UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

ist
Institut für Softwaretechnik

Ermittlung der Auswirkung von Modellfehlern auf die Zielgenauigkeit und Handhabbarkeit eines Fahrerassistenzsystems beim rückwärtigen Rangieren

Masterarbeit

zur Erlangung des Grades eines Master of Science
im Studiengang Informatik

vorgelegt von

Daniel Mc Stay

Matrikelnummer: 209110589

Erstgutachter:	Prof. Dr. Dieter Zöbel Institut für Softwaretechnik
Zweitgutachter:	Dipl.-Inform. Benjamin Knopp Institut für Softwaretechnik

Koblenz, im Januar 2015

Erklärung

Hiermit bestätige ich, dass die vorliegende Arbeit von mir selbstständig verfasst wurde und ich keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet–Quellen — benutzt habe und die Arbeit von mir vorher nicht in einem anderen Prüfungsverfahren eingereicht wurde. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium (CD-Rom).

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

. . . . . .

(Ort, Datum) (Unterschrift)

Abstract

In this master thesis it will be examined how the modeling errors influence the positioning accuracy and usability of a driver assistance system while reversing a vehicle. Special emphasis will be put on the identification of error thresholds. It will be looked into how big an initial error is allowed to be for the assistance system to still offer acceptable quality traits regarding precision and robustness. First a quantitative error analysis of the kinematic model of the vehicle will be considered. Based on that a qualitative approach will be conducted including the execution of systematic experiments. Therefore a controller will be developed, that allows the execution of a maneuver based on the visual information the assistance system provides. A method will be proposed to asses the maneuver by means of predefined error thresholds. To allow an efficient search through all possible error configurations, a probabilistic method, called Annealed Particle Filter will be used. Eventually systematic experiments will be carried out with the help of a testtool. Further evaluation will take place in the context of a collaboration between Koblenz and the Fraunhofer Institute in Kaiserslautern. The assistance system will be ported to the local RODOS® simulation environment, which will enable tests in a controllable and reproducible environment.

Zusammenfassung

Im Rahmen dieser Arbeit wird untersucht, wie sich Modellfehler auf die Positionsgenauigkeit und Handhabbarkeit beim Rangieren mit einem Fahrerassistenzsystem auswirken. Besonderen Wert wird dabei auf die Bestimmung von Fehlergrenzen gelegt. Es wird der Frage nachgegangen wie groß der Eingangsfehler sein darf, damit die Assistenz noch hinreichende Qualitätseigenschaften hinsichtlich seiner Präzision und Robustheit aufweist. Dazu erfolgt zunächst eine quantitative Betrachtung der Fehler anhand des kinematischen Modells. Danach wird eine qualitative Betrachtung anhand von systematischen Experimenten durchgeführt. Es wird zunächst ein Controller entwickelt, mit dem sich ein Manöver mithilfe der visuellen Informationen des Assistenz simulieren lässt. Dann wird eine Methode vorgestellt, mit deren Hilfe man das Manöver anhand definierter Fehlergrenzen bewerten kann. Um einen großen Raum möglicher Fehlerkombinationen effizient zu durchsuchen, wird das probabilistische Verfahren des Annealed Particle Filters benutzt. Mithilfe einer Testumgebung werden schließlich systematische Experimente durchgeführt. Zur weiteren Evaluation des Assistenzsystems in einer kontrollierten Umgebung erfolgte in Zusammenarbeit mit dem Fraunhofer ITWM in Kaiserslautern die Portierung des Assistenzsystems auf die dortige Simulationsumgebung RODOS®.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die vorliegende Masterarbeit entstand in der Arbeitsgruppe Echtzeitsysteme des Informatik-Fachbereichs an der Universität Koblenz-Landau. Einer der Forschungsschwerpunkte der Arbeitsgruppe Echtzeitsysteme liegt im Automotive-Bereich, konkret im autonomen und assistierten Fahren. Der Fokus der Betrachtung liegt dabei auf allgemeinen Gliederfahrzeugen, sogenannten *General-N-Trailern*, welche eine besondere Herausforderung in der Steuerbarkeit darstellen.

1.1 Motivation

Im Jahr 2011 wurde in Zusammenarbeit mit der Firma Orten [Orten, 2015] das Projekt "**petra** - prototyping and evaluation of tractor reverse driving assistance" initiiert. Dieses Projekt setzte sich zum Ziel,

> "ein Rückfahrassistenzsystem (RAS) in Form eines modularen Baukastensystems, bestehend aus Soft- und Hardwarekomponenten, konzeptioniert und in Form von Demonstratoren [umzusetzen]."[pet, 2015]

Im Rahmen dieses Projekts erfolgte über die folgenden 3 Jahre die Entwicklung eines optischen Fahrerassistenzsystems für Fahrzeuggespanne mit Einachsanhänger auf der Basis bereits in der Simulation erprobter Softwarekomponenten der Arbeitsgruppe.

Zum Ende des Projektes entstand der Wunsch nach einer Evaluation der Robustheit des Assistenz-Prototyps hinsichtlich der Präsenz von Modellfehlern. Bislang gab es

keine qualitative und quantitative Analyse, wie sich die angenommenen Modellparameter auf die Positionsgenauigkeit und Handhabbarkeit des Assistenz-Systems auswirken. Der Parameterraum des Modells und damit der potentiellen Fehler ist dabei hochdimensional: Neben den Parametern des kinematischen Modells spielen Wechselwirkungen innerhalb des Fahrzeugs zwischen Sensorik und Assistenz eine zentrale Rolle. Die Auswirkung und die Beeinflussung von Fehlern an verschiedenen Stellen des Datenflusses ist aufgrund der Komplexität der Zusammenhänge nicht direkt ersichtlich.

Im Rahmen des dritten **Commercial Vehicle Technology Symposium** in Kaiserslautern entstand zudem der Kontakt zum dortigen Fraunhofer ITWM. Es wurde diskutiert, ob man deren kommerziellen Fahrsimulator RODOS® zur weiteren Evaluation der Assistenz nutzen könnte. Durch eine Evaluation in einer hochwertigen und realistischen Simulationsumgebung kann das Fahrgefühl eines realen Fahrzeugs weitestgehend nachgebildet werden. In der Simulation lässt sich zudem auf exakte Positions- und Winkelwerte des Fahrzeugmodells zurückgreifen, was bei der Evaluation im realen Fahrzeug nicht möglich wäre.

Der Fokus dieser Arbeit besteht daher zunächst in der Simulaion verschiedener Fehlereinflüsse und deren Auswirkung auf die Robustheit der Fahrerassistenz. Ein solches Vorgehen erlaubt es Zusammenhänge und mögliche Grenzwerte für Fehler zu identifizieren. Des Weiteren erfolgt im Rahmen dieser Arbeit eine Portierung der entwickelten Fahrerassistenz auf die Simulationsumgebung RODOS®.

1.2 Ziele der Arbeit

Das entwickelte Fahrerassistenz-System dient dazu, den Fahrer eines Gespanns mit Einachsanhänger beim rückwärts rangieren zu unterstützen. Das Fahrverhalten lässt sich dabei anhand verschiedener Faktoren charakterisieren, etwa wie genau die anvisierte Position erreicht wurde, Eigenschaften der abgefahrenen Trajektorie sowie

des Lenkverhaltens. Das grundsätzliche Ziel der Arbeit ist die Analyse, wie sich Modellfehler im Fahrerassistenzsystem auf besagte Faktoren auswirken und wie groß der angenommene Fehler sein darf, damit das System noch hinreichende Qualitätseigenschaften aufweist. Konkret soll in dieser Ausarbeitung den folgenden Leitfragen nachgegangen werden:

- **Wie wirken sich Modellfehler auf die Güte der Navigation aus?** Diese Frage richtet sich sowohl nach dem Ausmaß als auch der Auswirkung der Fehler. Das Ausmaß beschreibt wie stark sich der Eingabefehler auf den Ausgabefehler überträgt. Die Auswirkung beschreibt, auf welche Weise und an welcher Stelle der Eingabefehler sich auf die Ausgabe auswirkt.
- **Wie groß dürfen Modellfehler sein, damit die Ausgabefehler ein tolerables Maß nicht überschreiten?** Diese Frage richtet sich nach den Fehlergrenzen des Ausgabefehlers. Es soll betrachtet werden, wie groß die Eingabefehler sein dürfen, sodass die Assistenz sich noch hinreichend korrekt verhält und handhabbar bleibt.

Zur Klärung dieser Fragen soll eine Testumgebung entwickelt werden, welche die Simulation von Fehlern in einem Manöver erlaubt. Aufgrund der hohen Dimension des Fehlers soll ein probabilistisches Verfahren genutzt werden, um den Fehlerraum möglichst effizient abzusuchen. Zur Entwicklung einer solchen Umgebung sind die folgenden Teilziele zu erreichen:

- Zunächst sollen die Fehlerquellen im Datenfluss zwischen Fahrzeug und Assistenz-System identifiziert werden. Auf Grundlage dessen soll eine zunächst analytische Betrachtung des Einflusses einzelner Fehler erfolgen. Eine solche Betrachtung erlaubt die Vorklassifizierung der Eingabefehler nach ihren Auswirkungen.
- Daraufhin soll ein geeignetes Verfahren modelliert und implementiert werden, welches eine Simulation des menschlichen Verhaltens bei der Nutzung der Assistenz darstellt. Dieses Verfahren soll den ”perfekten” Menschen simulieren, welcher Assistenz-Informationen unmittelbar in Steuerbefehle überträgt. Es soll

zudem die Basis für eine automatisierte Auswertung von Manövern bilden.

- Dann sollen Gütemaße zur Klassifizierung eines Manövers aus der Literatur ermittelt, gegenübergestellt und auf ihre Tauglichkeit untersucht werden. Dabei soll vor allem auf Maße für den Vergleich von Trajektorien sowie das Lenkverhalten eingegangen werden. Es soll möglich sein die einzelnen Gütemaße zu einer einzigen Maßzahl zu vereinen.

- Für die experimentelle Auswertung von Fehlereinflüssen soll in der Folge ein Testtool entwickelt werden. Die folgenden Komponenten werden dazu benötigt:

  – Die Fehler im Datenfluss der Assistenz sollen in der Software modelliert werden, sodass eine Simulation des Eingangsfehlers mittels einer Fehlerkonfiguration ermöglicht wird.

  – Es soll möglich sein die Fehlerkonfiguration mithilfe der automatisierten Ausführung eines Manövers zu gewichten.

  – Zur Betrachtung eines möglichst großen Raums verschiedener Fehlerkombinationen soll dazu das probabilistische Optimierungsverfahren des Annealed Particle Filters genutzt werden. Es soll dabei möglich sein, die Eingabefehler anhand definierter Zielvorgaben des Ausgabefehlers zu optimieren.

- Anschließend sollen mithilfe der Testumgebung Experimente durchgeführt werden, welche die Zusammenhänge der zuvor genannten Leitfragen untersuchen sollen. Anschließend sollen die Experimente mit Hinblick auf die Leitfragen evaluiert werden. Ziel ist es, eindeutige Aussagen zu Auswirkungen und Fehlergrenzen treffen zu können.

Der zweite Teil der Arbeit befasst sich mit der Kooperation mit dem Fraunhofer Institut in Kaiserslautern. Um eine Evaluation auf dem dortigen Simulator zu ermöglichen, müssen die im Zusammenhang mit dem Assistenzsystem entwickelten Softwarekomponenten auf das dortige System portiert werden. Die folgenden Teilziele können

definiert werden:

- Die Assistenz soll als **Blackbox** auf einem externen Kabinenrechner laufen. Sie kommuniziert lediglich über Netzwerkschnittstellen mit der Simulationsumgebung. Dies ermöglicht eine Evaluierung des abgeschlossenen Systems.
- Es soll ein Fahrzeugmodell entwickelt werden, welches in der Simulationsumgebung RODOS® genutzt werden kann. Dazu soll die vorhandene Kinematikbibliothek der Arbeitsgruppe in ein Simulink-Modell übertragen werden.
- Eine Visualisierung des Fahrzeugs erfordert dann die Portierung der 3D-Modelle verschiedener Fahrzeuggespanne der Arbeitsgruppe auf die Grafikengine, welche die dortige Simulationsumgebung nutzt.
- Schließlich sollen Netzwerkschnittstellen implementiert werden, welche die Kommunikation der Assistenz mit der Simulationsumgebung und dem Fahrzeugmodell ermöglichen.
- Den Abschluss soll eine Evaluation der Assistenz auf dem Simulator bilden. Dazu sollen menschliche Fahrer verschiedene Manöver einmal mit und einmal ohne Assistenz durchführen. Die Resultate sollen in der Folge anhand verschiedener Kriterien ausgewertet werden.

1.3 Einordnung in den wissenschaftlichen Kontext

Die Analyse von Fahrerassistenzsystemen ist in der Literatur vor allem durch Betrachtungen von sicherheitsrelevanten Kriterien vertreten [Hummel u. a., 2011] [Aust u. a., 2011] [Carsten u. Nilsson, 2001], vor allem im Kontext von Spurhalte- und Bremsassistenzen. Diese Assistenzsysteme kommen in einem hochdynamischen Umfeld, etwa einer Autobahn, zum Einsatz. Eine korrekte Funktionsweise dieser Assistenzsysteme ist daher sicherheitskritisch. Beeinflussen externe oder interne Fehler das System, so kann es sehr schnell zu einer Gefährdung des Fahrers führen.

Andere Assistenzsysteme, welche in weniger dynamischen und besser kontrollierbaren Umgebungen genutzt werden, umfassen Parkassistenten oder Rückfahrassistenzsysteme. In der Literatur finden sich wenige Beispiele nur von der Durchführung einer Fehleranalyse solcher Assistenzsysteme. In [Sang u. a., 2011] erfolgt lediglich die Evaluierung des genutzten Algorithmus zur Erkennung von Hindernissen im Kamerabild. Auch in [Vestri u. a., 2005] wird nur auf die Robustheit des Bildverarbeitungsalgorithmus eingegangen. Eine Betrachtung eines Assistenz-Systems für Fahrzeuggespanne erfolgt in [Roh u. Chung, 2010]. Hierbei wird zwar auf Positionierung und die zurückgelegte Trajektorie eingegangen, allerdings werden Modellfehler gänzlich ignoriert.

Neben der direkten Betrachtung der Assistenz-Systeme existieren viele Veröffentlichungen, welche Modellfehler auf unterster Kontroll- und Sensorebene untersuchen. In [Kelly, 2002] wird beispielsweise betrachtet wie sich Fehler in der Fahrzeugodometrie mit der Zeit fortpflanzen. In [Chung u. a., 2011] wird betrachtet wie sich Fehler in der Messung des Einknickwinkels eines Fahrzeuggespanns auf die Poseschätzung des Anhängers auswirkt.

Der betrachtete Ansatz zur Evaluierung der Fahrerassistenz stellt somit ein bislang wenig erforschtes Gebiet dar. Daher kann nicht auf bestehende Verfahren zur Evaluation zurückgegriffen werden.

1.4 Aufbau der Arbeit

Zunächst sollen in Kapitel 2 einige Grundlagen sowie der Stand der Wissenschaft behandelt werden. Dazu gehört die Vorstellung des Fahrerassistenzsystems sowie des kinematischen Modells, auf dem es basiert. Es folgen verschiedene Verfahren, mit deren Hilfe man die Fortpflanzung von Eingangsfehler in einem nichtlinearen Modell untersuchen kann. Des Weiteren werden Gütemaße vorgestellt, mit denen man zum einen die Ähnlichkeit zweier Trajektorien charakterisieren, zum anderen Aussa-

gen zur Güte des Lenkverhaltens treffen kann. Den Abschluss bildet die Vorstellung des Annealed Particle Filters, einem heuristischen Optimierungsverfahren für hochdimensionale Suchprobleme.

In Kapitel 3 erfolgt eine analytische Betrachtung der Auswirkung von Fehlern, welche im Datenfluss der Assistenz identifiziert wurden. Danach wird das Konzept eines Controllers dargelegt, welches die Simulation einen ”perfekten” Nutzer der Assistenz ermöglicht. Einer Auswertung geeigneter Metriken folgt die Vorstellung des Testtools, welches eine Simulation verschiedener Fehlereinflüsse ermöglicht. Den Abschluss dieses Kapitels bildet die Vorstellung der für die Portierung auf RODOS® entwickelten Softwarekomponenten.

Kapitel 4 befasst sich mit der Beschreibung und Durchführung von Experimenten, auf welche in der in Kapitel 5 folgenden Evaluation näher eingegangen wird.

Im abschließenden Kapitel 6 erfolgt eine Zusammenfassung der Ergebnisse dieser Arbeit sowie ein Ausblick auf Ansätze zur Verbesserung des Evaluations-Verfahren sowie das weitere Vorgehen in der Zusammenarbeit mit dem Fraunhofer Institut.

2 Stand von Wissenschaft und Technik

In diesem Kapitel werden die für die Arbeit relevanten Verfahren und Vorgehensweisen vorgestellt. Zunächst wird eine Übersicht über die Funktionsweise der Fahrerassistenz gegeben sowie das zugrundeliegende Fahrzeugmodell für *General-N-Trailer* erläutert. Danach erfolgt die Vorstellung in der Literatur stark vertretener Gütemaße, welche zur Charakterisierung eines Manövers dienen. Der nächste Abschnitt befasst sich mit grundlegenden Techniken zur Analyse von Fehlerfortpflanzungen in nichtlinearen Modellen. Den Abschluss bildet eine Vorstellung des RODOS®-Simulationssystems des Fraunhofer ITWM Kaiserslautern.

2.1 Fahrerassistenz-System der AGRT

2.1.1 Kinematisches Modell

Eine Modellierung von allgemeinen Gliederfahrzeugen erfolgt in der Arbeitsgruppe Echtzeitsysteme durch das sogenannte Einspurmodell [Zöbel, 2013]. Die Bewegung des Fahrzeugs wird durch die Gleichungen von Altafini [Altafini, 2001] beschrieben.

Die folgenden betrachteten Zusammenhänge orientieren sich vor allem an den entsprechenden Abschnitten der Diplomarbeit von Christian Schwarz [Schwarz, 2009] sowie der Masterarbeit von Christian Eiserloh [Eiserloh, 2013], welche ebenfalls in der Arbeitsgruppe entstanden sind.

Beschreibung von allgemeinen Gliederfahrzeugen

Das Einspurmodell eignet sich zur Beschreibung allgemeiner Gliederfahrzeuge (*General-N-Trailer*), welche sich mit niedriger Geschwindigkeit bewegen [Zöbel, 2013][Schwarz, 2009][Eiserloh, 2013]. Es erfolgt eine Beschränkung auf das kinematische Modell des Fahrzeugs. Eine Betrachtung der bei der Bewegung wirkenden Kräfte findet nicht statt. Ebenso wird der Einfluss von Schlupf vernachlässigt, was zu der Annahme führt, dass die Bewegung des Fahrzeugs direkt und einzig von den Steuergrößen abhängt.

Ein Fahrzeuggespann wird als Kette von Fahrzeuggliedern beschrieben. Nachfolgende Fahrzeuggleder sind dabei über eine Kupplung miteinander verbunden. Jedes Fahrzeuggleid bekommt als weiteren Abstraktionsschritt genau eine Achse zugewiesen. Besitzt ein reales Fahrzeug mehrere Achsen, welche starr miteinander verbunden sind, so wird diese im Modell durch eine virtuelle Achse ersetzt, welche genau in der Mitte zwischen den ursprünglichen Achsen liegt. Die Lenkachse wird fortan als separates Fahrzeuggleid betrachtet, wodurch ein Zugfahrzeug, etwa ein PKW, automatisch aus zwei Fahrzeuggliedern besteht.

Der letzte Abstraktionsschritt besagt, dass Räder den Boden in genau einem Punkt berühren. Die Räder einer Achse werden in der Folge durch ein einziges virtuelles Rad in der Mitte der ursprünglichen Räder ersetzt. Die Ausrichtung des Rads entspricht der mittleren Ausrichtung der ursprünglichen Räder. Das Fahrzeug besitzt somit nur noch eine Spur, daher die Bezeichnung “Einspurmodell”. Abbildung 2.2 zeigt die Bezeichner für die Größen im Fahrzeug. Die Fahrzeuggleder werden einzeln betrachtet und folgen einer einheitlichen Beschreibung im Modell. Beginnend bei 0 erhält jedes Fahrzeuggleid zunächst einen Index zugewiesen.

Die Beschreibung des i-ten Fahrzeuggleids erfolgt durch folgende Werte:

- (x_i,y_i) beschreibt die **Position** des Fahrzeuggleids anhand seines Achsmittelpunkts.

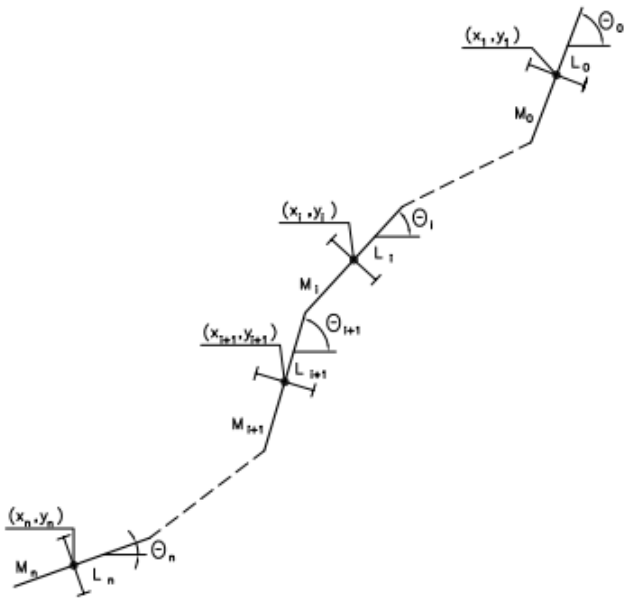


Abbildung 2.1: Bezeichner der Größen im Einspurmodell, aus [Schwarz, 2009].

- Θ_i beschreibt die **Ausrichtung** des Fahrzeugglieds.
- L_i beschreibt den Abstand des Achsmittelpunkts zum vorderen Kupplungspunkt, welcher als **vorderer Kupplungsoffset** bezeichnet wird. Liegt die Kupplung vor der Achse, ist der Wert positiv, ansonsten negativ.
- M_i beschreibt den Abstand des Achsmittelpunkts zum hinteren Kupplungspunkt, welcher als **hinterer Kupplungsoffset** bezeichnet wird. Liegt die Kupplung hinter der Achse, ist der Wert positiv, ansonsten negativ.

Ein Modell mit n Fahrzeuggliedern wird als *General-(n-1)-Trailer* bezeichnet. Fällt der hintere Kupplungspunkt eines Fahrzeugglieds i mit der Achse zusammen, gilt $M_i = 0$. Ist diese Bedingung für alle Fahrzeuglieder $i < n$ erfüllt, spricht man von einem *Standard-(n-1)-Trailer*.

Der aktuelle Fahrzeugzustand wird durch eine Fahrzeugkonfiguration beschrieben, welche redundanzfreie Informationen zur aktuellen Lage des Fahrzeugs beinhaltet. Dazu gehören die Position des ersten Fahrzeugglieds, die Ausrichtung aller Fahrzeug-

glieder sowie der Lenkwinkel. Aus diesen Informationen lassen sich mithilfe der zuvor erläuterten Fahrzeugbeschreibung die Positionen aller Fahrzeuggelieder bestimmen. Es ist zudem sinnvoll, die Ausrichtungen der einzelnen Fahrzeuggelieder als relative Ausrichtungen in Bezug auf das Vorgängerfahrzeug zu betrachten [Schwarz, 2009].

Die resultierenden Lenk- und Einknickwinkel sind folgendermaßen definiert:

- $\alpha_{L_1} := \Theta_0 - \Theta_1$ beschreibt den **Lenkwinkel** des Fahrzeugs.
- $\Delta\Theta_{i(i+1)} := \Theta_{i+1} - \Theta_i$ beschreibt den i-ten **Einknickwinkel** des Fahrzeugs, d.h. den Einknickwinkel zwischen i-tem und $(i+1)$-tem Fahrzeuggelied.

Beschreibung der Bewegung von allgemeinen Gliederfahrzeugen

Basierend auf den Erkenntnissen der vorangegangenen Abschnitts lässt sich im Folgenden die Bewegung allgemeiner Gliederfahrzeuge beschreiben. Dies erfordert zunächst die Einführung eines weiteren Bezeichners [Schwarz, 2009]:

- v_i stellt die Geschwindigkeit des i-ten Fahrzeuggelieds in Richtung Θ_i dar. Der Wert ist positiv, wenn sich das Fahrzeug in Richtung seiner Ausrichtung vorwärts bewegt, und entsprechend negativ, wenn es sich rückwärts bewegt.

Die Beschreibung der Bewegung basiert auf den in [Schwarz, 2009] und [Altafini, 2001] vorgestellten Differentialgleichungen:

$$\dot{\Theta}_{i+1} = \frac{1}{L_{i+1}} \cdot \Big(v_i \cdot \sin(\Theta_i - \Theta_{i+1}) - M_i \cdot \cos(\Theta_i - \Theta_{i+1} \Big) \cdot \dot{\Theta}_i \tag{2.1}$$

$$v_{i+1} = v_i \cdot \cos\left(\Theta_i - \Theta_{i+1}\right) + M_i \cdot \sin\left(\Theta_i - \Theta_{i+1}\right) \cdot \dot{\Theta}_i \tag{2.2}$$

$$\dot{x}_i = v_i \cdot \cos\left(\Theta_i\right) \tag{2.3}$$

$$\dot{y}_i = v_i \cdot \sin\left(\Theta_i\right) \tag{2.4}$$

Als Anfangswerte lassen sich v_0 und $\dot{\theta}_0$ bei gegebenem $\dot{\alpha}_{L_1}$ und v_1 folgendermaßen berechnen:

$$v_0 = \frac{M_0\Big(v_1 \cdot \cos\left(\alpha_{L_1}\right) - L_1 \cdot \dot{\alpha}_{L_1} \cdot \sin\left(\alpha_{L_1}\right)\Big) + L_1 \cdot v_1}{L_1 \cdot \cos\left(\alpha_{L_1}\right) + M_0} \tag{2.5}$$

$$\dot{\Theta}_0 = \frac{v_1 \cdot \sin\left(\alpha_{L_1}\right) - L_1 \cdot \dot{\alpha}_{L_1} \cdot \cos\left(\alpha_{L_1}\right)}{L_1 \cdot \cos\left(\alpha_{L_1}\right) + M_0} \tag{2.6}$$

Aus den gegebenen Anfangswerten lassen sich mithilfe der Gleichungen 2.1 - 2.4 die Änderungen in Position und Ausrichtung rekursiv für alle folgenden Fahrzeuggliedern berechnen.
Die Änderung der Konfiguration des Gespanns lässt sich in der Folge berechnen, sofern die vorangegangene Konfiguration sowie der Lenkwinkel α_{L_1} und die Geschwindigkeit v_1 des Zugfahrzeugs gegeben sind. Die letzten beiden Werte werden auch als die Steuergrößen des Fahrzeuggespanns bezeichnet.

2.1.2 Kinematikbibliothek ezKine

Die in der Arbeitsgruppe genutzte Kinematikbibliothek *ezKine* basiert auf der Masterarbeit von Christian Eiserloh [Eiserloh, 2013]. Die Entwicklung dieser Bibliothek zielte vor allem auf Schnelligkeit und eine einfach Handhabung der Kinematik-Berechnungen ab. Vorgestellt werden soll an dieser Stelle die Fahrzeugdatenstruktur, welcher die im vorangegangenen Abschnitt beschriebene Modellierung zugrunde liegt.

Die Fahrzeugdatenstruktur umfasst die Klassen *Vehicle*, *VehicleUnit* und *Coupling*, deren Relation in Abbildung 2.2 schematisch dargestellt wird. Die Klasse *Vehicle* repräsentiert ein Fahrzeuggespann, welches aus einer beliebigen Anzahl an Fahrzeuggliedern bestehen kann. Ein solches Fahrzeugglied wird durch die Klasse *VehicleUnit* beschrieben. Die Klasse *Coupling* stellt eine Verknüpfung zwischen zwei Fahrzeuggliedern dar.

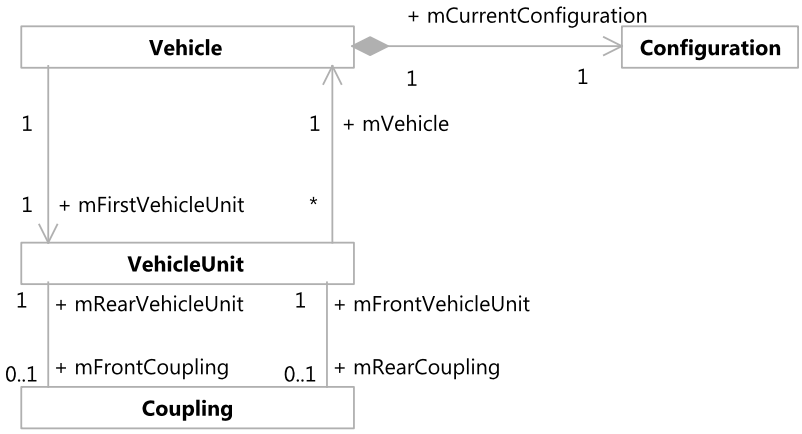


Abbildung 2.2: Zusammensetzung der Fahrzeugdatenstruktur.

