahrzeug-Modell

Die Planungsmethode von Fraichard und Scheuer basiert auf dem Fahrzeug-Modell eines sogenannten *Continuous Curvature Cars* (kurz: *CC Car*). Dieses entspricht einem *Car-like Robot*, also einem Fahrzeug mit vier Rädern ohne Anhänger.

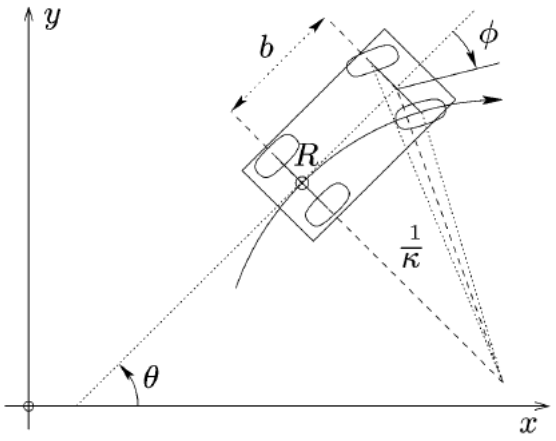

Abbildung 2.2: Fahrzeug-Modell aus [Fraichard u. Scheuer, 2004].

Wie in Abbildung 2.2 zu sehen, bezieht sich die Fahrkurve hier immer auf den Mittelpunkt der hinteren Fahrzeugachse R. Die weiteren dargestellten Werte sind der Radstand L_1 (hier: b), die aktuelle Ausrichtung θ, der aktuelle Lenkwinkel α (hier: ϕ) sowie die aus dem Radstand und dem Lenkwinkel resultierende aktuelle Krümmung κ der Kurve, deren Kehrwert $\frac{1}{\kappa}$ dem Radius eines Kreisbogens dieser Krümmung entspricht. Weiterhin für die Berechnung wichtig sind die Lenkgeschwindigkeit $\dot{\alpha}$, die daraus resultierende Krümmungsänderung der Fahrkurve $\sigma = \dot{\kappa}$, sowie die Fahrgeschwindigkeit v. Die Krümmungsänderung σ entspricht dabei der in Abschnitt 2.1 beschriebenen Schärfe der Klothoide. Abbildung 2.2 veranschaulicht auch folgende Zusammenhänge:

$$\kappa = \frac{\tan(\alpha)}{L_1} \tag{2.3}$$

$$\text{woraus folgt} \qquad \sigma = \dot{\kappa} = \frac{\dot{\alpha}}{L_1 \cdot cos^2(\alpha)} \tag{2.4}$$

Aufgrund natürlicher Beschränkungen sind dabei der Lenkwinkel und die Lenkgeschwindigkeit begrenzt und damit auch die Krümmung ($|\kappa| \leq \kappa_{max}$) und die Krümmungsänderung ($|\sigma| \leq \sigma_{max}$). Zudem wird von einer konstanten Fahrgeschwindigkeit ausgegangen. Es wird festgelegt, dass $|\nu| = 1$, wobei das Vorzeichen die Fahrtrichtung (vorwärts oder rückwärts) bestimmt.

Das Modell wird nach [Fraichard u. Scheuer, 2004] beschrieben durch die aktuelle *Konfiguration* des Fahrzeugs und deren Änderung. Die Konfiguration $q = (x, y, \theta, \kappa)$ setzt sich zusammen aus der aktuellen Position, Ausrichtung und Krümmung. Die Änderung der Konfiguration ist folgendermaßen definiert:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\kappa} \end{pmatrix} = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \\ \kappa \\ 0 \end{pmatrix} \upsilon + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \sigma \tag{2.5}$$

2.2.2 Konstruktion eines Kurvenmanövers

Ein *CC Turn* definiert sich wie alle Kurvenmanöver über die damit erzielte Ausrichtungsänderung δ des Fahrzeugs. Zudem muss die Krümmung des Kreisbogens κ_{max} bekannt sein. Diese ergibt sich aus dem maximal verwendeten Lenkwinkel α_{max}[1] und dem Radstand L_1 des Fahrzeugs (siehe Gleichung 2.3). Die Kreisbogenkrümmung ist gleichzeitig auch die maximale Krümmung der Klothoiden. Weiterhin wichtig zur Berechnung der Klothoiden ist deren Schärfe σ. Diese ergibt sich aus der verwendeten Lenkgeschwindigkeit des Fahrzeugs (siehe Gleichung 2.4). Aus diesen Werten lässt sich nun ein *CC Turn* berechnen.

[1] Im Rahmen dieser Bachelorarbeit ist α_{max} der Lenkwinkel des Kreisbogens eines *CC Turns*, also der maximal verwendete Lenkwinkel. In den Veröffentlichungen der Arbeitsgruppe Echtzeitsysteme bezeichnet α_{max} dagegen den maximal möglichen Lenkwinkel eines Fahrzeugs.

Zunächst wird davon ausgegangen, dass es sich um eine Linkskurve handelt, die im Punkt $(x_s, y_s) = (0, 0)$ mit der Ausrichtung $\theta_s = 0$ und Krümmung $\kappa_s = 0$ beginnt und im Punkt (x_g, y_g) mit der Ausrichtung $\theta_g = \delta$ und Krümmung $\kappa_g = 0$ endet. Die Berechnung des Manövers findet unter diesen Einschränkungen statt. Bei der Wegplanung kann das berechnete Manöver durch entsprechende Transformation an die vorgesehene Stelle verschoben werden.

Die Start-Konfiguration $q_s = (x_s, y_s, \theta_s, \kappa_s)$ ist also $q_s = (0, 0, 0, 0)$. Nach der ersten Klothoide mit der Schärfe σ und der Länge $l_{kloth} = \kappa_{max}/\sigma$ (siehe Gleichung 2.2) erreicht man die Konfiguration q_i (siehe Abbildung 2.3).

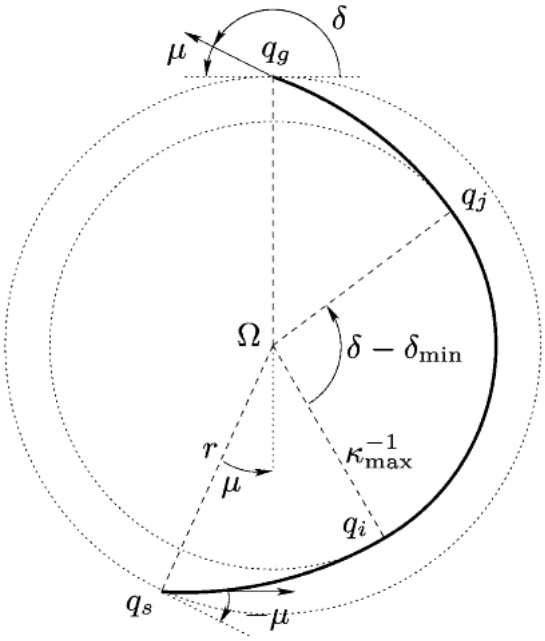

Abbildung 2.3: *CC Turn* aus [Fraichard u. Scheuer, 2004].

Die Konfiguration q_i berechnet sich nach [Fraichard u. Scheuer, 2004] wie folgt:

$$q_i = \begin{cases} x_i &= \sqrt{\frac{\pi}{\sigma}}\, C_f\left(\sqrt{\frac{\kappa_{max}^2}{\pi\cdot\sigma}}\right) \\ y_i &= \sqrt{\frac{\pi}{\sigma}}\, S_f\left(\sqrt{\frac{\kappa_{max}^2}{\pi\cdot\sigma}}\right) \\ \theta_i &= \frac{\kappa_{max}^2}{2\cdot\sigma} \\ \kappa_i &= \kappa_{max} \end{cases} \tag{2.6}$$

C_f und S_f sind die Fresnel-Integrale. Diese sind definiert als:

$$C_f(t) = \int_0^t \cos\left(\frac{\pi\cdot u^2}{2}\right) \mathrm{d}u \tag{2.7}$$

$$S_f(t) = \int_0^t \sin\left(\frac{\pi\cdot u^2}{2}\right) \mathrm{d}u \tag{2.8}$$

Auf die erste Klothoide folgt der Kreisbogen mit dem Radius κ_{max}^{-1} und dem Winkel $\delta - \delta_{min}$, wobei nach [Fraichard u. Scheuer, 2004] gilt:

$$\delta_{min} = \frac{\kappa_{max}^2}{\sigma} = 2\cdot\theta_i \tag{2.9}$$

δ_{min} ist also die Ausrichtungsänderung, die mit den beiden Klothoiden erreicht wird.

Der Mittelpunkt Ω des Kreisbogens berechnet sich nach [Fraichard u. Scheuer, 2004] folgendermaßen:

$$\Omega = \begin{cases} x_\Omega &= x_i - \kappa_{max}^{-1}\cdot\sin(\theta_i) \\ y_\Omega &= y_i + \kappa_{max}^{-1}\cdot\cos(\theta_i) \end{cases} \tag{2.10}$$

Der dazugehörige äußere Kreis nennt sich *CC Circle* und hat den Radius

$$r = \sqrt{x_\Omega^2 + y_\Omega^2} \tag{2.11}$$

Die Tangenten am *CC Circle* bilden mit den Ausrichtungen von q_s und q_g jeweils den Winkel μ. Diesen findet man auch zwischen $\overline{\Omega\, R_s}$ (mit $R_s = (x_s, y_s)$) und der

Senkrechten zur x-Achse durch Ω (siehe Abbildung 2.3). Damit gilt

$$\mu = \arctan(x_\Omega / y_\Omega) \tag{2.12}$$

Weitergehend lässt sich zeigen, dass der Rest des Winkels der Klothoide, also der Winkel zwischen der Senkrechten zur x-Achse durch Ω und $\overline{\Omega R_i}$ (mit $R_i = (x_i, y_i)$) θ_i entspricht. Damit schließt die Klothoide den Winkel $\mu + \theta_i$ ein.

Mit dem Kreisbogen erreicht man also die Konfiguration $q_j = (x_j, y_j, \theta_j, \kappa_j)$ mit $\kappa_j = \kappa_{max}$. Die Ausrichtung θ_j ergibt sich aus den Ausrichtungsänderungen, die mit der ersten Klothoide (θ_i) und mit dem Kreisbogen ($\delta - \delta_{min}$) erreicht werden:

$$\begin{aligned} \theta_j &= \theta_i + (\delta - \delta_{min}) \\ &= \theta_i + (\delta - 2 \cdot \theta_i) \\ &= \theta_i + \delta - 2 \cdot \theta_i \\ &= \delta - \theta_i \end{aligned} \tag{2.13}$$

Die Position (x_j, y_j) lässt sich berechnen, indem man (x_i, y_i) mit dem Winkel des Kreisbogens $\delta - \delta_{min}$ um Ω rotiert.

Auf den Kreisbogen folgt nun die zweite Klothoide. Diese ist symmetrisch zur ersten, hat also ebenfalls die Länge $l_{kloth} = \kappa_{max}/\sigma$ (siehe Gleichung 2.2) und schließt ebenfalls den Winkel $\mu + \theta_i$ ein. Damit beträgt der Gesamtwinkel, der vom *CC Turn* eingeschlossen wird

$$\begin{aligned} & 2 \cdot (\mu + \theta_i) + (\delta - \delta_{min}) \\ = \; & 2 \cdot (\mu + \theta_i) + (\delta - (2 \cdot \theta_i)) \\ = \; & 2 \cdot \mu + 2 \cdot \theta_i + \delta - 2 \cdot \theta_i \\ = \; & 2 \cdot \mu + \delta \end{aligned} \tag{2.14}$$

Da sich die Krümmung der zweiten Klothoide im Gegensatz zur ersten von κ_{max} nach 0 **verkleinern** soll, hat sie die invertierte Schärfe $-\sigma$. Mit der zweiten Klothoide erreicht man $q_g = (x_g, y_g, \theta_g, \kappa_g)$ mit $\kappa_g = 0$ und $\theta_g = \delta$. Die Position (x_g, y_g) lässt sich berechnen, indem man (x_s, y_s) mit dem Winkel $2 \cdot \mu + \delta$ (siehe Gleichung 2.14) um Ω rotiert (siehe Abbildung 2.3).

Somit sind nun die vier Konfigurationen q_s, q_i, q_j und q_g sowie δ_{min} und der *CC Circle* mit Ω, r und μ bekannt. Weitere Konfigurationen innerhalb der ersten Klothoide lassen sich berechnen, indem man Gleichung 2.6 anwendet und dabei anstelle von κ_{max} eine von der entsprechenden Länge abhängige Krümmung (zwischen 0 und κ_{max}) verwendet. Diese Konfigurationen lassen sich durch Spiegelung und Rotation auf die zweite Klothoide übertragen. Weitere Konfigurationen innerhalb des Kreisbogens erhält man durch Rotieren von q_i um Ω. Es ist also möglich, jeden Punkt des Pfades genau zu bestimmen.

Im Folgenden werden zwei Möglichkeiten vorgestellt, bei zunächst ungünstigen Ausgangswerten dennoch einen möglichst kurzen Pfad zu erhalten.

Spezialfall *Elementary Path* ($\delta < \delta_{min}$)

Dieser erste Sonderfall tritt auf, wenn $\delta < \delta_{min}$ gilt. Das bedeutet, dass die Ausrichtungsänderung, die man mit beiden Klothoiden erreicht (δ_{min}), größer ist, als die insgesamt gewünschte Ausrichtungsänderung δ. Das hat zur Folge, dass man eine Schleife fährt (siehe Abbildung 2.4).

Ursache dafür sind die für diese geringe Ausrichtungsänderung zu stark gewählte Schärfe σ und Krümmung κ_{max}. Man lenkt also zu schnell und insgesamt zu stark ein. Um eine Schleifenfahrt zu vermeiden und damit, wie in Abbildung 2.4 zu sehen, den direkten Weg von q_s nach q_g zu nehmen, wählt man stattdessen die Schärfe und Maximalkrümmung der Klothoide genau so, dass man mit zwei hintereinander liegenden Klothoiden (ohne dazwischen liegenden Kreisbogen) q_g und damit die ge-

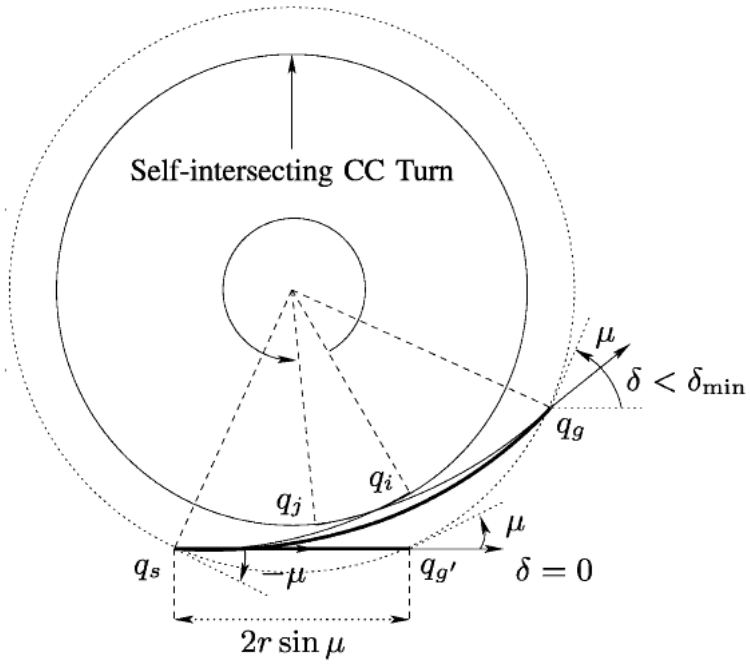

Abbildung 2.4: *Elementary Path* aus [Fraichard u. Scheuer, 2004].

wünschte Ausrichtungsänderung erreicht. Die entsprechende neue Schärfe σ_{elem} und Krümmung $\kappa_{max_{elem}}$ sind dabei in jedem Fall kleiner. Somit sind auch Lenkwinkel und Lenkgeschwindigkeit geringer und können damit aus kinematischer Sicht vom Fahrzeug gefahren werden.

Die entsprechende Schärfe berechnet sich nach [Fraichard u. Scheuer, 2004] folgendermaßen:

$$\sigma_{elem} = \frac{\pi \cdot (\cos(\delta/2) \cdot C_f(\sqrt{\delta/\pi}) + \sin(\delta/2) \cdot S_f(\sqrt{\delta/\pi}))^2}{r^2 \cdot \sin^2(\delta/2 + \mu)} \tag{2.15}$$

C_f und S_f sind hierbei wieder die Fresnel-Integrale (siehe Gleichungen 2.7, 2.8).

Die Ausrichtungsänderung $\delta_{min_{elem}}$ der beiden angepassten Klothoiden entspricht dabei der gewünschten Ausrichtungsänderung δ. Es gilt also nach Gleichung 2.9:

$$\delta_{min_{elem}} = \frac{\kappa^2_{max_{elem}}}{\sigma_{elem}} = \delta$$

$$\text{woraus folgt} \qquad \kappa_{elem} = \sqrt{\delta \cdot \sigma_{elem}} \tag{2.16}$$

Aus der Krümmung lässt sich mit Gleichung 2.3 wiederum der maximale Lenkwinkel $\alpha_{max_{elem}}$ berechnen. Damit stehen wieder alle für die Berechnung eines *CC Turns* benötigten Werte zur Verfügung. Die Konfigurationen $q_{i_{elem}} = q_{j_{elem}}$ und der *CC Circle* sind also wie im Normalfall berechenbar.

Beim *Elementary Path* ist zu beachten, dass bei ungünstigen Werten von δ und μ der Nenner der Gleichung 2.15 gegen 0 gehen kann. In solchen Fällen kann es zu extrem verfälschten Werten von σ_{elem} kommen, was sich auch auf die im weiteren Verlauf berechneten Werte und schließlich auf die gesamte Kurve auswirkt. Es ist dabei sogar möglich, dass die Schärfe größer wird als ursprünglich vorgegeben und die resultierende Fahrkurve somit für das Fahrzeug unter Umständen nicht mehr fahrbar ist. Daher ist es empfehlenswert, im Fall $\sin(\delta/2 + \mu) \to 0$ auf den *Elementary Path* zu verzichten oder zumindest die entsprechenden Werte auf Fahrbarkeit zu überprüfen.

Spezialfall *Rückwärtsfahrt des Kreisbogens* ($\delta > \delta_{min} + \pi$)

Der zweite Sonderfall tritt auf, wenn $\delta > \delta_{min} + \pi$ gilt. Dabei ist einerseits die Ausrichtungsänderung δ sehr groß und andererseits die Ausrichtungsänderung, die man mit den beiden Klothoiden erreicht (δ_{min}) verhältnismäßig klein. Das bedeutet, dass der Kreisbogen zwischen den Klothoiden sehr groß, genauer gesagt $> \pi$ ist. In diesem Fall ist es günstiger, nicht den Kreisbogen (vorwärts) zu fahren, sondern stattdessen den restlichen Teil des Kreises, der ja damit $< \pi$ ist, rückwärts zu fahren (siehe Abbildung 2.5).

Damit ändert sich der Winkel des Kreisbogens zu $\delta - \delta_{min} - 2 \cdot \pi$. Auch die Länge des Kreisbogens und die Konfigurationen innerhalb des Kreisbogens müssen dementsprechend anders berechnet werden. Ansonsten bleibt alles unverändert (Konfigurationen q_s, q_i, q_j und q_g sowie δ_{min} und der *CC Circle* mit Ω, r und μ).

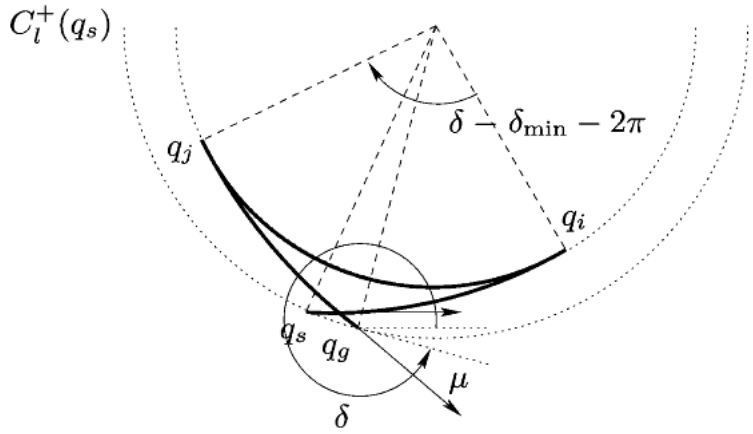

Abbildung 2.5: *Rückwärtsfahrt des Kreisbogens* aus [Fraichard u. Scheuer, 2004].

2.3 Softwarebibliothek EZauto

Die in der Arbeitsgruppe Echtzeitsysteme entwickelte Softwarebibliothek *EZauto* bietet die Möglichkeit verschiedene Fahrzeug-Modelle zu verwalten und aus verschiedenen Sichten zu betrachten. Der Kern der Software ist dabei an keine spezielle Anwendung gekoppelt, sondern auf die flexible Nutzung durch beliebige Anwendungen im Bereich des autonomen und assistierten Fahrens ausgelegt. Implementierte Anwendungen können die vorhandenen Funktionalitäten ohne große Änderungen nutzen, um beispielsweise Simulationen durchzuführen und dabei bestimmte Informationen des Fahrzeugs sowie das Fahrzeug selbst auf verschiedene Weise darzustellen (vgl. [Wojke, 2005]). Als Programmiersprache wird C++ verwendet.

Die Architektur der Software ist in Abbildung 2.6 dargestellt. Sie ist in drei Schichten gegliedert: Das Fundament, der Kern und die Anwendungen. Das Fundament stellt grundlegende Funktionalitäten bereit, die möglichst abstrakt gehalten sind, um sie in beliebigen Anwendungsbereichen zu verwenden. Dazu gehören Themengruppen wie die Objektverwaltung, die Mathematik oder die Darstellung. Die Funktionalitäten des Kerns beschränken sich jeweils auf einen Anwendungsbereich, sind aber immer noch so abstrakt gehalten, dass sie von verschiedenen Anwendungen genutzt werden können. Dazu gehören beispielsweise die Fahrzeugbeschreibung oder die Wegverfol-

gung. Die Anwendungen selbst behandeln konkrete Problemstellungen und greifen dazu auf die Funktionalitäten des Fundaments und des Kerns zurück. Zu den bereits implementierten Anwendungen gehören zum Beispiel ein Tool zum Generieren von Abbildungen von Fahrzeug-Modellen oder eine Anwendung zum Steuern von autonomen Fahrzeugen (vgl. [Wojke, 2005]).

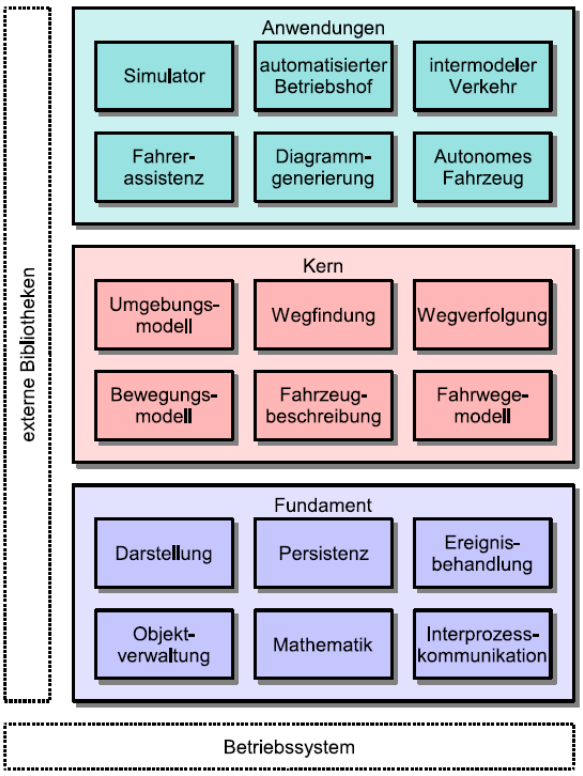

Abbildung 2.6: Architektur der Softwarebibliothek *EZauto* aus [Wojke, 2005].