Die Klasse **VehicleUnit** verwaltet zum einen die geometrischen Abmessungen eines Fahrzeuggliedes. Diese Abmessungen umfassen die statischen Fahrzeugparameter:

- **mLength**: Beschreibt die Länge des Fahrzeuggliedes (inklusive Deichsel bei Anhängern)
- **mWidth**: Beschreibt die Breite des Fahrzeuggliedes
- **mRearOverhang**: Beschreibt den Abstand zwischen Hinterachse und Heck des Fahrzeugglieds
- **mFrontCouplingOffset**: Beschreibt den Abstand von der Hinterachse des Fahrzeugglieds zur Lenkachse (beim Zugfahrzeug) bzw. zur vorderen Kupplung (bei Anhängern)
- **mFrontVehicleUnitOffset**: Beschreibt den Abstand der vorderen Kupplung des Fahrzeugglieds zur Hinterachse des vorherigen Fahrzeugglieds (nur für Anhänger)

Zum anderen hält die Klasse Referenzen auf sowohl die vordere als auch die hintere Kupplung des Fahrzeugglieds über je eine Instanz der Klasse Coupling. Die

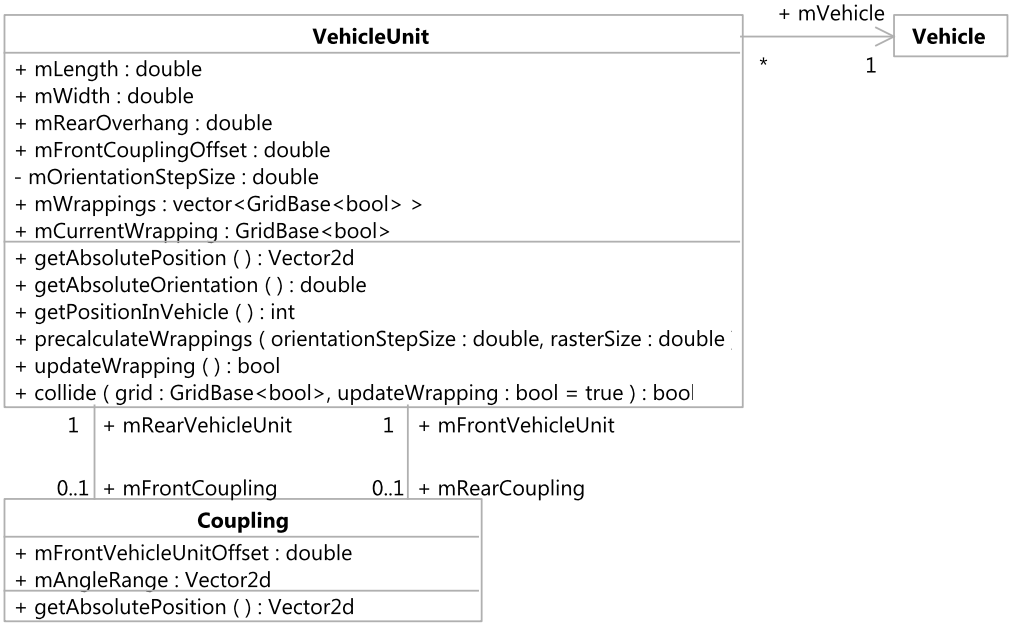


Abbildung 2.3: Die Klassen `VehicleUnit` und `Coupling`.

Coupling-Klasse wiederum hält Referenzen auf das vorherige und das nachfolgende Fahrzeuggelied und stellt so eine Verknüpfung zwischen den Fahrzeuggliedern dar, siehe Abbildung 2.3. Durch das Zusammenspiel dieser beiden Klassen wird eine strukturierte Darstellung und Verwaltung der Fahrzeugparameter des Gespanns ermöglicht. Aufgrund der Tatsache, dass Fahrzeuggelieder jeweils indirekt Referenzen auf ihre Vorgänger und Nachfolger halten, kann man durch Kenntnis eines einzigen Fahrzeuggelieds durch das gesamte Gespann navigieren.

Die Klasse **Vehicle** schließlich dient als Schnittstelle des Fahrzeuggespanns nach außen und hält eine Referenz auf das erste Fahrzeuggelied, das Zugfahrzeug. Dies ermöglicht es der Klasse indirekt auf alle nachfolgenden Fahrzeuggelieder und Kupplungen zuzugreifen. Die Klasse stellt Methoden bereit, um diese Fahrzeuggelieder und Kupplungen zu erzeugen. Ein Fahrzeuggespann benötigt dabei genau ein Zugfahrzeug, welchem eine beliebige Anzahl an Anhängern folgen kann. Die Struktur der

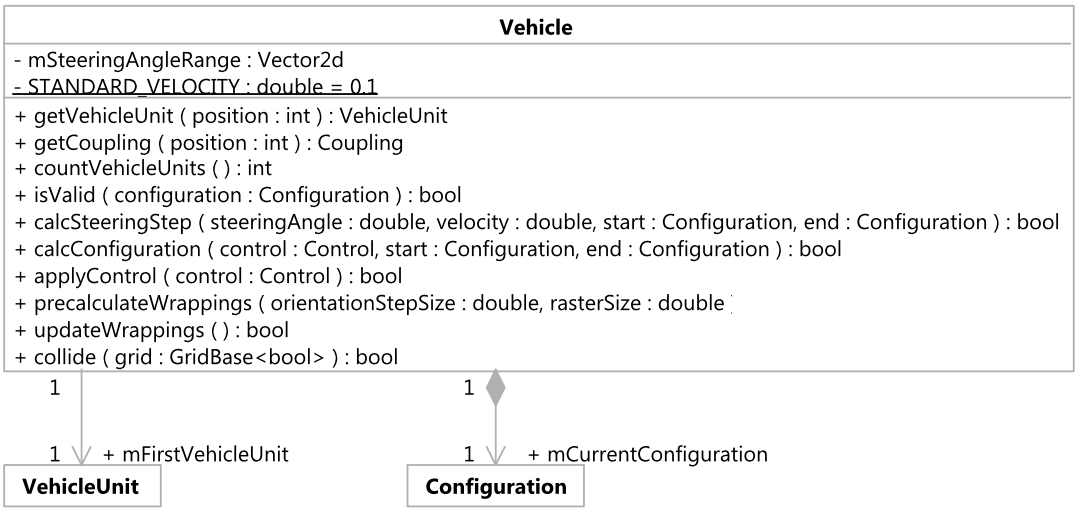


Abbildung 2.4: Die Klasse `Vehicle`.

Klasse ist in Abbildung 2.4 zu sehen.

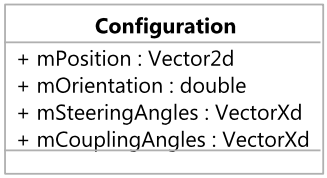


Abbildung 2.5: Die Klasse `Configuration`.

In der Klasse Vehicle finden zudem alle kinematischen Berechnungen statt, welche das Fahrzeuggespann betreffen. Der aktuelle Zustand des Fahrzeugs wird in einem Objekt vom Typ **Configuration** gespeichert (Abbildung 2.5). Dieser setzt sich aus den folgenden Datenfeldern zusammen:

- **mPosition**: Die aktuelle (2-dimensionale) Position des Zugfahrzeugs als Referenzposition für das gesamte Gespann
- **mOrientation**: Die aktuelle Ausrichtung des Zugfahrzeugs als Referenzausrichtung des gesamten Gespanns

- **mSteeringAngles**: Lenkwinkel des Zugfahrzeugs und ggf. aller gelenkter Achsen der Anhänger
- **mCouplingAngles**: Einknickwinkel zwischen allen Fahrzeuggliedern, entspricht der Anzahl an Kupplungen

Diese Klasse enthält alle relevanten Größen, um den Zustand des Fahrzeugs eindeutig zu beschreiben. Verwandte Größen wie die Position des Anhängers lassen sich aus dem Zustand sowie der Fahrzeuggeometrie aus 2.1.2 berechnen.

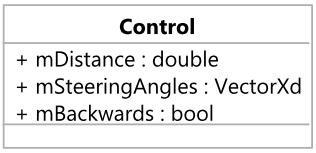


Abbildung 2.6: Die Klasse `Control`.

Eine Änderung des Fahrzeugzustands erfolgt durch die Anwendung von Steuergrößen auf das Fahrzeug. Diese Steuergrößen werden in der Klasse **Control** (Abbildung 2.6) zusammengefasst und beschreiben eine Bewegung des Fahrzeugs. Sie umfassen:

- **mDistance**: Die abzufahrende Gesamtdistanz des Bewegungsschritts
- **mSteeringAngles**: Die Lenkwinkel, unter denen die Bewegung ausgeführt werden soll
- **mBackwards**: Flag, welches die Fahrtrichtung der Bewegung bestimmt, *true* entspricht einer Rückwärtsbewegung, *false* einer Vorwärtsbewegung

Die Berechnung der Auswirkung dieser Bewegung erfolgt wie zuvor erwähnt in der Klasse **Vehicle**. Die Berechnung unterliegt dabei einigen Beschränkungen wie der Beschränkung von Lenk- und Einknickwinkeln, welche in der Initialisierung der einzelnen Fahrzeugglieder angegeben werden können.

2.1.3 Optische Fahrerassistenz

Gesamtkonzept

Ein Hauptanliegen der Arbeitsgruppe Echtzeitsysteme der Universität Koblenz ist die Erforschung und Entwicklung von Fahrerassistenzsystemen, welche Fahrer von **mehrgliedrigen Fahrzeuggespannen** beim Rangieren des Fahrzeugs unterstützen können. Die im Rahmen dieser Arbeit betrachtete Ausprägung der Assistenz umfasst **visuelle Hilfestellungen** für das Rangieren mit einem Einachsanhänger. Eine am Heck des Anhängers befestigte Rückfahrkamera überträgt ein Bild auf einen kleinen Monitor in der Fahrerkabine. Zusätzlich zu der Rückansicht des Gespanns werden dem Fahrer Hilfslinien zur Verfügung gestellt, welche ihm Informationen liefern, wie sich das Fahrzeug bei der Rückwärtsfahrt kinematisch verhält. Zwei Arten von Hilfslinien werden dabei unterschieden: die Gespann-Trajektorie sowie die Anhänger-Trajektorie. Die Berechnung dieser beiden Trajektorien hängt zum einen von statischen Fahrzeugparametern wie der Fahrzeuggeometrie aus 2.1.2 ab, zum anderen von dynamischen Sensorwerten, welche Lenk- und Einknickwinkel liefern. Die Gespann-Trajektorie wird durch zwei blaue Kurven visualisiert, welche am Aufstandspunkt des linken bzw. rechten Rad des Anhängers beginnen. Durch die Einblendung dieser Trajektorie wird dem Fahrer mitgeteilt, wie sich das Fahrzeug auf einer stabilen Kreisbahn mit dem aktuell anliegenden Einknickwinkel bewegen würde. Die Bewegung auf einer stabilen Kreisbahn ist dabei äquivalent zu einer Bewegung mit konstantem Einknickwinkel. Den hierbei angenommenen Lenkwinkel nennt man den stabilen Lenkwinkel. Die Anhänger-Trajektorie wiederum wird durch eine grüne Kurve repräsentiert, welche im Mittelpunkt der Hinterachse des Anhängers beginnt. Diese Trajektorie teilt dem Fahrer mit, wie sich der Anhänger unter dem aktuell anliegenden Lenkwinkel bewegen würde.

Durch die Einblendung dieser beiden Trajektorien wird der Fahrer in zweifacher Weise bei der Rückwärtsfahrt unterstützt. Zum einen erlaubt ihm die Anhänger-Trajektorie

das Fahrzeug zu stabilisieren, zum anderen kann er die Gespann-Trajektorie dazu benutzten, um ein bestimmtes Ziel anzuvisieren:

- Um das Gespann bei der Rückwärtsfahrt zu stabilisieren, muss der Fahrer die Ausrichtung der Anhänger-Trajektorie durch Lenkbewegungen so ändern, dass sie mittig zwischen den beiden Kurven der Gespann-Trajektorie liegt. Die Zentrierung der Anhänger-Trajektorie ist äquivalent zur Einstellung des stabilen Lenkwinkels, der zum aktuellen Einknickwinkel gehört. Ist die stabile Bahn erreicht, so sind keine weiteren Lenkbewegungen notwendig, um das Fahrzeug auf dieser Bahn zu halten.
- Möchte der Fahrer ein bestimmtes Ziel anfahren, so stehen ihm zwei Möglichkeiten offen, dieses ”anzuvisieren”. Da die Anhänger-Trajektorie direkt am Lenkrad hängt, kann diese durch eine Korrektur des Lenkwinkels direkt auf das Ziel ausgerichtet werden. Eine weitere Möglichkeit ist, die Gespann-Trajektorie auf das Ziel auszurichten, was durch eine Veränderung des Einknickwinkels bewirkt wird. Die Gespann-Trajektorie folgt dabei der Richtung der Anhänger-Trajektorie und wird von ihr wie ein Gummiband hinterher gezogen. Je stärker die Abweichung der beiden Trajektorien, desto schneller folgt sie der Anhänger-Trajektorie.

Architektur

Das zuvor beschriebene Fahrerassistenzsystem lag im Verlauf dieser Arbeit als Prototyp vor, es erfolgt eine Betrachtung der Implementierung, welche zum 1.9.2014 zur Verfügung stand. Sämtliche Änderungen nach diesem Zeitpunkt betreffen Implementierungsdetails, jedoch nicht das grundlegende Konzept der Trajektorienberechnung.

Im folgenden wird die Architektur der Assistenzkomponente erläutert. Die Assistenz nutzt verschiedene Softwarekomponenten, welche in der Arbeitsgruppe ent-

wickelt wurden. In Kapitel 2.1.2 wurde bereits die Kinematik-Komponente *ezKine* vorgestellt, welche zur Berechnung der Hilfs-Trajektorien genutzt wird. Über die Komponente *ezCom*, auf die im folgenden Kapitel näher eingegangen wird, wird die Netzwerkkommunikation mit anderen Applikationen ermöglicht. Die eigentliche Assistenz-Komponente koordiniert den Empfang neuer Daten und Sensorwerte und sorgt für eine regelmäßige Aktualisierung der angezeigten Trajektorien. Der grundlegende Datenfluss im Fahrzeug ist in Abbildung 2.7 dargestellt. Die Assistenzlogik

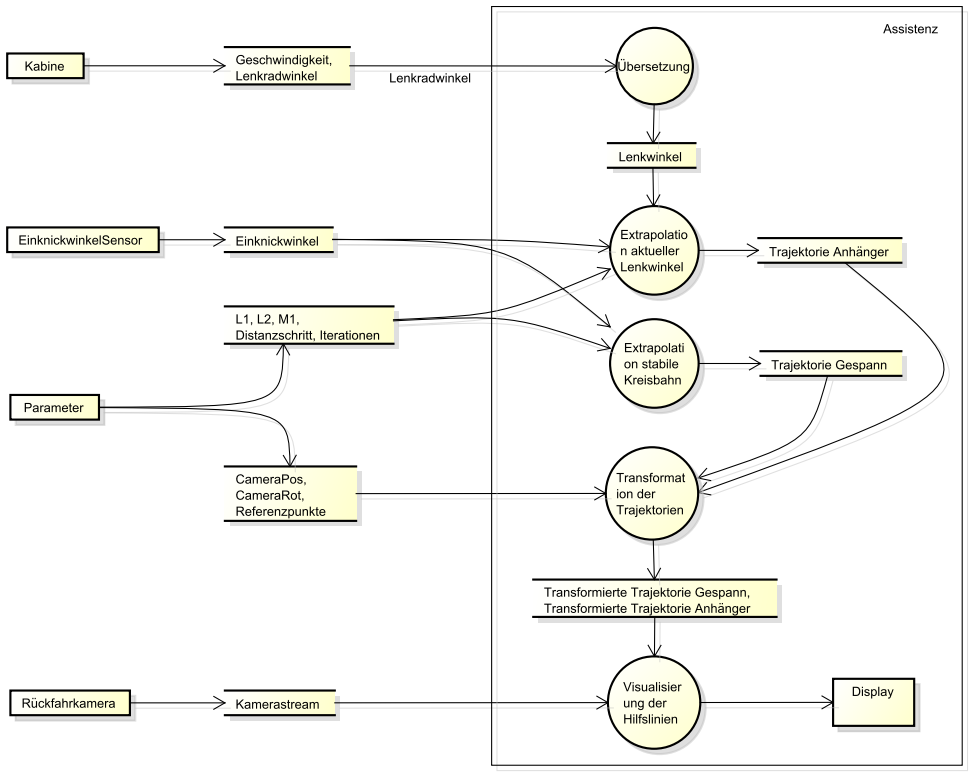


Abbildung 2.7: Datenfluss zwischen Fahrzeug und Assistenz.

findet sich in der Klasse *AssistanceController*. Abbildung 2.8 zeigt den Ablauf eines Aktualisierungsschritts des Assistenzsystems. Über eine TCP-Server-Anbindung durch *ezCom* empfängt sie asynchrone Nachrichten, welche jeweils aktualisierte Datenwerte von verschiedenen Datenquellen enthalten. Der modulare Aufbau der Assistenz erlaubt eine flexible Anbindung an verschiedene Datenquellen, welche allein die geforderten Datentypen liefern müssen. Die Assistenz benötigt ansonsten keine

Kenntnis über den Sender der Daten. Bei dem *CameraFrameProvider* kann es sich demnach entweder um eine echte Kamera handeln oder um eine virtuelle Kamera aus einer Simulationsumgebung. Ebenso kann es sich bei den Datenquellen *SteeringAngleProvider* und *CouplingAngleProvider* um echte oder simulierte Sensorik handeln. Dies erlaubt es, das System zunächst in einer simulierten Fahrzeugumgebung zu testen und es danach ohne Änderungen in ein reales Fahrzeug einzubetten.

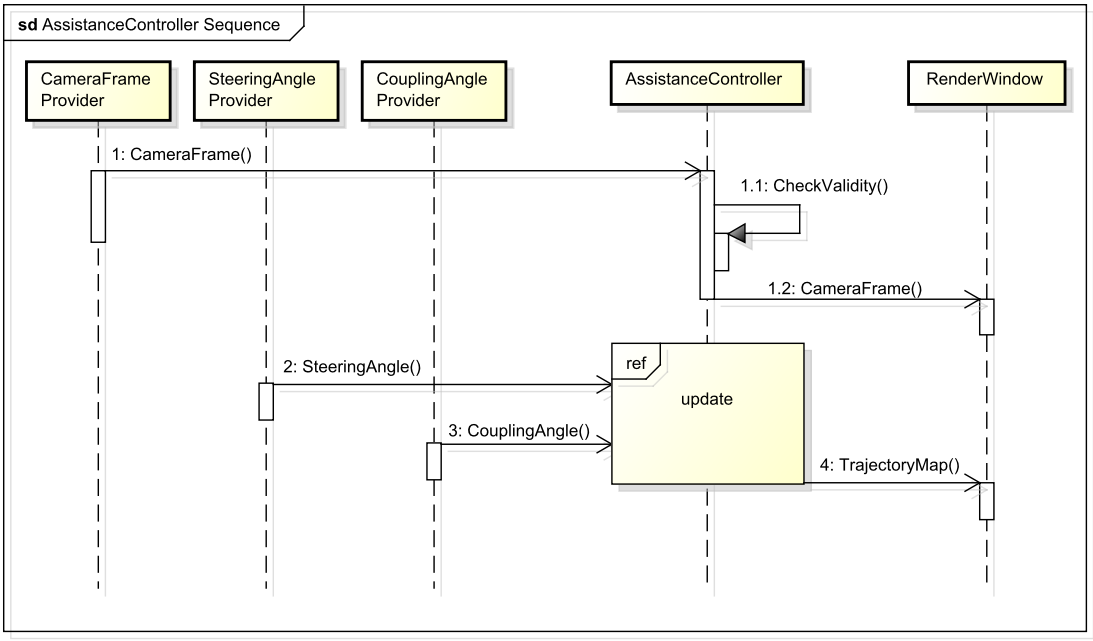


Abbildung 2.8: Datenkommunikation des Assistenzsystems. Das *RenderWindow* beschreibt dabei ein Fenster, welches zur Visualisierung genutzt wird.

Anbindung an ezCom

Die Netzwerkbibliothek *ezCom* erlaubt es Clients über einen Server mittels **TCP-Nachrichten** zu kommunizieren. Eine Nachricht besitzt einen Header, welcher Meta-Informationen der versendeten Nachricht beinhaltet und einen Body, welcher die serialisierten, zu übertragenden Daten enthält. Der Body kann jeden beliebigen Datentyp als Datenfeld besitzen, sofern dieser Datentyp einen Serialisierungs-Operator besitzt. Die Nachricht kann durch drei spezielle Datenfelder im Header identifiziert

werden:

- *DataType*: Der Datentyp, den das Datenfeld des Nachrichten-Bodys hat.
- *ContentType*: Eine semantische Beschreibung des Inhalts der übertragenen Daten.
- *ProviderType*: Eine Beschreibung der Datenquelle bzw. des Senders der Nachricht.

Um einen dieser Typen zur Identifikation des Inhalts der Nachricht nutzen zu können, muss man den Typ zunächst mit dem Server registrieren. Man erhält daraufhin eine ID, welche sich mit den IDs im Header einer empfangenen Nachricht vergleichen lässt. Es ist wichtig zu überprüfen, dass der tatsächliche Datentyp der Nachricht mit dem erwarteten Datentyp übereinstimmt, da sonst die Deserialisierung der Daten fehlschlagen würde.

Der CameraFrameProvider sendet Bilddaten mit ProviderType *trailer-rear-camera*, was der Rückfahrkamera entspricht, ContentType *camera-frame*, einem einzelnen aufgenommenen Frame der Kamera, und DataType *cv-mat*, einer Klasse zur Repräsentation von Bilddaten aus der Bildverarbeitungs-Bibliothek OpenCV [Bradski]. Der *AssistanceController* empfängt jeden einzelnen Camera-Frame des Providers, überprüft diesen auf seine Validität und speichert ihn daraufhin für die Weiterverarbeitung.
Sowohl SteeringAngleProvider als auch CouplingAngleProvider senden Winkelwerte mit beliebigem ProviderType, ContentType *steering-angle* bzw. *bending-angle* und jeweils dem DataType *double*. Diese beiden Werte werden nach dem Empfang gespeichert und werden erst zu Beginn des nächsten Aktualisierungszyklus weiterverarbeitet.

Berechnung der Trajektorien

Unabhängig vom Empfang der oben beschriebenen Nachrichten findet im AssistanceController eine regelmäßige Aktualisierung der eingeblendeten Trajektorien statt. Abbildung 2.9 zeigt die Methode *update*, welche die Neuberechnung der Trajektorien durchführt auf Grundlage der zuvor empfangenen Lenk- und Einknickwinkel.

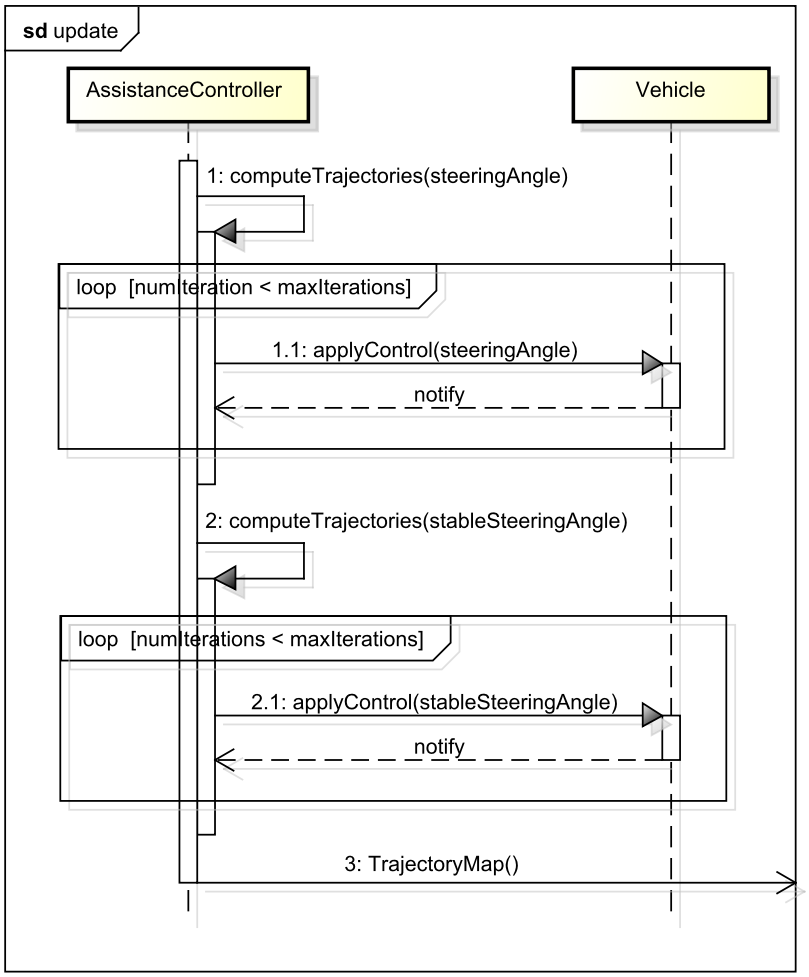


Abbildung 2.9: Aktualisierungsschritt des Assistenzsystems.

Die Methode nutzt eine Instanz der in Kapitel 2.1.2 beschriebenen Vehicle-Klasse der Softwarebibliothek **ezKinematics**. Dieses Vehicle-Objekt hat dieselben Abmessungen wie das reale Fahrzeug, in welche die Assistenz eingebettet ist. Wie in Kapitel

2.1.2 erläutert wird die Bewegung eines Vehicles durch ein Control-Objekt beschrieben. Um eine Trajektorie zu berechnen, werden daher wiederholt Control-Objekte auf das Vehicle angewandt. Die Anzahl der Wiederholungen richtet sich nach der gewünschten Länge der eingeblendeten Trajektorien. Über ein **Observer-Pattern** teilt das Fahrzeug einer Logging-Methode mit, dass sich der Fahrzeugzustand geändert hat:

- **Anhänger-Trajektorie**: Es wird eine einzige Trajektorie für die Position der Anhänger-Hinterachse gespeichert. Auf das Fahrzeug wird ein Control-Objekt mit dem aktuellen Lenkwinkel angewandt. Die neue Position des Anhängers wird der bisherigen Trajektorie hinzugefügt.
- **Gespann-Trajektorien**: Es werden zwei Trajektorien gespeichert, eine für das linke Hinterrad und eine für das rechte. Auf das Fahrzeug wird ein Control-Objekt mit dem stabilen Lenkwinkel angewandt, welcher über den aktuellen Einknickwinkel berechnet werden kann. Die neuen Positionen des linken und des rechten Anhängerrads werden ihren jeweiligen Trajektorien hinzugefügt.

Die resultierende Trajectory-Map enthält alle berechneten Kurven, welche später auf das Kamerabild gelegt werden. Die Kurven können dabei beliebigen Transformationen unterzogen werden, um an verschiedenen Stellen in verschiedenen Kamerabildern angezeigt zu werden.

2.2 Gütemaße für die qualitative Bewertung von Trajektorien

Führt ein Nutzer mit einem Fahrzeug ein Manöver aus, so fallen in der Folge viele verschiedene Arten von Daten an, mit Hilfe derer sich das Manöver charakterisieren lässt. Essentiell für die Bewertung ist zum einen die Trajektorie des Fahrzeugs bzw. einzelner Fahrzeugglieder im Fall von mehrgliedrigen Zugfahrzeugen, zum anderen die zeitliche Entwicklung kritischer Winkel wie Lenk- und Einknickwinkel. In

den folgenden Abschnitten werden Qualitätsmaße betrachtet, die es ermöglichen, ein Manöver anhand der oben genannten Datensätze qualitativ zu bewerten.

Dazu werden zum einen Distanzmaße betrachtet, welche die Ähnlichkeit von Trajektorien anhand einer Kennzahl beschreiben. Große Ähnlichkeit oder räumliche Nähe der Kurven entsprechen dabei einer geringen Distanz. Zu den Distanzmaßen gehören die diskrete **Fréchet-Distanz**, welche jedoch wenig robust gegen Ausreißer im Datensatz ist, das **Dynamic Time Warping**, einer Variante der Fréchet-Distanz, die weniger Störungs-empfindlich gegen Ausreißer ist, sowie die **Earth Mover's Distance**, einem Distanzmaß, welches zusätzlich die Gewichtung einzelner Werte der Kurve zulässt.

Neben dem abgefahrenen Pfad ist das Lenkverhalten des Nutzers ein weiterer wichtiger Punkt in der Betrachtung der Durchführung eines Manövers. Zur Bestimmung des Ausmaß und der Stärke von Lenkbewegungen kann die **Steering Variance** genutzt werden, welche die Streuung des Lenkwinkels um den Mittelwert beschreibt. Unsicherheiten und ausgeprägtes Korrekturverhalten kann über die **Steering Reversal Rate** bestimmt werden, welche ein Maß für Änderungsrate der Lenkrichtung angibt.

2.2.1 Discrete Average Frechet Distance (DAFD)

Originale Fréchet-Distanz

Die Fréchet-Distanz, benannt nach Maurice Fréchet, ist ein Maß, um die **Distanz zwischen parametrisierten Kurven** zu messen. Zusätzlich zur Position eines Punktes auf der Kurve berücksichtigt sie die Reihenfolge der Punkte. Eine intuitive Definition kann folgendermaßen formuliert werden:

> **Fréchet-Distanz**
> Gegeben seien zwei unterschiedliche Pfade, auf denen sich jeweils ein Hund und sein Besitzer bewegen, welche durch eine Leine miteinander verbunden sind.

> Beide folgen ihrem eigenen Pfad vom Anfang bis zum Ende. Dabei ist es ihnen erlaubt ihre Geschwindigkeit zu variieren und stehen zu bleiben, aber niemals auf dem Pfad rückwärts zu gehen. Für jede Variation der Geschwindigkeiten existiert nun eine maximale Länge der Leine, welche durch das Abgehen der Pfade nötig ist. Die Fréchet-Distanz beider Pfade ist definiert als die kürzeste aller möglichen maximalen Längen der Leine.

Eine formale Definition für die Fréchet-Distanz, entnommen aus [Alt u. Godau, 1995], ist die folgende:

Definition 1. *Sei V ein euklidischer Vektorraum. Seien weiterhin $f : [a, a'] \to V$ und $g : [b, b'] \to V$ zwei Kurven. Die Fréchet-Distanz $\delta_F(f, g)$ zwischen beiden Kurven ist dann definiert als:*

$$\delta_F(f,g) := \inf_{\substack{\alpha[0,1]\to[a,a'] \\ \beta[0,1]\to[b,b']}} \max_{t\in[0,1]} \|f(\alpha(t)) - g(\beta(t))\| \tag{2.7}$$

wobei α, β stetige, monoton wachsende Funktionen sind mit den eindeutigen Start- und Endpunkten $\alpha(0) = a$, $\alpha(1) = a'$, $\beta(0) = b$ und $\beta(1) = b'$. Die Abstandsnorm $\| * \|$ zwischen zwei Funktionswerten kann beliebig gewählt werden.

Die Definition der Fréchet-Distanz berücksichtigt neben der Position auch die **Reihenfolge der Punkte**, welche durch die Annahme α und β seien monoton wachsend gegeben ist. Dies entspricht der Annahme der informellen Definition, dass Hund und Besitzer nicht auf ihrem Pfad umkehren dürfen. Das Infimum über die maximale Distanz aller Parametrisierungen der Kurve entspricht der kürzesten möglichen Leine zwischen Hund und Besitzer.

Dieses Distanzmaß kann in seiner rohen Fassung nicht zur Abstandsmessung zwischen zwei Trajektorien genutzt werden, da es unter mehreren Problemen leidet:

- Die obige Definition gilt nur für stetige Funktionen. Diskrete Datensätze können so nicht modelliert werden.
- Die Fréchet-Distanz ist empfindlich gegenüber Störungen, d.h. ein Ausreißer im

Datensatz kann das gesamte Distanzmaß verfälschen.

Problem 1 kann durch die Definition einer diskreten Fréchet-Distanz gelöst werden, welche das Distanzmaß auf beliebige Datensätze anwendbar macht. Problem 2 kann durch eine Durchschnittsbildung aller Punktabstände zu jeder Zeit auf beiden Kurven gelöst werden. Dies bedeutet, dass man den Durchschnitt aller Leinenlängen einer Parametrisierung betrachtet statt nur der maximalen Länge.

Diskrete Fréchet-Distanz

Die diskrete Fréchet-Distanz wurde zum ersten Mal von Eiter und Mannila definiert [Eiter u. Mannila, 1994] und stellt eine Approximation zur stetigen Variante dar. Dazu werden zunächst die Funktionen der beiden Kurven durch einen **Polygonzug** approximiert.

Definition 2. *Ein Polygonzug ist eine Funktion* $P : [0, n] \to V$ *mit* $n \in \mathbb{N}$*, für die gilt: Wenn* $\forall i \in \{0, 1, ..., n-1\} : P(i+\lambda) = \{(1-\lambda)P(i) + \lambda P(i+1) | \lambda \in [0, 1]\}$*, dann nennt man die Vereinigung* $\bigcup_{i=0}^{n-1} P(i+\lambda)$ *einen Polygonzug von* $P(0)$ *nach* $P(n)$*.*

Anschaulich werden n Punkte $P(i)$ der originalen Funktion ausgewählt, welche als Endpunkte für $n-1$ Liniensegmente dienen. Die Vereinigung aller Liniensegmente nennt man schließlich den Polygonzug. Der Abstand zwischen den resultierenden Polygonzügen beider Kurven wird nur noch zwischen diesen Endpunkten $P(i)$ bestimmt. In der Hundeleine-Analogie bedeutet dies, dass man die Pfade nur abschnittweise abgehen kann, ohne zwischen zwei Endpunkten eines Abschnitts stehen bleiben zu dürfen. Die formale Definition nach Eiter und Mannila [Eiter u. Mannila, 1994] lautet folgendermaßen:

Definition 3. *Sei* $P : [0, n] \to V$ *ein Polygonzug. Sei die Sequenz* $\sigma(P)$ *der Endpunkte der Liniensegmente von* P *gegeben durch* $(P(0), P(1), ..., P(n))$*. Seien* P *und* Q *Polygonzüge mit* $\sigma(P) = (u_1, ..., u_p)$ *und* $\sigma(Q) = (v_1, ..., v_q)$ *als entsprechende Se-*

quenzen. Eine Kopplung L zwischen P und Q ist eine Sequenz

$$(u_{a_1}, v_{b_1}), (u_{a_2}, v_{b_2}), ...(u_{a_n}, v_{b_n}) \tag{2.8}$$

eindeutiger Paare aus $\sigma(P) \times \sigma(Q)$ mit $a_1 = 1$, $b_1 = 1$, $a_m = p$, $b_m = q$, und $\forall i = 1, ..., q : (a_{i+1} = a_i \vee a_{i+1} = a_i + 1) \wedge (b_{i+1} = b_i \vee b_{i+1} = b_i + 1)$.
Die Länge $\|L\|$ einer Kopplung L ist definiert als der größte Abstand zweier gekoppelter Punkte der Polygonzugs:

$$\|L\| = \max_{i=1,...,m} d(u_{a_i}, v_{b_i}) \tag{2.9}$$

wobei d ein beliebiges Distanzmaß, beispielsweise die euklidische Distanz darstellt.
Die diskrete Fréchet-Distanz zwischen zwei Polygonzügen P und Q ist schließlich definiert als

$$\delta_{dF}(P, Q) = \min(\|L\| \mid L \text{ ist eine Kopplung zwischen } P \text{ und } Q) \tag{2.10}$$

Durch die Diskretisierung ist es möglich, die Fréchet-Distanz auf beliebige Datensätze anzuwenden.

Durchschnittliche diskrete Fréchet-Distanz: Dynamic Time Warping (DTW)

Um nun das Distanzmaß **robuster gegen Ausreißer** zu machen, wird fortan nicht mehr nur die maximale Distanz der Leine betrachtet, sondern der durchschnittliche Abstand zwischen Hund und Besitzer, welcher minimiert werden soll. Den durchschnittlichen Abstand zu minimieren bedeutet, dass Hund und Besitzer auf ihrem Weg vom Start zum Ende stets so nah wie möglich beieinander sind. Formal definiert ist diese Distanz nicht mehr als:
Definition 4. *Sei $L = (u_{a_1}, v_{b_1}), (u_{a_2}, v_{b_2}), ...(u_{a_n}, v_{b_n})$ eine Kopplung zweier Poly-*

gonzüge P und Q. Dann nennt man

$$\delta_{sdF} = \min \sum_{(u_{a_i}, v_{b_i}) \in L} \|u_{a_i} - v_{b_i}\| \tag{2.11}$$

die aufsummierte diskrete Fréchet-Distanz zweier Polygonzüge P und Q.
Die durchschnittliche diskrete Fréchet-Distanz ergibt sich durch Normalisierung:

$$\delta_{adF} = \frac{\delta_{sdF}}{n} \tag{2.12}$$

*wobei n die Anzahl der Kopplungspaare darstellt. Diese Distanz nennt sich auch **Dynamic Time Warping**-Distanz, da sie vornehmlich im Vergleich zwischen Zeitserien unterschiedlicher Geschwindigkeit oder Frequenz angewendet wird.*

Durch den in Abbildung 1 dargestellten Dynamic-Programming-Algorithmus kann die Distanz berechnet werden. Der Algorithmus läuft in $\mathcal{O}(n^2)$, effizientere Versionen

Algorithm 1 Average Discrete Fréchet Distance

```
 1: procedure ADFDistance(P: array [1..p], Q: array [1..q])
 2:     ADF = array [0..p, 0..q]
 3:     for i = 1 to p do
 4:         ADF[i, 0] := infinity
 5:     for i = 1 to q do
 6:         ADF[0, i] := infinity
 7:     ADF[0,0] = 0
 8:     for i = 1 to p do
 9:         for j = 1 to q do
10:             cost = dist(P[i],Q[j])
11:             ADF[i,j] = cost + min(AFD[i-1,j],AFD[i,j-1],AFD[i-1,j-1])
12:     return ADF[p,q]
```

existieren, sind aber im Rahmen dieser Arbeit nicht relevant. Der obige Algorithmus ist für den Vergleich zweier Trajektorien vollkommen ausreichend, da keine zeitlichen Beschränkungen der Berechnungszeit gegeben sind.

2.2.2 Earth Mover's Distance (EMD)

Die Earth Mover's Distance ist im Gegensatz zur Fréchet-Distanz ein Distanzmaß für Wahrscheinlichkeitsverteilungen einer gegebenen Domäne, beispielsweise dem $\mathbb{R}^2$. Das Konzept geht dabei auf eine Veröffentlichung Gaspard Monges aus dem Jahr 1781 zurück [Monge, 1781]. Diese Distanz wurde in [Boem u. a., 2011] im Kontext der Klassifizierung von Trajektorien verwendet.

Auch hier existiert eine informelle Definition zur Veranschaulichung des Konzepts:

> **Earth Mover's Distance**
> Gegeben seien zwei Wahrscheinlichkeitsverteilungen. Man stelle sich vor, eine der beiden Verteilungen hinterlässt Erdhaufen in der gegebenen Domäne entsprechend ihrer Verteilungsfunktion. Die andere Verteilung hinterlässt stattdessen Löcher entsprechend ihrer Verteilungsfunktion. Die Earth Mover's Distance entspricht den minimalen Kosten, die Löcher mit der angehäuften Erde zu füllen. Die Kosten setzen sich aus der transportierten Erdmenge multipliziert mit der Distanz, um welche bewegt wurde, zusammen.

Das Konzept basiert auf der Minimierung einer Energiefunktion, um eine der beiden Verteilungen auf die andere abbilden zu können. Aus dieser Definition ist ersichtlich, dass dieses Distanzmaß ohne Anpassungen nur auf Verteilungen mit gleicher Erdmenge/-bedarf angewendet werden kann, formal bedeutet es, dass das Integral über den beiden Verteilungen identisch sein muss. Für den Zweck des Vergleichs zweier Trajektorien gibt es einmal mehr Probleme mit der Standardform dieses Distanzmaßes.

- Die Trajektorien liegen als diskrete Punktmenge vor, wohingegen die EMD nur für Verteilungen definiert ist.
- Die EMD kann nur auf Verteilungen mit identischem Integral angewendet werden, für den diskreten Fall entspricht dies der gleichen Größe beider Datensätze bzw. einer identischen Anzahl an Erdquellen und -senken.

Problem 1 lässt sich über eine Umwandlung der Trajektorie in eine diskrete Verteilung lösen. Problem zwei erfordert die Einführung einer zusätzlichen Quelle bzw. Senke, welche keine Transportkosten besitzt.

EMD für diskrete Verteilungen

Da die EMD nur für Verteilungen definiert ist, muss eine Möglichkeit gefunden werden die zu vergleichenden Kurven in eine Verteilung umzuwandeln. Datensätze für Trajektorien oder die Entwicklung von Winkeln über die Zeit haben jeweils unterschiedliche Domänen.

- Trajektorien beschreiben die Entwicklung einer 2-dimensionalen Position über die Zeit und bestehen aus einer Menge an Tripeln (x, y, t), wobei x und y die karthesischen Koordinaten und t den Zeitpunkt darstellen. Zur Vereinfachung wird angenommen, dass die Zeitpunkte, an denen eine Position abgefahren wurde, nicht von Bedeutung sind. Stattdessen wird allein der Pfad, also die Sequenz von Positionen betrachtet, was die Dimension dieser Datensätze von 3 auf 2 reduziert.
- Die Entwicklung 1-dimensionaler Werte wie Winkel erfolgt als Menge von Tupeln (a, t), wobei a den Wert und t wiederum den zugehörigen Zeitstempel darstellt.

Beide Arten an Datensätzen besitzen die Dimension 2, was eine Vereinheitlichung der weiteren Vorgehensweise bedeutet.

Das Vorgehen, die Datensätze auf diskrete Verteilungen abzubilden, wurde aus [Boem u. a., 2011] entnommen. Zunächst wird ein 2-dimensionales Gitter beliebiger Zellgröße über die Domäne gelegt, in diesem Fall den $\mathbb{R}^2$. An jedem Punkt des Datensatzes wird nun eine konstante Erdmenge hinterlassen, sodass jede Zelle schließlich eine bestimmte Menge an Erde enthält. Die resultierende Verteilung der Erdmengen über die Zellen des Gitters entspricht der Verteilung der Punkte über die Domäne. Indem man die

Zellgröße des Gitters verkleinert, lässt sich die Verteilung bis zu einem gewissen Grad verfeinern, wo jede Zelle maximal noch eine Einheit an Erde enthält. Ein Vorteil der Gitterdarstellung ist, dass der Datensatz robuster gegen Ausreißer gemacht werden kann durch eine Vergrößerung der Zellen. Es muss jedoch ein Kompromiss eingegangen werden zwischen Robustheit und Ungenauigkeit der Darstellung. Je größer die Zelle desto robuster wird zwar der Datensatz, aber desto ungenauer wird ebenfalls die Verteilung der Punktemenge auf das Gitter abgebildet. Die formale Definition des Gitters lautet folgendermaßen:

Definition 5. *Sei η die konstante Erdmenge, welche von einem Datensatz D zu einem Zeitpunkt an der Position eines gesampelten Punktes hinterlassen wird. Sei weiterhin $N^D_{(i,j)}$ die Anzahl der Samplepunkte des Datensatzes D in der Zelle (i,j) des Gitters. Dann nennt man $g^D_{i,j} = \eta * N^D_{(i,j)}$ das Gewicht der Zelle (i,j) $\forall i = 1,...,n_{rows} \wedge j = 1,...,n_{cols}$. Die Matrix $G^D = [g^D_{i,j}]$ über alle Gewichte nennt sich schließlich Gitter über den Datensatz D.*

Will man nun die Gewichte dieses Gitters benutzen, so sind nur die Gewichte ungleich 0 von Relevanz. Wir definieren daher einen Gewichtsvektor $\omega^D = (g^D_{i,j} \mid g^D_{i,j} \neq 0, \forall i = 1,...,n_{rows} \wedge j = 1,...,n_{cols})$, welcher nur die von 0 verschiedenen Gewichte enthält. Jedem Datensatz wird fortan ein solcher Gewichtsvektor zugeordnet. Die Summe aller Gewichte entspricht der Gesamtmasse des Gitters und ist proportional zur Länge des Datensatzes.

Die Berechnung der EMD für den diskreten Fall geht zurück auf die Lösung des Transportproblems nach Kantorovich [Kantorovich, 1942]. Das Transportproblem beschreibt die Frage nach dem optimalen Plan zum Transport einheitlicher Objekte von Angebots- zu Nachfrageorten. Es kann formuliert werden als Optimierungsproblem der folgenden Form:

Minimiere die Funktion:

$$J_{min} = \min \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} d_{ij} f_{ij} \tag{2.13}$$

wobei n_1 und n_2 der Größe der Gewichtsvektoren von Datensatz 1 bzw. 2 entspricht, d_{ij} die Distanz zwischen zwei Zellen ist, denen Gewicht i und j der Datensätze 1 bzw. 2 zugeordnet ist, sowie f_{ij} den Fluss zwischen den entsprechenden Zellen darstellt. Ziel ist es eine Flussmatrix $F = [f_{ij}]$ zu finden, welche die Transportkosten zwischen den beiden Gittern minimiert.

Folgende Bedingung sind dabei zu beachten:

$$f_{ij} \geq 0 \quad \forall (i,j) : \; 1 \leq i \leq n_1, \, 1 \leq j \leq n_2 \tag{2.14}$$

$$\sum_{j=1}^{n_2} f_{ij} = \omega_i^{(1)} \; \forall i = 1, ..., n_1 \tag{2.15}$$

$$\sum_{i=1}^{n_1} f_{ij} = \omega_j^{(2)} \; \forall j = 1, ..., n_2 \tag{2.16}$$

Die erste Bedingung besagt, dass der Fluss jeweils nur in eine Richtung, d.h. von einer Quelle zu einer Senke, erfolgen darf. Die letzten beiden besagen, dass jeweils die gesamte Menge an Erde transportiert werden muss. Existiert eine Lösung des Problems, so wird sie auch gefunden. Ist eine optimale Lösung schließlich gefunden, kann die EMD wie folgt berechnet werden.

Definition 6. *Sei J_{min} der optimale Kostenwert des oben genannten Transportproblems und sei $F = [f_{ij}]$ der zugehörige optimale Fluss zwischen den Gittern. Seien zudem n_1 und n_2 die Längen der Gewichtsvektoren zweier Datensätze. Dann nennt man*

$$EMD = \frac{J_{min}}{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f_{ij}} \tag{2.17}$$

die Earth Mover's Distance zwischen diesen beiden Datensätzen.

Es existieren unterschiedliche Methoden das Transportproblem zu lösen, unter anderen durch lineare Programmierung. Das Dantzig-Simplex-Verfahren beispielsweise liefert optimale Lösungen in annähernd polynomieller Laufzeit [Dantzig, 1998], sofern die Gewichte ganzzahlig sind und eine Lösung existiert.

Umgang mit Datensätzen unterschiedlicher Größe

Probleme treten auf, wenn die betrachteten Datensätze nicht dieselbe Größe besitzen. Dies führt zu einem Ungleichgewicht zwischen Angebot und Nachfrage. Die beiden Gitter haben folglich unterschiedliche Gesamtmassen. Wendet man das Transportproblem auf ein Paar solcher Datensätze an, so ergäbe sich zwangsläufig entweder eine höhere Nachfrage des einen Gitters als das andere liefern kann, oder analog ein höheres Angebot als das andere benötigt. Die Lösung besteht darin, je eine virtuelle Datenquelle und -senke zu erstellen, welche Überschüsse bzw. Defizite auffangen können. Um den Kostenwert des Optimierungsproblems nicht zu beeinflussen, bekommen besagte Quelle und Senke Transportkosten von 0 zugewiesen.

2.2.3 Steering Variance

Ein relativ simples Gütemaß für das Lenkverhalten bildet die Varianz bzw. die Standardabweichung des Lenkwinkels. Aufgrund seiner Einfachheit wird es in vielen Studien genutzt, welche eine Bewertung des Lenkverhaltens durchführen [Östlund u. a., 2005] [Johansson u. a., 2004] [Liu u. a., 1999].

Die Varianz bildet ein Maß für die Streuung der Lenkwinkel. Sie gibt an, wie stark der Nutzer Schwankungen in seinem Lenkverhalten unterliegt. Eine geringe Varianz deutet auf ein stabiles Lenkverhalten hin, eine hohe entsprechend auf ein unsicheres Verhalten.

Die Varianz eines Datensatzes D von Lenkwinkeln wird folgendermaßen berechnet:

$$Var = \sum_{x \in D} (x - \mu)^2 \tag{2.18}$$

wobei x einen einzelnen Lenkwinkel des Datensatzes darstellt und μ den Mittelwert über alle $x \in D$.

Die Varianz ist gering, wenn kontinuierlich kleine Lenkbewegungen relativ zum Mittelwert ausgeführt werden und entsprechend hoch, wenn kontinuierlich starke Lenkbewegungen relativ zum Mittelwert ausgeführt werden. Ein Problem mit dieser Messung ist die Abhängigkeit vom Mittelwert des Datensatzes. Dieser wiederum hängt größtenteils mit der Fahraufgabe zusammen, welche durchgeführt wurde. Je stärker die gefahrene Kurve, desto größer ist in der Regel auch der resultierende Mittelwert.

2.2.4 Steering Reversal Rate

Die Steering Reversal Rate ist eine der am weitest verbreiteten Metriken zur Bewertung des Lenkverhaltens [Östlund u. a., 2005] [Johansson u. a., 2004] [Macdonald u. Hoffmann, 1980] [Östlund u. a., 2004]. Sie geht zurück auf eine Veröffentlichung von McLean und Hoffmann [Mclean u. Hoffmann, 1971].

Im Gegensatz zur Streuung des Lenkwinkels betrachtet sie die Häufigkeit von Richtungsänderungen des Lenkrads, welche einen gewissen Schwellwinkel überschreiten. Niedrige Werte deuten auf ein sehr gezieltes Lenkverhalten, wohingegen hohe Werte darauf hinweisen, dass sehr viel Korrekturen durchgeführt werden mussten. Die Richtungsänderung des Lenkrads kann formal als die Vorzeichenänderung der Lenkwinkeländerungsrate betrachtet werden. Die Größe des betrachteten Schwellwinkels ist dabei von zentraler Bedeutung für die Berechnung des Gütemaßes. Die in der Literatur betrachteten Schwellwerte variieren zwischen 0.5 und 10°.

Das Vorgehen zur Berechnung der Steering Reversal Rate findet sich in Algorithmus 2, welcher [Östlund u. a., 2004] entnommen wurde.

Algorithm 2 Algorithmus zur Berechnung der Steering Reversal Rate

1: Eingabe: Zeitserie von Lenkwinkeln L, Schwellwert-Winkel s
2: Ausgabe: Anzahl der Lenkwinkelumkehrungen r
3: Zunächst werden stationäre Punkte in L, sogenannte **stationary points**, bestimmt:
4: * Zur besseren Identifizierung der **stationary points** kann im ersten Schritt ein optionales Resampling von L auf eine schnellere Samplerate durchgeführt werden.
5: * Der zweite Schritt umfasst die Filterung von L mittels eines **Butterworth** Low-Pass-Filters mit dem Ziel hochfrequente Rausch- und Störeinflüsse aus dem Datensatz herauszufiltern.
6: * Ein **stationary point** ist ein Punkt in der Lenkwinkel-Kurve von L, welcher die Ableitung 0 besitzt, der Lenkwinkel sich also nicht ändert. Es folgt eine Approximation solcher Punkte durch die Identifizierung von Nulldurchgängen in der Differenz zwischen aufeinanderfolgenden Lenkwinkeln des gefilterten Datensatzes L.
7: Es folgt eine Unterscheidung der Umkehrrichtung in **Upward Reversal** (von Minimum zu Maximum) und **Downward Reversal** (von Maximum zu Minimum). Für jede identifizierte Umkehrung wird r um eins erhöht.
8: * **Upward Reversal**: Man startet beim ersten **stationary point** und sucht nach einem nachfolgenden **stationary point**, welcher um mehr als s größer als der erste ist. Tritt der Fall auf, dass man einen **stationary point** findet, welcher einen geringeren Winkelwert als der erste besitzt, dann wird dieser als der neue initiale Punkt genutzt und die Suche wird weitergeführt.
9: * **Downward Reversal**: Analog zum vorherigen Schritt, nur auf negative Wertigkeiten des Datensatzes L angewendet.

2.3 Fehleranalyse

Im folgenden Kapitel werden zunächst gängige Verfahren und Techniken vorgestellt, welche zur Abschätzung und Analyse von Modellfehlern genutzt werden können. Zunächst wird erläutert, wie man mittels der Taylor-Expansion ein beliebiges nichtlineares Modell in eine lineare Approximation umwandeln kann. Dies vereinfacht die Quantifizierung des Einflusses von Eingangsfehlern auf den Folgezustand des betrachteten Modells. In der Folge werden zwei Verfahren betrachtet, welche eine Betrachtung der Fortpflanzung probabilistischer Fehler erlauben, die Linear Transform sowie die Unscented Transform. Den Abschluss bilden zwei heuristische Optimierungs-Verfahren, welche eine experimentelle Betrachtung verschiedener Fehler ermöglichen,

der Partikelfilter sowie die simulierte Abkühlung.

2.3.1 Taylor-Expansion

Die Taylorreihe wird in der Analysis genutzt, um eine stetig-differenzierbare Funktion in der unmittelbaren Umgebung einer Entwicklungsstelle als Potenzreihe darzustellen. Die Reihe bezeichnet den Grenzwert der Taylor-Polynome und wird auch Taylor-Expansion genannt.

Definition 7. *Sei $I \subset \mathcal{R}$ ein offenes Intervall und $f : I \to \mathcal{R}$ eine stetig-differenzierbare Funktion. Dann nennt sich die unendliche Reihe*

$$Tf(x)|_a = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{2.19}$$

die Taylorreihe von f mit Entwicklungsstelle $a \in I$. Hierbei bezeichnet $f^{(n)}$ die n-te Ableitung von f, wobei $f^(0) = f$ gilt.

Die Summe der ersten beiden Terme der Taylorreihe wird auch die Taylor-Approximation erster Ordnung oder auch *Linearisierung* von f an der Stelle a genannt.

$$T_1 f(x)|_a = f(a) + \frac{\partial f}{\partial x}(a) * (x-a) \tag{2.20}$$

Diese Definition lässt sich ohne weiteres auf den multivariablen Fall erweitern. Sei $I_0, I_1, ..., I_n \subset \mathcal{R}$ n offene Intervalle und $f : I_0 \times I_1 \times ... \times I_n \to \mathcal{R}$ eine stetig-differenzierbare Funktion. Für die Linearisierung an der Entwicklungsstelle $(a_0, a_1, ..., a_n) \in I_0 \times I_1 \times ... \times I_n$ ergibt sich folglich

$$\begin{aligned} &T_1 f(x_0, x_1, ..., x_n)|_{(a_0,a_1,...,a_n)} \\ &= f((a_0, a_1, ..., a_n)) + \frac{\partial f}{\partial x_0, x_1, ..., x_n}(a_0, a_1, ..., a_n) * ((x_0, x_1, ..., x_n) - (a_0, a_1, ..., a_n)) \\ &= f((a_0, a_1, ..., a_n)) + \frac{\partial f}{\partial x_0}(a_0) * (x_0 - a_0) + ... + \frac{\partial f}{\partial x_n}(a_n) * (x_n - a_n) \end{aligned} \tag{2.21}$$

2.3.2 Systematische Fehler

Systematische Fehler sind systemimmanente Fehler, welche sich bei wiederholter Messung **nicht im Mittel aufheben**. Diese Fehler sind konstant und beschreiben die Tendenz eines Messwertes zu einer bestimmten Seite. Systematische Fehler treten unter anderem durch falsche Kalibrierung von Sensoren oder durch Fehler im Systemmodell auf.

Linearisierung des Fehlers

Sei $f : \mathcal{R}^n \rightarrow \mathcal{R}^m$ das gegebene Systemmodell mit $n, m \in \mathcal{N}$ als Dimension der Ein- bzw. Ausgabe. Eine Eingabe $\vec{x} \in \mathcal{R}^n$ soll nun mit dem Systemmodell transformiert werden. Dieser Vektor unterliegt jedoch einem deterministischen, d.h. konstanten Fehler ϵ. Der fehlerbehaftete Vektor wird fortan mit $\vec{x}_\epsilon = \vec{x} + \epsilon$ bezeichnet. Durch den Fehler im Eingabevektor pflanzt sich der Fehler in den Folgezustand fort. Es bezeichnet

$$\Delta f = f(\vec{x}_\epsilon) - f(\vec{x}) \tag{2.22}$$

den absoluten Fehler zwischen den fehlerfreien und fehlerbehafteten Folgezuständen Systems. Im Folgenden wird davon ausgegangen, dass $|\epsilon| << 1$ gilt, der Fehler also sehr klein ist. Um die Auswirkung von kleinsten Änderungen im Eingangsparameter auf die Ausgabe der Funktion zu verdeutlichen, bietet es sich an Δf in allgemeiner Abhängigkeit von ϵ darzustellen. Da f jedoch eine beliebige, potentiell nicht-lineare Funktion darstellt, ist eine allgemeine Darstellung einer solchen Abhängigkeit nicht möglich. In einem solchen Fall liefert eine Taylor-Approximation erster Ordnung eine gute Annäherung, da $\vec{x}_\epsilon$ in der unmittelbaren Umgebung von $\vec{x}$ liegt. Für die Linearisierung der Funktion um den Entwicklungspunkt $\vec{x}_\epsilon$ gilt:

$$f(\vec{x}) \approx f(\vec{x}_\epsilon) + \frac{\partial}{\partial \vec{x}} f|_{\vec{x}_\epsilon}(\vec{x} - \vec{x}_\epsilon) \tag{2.23}$$

Setzt man 2.23 in 2.22 ein ergibt sich schließlich ein linearer Zusammenhang zwischen Eingangs- und Ausgangsfehler:

$$\begin{aligned}\Delta f &= f(\vec{x}_\epsilon) - f(\vec{x}) \\ &\approx f(\vec{x}_\epsilon) - (f(\vec{x}_\epsilon) + \frac{\partial}{\partial \vec{x}} f|_{\vec{x}_\epsilon}(\vec{x} - \vec{x}_\epsilon)) \\ &\approx \frac{\partial}{\partial \vec{x}} f|_{\vec{x}_\epsilon}(\vec{x}_\epsilon - \vec{x}) \\ &\approx \frac{\partial}{\partial \vec{x}} f|_{\vec{x}_\epsilon} \epsilon \end{aligned} \tag{2.24}$$

Für multivariable Funktionen ergibt sich entsprechend

$$\begin{aligned}\Delta f &= f(\vec{x}_{0,\epsilon}, \vec{x}_{1,\epsilon}, ..., \vec{x}_{n,\epsilon}) - f(\vec{x}_0, \vec{x}_1, \vec{x}_2) \\ &\approx J_{\vec{x}_0}|_{\vec{x}_{0,\epsilon}} * \epsilon_0 + J_{\vec{x}_1}|_{\vec{x}_{1,\epsilon}} * \epsilon_1 + ... + J_{\vec{x}_n}|_{\vec{x}_{n,\epsilon}} * \epsilon_n \end{aligned} \tag{2.25}$$

wobei ϵ_i dem Fehler des i-ten Eingabeparameters entspricht für $i = 1..n$. Die Matrix der partiellen Ableitungen

$$J_{\vec{x}}|_{\hat{x}} = \frac{\partial f}{\partial \vec{x}}(\hat{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\hat{x}) & \frac{\partial f_1}{\partial x_2}(\hat{x}) & \dots & \frac{\partial f_1}{\partial x_n}(\hat{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\hat{x}) & \frac{\partial f_m}{\partial x_2}(\hat{x}) & \dots & \frac{\partial f_m}{\partial x_n}(\hat{x}) \end{pmatrix} \tag{2.26}$$

nennt man die Jacobi-Matrix von f an der Stelle $\hat{x}$. Die Jacobi-Matrix einer Funktion in einem Punkt entspricht einer linearen Annäherung der Funktion in einer lokalen Umgebung um den betrachteten Punkt. Um die Wachstumsrate des Eingabefehlers einer Funktion zu bestimmen, reicht es folglich aus, die Jakobi-Matrizen der verschiedenen Eingaben zu berechnen. Ist der Eingabefehler sehr gering, so kann ein annähernd linearer Zusammenhang zwischen Eingabe- und Ausgabefehler nachgewiesen werden.

2.3.3 Probabilistische Fehler

Probabilistische Fehler sind zufällige Fehler, welche sich bei (theoretisch unendlich oft) wiederholter Messung im Mittel aufheben. Diese Fehler beschreiben die Abweichung vom Erwartungswert der Messung und ändern sich mit jedem Messvorgang. Ursache für probabilistische Fehler sind unter anderem Sensorrauschen oder unkontrollierbare Umwelteinflüsse.

Gegeben sei ein Modell des zu betrachtenden Systems mit einem n-dimensionalen Zustandsvektor $x_t \in \mathbb{R}^n$ zum Zeitpunkt t und einer Zustandsübergangsfunktion $f : \mathbb{R}^n \to \mathbb{R}^n$ mit $x_t = f(x_{t-1})$. Der probabilistische Fehler wird zumeist über ein normalverteiltes additives Rauschen auf den Zustandsvektor modelliert.

$$x_t = f(x_{t-1}) + \epsilon_t \tag{2.27}$$

wobei ϵ_t eine n-dimensionale normalverteilte Zufallsvariable zum Zeitpunkt t ist mit $\epsilon_t \sim \mathcal{N}(0, Q_t)$. Der Erwartungswert dieser Zufallsvariable ist üblicherweise 0, sodass sich der Fehler bei wiederholten Messungen im Mittel aufhebt. Sofern die Messungen der Werte des Zustandsvektors unabhängig voneinander sind, bildet die Kovarianzmatrix Q_t eine Diagonalmatrix mit den einzelnen Varianzen auf der Diagonalen. Eine alternative Schreibweise ist die folgende:

$$x_t \sim \mathcal{N}(\hat{x}_t, \Sigma_t) \tag{2.28}$$

$\hat{x}_t$ entspricht dabei dem n-dimensionalen Erwartungswert der Verteilung, welcher in diesem Fall $f(x_{t-1})$ entspricht. Σ_t hingegen beschreibt die $n \times n$-dimensionale

Kovarianzmatrix:

$$\Sigma_t = \begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \dots & \sigma_{1n}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \dots & \sigma_{2n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1}^2 & \sigma_{n2}^2 & \dots & \sigma_{nn}^2 \end{pmatrix} \tag{2.29}$$

Gegeben seien der Zustandsvektor x_{t-1}, dessen Verteilung $x_{t-1} \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$ sowie die Zustandsübergangsfunktion f. Will man nun eine Aussage über den Zustand x_t treffen treffen, so kann man diesen ganz einfach über eine Transformation mit f erhalten. Eine Aussage über die folgende Verteilung von x_t gestaltet sich jedoch schwieriger. Ist f eine lineare Funktion der Form $y = Ax + b$, so gilt:

$$x_t \sim \mathcal{N}(Ax_{t-1} + b, A\Sigma_{t-1}A^T) \tag{2.30}$$

Lineare Transformationen normalverteilter Zufallsvariablen bleiben normalverteilt. Anders gestaltet sich der Fall für nichtlineare Transformationen der Form $y = g(x)$. In diesem Fall sind die normalverteilten Zufallsvariablen leider nicht mehr normalverteilt.

$$x_t \not\sim \mathcal{N}(f(x_{t-1}), F\Sigma_t F^T) \tag{2.31}$$

Es gilt daher eine Möglichkeit zu finden die resultierende Verteilung bestmöglich zu approximieren. Dazu werden im Folgenden zwei verschiedene Transformationsmethoden vorgestellt, welche die resultierende Verteilung bis zur ersten (*Linear Transform*) bzw. zweiten Ordnung (*Unscented Transform*) genau approximieren.