Die einzelnen Funktionalitäten innerhalb der Themengruppen werden von Schnittstellen und diese implementierenden Komponenten bereitgestellt. Die Schnittstellen sind dabei möglichst abstrakt gehalten. Das Ziel ist auch hier die Flexibilität und Wiederverwendbarkeit (vgl. [Wojke, 2005]).

Im Folgenden werden die Themengruppen der Softwarebibliothek näher vorgestellt, auf deren Funktionalitäten die im Rahmen dieser Bachelorarbeit entwickelte Software zurückgreift.

Eine zentrale Themengruppe ist die Objektverwaltung, die dem Fundament zuzuordnen ist und ein grundlegendes Element der gesamten Softwarebibliothek darstellt. Die Objektverwaltung ermöglicht es, bestimmte Informationen eines Objekts zu erhalten. Dazu gehört insbesondere die Anzahl der Referenzen auf das Objekt. Objekte, die nicht mehr referenziert werden, können somit automatisch gelöscht werden. Entsprechende Methoden werden in den Schnittstellen bereitgestellt. Um diese Funktionalitäten in den eigenen Komponenten zu realisieren, reicht es aus, auf die dazu bereitgestellten Makros zurückzugreifen (vgl. [Wojke, 2005]).

Eine weitere Themengruppe ist die Mathematik (Fundament) mit ihren beiden Modulen für Funktionen und Geometrie. Diese stellen Funktionalitäten bereit, die insbesondere für die Berechnung der Kurvenmanöver hilfreich sind. Dazu gehören unter anderem die im Modul Geometrie implementierten Datentypen für Punkte, Polygone und Winkelberechnungen sowie die im Modul Funktion bereitgestellten Funktionsklassen, die sich speziell für die Realisierung von Manöverpfaden eignen.

Wichtig für die Visualisierung des Fahrzeugs und der Pfade und Hüllen des Manövers ist die Themengruppe Darstellung (Fundament). Diese bietet die Möglichkeit, visualisierbare Komponenten zum Beispiel auf dem Bildschirm oder in PostScript-Dateien auszugeben. Dazu wird eine Schnittstelle für Zeichenflächen verwendet (vgl. [Wojke, 2005]).

Als relevante Themengruppe des Fundaments ist noch die Ereignisbehandlung zu nennen. Diese bietet die Möglichkeit einer ereignisbasierten Kommunikation mit dem Fahrzeug (vgl. [Wojke, 2005]). Das im Rahmen dieser Arbeit entwickelte Planungsmodul macht davon zwar nur rudimentär Gebrauch, allerdings werden die entsprechenden Funktionalitäten bei einer Weiterentwicklung des Planungsmoduls, insbesondere bei der Implementierung des Fahrens, von Bedeutung sein (mehr dazu in Abschnitt 5.5).

Die zentrale Themengruppe des Kerns ist die Fahrzeugbeschreibung. Dazu existieren eine kinematische, eine struktur- und eine darstellungsorientierte Sichtweise. Die ki-

nematische Beschreibung des Fahrzeugs enthält sowohl statische Attribute wie den Radstand oder den maximalen Lenkwinkel, als auch dynamische Attribute wie die aktuelle Position oder den aktuellen Lenkwinkel. Die Struktur eines Fahrzeugs entspricht einem Baum abstrakter Teile. Diese Teile können mit weiteren Informationen ausgestattet sein. Dazu gehört beispielsweise auch eine visuelle Repräsentation, die als darstellungsorientierte Sichtweise verstanden werden kann (vgl. [Wojke, 2005]). Die im Rahmen dieser Arbeit entwickelte Software macht auch hiervon nur rudimentär Gebrauch, nämlich durch Abfragen bestimmter Attribute des Fahrzeugs. Aber auch die in dieser Themengruppe enthaltenen Funktionalitäten werden bei einer Weiterentwicklung des Planungsmoduls relevant sein.

2.4 Anwendung EZLeitstand

Die Anwendung *EZLeitstand* ist Teil der im vorherigen Abschnitt vorgestellten Softwarebibliothek *EZauto*. Sie bietet in erster Linie die Möglichkeit eine Verbindung zu einem simulierten oder realen (Modell-)Fahrzeug herzustellen, um dieses zu steuern. Dazu kann eines der in die Anwendung integrierten Anwendungsmodule ausgewählt werden. Dieses verfügt wie das Hauptprogramm über eine Benutzeroberfläche, über die das Fahrzeug je nach Anwendungsmodul auf verschiedene Weise manuell oder autonom gesteuert werden kann. Das Fahrzeug kann dabei aufgrund der in *EZauto* vorhandenen Funktionalitäten in einem Koordinatensystem innerhalb der Benutzeroberfläche grafisch dargestellt werden (siehe Abbildung 2.7). So gibt es zum Beispiel ein Anwendungsmodul, das es erlaubt, ein Fahrzeug per Tastatur und Maus oder per handelsüblichem Joystick zu steuern. Ein anderes bereits implementiertes Anwendungsmodul ermöglicht die Planung und Verfolgung von Pfaden für Fahrzeuggespanne bestehend aus Zugfahrzeug und Anhänger. Dabei können die geplanten Wege genauso visualisiert werden wie das Fahrzeug selbst. Die Anwendung ist mit Hilfe der plattformübergreifenden *C++*-Bibliothek *wxWidgets* zur Entwicklung grafischer Benutzeroberflächen realisiert.

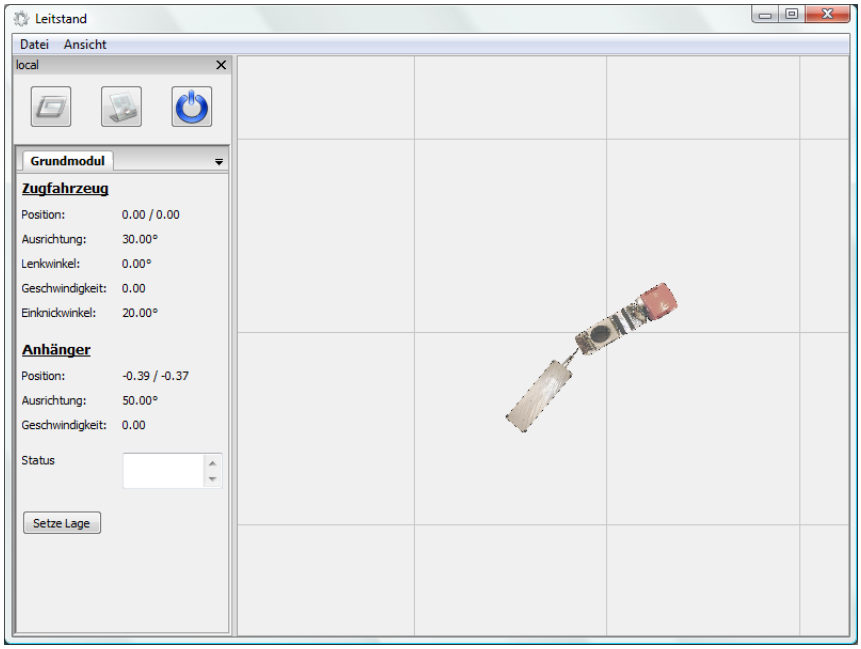

Abbildung 2.7: Benutzeroberfläche der Anwendung *EZLeitstand*.

Da im Rahmen dieser Bachelorarbeit ein solches Anwendungsmodul entwickelt werden soll, wird der Aufbau der Anwendung *EZLeitstand*, insbesondere im Hinblick auf die Integration eines neuen Anwendungsmoduls, im Folgenden näher betrachtet (siehe Abbildung 2.8).

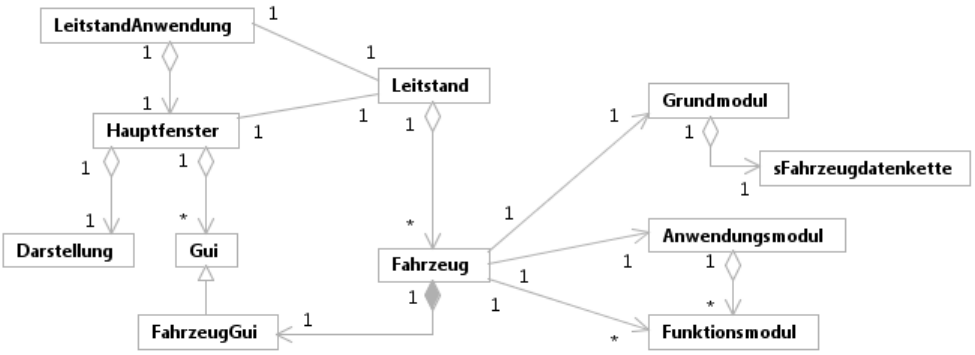

Abbildung 2.8: Aufbau der Anwendung *EZLeitstand*.

Die Anwendung selbst wird durch die von *wxApp* abgeleitete Klasse *LeitstandAnwendung* realisiert. Diese verwaltet lediglich zwei Timer, die letztlich bewirken, dass der

am Anfang erzeugte *Leitstand* regelmäßig die Benutzeroberfläche zeichnen lässt und mit dem Fahrzeug kommuniziert. Die Klasse *Leitstand* besitzt die übergeordnete Benutzeroberfläche (*Hauptfenster*, abgeleitet von *wxFrame*) und verwaltet beliebig viele verbundene Fahrzeuge (Klasse *Fahrzeug*). Jedes Fahrzeug besitzt ein *Grundmodul*, ein *Anwendungsmodul* und mehrere *Funktionsmodule*. Das Grundmodul kommuniziert direkt mit dem Fahrzeug und aktualisiert die Fahrzeugdaten. Anwendungsmodule können, wie oben beschrieben, verschiedenste Funktionalitäten bereitstellen. Funktionsmodule dienen unter anderem der Visualisierung bestimmter Fahrzeugdaten. Implementiert ist zum Beispiel ein Funktionsmodul, das den Lenkwinkelverlauf während der Fahrt aufzeichnet und visuell darstellt. Jedes Fahrzeug besitzt zudem eine eigene GUI (*FahrzeugGui*), über die die Verbindung zum Fahrzeug geregelt und Anwendungsmodule ausgewählt werden können.

Grundmodul, *Anwendungsmodule* und *Funktionsmodule* sind von der Klasse *Modul* abgeleitet (siehe Abbildung 2.9). Ein Modul besitzt eine GUI (*ModulGui*, abgeleitet von *Gui*) und ist einem Fahrzeug zugeordnet. Weiterhin ist ein Modul von *sEreignisempfaenger* abgeleitet, wodurch es zur Kommunikation mit dem Fahrzeug genutzte Ereignisse verarbeiten kann.

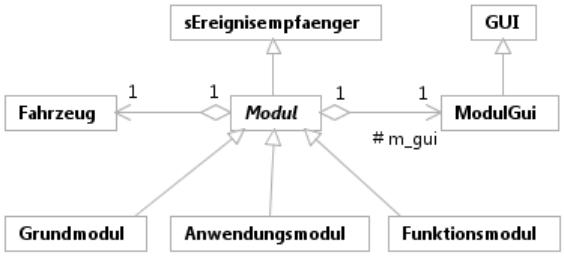

Abbildung 2.9: Anwendungsmodul aus *EZLeitstand*.

3 Definition der Anforderungen

In diesem Kapitel werden die Anforderungen an die Software definiert, die im Rahmen dieser Bachelorarbeit entwickelt werden soll. Dazu gehört zum einen eine Manöverdatenstruktur für Car-like Robots. Diese soll die in Abschnitt 2.2 eingeführten Kurvenmanöver mit stetiger Lenkwinkeländerung (*CC Turns*) realisieren. Zum anderen gehört dazu ein Planungsmodul zum Planen von Wegen für Car-like Robots. Die Wegplanung soll dabei auf die Manöverdatenstruktur zurückgreifen. Das Planungsmodul soll als Anwendungsmodul der vorhandenen Anwendung *EZLeitstand* realisiert werden.

3.1 Eingabeparameter des Car-like Robots

Folgende Parameter und Eigenschaften des Car-like Robots muss man aus einer Fahrzeugbeschreibungskette laden können:

- Geometrische Eigenschaften wie Breite, Länge und hinterer Überhang des Fahrzeugs
- Kinematische Eigenschaften wie den maximalen Lenkwinkel des Fahrzeugs

3.2 Eingabeparameter eines CCTurns

Folgende Parameter muss man für einen *CC Turn* vorgeben können:

- Radstand L_1 (aus Fahrzeugbeschreibungskette)
- Lenkwinkel des Kreisbogens α_{max} (< maximaler Lenkwinkel des Fahrzeugs)
- Krümmungsänderung bzw. Schärfe der Klothoiden σ
- Ausrichtungsänderung δ
- *elementary* (Wahrheitswert): Verwendung des *Elementary Path*
- *rueckwaerts* (Wahrheitswert): *Rückwärtsfahrt des Kreisbogens*
- Startkonfiguration $q_s = (x_s, y_s, \theta_s, \kappa_s)$

3.3 Berechenbare Werte eines CCTurns

Folgende Werte müssen für einen *CC Turn* aus den vorgegebenen Werten berechnet werden können:

- Krümmung des Kreisbogens κ_{max}
- Radius des Kreisbogens κ_{max}^{-1}
- Ausrichtungsänderung der Klothoiden δ_{min}
- Konfigurationen q_i, q_j, q_g
- *CC Circle*: $\Omega = (x_\Omega, y_\Omega)$, r, μ
- Länge der Klothoiden und des Kreisbogens
- Schärfe der Klothoiden σ_{elem} für *Elementary Path*
- An σ_{elem} angepasste Werte bei Verwendung des *Elementary Path*

3.4 Manöverdatenstruktur

(M) Es muss eine Manöverdatenstruktur für Car-like Robots, basierend auf der Methode von Fraichard und Scheuer realisiert werden.

(M.1) Die Datenstruktur muss die Möglichkeit bieten, folgende Arten von Manövern zu verwalten:

- *CC Turn*
- Geradeausfahrt

(M.2) Die Datenstruktur muss dem Benutzer die Möglichkeit bieten, die in Abschnitt 3.1 angegebenen Werte und Eigenschaften des Car-like Robots aus einer *Fahrzeugbeschreibungskette* (*EZauto*) auszulesen.

(M.3) Die Datenstruktur muss dem Benutzer die Möglichkeit bieten, die in Abschnitt 3.2 angegebenen Werte für einen *CC Turn* festzulegen.

(M.4) Die Datenstruktur muss Funktionen bereitstellen, mit denen die in Abschnitt 3.3 angegebenen Werte für einen *CC Turn* berechnet werden können.

(M.5) Die Datenstruktur muss Funktionen bereitstellen, um die in den Abschnitten 3.1, 3.2 und 3.3 angegebenen Werte abfragen zu können.

(M.6) Die Datenstruktur muss Funktionen bereitstellen, die einen Pfad für ein Manöver zurückliefern, den ein Bezugspunkt des Fahrzeugs bei diesem Manöver abfährt.

(M.7) Die Datenstruktur muss Funktionen bereitstellen, mit denen eine Hülle (Fläche) für ein Manöver berechnet werden kann, innerhalb der sich das Fahrzeug bei diesem Manöver befindet.

(M.8) Die Datenstruktur muss die Möglichkeit bieten, das Manöver inklusive des Pfades und der Hülle im Koordinatensystem zu verschieben und zu drehen.

3.5 Planungsmodul

(P) Die vorhandene Software *EZLeitstand* muss um ein Planungsmodul erweitert werden, das dem Benutzer ermöglicht, Polygonzüge für einen Car-like Robot vorzugeben, auf denen basierend das Planungsmodul fahrbare, aus Manövern zusammengesetzte Wege erzeugt, die anschließend von einem simulierten Fahrzeug abgefahren werden können.

(P.1) Das Planungsmodul muss die Möglichkeit bieten, ein Fahrzeug aus einer entsprechenden xml-Datei zu laden und dazu eine *Fahrzeugbeschreibungskette* (*EZauto*) zu erzeugen. Dabei kann auf die bereits vorhandenen Funktionalitäten der Anwendung *EZLeitstand* zurückgegriffen werden.

(P.2) Das Planungsmodul muss dem Benutzer die Möglichkeit bieten, einen Polygonzug, also aneinanderhängende Geraden, mit einer Richtungsangabe festzulegen.

(P.3) Das Planungsmodul muss einen Algorithmus bereitstellen, der zu einem Polygonzug und einer Fahrzeugbeschreibungskette mit Hilfe der Manöverdatenstruktur einen aus Manövern zusammengesetzten Weg erzeugt. Anforderungen an den Algorithmus sind in Abschnitt 3.6 aufgeführt.

(P.4) Das Planungsmodul muss den Pfad eines berechneten Wegs grafisch anzeigen lassen können.

(P.5) Das Planungsmodul muss die Hülle eines berechneten Wegs grafisch anzeigen lassen können.

(P.6) Das Planungsmodul muss die Informationen bereitstellen, die es erlauben, den berechneten Weg von einem simulierten Fahrzeug abfahren lassen zu können.

3.6 Algorithmus des Planungsmoduls

(P.A.1) Der Algorithmus muss die im vorgegebenen Polygonzug vorhandenen Winkel berechnen.

(P.A.2) Der Algorithmus muss zu den berechneten Winkeln mit Hilfe der in Abschnitt 3.4 definierten Manöverdatenstruktur entsprechende *CC Turns* erzeugen, deren Ausrichtungsänderung δ dem jeweiligen Winkel entspricht.

(P.A.3) Der Algorithmus muss die Stellen des Polygonzugs berechnen können, an denen ein *CC Turn* beginnt oder endet. Falls sich zwei *CC Turns* überschneiden oder ein *CC Turn* am Anfang oder Ende des Polygonzugs über diesen hinausragt, soll der Algorithmus abgebrochen werden und dem Benutzer angezeigt werden, dass zu dem vorgegebenen Polygonzug kein Weg berechnet werden kann.

(P.A.4) Der Algorithmus muss zu den am Anfang und Ende sowie zwischen den *CC Turns* vorhandenen Geradenstücken des Polygonzugs mit Hilfe der in Abschnitt 3.4 definierten Manöverdatenstruktur entsprechende Geradenstücke erzeugen. Die Ausrichtung des Fahrzeugs am Anfang und Ende entspricht damit der des ersten bzw. letzten Polygonzugabschnitts unter Berücksichtigung der vorgegebenen Richtung.

(P.A.E) Der Algorithmus muss über eine Erweiterung der in Abschnitt 3.4 definierten Manöverdatenstruktur verfügen, die es ermöglicht, aus Manövern zusammengesetzte Wege zu verwalten.

(P.A.E.1) Die Erweiterung der Datenstruktur muss die Möglichkeit bieten, für einen gemäß (P.A.E) aus Manövern zusammengesetzten Weg aus den gemäß (M.6) zur Verfügung gestellten Pfaden der einzelnen Manöver einen zusammengesetzten Pfad zu erzeugen.

(P.A.E.2) Die Erweiterung der Datenstruktur muss die Möglichkeit bieten, für einen

gemäß (P.A.E) aus Manövern zusammengesetzten Weg aus den gemäß (M.7) berechenbaren Hüllen der einzelnen Manöver eine zusammengesetzte Hülle zu errechnen.

(P.A.5) Der Algorithmus muss aus den gemäß (P.A.2) berechneten *CC Turns* und den gemäß (P.A.4) berechneten Geradenstücken mit Hilfe der in (P.A.E) definierten Erweiterung der Manöverdatenstruktur einen zusammengesetzten Weg erzeugen.

4 Manöverdatenstruktur

Die in diesem Kapitel behandelte Manöverdatenstruktur ist Teil der Software, die im Rahmen dieser Bachelorarbeit entwickelt werden soll. Sie realisiert die Planungsmethode von Fraichard und Scheuer (siehe Abschnitte 2.1 und 2.2). Darüber hinaus stellt sie die Grundlage für die Wegplanung des Planungsmoduls (Kapitel 5) dar, das ebenfalls im Rahmen dieser Arbeit entwickelt werden soll.

In den folgenden Abschnitten wird der Entwurf der Manöverdatenstruktur zunächst abstrakt betrachtet, um den Bezug zu den in Kapitel 3 definierten Anforderungen herzustellen. Im weiteren Verlauf erfolgt eine genauere Betrachtung der einzelnen Bestandteile und Funktionalitäten.

4.1 Grundlegende Idee zum Entwurf von Manövern

Wie in Abschnitt 2.1 erläutert, besitzt ein Manöver einen Pfad und eine Hülle. Als Manöver für diese Manöverdatenstruktur sind Geradeausfahrten und Kurvenmanöver (*CC Turns*) vorgesehen (siehe Abbildung 4.1).

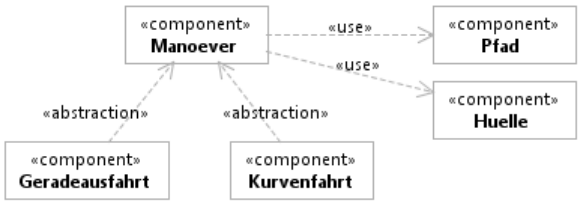

Abbildung 4.1: Abstrakter Entwurf eines Manövers.

Ein Manöver soll zunächst nicht-transformiert sein, d.h. die Berechnung findet unter bestimmten Einschränkungen statt. So wird beispielsweise davon ausgegangen, dass das Manöver im Ursprung des Koordinatensystems beginnt. Die Einschränkungen können im weiteren Verlauf der Wegplanung durch entsprechende Transformation des berechneten Manövers korrigiert werden. Damit wird das Manöver also unter anderem an die gewünschte Stelle verschoben. Die Transformation wird realisiert durch transformierte Manöver. Diese beinhalten ein nicht-transformiertes Manöver sowie bestimmte Transformationsinformationen. In Abschnitt 4.4 werden die verschiedenen Transformationsinformationen näher beschrieben. Ein transformiertes Manöver verfügt über einen transformierten Pfad sowie eine transformierte Hülle (siehe Abbildung 4.2).

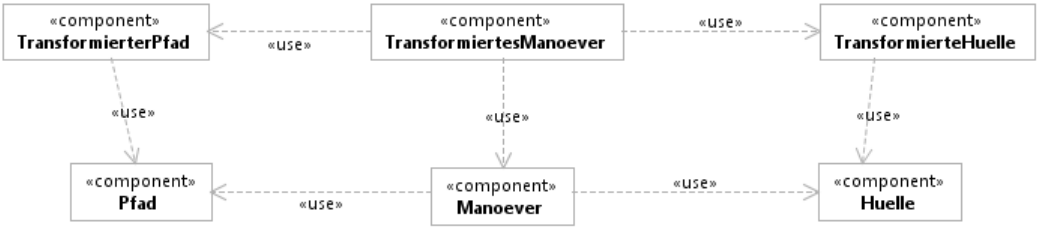

Abbildung 4.2: Abstrakter Entwurf eines transformierten Manövers.

Zur Umsetzung der Wegplanung setzen sich schließlich mehrere transformierte Manöver zu einem Weg zusammen. Ein Weg verfügt dabei über einen zusammengesetzten Pfad sowie eine zusammengesetzte Hülle (siehe Abbildung 4.3).

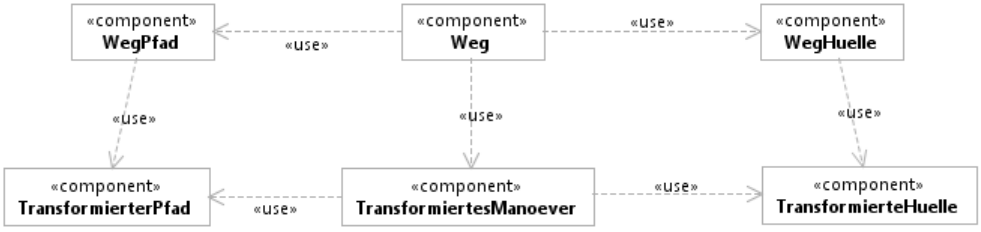

Abbildung 4.3: Abstrakter Entwurf eines Weges.

4.2 Modellierung von Manövern und Wegen

Da Geradeausfahrten, Kurvenmanöver und auch transformierte Manöver gleichermaßen Manöver darstellen, und auch weitere Manövertypen denkbar sind, wird eine Schnittstelle für Manöver bereitgestellt. Diese Schnittstelle (*sManoever*) wird von Klassen für Geradeausfahrten (*Geradeausfahrt*), Kurvenmanöver (*Turn*) und transformierte Manöver (*TransformiertesManoever*) implementiert (siehe Abbildung 4.4). Ein *TransformiertesManoever* verfügt dabei, wie im vorherigen Abschnitt erwähnt, über ein Attribut vom Typ *sManoever*. Die Klasse für Wege heißt *CCWeg* und nicht etwa *Weg*, da eine Klasse mit diesem Namen bereits in der Softwarebibliothek *EZ-auto* (siehe Abschnitt 2.3) existiert. Ein Weg besteht aus mehreren transformierten Manövern.

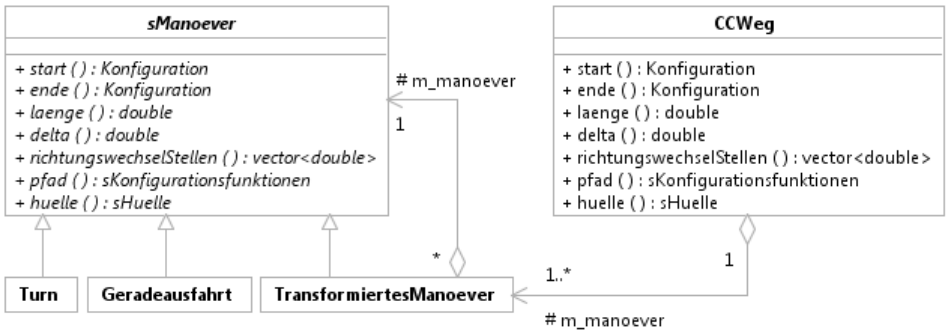

Abbildung 4.4: Schnittstelle *sManoever*.

Jedes Manöver und jeder Weg bietet Zugriff auf folgende Attribute oder Eigenschaften:

Start (*Konfiguration*): Die Konfiguration am Anfang des Manövers oder Weges.

Ende (*Konfiguration*): Die Konfiguration am Ende des Manövers oder Weges.

Länge (*double*): Die Länge des Manövers oder Weges.

Delta (*double*): Die Ausrichtungsänderung, die das Manöver oder der Weg bewirkt.

Richtungswechselstellen (*vector<double>*): Die Stellen des Pfades, an denen das

Fahrzeug die Fahrtrichtung wechselt. Ein Richtungswechsel kann bei einem *CC Turn* auftreten, wenn das Kreisbogenstück rückwärts gefahren wird, beim Weg, wenn zwei hintereinander liegende Manöver nicht dieselbe Fahrtrichtung haben. Die Stelle wird angegeben durch einen *double*-Wert, der als Parameter für die Funktionsklassen des Pfades dient.

Pfad (*sKonfigurationsfunktionen*): Der Pfad des Manövers oder Weges.

Hülle (*sHuelle*): Die Hülle des Manövers oder Weges.

Die Konfiguration beschreibt eine bestimmte Stelle des Manövers bzw. des Pfades. Sie setzt sich zusammen aus der Lage des Fahrzeugs (Position und Ausrichtung) und der Krümmung der Fahrkurve an der entsprechenden Stelle.

Erst im Verlauf der Implementierung hat sich ergeben, dass der Pfad und die Hülle eines Weges (*CCWeg*) jeweils denselben Typ haben wie der Pfad und die Hülle der Schnittstelle *sManoever* und der Klassen, die diese Schnittstelle implementieren. Da *sManoever* und *CCWeg* letztlich Zugriff auf die gleichen Attribute bieten (siehe Abbildung 4.4) ist es denkbar, eine gemeinsame Schnittstelle einzuführen (mehr dazu in Abschnitt 4.8.2).

Jeder *Geradeausfahrt* und jedem *Turn* ist ein Attribut des Typs *Fahrzeugbeschreibungskette* (aus *EZauto*) zugewiesen (siehe Abbildung 4.5). Diese stellt die strukturelle Beschreibung eines Fahrzeugs dar und liefert verschiedene Werte, die für die Berechnung des Manövers wichtig sind, wie den maximalen Lenkwinkel oder den Radstand. Beide Manöver-Klassen bieten jeweils Funktionen zur Initialisierung des Manövers mit bestimmten Parametern sowie zur Berechnung des Manövers, des Pfades und der Hülle. Zudem bieten sie, wie jede Klasse, die *sManoever* implementiert, Zugriff auf die oben genannten Attribute. Darüber hinaus bietet die Klasse *Turn* unter anderem die Möglichkeit, die Länge der einzelnen Klothoiden, des Kreisbogenstücks und der Konstruktionsgeraden abzufragen. Die Konstruktionsgeraden entsprechen den Tangenten am Anfang und am Ende der Fahrkurve eines Kurvenmanövers. Der Abstand zwischen dem Startpunkt bzw. dem Endpunkt des Manövers und dem Schnittpunkt

dieser Konstruktionsgeraden wird hier als deren Länge bezeichnet. Diese ist bei der Wegplanung des im Rahmen dieser Arbeit zu entwickelnden Planungsmoduls von Bedeutung.

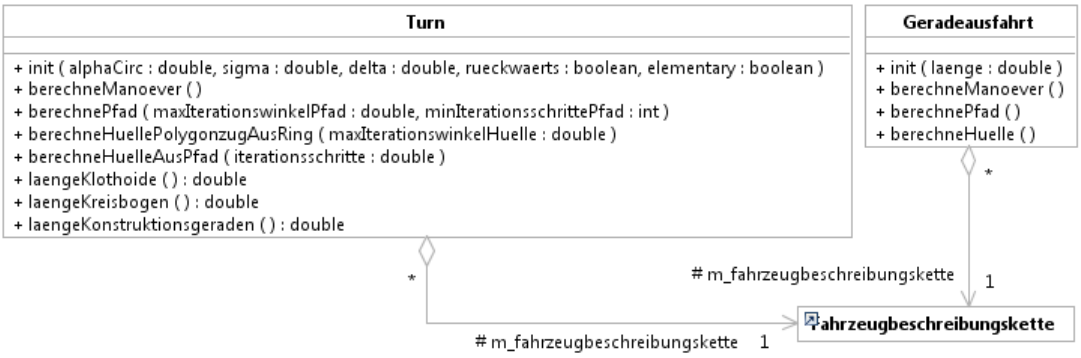

Abbildung 4.5: Klassen *Geradeausfahrt* und *Turn*.

4.3 Modellierung von Pfad und Hülle

Zur Realisierung des Pfades wird auf die Funktionsklassen zurückgegriffen, die in der Themengruppe Mathematik der Softwarebibliothek *EZauto* bereits implementiert sind. Es existieren Schnittstellen sowohl für reelle als auch für planare Funktionen (siehe Abbildung 4.6).

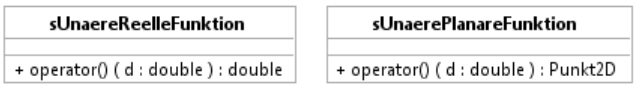

Abbildung 4.6: Schnittstellen für Funktionsklassen.

Diese stellen die Funktion *operator()* bereit, über die per Angabe eines Parameters d (Typ double) der Wert der Funktion (vom Typ double bzw. Punkt2D) an der entsprechenden Stelle des Definitionsbereichs abgefragt werden kann. Jede Funktion verfügt dabei über ein Definitionsminimum und -maximum. Im Sinne einer einheitlichen, intuitiven Gestaltung wird der Definitionsbereich der für den Pfad verwendeten Funktionen durch 0.0 (Start des Manövers) und 1.0 (Ende des Manövers) begrenzt.

Der Pfad eines Manövers soll im Wesentlichen Informationen zur Position des Fahrzeugs bereitstellen. Dies wäre also mit einer planaren Funktion realisierbar. Allerdings existiert an anderer Stelle innerhalb der Softwarebibliothek *EZauto* ein Datentyp für Funktionsklassen, der speziell für die Darstellung von Pfaden ausgelegt ist. Die Schnittstelle heißt *sKonfigurationsfunktionen*, die implementierte Klasse *Konfigurationsfunktionen*. Diese vereinen Funktionen für Position, Lenkwinkel und Ausrichtung des Fahrzeugs (siehe Abbildung 4.7) und bieten sich daher speziell zur Realisierung des Pfades an.

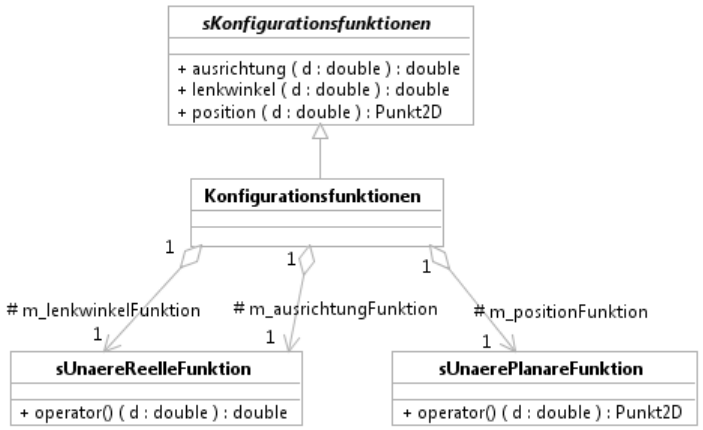

Abbildung 4.7: Schnittstelle *sKonfigurationsfunktionen*.

Die Hülle eines Manövers wird durch die dafür entworfene Schnittstelle *sHuelle*, sowie die implementierende Klasse *Huelle* realisiert (siehe Abbildung 4.8). Diese verwaltet intern ein *Polygon2D* (ebenfalls bereits in *EZauto* enthalten), das wiederum eine Folge von Punkten enthält, die zusammen eine Fläche repräsentieren.

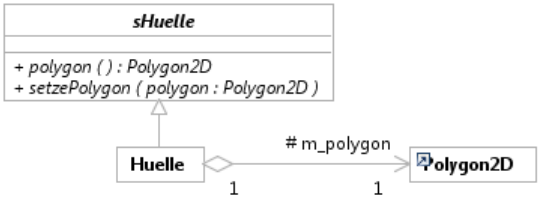

Abbildung 4.8: Schnittstelle *sHuelle*.

Geradeausfahrt, *Turn*, *TransformiertesManoever* und *CCWeg* besitzen also jeweils ein Attribut vom Typ *sKonfigurationsfunktionen* für den Pfad (siehe Abbildung 4.9) sowie ein Attribut vom Typ *sHuelle* für die Hülle (siehe Abbildung 4.10). Welche implementierenden Klassen schließlich für den Pfad und dessen Funktionen sowie für die Hülle verwendet werden, ist dem Anhang A zu entnehmen.

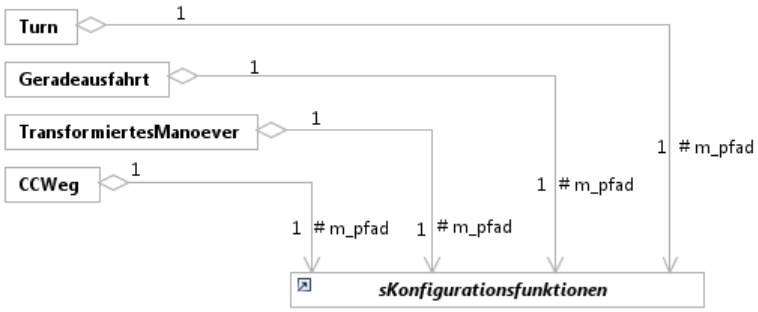

Abbildung 4.9: Pfad als *sKonfigurationsfunktionen*.

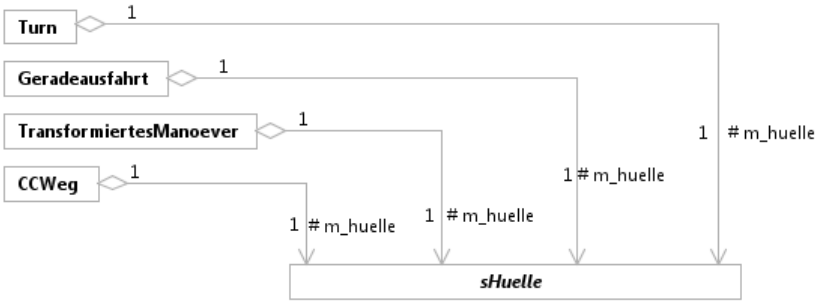

Abbildung 4.10: Hülle als *sHuelle*.

4.4 Transformation eines Manövers

Manöver wie *Geradeausfahrt* und *Turn* sollen, wie in Abschnitt 4.1 beschrieben, zunächst nicht-transformiert sein. Das bedeutet, sie starten im Ursprung des Koordinatensystems mit der Ausrichtung $\theta_s = 0$, werden vorwärts gefahren und sind nach links gerichtet (bei einem Kurvenmanöver wie dem *Turn* handelt es sich also um eine

Linkskurve). Zur Transformation gehören dementsprechend folgende vier Informationen:

Translation (*Punkt2D*): Die Verschiebung im Koordinatensystem entspricht dem Ortsvektor der Start-Position des Fahrzeugs.

Rotation (*double*): Die Rotation des Manövers beschreibt die Start-Ausrichtung des Fahrzeugs.

Rückwärts (*bool*): Dieser Wahrheitswert legt fest, ob das Manöver vorwärts oder rückwärts gefahren werden soll. Rückwärts bedeutet hierbei, dass das Fahrzeug rückwärts vom Ende zum Anfang des ursprünglichen Manövers fährt. Die Pfad-Funktionen werden dazu entsprechend umgekehrt.

Rechts (*bool*): Dieser Wahrheitswert gibt an, ob das Manöver an der x-Achse gespiegelt wird. Rechts bedeutet, dass aus einer ursprünglichen Linkskurve (nicht-transformiert) eine Rechtskurve wird. Auch andere Manöver sind entsprechend nach rechts (statt nach links) gerichtet. Auf die Geradeausfahrt hat dieser Wert hingegen keine Auswirkung.

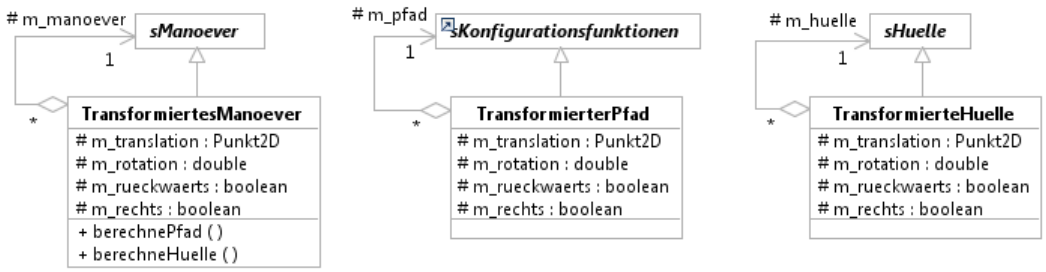

Abbildung 4.11: Klassen für transformierte Manöver.

Wie in Abschnitt 4.1 beschrieben, ist jedem transformierten Manöver ein transformierter Pfad und eine transformierte Hülle zugewiesen. Diese verfügen ebenfalls über die oben genannten Transformationsinformationen (siehe Abbildung 4.11). Da Pfad und Hülle des transformierten Manövers als Attribute des Typs *sKonfigurationsfunktionen* bzw. *sHuelle* definiert sind (siehe Abbildungen 4.9 und 4.10), implementieren

TransformierterPfad und *TransformierteHuelle* eben diese Schnittstellen. Zudem besitzen beide jeweils ein Attribut des entsprechenden Typs, das den Pfad bzw. die Hülle des zugehörigen nicht-transformierten Manövers repräsentiert.

4.5 Spezielle Funktionsklassen für den Pfad

In der Softwarebibliothek *EZauto* bereits enthalten sind unter anderem folgende Funktionsklassen (siehe Abbildung 4.12):

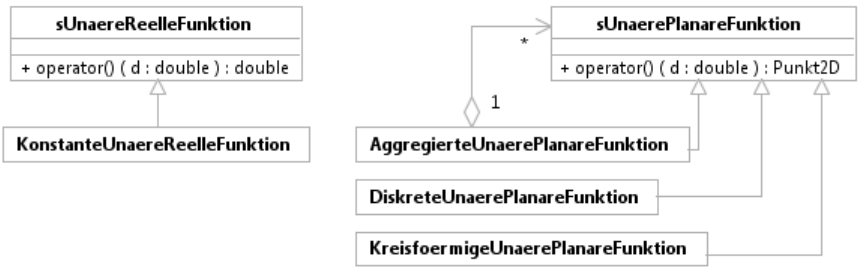

Abbildung 4.12: Vorhandene Funktionsklassen.

KonstanteUnaereReelleFunktion: Reelle Funktion, die für alle Werte des Definitionsbereichs denselben Funktionswert zurückliefert.

KreisfoermigeUnaerePlanareFunktion: Planare Funktion, die einen Kreisbogen beschreibt und über Mittelpunkt, Radius, Start- und Endwinkel des Kreisbogens definiert wird.

DiskreteUnaerePlanareFunktion: Planare Funktion, die durch beliebig viele Paare von Definitionswert und Funktionswert definiert wird und dazwischen liegende (undefinierte) Werte durch verschiedene Interpolationsverfahren bestimmen kann.

AggregierteUnaerePlanareFunktion: Planare Funktion, die aus mehreren (potenziell unterschiedlichen) planaren Funktionen zusammengesetzt ist.

Diese Funktionsklassen eignen sich nicht für alle Funktionen der Pfade der verschiedenen Manövertypen. Im Folgenden werden daher speziell für den Pfad entworfene Funktionsklassen vorgestellt, die teilweise auch auf die vorhandenen Funktionsklassen zurückgreifen. Diese Funktionsklassen werden entsprechend als *AusrichtungFunktion*, *LenkwinkelFunktion* und *PositionFunktion* des Pfades verwendet (mehr dazu im Anhang A).

Geradeausfahrt

Ausrichtung und Lenkwinkel sind bei der (nicht-transformierten) Geradeausfahrt konstant 0. Daher bietet sich die bereits vorhandene *KonstanteUnaerePlanareFunktion* an. Die x-Koordinate der Position ändert sich linear zur zurückgelegten Strecke, die y-Koordinate ist konstant 0. Hier wird die speziell dafür entworfene *LineareUnaerePlanareFunktion* verwendet (siehe Abbildung 4.13). Diese wird über einen Start- und einen Endpunkt (*Punkt2D*) definiert.

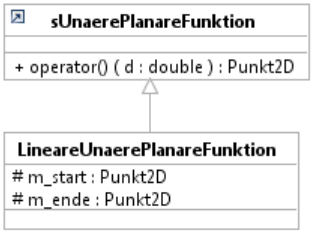

Abbildung 4.13: Funktionsklassen für den Pfad von *Geradeausfahrt*.

Turn

Ausrichtung, Lenkwinkel und Position von *Turn* (*CC Turn*) lassen sich wie in Abschnitt 2.2.2 beschrieben berechnen. Diese Berechnungen sind in den dafür entworfenen Funktionsklassen implementiert (siehe Abbildung 4.14): *TurnAusrichtungFunktion*, *TurnLenkwinkelFunktion* und *TurnPositionFunktion*.

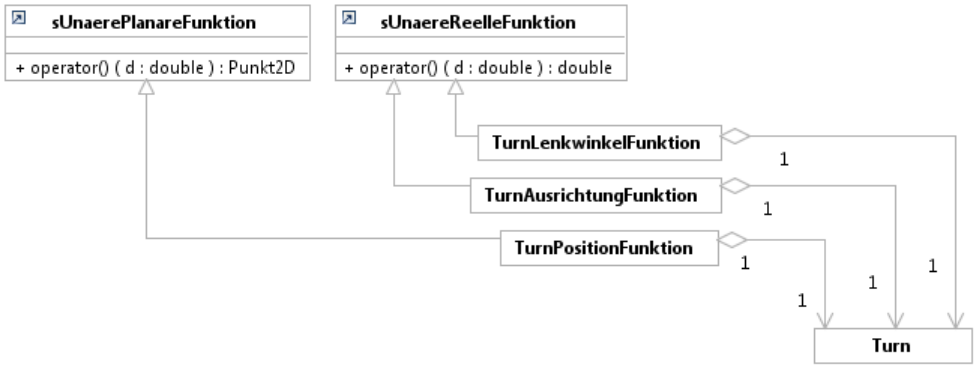

Abbildung 4.14: Funktionsklassen für den Pfad von *Turn*.

TransformiertesManoever

Ausrichtung, Lenkwinkel und Position von *TransformiertesManoever* lassen sich anhand der Transformationsinformationen (siehe Abschnitt 4.4) aus den Funktionen des zugehörigen nicht-transformierten Pfades berechnen. Diese Berechnungen sind in den dafür entworfenen Funktionsklassen implementiert (siehe Abbildung 4.15): *TransformierteAusrichtungFunktion*, *TransformierteLenkwinkelFunktion* und *TransformiertePositionFunktion*.

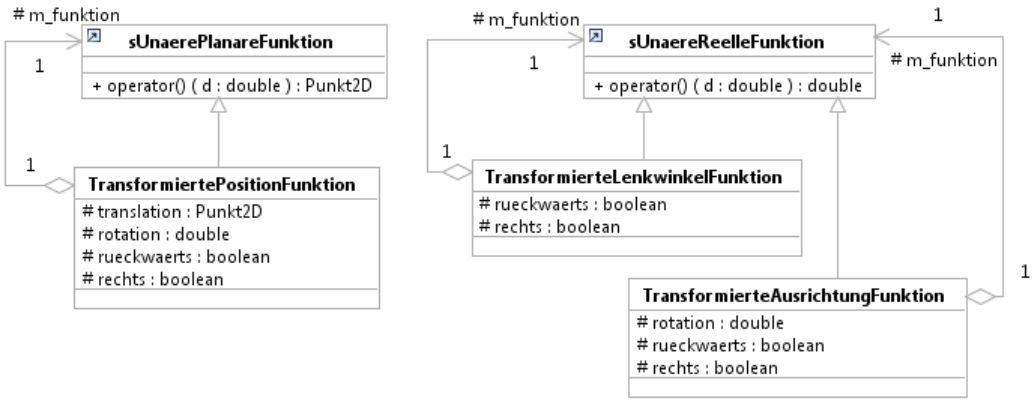

Abbildung 4.15: Funktionsklassen für den Pfad von *TransformiertesManoever*.

CCWeg

Ausrichtung, Lenkwinkel und Position von *CCWeg* ergeben sich aus den enthaltenen transformierten Manövern. Dazu wird zum übergebenen Parameter das entsprechende Manöver gewählt und der Parameter für dieses Manöver entsprechend umgerechnet. Diese Berechnungen sind in den dafür entworfenen Funktionsklassen implementiert (siehe Abbildung 4.16): *WegAusrichtungFunktion*, *WegLenkwinkelFunktion* und *WegPositionFunktion*.