Linear Transform

Die *Linear Transform* ist eine Methode, welche unter anderem im Extended Kalman Filter [Kalman, 1960] [Smith u. a., 1962] eingesetzt wird, um die nichtlinea-

re Transformation einer gegebenen Wahrscheinlichkeitsverteilung zu approximieren. Die Transformation einer Wahrscheinlichkeitsverteilung ist für den linearen Fall exakt. Im Falle der Transformation mit einer nichtlinearen Funktion muss diese durch eine lineare approximiert werden.

Nimmt man nun die Funktion f des vorangegangenen Kapitels, so lässt sich deren Linearisierung an der Stelle $\hat{x}$ mithilfe der Taylorreihe folgendermaßen schreiben:

$$f'(x_{t-1}) = f(\hat{x}) + \frac{\partial f}{\partial x}(\hat{x})(x - \hat{x}) \tag{2.32}$$

Damit ergibt sich als Approximation der Funktion f eine lineare Funktion der Form $y = Ax + b$:

$$f'(x_{t-1}) = F(x - \hat{x}) + f(\hat{x}) \tag{2.33}$$

Mithilfe dieser neuen Funktion f' ist es nun möglich die Wahrscheinlichkeitsverteilung des Folgezustands zu bestimmen. Ist

$$x_{t-1} \sim \mathcal{N}(x_{t-1}, \Sigma_{t-1}) \tag{2.34}$$

dann gilt:

$$x_t \sim \mathcal{N}(F(x - \hat{x}) + f(\hat{x}), F\Sigma_{t-1}F^T) \tag{2.35}$$

Ein Problem dieser Methode ist, dass lediglich Taylorpolynome erster Ordnung betrachtet werden. Das Intervall um die Entwicklungsstelle, in der die approximierte Funktion eine gute Annäherung darstellt, ist in diesem Fall sehr klein. Ist die zu approximierende Funktion zudem hochgradig nicht-linear, so kann es sehr schnell zu großen Ungenauigkeiten kommen, je weiter man sich von diesem Intervall entfernt. Eine Linearisierung macht daher nur Sinn, wenn die nichtlinearen Terme der Taylorreihe vernachlässigbar sind.

Eine Möglichkeit dieses Problem zu umgehen ist es, weitere Terme der Taylorreihe zu betrachten, etwa bis zur zweiten oder dritten Ordnung je nach gewünschter Genauigkeit. Jedoch geht dann erneut die Linearität verloren. Ein weiterer Nachteil, den die Approximierung der Funktion mittels Taylorreihe mit sich bringt, ist die analytischen Bestimmung der Jacobi-Matrix der Funktion, welche aufwändig sein kann, und welche differenzierbare Modelle erfordert. Eine Alternative zu dieser Vorgehensweise, welche direkt mit der nichtlinearen Funktion arbeitet ohne die Notwendigkeit der Berechung von Ableitungen, bildet die Unscented Transform.

Unscented Transform

Wie im vorangegangenen Kapitel beschrieben hat die direkte Approximierung einer Funktion mittels Taylor-Reihenentwicklung den Nachteil, dass man zum Teil komplexe Ableitungsmatrizen berechnen muss, um eine annähernde Repräsentation der Funktion zu erhalten. Zudem ist es auf diesem Wege nur möglich eine maximal lineare Approximierung zu erreichen, da man ansonsten wieder das Problem hat, dass die Approximation der transformierten Wahrscheinlichkeitsdichte der Verteilung nicht mehr normalverteilt ist.

Die Unscented Transform liefert eine Approximation zweiter Ordnung, welche unter anderem im Unscented Kalman Filter [Julier u. Uhlmann, 1997] zum Einsatz kommt. Im Gegensatz zur Linear Transform nutzt die Unscented Transform die ersten drei Terme der Taylorreihe, hat aber den Vorteil, dass keine numerischen oder analytischen Ableitungen verwendet werden.

Das Grundprinzip der Unscented Transform lautet folgendermaßen: Man wendet die nichtlineare Transformation auf einige deterministisch gewählte Sigma-Punkte an, welche die Wahrscheinlichkeitsdichte des Modells charakterisieren. Aus den transformierten Punkten wird daraufhin die transformierte Wahrscheinlichkeitsdichte rekonstruiert, indem man Mittelwert und Kovarianzmatrix berechnet. Man hat so den Vorteil mit der exakten Funktion arbeiten zu können ohne die Notwendigkeit einer

aufwändigen Bestimmung von Ableitungen.

Gegeben seien erneut eine n-dimensionale Zufallsvariable $x_{t-1} \sim \mathcal{N}(\mu, \Sigma)$ sowie die nichtlineare Zustandsübergangsfunktion f. Die einzelnen Schritte der Unscented Transform lauten dann folgendermaßen:

- Wähle $2n + 1$ Sigma-Punkte mit Gewichten:

$$\Omega = \{(x^{(i)}, \omega^{(i)}) \; \forall i = 0..2n + 1\}$$

  sodass

$$\sum_{i=0}^{2n+1} \omega^{(i)} = 1$$

- Transformiere die Sigma-Punkte mit der nichtlinearen Funktion f:

$$y^{(i)} = f(x^{(i)}) \quad \forall i = 0..2n + 1$$

- Berechne Mittelwert und Kovarianz der transformierten Punkte:

$$\mu_y = \sum_{i=0}^{2n+1} \omega_m^{(i)} y^{(i)} \qquad \Sigma_y = \sum_{i=0}^{2n+1} \omega_c^{(i)} (y^{(i)} - \mu_y)(y^{(i)} - \mu_y)^T$$

  wobei $\omega_m^{(i)}$ und $\omega_c^{(i)}$ die Gewichte für die Mittelwert- bzw. die Kovarianzberechnung beschreiben.

Die Sigma-Punkte werden so gewählt, dass Erwartungswert und empirische Kovarianz der gegebenen normalverteilten Zufallsvariable $x_{t-1} \sim \mathcal{N}(\mu, \Sigma)$ entsprechen, da auf diese Weise die unterliegende Wahrscheinlichkeitsverteilung bestmöglich charakterisiert wird. Das Vorgehen zur Bestimmung der Punkte wird im Folgenden näher erläutert.

Auf die Kovarianzmatrix Σ wird zunächst eine Cholesky-Zerlegung angewendet:

$$\Sigma = LL^T \tag{2.36}$$

wobei L eine untere Dreiecksmatrix darstellt. Man kann die Cholesky-Zerlegung mit dem Ziehen der Quadratwurzel einer Matrix vergleichen. Ist die Matrix positiv definit, so enthält die Matrix L die Quadratwurzeln der Elemente der Kovarianzmatrix. Ist Σ eine Diagonalmatrix, so ist $L = L^T$ mit jeweils den Standardabweichungen der Verteilung auf ihrer Diagonalen.

Danach werden $2n + 1$ Sigma-Punkte bestimmt:

$$x^{(0)} = \mu \tag{2.37}$$

$$x^{(i)} = \mu + \sqrt{n + \lambda}(L)_i \quad i = 1..n \tag{2.38}$$

$$x^{(i)} = \mu - \sqrt{n + \lambda}(L)_{i-n} \quad i = n + 1..2n \tag{2.39}$$

wobei $(L)_i$ dem i-ten Spaltenvektor von L entspricht und λ einen Skalierungsparameter darstellt der Form:

$$\lambda = \alpha^2(n + \kappa) - n \tag{2.40}$$

Die beiden zusätzlichen Parameter α und κ bestimmen die Streuung der Sigma-Punkte und werden meist sehr klein gesetzt. Gebräuchliche Werte sind $\alpha = 1e - 3$ und $\kappa = 0$ [Julier u. Uhlmann, 1997].

Schließlich werden die Gewichte der Punkte für die Erwartungswert- und die Kovarianzberechnung ermittelt:

$$\omega_m^{(0)} = \frac{\lambda}{n + \lambda} \qquad \omega_c^{(0)} = \frac{\lambda}{n + \lambda} + (1 - \alpha^2 + \beta) \tag{2.41}$$

$$\omega_m^{(i)} = \frac{1}{2(n + \lambda)} \qquad \omega_c^{(i)} = \frac{1}{2(n + \lambda)} \tag{2.42}$$

wobei β ein Parameter zur Charakterisierung der Verteilung ist. Für Normalverteilungen ist $\beta = 2$ optimal [Julier u. Uhlmann, 1997]. Wie man sieht ist die Summe der Kovarianz-Gewichte größer als 1, um die Ungenauigkeit durch die unterliegende Taylor-Approximierung auszugleichen.

2.3.4 Der Partikelfilter

Der Partikelfilter gehört zur Klasse der **sequenziellen Monte-Carlo-Methoden** (SMC), welche den Zustand innerhalb eines dynamischen Prozesses mittels eines stochastischen Verfahrens schätzen. Der Zustand ist in diesem Fall nicht direkt bestimmbar. Er liegt nur durch Stichproben in Form von fehlerbehafteten Beobachtungen bzw. Messungen vor.

SMC-Methoden sind in der Lage beliebige, multimodale Verteilungen zu modellieren mit dem Ziel die **a posteriori** Wahrscheinlichkeitsdichte der Verteilung anhand aller zur Verfügung stehender Informationen und Messungen zu bestimmen.

Ein typisches Anwendungsgebiet ist das Objekttracking, bei der die Pose eines Objekts anhand von fehlerhaften Messungen über die die Zeit verfolgt wird. Die Idee des Partikelfilters geht auf Andrew Blake und Michael Isard zurück, welche ihn erstmals 1998 in Form des **Condensation Algorithmus** im Kontext von visuellem Personentracking [Isard u. Blake, 1998] eingesetzt haben.

Markov-Ketten und die Markov-Annahme

Eine Markov-Kette beschreibt einen stochastischen Prozess, der auf der Markov-Annahme basiert. Die Markov-Annahme besagt, dass der aktuelle Zustand eines Systems nur von einer endlichen Historie von vorherigen Zuständen abhängt. Eine Markov-Kette erster Ordnung bedeutet, dass lediglich der aktuelle Zustand benötigt wird, um eine Prognose über den Folgezustand zu treffen.

Bayes Theorem

Das Bayes-Theorem beschreibt den Zusammenhang bedingter Wahrscheinlichkeiten zwischen zwei Ereignismengen A und B. Sind die Grundwahrscheinlichkeiten $P(A)$ und $P(B)$ sowie die bedingt Wahrscheinlichkeit $P(B|A)$ gegeben, so lässt sich die Wahrscheinlichkeit $P(A|B)$ folgendermaßen berechnen:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{2.43}$$

Es ist zu beachten, dass $P(B) > 0$ gelten muss.

Bayes Filter

Der Bayes Filter ist ein rekursiver Zustandsschätzer. Er beschreibt einen Ansatz zur Berechnung der Annahme des aktuellen Zustand eines dynamischen Systems. Die Berechnung basiert auf dem vorherigen Zustand, der Bewegung des Systems sowie aktuellen Messwerten.

Sei $\vec{x}_{t-1}$ der Zustand des dynamischen Systems zum Zeitpunkt $t-1$. Gesucht ist der Folgezustand des Systems.

$$\vec{x}_t = f(\vec{x}_{t-1}, \vec{v}_{t-1}) \tag{2.44}$$

Das Zustandsübergangsmodell $f : \mathcal{R}^{n_x} \times \mathcal{R}^{n_v} \rightarrow \mathcal{R}^{n_x}$ überführt einen Zustand $\vec{x}_{t-1}$ in seinen Folgezustand $\vec{x}_t$ unter Berücksichtigung des Prozessrauschens $\vec{v}_{t-1}$. Die beiden Werte n_x und n_v beschreiben die Dimension des Zustandsvektors bzw. des Prozessrauschens.

Ziel ist es nun, den aktuellen Zustand $\vec{x}$ rekursiv anhand vorangegangener Messungen $\vec{z}_t$ zu ermitteln.

$$\vec{z}_t = h(\vec{x}_t, \vec{n}_t) \tag{2.45}$$

Das Messmodell $h : \mathcal{R}^{n_x} \times \mathcal{R}^{n_n} \rightarrow \mathcal{R}^{n_z}$ dient der Überführung des Zustandsvektors $\vec{x}_t$ in die Messung $\vec{z}_t$ unter Berücksichtigung des Messrauschens $\vec{n}_t$. Die Dimensionen von Messrauschen und Messung werden durch n_n und n_z bezeichnet.

Der Bayes Filter hat nun die Aufgabe, die Wahrscheinlichkeit des Zustandsvektors $\vec{x}_t$ zum Zeitpunkt t zu bestimmen unter Berücksichtigung aller bisherigen Messungen $\{\vec{z}_1, ..., \vec{z}_t\}$. Es wird vorausgesetzt, dass die initiale Verteilung mit $p(\vec{x}_0|\vec{z}_0) = p(\vec{x}_0)$ gegeben ist. Dann kann die Wahrscheinlichkeitsdichte-Funktion $p(\vec{x}_t|\vec{z}_{1:t})$ folgendermaßen berechnet werden:

- **Vorhersage:**

  Es wird vorausgesetzt, dass die **a posteriori** Wahrscheinlichkeitsdichte des vorangegangenen Zeitschritts $p(\vec{x}_{t-1}|\vec{x}_{i:t-2})$ bekannt ist. Die Wahrscheinlichkeitsverteilung von $\vec{x}_t$ und $\vec{x}_{t-1}$ kann unter Vorgabe von $\vec{z}_{1:t-1}$ folgendermaßen berechnet werden

$$\begin{aligned} p(\vec{x}_t, \vec{x}_{t-1}|\vec{z}_{1:t-1}) &= p(\vec{x}_t|\vec{x}_{t-1}, \vec{z}_{1:t-1})p(\vec{x}_{t-1}|\vec{z}_{1:t-1}) \\ &= p(\vec{x}_t|\vec{x}_{t-1})p(\vec{x}_{t-1}|\vec{z}_{1:t-1}) \end{aligned}$$

  Da Gleichung 2.46 eine Markov-Kette erster Ordnung beschreibt, kann $p(\vec{x}_t|\vec{x}_{t-1}, \vec{z}_{1:t-1})$ zu $p(\vec{x}_t|\vec{x}_{t-1})$ vereinfacht werden. Integriert man über $\vec{x}_{t-1}$ so erhält man die Chapman-Kolmogorov-Gleichung des Vorhersageschritts:

$$p(\vec{x}_t|\vec{z}_{1:t-1}) = \int p(\vec{x}_t|\vec{x}_{t-1})p(\vec{x}_{t-1}|\vec{z}_{1:t-1})d\vec{x}_{t-1} \tag{2.46}$$

- **Aktualisierung**

  Da sowohl die **a priori** Verteilung $p(\vec{x}_t|\vec{z}_{1:t-1})$ als auch das Messmodell $p(\vec{z}_t|\vec{x}_t)$ gegeben sind, lässt sich daraus die *a posteriori* Verteilung mithilfe des Bayes-

Theorems bestimmen:

$$p(\vec{x}_t|\vec{z}_{1:t}) = \frac{p(\vec{z}_t|\vec{x}_t)p(\vec{x}_t|\vec{z}_{1:t-1})}{p(\vec{z}_t|\vec{z}_{1:t-1})} \tag{2.47}$$

In der Realität ist eine rekursive Ermittlung der Wahrscheinlichkeitsdichte oftmals nicht möglich. Der Kalman-Filter etwa ermöglicht eine analytische Lösung, unterliegt aber strikten Randbedingungen wie der Limitierung auf lineare Prozess- und Messmodelle sowie der Annahme, dass die Wahrscheinlichkeitsverteilung einer Normalverteilung entspricht. Gelten diese Randbedingungen nicht, so muss auf alternative Verfahren wie den Partikelfilter zurückgegriffen werden, der auf einer Punktmengen-Repräsentation der Wahrscheinlichkeitsdichte basiert.

Approximation der Wahrscheinlichkeitsdichte

Als Basis für die meisten Partikelfilter dient der *Sequential Importance Resampling* (SIR) Filter [Gordon u. a., 1993]. Hierbei erhalten die Partikel zunächst Gewichtungen, welche der Wichtigkeit (Importance) des Partikels in der Zustandsschätzung entsprechen. Sei $\{\vec{x_t^i}, w_t^i\}_{i=1}^N$ eine gewichtete Partikelmenge zum Zeitpunkt t, die als Approximation für die Wahrscheinlichkeitsverteilung dient. Hierbei bezeichnet $w_t^i = w(\vec{x_t^i})$ die Gewichtung des i-ten Partikels zum Zeitpunkt t und N die Anzahl der Partikel. Die Gewichte sind so normiert, dass gilt:

$$\sum_{i=1}^{N} w_t^i = 1 \tag{2.48}$$

Die Wahrscheinlichkeitsdichtefunktion zum Zeitpunkt t wird durch die Menge gewichteter Partikel schließlich wie folgt approximiert:

$$p(\vec{x}_t) \approx \sum_{i=1}^{N} w_t^i \delta(\vec{x}_t - \vec{x}_t^i) \tag{2.49}$$

Die Funktion δ steht für die *Dirac-Delta*-Funktion [Dirac], welche theoretisch an allen Stellen 0 annimmt außer im Ursprung.

Sei nun $q(\vec{x}_t)$ die *Importance*-Dichte der Partikel mit $\vec{x}_t^i \propto q(\vec{x}_t), i = 1, ..., N$. Für die Gewichte der Partikel gilt nun:

$$w_t^i \propto \frac{p(\vec{x}_t^i)}{q(\vec{x}_t^i)} \tag{2.50}$$

Man will nicht den gesamten Zustandsraum mit einer sehr große Partikelanzahl approximieren, sondern nur die für die Zustandsschätzung wichtigen und interessanten Stellen. An diesen Stellen will man daher möglichst viele Partikel platzieren. Die *Importance*-Dichte gibt an, wo sich diese Stellen im Zustandsraum befinden.

Der darauffolgende *Resampling*-Schritt sorgt für eine zufällige Auswahl von Partikeln für den nächsten Iterationsschritt anhand der ermittelten Wahrscheinlichkeitsverteilung. Die Wahrscheinlichkeit eines Partikels ausgewählt zu werden, richtet sich demnach nach dem zuvor ermittelten Gewicht des Partikels.

Ablauf eines Iterationsschritts

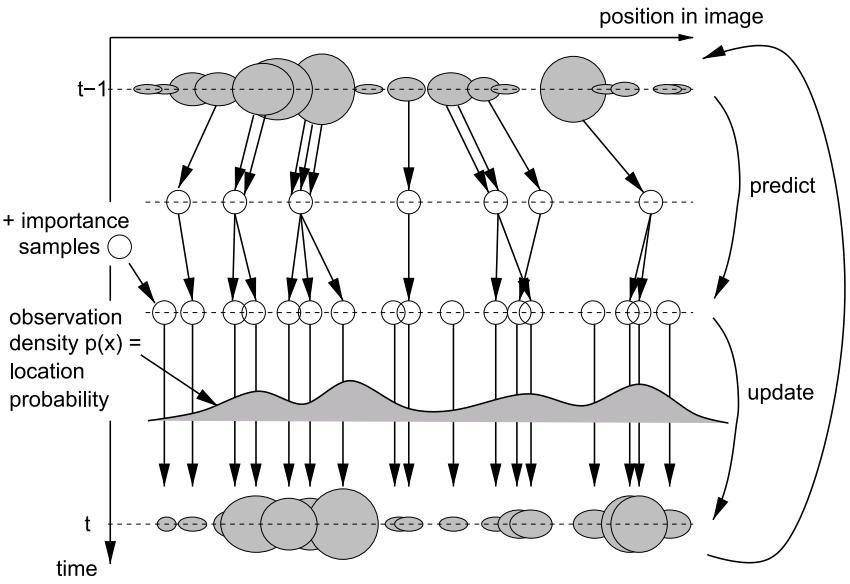


Abbildung 2.10: Grafische Repraesentation eines Iterationsschritts des Partikelfilter, entnommen aus [Happe, 2008].

Dieses Kapitel gibt einen Einblick in den allgemeinen Ablauf des Partikelfilters. Der Zustand eines Systems wird im Partikelfilter durch eine Approximation der Wahrscheinlichkeitsverteilung im Zustandsraum geschätzt.

Die Approximation der Wahrscheinlichkeitsverteilung erfolgt durch die Wahl zufälliger Stichproben (Partikel), welche bestimmten Stellen im Zustandsraum entsprechen. Je größer die Anzahl der Partikel, desto weiter nähert sich die Schätzung der Wahrscheinlichkeitsverteilung der tatsächlichen Verteilung an.

Abbildung 2.10 zeigt eine graphische Veranschaulichung eines Iterationsschritts des Partikelfilters. Die vertikal verlaufende Achse repräsentiert den zeitlichen Fortschritt, die horizontale Achse den Zustandsraum der Partikel. Zu sehen ist der Iterationsschritt zum Zeitpunkt t. Zu Beginn liegen die bereits gewichteten Partikel des vorherigen Schritts $t-1$ vor, welche als Basis der folgenden Vorhersagen dienen. Die

Größe der Ellipsen, welche jeweils für einen Partikel stehen, ist proportional zur Wahrscheinlichkeit des Partikels.

Die Vorhersagen für die nächste Iteration werden im *Predict*-Schritt getätigt. Dazu werden zunächst Partikel aus der alten Menge "mit Zurücklegen" gezogen, wobei die Wahrscheinlichkeit, einen Partikel zu ziehen, dem Partikelgewicht entspricht. Es wird "mit Zurücklegen" gezogen, da dies es ermöglicht, dass man so gute Schätzungen mit mehreren Partikeln weiterverfolgt, während schlechte Partikel komplett verworfen werden. Dieser Teilschritt nennt sich *Resampling*. *Resampling* ist nicht zwingend notwendig, trägt aber dazu bei, dass nur die besten Schätzungen weiterverfolgt werden.

Die gezogenen Partikel werden nun (stochastisch) anhand des gewählten Bewegungs- und Rauschmodells bewegt und gestreut. Die Streuung verhindert, dass mehrmals gezogene Partikel auf die identische Vorhersage hinauslaufen. Dieser Schritt nennt sich *Sampling*. Direkt nach dem *Sampling* besitzen die Partikel vorerst noch ein einheitliches Gewicht, da sie noch nicht bewertet wurden.

Dies geschieht im nächsten Schritt. Im *Update*-Schritt werden die Vorhersagen anhand des Messmodells gewichtet. Das Messmodell bestimmt, wie gut die Messung eines Partikels mit einem gegebenen Zustand übereinstimmt. Die Gewichtung erfolgt auf Basis einer *Likelihood*-Funktion, welche die Wahrscheinlichkeit eines Zustands unter dem gegebenen Messwert bestimmt. Die Gewichtung des Partikels richtet sich nach der ermittelten Wahrscheinlichkeit. Dieser Schritt wird auch *Importance*-Schritt genannt. Die resultierende gewichtete Partikelmenge dient als Eingabe für den nächsten Iterationsschritt.

2.3.5 Simulierte Abkühlung und der Annealed ParticleFilter

Die simulierte Abkühlung (*Simulated Annealing*) ist ein iteratives, heuristisches Optimierungsverfahren, welches sich zur Bestimmung einer approximativen Lösung für

Probleme mit hoher Komplexität eignet.

Viele hochdimensionale Optimierungsprobleme leiden darunter, dass sie sich durch ein kombinatorisches Vorgehen nicht effizient lösen lassen. Die Grundidee des Algorithmus ist die Nachbildung eines Abkühlungsprozesses, etwa von Metall. Während der Abkühlphase des Metalls erfolgt eine Neuausrichtung und -ordnung der Atome, sodass ein möglichst energiearmer Zustand nahe des Optimums erreicht wird. Übertragen auf das algorithmische Vorgehen entspricht dies einer Bewegung der betrachteten Partikel zu verschiedenen, zeitdiskreten Schritten. Die Temperatur beschreibt hierbei die Wahrscheinlichkeit, dass sich das Ergebnis des aktuellen Zeitschritts auch verschlechtern darf.

Ein Vorteil des Verfahrens ist, dass es in der Lage ist, lokale Optima wieder zu verlassen. Gleichzeitig ermöglicht es durch eine Verlangsamung der Partikelbewegung über die Zeit, dass Bereiche nahe der tatsächlichen, globalen Optima mit zunehmender Genauigkeit untersucht werden.

Der algorithmische Ansatz der simulierten Abkühlung basiert auf dem von Nicholas Metropolis et al. entwickelten *Metropolisalgorithmus* [Metropolis u. a., 1953]. Eine Integration in den in Abschnitt 2.3.4 vorgestellten Partikelfilter wurde zuerst von Jonathan Deutscher et al. vorgestellt [Deutscher u. a., 2000]. Die simulierte Abkühlung ist eng verwandt mit der Klasse der SIR-Filter, zu der auch der Partikelfilter gehört, und die einen stochastischen Ansatz zur Abtastung eines hochdimensionalen Zustandsraums verfolgen. Der einzige Unterschied ist, dass die zu bewegenden Partikel der simulierten Abkühlung nicht entsprechend der zugrunde liegenden, unbekannten Wahrscheinlichkeitsverteilung gestreut sind, sodass das Verfahren nicht konform mit dem Bayes-Theorem ist.

Glättung der Gewichtungsfunktion

Die Funktionsweise des im Abschnitt 2.3.4 vorgestellten Partikelfilters bleibt auch im Falle des Annealed Particle Filters erhalten. Die simulierte Abkühlung ersetzt in diesem Falle lediglich den *Sampling*-Schritt. Der restliche Algorithmus entspricht der Beschreibung aus Abschnitt 2.3.4.

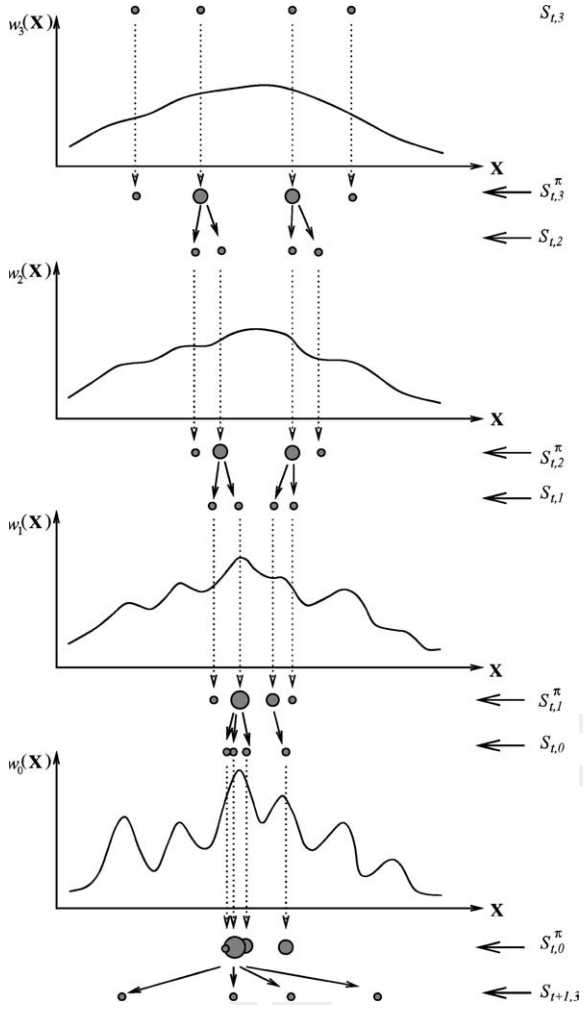


Abbildung 2.11: Darstellung der Partikelbewegung im Annealed Particle Filter mit M = 3, entnommen aus [Deutscher u. Reid, 2005]. Mit jeder Iteration nähert sich die Partikelmenge stetig dem globalen Maximum an, ohne von den lokalen Maxima abgelenkt zu werden.

Der *Sampling*-Schritt umfasst die wiederholte Ausführung einer Abkühlphase des *Simulated Annealing*, unterteilt in eine bestimmte Anzahl an Layern. In jedem Iterations-Schritt des Partikelfilters erfolgt daher eine wiederholte Gewichtung der einzelnen Partikel. Die Partikelbewegung nimmt mit der Zeit langsam ab, sodass sich die Partikel mit jeder Iteration (theoretisch) immer weiter dem globalen Optimum nähern.

Hierzu werden $m = 0...M$ Wahrscheinlichkeitsverteilungen erzeugt, wobei M die Anzahl der Annealing-Layer darstellt. Die entsprechenden Wahrscheinlichkeitsdichten werden mit $p_0(\vec{x})...p_M(\vec{x})$ bezeichnet, wobei sich eine Wahrscheinlichkeitsdichte $p_m(\vec{x})$ nur geringfügig von ihrem Nachfolger $p_{m-1}(\vec{x})$ unterscheidet. Für jeden Layer existiert nun eine Gewichtungsfunktion $w_m(\vec{z}, \vec{x})$, welche von geglättet (Layer M) zu schmalbandig (Layer 0) verläuft. Erreicht wird dies durch

$$w_m(\vec{z}, \vec{x}) = w(\vec{z}, \vec{x})^{\beta_m} \tag{2.51}$$

mit $w(\vec{z}, \vec{x})$ als originaler Gewichtungsfunktion und $\beta_M \leq \beta_{M-1} \leq \beta_0$ als eine Reihe an Exponenten für die einzelnen Layer. Die initiale Glättung der Gewichtungsfunktion erlaubt eine weitreichende Streuung der Partikel, die für den nächsten Layer ausgewählt werden, sodass lokale Optima ohne Probleme wieder verlassen werden können. Mit sinkender Temperatur reduziert sich die Partikelbewegung und die Streuung der Partikel konzentriert sich mehr auf die Umgebung des globale Optimums. Dieses Vorgehen erlaubt eine gezielte Auswahl der besten Partikel für den nächsten Iterationsschritt. Da die Partikelgewichtung durch ein stochastisches Ziehen mit Zurücklegen geschieht mit den Gewichten als Wahrscheinlichkeiten, ist die Gewichtungsfunktion der entscheidende Faktor für das Überleben einzelner Partikel.

Ablauf eines Zeitschritts

Das folgende Kapitel beschreibt den Ablauf eines Zeitschritts (Layers) des Annealed Particle Filters. Da sowohl der *Resampling*- als auch der *Importance*-Schritt dem allgemeinen Partikelfilter entsprechen, wird hier vor allem auf den Ablauf des *Sampling*-Schritts eingegangen.

Abbildung 2.11 zeigt schematisch die Partikelbewegung eines Zeitschritts mit 4 Abkühlphasen. Die horizontale Achse repräsentiert den Zustandsraum, welcher zur Vereinfachung eindimensional angezeigt wird. Die vertikale Achse bezeichnet das Gewicht des Partikels an der entsprechenden Stelle im Zustandsraum. Ein Partikel entspricht jeweils einer Ellipse, deren Größe proportional zur Wahrscheinlichkeit des Partikels ist. Begonnen wird in Layer 3, daher die absteigende Reihenfolge der Indizes. Durch die Glättung der Gewichtungsfunktion erfolgt zu Beginn eine Auswahl der Partikel über einen sehr großen Abschnitt des Definitionsbereichs. Im weiteren Verlauf der Abkühlung konzentriert sich die Auswahl zunehmend auf die lokalen Optima der Funktion, welche immer spitzer und schmalbandiger wird. Eine Formalisierung des Vorgehens wird in Algorithmus 3 beschrieben.

Algorithm 3 Iterationsschritt eines Layers des Simulated Annealing

1: Für jeden Zeitschritt t des Partikelfilters wird in Layer M gestartet mit $m = M$ als aktuellem Layer.
2: Als initiale Partikelmenge $\Pi_{t,m}$ wird die ungewichtete Partikelmenge des letzten Partikelfilter-Schritts $t-1$ genommen.
3: Jedem Partikel $\pi^i_{t,m}$ wird nun ein Gewicht zugewiesen:

$$\tilde{w}^i_{t,m} \propto w_m(\pi^i_{t,m}) \quad i = 1, ..., N \tag{2.52}$$

4: wobei N die Anzahl der Partikel und $\tilde{w}^i_{t,m}$ das normalisierte Gewicht des i-ten Partikels bezeichnet mit

$$\sum_{i=1}^{N} \tilde{w}^i_{t,m} = 1 \tag{2.53}$$

5: Es werden N Partikel aus $\Pi_{t,m}$ mit Zurücklegen gezogen. Die Wahrscheinlichkeit des Ziehens entspricht der aktuellen Gewichtung des Partikels.
6: Jeder der gezogenen Partikel geht wie folgt in seinen Nachfolger über:

$$\pi^i_{t,m-1} = \pi^i_{t,m} + X_m \tag{2.54}$$

7: X_m entspricht einer multivariaten, normalverteilten Zufallsvariable mit Mittelwert 0 und Varianz P_m. Die resultierende Partikelmenge wird mit $\Pi_{t,m-1}$ bezeichnet.
8: $\Pi_{t,m-1}$ wird als Eingabemenge für den folgenden Layer $m-1$ des Annealing-Prozesses genutzt. Dieser Prozess wird wiederholt bis $m = 0$ ist.
9: Der optimale Zustand ergibt sich aus dem gewichteten Mittel aller Partikel in $\Pi_{t,0}$:

$$\tilde{\pi_t} = \sum_{i=1}^{N} \pi^i_{t,0} * \tilde{w}^i_{t,0} \tag{2.55}$$

10: Die initiale Partikelmenge des nächsten Partikelfilter-Schritts ergibt sich erneut aus:

$$\pi^i_{t+1,M} = \pi^i_{t,0} + X_0 \tag{2.56}$$

Laufzeitparameter

Das Verfahren lässt sich anhand einiger kritischer Laufzeitparameter charakterisieren, welche im Folgenden vorgestellt werden. Die im vorangegangenen Abschnitt vorge-

stellte Gewichtungsfunktion ist abhängig von zweierlei Parametern: Die Exponenten β_m bestimmen den Abkühlungsgrad der Gewichtungsfunktion, deren Wahl wiederum beeinflusst wird von der Überlebensrate α der Partikel. Die Überlebensrate bestimmt zudem den Grad der Partikelbewegung, welcher im Diffusions-Schritt durchgeführt wird. Eine große Rolle in der Exaktheit der Resultate spielt schließlich die Wahl der Anzahl der Annealing-Layer.

Überlebensrate α: Die Überlebensrate (*survial rate*) α beschreibt wie viele Partikel effektiv den *Resampling*-Schritt einer Abkühlphase überleben. Die Überlebensrate kann aus den Gewichten der N Partikel zum Ende einer Abkühlphase abgeleitet werden. Aus der *survival diagnostic* D

$$D = \left(\sum_{i=1}^{N} \tilde{w}_i^2 \right)^{-1} \tag{2.57}$$

ergibt sich nach MacCormick [MacCormick u. Isard, 2000] die Überlebensrate als

$$\alpha = \frac{D}{N} \tag{2.58}$$

Liegt eine perfekte Hypothese vor mit $(\vec{x}_1, 1)$ und sind alle Partikel mit 0 gewichtet $\{(\vec{x}_i, 0)_{i=2}^{N}\}$, so ist $D = 1$ minimal. Dies würde zum Überleben eines Partikels aus N führen.

Ist die Wahrscheinlichkeit über alle Partikel gleich verteilt, gilt $\tilde{w}_i = \frac{1}{N}, i = 1..N$. Die *survival diagnostic* berechnet sich folglich aus

$$D = \left(\sum_{i=1}^{N} \frac{1}{N^2} \right)^{-1} = \left(N * \frac{1}{N^2} \right)^{-1} = N$$

Alle Partikel überleben den Auswahlprozess und D wird maximal. Der Wertebereich für die Überlebensrate liegt demnach zwischen 0 und 1.

Gewichtungsexponenten β: Der Wert β bestimmt die Glattheit bzw. Abkühlrate der Gewichtungsfunktion in jedem Annealing-Schritt. Große Werte von β führen zu schmalbandigen Funktionen mit großen Ausschlägen, kleine Werte haben den gegenteiligen Effekt. β wird in jedem Zeitschritt genau so gewählt, dass eine vorgegebene Überlebensrate erreicht werden kann. [Deutscher u. a., 2000] zeigt, dass die *survival diagnostic* D in Abhängigkeit von β monoton fallend ist, sodass

$$D(\beta) = N\alpha \tag{2.59}$$

eine eindeutige Lösung besitzt. Mittels Gradientenabstieg kann die Fehlerfunktion

$$\Delta_\alpha(\beta) = (\alpha_m - \alpha_{init}(\beta)) \tag{2.60}$$

gelöst werden und das gesuchte β_m^t bestimmt werden. Die Funktion α_{init} berechnet hierbei die Überlebensrate, welche sich aus dem aktuellen Wert für β ergeben würde. Für jede Änderung von β_m^t während des Gradientenabstiegs müssen die Gewichte der Partikel nicht neu berechnet werden. Da $w_m(\vec{z}, \vec{x}) = w(\vec{z}, \vec{x})^{\beta_m}$ gilt, genügt es die rohen Gewichte w^i für alle Partikel $i = 1..N$ zu speichern und dann verschiedene Exponenten darauf anzuwenden.

Diffusionsrate: Die Bewegung von Partikeln beim Übergang zweier Abkühlphasen erfolgt auf Basis eines Rauschmodells. Dabei wird eine multimodale, normalverteilte Zufallsvariable auf den Zustandsvektor des Partikels aufaddiert. Eine Parametrisierung der Normalverteilung erfolgt durch Mittelwert μ, welcher zumeist als 0 angenommen wird, und Varianzvektor P.

Die zuvor vorgestellte Überlebensrate α_m bestimmt die Abkühlrate der Partikel-Bewegung in jedem Zeitschritt. Der Varianzvektor P_m geht dabei wie folgt in seinen

Nachfolger über:

$$P_{m-1} = P_m * \alpha_m \tag{2.61}$$

Die Partikelbewegung ist folglich maximal in Layer M und minimal in Layer 0, da $0 \leq \alpha \leq 1$ gilt.

Anzahl der Annealing-Layer: Die Gesamtzahl der auszuwertenden Partikel ergibt sich aus dem Produkt aus der Anzahl der Partikel pro Iterationsschritt sowie der Anzahl der Iterationen. Will man die Zahl der Auswertungen konstant halten, so existieren zwei Möglichkeiten der Parametrisierung, entweder eine Erhöhung der Partikelzahl bei gleichzeitiger Reduktion der Annealing-Layer oder das gegenteilige Vorgehen.

In [Deutscher u. a., 2000] wird beschrieben, dass eine Verdopplung der Iterationen bis zu einem gewissen Punkt eine Halbierung der Anzahl der Partikel zulässt. Die schlussendliche Wahl beider Parameter muss von Problem zu Problem unterschieden werden. Generell jedoch kann gesagt werden, dass hochdimensionale Suchprobleme eine höhere Partikelzahl erfordern als niedrigdimensionale. Will man zudem die Exaktheit der Ergebnisse verbessern, so empfiehlt es sich die Iterationszahl zu erhöhen, denn erst durch wiederholtes Iterieren erfolgt eine Abtastung der interessanten Bereiche des Suchraums.

2.4 RODOS® -System des Fraunhofer ITWM

Dieses Kapitel widmet sich dem **R**obot based **D**riving and **O**peration **S**imulator (RODOS®) des Fraunhofer ITWM in Kaiserslautern [Kleer u. a., 2014]. Dieses Simulationssystem für Bodenfahrzeuge widmet sich vor allem dem Einfluss des menschlichen Operators auf die Simulation, ein Faktor, der von vielen kommerziell genutzten Simulatoren vernachlässigt wird [Kleer u. a., 2014].

Durch eine realistische Simulation von Kräften und Beschleunigung, welche auf den Nutzer wirken, wird so ein hoher Grad an Immersion erreicht. Die grundlegenden Vorteile des Konzepts sind ein hohes Maß an Kontrollierbarkeit von sowohl Umwelt- als auch Fahrzeugzuständen, Reproduzierbarkeit, Sicherheit, vor allem im Kontext von risikoreichen Manövern, sowie die Möglichkeit verschiedene Versionen von Hard- und Softwaremodulen testen zu können, ohne ändernd in das physikalische System eingreifen zu müssen.

Der allgemeine Aufbau des Systems umfasst einen industriellen Roboterarm mit 1000kg Nutzlast, ein Cluster bestehend aus 18 Projektoren, ein Sound-System mit Vibrations-Simulation sowie eine handelsübliche Bagger-Kabine, welche an der Spitze des Roboterarms montiert ist.

2.4.1 Human-In-The-Loop Konzept

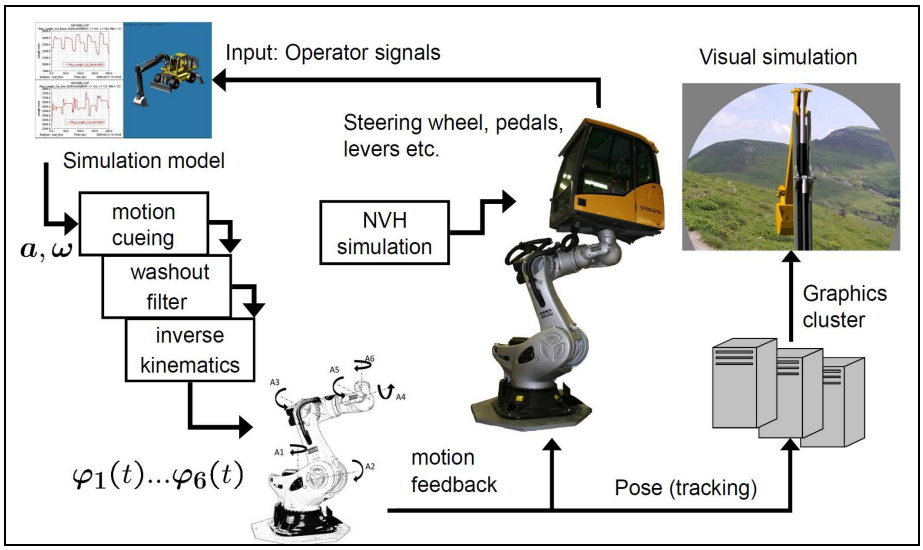


Abbildung 2.12: Grafische Darstellung des Human-In-The-Loop Konzepts des RODOS-Systems, aus [Kleer u. a., 2014].

Im Folgenden wird das in Abbildung 2.12 beschriebene **Human-In-The-Loop**-Konzept näher erläutert. Über die Eingabegeräte in der Fahrerkabine ist es dem

Nutzer möglich mit dem System zu interagieren. Pedale sind mit Analogsensoren ausgestattet, während das Lenkrad und die Joysticks über ein CAN-Interface kommunizieren. Die auf diese Weise anfallenden Daten werden mithilfe eines Echtzeitcomputers innerhalb der Kabine gesammelt und an ein weiteres Echtzeitcluster weitergeleitet. Die Nutzung eines gesonderten Rechners in der Kabine erlaubt die Standardisierung der Schnittstelle zwischen verschiedenen Kabinentypen und der Hardware des Simulators.

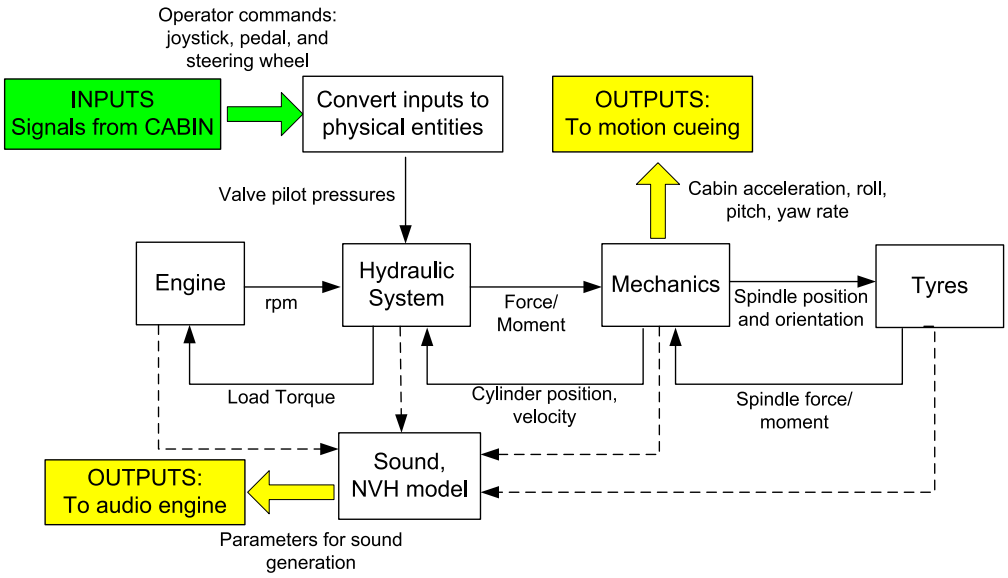


Abbildung 2.13: Grafische Darstellung der Subsysteme von RODOS, aus [Kleer u. a., 2014].

Eine Modellierung des simulierten Fahrzeugs umfasst eine Vielzahl an Komponenten wie Motor, Hydraulik, Reifen, mechanische Verbindungen, Elektronik und Informationssysteme. Abbildung 2.13 zeigt die verschiedenen Subsysteme sowie deren Schnittstellen untereinander. Eine Steuerung des Fahrzeugmodells erfolgt durch die zuvor genannten Nutzereingaben. Durch die Nutzung einer Standard-Kabine stehen alle Eingabegeräte zur Verfügung, welche man auch in einem realen Fahrzeug finden würde.

Auswirkungen einer Nutzereingabe machen sich auf drei separaten Ausgabekanälen bemerkbar:

- Die Bewegung der Simulator-Kabine erfolgt auf Basis von Beschleunigungen sowie Rotationen um die drei Raumachsen der Kabine. Diese Werte werden an die **Motion-Cueing**-Komponente gesendet, welche schließlich die Bewegung der Kabine kalkuliert.
- Verschiedene Nutzereingaben führen zu unterschiedlichen Sound-Ausgaben. Es ist sowohl möglich bestehende Sound-Samples abzuspielen als auch Rausch- und Vibrationseinflüsse hinzuzufügen.
- Der in Abbildung 2.13 nicht gezeigte Teil der Ausgabe umfasst die Grafik-Engine des Simulators, welche für eine Visualisierung der Szene und des Fahrzeugs sorgt.

Eine Modellierung der vorgestellten Subsysteme sowie des kinematischen Fahrzeugmodells erfolgt entweder über selbstgeschriebenen, echtzeitfähigen Code oder über die Nutzung von kommerzieller Simulations-Software wie MATLAB® [MATLAB, 2014].

Es bedarf hierbei einer Bereitstellung von Schnittstellen zwischen dem Fahrzeugmodell und anderer Komponenten wie dem Robot Motion Control System, welches sich unter anderem um das Motion Cueing kümmert, der inversen Kinematik des Roboterarms, der Verarbeitung von Kabinensignalen sowie Monitoring-Komponenten, die sich um Sicherheitsbeschränkungen des Roboterarms kümmern.

2.4.2 Hardware-Architektur

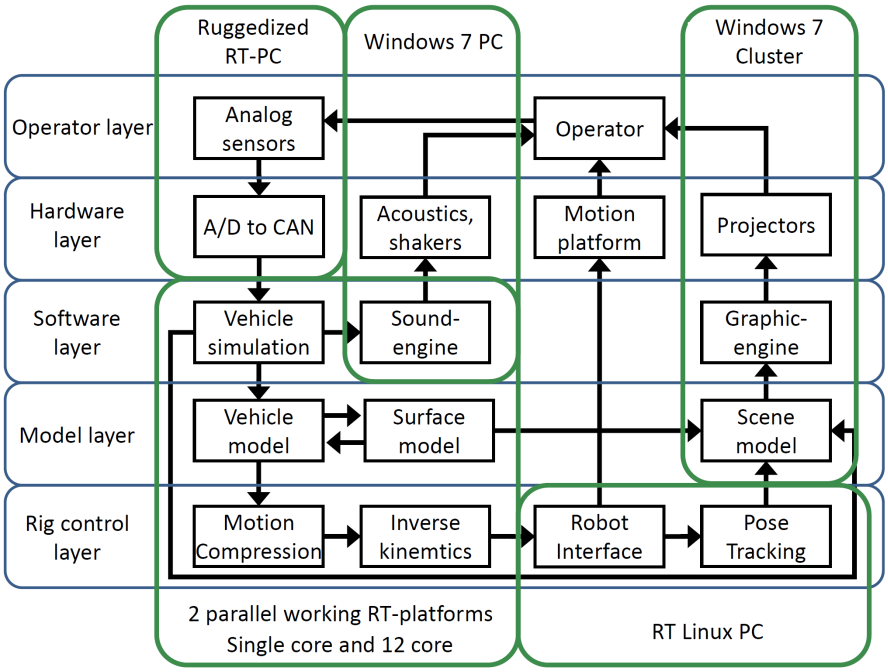


Abbildung 2.14: Die Hardwarearchitektur von RODOS, welche die hierarchische Aufteilung des Systems veranschaulicht, entnommen aus [Kleer u. a., 2014]. Die grünen Umrandungen zeigen jeweils an, auf welchen Plattformen die einzelnen Ebenen ausgeführt werden.

Der Berechnungsaufwand einer im hohen Maße immersiven Simulation übersteigt die Kapazitäten eines einzelnen Rechensystems. Stattdessen wird ein Cluster verschiedener Systeme genutzt, welchen jeweils eine gesonderte Aufgabe zufällt.

Neben der gesteigerten Leistungsfähigkeit erlaubt ein solches Cluster eine klare Trennung der Belange zwischen voneinander unabhängigen Komponenten. Der modulare Aufbau ermöglicht den flexiblen Austausch und die Hinzunahme von Komponenten ohne Änderungen an den Komponenten selbst vornehmen zu müssen.

Abbildung 2.14 zeigt den generellen Aufbau des Clusters. Der Aufbau lässt sich in 5 hierarchische Ebenen unterteilen, welche jeweils einem anderen Aufgabengebiet zugeordnet werden:

- Operator Layer: Auf dieser Ebene kann der Nutzer Eingaben über die von der Kabine zur Verfügung gestellten Eingabegeräte tätigen.
- Hardware Layer: Dieser Ebene enthält alle Feedback Devices wie den Roboterarm und die Hardware des Sound-Systems.
- Software Layer: Auf dieser Ebene werden alle Eingabe- und Ausgabedaten der Simulations-Engine verarbeitet. Von zentraler Bedeutung ist dabei das Fahrzeugmodell, da alle bedeutenden Interaktionen mit diesem Teil des Modells stattfinden.
- Model Layer: Diese Ebene enthält das Fahrzeugmodell zusammen mit Szene- und Oberflächenmodell der Umgebung. Aus dem Szenemodell lassen sich Eingaben für die Grafik-Engine generieren, in der Clustering, Synchronisierung und Image Blending zwischen den 18 Projektoren ausgeführt werden.
- Rig Control Layer: Diese Ebene enthält allgemeine Steueralgorithmen sowie Algorithmen für das Motion Rendering.

Der Nutzer bildet die zentrale Steuereinheit des Systems, indem er auf die visuellen und akustischen Reize sowie die Bewegung der Kabine mit einer entsprechenden Eingabe reagiert. Die möglichen Eingabevariablen umfassen Lenkwinkel, Pedale sowie Ausschläge der Joysticks.

3 Eigene Arbeit

Das vorliegende Kapitel teilt sich in zwei voneinander unabhängige Teilabschnitte.

Der erste Abschnitt beschreibt das Vorgehen zur Untersuchung der Leitfragen aus Kapitel 1.2. Dazu wurde eine Fehleranalyse durchgeführt, welche einen analytischen Teil und einen experimentellen Teil besitzt. Während der analytische Teil versucht allgemeine Gesetzmäßigkeiten in der Fortpflanzung der Fehler zu finden, beschäftigt sich der experimentelle Teil mit der Betrachtung konkreter Fahrsituationen und Fahrzeugmodelle. Die einzelnen Komponenten dieser Betrachtungen werden in der Folge vorgestellt.

Im zweiten Abschnitt werden die Ergebnisse der Kooperation mit dem Fraunhofer Institut dargelegt. Ziel der Zusammenarbeit war die Evaluation der Assistenz in einer Simulationsumgebung unter kontrollierbaren Bedingungen. Dazu wurde eine Reihe an Softwarekomponenten entwickelt, welche in der Folge vorgestellt werden.

3.1 Gesamtkonzept

Die folgenden Abschnitte orientieren sich an den in Kapitel 1.2 formulierten Leitfragen.

Die erste Leitfrage richtet sich nach der Auswirkung von Fehlern in der Assistenz. Die in diesem Zusammenhang betrachteten Ansätze sind zum einen analytischer, zum anderen experimenteller Natur.

In Kapitel 3.2 erfolgt zunächst eine Identifizierung der Fehlerquellen im Datenfluss der Assistenz-Software. Unterschieden wird dabei zwischen Fehlern in der Berechnung der Assistenz-Trajektorien und Fehlern in der Darstellung. Erstere wirken sich auf die Form der Trajektorien aus, welche nicht der tatsächlichen Fahrzeugbewegung entsprechen. Letztere wirken sich auf die Position und Ausrichtung der Trajektorie in der Umgebung des gezeigten Rückfahrkamerabilds aus.

Auf Basis der identifizierten Fehler wird daraufhin eine quantitative Betrachtung der Auswirkung unterschiedlicher Fehlertypen durchgeführt. Dieses Vorgehen soll helfen, Fehlerzusammenhänge auf grundlegender Ebene zu erkennen und zu bewerten.

Eine Validierung der Erkenntnisse erfolgte anhand von Experimenten, welche mithilfe einer Testumgebung durchgeführt wurden. Darauf wird im Rahmen der zweiten Leitfrage näher eingegangen.

Die zweite Leitfrage befasste sich damit, wie groß der Eingabefehler sein darf, damit Ausgabefehler sich in einem tolerablen Bereich bewegen. Eine Untersuchung dieser Frage erfolgte anhand der Durchführung qualitativer Experimente. Zu diesem Zweck wurde eine Testumgebung benötigt, welche die Simulation verschiedener Fehlerkonfigurationen eines Manövers ermöglicht.

Der betrachtete Fehler ist hochdimensional. Ein kombinatorisches Vorgehen zur Bestimmung von Fehlergrenzen war daher nicht praktikabel. Der Annealed Particle Filter beschreibt ein heuristisches Verfahren, welches sich vor allem für hochdimensionale Optimierungsprobleme eignet, siehe Abschnitt 2.3.5. Dieses soll dafür genutzt werden, eine möglichst große Abdeckung des Fehlerraums zu erzielen. Die Nutzung dieses Verfahrens zur Bestimmung von Fehlergrenzen beim Manövrieren ist nach Kenntnis des Autors bislang unerforscht und stellt somit eine Neuerung dar.

Ein Partikel wird in diesem Zusammenhang durch eine Fehlerkonfiguration repräsentiert. Die Gewichtung eines Partikels erfolgt durch die Ausführung eines Manövers unter den Einflüssen der durch den Partikel beschriebenen Fehler. Im Vorfeld soll der Nutzer in der Lage sein, tolerierbare Fehlergrenzen zu definieren, welche sich auf

die gewählte Gewichtungsfunktion auswirken. Ein Partikel erhält somit ein höheres Gewicht, je näher er der gewünschten Fehlergrenze kommt.

Die Gewichtung der Partikel erfordert mehrere Komponenten, welche in der Folge vorgestellt werden.

In Kapitel 3.3 erfolgt die Vorstellung eines Controller, welcher entwickelt wurde, um das menschliche Fahrverhalten bei der Nutzung der Assistenz zu simulieren. Der hier betrachtete Nutzer handelt "ideal", das heißt er reagiert unmittelbar auf die Informationen, welche ihm die Assistenz liefert. Dieser Controller wird zur automatisierten Ausführung des Manövers genutzt und wird in der Folge als die bestmögliche Nutzung der Assistenz betrachtet.

In Kapitel 3.4 wird die Wahl der Gewichtungsfunktion näher erläutert. Zur Gewichtung einzelner Partikel wird ein skalares Maß benötigt, durch welches ein Manöver bewertet werden kann. In Abschnitt 2.2 wurden verschiedene Gütemaße vorgestellt, welche eine Bewertung anhand verschiedener Kriterien ermöglichen. Die hier vorgestellte Methode führt zunächst eine Gewichtung der Gütemaße durch anhand der Nähe zu den definierten Fehlergrenzen. Die normalisierten Kennzahlen werden dann anhand ihrer Priorität aggregiert und zu einer einzigen Maßzahl zusammengeführt.

In Kapitel 3.5 werden schließlich die realisierten Komponenten der Testumgebung vorgestellt. Dabei wird vor allem auf die Zusammenhänge zwischen einzelnen Komponenten eingegangen. Zudem erfolgt eine Vorstellung des Konzepts zur Modellierung von Fehlern in der Software.

Den Abschluss der Betrachtung umfasst die Durchführung qualitativer Experimente für verschiedene Fahrsituationen. Als Eingabe erhält das Testtool Fehlergrenzen sowie eine initiale Partikelmenge. Über mehrere Optimierungsschritte soll schließlich eine Menge an Fehlerkonfigurationen erzeugt werden, welche den gewünschten Fehlergrenzen so nah wie möglich kommen. Die Durchführung der Experimente wird in Kapitel 4 beschrieben. Eine anschließende Evaluation des Verfahrens sowie der Experimente findet schließlich in Kapitel 5 statt.

Neben der Betrachtung der Leitfragen aus Kapitel 1.2 werden die Ergebnisse der Kooperation mit dem Fraunhofer Institut in Kaiserslautern vorgestellt.

Kapitel 3.6 widmet sich der Vorstellung des Konzepts zur Portierung der benötigten Assistenz- und Softwarekomponenten auf RODOS®. Da das System einem modularen Aufbau folgt, konnten die benötigten Komponenten unabhängig voneinander entwickelt werden. Betrachtet wird die Realisierung eines Fahrzeugmodells, welches als Grundlage der Kinematikberechnungen dient, sowie die Implementierung von Netzwerkschnittstellen, welche es der Assistenz ermöglichen mit der Fahrerkabine und der Kinematikkomponente zu kommunizieren.

Aufgrund von Zeitmangel sowie Problemen bei der Portierung von statischen Bibliotheken gegen Ende der Arbeit gelang es nicht, das System in einen lauffähigen Zustand zu bringen. Die angedachten Tests in der Simulationsumgebung (siehe Abschnitt 1.2) konnten somit im Rahmen der Arbeit nicht durchgeführt werden.

3.2 Fehleranalyse der Assistenz

Das folgende Kapitel beschäftigt sich mit der Identifizierung und der Analyse von Fehlern im Datenfluss des Fahrer-Assistenzsystems.

Zunächst werden in Abschnitt 3.2.1 potentielle Fehlerquellen im Datenfluss der Assistenz-Software identifiziert und anhand ihrer Relevanz für die Modellierung bewertet.

In Abschnitt 3.2.2 werden daraufhin die Fehler betrachtet, welche zu einer inkorrekten Berechnung der Assistenz-Trajektorien führen.

Abschnitt 3.2.3 beschäftigt sich schließlich mit der inkorrekten Darstellung der Hilfskurven.

3.2.1 Identifizierung von Fehlerquellen im Datenfluss

In Kapitel 2.1.3 wurde die Funktionsweise der optischen Fahrerassistenz u.a. anhand des Datenflussdiagramms aus Abbildung 2.7 vorgestellt. Dieses Datenflussdiagramm dient im Folgenden als Basis, um Fehler identifizieren zu können, die Auswirkungen auf die Visualisierung der Hilfs-Trajektorien haben.

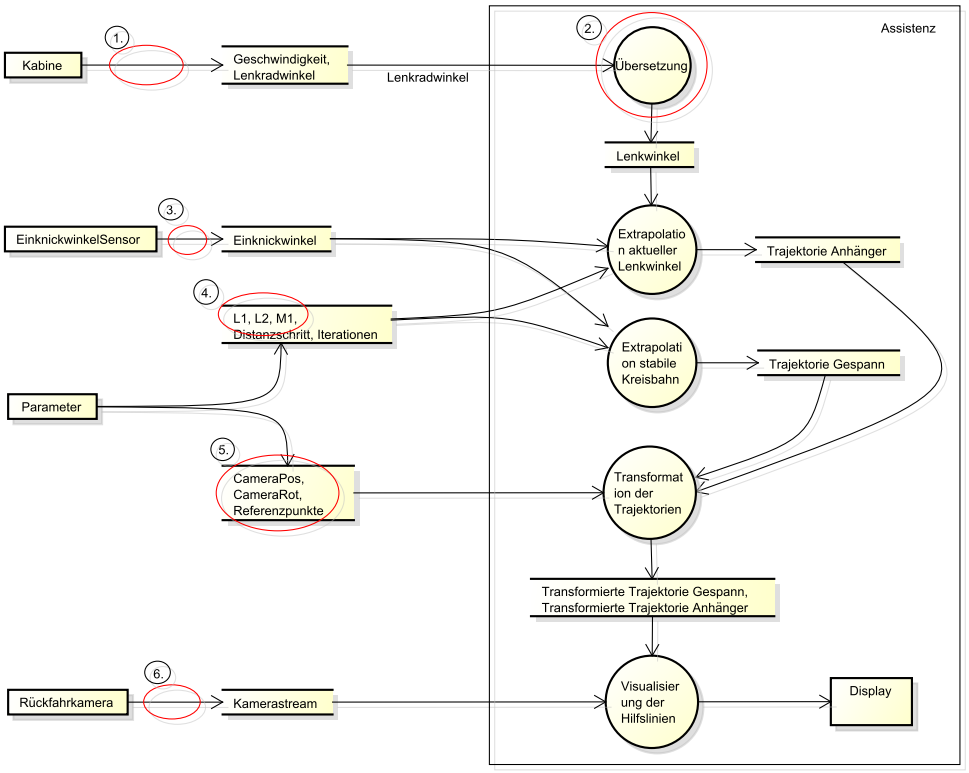


Abbildung 3.1: Die Fehlerquellen im Datenfluss, rot hervorgehoben.