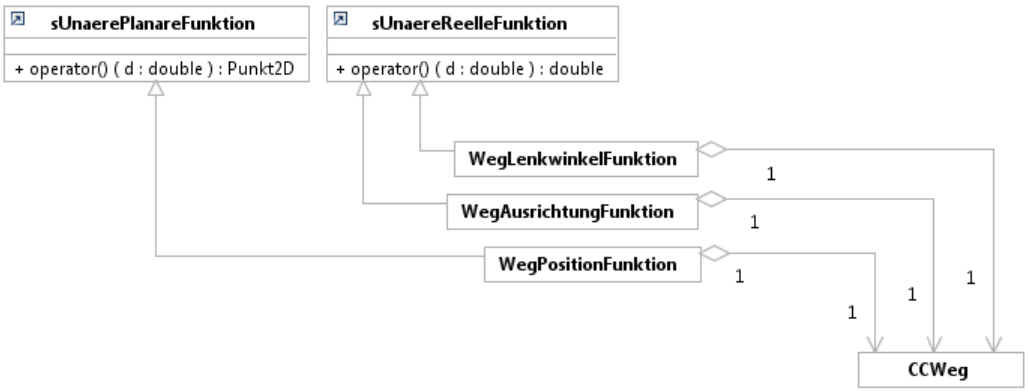

Abbildung 4.16: Funktionsklassen für den Pfad von *CCWeg*.

4.6 Algorithmus zur Berechnung eines CCTurns

Da bei der Berechnung eines *CC Turns*, insbesondere im Spezialfall *Elementary Path* (siehe S. 16), eine bestimmte Reihenfolge bei der Berechnung der einzelnen Teilergebnisse eingehalten werden muss, wird an dieser Stelle ein entsprechender Algorithmus vorgestellt.

Eingabeparameter:

- Start-Konfiguration $q_s = (0, 0, 0, 0)$
- Radstand L_1 (aus Fahrzeugbeschreibungskette)

- Lenkwinkel des Kreisbogens α_{max} (muss kleiner sein als der maximal mögliche Lenkwinkel des Fahrzeugs)
- Schärfe der Klothoiden σ (ergibt sich aus der verwendeten Lenkgeschwindigkeit, siehe Gleichung 2.4)
- Ausrichtungsänderung δ im Bogenmaß
- *elementary* (Wahrheitswert): Verwendung des *Elementary Path* (siehe S. 16)
- *rueckwaerts* (Wahrheitswert): *Rückwärtsfahrt des Kreisbogens* (siehe S. 18)

Berechnung:

1. Krümmung des Kreisbogens κ_{max} (Gleichung 2.3)
2. Ausrichtungsänderung der Klothoiden δ_{min} (Gleichung 2.9)
3. Konfiguration q_i (Gleichung 2.6)
4. *CC Circle*: Ω, r, μ (Gleichungen 2.10–2.12)
5. Konfiguration q_g (siehe S. 15)

Sobald δ_{min} bekannt ist, wird überprüft, ob die Verwendung eines *Elementary Path* oder die *Rückwärtsfahrt des Kreisbogens* möglich und sinnvoll sind. Falls dies für einen Spezialfall nicht zutrifft, wird der entsprechende Wahrheitswert auf *false* gesetzt, andernfalls bleibt er unverändert:

- $\delta > \delta_{min} \Rightarrow \mathit{elementary} = \mathit{false}$
- $|\delta - \delta_{min}| < \pi$ oder $\mathit{elementary} = \mathit{true} \Rightarrow \mathit{rueckwaerts} = \mathit{false}$

Für den Fall $\mathit{elementary} = \mathit{false}$ wird schließlich q_j berechnet (siehe S. 15).

Für den Fall $\mathit{elementary} = \mathit{true}$ werden folgende Berechnungen durchgeführt:

1. angepasste Schärfe der Klothoiden σ_{elem} (Gleichung 2.15)
2. angepasste maximale Krümmung κ_{elem} (Gleichung 2.16)

3. entsprechender maximaler Lenkwinkel α_{elem} (Gleichung 2.3)
4. Ausrichtungsänderung der Klothoiden $\delta_{min_{elem}}$ (Gleichung 2.16)
5. Konfiguration $q_i = q_j$ (Gleichung 2.6)
6. *CC Circle*: Ω, r, μ (Gleichungen 2.10–2.12)

4.7 Konzepte zur Berechnung der Hülle eines CCTurns

Die Hülle eines Manövers muss die Fläche, die das Fahrzeug bei diesem Manöver einnimmt, komplett umschließen. Zudem soll sie effizient zu berechnen sein und einfach zu realisierende Kollisionstests ermöglichen. In der Arbeitsgruppe Echtzeitsysteme hat sich bei Kurvenmanövern für Fahrzeuggespanne die Ringhülle bewährt. Diese bietet sich aufgrund ihrer Form auch für *CC Turns* an. Im Hinblick auf einfach zu realisierende Kollisionstests ist die Verwendung eines Polygons jedoch besser geeignet. Innerhalb der Softwarebibliothek *EZauto* existiert mit der Klasse *Polygon2D* bereits ein Datentyp für Polygone. Die Hülle soll daher als eine durch ein Polygon angenäherte Ringhülle realisiert werden. Diese lässt sich unabhängig vom Pfad berechnen.

Im weiteren Verlauf der Bachelorarbeit wurde klar, dass die Ringhülle verhältnismäßig viel Platz einnimmt (siehe Abschnitt 6.2). Daher wurde noch ein weiteres Konzept erarbeitet. Bei diesem zweiten Konzept wird die Hülle mit Hilfe des Pfades berechnet. Im Folgenden werden beide Konzepte vorgestellt. Eine Gegenüberstellung und Bewertung folgt in Abschnitt 6.2.

4.7.1 Ringhülle

Bevor die Ringhülle berechnet werden kann, müssen zunächst die Extrempunkte des Fahrzeugs ermittelt werden. Dabei handelt es sich um die Punkte, die beim Manöver am weitesten innen oder außen liegen (bezogen auf den Mittelpunkt Ω). Außerdem

muss ermittelt werden, welchen Winkel das Manöver um Ω einnimmt. Dazu werden ebenfalls die entsprechenden Extrempunkte des Fahrzeugs identifiziert. Letztlich definieren folgende Werte die Ringhülle (siehe dazu Abbildung 4.17):

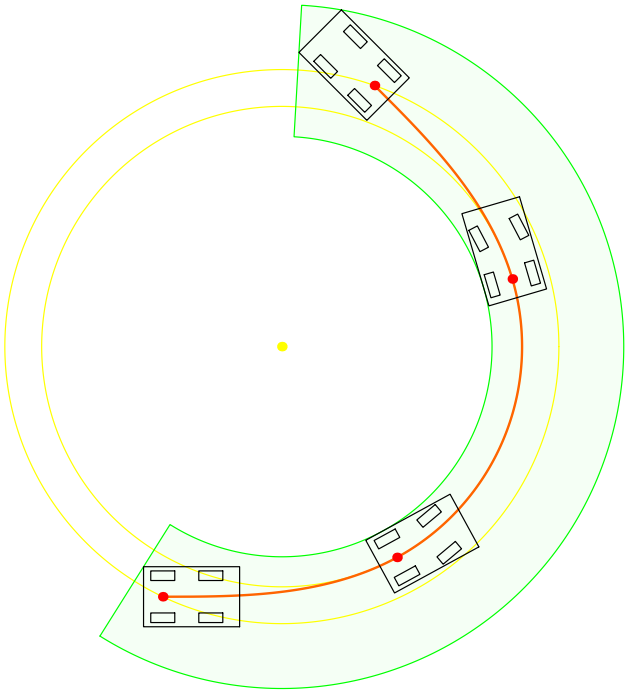

Abbildung 4.17: Veranschaulichung der Ringhülle.

Mittelpunkt des Rings Als Mittelpunkt der Ringhülle wird der Mittelpunkt Ω des *CC Circles* verwendet.

Radius des inneren Kreisbogens: Der innere Kreisbogen der Ringhülle begrenzt den *CC Turn* nach innen. Das Fahrzeug befindet sich bei der Fahrt auf dem Kreisbogen am weitesten innen. Der Mittelpunkt der hinteren Achse stellt den Bezugspunkt dar, der die berechnete Fahrkurve abfährt. Das linke hintere Rad ist dabei dem Mittelpunkt am nächsten (der nicht-transformierte *CC Turn* ist eine Linkskurve, s. Abschnitt 4.4). Der Radius des inneren Kreisbogens entspricht also $\kappa_{max}^{-1} -$ Fahrzeugbreite/2. Falls dieser Wert kleiner ist als 0, wird der Radius auf 0 gesetzt.

Radius des äußeren Kreisbogens: Nach außen wird der *CC Turn* durch den äußeren Kreisbogen begrenzt. Das Fahrzeug ist entweder am Start oder am Ende des Manövers am weitesten außen. Der am weitesten vom Mittelpunkt Ω entfernte Punkt ist dabei am Start die hintere und am Ende die vordere rechte Ecke des Fahrzeugs. Der Radius des äußeren Kreisbogens entspricht also dem Maximum der Entfernung des jeweiligen Eckpunkts zum Mittelpunkt.

Startwinkel: Der Winkel, an dem der Ring beginnt, entspricht der Ausrichtung der Geraden durch den Mittelpunkt Ω und die linke hintere Ecke des Fahrzeugs am Startpunkt (x_s, y_s) des Manövers. Im Spezialfall *Rückwärtsfahrt des Kreisbogens* kann es vorkommen, dass die Punkte (x_i, y_i) und (x_j, y_j) hinsichtlich des Winkels zum Mittelpunkt Ω weiter auseinander liegen als der Start und das Ende des Manövers. In diesem Fall entspricht der Startwinkel der Ausrichtung der Geraden durch den Mittelpunkt Ω und die linke hintere Ecke des Fahrzeugs am Punkt (x_j, y_j).

Endwinkel: Der Winkel, an dem der Ring endet, entspricht der Ausrichtung der Geraden durch den Mittelpunkt Ω und die linke vordere Ecke des Fahrzeugs am Endpunkt (x_g, y_g) des Manövers. Im Spezialfall *Rückwärtsfahrt des Kreisbogens* mit oben genannter Auswirkung entspricht der Endwinkel der Ausrichtung der Geraden durch den Mittelpunkt Ω und die linke vordere Ecke des Fahrzeugs am Punkt (x_i, y_i). Falls eine Schleife gefahren wird oder der Endwinkel bei einer Ausrichtungsänderung $\delta > \pi$ größer ist als der Startwinkel, wird der Endwinkel mit dem Startwinkel gleichgesetzt. In diesem Fall bildet die Hülle einen vollständigen Ring. Der Gesamtwinkel des Rings ist die Differenz aus End- und Startwinkel. Falls diese gleich sind, ist der Gesamtwinkel 2π.

Iterationswinkel: Der Iterationswinkel ist ein Bruchteil des Gesamtwinkels des Rings. Er gibt letztlich an, wie genau die Ringhülle durch den Polygonzug angenähert wird.

Zur Berechnung des Polygonzugs werden nacheinander zuerst der äußere, dann der innere Kreisbogen des Rings betrachtet. Der äußere Kreisbogen wird durch Tangenten angenähert, der innere durch Sekanten (siehe Abbildung 4.18). Die jeweiligen Schnittpunkte der Tangenten bzw. Sekanten bilden den Polygonzug. Da dies eine Annäherung von außen ist, umschließt der resultierende Polygonzug in jedem Fall das Fahrzeug. Damit handelt es sich um eine gültige Hülle. Die erste Tangente des äußeren Kreisbogens befindet sich an dem Punkt, an dem der Kreisbogen mit der seitlichen Begradigung abschließt. Die weiteren Tangenten folgen jeweils nach dem Iterationswinkel. Die Schnittpunkte der Tangenten haben alle denselben Abstand zum Mittelpunkt Ω. Sie lassen sich durch fortlaufende Rotation des ersten Schnittpunktes mit dem Iterationswinkel um Ω berechnen. Die Schnittpunkte der Sekanten liegen auf dem inneren Kreisbogen des Rings. Sie lassen sich analog durch Rotation des ersten Punktes auf dem inneren Kreisbogen berechnen.

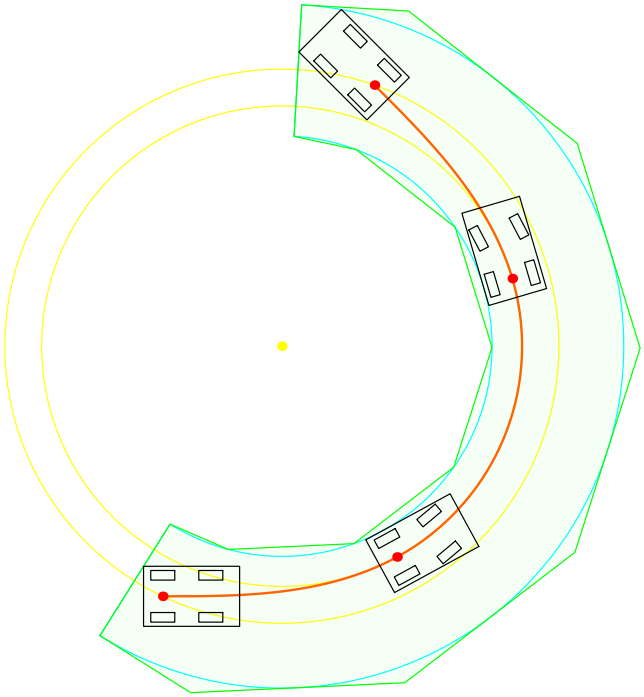

Abbildung 4.18: Annäherung der Ringhülle durch ein Polygon.

4.7.2 Hülle aus Pfad

Bei diesem zweiten Konzept wird die Hülle aus dem Pfad berechnet. Im Gegensatz zur Ringhülle werden dabei keine festen Radien ermittelt, die für das gesamte Manöver gültig sind. Stattdessen werden die Fahrkurven der jeweiligen Extrempunkte betrachtet, also der Punkte des Fahrzeugs, die am weitesten innen oder außen liegen. „Innen“ und „außen“ bezieht sich hierbei nicht auf einen festen Mittelpunkt, damit werden lediglich die Abstände zur Fahrkurve nach links (innen) und rechts (außen) beschrieben. Wie beim Ansatz der Ringhülle festgestellt, liegt das hintere linke Rad am weitesten innen. Das trifft nicht nur für die Fahrt auf dem Kreisbogen zu, sondern für das gesamte Manöver. Somit begrenzt die Fahrkurve dieses Punktes die Hülle nach innen. Für die äußere Begrenzung der Hülle betrachtet man wie bei der Ringhülle die rechten Eckpunkte des Fahrzeugs. Da sich die Fahrkurven der beiden Eckpunkte unter Umständen überschneiden, also nicht einer der beiden immer außen liegt, müssen beide bei der Berechnung der Hülle berücksichtigt werden.

Für die Berechnung werden möglichst viele Stellen der Fahrkurve betrachtet. Dazu werden vom bereits berechneten Pfad in möglichst kleinen Schritten per Iteration von vorne nach hinten die jeweiligen Positionen und Ausrichtungen ausgelesen. Zu jeder Position werden anhand der Ausrichtung die dazugehörigen Positionen der oben genannten Extrempunkte bestimmt. Die berechneten Positionen der Extrempunkte bilden schließlich das Polygon, das die Hülle darstellt (siehe Abbildung 4.19).

Bei der hier vorgestellten Hülle handelt es sich um eine Annäherung der tatsächlich eingenommen Fläche des Fahrzeugs. Da die Annäherung von „innen“ erfolgt, ist die resultierende Fläche kleiner als die tatsächlich eingenommene. Die Hülle ist daher streng genommen nicht gültig. Bei entsprechend hoher Iteration ist der Unterschied allerdings marginal. Daher wurde dieses Konzept als Alternative zur Ringhülle beibehalten. Genaueres dazu ist dem Vergleich der Hüllenkonzepte in Abschnitt 6.2 zu entnehmen.

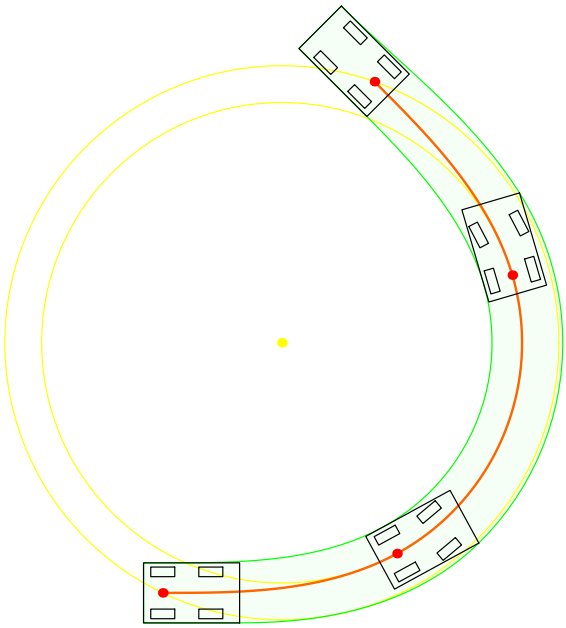

Abbildung 4.19: Veranschaulichung der aus dem Pfad berechneten Hülle.

4.8 Implementierung und Erweiterbarkeit

In diesem letzten Abschnitt soll auf Besonderheiten bei der Implementierung eingegangen werden. Abschließend werden Vorschläge zur Erweiterung und Weiterentwicklung der Manöverdatenstruktur vorgestellt.

4.8.1 Besonderheiten bei der Implementierung

Berechnung der Fresnel-Integrale bei *CC Turns*

Bei der Berechnung eines *CC Turns* müssen sowohl bei der Berechnung der Klothoiden (Gleichung 2.6) als auch bei der Berechnung der Schärfe der Klothoiden im Spezialfall *Elementary Path* (Gleichung 2.15) Fresnel-Integrale berechnet werden (Gleichungen 2.7, 2.8). Diese lassen sich nur numerisch berechnen. Bei der Implementierung wurde daher eine Annäherung über Trapez-Integrale verwendet. Dabei wird

die Funktion, die integriert werden soll, in möglichst viele, gleich große Intervalle auf der x-Achse unterteilt. Die Fläche, die die Funktion in jedem dieser Intervalle einnimmt, wird jeweils durch ein Trapez angenähert. Die Flächen der Trapeze werden berechnet und anschließend summiert. Das Ergebnis stellt schließlich die Annäherung an das tatsächliche Integral dar. Damit die Annäherung möglichst genau ist, wird die Zahl der Intervalle bzw. der Trapeze so lange erhöht, bis sich die resultierende Fläche zur vorherigen Berechnung nur noch marginal ändert. Um die Terminierung zu garantieren, wird dabei die Anzahl der Intervalle beschränkt. Die Implementierung des Verfahrens basiert auf [Press u. a., 2002, Seiten 134-144].

Berechnung der Hülle bei *CC Turns*

Wie in Abschnitt 4.3 beschrieben, wird die Hülle als *Polygon2D* implementiert. Bei der Ringhülle wird direkt ein solches Polygon erzeugt. Bei der Hülle, die aus dem Pfad berechnet wird, wird zusätzlich mit der Klasse *Polygonmenge2D* gearbeitet. Objekte dieser Klasse aggregieren mehrere Polygone. Außerdem gibt es die Möglichkeit, zwei Objekte des Typs *Polygonmenge2D* miteinander zu vereinigen. Die vereinigte Menge kann anschließend wieder in ein *Polygon2D* umgewandelt werden[1]. Dieser Vorgang wird bei der *Hülle aus Pfad* durchgeführt. Die Hülle wird, wie in Abschnitt 4.7.2 angesprochen, iterativ vom Start bis zum Ende des Manövers aufgebaut. Dazu werden in jedem Schritt die Extrempunkte bestimmt und dazu eine Polygonmenge, bestehend aus einem Polygon gebildet. Die Polygonmenge wird mit der bereits bestehenden Polygonmenge vereinigt und wiederum in ein Polygon bzw. eine Polygonmenge umgewandelt. Nachdem das Ende das Manövers erreicht ist, umfasst das Polygon schließlich die gesamte Fläche. Bei der Zusammensetzung der Hülle eines Weges aus den Hüllen der einzelnen Manöver wird ähnlich vorgegangen (siehe unten).

[1] Die Konvertierung einer *Polygonmenge2D* in ein *Polygon2D* funktioniert nur, wenn die Polygonmenge zusammenhängend ist, also keine disjunkten Teilmengen enthält. Dies ist hier der Fall, da die entsprechende Polygonmenge in jedem Iterationsschritt aus zwei jeweils zusammenhängenden, sich überschneidenden Polygonmengen gebildet wird.

Umrechnungen bei der Transformation eines Manövers

In Abschnitt 4.4 wurden die Transformation sowie die dazu benötigten Transformationsinformationen beschrieben. Bei der Transformation der einzelnen Funktionen des Pfades sind verschiedene Transformationsinformationen von Bedeutung. Bei allen drei Funktionen wird zunächst der Parameter angepasst, falls rückwärts gefahren wird (*rueckwaerts* = *true*). Anschließend wird der Funktionswert des nicht-transformierten Pfades ausgelesen. Falls das Manöver an der x-Achse gespiegelt werden soll (*rechts* = *true*), wird der Funktionswert jeweils invertiert. Der resultierende Wert wird unterschiedlich weiter verarbeitet:

PositionFunktion: Das Zwischenergebnis wird gemäß des Rotationswerts um den Ursprung rotiert und anschließend gemäß der Translationswerte verschoben. Hier werden also alle vier Transformationsinformationen benötigt.

AusrichtungFunktion: Die Rotation wird auf das Zwischenergebnis addiert. Das Ergebnis entspricht dem transformierten Wert. Die Translation hat keinen Einfluss auf die Ausrichtung.

LenkwinkelFunktion: Die Rotation und die Translation haben keinen Einfluss auf den Lenkwinkel. Das Zwischenergebnis entspricht daher dem endgültigen transformierten Wert.

Zur Transformation der Hülle werden die Punkte des dazugehörigen Polygons einzeln transformiert. Falls das Manöver an der x-Achse gespiegelt werden soll (*rechts* = *true*), wird dazu die y-Koordinate des jeweiligen Punktes invertiert. Der Punkt wird nun gemäß des Rotationswertes um den Ursprung rotiert und anschließend gemäß der Translationswerte verschoben.

Zusammensetzung des Weges

Der Weg ist aus mehreren transformierten Manövern zusammengesetzt (siehe Abschnitt 4.2). Der Pfad des Weges greift dabei auf die Pfade der entsprechenden Ma-

növer zurück. Dabei wird der übergebene Parameter, der zwischen 0 und 1 liegt, zunächst auf die Gesamtlänge des Manövers umgerechnet. Anhand des resultierenden Wertes wird das Manöver bestimmt, in dem die gewünschte Stelle des Weges liegt. Durch Subtraktion der Länge der vorherigen Manöver erhält man den Parameter für das entsprechende Manöver. Anhand der Länge des Manövers wird der Parameter schließlich wieder auf einen Wert zwischen 0 und 1 umgerechnet. Mit dem resultierenden Wert wird der Funktionswert des Pfades abgefragt und schließlich zurückgeliefert.

Die Hülle eines Weges wird aus den Hüllen der einzelnen Manöver zusammengesetzt. Dazu wird ähnlich vorgegangen wie bei der Berechnung der *Hülle aus Pfad*. Zu den Polygonen der Hüllen des ersten und zweiten Manövers wird jeweils eine Polygonmenge erzeugt. Die beiden Polygonmengen werden vereinigt. Nun wird zum Polygon der Hülle des dritten Manövers eine Polygonmenge erzeugt und mit der bereits vorhandenen Polygonmenge vereinigt. Nach diesem Verfahren wird fortgefahren bis die Hüllen aller Manöver vereinigt wurden. Abschließend wird die resultierende Polygonmenge in ein Polygon konvertiert.

4.8.2 Vorschläge zur Weiterentwicklung

Gemeinsame Schnittstelle für Bewegungen

Wie in Abschnitt 4.2 bereits erwähnt, wurde erst im Verlauf der Implementierung klar, dass der Pfad und die Hülle eines Weges (*CCWeg*) dieselbe Schnittstelle haben wie der Pfad und die Hülle der Klassen, die die Schnittstelle *sManoever* implementieren. Da *sManoever* und *CCWeg* letztlich Zugriff auf die gleichen Attribute bieten (siehe Abbildung 4.4) ist es denkbar, eine gemeinsame Schnittstelle einzuführen. Dazu kann man die Schnittstelle *sManoever* umbenennen, beispielsweise in *sBewegung*, und dann die Klasse *CCWeg* von dieser neuen Schnittstelle *sBewegung* ableiten.

Erweiterung um ein effizienteres Hüllenkonzept

In Abschnitt 4.7 wurden zwei Konzepte zur Berechnung der Hülle eines *CC Turns* vorgestellt. Diese beiden Konzepte wurden auch implementiert. Beide haben allerdings kleinere Schwächen (siehe dazu Abschnitt 6.2). Grundsätzlich ist es denkbar, weitere Hüllenkonzepte zu entwerfen und zu implementieren. Falls dies zu späterem Zeitpunkt erwünscht ist, ist die Klasse *Turn* um eine entsprechende Funktion zu erweitern. Diese sollte analog zu den bereits vorhandenen Funktionen zur Hüllenberechnung aufgebaut sein. Die Änderungen sollten dabei nur das Polygon betreffen.

Implementierung weiterer Manövertypen

Falls weitere Manövertypen gewünscht sind, sollte man diese wie die bereits vorhandenen Typen für Geradeausfahrten und *CC Turns* von der gemeinsamen Schnittstelle *sManoever* bzw. einer möglichen neuen Schnittstelle für Bewegungen (s.o.) ableiten. Außerdem sollte das Manöver als nicht-transformiertes Manöver entworfen werden (siehe dazu Abschnitt 4.4). Nach Möglichkeit sollte man für Pfad und Hülle auf die bereits vorhandenen implementierenden Klassen zurückgreifen (siehe Abschnitt 4.3). Für den Pfad kann man analog zu den in Abschnitt 4.5 vorgestellten Funktionsklassen eigene Funktionsklassen entwerfen.

5 Planungsmodul

Das in diesem Kapitel behandelte Planungsmodul ist Teil der Software, die im Rahmen dieser Bachelorarbeit entwickelt werden soll. Es erweitert die vorhandene Anwendung *EZLeitstand* um ein Modul zur Planung von Wegen, die auf der Planungsmethode von Fraichard und Scheuer (siehe Abschnitte 2.1 und 2.2) basieren. Dazu wird die im vorherigen Kapitel vorgestellte Manöverdatenstruktur verwendet.

Das Planungsmodul soll dem Benutzer die Möglichkeit bieten, schrittweise einen Weg zu planen. Dazu soll der Benutzer im ersten Schritt einen Polygonzug mit Richtungsangaben definieren können, der als grober, aber nicht fahrbarer Weg fungiert. Im zweiten Schritt soll er die Parameter für die Wegplanung definieren können. Nach der Berechnung des Weges soll dieser als Kombination aus Pfad und Hülle auf der Zeichenfläche der Anwendung *EZLeitstand* dargestellt werden. Der Weg soll letztlich alle Informationen bereitstellen, die ein Fahrzeug zum Fahren dieses Weges benötigt. Dies dient dem Ziel, die Funktionalität des Fahrens bei einer späteren Weiterentwicklung des Planungsmoduls zu realisieren. Das schrittweise Vorgehen bei der Planung von Wegen ist im Anhang B veranschaulicht.

5.1 Aufbau des Planungsmoduls

Das Planungsmodul wird als Anwendungsmodul *WegPlanungModul* mit der dazugehörigen ModulGui *WegPlanungModulGui* realisiert (siehe Abbildung 5.1). Zur Einordnung in die Struktur der Anwendung *EZLeitstand* sei an dieser Stelle auf den

Abschnitt 2.4 und insbesondere auf die darin enthaltenen Klassendiagramme (Abbildungen 2.8 und 2.9) verwiesen.

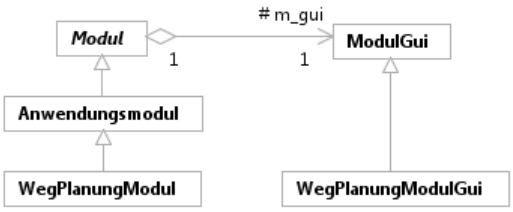

Abbildung 5.1: Modellierung des Planungsmoduls.

Wie jedes Anwendungsmodul bietet das Planungsmodul die Möglichkeit, verschiedene Ereignisse zu verarbeiten und eigene visualisierbare Objekte auf die Zeichenfläche der Anwendung *EZLeitstand* zu zeichnen. Über die Ereignisbehandlung ist auch die Kommunikation mit dem verbundenen Fahrzeug möglich. Allerdings wird davon noch kein Gebrauch gemacht. Bei einer möglichen Weiterentwicklung des Planungsmoduls hinsichtlich der Realisierung des Fahrens wird die ereignisbasierte Kommunikation mit dem Fahrzeug aber von zentraler Bedeutung sein.

5.2 Aufbau und Funktionalitäten der Klasse WegPlanungModul

Die Klasse *WegPlanungModul* verwaltet und berechnet alle Daten, die zur Wegplanung benötigt werden. Außerdem kontrolliert sie die dazugehörige grafische Benutzeroberfläche *WegPlanungModulGui* und damit die Interaktion des Benutzers in den einzelnen Planungsschritten. Im Folgenden werden die wichtigsten Funktionalitäten vorgestellt.

5.2.1 Zustände des Planungsmoduls

Die am Anfang des Kapitels erwähnten Schritte der Wegplanung werden durch folgende Zustände (*WegPlanungModulZustand*) realisiert:

wpPolygonzugBearbeiten: In diesem Zustand hat der Benutzer die Möglichkeit, den Polygonzug zu erstellen oder zu bearbeiten. Das Hinzufügen von Punkten ist über Mausklicks auf die Zeichenfläche möglich. Der Polygonzug wird auf dieser grafisch dargestellt. Die einzelnen Punkte sowie weitere Informationen des Polygonzugs werden in einer Tabelle der entsprechenden GUI *polygonzugGui* (siehe Abschnitt 5.3) festgehalten. Das Planungsmodul befindet sich am Anfang in diesem Zustand und kann jederzeit in diesen Zustand wechseln.

wpWegBearbeiten: In diesem zweiten Zustand bietet das Planungsmodul dem Benutzer die Möglichkeit, die Parameter für die Planung und Berechnung des Weges aus dem vorgegebenen Polygonzug festzulegen. Dazu gehören:

- Lenkwinkel der Kreisbögen der *CC Turns*
- Schärfe der Klothoiden der *CC Turns*
- Verwendung des *Elementary Path* bei *CC Turns*
- Verwendung der *Rückwärtsfahrt des Kreisbogens* bei *CC Turns*
- Iterationen bei der Berechnung des Pfades eines *CC Turns*
- Iterationswinkel bei der Berechnung der Hülle eines *CC Turns* (siehe Abschnitt 4.7.1)
- Verwendung der *Hülle aus Pfad* bei *CC Turns* (siehe Abschnitt 4.7.2)

Die entsprechende GUI, über die der Benutzer diese Parameter festlegen kann, heißt *wegGui* (siehe Abschnitt 5.3). Ein Wechsel vom ersten in diesen zweiten Zustand ist nur möglich, falls der Polygonzug bereits aus mindestens zwei Punkten besteht. Ansonsten kann jederzeit in diesen Zustand gewechselt wer-

den.

wpWegBerechnen: Bei der Berechnung des Weges aus dem Polygonzug befindet sich das Planungsmodul in diesem dritten Zustand. Die Berechnung kann nur im vorherigen Zustand ausgelöst werden. Schlägt die Berechnung fehl, findet ein Wechsel zum vorherigen Zustand statt, ist sie erfolgreich, wechselt das Planungsmodul in den nächsten Zustand *wpFahrtBearbeiten*.

wpFahrtBearbeiten: Dieser vierte Zustand wird bei einer erfolgreichen Berechnung des Weges aus dem Polygonzug erreicht. Dabei werden neben dem Polygonzug auch der berechnete Weg in Form von Pfad und Hülle auf der Zeichenfläche grafisch dargestellt. Der Benutzer hat hier die Möglichkeit, einzelne Elemente ein- und auszublenden. Dies geschieht über die entsprechende GUI *fahrtGui* (siehe Abschnitt 5.3). Der Zustand ist außerdem dafür vorgesehen, Parameter für eine Fahrt festzulegen und diese schließlich zu starten.

wpFahren: Dieser fünfte Zustand ist für die Fahrt des verbundenen Fahrzeugs vorgesehen. Diese ist erst möglich, sobald der Weg berechnet wurde. Dabei soll dem Benutzer lediglich die Möglichkeit gegeben werden, die Fahrt zu unterbrechen oder abzubrechen. Andere Veränderungen durch den Benutzer sollen ausgeschlossen werden. Dieser Zustand wird noch nicht genutzt, da die entsprechende Funktionalität noch nicht existiert (siehe Anfang des Kapitels).

5.2.2 Informationen des Polygonzugs

Befindet sich das Planungsmodul im Zustand *wpPolygonzugBearbeiten*, kann der Polygonzug bearbeitet werden. Dazu werden Punkte hinzugefügt oder entfernt. Dabei wird jeweils eine Richtung festgelegt (vorwärts oder rückwärts). Wenn kein Punkt vorhanden ist, hat der Benutzer die Möglichkeit, die Lage (Position und Ausrichtung) des Fahrzeugs zu übernehmen. Dazu wird die Position des Fahrzeugs als erster Punkt übernommen. Die Linie zum zweiten Punkt richtet sich nach der Ausrichtung

des Fahrzeugs. Der Polygonzug verfügt über folgende Informationen:

Punkte: Die Koordinaten der einzelnen Punkte.

Richtungen: Ab dem zweiten Punkt (bzw. der ersten Linie) wird die eingestellte Fahrtrichtung der Linie festgehalten.

Entfernungen: Ebenfalls ab dem zweiten Punkt wird die Entfernung zum vorherigen Punkt (bzw. die Länge der Linie zwischen den Punkten) berechnet und festgehalten.

Deltas: Ab dem dritten Punkt (bzw. der zweiten Linie) wird die Ausrichtungsänderung zur vorherigen Linie berechnet und festgehalten.

Mit den Informationen des Polygonzugs kann schließlich ein fahrbarer Weg berechnet werden (siehe Abschnitt 5.4).

5.2.3 Verarbeitung von Ereignissen

Da die Klasse *WegPlanungModul* wie jedes Anwendungsmodul die Schnittstelle *sEreignisempfaenger* implementiert (siehe Abschnitt 2.4), kann sie Ereignisse (*sEreignis*) empfangen und verarbeiten. Dabei sind zwei Ereignistypen von Bedeutung:

- *ereignisEmpfangePeriode* (“leitstand/periode”)
- *ereignisEmpfangeDarstellung* (“leitstand/darstellung”)

Das Ereignis *ereignisEmpfangePeriode* wird regelmäßig empfangen und dient der Kommunikation mit dem Fahrzeug (siehe Abschnitt 2.4). Beim Empfang können zum Beispiel die Fahrzeugdaten aktualisiert werden. Dies ist aber erst bei einer Weiterentwicklung des Planungsmoduls, insbesondere bei der Implementation des Fahrens, notwendig. Das Ereignis *ereignisEmpfangeDarstellung* wird von der Klasse *Darstellung* (siehe Abschnitt 2.4) unter anderem bei Mausbewegungen oder -klicks auf der Zeichenfläche gesendet. Das Empfangen dieses Ereignisses erlaubt es dem Planungsmodul, die Eingaben des Benutzers zu verarbeiten. Wenn sich das Planungsmodul

im Zustand *wpPolygonzugBearbeiten* befindet und Punkte hinzugefügt werden sollen, wird bei jeder Mausbewegung der entsprechende Punkt im Koordinatensystem ermittelt. Zu diesem Punkt werden die Daten berechnet, die für den Polygonzug relevant sind. Diese Informationen werden als Hilfsanzeigen auf der Zeichenfläche dargestellt. Falls nun ein Mausklick registriert wird, werden die berechneten Werte für den Polygonzug übernommen.

5.2.4 Zeichnen auf die Zeichenfläche

Wie jedes Anwendungsmodul implementiert die Klasse *WegPlanungModul* die Funktion *zeichne*. Über diese Funktion können die zu zeichnenden Objekte des Planungsmoduls auf die Zeichenfläche der Anwendung *EZLeitstand* gezeichnet werden. Das Planungsmodul zeichnet Folgendes:

Polygonzug: Vor der Berechnung des fahrbaren Weges wird der Polygonzug in jedem Fall gezeichnet (siehe Abbildung 5.2[1]). Nach der Berechnung kann der Benutzer den Polygonzug ein- oder ausblenden.

Pfad: Sobald der Weg berechnet wurde, wird dessen Pfad gezeichnet (siehe Abbildung 5.3). Dieser kann ebenfalls ein- und ausgeblendet werden. Bei eingeblendetem Pfad kann der Benutzer außerdem festlegen, ob die Richtungswechselstellen grafisch hervorgehoben werden.

Hülle: Sobald der Weg berechnet wurde, wird dessen Hülle gezeichnet (siehe Abbildung 5.3). Der Benutzer kann auch die Hülle ein- oder ausblenden.

Eingabehilfen: Im Zustand *wpPolygonzugBearbeiten* werden beim Hinzufügen von Punkten Eingabehilfen angezeigt (siehe Abbildung 5.2). Falls sich die Maus

[1] In dieser und in weiteren Abbildungen ist ein LKW mit Anhänger zu sehen. Es handelt sich dabei um das verbundene Fahrzeug. Das verbundene Fahrzeug wird jeweils über eine eigene Software realisiert. Es existiert allerdings noch keine entsprechende Software für einen Car-like Robot. Im Rahmen dieser Bachelorarbeit ist die Beschäftigung mit dieser Software auch nicht vorgesehen. Daher wird auf die vorhandene Software zurückgegriffen (mehr dazu in Abschnitt 5.5). Unabhängig davon sind die berechneten Wege für einen Car-like Robot gedacht.

auf der Zeichenfläche befindet, wird die Position der Maus als möglicher Punkt für den Polygonzug dargestellt. Dazu wird eine Linie vom letzten Punkt des Polygonzugs zu diesem Punkt gezeichnet. Außerdem werden die Koordinaten des Punktes, die Entfernung zum vorherigen Punkt, die Richtungsangabe sowie die Ausrichtungsänderung angezeigt. Zu den anderen Zuständen gibt es keine Eingabehilfen, da in diesen keine Eingaben über die Zeichenfläche vorgenommen werden.

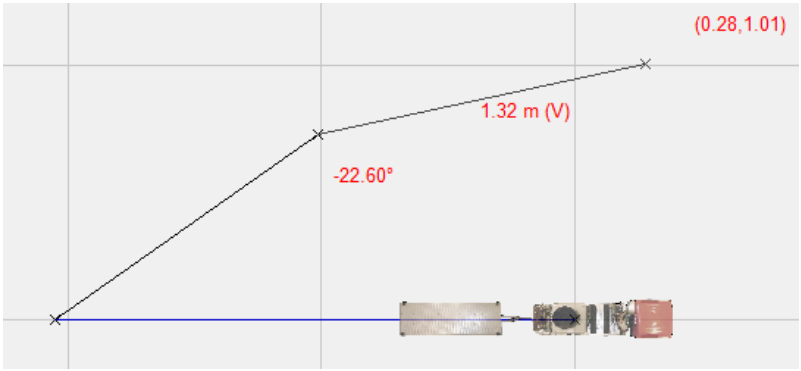

Abbildung 5.2: Polygonzug und Eingabehilfen auf der Zeichenfläche.

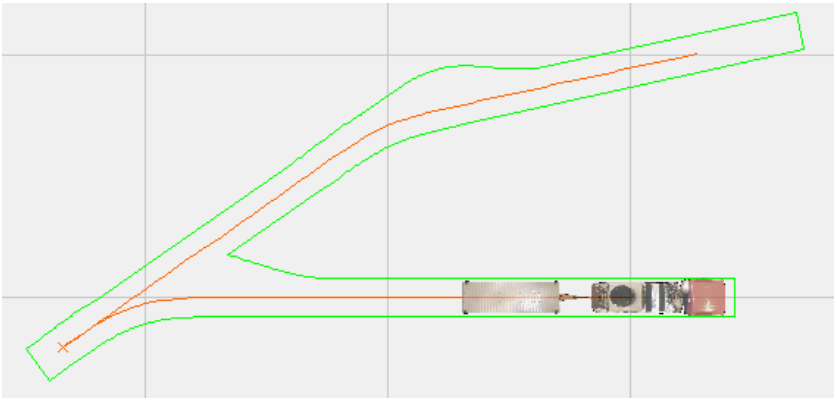

Abbildung 5.3: Pfad und Hülle auf der Zeichenfläche.

5.3 Aufbau und Funktionalitäten der Klasse WegPlanungModulGui

Die Klasse *WegPlanungModulGui* stellt die grafische Benutzeroberfläche des Planungsmoduls dar. Sie ist abgeleitet von *ModulGui* und somit indirekt von *wxPanel*. Sie wird also wie die gesamte Benutzeroberfläche der Anwendung *EZLeitstand* mit *wxWidgets* realisiert (siehe Abschnitt 2.4). Als GUI bietet sie dem Benutzer die Möglichkeit, das Planungsmodul zu bedienen. Dabei greift sie auf die Zustände der Klasse *WegPlanungModul* zu (siehe Abschnitt 5.2), um die Bedienung entsprechend einzuschränken. Dazu werden Teile der GUI deaktiviert. Die GUI hat folgende Bestandteile (jeweils Attribute vom Typ *wxPanel*, siehe Abbildung 5.4):

polygonzugGui: Diese GUI beinhaltet Informationen und Bedienmöglichkeiten zum Bearbeiten des Polygonzugs. Dazu gehören CheckBoxes (*wxCheckBox*), Buttons (*wxButton* und *wxToggleButton*) und eine Tabelle (*wxGrid*) für die Informationen des Polygonzugs. Die GUI ist ausschließlich im Zustand *wpPolygonzugBearbeiten* aktiviert.

wegGui: Über diese GUI lassen sich die Parameter für die Berechnung des Weges aus dem Polygonzug festlegen. Dazu werden Slider mit Textfeldern (*wxSlider* und *wxStaticText*) und CheckBoxes verwendet. Die GUI ist ausschließlich im Zustand *wpWegBearbeiten* aktiviert.

fahrtGui: Diese GUI bietet die Möglichkeit, den Polygonzug sowie Pfad, Richtungswechselstellen und Hülle des berechneten Weges ein- und auszublenden. Dazu werden CheckBoxes verwendet. Sie ist außerdem dafür vorgesehen, Parameter für eine Fahrt des verbundenen Fahrzeugs festzulegen.

Die GUIs sind im Planungsmodul übereinander angeordnet. Für den Wechsel des Zustandes des Planungsmoduls sind zwischen den einzelnen GUIs entsprechende Buttons platziert. Beim Wechsel des Zustandes wird die entsprechende GUI eingeblendet und aktiviert, alle anderen GUIs ausgeblendet und deaktiviert. Die GUIs lassen sich

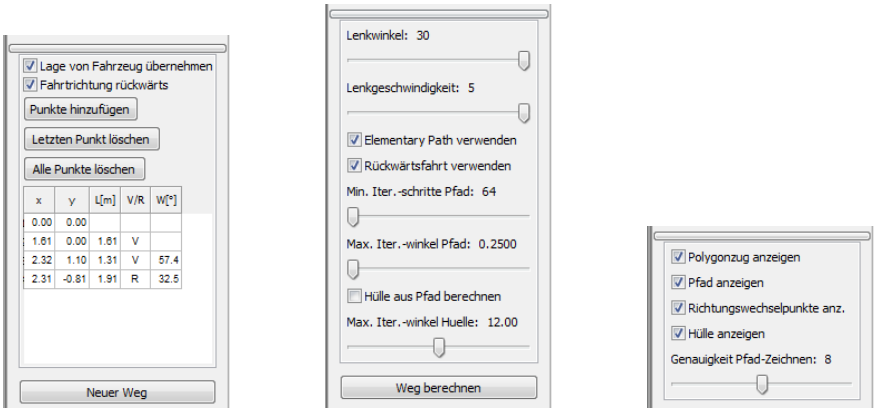

Abbildung 5.4: GUIs des Planungsmoduls: *polygonzugGui*, *wegGui*, *fahrtGui*.

aber auch manuell ein- und ausblenden. Damit ist es möglich, die Informationen vorheriger Planungsschritte einzusehen, wobei gleichzeitig sichergestellt ist, dass diese nicht verändert werden. Einen Eindruck vom Layout der einzelnen GUIs sowie der gesamten Benutzeroberfläche vermitteln die Abbildungen in Anhang B.

5.4 Berechnung des Weges aus dem Polygonzug

Das Planungsmodul kann aus den Informationen des vorgegebenen Polygonzugs einen Weg berechnen. Dazu werden entsprechende Objekte aus der Manöverdatenstruktur erzeugt, die in Kapitel 4 vorgestellt wurde.

Grundlegende Idee

Zu den Ausrichtungsänderungen des Polygonzugs sollen entsprechende *CC Turns* (Klasse *Turn*) erzeugt werden. Diese sollen so transformiert (Klasse *TransformiertesManoever*) werden, dass sie jeweils genau zwischen zwei Linien des Polygonzugs platziert sind. Jeder *CC Turn* nimmt einen bestimmten Platz der angrenzenden Linien ein. Diese Länge vom jeweiligen Eckpunkt des Polygonzugs zum Start- oder Endpunkt des *CC Turns* ist die Länge der Konstruktionsgeraden dieses *CC Turns* (siehe Abschnitt 4.2). Der Platz, der zwischen zwei *CC Turns* frei bleibt, soll schließ-

lich durch Geradenstücke (Klasse *Geradeausfahrt*) gefüllt werden. Es kann aber auch vorkommen, dass sich zwei *CC Turns* überschneiden, die Konstruktionsgeraden der *CC Turns* also zusammen länger sind als die Linie des Polygonzugs. In diesem Fall soll dies durch ein entsprechendes Geradenstück mit umgekehrter Fahrtrichtung ausgeglichen werden. Dabei handelt es sich um eine heuristische Lösung. Es sind auch andere Möglichkeiten denkbar.

Bei einer Richtungsänderung, also zwei hintereinander liegenden Linien mit umgekehrter Fahrtrichtung, soll der *CC Turn* auf der ersten Linie platziert sein. Es wird also direkt die Kurve gefahren und anschließend mit geänderter Fahrtrichtung geradeaus auf die zweite Linie gefahren (wie im Beispiel in Anhang B veranschaulicht). Die Alternative wäre, über den Eckpunkt hinaus zu fahren, um anschließend die Kurve mit geänderter Fahrtrichtung zur zweiten Linie zu fahren.

Damit hat also jeder *CC Turn* die Fahrtrichtung der jeweils ersten Linie. Die erzeugten transformierten Manöver sollen schließlich zu einem Weg (Klasse *CCWeg*) zusammengesetzt werden.

Algorithmus

Die Berechnung lässt sich durch folgenden Algorithmus beschreiben:

1. ***CC Turns* berechnen:** Zu den jeweiligen Ausrichtungsänderungen (Deltas) des Polygonzugs werden entsprechende *CC Turns* erzeugt. Da nicht-transformierte *CC Turns* immer Linkskurven sind, wird jeweils der Betrag der Ausrichtungsänderung verwendet.

2. ***CC Turns* transformieren:** Die *CC Turns* sollen an den entsprechenden Stellen zwischen den jeweiligen Linien des Polygonzugs platziert werden. Man betrachte hierzu Abbildung 5.5.