In Abbildung 3.1 ist eine modifizierte Variante des Diagramms dargestellt, welche die kritischen Stellen hervorhebt. Gekennzeichnet wurden 6 Stellen im Datenfluss, welche als besonders fehleranfällig eingestuft wurden. Es erfolgt zunächst eine Klassifizierung der Eingabefehler anhand des Fehlertyps.

Deterministische Fehler

Zu den deterministischen Fehlern gehören die **Übersetzung des Lenkradwinkels in den Lenkwinkel** (Punkt 2), Fehler in der **Fahrzeugvermessung** (Punkt 4) sowie Fehler in der **Vermessung der Kamera-Pose** (Punkt 5). Der Einfluss dieser Fehler ist zu jedem Zeitpunkt konstant.

Die Übersetzung von Lenkradwinkel und Lenkwinkel ist in der Regel nicht linear und unterscheidet sich von Fahrzeug zu Fahrzeug. Da die korrekte Übersetzung nicht direkt einsehbar ist bzw. vom Hersteller nicht zur Verfügung gestellt wird, muss eine Näherungslösung bestimmt werden. Dazu wird eine Menge an Referenz-Lenkradwinkeln gewählt und über Experimente der zugehörige Lenkwinkel bestimmt. Schließlich lässt sich zwischen den Referenzpunkten interpolieren, um eine Approximation des tatsächlichen Lenkwinkels zu erhalten.

Es ist zu erwarten, dass die experimentelle Bestimmung der Lenkwinkel fehlerbehaftet ist und so die Interpolation zu Folgefehlern im "gemessenen" Lenkwinkel führt, welcher sich auf die Berechnung der Anhänger-Trajektorie auswirkt.

Da hier auf keinerlei Grundwahrheit in Form von exakten Übersetzungen zurückgegriffen werden kann, wird dieser Fehler in der Analyse vernachlässigt. In den folgenden Kapiteln wird die Übersetzung demnach als linear angenommen. Experimente der Arbeitsgruppe zeigten, dass ein Verhältnis von 15 zu 1 zwischen Lenkradwinkel und Lenkwinkel eine gute Näherung darstellt.

Die wichtigen Fahrzeugabmessungen in der Berechnung der Assistenz-Trajektorien umfassen *L1*, *L2* und *M1*. Diese Werte erhält man entweder über Angaben des Herstellers oder durch manuelle Vermessung.

Herstellerangaben stellen oftmals lediglich Zielvorgaben für ein Fahrzeug dar. Die Werte der tatsächlichen Verarbeitung können von diesen Vorgaben abweichen. Das Problem bei manueller Vermessung ist es, dass entweder nicht genau gemessen wird, oder die Referenzpunkte, von denen man die drei Abstände bestimmt, nicht korrekt

bestimmt werden, etwa der exakte Mittelpunkt einer Achse.

Ein Fehler dieser Art wird in Abschnitt 3.2.2 näher behandelt.

Der letzte deterministische Fehler umfasst die Vermessung und Bestimmung der Kamera-Pose, bestehend aus der 3-dimensionalen Position $\vec{p}_{cam}$ und Ausrichtung $\vec{\theta}_{cam}$ der Rückfahrkamera.

Die Kamera-Pose wird im Anhänger-Koordinatensystem repräsentiert, dessen Ursprung in der Mitte der Anhänger-Hinterachse liegt. Die Vermessung der Kamera-Pose ist somit zusätzlich abhängig von der Vermessung des Fahrzeugs. Sind Länge des Fahrzeugs oder die Position der Hinterachse ungenau vermessen, so nützt selbst eine exakte Vermessung der Kamera nichts, da das Referenz-Koordinatensystem fehlerbehaftet ist.

Dieser Fehler wird in Abschnitt 3.2.3 detaillierter behandelt.

Sensor-induzierte Fehler

Sensoren sind potentiell fehlerbehaftet und liefern nur eine Näherung des tatsächlich anliegenden physikalischen Wertes. Der Einfluss dieser Fehler variiert dabei zu jedem Zeitpunkt.

Unterschieden werden zwei Arten von Fehlern, die die Sensorwerte beeinflussen können, zum einen das normalverteilte, additive Rauschen, parametrisiert über Mittelwert μ und Standardabweichung σ, zum anderen die Auflösung des Sensors, definiert über den kleinsten messbaren Abstand zweier aufeinanderfolgender Sensorwerte Δr.

Da die Sensoren zumeist als Blackbox vorliegen, muss man sich als Referenz für die genannten Parameter auf Herstellerangaben verlassen.

Sensor-induzierte Fehler äußern sich in der Messung des **Lenkradwinkels** (Punkt 1) sowie des **Einknickwinkels** (Punkt 3).

Der Lenkradwinkel ist nicht direkt ablesbar und muss über einen entsprechenden Sensor am Lenkrad gemessen werden, wobei die zuvor genannten Verfälschungen des Sensorwertes auftreten. Da der Fehler in der Übersetzung zum Lenkwinkel nicht betrachtet wird, und die Übersetzung als linear angenommen wird, können die betrachteten Fehlermodelle direkt auf den Lenkwinkel angewandt werden.

Die Messung des Einknickwinkels unterliegt denselben potentiellen Fehlern wie die Messung des Lenkradwinkels. Die hierfür zuständigen Sensoren sitzen für gewöhnlich direkt an der Kupplung zwischen Zugfahrzeug und Anhänger (Potentiometer, optische Inkrementgeber) und sind somit direkten Umwelteinflüssen ausgesetzt, was die Messung zusätzlich beeinflussen kann.

Die Arbeitsgruppe Echtzeitsysteme hat zudem einen optischen Einknickwinkelsensor entwickelt, welcher den Einknickwinkel anhand Marker-basierten Ansatzes berechnet [Fuchs u. a., 2014].

Eine genauere Betrachtung von Sensorfehlern erfolgt in Abschnitt 3.2.2.

Fehler durch Verzögerungen

Fehler durch Verzögerung beschreiben Fehler, welche durch das Alter von Daten entstehen. Betroffen sind die **Verzögerung in der Akquirierung** des Lenkradwinkels bzw. des **Lenkwinkels** (Punkt 1), des **Einknickwinkels** (Punkt 3) sowie des **Kamerabilds** der Rückfahrkamera (Punkt 6).

Die zeitliche Verzögerung zwischen Aufnahme und Verwertung eines Messwerts Δt führt dazu, dass die Assistenz auf veralteten Daten arbeitet. Die resultierenden Hilfskurven zeigen somit die Fahrzeugbewegung zu einem bereits vergangenen Zeitpunkt an. Dies hat zur Folge, dass dem Nutzer einerseits falsche Informationen geliefert werden, andererseits verringert sich die Reaktivität der Trajektorien, was zu einem unintuitiven Verhalten führt.

Abschnitt 3.2.3 widmet sich einer genaueren Betrachtung dieses Fehlertyps.

Die zeitliche Verzögerung des Kamerabilds $\Delta\ t_{cam}$ äußert sich darin, dass auf dem Assistenz-Monitor eine veraltete Ansicht gezeigt wird. Da die Berechnung der Trajektorien in diesem Fall nicht beeinflusst wird, findet keine Beeinträchtigung der Steuerung des Fahrzeugs statt.

Probleme entstehen, wenn die Trajektorien und die Umgebung nicht synchron angezeigt werden. Visiert man einen Punkt im veralteten Kamerabild an, so stimmt dieser nicht mit der aktuellen Position des Punktes überein. Dies kann zu Fehlern in der Navigation führen.

Dieser Fehler wird schließlich in Abschnitt 3.2.3 genauer betrachtet.

3.2.2 Fehler in der Berechnung der Trajektorien

Fehler in der Berechnung der Trajektorien treten auf, wenn der Assistenz fehlerhafte Sensorwerte geliefert werden. Die berechneten Hilfs-Trajektorien entsprechen folglich nicht der tatsächlichen Bewegung des Fahrzeuggespanns bzw. des Anhängers. Die Auswirkungen dieser Fehler werden in der Folge analysiert.

Fehler in der Fahrzeugvermessung

Fehler in der Fahrzeugvermessung betreffen vor allem die für die kinematischen Berechnungen wichtigen Werte *L1*, *L2* und *M1*. Das in Kapitel 2.1.1 vorgestellte kinematische Modell ist iterativ und basiert auf einer Reihe nicht-linearer Differentialgleichungen. Für diese Gleichungen existiert leider keine geschlossene Lösung, sodass es nicht trivial ist, einen Fahrzeugzustand zu einem bestimmten Zeitpunkt direkt zu berechnen. Es muss stattdessen der Umweg über mehrere sukzessive Iterationen des Modells gegangen werden, wenn man Folgezustände des Fahrzeugs berechnen will.

Unterliegt einer der Eingangsparameter einem Fehler, so entsteht eine Abweichung zwischen fehlerfreiem und fehlerbehaftetem Folgezustand. Diese Abweichung ist im

Falle der nicht-linearen kinematischen Gleichungen eines Fahrzeugs ebenfalls nicht linear. Eine (exakte) allgemeine Lösung skaliert äußerst schlecht, da man den initialen Fehler über jede einzelne Iteration verfolgen müsste. Die Zustandsinkremente werden pro Iterationsschritt immer tiefer in trigonometrischen Funktionen geschachtelt (etwa bei $\Delta\Theta_{12}$) und die gesamte Formel verliert sich in ihrer Komplexität.

Um effizient analysieren zu können, wie sich geringe Abweichungen in einem oder mehreren Eingabewerten auf den folgenden Fahrzeugzustand auswirken, müssen gewisse Vereinfachungen des kinematischen Modells getroffen werden.

Im Folgenden wird daher der Umweg über die Linearisierung des Fahrzeugmodells gegangen. Es wird angenommen, dass Fehler in den Eingabeparametern sehr gering sind. Nimmt man diese Parameter als Entwicklungsstellen, so ist bildet die resultierende Linearisierung eine hinreichend genaue Approximierung der kinematischen Gleichungen wie in Kapitel 2.3.2 beschrieben.

Zunächst wird das kinematisch Modell eines *General-2-Trailers* wiederholt und in der Folge angepasst. Die Eingabeparameter umfassen:

$$\vec{c} = \begin{pmatrix} x_1 \\ y_1 \\ \Theta_1 \\ \phi \\ \Delta\Theta_{12} \end{pmatrix} \quad \vec{c} \in C \tag{3.1}$$

wobei $\vec{c}$ einen Fahrzeugzustand und C die Menge aller Fahrzeugzustände beschreibt. Statt α_{L_1} wird hier der Bezeichner ϕ für den aktuellen Lenkwinkel des Zugfahrzeugs genutzt.

Sei zudem

$$\vec{v} = \begin{pmatrix} v_1 \\ \phi_v \end{pmatrix} \quad \vec{v} \in V \tag{3.2}$$

der zugehörige Steuervektor des Fahrzeugs, wobei V die Menge aller Steuervektoren beschreibt. Statt der Lenkwinkeländerung wird direkt der neu anliegende Lenkwinkel ϕ_v betrachtet, ein Vorgehen wie es auch in *ezKine* praktiziert wird.

Zum Zweck der Analyse der Fahrzeuggeometrie wird ein weiterer Vektor $\vec{p}$ als Eingabe hinzugenommen, welcher die Fahrzeugabmessungen enthält:

$$\vec{p} = \begin{pmatrix} L_1 \\ L_2 \\ M_1 \end{pmatrix} \quad \vec{P} \in P \tag{3.3}$$

P beschreibt hierbei analog die Menge aller Fahrzeugabmessungen.

Die resultierende Übergangsfunktion $f : C \times V \times P \to C$ von einer Fahrzeugkonfiguration auf die nächste kann nun entsprechend der Differentialgleichungen aus Kapitel 2.1.1 gebildet werden:

$$f(\vec{c}, \vec{v}, \vec{p}) = \begin{pmatrix} x_1 + v_1 \cos\Theta \\ y_1 + v_1 \sin\Theta \\ \Theta_1 + v_1 \frac{\tan\phi}{L_1} \\ \phi_v \\ \Delta\Theta_{12} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin\Delta\Theta_{12}}{L_2} + \frac{M_1 \cos\Delta\Theta_{12} \tan\phi}{L_1 L_2}) \end{pmatrix} \tag{3.4}$$

Es wird nun angenommen, dass die oben beschriebenen Eingabeparameter konstanten, additiven Fehlern unterliegen.

$$\vec{c}_\epsilon = \vec{c} + \epsilon_{\vec{c}} = \vec{c} + \begin{pmatrix} \epsilon_{x_1} \\ \epsilon_{y_1} \\ \epsilon_{\Theta_1} \\ \epsilon_{\phi} \\ \epsilon_{\Delta\Theta_{12}} \end{pmatrix} \quad \vec{c}_\epsilon \in C \tag{3.5}$$

$\epsilon_{\vec{c}}$ beschreibt den additiven Fehlerterm des fehlerbehafteten Fahrzeugzustands $\vec{c}_\epsilon$.

Für den Steuervektor und die Fahrzeugbeschreibung gilt analog

$$\vec{v}_\epsilon = \vec{v} + \epsilon_{\vec{v}} = \vec{v} + \begin{pmatrix} \epsilon_{v_1} \\ \epsilon_{\phi_v} \end{pmatrix} \quad \vec{v}_\epsilon \in V \tag{3.6}$$

$$\vec{p}_\epsilon = \vec{p} + \epsilon_{\vec{p}} = \vec{p} + \begin{pmatrix} \epsilon_{L_1} \\ \epsilon_{L_2} \\ \epsilon_{M_1} \end{pmatrix} \quad \vec{p}_\epsilon \in P \tag{3.7}$$

Es gilt nun den Folgefehler zu bestimmen, der sich ergibt, wenn man fehlerbehaftete Eingabeparameter in die Zustandsübergangsfunktion f übergibt. Die Differenz zwischen fehlerbehaftetem und fehlerfreiem Folgezustand ist

$$\epsilon_{total} = f(\vec{c}_\epsilon, \vec{v}_\epsilon, \vec{p}_\epsilon) - f(\vec{c}, \vec{v}, \vec{p}) = f(\vec{c} + \epsilon_{\vec{c}}, \vec{v} + \epsilon_{\vec{v}}, \vec{p} + \epsilon_{\delta\vec{p}}) - f(\vec{c}, \vec{v}, \vec{p}) \tag{3.8}$$

Dies ist der Fehler, der sich pro Iterationsschritt akkumuliert. Ausgeschrieben würden sich die Fehler ϵ_i in den trigonometrischen Funktionen in f wiederfinden. Über Gleichung 2.25 ergibt sich für eine Approximation des Gesamtfehlers

$$\epsilon_{total} \approx J_c\epsilon_{\vec{c}} + J_v\epsilon_{\vec{v}} + J_p\epsilon_{\vec{p}} \tag{3.9}$$

mit den folgenden Jacobi-Matrizen:

$$J_c = \begin{pmatrix} 0 & 0 & -v_1 \sin\Theta_1 & 0 & 0 \\ 0 & 0 & v_1 \cos\Theta_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{L_2}(v_1 \cos\Delta\Theta_{12} + M_1 \sin\Delta\Theta_{12}\dot{\Theta}_1) & 0 & -\frac{1}{L_2}(v_1 \cos\Delta\Theta_{12} + M_1 \sin\Delta\Theta_{12}\dot{\Theta}_1) \end{pmatrix} \tag{3.10}$$

$$J_v = \begin{pmatrix} \cos\Theta_1 & 0 \\ \sin\Theta_1 & 0 \\ \frac{\tan\phi}{L_1} & v_1 \frac{1}{L_1 \cos^2\phi} \\ 0 & 1 \\ \frac{\tan\phi}{L_1} - \frac{\sin\Delta\Theta_{12}}{L_2} + \frac{M_1 \cos\Delta\Theta_{12}\tan\phi}{L_1 L_2} & v_1 \frac{1}{L_1 \cos^2\phi}(1 + \frac{M_1 \cos\Delta\Theta_{12}}{L_2}) \end{pmatrix} \tag{3.11}$$

$$J_p = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\frac{1}{L_1}\dot{\Theta}_1 & 0 & 0 \\ 0 & 0 & 0 \\ -\frac{1}{L_1}\dot{\Theta}_1(1 + \frac{M_1 \cos\Delta\Theta_{12}}{L_2}) & \frac{1}{L_2^2}(v_1 \sin\Delta\Theta_{12} - M_1 \cos\Delta\Theta_{12}\dot{\Theta}_1) & \frac{1}{L_2}\cos\Delta\Theta_{12}\dot{\Theta}_1 \end{pmatrix} \tag{3.12}$$

Eingesetzt in 3.9 lassen sich die Fehlerterme für jeden einzelnen Parameter des Fahrzeugzustands aufstellen. Interessant ist im Kontext der Assistenz jedoch vor allem der Fehler im Einknickwinkel und im Lenkwinkel, von denen die berechneten Assistenz-Trajektorien abhängen:

$$\begin{aligned} \epsilon_{total,\Delta\Theta_{12}} = & -\epsilon_{\Delta\Theta_{12}} \frac{1}{L_2}(v_1 \cos\Delta\Theta_{12} + M_1 \sin\Delta\Theta_{12}\dot{\Theta}_1) \\ & + \epsilon_{\phi_v} v_1 \frac{1}{L_1 \cos^2\phi}(1 + \frac{M_1 \cos\Delta\Theta_{12}}{L_2}) \\ & - \epsilon_{L_1} \frac{1}{L_1}\dot{\Theta}_1(1 + \frac{M_1 \cos\Delta\Theta_{12}}{L_2}) \\ & + \epsilon_{L_2} \frac{1}{L_2^2}(v_1 \sin\Delta\Theta_{12} - M_1 \cos\Delta\Theta_{12}\dot{\Theta}_1) \\ & + \epsilon_{M_1} \frac{1}{L_2}\cos\Delta\Theta_{12}\dot{\Theta}_1 \end{aligned} \tag{3.13}$$

$$\epsilon_{total,\phi} = \epsilon_{\phi_v} \tag{3.14}$$

Es ist zu erkennen, dass der Fehler im Einknickwinkel komplexen Zusammenhängen

folgt, währen der Fehler im Lenkwinkel trivialerweise genau dem Eingabefehler entspricht. Der Fokus liegt in der Folge daher vor allem auf dem Einknickwinkel.

Abbildung 3.2 zeigt beispielhaft, wie sich die verschiedenen Fehler gegenseitig beeinflussen können. Betrachtet wurden jeweils die Fehleranteile des Ausgabefehlers, den die einzelnen Eingabefehler verursachen, wenn der stabile Lenkwinkel anliegt. So lässt sich betrachten, wie sich der Fehler auf die Gespann-Trajektorie auswirkt.

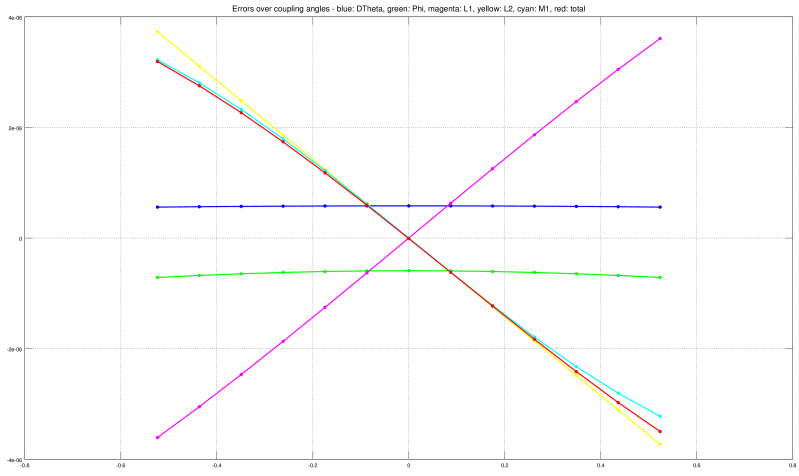


Abbildung 3.2: Vergleich des Einflusses von Fehlern in der Fahrzeugabmessung. Zu sehen sind die Fehler in *L1*(magenta), *L2*(gelb) und*M1*(cyan). Zum Vergleich werden zudem Fehler in ϕ(grün) sowie $\Delta\Theta_{12}$(blau) betrachtet. Der Gesamtfehler ist in rot dargestellt. Das zugrunde liegende Fahrzeugmodell findet sich in Tabelle 4.1. Als Lenkwinkel wurde der stabile Lenkwinkel gewählt. Alle Eingabefehler besitzen den Wert 0.1.

Zu beobachten ist, dass sich die Fehlerterme für *L1* und *L2* sowie *L2* und *M1* nahezu gegenseitig auslöschen. Dasselbe Phänomen ist für Einknick- und Lenkwinkelfehler zu beobachten. Dies ist auf eine Eigenheit des betrachteten Fahrzeugmodells zurückzuführen. Für ein anderes Fahrzeugmodell lassen sich dementsprechend andere Zusammenhänge zwischen den Eingabefehlern bestimmen.

Einige Abschwächungs- und Verstärkungserscheinungen sind in den Abbildungen 3.3 bis 3.6 zu sehen. Betrachtet wird dabei vor allem der Zusammenhang zwischen ϵ_{L2}

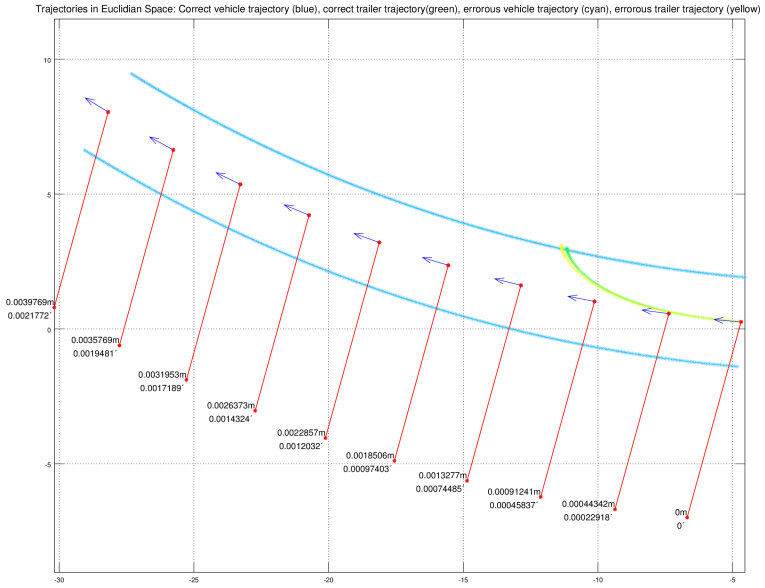


Abbildung 3.3: Fehlerbeeinflussung fuer $\Delta\Theta_{12} = 5°$, $\epsilon_{L2} = 0.1$, $\epsilon_{M1} = 0.1$.

und ϵ_{M1}. Es werden jeweils die korrekten (blau, grün) sowie die fehlerbehafteten Trajektorien (cyan, gelb) für Gespann und Anhänger gezeigt. Die roten Punkte zeigen die Abweichung der beiden Trajektorien voneinander nach äquidistanten Distanzschritten an. Die Vektorpfeile wiederum geben die Abweichung der Ausrichtungen an.

Es ist zu erkennen, dass die resultierenden Trajektorien bei einer Abschwächung nahezu identisch sind in ihren Endpunkten, während bei einer Verstärkung deutliche Abweichungen von 25 bzw. 50cm auftreten. Je größer der Einknickwinkel, desto größer ist die Verstärkung. Auch kann davon ausgegangen werden, dass die Abweichung welche vom Fahrer angenommen wird mit der Länge der Hilfs-Trajektorien steigt.

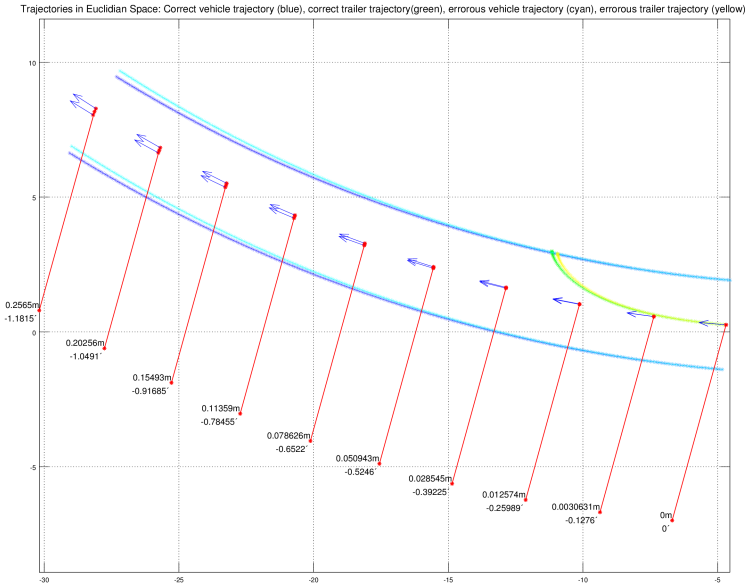


Abbildung 3.4: Fehlerbeeinflussung fuer $\Delta\Theta_{12} = 5°$, $\epsilon_{L2} = -0.1$, $\epsilon_{M1} = 0.1$.

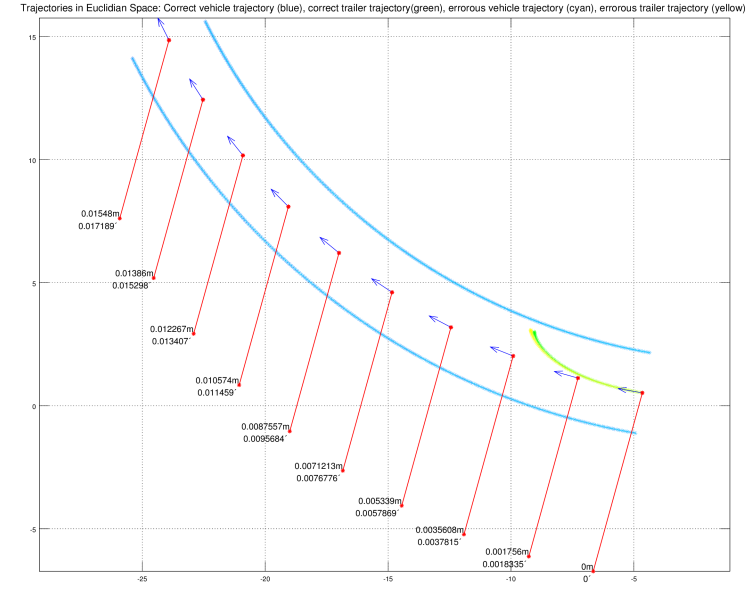


Abbildung 3.5: Fehlerbeeinflussung fuer $\Delta\Theta_{12} = 10°$, $\epsilon_{L2} = 0.1$, $\epsilon_{M1} = 0.1$.

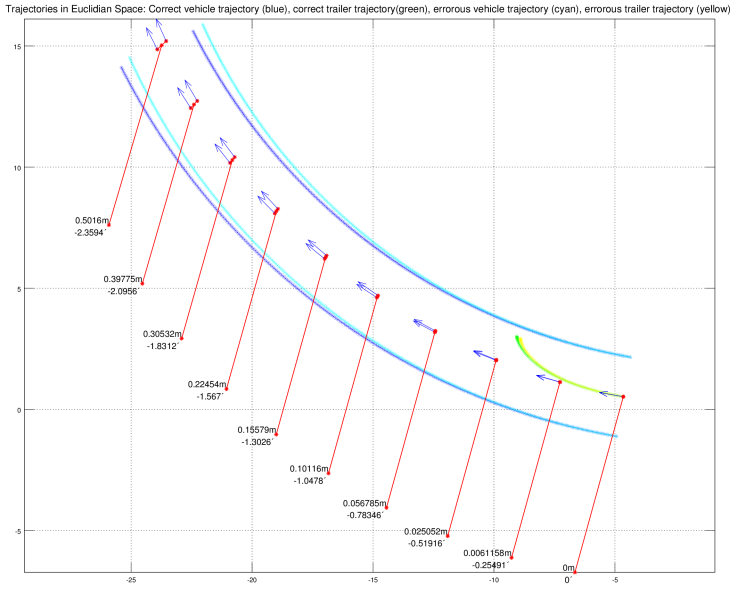


Abbildung 3.6: Fehlerbeeinflussung fuer $\Delta\Theta_{12} = 10°$, $\epsilon_{L2} = -0.1$, $\epsilon_{M1} = 0.1$.

Fehler in den Sensordaten

Die Unscented Transform hilft zu verstehen, wie sich probabilistische Fehler in den Eingabeparametern fortpflanzen. Dadurch lässt sich unter anderem analysieren, ob die resultierende rauschende Assistenz-Trajektorie einen Bias zu einer Seite besitzt oder nicht, was das Verhalten des Fahrers wiederum beeinflussen würde.

In Kapitel 3.2.2 wurden die Fehlerterme für die Fahrzeugkonfiguration $\epsilon_{\vec{c}}$ sowie den Steuervektor $\epsilon_{\vec{v}}$ vorgestellt. Im Folgenden wird exemplarisch der Fehler im Einknickwinkel des Fahrzeugs näher betrachtet, da dies ein Wert ist, welcher im realen Fahrzeug nur über Sensorik zur Verfügung steht, daher normalverteiltem Rauschen unterliegt.