   Die durchgezogene schwarze Linie stellt einen Ausschnitt des Polygonzugs dar, wobei p_0, p_1 und p_2 drei der Punkte sind, die den Polygonzug definieren. Dabei

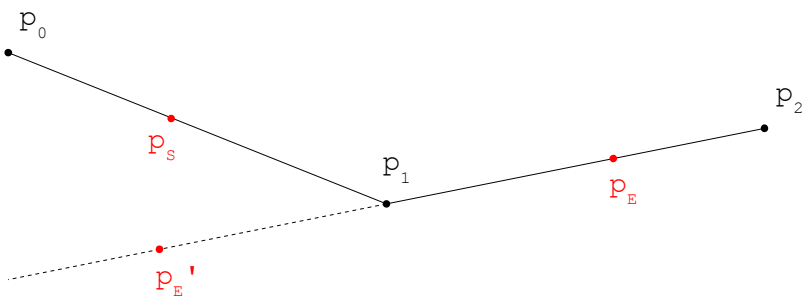

Abbildung 5.5: Veranschaulichung der Wegplanung.

soll von p_0 nach p_2 gefahren werden. Dazwischen soll der bereits berechnete *CC Turn* platziert werden. Zunächst wird die Länge der Konstruktionsgeraden des *CC Turns* bestimmt. Im Abstand dieser Länge zum Punkt p_1 werden die Punkte p_S, p_E und p'_E platziert, wobei p_S zwischen p_0 und p_1 liegt und p_E zwischen p_1 und p_2. Durch Spiegelung von p_E an p_1 erhält man p'_E. Der Punkt p_S sowie der Punkt p_E oder p'_E begrenzen den *CC Turn*. Falls die Linien $\overline{p_0p_1}$ und $\overline{p_1p_2}$ unterschiedliche Fahrtrichtung haben, entspricht die Ausrichtungsänderung des *CC Turns* dem Winkel zwischen p_S und p'_E bezüglich p_1. In diesem Fall wird der *CC Turn* entsprechend zwischen p_S und p'_E platziert. Bei gleicher Fahrtrichtung entspricht die Ausrichtungsänderung des *CC Turns* dem Winkel zwischen p_S und p_E bezüglich p_1. In diesem Fall wird der *CC Turn* entsprechend zwischen p_S und p_E platziert. Der jeweils dritte Punkt, p_E oder p'_E, ist irrelevant. Welcher der beiden begrenzenden Punkte der Startpunkt des transformierten Manövers ist, hängt von der Fahrtrichtung ab. Diese entspricht der Fahrtrichtung der jeweils ersten Linie $\overline{p_0p_1}$. Fährt man vorwärts, ist p_S der Startpunkt, fährt man rückwärts ist es p_E bzw. p'_E. Der Ortsvektor dieses Startpunkts definiert die Translation des transformierten *CC Turns*. Die Rotation entspricht der Ausrichtung der Linie, auf der der Startpunkt liegt, jeweils in Richtung p_1. Die Kurve ist eine Rechtskurve, falls die ursprüngliche Ausrichtungsänderung bei Fahrtrichtung vorwärts negativ oder bei Fahrtrichtung rückwärts positiv ist. Damit sind alle Transformationsinformationen ermittelt.

3. **Geradeausfahrten berechnen:** Auf den Linien des Polygonzugs sollen zwischen den Start- bzw. Endpunkten der beiden jeweils angrenzenden *CC Turns* passende Geradenstücke platziert werden. Der Abstand dieser Punkte bestimmt die Länge der *Geradeausfahrt*. Außerdem werden am Anfang und Ende des Polygonzugs, also vor dem ersten und nach dem letzten *CC Turn*, passende Geradeausfahrten erzeugt.

4. **Geradeausfahrten transformieren:** Zunächst muss die Fahrtrichtung bestimmt werden. Im Normalfall entspricht diese der Fahrtrichtung der entsprechenden Linie des Polygonzugs. Falls sich die beiden angrenzenden *CC Turns* aber überschneiden, wird die Fahrtrichtung umgekehrt. Der Startpunkt der transformierten Geradeausfahrt entspricht bei Fahrtrichtung vorwärts dem anliegenden begrenzenden Punkt des ersten *CC Turns* und bei Fahrtrichtung rückwärts dem anliegenden begrenzenden Punkt des zweiten *CC Turns*. Der jeweils andere Punkt ist der Endpunkt. Wie bei den *CC Turns* entspricht die Translation auch hier dem Ortsvektor des Startpunkts. Die Rotation entspricht der Ausrichtung der Geraden vom Start- zum Endpunkt des Manövers.

5. **Weg berechnen:** Die berechneten transformierten Manöver werden schließlich zu einem Weg zusammengesetzt.

5.5 Implementierung und Erweiterbarkeit

In diesem letzten Abschnitt soll abschließend auf Besonderheiten bei der Implementierung eingegangen werden. Zudem werden Vorschläge zur Erweiterung und Weiterentwicklung des Planungsmoduls vorgestellt.

Das verbundene Fahrzeug

Das Fahrzeug, das mit der Anwendung *EZLeitstand* und somit auch mit dem Planungsmodul verbunden ist, wird durch eine eigene Software realisiert. Die Beschäf-

tigung mit dieser Software ist im Rahmen dieser Bachelorarbeit nicht vorgesehen. Daher wird auf die vorhandene Software zurückgegriffen. Diese simuliert ein Fahrzeuggespann, bestehend aus Zugfahrzeug und Anhänger. Da die Konstruktoren der Manöverklassen *Turn* und *Geradeausfahrt* eine Fahrzeugbeschreibungskette übergeben bekommen müssen (siehe dazu Abschnitt 4.2, insbesondere Abbildung 4.5), werden bei der Wegplanung die Werte des Zugfahrzeugs eingelesen, um daraus eine neue Fahrzeugbeschreibungskette zu erzeugen. Die erzeugte Fahrzeugbeschreibungskette repräsentiert einen Car-like Robot mit den Abmessungen des Zugfahrzeugs und eignet sich daher zur Berechnung der *CC Turns*.

Bei einer Weiterentwicklung des Planungsmodul hinsichtlich der Realisierung des Fahrens (s.u.) ist es notwendig, die Fahrzeug-Software für einen Car-like Robot zu adaptieren. Zudem muss die Ereignisverarbeitung des Planungsmoduls für die Kommunikation mit diesem Fahrzeug angepasst werden (siehe Abschnitt 5.2.3). Bei der Berechnung des Weges aus dem Polygonzug sollte die entsprechende Fahrzeugbeschreibungskette des Car-like Robots eingelesen werden.

Realisierung des Fahrens

Bei der Entwicklung des Planungsmoduls wurde bereits darauf hingearbeitet, die Realisierung der Funktionalität des Fahrens zu ermöglichen. Dabei wurde darauf geachtet, dass der berechnete Weg alle Informationen bereitstellt, die ein Fahrzeug zum Fahren dieses Weges benötigt. Zudem lässt sich die Datenstruktur des Weges in die für das Fahren benötigte Datenstruktur konvertieren, die bereits in anderen Anwendungsmodulen der Anwendung *EZLeitstand* verwendet wird.

Bei einer Weiterentwicklung in diese Richtung sollten die dafür vorgesehenen Zustände *wpFahrtBearbeiten* und *wpFahren* verwendet werden (siehe Abschnitt 5.2.1). Außerdem sollte die dafür vorgesehene GUI *fahrtGui* verwendet und um entsprechende Elemente erweitert werden (siehe Abschnitt 5.3).

Implementierung weiterer Berechnungsmethoden

Im vorherigen Abschnitt 5.4 wurde die implementierte Methode für die Berechnung des Weges aus dem Polygonzug vorgestellt. Diese gleicht die Überschneidung zweier *CC Turns* durch eine Geradeausfahrt aus. Wie im genannten Abschnitt beschrieben, handelt es sich dabei um eine heuristische Lösung. Hier sind durchaus andere Möglichkeiten denkbar. Falls man das Planungsmodul um eine solche Berechnungsmethode erweitern möchte, genügt es, eine entsprechende Funktion zu implementieren und den Aufruf der Funktion entsprechend anzupassen.

6 Qualitative Ergebnisse

In den vorherigen Kapiteln wurden die Berechnung der Kurvenmanöver (*CC Turns*) sowie verschiedene Konzepte zur Konstruktion der Hülle vorgestellt. Außerdem wurde die Wegplanung des Planungsmoduls näher betrachtet. Diese drei wesentlichen Punkte der Bachelorarbeit sollen nun qualitativ analysiert werden. Dazu sollen die Ergebnisse verschiedener Berechnungen verglichen und bewertet werden. Zunächst wird die Berechnung der *CC Turns* betrachtet. Anschließend wird auf deren Hülle eingegangen. Dabei werden die beiden entwickelten Hüllenkonzepte gegenübergestellt. Abschließend werden die Auswirkungen dieser Berechnungen unter dem Einfluss verschiedener Faktoren auf die Wegplanung betrachtet. Die in diesem Zusammenhang gezeigten Beispiele und Vergleiche sollen eine abschließende Bewertung der entwickelten Software ermöglichen.

6.1 Vergleich der Spezialfälle bei CCTurns

Bei bestimmten Parametern kann es vorkommen, dass bei *CC Turns* eine Schleife gefahren wird. In diesem Fall ist es unter Umständen günstiger, die gewünschte Ausrichtungsänderung auf andere Weise zu erreichen. Dazu kann man auf die Spezialfälle eines *CC Turns* zurückgreifen, die in Abschnitt 2.2 vorgestellt wurden. Beim folgenden Beispiel eines *CC Turns* bietet sich sowohl die *Rückwärtsfahrt des Kreisbogens* als auch der *Elementary Path* an. Das Beispiel ist das Ergebnis folgender Parameter:

- Fahrzeug: Länge = 0.8, $L_1 = 0.4$, hinterer Überhang = 0.16, Breite = 0.5
- *CC Turn*: $\kappa_{max} = 0.5$, $\sigma = 0.18$, $\delta = \pi/5$

Die folgenden Abbildungen zeigen sowohl den Normalfall als auch die beiden Spezialfälle. Der Schwerpunkt der Betrachtung liegt dabei auf dem Pfad. Das Fahrzeug ist jeweils an den Stellen der Konfigurationen q_s, q_i, q_j und q_g eingezeichnet. Als Hülle wird jeweils die Ringhülle verwendet. Es handelt sich dabei, wie auch in den folgenden Abschnitten, jeweils um eine durch einen Polygonzug angenäherte Ringhülle (siehe Abschnitt 4.7). Der Iterationswinkel ist dabei so klein gewählt, dass die Kanten und Ecken des Polygonzugs in den Abbildungen nicht mehr zu erkennen sind.

Im Normalfall ergibt sich bei den gegebenen Parametern eine Schleife (siehe Abbildung 6.1). Die Fahrkurve überschneidet sich also selbst. Die Hülle bildet dabei einen vollständigen Ring.

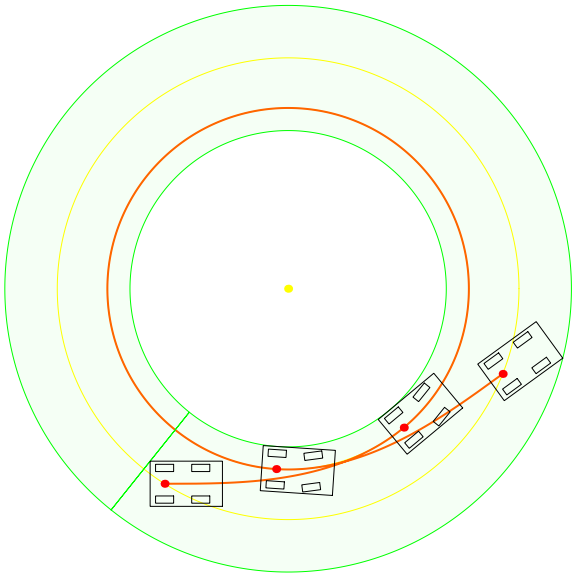

Abbildung 6.1: *CC Turn* mit Schleife im Normalfall.

An der Abbildung zum Normalfall ist sofort erkennbar, dass es günstiger wäre, zwischen den Konfigurationen q_i und q_j, die auf dem inneren Kreis liegen, rückwärts zu fahren. Damit würde man die Schleife vermeiden und die zu fahrende Strecke deutlich verkürzen. Der Spezialfall *Rückwärtsfahrt des Kreisbogens* ist in diesem Beispiel

also besonderes effizient. Nach [Fraichard u. Scheuer, 2004] würde man in diesem Fall die Anwendung der *Rückwärtsfahrt des Kreisbogens* gar nicht in Betracht ziehen, da die Bedingung $\delta > \delta_{min} + \pi$ (siehe Abschnitt 2.2.2) nicht erfüllt ist. Man würde stattdessen den *Elementary Path* verwenden. Die implementierte Berechnungsmethode erkennt aber auch in diesem Fall, dass der ursprüngliche Kreisbogen $> \pi$ ist und erlaubt daher die *Rückwärtsfahrt des Kreisbogens*. Wie bereits erwähnt ist hier aber auch die Verwendung des *Elementary Path* möglich. Wenn man weniger schnell einlenkt (geringere Schärfe σ), kann man „direkt“ von q_s nach q_g fahren.

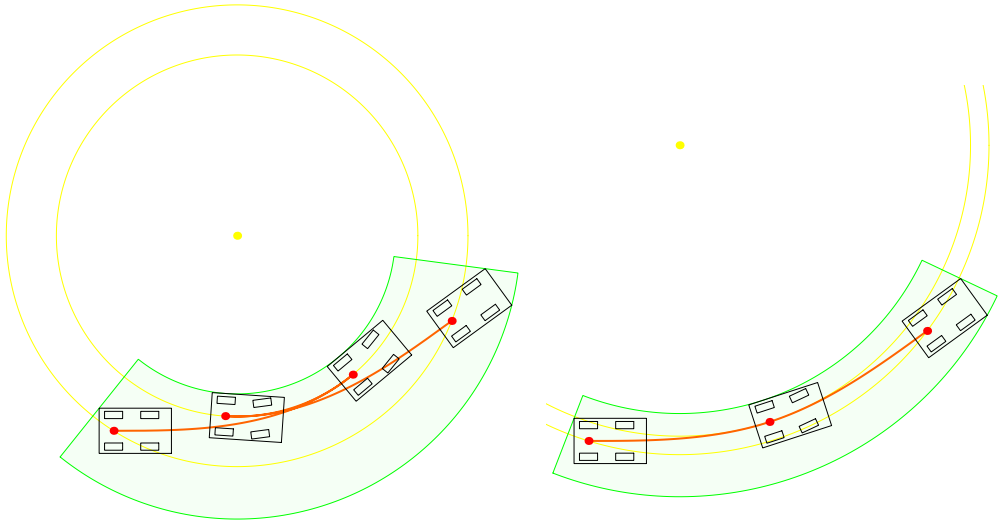

Abbildung 6.2: *CC Turn* in den Spezialfällen *Rückwärtsfahrt des Kreisbogens* und *Elementary Path*.

In Abbildung 6.2 sind beide Spezialfälle gegenübergestellt. Die Abbildung zeigt, dass die Anwendung jeder der beiden Spezialfälle eine enorme Verkürzung des Pfades ermöglicht. Die Hülle wird damit auch deutlich kleiner. Im direkten Vergleich wird an diesem Beispiel auch ersichtlich, dass der *Elementary Path* die günstigere Alternative darstellt. Der Weg ist dabei noch kürzer und die Ringhülle noch kleiner. Dies ist nicht nur in diesem Beispiel, sondern allgemein der Fall. Aufgrund der geringeren Schärfe der Klothoiden befindet sich das Fahrzeug beim *Elementary Path* nicht so weit innen. Dadurch ist auch der innere Kreisbogen der Ringhülle weiter außen und die eingenommene Fläche damit kleiner. Falls die Anwendung beider Spezialfälle

möglich ist, ist somit immer der *Elementary Path* vorzuziehen.

6.2 Vergleich der Hüllenkonzepte beim CCTurn

Nachdem im ersten Vergleich der Pfad im Fokus stand, soll nun die Hülle näher betrachtet werden. In diesem Abschnitt soll insbesondere ein Vergleich der beiden Hüllenkonzepte (siehe Abschnitt 4.7) erfolgen. Dazu werden die Ringhülle und die aus dem Pfad berechnete Hülle an verschiedenen Beispielen gegenübergestellt. Zunächst wird jedoch die Ringhülle näher betrachtet, da diese zuerst entwickelt wurde und deren Schwächen letztlich die Motivation für die Entwicklung des zweiten Konzepts war.

Anhand der Abbildungen im vorherigen Abschnitt (Abb. 6.1 und 6.2) war bereits zu erkennen, dass die Fläche der Ringhülle deutlich größer ist als die Fläche, die vom Fahrzeug tatsächlich eingenommen wird. Eine ähnliche Ringhülle wie bei der Schleifenfahrt in genanntem Beispiel zeigt Abbildung 6.3.

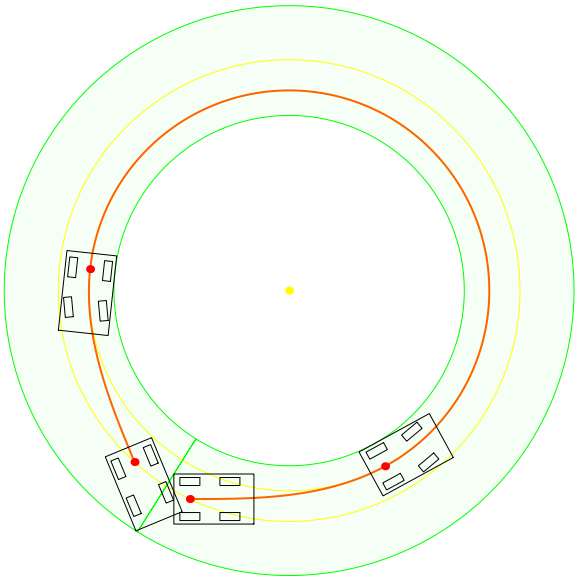

Abbildung 6.3: Ringhülle bei *CC Turn* mit großer Ausrichtungsänderung.

Es handelt sich hierbei um einen *CC Turn* mit sehr großer Ausrichtungsänderung ($\delta = 1.625\pi$). Die Hülle bildet hier ebenfalls einen vollständigen Ring. Das ist in diesem Beispiel der Fall, obwohl keine Schleife gefahren wird. Das eingezeichnete Fahrzeug in den Konfigurationen q_s und q_g macht allerdings deutlich, dass die Hülle so korrekt ist. Das zeigt zumindest, dass die Berechnung auch solche Sonderfälle beachtet. Das Beispiel zeigt aber auch, wo die Ringhülle im Normalfall eines *CC Turns* am meisten Platz zusätzlich einnimmt, verglichen mit der tatsächlich eingenommenen Fläche des Fahrzeugs: Am Kreisbogen. Die Ringhülle ermittelt den äußeren Radius aus dem am weitesten außen liegenden Eckpunkt des Fahrzeug am Start oder Ende des Manövers (hier: vorderer rechter Eckpunkt am Ende des *CC Turns*). Das Fahrzeug befindet sich bei der Fahrt auf dem Kreisbogen aber viel weiter innen. Das macht deutlich, dass das Konzept einer Ringhülle zwar geeignet, aber nicht optimal ist. Besonders in ungünstigen Fällen wird enorm viel Platz eingenommen. Der Vergleich mit der aus dem Pfad berechneten Hülle in den folgenden Beispielen verdeutlicht diesen Nachteil.

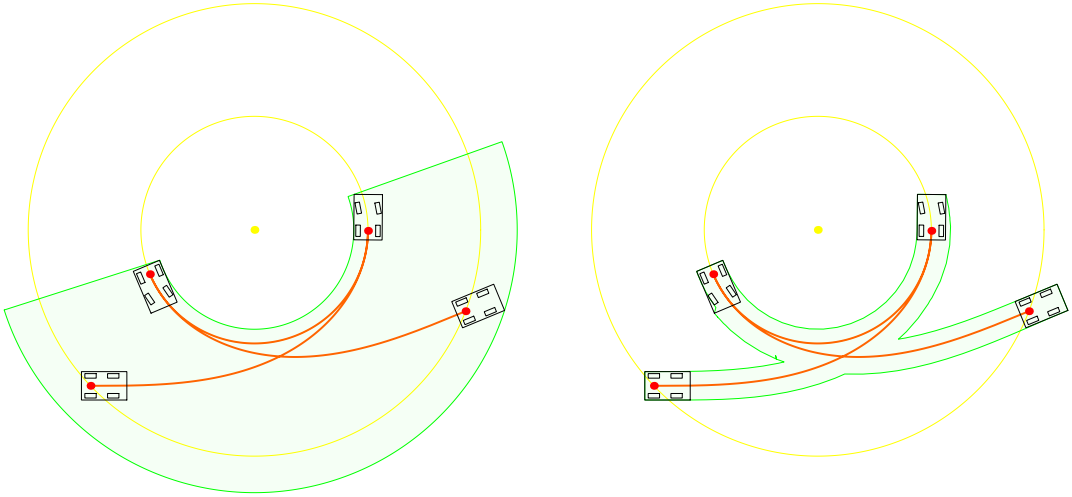

Abbildung 6.4: Vergleich der Hüllenkonzepte bei *Rückwärtsfahrt des Kreisbogens*.

Abbildung 6.4 vergleicht die beiden Hüllenkonzepte im Spezialfall *Rückwärtsfahrt des Kreisbogens*. Das Beispiel stellt einen Sonderfall dar, da die Konfigurationen q_i und q_j hinsichtlich des Winkels zum Mittelpunkt Ω weiter auseinander liegen als der

Start und das Ende des Manövers. Dieses Beispiel zeigt also, dass die Berechnung der Ringhülle (links in der Abbildung) auch diesen Sonderfall beachtet. Andererseits wird hier die bereits genannte Schwäche der Ringhülle besonders deutlich. Da der Kreisbogen des *CC Turns* sehr viel weiter innen liegt als der Start- und Endpunkt, wird der Ring sehr breit. Im Vergleich zum Konzept *Hülle aus Pfad* (rechts in der Abbildung), ist die zusätzlich eingenommene Fläche um ein Vielfaches größer.

Die folgenden beiden Beispiele basieren auf dem gleichen *CC Turn*, allerdings auf unterschiedlichen Fahrzeugabmessungen. Dies soll zeigen, dass die Abmessungen des Fahrzeugs ebenfalls Einfluss auf die Hülle haben. Abbildung 6.5 zeigt den *CC Turn* bei einem verhältnismäßig breiten Fahrzeug. Da das Fahrzeug sehr viel breiter ist als lang, fällt der Unterschied der Entfernungen zum Mittelpunkt Ω an den unterschiedlichen Stellen des Manövers kaum ins Gewicht. Daher fällt der eingenommene Platz der Ringhülle am Kreisbogenabschnitt moderat aus (links in der Abbildung). Im Vergleich zur Hülle, die aus dem Pfad berechnet wurde (rechts in der Abbildung), fallen nur noch die Flächen hinter dem Fahrzeug am Start des Manövers sowie vor dem Fahrzeug am Ende des Manövers negativ auf.

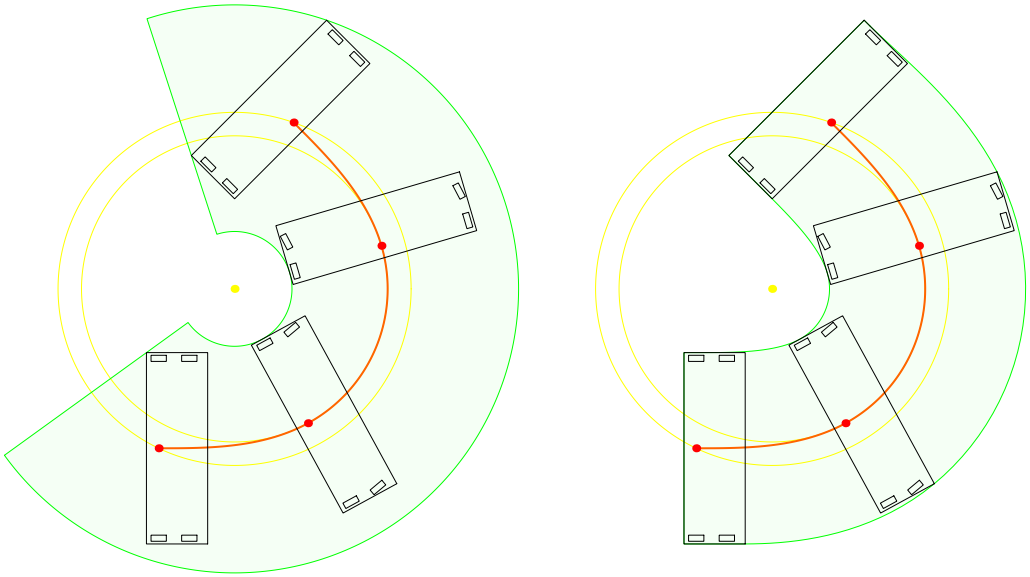

Abbildung 6.5: Vergleich der Hüllenkonzepte bei einem breiten Fahrzeug.

Abbildung 6.6 zeigt den gleichen *CC Turn* bei einem Fahrzeug mit verhältnismäßig großem hinteren Überhang. Das Beispiel ist repräsentativ für Fahrzeuge, bei denen der Radstand L_1 im Vergleich zur Gesamtlänge relativ klein ist. Im Gegensatz zu den vorherigen Beispielen bestimmt hier der hintere rechte Eckpunkt des Fahrzeugs am Start des Manövers den äußeren Radius der Ringhülle (links in der Abbildung). Auch hier ist im Vergleich zum Konzept *Hülle aus Pfad* (rechts in der Abbildung) die zusätzliche Platzeinnahme zu erkennen.

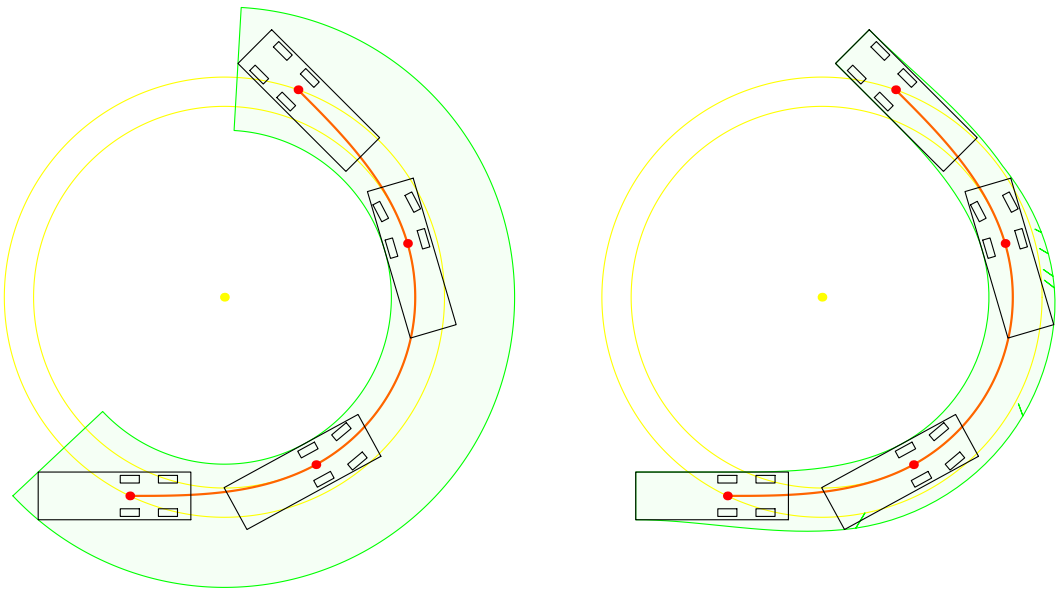

Abbildung 6.6: Vergleich der Hüllenkonzepte bei einem Fahrzeug mit großem hinteren Überhang.

Dieses Beispiel soll auch die Schwierigkeit bei der Entwicklung eines Hüllenkonzepts zeigen. Im abschließenden Fazit am Ende dieses Abschnitts wird daher nochmal auf dieses Beispiel eingegangen. Zunächst folgt jedoch ein letzter Vergleich.

Das folgende Beispiel soll zeigen, dass die eingenommene Fläche der Ringhülle auch von der Lenkgeschwindigkeit abhängt. In den bisher gezeigten Beispielen wurde von einer relativ geringen Lenkgeschwindigkeit ausgegangen. Die daraus resultierende Schärfe der Klothoiden ist ebenfalls gering. Das hat letztlich zur Folge, dass die Klo-

thoiden verhältnismäßig lang werden. Der Abstand des Fahrzeugs zum Mittelpunkt Ω ist dabei am Start und am Ende des Manövers deutlich größer als auf dem Kreisbogenabschnitt. Daraus resultiert eine sehr breite Ringhülle. Beim folgenden Beispiel wird nun eine höhere Lenkgeschwindigkeit gewählt. Daher sind die Klothoiden sehr kurz und der Abstand des Fahrzeugs zum Mittelpunkt Ω variiert nur minimal (siehe Abbildung 6.7). Die resultierende Ringhülle ist daher sehr schmal (links in der Abbildung). Im Vergleich zur Hülle, die nach dem Konzept *Hülle aus Pfad* berechnet wurde (rechts in der Abbildung), ist die zusätzliche Platzeinnahme nahezu vernachlässigbar.

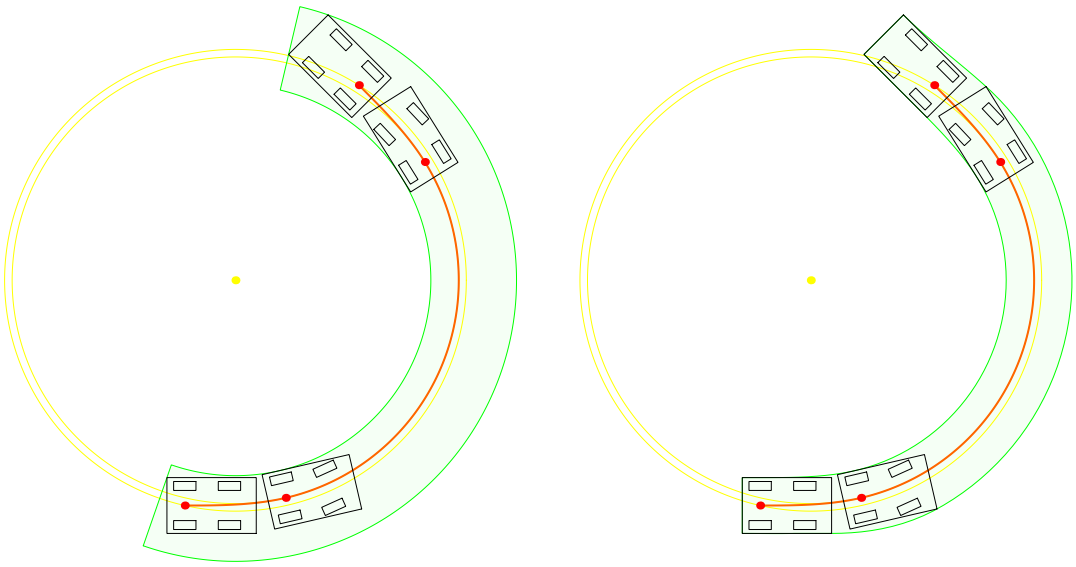

Abbildung 6.7: Vergleich der Hüllenkonzepte bei hoher Lenkgeschwindigkeit.

Das Beispiel zeigt, dass sich die Ringhülle bei hoher Lenkgeschwindigkeit an die aus dem Pfad berechnete Hülle annähert. Theoretisch haben die Hüllen bei unbegrenzter Lenkgeschwindigkeit die gleiche Breite. Allerdings würde das eine sprunghafte Lenkwinkeländerung bedeuten und das soll ja gerade durch die *CC Turns* vermieden werden. Dennoch ist es beim Fahren sicherlich wünschenswert, so schnell wie möglich einzulenken. Daher ist eine hohe Lenkgeschwindigkeit im Rahmen der kinematischen Möglichkeiten des Fahrzeugs bei *CC Turns* empfehlenswert.

Fazit

Die gezeigten Beispiele verdeutlichen, dass die Ringhülle verhältnismäßig viel Platz einnimmt. Solche extremen Fälle wie im Beispiel mit der *Rückwärtsfahrt des Kreisbogens* (Abbildung 6.4) treten aber nicht auf, wenn man den *Elementary Path* vorzieht, wie im vorherigen Abschnitt 6.1 empfohlen. Die gezeigten Beispiele sind also größtenteils Extremfälle. Diese wurden bewusst gewählt, um den Einfluss verschiedener Faktoren auf die Größe der Ringhülle zu zeigen. Zusammenfassend lässt sich festhalten, dass man die Fläche der Ringhülle in Grenzen halten kann, indem man nach Möglichkeit die Spezialfälle, insbesondere den *Elementary Path*, verwendet und eine möglichst hohe Lenkgeschwindigkeit wählt.

An dieser Stelle soll ausdrücklich darauf hingewiesen werden, dass es sich bei der verwendeten Ringhülle um die kleinstmögliche Ringhülle mit dem Mittelpunkt Ω handelt. Zudem ist die Berechnung selbst in den verschiedensten Sonderfällen korrekt. Da die Hülle in jedem Fall die Fläche umschließt, die das Fahrzeug beim Manöver einnimmt, handelt es sich um eine gültige Hülle. Der Nachteil der verhältnismäßig großen Fläche ist dennoch unbestreitbar. Die aus dem Pfad berechnete Hülle scheint in dieser Hinsicht optimal zu sein, da sie scheinbar genau der Fläche entspricht, die das Fahrzeug einnimmt. Genau genommen ist das aber nicht der Fall. Bei der Berechnung werden die Extrempunkte an verschiedenen Stellen des Manövers betrachtet und durch gerade Linien miteinander verbunden (siehe Abschnitt 4.7.2). Für die innere Begrenzung der Hülle ist dies korrekt, da die Linien mit den Sekanten der Ringhülle vergleichbar sind. Bei der äußeren Begrenzung jedoch wird mit jeder Linie ein Stück der tatsächlich benötigten Fläche „abgeschnitten“. Bei entsprechend großer Iteration fällt dies kaum ins Gewicht. Allerdings handelt es sich – streng betrachtet – nicht um eine gültige Hülle. Die aus dem Pfad berechnete Hülle stellt lediglich eine sehr gute Annäherung dar.

Bei der Entwicklung des Konzepts *Hülle aus Pfad*, das hier vorgestellt wurde, wurde auch eine alternative Konstruktion in Betracht gezogen. Die Idee war ebenfalls,

die Hülle mit Hilfe des Pfades zu berechnen. Anstatt Extrempunkte an einzelnen Stellen des Manövers einfach durch Linien zu verbinden, war aber die Verwendung von Tangenten vorgesehen. Die Schwierigkeit dabei ist die Ermittlung der jeweiligen Ausrichtung bzw. Steigung der Tangenten. Diese entspricht nämlich nicht der Fahrzeugausrichtung. Wie bereits angedeutet, macht dies das letzte Beispiel (Abbildung 6.6) deutlich. Beim zweiten Hüllenkonzept fällt hier auf, dass das Heck des Fahrzeugs aufgrund seiner Abmessungen auf der ersten Klothoide derart nach hinten ausschlägt, dass die Hülle nach unten breiter wird. Würde man eine Tangente an den Extrempunkt (hinterer rechter Eckpunkt des Fahrzeugs) am Start des Manövers anlegen, mit der Ausrichtung des Fahrzeugs im Startpunkt (also 0), so würde man den unteren Teil der tatsächlich eingenommenen Fläche abschneiden (siehe Abbildung 6.6). Um eine gültige Hülle zu entwerfen, müssten also die Bewegungslinien der einzelnen Extrempunkte inklusive einer Ausrichtungsinformation ermittelt werden. Dies kann aus dem vorhandenen Pfad nicht trivial berechnet werden. Dazu wären umfangreichere Überlegungen notwendig. Da im Rahmen dieser Bachelorarbeit die Zeit dafür nicht mehr gegeben war, wurde darauf verzichtet, in diese Richtung weiterzuarbeiten. Die beiden entwickelten Hüllenkonzepte sind ohnehin für die meisten Anwendungsbereiche gut geeignet. Bei einer vorläufigen Planung, die keine absolute Genauigkeit voraussetzt, bietet die aus dem Pfad berechnete Hülle eine sehr gute Annäherung. Falls absolute Genauigkeit gefordert ist, kann man auf die Ringhülle zurückgreifen.

6.3 Analyse der Berechnung des Weges aus dem Polygonzug

Nachdem in den vorherigen Abschnitten die Realisierung der *CC Turns* innerhalb der Manöverdatenstruktur betrachtet und bewertet wurde, behandelt dieser Abschnitt die Realisierung der Wegplanung innerhalb des Planungsmoduls. Das Vorgehen bei der Wegplanung kann dem Anhang B entnommen werden. An dieser Stelle hingegen soll die Berechnung des Weges aus dem Polygonzug veranschaulicht werden.

Dazu wird zunächst ein Polygonzug zugrunde gelegt (siehe Abbildung 6.8). Zu diesem Polygonzug werden mit verschiedenen Parametern Wege erzeugt. Dies soll die Auswirkungen verschiedener Faktoren auf das Ergebnis der Berechnung zeigen. Ein Vergleich des Normalfalls und der Spezialfälle eines *CC Turns* hat bereits in Abschnitt 6.1 stattgefunden. Die Ergebnisse zeigen, dass nach Möglichkeit immer der *Elementary Path* verwendet werden sollte. Daher wird im Rahmen dieses Beispiels die entsprechende Option aktiviert und damit ungünstige Sonderfälle eines *CC Turns* ausgeschlossen.

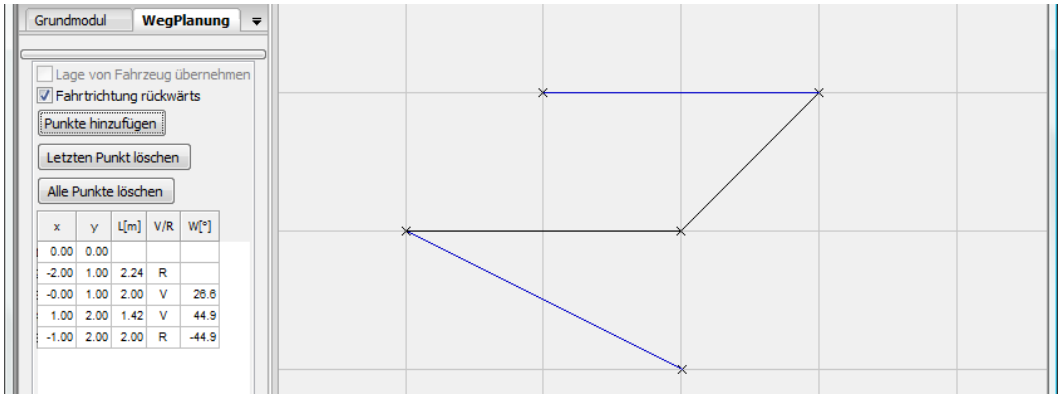

Abbildung 6.8: Wegplanung: Polygonzug.

Der Tabelle links in Abbildung 6.8 ist zu entnehmen, dass der daneben abgebildete Polygonzug unten mit der Fahrtrichtung rückwärts beginnt. An die erste Linie schließen sich zwei weitere Linien mit der Fahrtrichtung vorwärts an. An der letzten Linie soll wieder rückwärts gefahren werden. Der Polygonzug besteht also aus vier Linien. Daraus ergeben sich vier Geradeausfahrten und drei *CC Turns*.

Abbildung 6.9 veranschaulicht die erste Berechnung. Dabei wurde die Ringhülle verwendet. Der Vergleich des Pfades mit dem zugrundeliegenden Polygonzug zeigt, dass der *CC Turn* bei einem Wechsel der Fahrtrichtung zwischen zwei Linien an die jeweils erste Linie angelegt ist (erster und dritter *CC Turn*). Neben dem Pfad sind auch die Stellen der Richtungswechsel eingezeichnet (Kreuze am Ende von Turn 1 und 3).

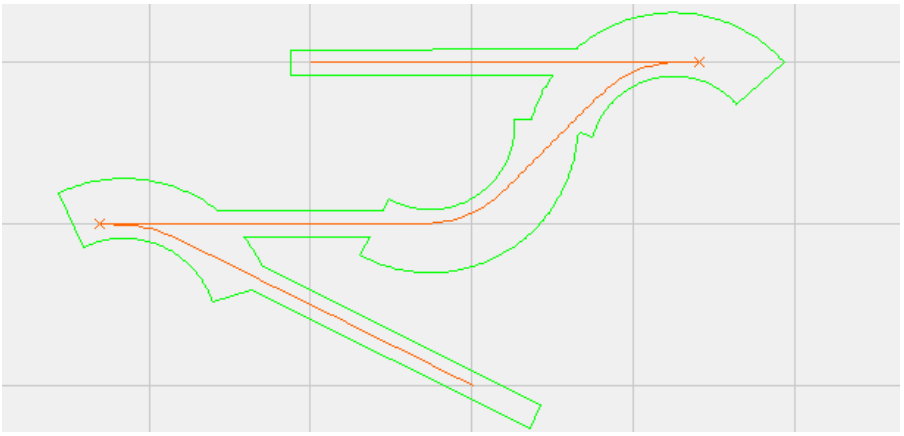

Abbildung 6.9: Wegplanung: Ergebnis mit Ringhülle.

Auch hier bei der Wegplanung wird deutlich, dass die Ringhülle relativ viel Platz einnimmt. Abbildung 6.10 zeigt die gleiche Berechnung, allerdings mit der aus dem Pfad berechneten Hülle. Im Vergleich zur anderen Methode nimmt die Hülle deutlich weniger Platz ein. Außerdem ergeben sich hier glatte Übergänge zwischen den Hüllen der einzelnen Manöver.

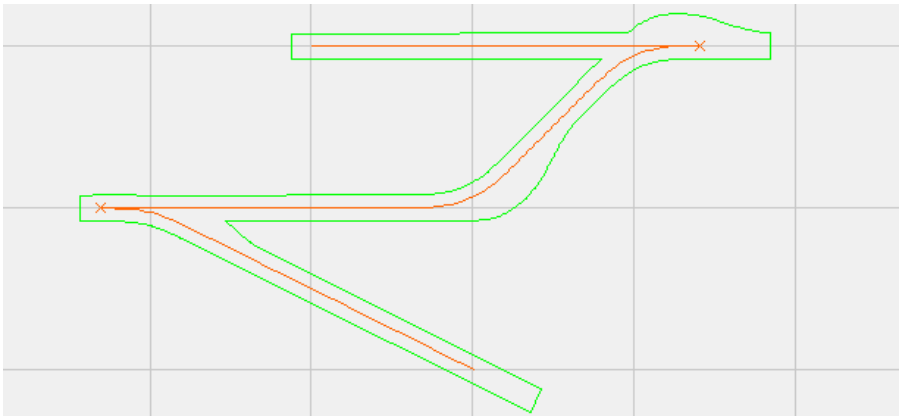

Abbildung 6.10: Wegplanung: Ergebnis mit *Hülle aus Pfad*.

Bei der dritten Berechnung wird ein deutlich geringerer maximaler Lenkwinkel verwendet (siehe Abbildung 6.11). Dadurch werden die *CC Turns* länger. Dies führt dazu, dass sich im gegebenen Beispiel zwei *CC Turns* überschneiden (Turn 2 und 3), da die Linie des Polygonzugs nicht genug Platz bietet. Diese Überschneidung

wird mit einer Geradeausfahrt mit umgekehrter Fahrtrichtung ausgeglichen. Die eingezeichneten Richtungswechselstellen verdeutlichen dies. Es ist erkennbar, dass die Linie des Polygonzugs, auf der diese ausgleichende Geradeausfahrt liegt, nicht komplett abgefahren wird. Das zeigt, dass sich die Berechnung im Wesentlichen an der jeweiligen Ausrichtungsänderung zwischen zwei benachbarten Linien des Polygonzugs orientiert. Diese Ausrichtungsänderung wird in jedem Fall erreicht. Allerdings kann es dabei vorkommen, dass der Pfad an einigen Stellen nicht auf dem Polygonzug verläuft oder sich sogar teilweise sehr weit vom Polygonzug entfernt.

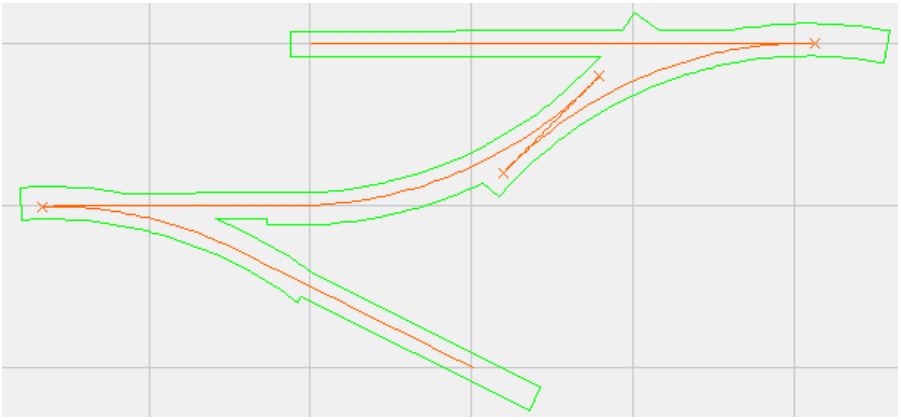

Abbildung 6.11: Wegplanung: Ergebnis bei kleinem maximalen Lenkwinkel.

Bei dieser letzten Berechnung wurde die Ringhülle verwendet. Aufgrund des geringeren Lenkwinkels bei gleichbleibender Lenkgeschwindigkeit im Vergleich zu den vorherigen Berechnungen, ist die Platzeinnahme der Ringhülle hier deutlich geringer. Die Hüllenberechnung nach dem Konzept *Hülle aus Pfad* würde hier ein nur minimal besseres Ergebnis liefern.

7 Schluss

Nachdem im vorherigen Kapitel 6 eine Betrachtung und Bewertung der Ergebnisse wesentlicher Punkte der Bachelorarbeit stattgefunden hat, soll in diesem letzten Kapitel die gesamte Arbeit rückblickend betrachtet und bewertet werden. Dazu werden die wichtigsten Ergebnisse der Bachelorarbeit zusammengefasst, um eine Gesamtbewertung zu ermöglichen. Die Bachelorarbeit schließt mit einem Ausblick.

7.1 Zusammenfassung

Die in Kapitel 2 vorgestellte Planungsmethode von Fraichard und Scheuer ([Fraichard u. Scheuer, 2004]) beschreibt Kurvenmanöver für Car-like Robots mit kontinuierlicher Lenkwinkeländerung (*CC Turns*). Diese bestehen aus je zwei Klothoiden und einem dazwischenliegenden Kreisbogen. Für ungünstige Parameter bieten die beiden Spezialfälle eines *CC Turns*, *Rückwärtsfahrt des Kreisbogens* und *Elementary Path*, die Möglichkeit, die gewünschte Ausrichtungsänderung dennoch auf möglichst kurzer Strecke zu erreichen.

Im Rahmen dieser Bachelorarbeit wurde die Planungsmethode durch die in Kapitel 4 behandelte Manöverdatenstruktur realisiert. Die Datenstruktur bietet neben einer Klasse für *CC Turns* auch Klassen für Geradeausfahrten und aus Manövern zusammengesetzte Wege. Da die Fläche, die das Fahrzeug beim Manöver einnimmt, in der Planungsmethode von Fraichard und Scheuer nicht betrachtet wird, wurden zudem zwei verschiedene Hüllenkonzepte entworfen und innerhalb der Manöverdatenstruk-

tur implementiert (siehe Abschnitt 4.7). Das erste Konzept ist das einer Ringhülle, die durch einen Polygonzug angenähert wird. Das im weiteren Verlauf als Alternative entwickelte zweite Konzept beschreibt die Berechnung der Hülle aus dem Pfad.