Der angepasste Fehlerterm hat die folgende Form:

$$\epsilon_{\vec{c}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \epsilon_{\Delta\Theta_{12}} \end{pmatrix} \tag{3.15}$$

wobei $\epsilon_{\Delta\Theta_{12}}$ eine normalverteilte Zufallsvariable ist mit $\epsilon_{\Delta\Theta_{12}} \sim \mathcal{N}(0, \sigma_{\epsilon_{\Delta\Theta_{12}}})$. Alle anderen Fehler werden ignoriert und 0 gesetzt, was äquivalent ist zu einer normalverteilten Zufallsvariablen mit Mittelwert 0 und Standardabweichung 0. Da alle Zeilen von $\epsilon_{\vec{c}}$ normalverteilt sind, ist auch $\epsilon_{\vec{c}}$ normalverteilt. Es gilt:

$$\epsilon_{\vec{c}} \sim \mathcal{N}(\mu_{\epsilon_{\Delta\Theta_{12}}}, \Sigma_{\epsilon_{\Delta\Theta_{12}}}) \tag{3.16}$$

mit

$$\mu_{\epsilon_{\Delta\Theta_{12}}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \Sigma_{\epsilon_{\Delta\Theta_{12}}} = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma^2_{\Delta\Theta_{12}} \end{pmatrix} \tag{3.17}$$

Der Erwartungswert ist 0, da angenommen wird, dass sich der Fehler bei wiederholter Messung im Mittel aufhebt. Überträgt man die Wahrscheinlichkeitsverteilung der Fehlerterme auf Konfigurationsvektor, so ändert sich lediglich der Mittelwert, welcher nun gleich dem fehlerfreien Vektor ist:

$$\mu_c = \vec{c} \tag{3.18}$$

Mit der auf diese Weise aufgestellten Wahrscheinlichkeitsverteilung lässt sich die Unscented Transform für Fehler im Einknickwinkel durchführen. Den erste Schritt bildet

die deterministische Wahl der Sigma-Punkte.

Für die Konfiguration werden 11 Sigma-Punkte benötigt, da es sich um einen 5-dimensionalen Vektor handelt:

$$
\begin{aligned}
\vec{c}^{(0)} &= \mu_c = \vec{c} \\
\vec{c}^{(1-4)} &= \mu_c + \sqrt{n+\lambda}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T = \vec{c} \\
\vec{c}^{(5)} &= \mu_c + \sqrt{n+\lambda}\begin{pmatrix} 0 & 0 & 0 & 0 & \sigma_{\Delta\Theta_{12}} \end{pmatrix}^T = \vec{c} + 10^{-3}\sqrt{5}\begin{pmatrix} 0 & 0 & 0 & 0 & \sigma_{\Delta\Theta_{12}} \end{pmatrix}^T \\
\vec{c}^{(6-9)} &= \mu_c - \sqrt{n+\lambda}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix} = \vec{c} \\
\vec{c}^{(10)} &= \mu_c - \sqrt{n+\lambda}\begin{pmatrix} 0 & 0 & 0 & 0 & \sigma_{\Delta\Theta_{12}} \end{pmatrix}^T = \vec{c} - 10^{-3}\sqrt{5}\begin{pmatrix} 0 & 0 & 0 & 0 & \sigma_{\Delta\Theta_{12}} \end{pmatrix}^T
\end{aligned}
$$

Wie man sieht entsprechen 9 der 11 Sigma-Punkte genau dem Mittelwert $\vec{c}^{(0)}$. Dies rührt daher, dass nur die Varianz des Einknickwinkels für die folgenden Berechnungen interessant ist und die Kovarianzmatrix folglich nur einen Wert auf der Diagonalen enthält. Dieser Umstand führt dazu, dass lediglich drei Transformationen berechnet werden müssen. Der Kontrollvektor $\vec{v}$ sei für alle Berechnungen konstant. Dann gilt

$$
z_{0-4,6-9} = f(\vec{c}^{(0-4,6-9)}) = \begin{pmatrix} x_1 + v_1 \cos\Theta_1 \\ y_1 + v_1 \sin\Theta_1 \\ \Theta_1 + v_1 \frac{\tan\phi}{L_1} \\ \phi_v \\ \Delta\Theta_{12} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin\Delta\Theta_{12}}{L_2} + \frac{M_1 \cos\Delta\Theta_{12}\tan\phi}{L_1 L_2}) \end{pmatrix} \tag{3.19}
$$

für die ersten 9 transformierten Sigma-Punkte sowie

$$z_5 = f(\vec{c}^{(5)}) = \begin{pmatrix} x_1 + v_1 \cos\Theta_1 \\ y_1 + v_1 \sin\Theta_1 \\ \Theta_1 + v_1 \frac{\tan\phi}{L_1} \\ \phi_v \\ \Delta\Theta_{12} + 10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin(\Delta\Theta_{12}+10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})}{L_2} \\ + \frac{M_1 \cos(\Delta\Theta_{12}+10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})\tan\phi}{L_1 L_2}) \end{pmatrix} \tag{3.20}$$

und

$$z_{10} = f(\vec{c}^{(12)}) = \begin{pmatrix} x_1 + v_1 \cos\Theta_1 \\ y_1 + v_1 \sin\Theta_1 \\ \Theta_1 + v_1 \frac{\tan\phi}{L_1} \\ \phi_v \\ \Delta\Theta_{12} - 10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin(\Delta\Theta_{12}-10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})}{L_2} \\ + \frac{M_1 \cos(\Delta\Theta_{12}-10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})\tan\phi}{L_1 L_2}) \end{pmatrix} \tag{3.21}$$

für die restlichen beiden Sigma-Punkte.

Der Mittelwert der transformierten Werte $\bar{z}$ berechnet sich nach Abschnitt 2.3.3 aus

$$\bar{z} = \sum_{i=0}^{14} w_i z_i \tag{3.22}$$

Die w_i bezeichnen die Gewichte der einzelnen Sigma-Punkte mit

$$w_0 = \frac{10^{-6} * 5 - 5}{10^{-6} * 5} = \frac{10^{-6} - 1}{10^{-6}} = 1 - \frac{1}{10^{-6}} \tag{3.23}$$

$$w_{1-10} = \frac{1}{2 * 10^{-6} * 5} = \frac{1}{10^{-5}} \tag{3.24}$$

Da die ersten vier Zeilen aller z_i nicht fehlerbehaftet und somit identisch sind, ist

die Mittelwertbestimmung dieser Zeilen trivial. Es bleibt also den Mittelwert des Einknickwinkels $\Delta\bar{\Theta}_{12}$ aller transformierten Sigma-Punkte zu bestimmen.

$$\begin{aligned}
\Delta\bar{\Theta}_{12} &= \sum_{i=0}^{14} w_i \Delta\Theta_{12,i} \\
&= (1 - \frac{1}{10^{-6}}) * (\Delta\Theta_{12} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin\Delta\Theta_{12}}{L_2} + \frac{M_1\cos\Delta\Theta_{12}\tan\phi}{L_1L_2})) \\
&+ 8 * \frac{1}{10^{-5}} * (\Delta\Theta_{12} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin\Delta\Theta_{12}}{L_2} + \frac{M_1\cos\Delta\Theta_{12}\tan\phi}{L_1L_2})) \\
&+ \frac{1}{10^{-5}} * (\Delta\Theta_{12} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin(\Delta\Theta_{12} + 10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})}{L_2} \\
&\qquad + \frac{M_1\cos(\Delta\Theta_{12} + 10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})\tan\phi}{L_1L_2})) \\
&+ \frac{1}{10^{-5}} * (\Delta\Theta_{12} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin(\Delta\Theta_{12} - 10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})}{L_2} \\
&\qquad + \frac{M_1\cos(\Delta\Theta_{12} - 10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})\tan\phi}{L_1L_2}))
\end{aligned}$$

Die ersten beiden Summanden entsprechen gerade den Transformationen der fehlerfreien Sigma-Punkte. Das Problem der restlichen beiden Summanden ist, dass sich die Fehlerterme innerhalb der trigonometrischen Funktionen befinden. Um den allgemeinen Mittelwert zu bestimmen, muss eine Möglichkeit gefunden werden, die Fehlerterme aus diesen Funktionen herauszuziehen. Dazu werden die folgenden trigonometrischen Identitäten betrachtet [Forster, 1983].

$$\begin{aligned}
\sin(x+y) + \sin(x-y) &= \sin x\cos y + \cos x\sin y + \sin x\cos y - \cos x\sin y \\
&= 2\sin x\cos y \qquad (3.25) \\
\cos(x+y) + \cos(x-y) &= \cos x\cos y - \sin x\sin y + \cos x\cos y + \sin x\sin y \\
&= 2\cos x\cos y \qquad (3.26)
\end{aligned}$$

Löst man die letzten beiden Summanden mithilfe dieser Abhängigkeiten auf, ergibt

sich:

$$
\begin{aligned}
\Delta\bar{\Theta}_{12} &= \sum_{i=0}^{14} w_i \Delta\Theta_{12,i} \\
&= (1 - \frac{1}{10^{-6}} + \frac{8}{10^{-5}}) * (\Delta\Theta_{12} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin\Delta\Theta_{12}}{L_2} + \frac{M_1\cos\Delta\Theta_{12}\tan\phi}{L_1 L_2})) \\
&+ \frac{1}{10^{-5}} * (2\Delta\Theta_{12} + v_1(2\frac{\tan\phi}{L_1} \\
&- \frac{1}{L_2}(\sin(\Delta\Theta_{12} + 10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}}) + \sin(\Delta\Theta_{12} - 10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})) \\
&+ \frac{M_1\tan\phi}{L_1 L_2}(\cos(\Delta\Theta_{12} + 10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}}) + \cos(\Delta\Theta_{12} - 10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})) \\
&= (1 - \frac{1}{10^{-6}} + \frac{8}{10^{-5}}) * (\Delta\Theta_{12} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin\Delta\Theta_{12}}{L_2} + \frac{M_1\cos\Delta\Theta_{12}\tan\phi}{L_1 L_2})) \\
&+ \frac{1}{10^{-5}} * (2\Delta\Theta_{12} + v_1(2\frac{\tan\phi}{L_1} - 2\frac{1}{L_2}\sin(\Delta\Theta_{12})\cos(10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}}) \\
&+ 2\frac{M_1\tan\phi}{L_1 L_2}\cos(\Delta\Theta_{12})\cos(10^{-3}\sqrt{5}\sigma_{\Delta\Theta_{12}})))
\end{aligned}
$$

Über die Taylor-Expansion des Kosinus

$$\cos(x) = \sum_{n=0}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!} \tag{3.27}$$

lassen sich die Kosinus-Terme weiter vereinfachen. Für Winkel kleiner als $0.664rad$ ($\approx 38°$) beträgt der Fehler für eine Approximation zweiter Ordnung

$$\cos(x) = 1 - \frac{x^2}{2} \tag{3.28}$$

weniger als 1% [Boas, 2006], wobei x dem Winkel in Radiant entspricht. Man erhält

schließlich

$$\begin{aligned}\Delta\bar{\Theta}_{12} &= \sum_{i=0}^{14} w_i \Delta\Theta_{12,i} \\ &= (1 - \frac{1}{10^{-6}} + \frac{8}{10^{-5}}) * (\Delta\Theta_{12} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin\Delta\Theta_{12}}{L_2} + \frac{M_1\cos\Delta\Theta_{12}\tan\phi}{L_1L_2})) \\ &+ \frac{1}{10^{-5}} * (2\Delta\Theta_{12} + v_1(2\frac{\tan\phi}{L_1} - 2(\frac{1}{L_2}\sin(\Delta\Theta_{12}) - \frac{10^{-6} * 5 * \sigma^2_{\Delta\Theta_{12}}}{2L_2}\sin(\Delta\Theta_{12}) \\ &+ 2(\frac{M_1\tan\phi}{L_1L_2}\cos(\Delta\Theta_{12}) - \frac{10^{-6} * 5 * \sigma^2_{\Delta\Theta_{12}}}{2}\frac{M_1\tan\phi}{L_1L_2}\cos(\Delta\Theta_{12}))))\end{aligned}$$

Über weitere Vereinfachungen ergibt sich schließlich

$$\begin{aligned}\Delta\bar{\Theta}_{12} &= \Delta\Theta_{12} + v_1(\frac{\tan\phi}{L_1} - \frac{\sin\Delta\Theta_{12}}{L_2} + \frac{M_1\cos\Delta\Theta_{12}\tan\phi}{L_1L_2}) \\ &+ \frac{\sigma^2_{\Delta\Theta_{12}}}{2} * v_1(-\frac{\sin(\Delta\Theta_{12})}{L_2} + \frac{M_1\cos(\Delta\Theta_{12})\tan\phi}{L_1L_2})\end{aligned} \tag{3.29}$$

Der Mittelwert des transformierten Einknickwinkel unterliegt also einem Fehler, der proportional zur Varianz der Wahrscheinlichkeitsverteilung ist. Wie groß der Fehler letztlich ist, hängt vom Einknickwinkel des letzten Zeitschritts sowie dem aktuellen Lenkwinkel ab.

Wie zuvor beschrieben, sind Position und Ausrichtung des Mittelwerts der Transformation identisch mit den entsprechenden Werten des transformierten Mittelwerts. Die einzige Abweichung besteht folglich im Einknickwinkel. Betrachtet werden nun die zu jedem Einknickwinkel gehörigen Gespann-Trajektorien. Aufgrund der hier vorgestellten Ergebnisse liegt der Schluss nahe, dass die Trajektorie des Einknickwinkel-Mittelwerts nicht mit dem Mittelwert der Trajektorien der Sigma-Punkte übereinstimmt.

Angenommen die Messung des Einknickwinkels unterliegt normalverteiltem, mittelwertfreiem, additivem Rauschen mit einer Standardabweichung von σ. Abbildung 3.7 zeigt Assistenz-Trajektorien, welche mit verschiedenen Einknickwinkeln berechnet wurden.

Die blauen Trajektorien beschreiben die Gespann-Trajektorien, welche mit dem korrekten Einknickwinkel berechnet würden. Die Gespann-Trajektorien in magenta und cyan wurden jeweils mit einem Einknickwinkel berechnet, der einem Fehler von $+2\sigma$ und -2σ unterliegt. In diesem Fall entspricht $\sigma = 0.25°$. Diese beiden Gespann-Trajektorien bilden die äußere Schranke der 2σ-Umgebung, in welcher 95% aller Trajektorien liegen werden.

In Abbildung 3.8 ist die zugehörige Mittelwert-Trajektorie zu dieser Standardabweichung zu sehen, welche mithilfe der Unscented Transform über die zu den Sigma-Punkten gehörigen Trajektorien berechnet wurde. Wie man sieht besteht zwar eine Abweichung, jedoch ist diese minimal.

Analog zu diesem Beispiel wurden weitere Parameter-Kombinationen getestet. Abbildungen 3.9 und 3.10 zeigen wiederum die 2σ-Umgebung, diesmal für $\sigma = 0.5°$ sowie die entsprechende Unscented Trajektorie. Mit steigender Krümmung der Assistenz-Trajektorien vergrößern sich zwar Positions- und Ausrichtungsunterschiede, jedoch sind diese für das menschliche Auge kaum wahrnehmbar.

Folglich unterliegt der Nutzer keinem merklichen Bias zu einer Seite, was sein Fahrverhalten nicht beeinflussen sollte.

3.2.3 Fehler in der Darstellung der Trajektorien

Diese Kategorie umfasst Fehler in der Darstellung der Trajektorien auf dem Assistenz-Display. Nachdem die Trajektorien berechnet wurden, müssen sie perspektivisch korrekt im Kamerabild der Rückfahrkamera angezeigt werden. Dazu werden sie anhand von Referenzpunkten, wie der Position der Rückfahrkamera, zwischen verschiedenen Koordinatensystemen transformiert.

Fehler können zwei in der Folge zwei Ursachen haben. Entweder wird die richtige Trajektorie ins falsche Koordinatensystem transformiert oder die falsche Trajektorie ins richtige Koordinatensystem. Die Terme *richtig* und *falsch* in der Beschreibung der

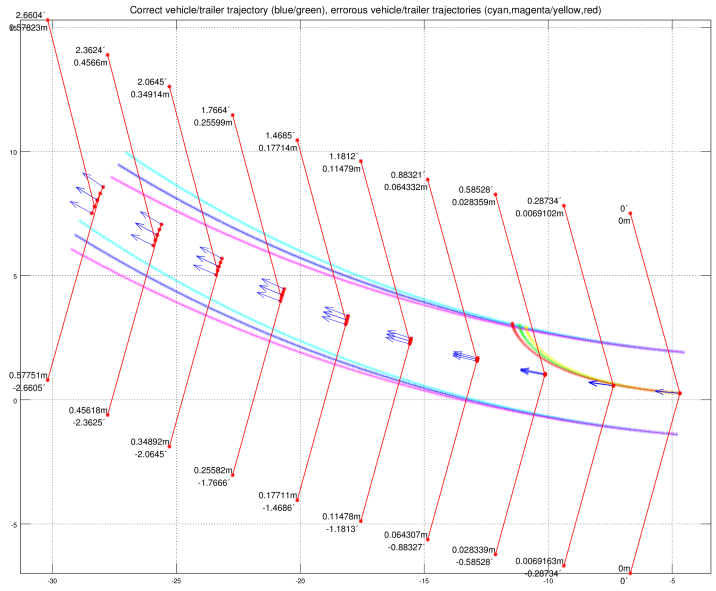


Abbildung 3.7: 2-Sigma Umgebung fuer ein additives Sensorrauschen mit Standardabweichung 0.25 und einem Einknickwinkel von 5 Grad.

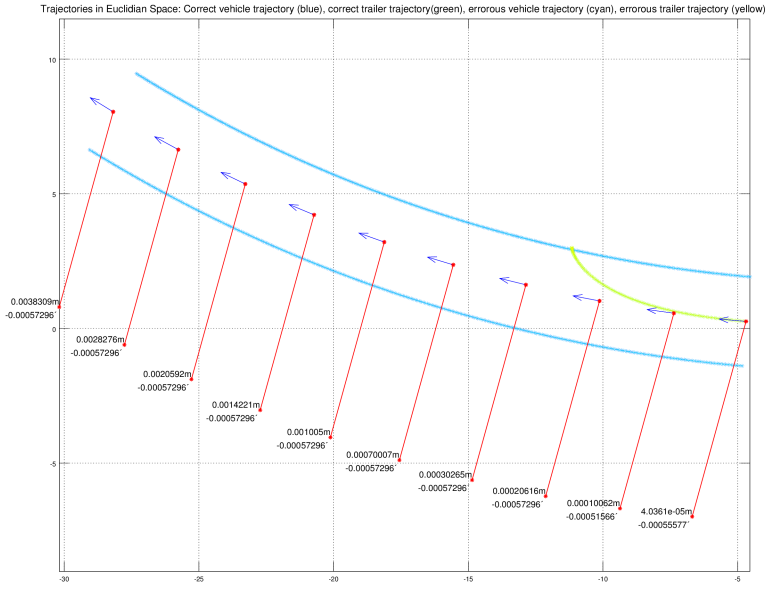


Abbildung 3.8: Durchschnittliche Trajektorie fuer ein additives Sensorrauschen mit Standardabweichung 0.25 und einem Einknickwinkel von 5 Grad.

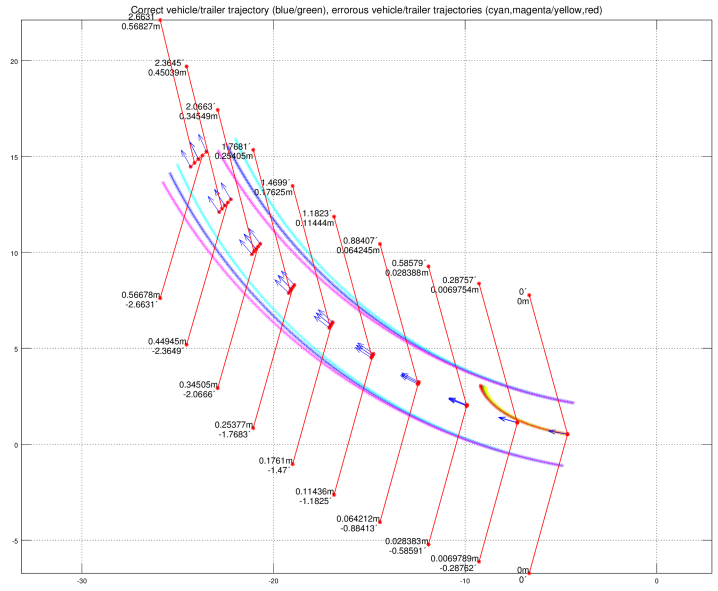


Abbildung 3.9: 2-Sigma Umgebung fuer ein additives Sensorrauschen mit Standardabweichung 0.25 und einem Einknickwinkel von 10 Grad.

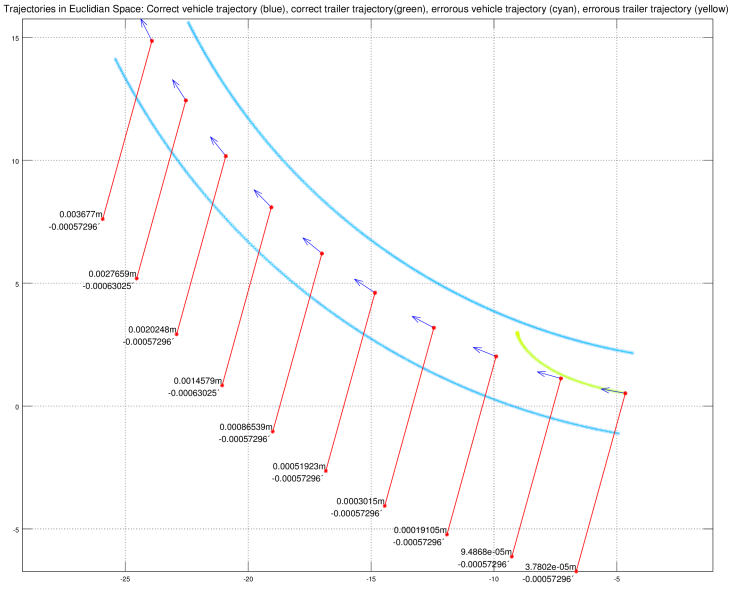


Abbildung 3.10: Durchschnittliche Trajektorie fuer ein additives Sensorrauschen mit Standardabweichung 0.25 und einem Einknickwinkel von 10 Grad.

Trajektorien bezeichnen hier nicht die Korrektheit der Fahrzeugbewegung, sondern die korrekte Zuordnung zwischen Trajektorie und Koordinatensystem.

Ein lokales Koordinatensystem, welches sich in Relation zum Fahrzeug bewegt, unterliegt zu verschiedenen Zeitpunkten unterschiedlichen Transformationen relativ zum Weltkoordinatensystem. Das aktuelle Referenz-Koordinatensystem kann dabei korrekt parametrisiert sein über Achsenrichtung und Ursprung, aber die anzuzeigende Trajektorie gehört zu einem veralteten Koordinatensystem. Dies entspricht dem Fehler durch Verzögerung.

Das lokale Koordinatensystem zum aktuellen Zeitpunkt kann allerdings auch falsch parametrisiert sein, etwa durch die Annahme eines verschobenen Ursprungs. Obwohl die Trajektorie zum aktuellen Koordinatensystem gehört, wird sie also dennoch falsch dargestellt relativ zur Grundwahrheit. Dies entspricht dem Fehler in der Kamerapose.
Ein weiterer, besonderer Fehlertyp betrifft nicht die Berechnung und Transformation der Assistenz-Trajektorien, sondern die Synchronität der Trajektorien mit dem angezeigten Kamerabild. Ist das Kamerabild verzögert, so ändert sich die Referenz-Umgebung, in der die Trajektorien eingebettet sind. Relativ zur Umgebung werden die Trajektorien daher falsch angezeigt, was zu Fehlern in der Navigation führen kann.

Fehler in der Kamerapose

Während die Trajektorien im $\mathcal{R}^2$ berechnet werden, wird die Pose der Rückfahrkamera im $\mathcal{R}^3$ definiert. Das Koordinatensystem ist ein Linke-Hand-System, dessen Ursprung in der Hinterachse des Anhängers liegt. Blickt man in Richtung des Hecks, zeigt die x-Achse nach rechts parallel zur Hinterachse, die y-Achse nach oben und die negative z-Achse in Blickrichtung.

Die Position der Kamera wird als Translation und die Ausrichtung als Rotation um

die drei Koordinatenachsen repräsentiert. Wird die Kamerapose vorab als Parameter zur Verfügung gestellt, kann der Ursprung des Koordinatensystems und damit die Anhängerposition rekonstruiert werden. Die so berechnete Anhängerposition dient als Startpunkt der Hilfs-Trajektorien und beeinflusst so, wo die Trajektorien im Assistenz-Display angezeigt werden.

Ist die Kameraposition nicht genau vermessen oder weicht die Ausrichtung um einige Grad vom tatsächlichen Wert ab, dann werden die Trajektorien an der falschen Stelle dargestellt. Wie genau sich der Fehler in der Kamerapose auf die Startposition der Trajektorien überträgt, wird im Folgenden näher erläutert. Zunächst werden dazu die Parameter der Transformation zwischen korrektem und fehlerbehaftetem Koordinatensystem bestimmt, danach wird erläutert, wie eine Trajektorie von einem Koordinatensystem in das andere transformiert werden kann.

Der Fehler durch falsche Positionierung sowie Ausrichtung besitzt je drei Freiheitsgrade in Form von Translationen in Richtung jeder der drei Koordinatenachsen bzw. Rotationen um ebenjene. Die Rotation um die z-Achse wird im folgenden ausgelassen, da diese eine merkliche Verzerrung der Perspektive verursacht und dem Fahrer zusätzliche Probleme bereiten kann, die über eine einfache Verschiebung oder Rotation der Trajektorien hinaus gehen.

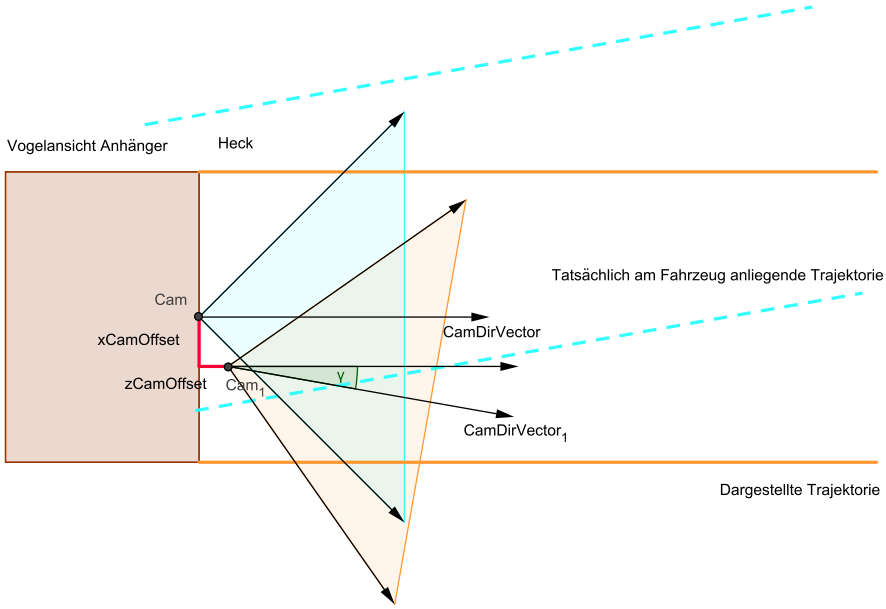


Abbildung 3.11: Anhänger und Kamerafrustum aus der Vogelperspektive.

Abbildung 3.11 zeigt die Ansicht des Anhängers aus der Vogelperspektive bzw. in Richtung der negativen y-Achse. Der Punkt **CamPos** beschreibt die x- und z- Koordinaten der realen Position der Rückfahrkamera sowie die Blickrichtung **CamDirVec** der Kamera. Der Punkt $\mathbf{CamPos}_1$ wiederum beschreibt die angenommene, fehlerhafte Position, welche um **xCamOffset** und **zCamOffset** von der realen Position abweicht. Der Vektor $\mathbf{CamDirVec}_1$ beschreibt die um γ rotierte Blickrichtung der Kamera um die y-Achse. Diese Positionen und Ausrichtungen beschreiben jeweils ein zweidimensionales Linke-Hand-Kamerakoordinatensystem, wobei die Blickrichtung der y-Achse entspricht. Das hellblaue Dreieck repräsentiert das zu **CamPos** gehörige Kamerafrustum von oben gesehen, das orange das Frustum von $\mathbf{CamPos}_1$.

Bei der Berechnung der Assistenz-Trajektorien wird von $\mathbf{CamPos}_1$ als Kameraposition ausgegangen. Der Startpunkt der berechneten Trajektorien richtet sich also nach der von $\mathbf{CamPos}_1$ abhängigen Anhängerposition, welche als Ursprung des Anhängerkoordinatensystem dient, in welches die Trajektorie transformiert werden muss.

Im Kamerabild angezeigt wird schließlich die orange Trajektorie. Ist die Kamera nun nicht an der erwarteten Position, so ändert sich lediglich, wo die Trajektorien in der Umgebung angezeigt werden, die das Kamerabild beschreibt. Die tatsächlich am Fahrzeug anliegende Trajektorie ist in Abbildung 3.11 hellblau dargestellt.

Würde die angenommene Kamerapose genau mit der realen Pose übereinstimmen, so würden sich die orange und die hellblaue Trajektorie exakt überlagern.

Besteht eine Abweichung, so muss die orange Trajektorie vom Kamerakoordinatensystem abhängig von $CamPos_1$ und $\mathbf{CamDirVec}_1$ ins Koordinatensystem abhängig von **CamPos** und **CamDirVec** transformiert werden. Dies entspricht einer Rotation der Trajektorien um $-\gamma$ und einer anschließenden Verschiebung um $\begin{pmatrix} -xCamOffset & -zCamOffset \end{pmatrix}^T$.

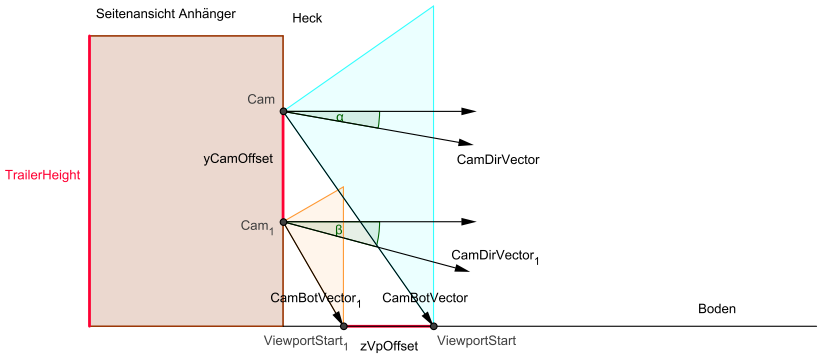


Abbildung 3.12: Seitenansicht von Anhänger und Kamerafrustum.

Abbildung 3.12 zeigt die Seitenansicht eines Anhängers in Richtung der negativen x-Achse. $CamPos$ und $CamPos_1$ werden nun in y- und z-Richtung angezeigt. Die Winkel α und β bezeichnen die jeweilige Rotation der Kameras um die x-Achse. Die Abweichung in y-Richtung zwischen dem beiden Kamerapositionen wird mit $yCamOffset$ bezeichnet. Der Abstand $zVpOffset$ repräsentiert die Abweichung zwischen den jeweiligen Punkten auf dem Boden, an denen der Viewport der betrachteten Kamera beginnt.