Auf der Manöverdatenstruktur aufbauend wurde schließlich die vorhandene Anwendung *EZLeitstand* um ein Planungsmodul erweitert. Das in Kapitel 5 behandelte Planungsmodul gibt dem Benutzer die Möglichkeit, einen Polygonzug vorzugeben, zu dem das Planungsmodul einen aus Manövern zusammengesetzten, fahrbaren Weg erzeugt.

In Kapitel 6 wurde abschließend eine qualitative Analyse der entwickelten Software durchgeführt. Dabei wurden insbesondere die *CC Turns* und die beiden Hüllenkonzepte betrachtet. Zudem wurde die Berechnung des Weges aus dem Polygonzug innerhalb des Planungsmoduls betrachtet und bewertet.

7.2 Bewertung

Die Planungsmethode von Fraichard und Scheuer ([Fraichard u. Scheuer, 2004]) ermöglicht die Planung von Wegen mit glatten Übergängen zwischen Geradeaus- und Kurvenfahrten. Dazu wird beim Kurvenmanöver (*CC Turn*) eine kontinuierliche Lenkwinkeländerung während der Fahrt mit konstanter Geschwindigkeit verwendet. Die Methode vermeidet somit eine sprunghafte Lenkwinkeländerung. Dennoch bietet die Planungsmethode die Möglichkeit, die Konfiguration des Fahrzeugs an jeder Stelle eines Manövers exakt zu bestimmen. Zudem bieten die Spezialfälle eines *CC Turns*, *Rückwärtsfahrt des Kreisbogens* und *Elementary Path*, die Möglichkeit, die gewünschte Ausrichtungsänderung auch bei ungünstigen Parametern auf möglichst kurzer Strecke zu erreichen. Da die kinematischen Eigenschaften des Fahrzeugs bei der Berechnung berücksichtigt werden, sind die resultierenden Manöver in jedem Fall fahrbar.

Die entwickelte Manöverdatenstruktur realisiert diese Planungsmethode. Der Pfad eines Manövers oder Weges liefert dabei zu jeder Stelle des Manövers bzw. Weges nicht nur die Position des Fahrzeugs, sondern auch die Ausrichtung und den Lenkwinkel. Zudem können die Stellen eines Wechsels der Fahrtrichtung abgefragt werden. Eine ausführliche Bewertung der beiden entwickelten Hüllenkonzepte für *CC Turns* ist bereits in Abschnitt 6.2 erfolgt. An dieser Stelle soll zusammenfassend herausgestellt werden, dass die Ringhülle einerseits verhältnismäßig viel Platz einnimmt, andererseits aber eine gültige Hülle darstellt und in jedem Fall Kollisionsfreiheit garantiert. Die Fläche der Hülle, die aus dem Pfad berechnet wird, ist deutlich kleiner. Allerdings ist die Hülle streng genommen nicht gültig, da sie die tatsächlich eingenommene Fläche des *CC Turns* von „innen“ annähert. Bei entsprechend hoher Iteration ist diese Annäherung jedoch sehr genau und somit der Unterschied zur tatsächlich eingenommenen Fläche nur noch marginal. Die beiden Konzepte haben somit unterschiedliche Qualitäten. Je nach Anwendungsbereich kann also die entsprechende Methode gewählt werden. Auch aufgrund dieser Wahlmöglichkeit stellen die entwickelten Konzepte eine gute Lösung der Hüllenproblematik dar.

Das entwickelte Planungsmodul realisiert die geforderte Möglichkeit der Planung eines Weges. Von zentraler Bedeutung ist dabei der Algorithmus, der aus dem vorgegebenen Polygonzug einen Weg berechnet. Eine ausführliche Betrachtung und Bewertung ist in Abschnitt 6.3 erfolgt. Zusammenfassend soll an dieser Stelle festgehalten werden, dass sich der Algorithmus im Wesentlichen an den Ausrichtungsänderungen zwischen den Linien des Polygonzugs orientiert. Falls der Platz auf einer Linie nicht für die beiden anliegenden Kurvenmanöver ausreicht, wird dies durch eine Geradeausfahrt ausgeglichen. Dabei wird zwar jeweils die gewünschte Ausrichtungsänderung erreicht, allerdings kann es in Extremfällen vorkommen, dass der Pfad an einigen Stellen deutlich vom Polygonzug entfernt verläuft. Daraus resultiert jedoch der Vorteil, dass theoretisch zu jedem beliebigen Polygonzug ein Weg berechnet werden kann.

7.3 Ausblick

Vorschläge zur Weiterentwicklung der entwickelten Manöverdatenstruktur wurden bereits in Abschnitt 4.8.2 gemacht. Dazu gehört zum einen die Implementierung weiterer Manövertypen. Dabei ist allerdings darauf zu achten, dass diese kompatibel zur Schnittstelle für Manöver und zum Konzept der Transformation sind. Ein wesentlicher Punkt, bei dem Spielraum für Verbesserungen besteht, ist die Hülle des *CC Turns*. Hier ist es denkbar, die bestehenden Konzepte zu optimieren oder neue Konzepte zu entwickeln.

Das entwickelte Planungsmodul bietet ebenfalls Möglichkeiten der Weiterentwicklung (siehe Abschnitt 5.5). Bei der Entwicklung wurde darauf geachtet, eine spätere Implementierung der Funktionalität des Fahrens zu ermöglichen. Mit den Informationen, die der Pfad des berechneten Weges bereitstellt, sowie der Möglichkeit, die Stellen eines Wechsels der Fahrtrichtung abzufragen, stehen alle Informationen zur Verfügung, die ein Fahrzeug zum Fahren benötigt. Da die Auslegung auf eine spätere Realisierung des Fahrens ausdrücklich gefordert war, wird diese Funktionalität sicherlich in naher Zukunft realisiert werden.

A Pfad und Huelle zur Laufzeit

Wie in Abschnitt 4.3 beschrieben, besitzen *Geradeausfahrt*, *Turn*, *Transformiertes-Manoever* und *CCWeg* jeweils ein Attribut vom Typ *sKonfigurationsfunktionen* für den Pfad sowie ein Attribut vom Typ *sHuelle* für die Hülle. Die folgenden Objektdiagramme zeigen, welche implementierenden Klassen für den Pfad und dessen Funktionen sowie für die Hülle zur Laufzeit verwendet werden.

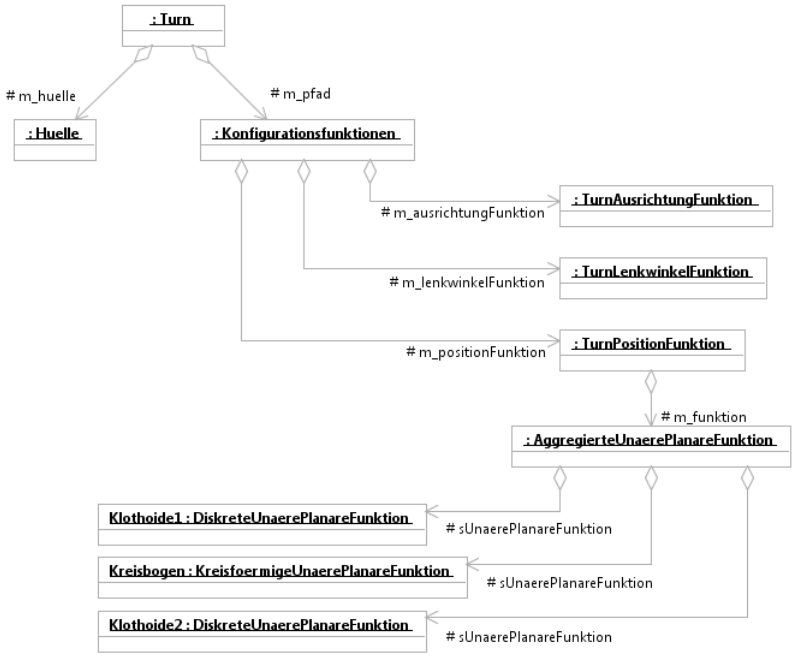

Abbildung A.1: Pfad und Hülle von *Turn* zur Laufzeit.

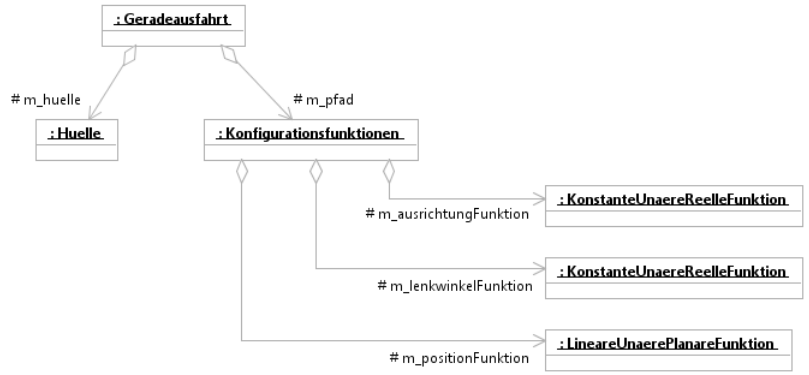

Abbildung A.2: Pfad und Hülle von *Geradeausfahrt* zur Laufzeit.

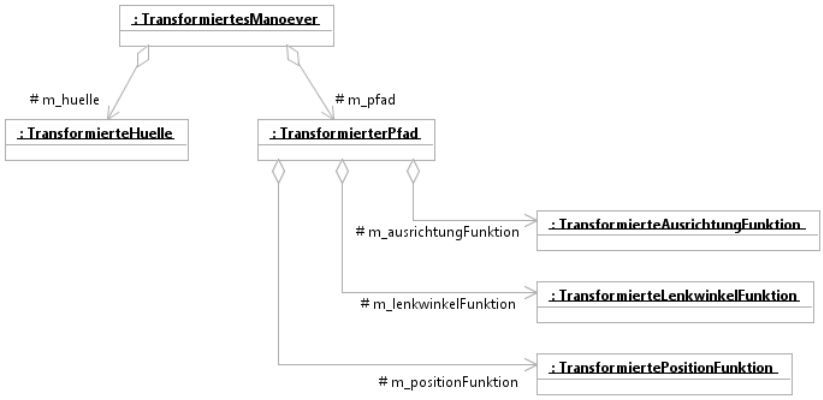

Abbildung A.3: Pfad und Hülle von *TransformiertesManoever* zur Laufzeit.

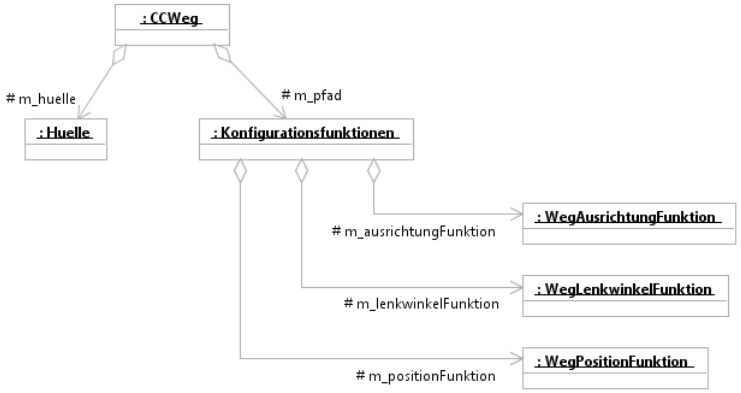

Abbildung A.4: Pfad und Hülle von *CCWeg* zur Laufzeit.

B Vorgehen bei der Wegplanung

Im Folgenden wird das Vorgehen bei der Wegplanung des Planungsmoduls (siehe Kapitel 5) beschrieben.

1. Schritt: Polygonzug erzeugen

Im ersten Schritt kann der Benutzer einen Polygonzug erzeugen (siehe Abbildung B.1).

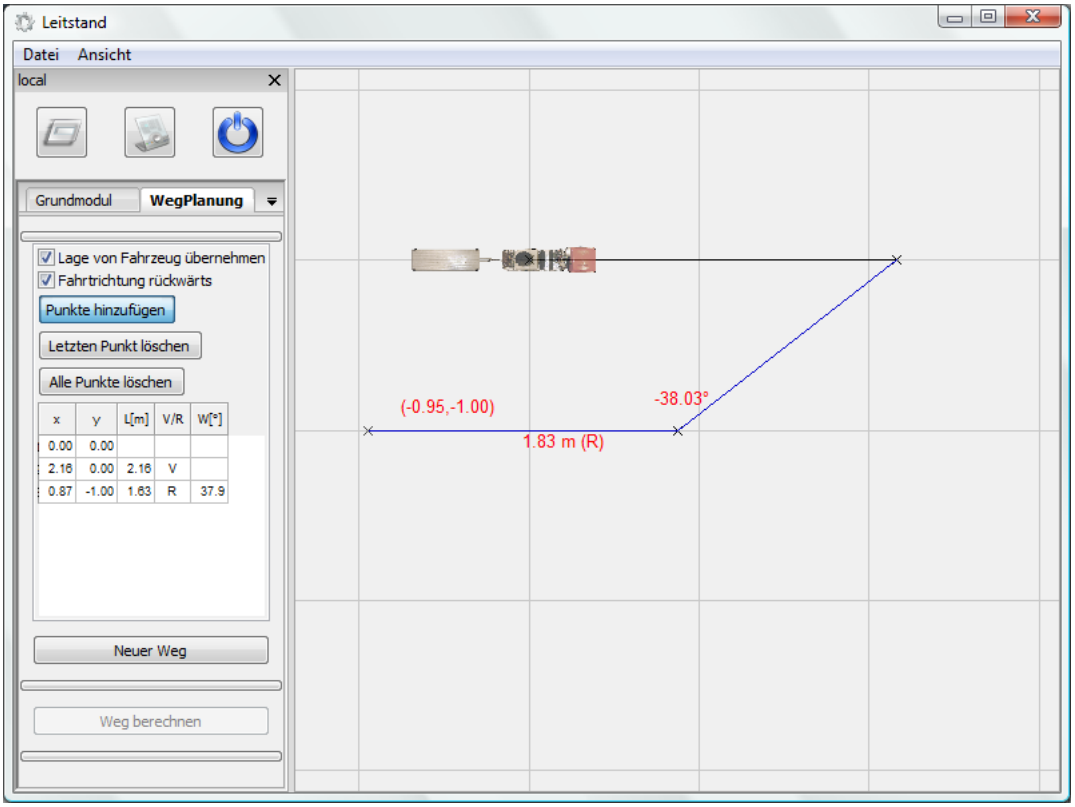

Abbildung B.1: Erzeugen des Polygonzugs.

Wie in Abschnitt 5.2.2 beschrieben, besteht dabei die Möglichkeit, die Lage des Fahrzeugs, das ebenfalls eingezeichnet ist, für den Anfang des Polygonzugs zu übernehmen. Die Eingabehilfen auf der Zeichenfläche unterstützen den Benutzer bei der Planung.

2. Schritt: Parameter festlegen und Weg berechnen

Im zweiten Schritt kann der Benutzer die Parameter für die Berechnung des Weges aus dem Polygonzug festlegen (siehe Abbildung B.2). Es handelt sich dabei um Parameter für die Berechnung der *CC Turns* und deren Pfade und Hüllen. Diese haben aber letztlich Einfluss auf den gesamten Weg.

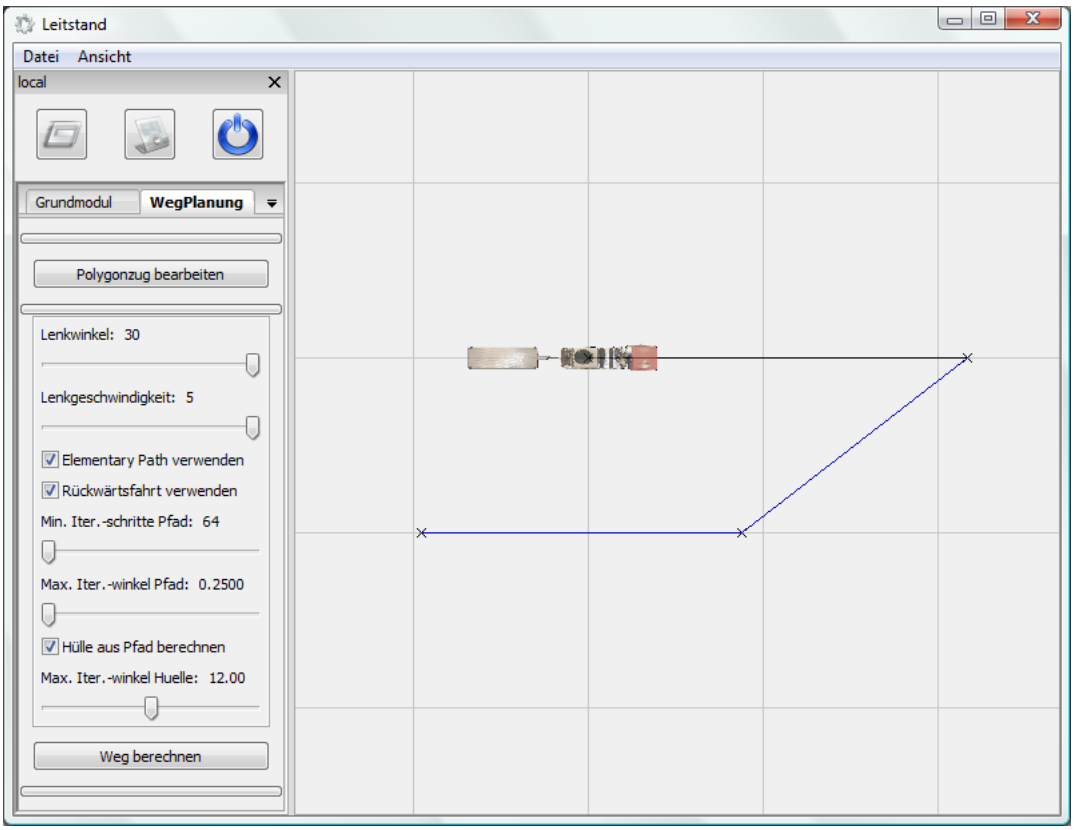

Abbildung B.2: Festlegen der Parameter zur Berechnung des Weges aus dem Polygonzug.

Ergebnis: Pfad und Hülle des Weges

Nach der Berechnung werden Pfad und Hülle des Weges auf der Zeichenfläche dargestellt (siehe Abbildung B.3). Der Benutzer hat nun die Möglichkeit, verschiedene Elemente ein- und auszublenden. Außerdem kann er die Genauigkeit festlegen, mit der der Pfad gezeichnet wird. Bei einer Weiterentwicklung des Planungsmoduls würde an dieser Stelle beispielsweise folgender dritte Planungsschritt folgen: Man würde Parameter für eine Fahrt festlegen, die dann gestartet werden könnte.

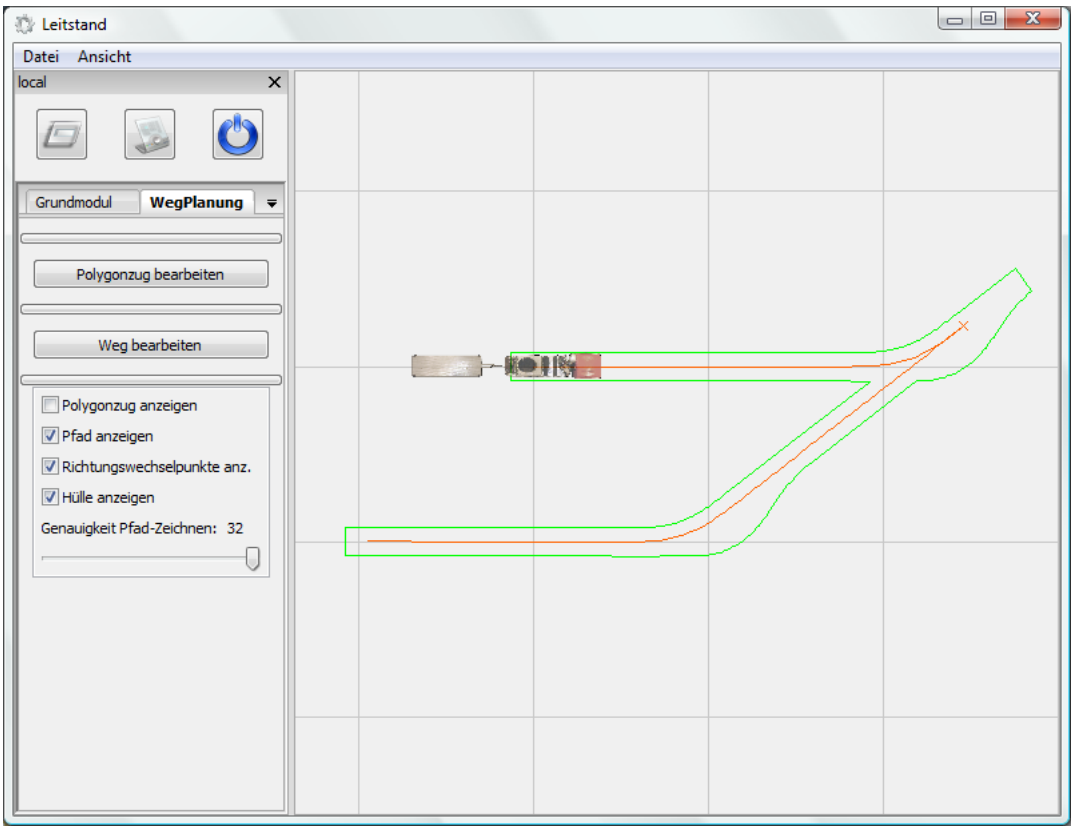

Abbildung B.3: Darstellung des Weges nach der Berechnung.

Literaturverzeichnis

[Dubins 1957] Dubins, L. E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. In: *American Journal of Mathematics* 79 (1957), S. 497–517

[Fraichard u. Scheuer 2004] Fraichard, Thierry ; Scheuer, Alexis: From Reeds and Shepp's to Continuous-Curvature Paths. In: *IEEE Transactions on Robotics* 20 (2004), Dezember, Nr. 6

[Press u. a. 2002] Press, William H. ; Teukolsky, Saul A. ; Vetterling, William T. ; Flannery, Brian P.: *The Art of Scientific Computing - Numerical Recipes in C++*. 2. Cambridge University Press, 2002

[Reeds u. Shepp 1990] Reeds, J. A. ; Shepp, L. A.: Optimal paths for a car that goes both forward and backward. In: *Pacific Journal of Mathematics* 145 (1990), Nr. 2, S. 367–393

[Scheuer 1999] Scheuer, Alexis: SubOptimal Continuous-Curvature Path Planning for Non-Holonomic Robots. In: *Journées des Jeunes Chercheurs 11ème édition*. Lausanne, April 1999

[Scheuer u. Fraichard 1997a] Scheuer, Alexis ; Fraichard, Thierry: Continuous-Curvature Path Planning for Car-Like Vehicles. In: *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*. Grenoble, September 1997

[Scheuer u. Fraichard 1997b] Scheuer, Alexis ; Fraichard, Thierry: Continuous-

Curvature Path Planning for Multiple Car-Like Vehicles. In: *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems.* Grenoble, September 1997

[Weyand u. a. 2010] Weyand, Christian ; Balcerak, Elisabeth ; Zöbel, Dieter: Iterative Generation of Smooth Maneuvers for Articulated Vehicles. In: *Proceedings of the IASTED International Conference Robotics and Applications (RA 2010).* Cambridge, Massachusetts, USA, 2010

[Wojke 2005] Wojke, Philipp: *EZauto – Softwarearchitektur für Anwendungen des assistierten oder autonomen Fahrens.* 2005. – unveröffentlicht

[Zöbel 2003] Zöbel, Dieter: Trajectory Segmentation for the Autonomous Control of Backward Motion for Truck and Trailer. In: *IEEE Transactions on Intelligent Transportation Systems* 4 (2003), Nr. 2, S. 59–66

[Zöbel 2008] Zöbel, Dieter: *Kinematic Modeling of Moving Vehicles - EZauto.* Dezember 2008. – unveröffentlicht