Der Abstand berechnet sich folgendermaßen: Sei ϕ der Öffnungswinkel der beiden Kameras. Seien zudem Cam_yund $Cam_{1,y}$ die y-Anteile der Kamerapositionen und $ViewportStart_{cdv}$ und $ViewportStart_{1,cdv}$ die Anteile der Viewport-Punkte in Blickrichtung der Kameras. Dann ist

$$\begin{aligned} zVpOffset &= ViewportStart_{cdv} - ViewportStart_{1,cdv} \\ &= Cam_y * \tan(90^\circ - \alpha - \frac{\phi}{2}) - Cam_{1,y} * \tan(90^\circ - \beta - \frac{\phi}{2}) \end{aligned}$$

Da die Kamera wie zuvor beschrieben eine Rotation um die y-Achse besitzen kann, entspricht der zugehörige Translationsvektor dem negativen x-z-Anteil des Blickrichtungsvektors der Kamera mit Länge $zVpOffset$. Dieser Translationsvektor muss zusätzlich auf die Transformation zwischen den beiden Koordinatensystemen angewandt werden.

Wurden die benötigten Translationen und Rotationen bestimmt, so lässt sich eine Trajektorie wie folgt von einem Koordinatensystem in das andere überführen.

Gegeben seien Anhänger A und Anhänger B mit ihren lokalen Koordinatensystemen O^A und O^B. Der Ursprung von O^A liegt in der absoluten Position $\vec{x}_2^A$ von Anhänger A, die Koordinatenachsen entsprechen den um die Ausrichtung Θ_2^A gedrehten Achsen des Weltkoordinatensystems. Analog ist O^B über Position und Ausrichtung von Anhänger B definiert. Seien

$$\Delta\vec{x}_2^{AB} = \vec{x}_2^B - \vec{x}_2^A \tag{3.30}$$

$$\Delta\Theta_2^{AB} = \Theta_2^B - \Theta_2^A \tag{3.31}$$

die relative Position und Ausrichtung zwischen Anhänger A und B. Dann ist

$$T = \begin{pmatrix} rot^{AB} & trans^{AB} \end{pmatrix} \tag{3.32}$$

die relative Transformation zwischen den beiden Anhängern mit rot^{AB} als Rotati-

onsmatrix der relativen Ausrichtung $\Delta\Theta_2^{AB}$ und $trans^{AB}$ als Translationsvektor der relativen Position $\Delta\vec{x}_2^{AB}$ zwischen den Anhängern.

Seien in der Folge $Traj^A$ und $Traj^B$ zwei von der Form her identische Trajektorien, welche jedoch unterschiedliche Startausrichtungen und -positionen besitzen. $Traj^A$ besitzt dieselbe Startausrichtung und -position wie Anhänger A, $Traj^B$ analog von Trajektorie B. Dann berechnet sich $Traj^B$ folgendermaßen aus $Traj^A$:

$$Traj_j^B = rot^{AB} * (Traj_j^A - \vec{x}_2^A) + \vec{x}_2^A + trans^{AB} \quad j \in Size(Traj^A) \tag{3.33}$$

Um die Trajektorie A um die Anhängerposition A zu rotieren, muss die Anhängerposition zunächst in den globalen Ursprung verschoben werden. Nach der Rotation wird diese Verschiebung rückgängig gemacht und der Translationsvektor angewandt.

Diese Transformation wird für alle Punkte der Trajektorie ausgeführt. Der Index j stellt den j-ten Punkt vom Beginn der Trajektorie dar. Die transformierte Trajektorie B lässt sich nun problemlos mit der tatsächlich an Anhänger B anliegenden Trajektorie vergleichen.

Verzögerungen in der Akquirierung von Lenk- und Einknickwinkel

Unabhängig von der Validität der gemessenen Winkel muss die Zeitigkeit der Werte betrachtet werden. Wann wurde ein Winkel gemessen und wann erreicht er die Assistenzsoftware bzw. wann werden die zugehörigen Hilfslinien angezeigt? Die Differenz zwischen diesen beiden Zeitpunkten ist die Verzögerung, welche beim Datentransport in einem realen Fahrzeug auftritt.

Da sich das Fahrzeug im Zeitraum der Verzögerung üblicherweise stetig weiterbewegt, stimmt der aktuelle Einknick- bzw. Lenkwinkel nicht mehr mit den in der Assistenz in Verarbeitung befindlichen Winkeln überein. Die aktuell angezeigten Hilfslinien der Assistenz basieren also auf veralteten Daten und stellen die Trajektorien des Fahrzeugs zu einem früheren Zeitpunkt dar.

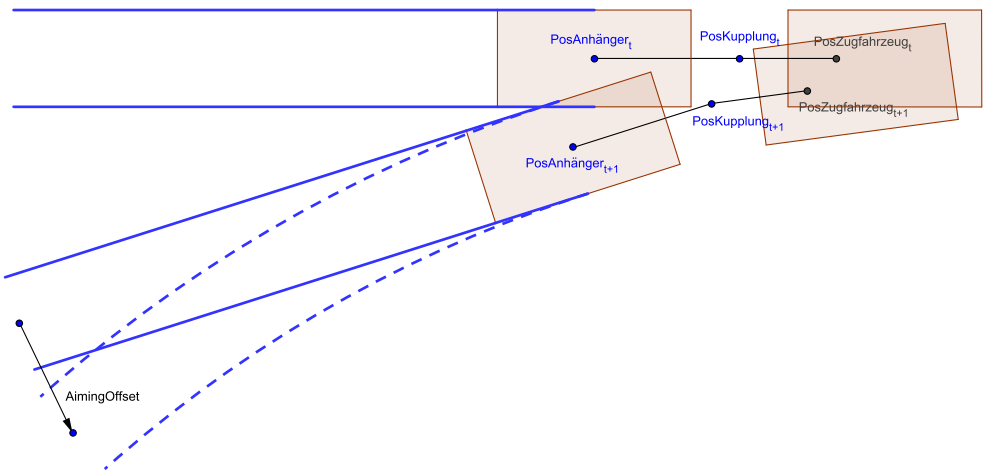


Abbildung 3.13: Der Fahrzeugzustand zu verschiedenen Zeitpunkten. Erst zum Zeitpunk t+1 wird aufgrund Verzögerungen die Trajektorie angezeigt, welche schon zum Zeitpunkt t anliegen sollte.

Abbildung 3.13 zeigt ein Fahrzeuggespann zu den verschiedenen Zeitpunkten t und t+1. Der Zustand t ist der Ausgangszustand des Fahrzeugs und der Zustand t+1 der Folgezustand, welcher nach einer Zeit angenommen wird, die äquivalent zur Verzögerung der Messwerte ist.

Zusätzlich eingeblendet sind die blauen Hilfstrajektorien für die stabile Fahrt des Gespanns. Die gestrichelten Linien entsprechen jeweils den Trajektorien, welche zu den Zeitpunkten t bzw. t+1 angezeigt werden sollten. Die durchgezogene Linie des unteren Fahrzeugs entspricht der Trajektorie, welche tatsächlich zum Zeitpunkt t+1 angezeigt wird. Sie entspricht der Form der veralteten Trajektorie vom Zeitpunkt t, also der veralteten Trajektorie. Zwischen den Endpunkten der beiden unteren Trajektorien entsteht so ein Offset in sowohl Richtung als auch Position. Dem Fahrer des Gespanns wird so die Orientierung erschwert, da ihm eine fehlerhafte Hilfslinie angezeigt wird. Eine analoge Betrachtung kann für den Lenkwinkel durchgeführt werden.

Dieser Fehler ist vor allem signifikant, wenn sich die benötigten Sensorwerte in rascher Folge ändern. Ändern sich Einknick- oder Lenkwinkel kaum oder gar nicht während

des Manövers, so wird der Nutzer kaum einen Unterschied bemerken. Steht das Fahrzeug still, so wird zumindest die Gespann-Trajektorie stets korrekt angezeigt.

Verzögerungen in der Akquirierung von Rückfahrkamera-Frames

In der derzeitigen Form der Assistenz werden Kamerabild und Trajektorien unabhängig voneinander visualisiert. Es wird also nicht sichergestellt, dass das derzeitige Kamerabild vom Alter her zu den empfangenen Winkeln passt. Durch eine Verzögerung des Kamerabildes werden daher zumindest die Hilfs-Trajektorien nicht weiter beeinflusst.

Das Problem, welches durch eine verzögerte Rückfahransicht auftritt, betrifft den Fahrer und dessen Fahrtziel. Ein Feature der Assistenz ist das Zielen mittels der blauen Hilfs-Trajektorien. Der Fahrer wählt sich ein Ziel und richtet die Trajektorien auf dieses Ziel aus.

Wird das Kamerabild verzögerungsfrei angezeigt, so entspricht das Bild der Grundwahrheit der aktuellen Perspektive der Rückfahrkamera. Entsteht nun eine Verzögerung zwischen Aufnahme und Anzeige des Kamerabildes, so ist nicht mehr sichergestellt, dass die Grundwahrheit mit der angezeigten Perspektive übereinstimmt.

Abbildung 3.14 zeigt einmal mehr zwei Gespanne zu verschiedenen Zeitpunkten t und t+1. Die orangenen Kegelstümpfe am Heck der Anhänger repräsentieren die Perspektive der Rückfahrkamera unter einem vorgegebenen Öffnungswinkel. Der obere Kegelstumpf beschreibt das zum Zeitpunkt t aufgenommene Bild.

Unterliegt die Aufnahme einer gewissen Verzögerung, so bewegt sich das Fahrzeug in der Zeit bis zur Anzeige in die Position zum Zeitpunkt t+1. Der hier dargestellte Kegelstumpf zeigt schließlich das zum Zeitpunkt t aufgenommene Bild. Die Verzögerung des Kamerabildes bewirkt also, dass die Assistenz eine veraltete Ansicht auf dem Monitor anzeigt. Statt nun den eigentlichen Zielpunkt $PosGoal_t$ anzuvisieren, fährt der Fahrer des Gespanns den Punkt $PosGoal_{t+1}$ an, von dem er glaubt, das

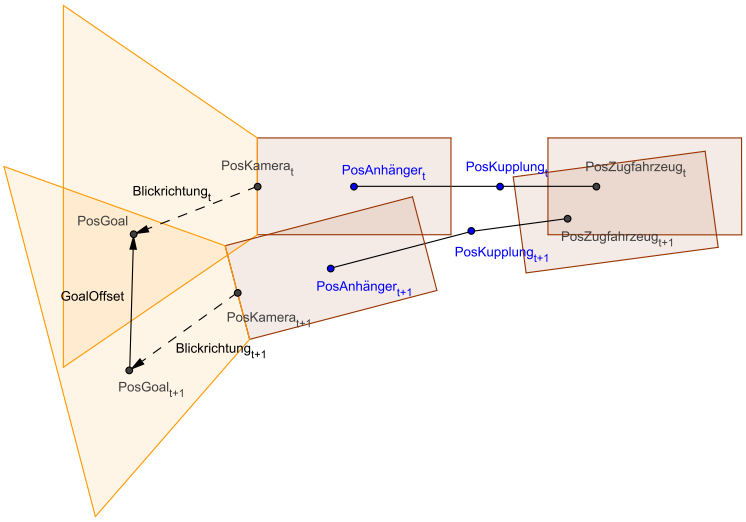


Abbildung 3.14: Der Fahrzeugzustand zu verschiedenen Zeitpunkten. Erst zum Zeitpunk t+1 wird aufgrund Verzögerungen das Kamerabild angezeigt, welche schon zum Zeitpunkt t gezeigt werden sollte.

Ziel zu sein.

Der Punkt $PosGoal_{t+1}$ besitzt dieselbe relative Pose zur Kameraposition $PosKamera_{t+1}$ wie $PosGoal_t$ zu $PosKamera_t$. Der Fahrer fährt zum Zeitpunkt $t+1$ also in dieselbe Richtung, in die er zum Zeitpunkt t ohne Verzögerung fahren würde. Das eigentliche Ziel jedoch liegt unverändert bei $PosGoal_t$.

Der globale Offset zwischen diesen beiden Positionen bestimmt den Grad des Fehlers, unter dem der Fahrer schließlich das Ziel anfährt. Da die relative Pose zwischen den beiden Zielpunkten wieder exakt der relativen Pose zwischen den beiden Anhängern entspricht, kann die im vorangegangenen Kapitel vorgestellte Methode zur Transformation zwischen Koordinatensystemen genutzt werden, um $PosGoal_{t+1}$ aus $PosGoal_t$ zu berechnen.

Analog zur Verzögerung im Einknickwinkel hat dieser Fehler keine Relevanz, wenn das Fahrzeug still steht.

3.3 Entwicklung eines Controllers zur Simulation des Fahrverhaltens bei Nutzung der Assistenz

Das folgende Kapitel befasst sich mit der Entwicklung eines Controllers, welcher im Gewichtungsschritt des Annealed Particle Filters zum Einsatz kommt. Dieser soll in der Lage sein, ein Manöver auszuführen unter Zuhilfenahme der Informationen des Fahrerassistenzsystems.

In Abschnitt 3.3.1 werden zunächst die Charakteristika eines Manövers erläutert, welche als Voraussetzung dienen. Im folgenden Abschnitt 3.3.2 werden die Anforderungen an den Controller definiert. Eine Vorstellung des erarbeiteten Konzepts erfolgt in Abschnitt 3.3.3. Schließlich wird im letzten Abschnitt 3.3.4 der fertige Algorithmus präsentiert, welcher als Basis für die geplante Implementierung dient.

3.3.1 Voraussetzungen

Um die Anforderungen eines Controller herzuleiten, der ein Manöver simulieren soll, muss man zunächst den Begriff des Manövers näher betrachten. Ein Manöver wird typischerweise durch folgende Parameter definiert:

- Kinematische Beschränkungen des Fahrzeugs
- Start- und Endposition (bzw. -ausrichtungen)
- Fahrtrichtung und -geschwindigkeit

Die kinematischen Beschränkungen geben an, mit wie vielen operativen Freiheitsgraden sich das Fahrzeug letztendlich im Arbeitsraum bewegen kann. Zusätzlich zu den natürlichen Beschränkungen der Fahrzeugbewegung können bestimmte Ein- bzw. Ausgabewerte auf ein festgelegtes Intervall beschränkt werden, Beispiele hierfür seien der Einknickwinkel eines Gespanns oder der Lenkwinkel, welche gewisse Werte nicht überschreiten dürfen. Als Fahrzeugmodell wird hier das bereits erläuterte kinematische Modell eines Gespanns mit Einachsanhänger genutzt

Die Start- und Endposition sowie die Startausrichtung bestimmen die Art des Manövers, das ausgeführt werden soll. Eine Zielausrichtung des Fahrzeugs wird nicht benötigt, da die Assistenz nicht darauf ausgelegt ist, eine feste Ausrichtung am Zielpunkt zu erreichen.

Die Fahrtrichtung beeinflusst die Operation des Fahrzeug drastisch, je nachdem welches Fahrzeugmodell zugrunde gelegt wird. Das Modell mit Einachsanhänger ist beispielsweise rückwärts sehr viel schwerer zu steuern, als ein Fahrzeugmodell ohne Anhänger.

Die Geschwindigkeit wiederum gibt an, wie schnell bzw. stetig sich ein Fahrzeug bewegt. Da das genutzte Fahrzeugmodell eine höhere Genauigkeit für geringe Geschwindigkeiten besitzt, sind diese zu bevorzugen.

Nutzt ein Fahrer die Assistenz, so muss zusätzlich beachtet werden, wie der Nutzer durch die visuellen Hilfestellungen der Assistenz beeinflusst wird. Die Assistenz liefert neben den extrapolierten Hilfslinien eine Endposition und Endausrichtung der Extrapolation, welche der Anhängerkonfiguration am Endpunkt der Hilfslinien entsprechen.

Dem Fahrer steht diese Konfiguration nicht unmittelbar zur Verfügung, er kann sich lediglich an der Richtung und dem geschätzten Endpunkt der Linien orientieren, wodurch Fehler auftreten können. Der Controller hingegen hat direkten Zugriff auf die genauen Werte der Konfiguration und kann somit beim Zielen auf einen bestimmten Punkt, der angefahren werden soll, eine höhere Genauigkeit erreichen. Ebenso ist es dem Fahrer oft nicht möglich die grüne Linie exakt in der Mitte der blauen Linien zu zentrieren, wodurch es zu Abweichungen bei der stabilen Fahrt kommen kann. Der Controller wiederum kann den stabilen Lenkwinkel direkt berechnen und ist so in der Lage, den Einknickwinkel zu halten.

3.3.2 Anforderungen

Es lassen sich folgende Anforderungen für den Controller definieren:

- Das Fahrzeug soll rückwärts fahren
- Die Informationen der Assistenz sollen zur Bestimmung eines Steuerbefehls genutzt werden
- Das Ziel soll bis auf eine vorgegebene Genauigkeit erreicht werden
- Die gefahrene Geschwindigkeit soll konstant sein
- Die Rate von Lenkwinkeländerungen soll beschränkt werden
- Der Fahrer soll unmittelbar auf die Informationen des Assistenzsystems reagieren

Die erste Anforderung ist trivial, da ohnehin die Rückfahrassistenz genutzt werden soll.

Die zweite Anforderung beinhaltet, dass sich der Controller bei der Berechnung eines Steuerbefehls an den visuellen Hilfestellungen der Assistenz orientiert. Zur Verfügung stehen die blaue Gespann-Trajektorie sowie die grüne Anhänger-Trajektorie.

Die dritte Anforderung ist eine Anforderung an die Korrektheit des Controllers. Sofern sich das Fahrzeug nicht in einem illegalen Zustand befindet, soll es in der Lage sein, den Zielpunkt mit einer festgelegten, maximalen Abweichung zu erreichen.

Die vierte Anforderung umfasst, dass das Fahrzeug nicht stehen bleiben darf, um etwa größere Korrekturen am Lenkwinkel vorzunehmen. Es soll gezeigt werden, dass mit der Sicherheit der Assistenz das Rangieren des Fahrzeugs ohne abzusetzen erfolgen kann.

Die fünfte Anforderung leitet sich aus der vierten ab. Würde man im Stehen den Lenkwinkel ändern, so entspräche dies einer unendlichen Lenkwinkeländerung im Lenkprofil. Doch selbst im physikalisch möglichen Bereich der Lenkwinkeländerun-

gen werden kleine Änderungsraten bevorzugt, da dies den Fahrer in seinen Lenkbewegungen entlastet.

Die letzte Anforderung betrifft das kognitive Fahrermodell. In der Realität besitzt der Fahrer eine gewisse Reaktionsverzögerung, bis er auf den visuellen Input der Assistenz reagieren kann. Der Informationsfluss, dem der Fahrer unterliegt, ist zudem kontinuierlich und schnell aufeinander folgende Eindrücke können sich überlappen. Die Komplexität eines solchen Modells macht eine adäquate Modellierung im Rahmen dieser Arbeit unmöglich. Folglich wird von einem vereinfachten Fahrermodell ausgegangen.

3.3.3 Konzept

Das im Folgenden vorgestellte Konzept beschreibt, wie aus der aktuellen Fahrzeugposition sowie den Hilfslinien der Fahrerassistenz ein Steuerbefehl generiert werden kann, der das Fahrzeug näher zum Ziel bringt. Wie in Kapitel 2.1.3 beschrieben, stehen dazu zwei Möglichkeiten zur Verfügung: die Nutzung der Gespann-Trajektorie oder die Nutzung der Anhänger-Trajektorie.

Im Folgenden wird die erste Möglichkeit betrachtet. Je nachdem, wo der anzufahrende Zielpunkt liegt, unterscheidet man zwei Fälle, um die Assistenz zum Zielen zu nutzen.

- **Fall 1:** Der Zielpunkt liegt aus Fahrerperspektive vor den Endpunkten der Assistenz-Trajektorie. Die Endpunkte lassen sich demnach ohne weiteres auf das Ziel ausrichten. Abbildung 3.16 zeigt eine schematische Darstellung dieses Vorgehens.

  Der **Richtungsvektor** entspricht dem Ist-Wert der Anhängerausrichtung am Endpunkt der Gespann-Trajektorien. Der **Zielvektor** beschreibt den Soll-Wert der Anhängerausrichtung. Für den Winkel α zwischen Ist- und Soll-Vektor gilt

hierbei

$$|\alpha| < 90°$$

Ziel des Controlling ist es, den Ist-Vektor in Richtung des Soll-Vektors zu verschieben. Die Ausrichtung des Ist-Vektors ändert sich dabei nur, wenn die Gespann-Trajektorien neu ausgerichtet werden. Dies erfordert die Wahl eines geeigneten Lenkwinkels, durch den der Einknickwinkel in der Art geändert wird, dass der resultierende Trajektorien-Endpunkt zu einer Verkleinerung von α führt.

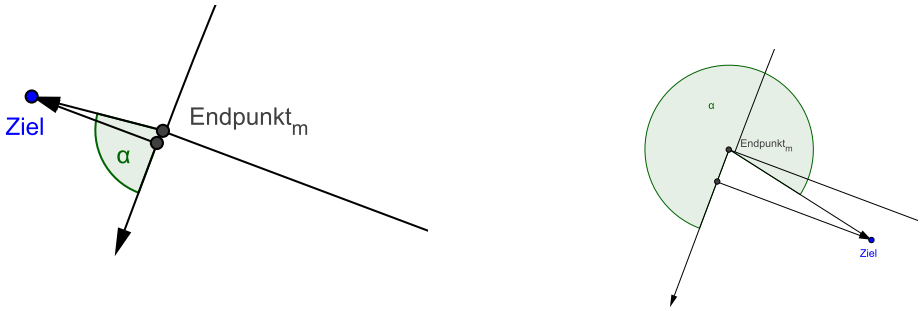


Abbildung 3.15: Wird das Ziel passiert, so geht der Winkel α gegen 90° (links). Obwohl sich der Endpunkt in der unmittelbaren Nähe des Ziels befindet, führt der große Winkel zu einem starken Überschwingungsverhalten (rechts).

Eine alleinige Abhängigkeit des Lenkwinkels vom Winkel α führt spätestens dann zu Problemen, wenn der **Endpunkt** dem **Ziel** sehr nahe kommt. Sind die beiden Punkte leicht versetzt zueinander, geht α gegen 90°, was zu sehr großen Lenkwinkeln führen kann. Die Folge ist ein Überschwingungsverhalten, zu sehen in Abbildung 3.15, bis $\alpha \geq 90°$ gilt.

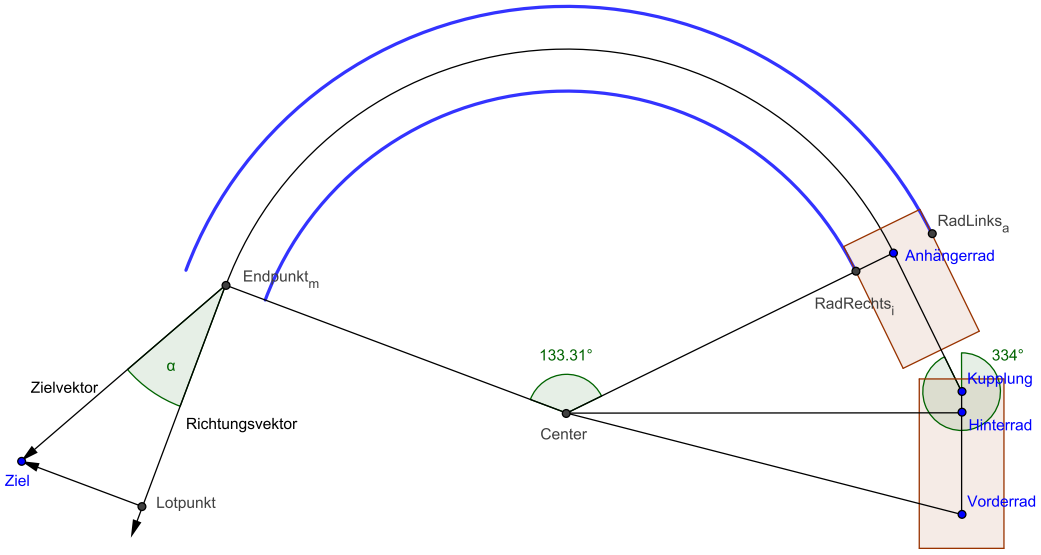


Abbildung 3.16: Der Ziel-Schritt des Controllers. Endpunkt und Richtungsvektor spannen eine Gerade auf. Der Abstand der Ziels zu der Gerade bestimmt die Stärke der Lenkbewegung.

Der Lenkwinkel wird daher über den folgenden Ansatz bestimmt: Der **Lotpunkt** entspricht dem **Zielpunkt** projeziert auf die Gerade, welche durch **Endpunkt** und **Richtungsvektor** definiert wird. Der Abstand zwischen **Lotpunkt** und **Ziel** beeinflusst in der Folge die Wahl des Lenkwinkels. Verschiebt sich der Ist-Vektor in Richtung Soll-Vektor, so nähern sich die beiden Punkte automatisch an. Der Vorteil ist jedoch, dass es keine starken Überschwingungen gibt, wenn sich **Ziel** und **Endpunkt** annähern. Sei **d** der Abstand zwischen **Ziel** und **Lotpunkt**. Dann berechnet sich der Lenkwinkel des folgenden Zeitschritts aus

$$\phi = \phi_{stable} + c * d \tag{3.34}$$

Der Faktor **c** ist ein Glättungsfaktor, der beschreibt wie stark der Abstand in den Lenkwinkel eingeht.

Der stabile Lenkwinkel ϕ_{stable} wird als Basis für endgültigen Lenkwinkel ge-

nommen. Der Hintergrund ist, dass sich der Einknickwinkel erhöht, wenn man einen Lenkwinkel kleiner als den stabilen anlegt. Umgekehrt vermindert sich der Einknickwinkel, wenn ein Lenkwinkel größer als der stabile anliegt. Es reicht also einen Lenkwinkel anzuwenden, welcher um einen kleinen Wert vom stabilen Lenkwinkel abweicht, um die Ausrichtung der Trajektorie in eine Richtung zu beeinflussen.

- **Fall 2:** Der Zielpunkt liegt aus Fahrerperspektive hinter den Endpunkten bzw. neben der Assistenz-Trajektorie. Statt einer Ausrichtung der Endpunkte auf das Ziel, welche nun nicht möglich ist, müssen die Gespann-Trajektorien so ausgerichtet werden, dass der anzufahrende Punkt genau zwischen ihnen liegt.

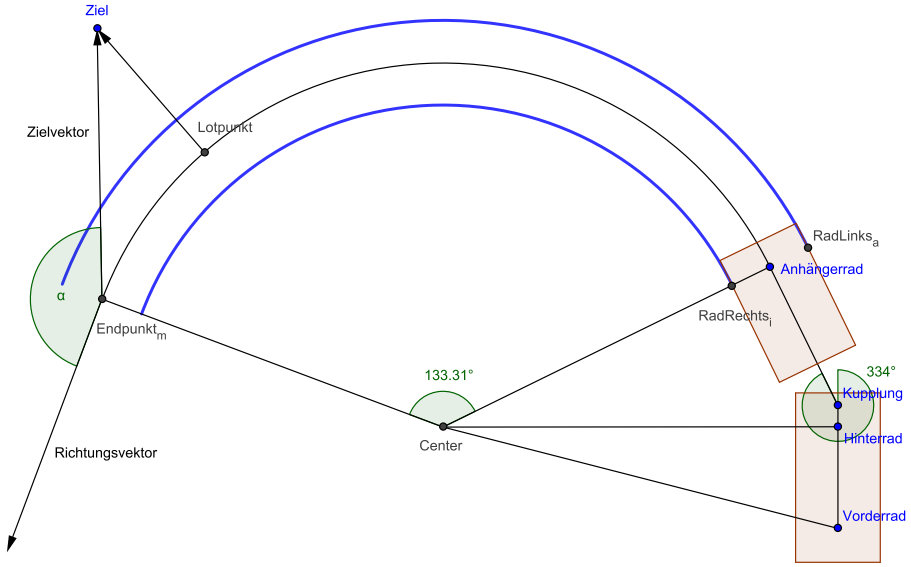


Abbildung 3.17: Der Radiusanpassungs-Schritt des Controllers. Dieses Mal beeinflusst der Abstand der Ziels zur Kreisbahn der Anhängerachse, wie stark gelenkt wird.

Abbildung 3.17 zeigt wie in diesem Fall vorgegangen wird. Für den Winkel α zwischen Ist- und Soll-Vektor gilt hier

$$|\alpha| \geq 90° \tag{3.35}$$

Algorithm 4 Update-Schritt des Controllers

```
1: procedure UPDATE
2:     while !endConditionMet() do
3:         T ← Assistance.VehicleTrajectory
4:         G ← GoalPoint
5:         TrailerToGoalVector ← G - T.EndPos
6:         TrailerOrientVector ← (cos(T.EndOrient), sin(T.EndOrient))
7:         Alpha ← getAngleBetween(TrailerOrientVector, TrailerToGoalVector)
8:         if |Alpha| < π/2 then
9:             changeDirection()
10:        else
11:            changeRadius()
```

Das Prinzip zur Lenkwinkelbestimmung aus Möglichkeit 1 bleibt erhalten, nur entspricht der **Lotpunkt** hierbei dem **Ziel** projeziert auf das Kreissegment, welches durch *Center*, *Anhängerrad* und *Endpunkt* aufgespannt wird. Der Vektor zwischen **Lotpunkt** und **Ziel** ist hierbei orthogonal zur Kreistangente, die am **Lotpunkt** anliegt.

3.3.4 Algorithmus

Das zuvor beschriebene Vorgehen dient als Basis für den letztendlich ausgearbeiteten Algorithmus, welcher in Algorithmus 4 zu sehen ist. Als Eingabe stehen die Trajektorie des Gespanns sowie der Zielpunkt zur Verfügung. Zunächst werden die benötigten Vektoren gebildet, um den Winkel α zu berechnen. Die Funktion *getAngleBetween* berechnet den Winkel zwischen zwei Vektoren anhand des Skalarprodukts. Ist α kleiner als 90°, tritt Fall 1 in Kraft, ansonsten Fall 2. Die Berechnung wird solange wiederholt, bis eine Endbedingung erreicht wird, worauf später näher eingegangen wird.

Zunächst werden die beiden Funktionen *changeDirection* und *changeRadius* betrachtet. Erstere ist in Algorithmus 5 zu sehen. Der hier berechnete Wert **Dist** beschreibt den Abstand zur Geraden, welche durch Endpunkt und Richtungsvektor aufgespannt wird wie in Abschnitt 3.3.3 beschrieben. Auf Grundlage dessen wird der Lenkwinkel

Algorithm 5 Zielen mit den Endpunkten der Assistenz-Trajektorien des Gespanns

```
1: procedure changeDirection
2:     T ← Assistance.VehicleTrajectory
3:     G ← GoalPoint
4:     S ← SmoothnessDirection
5:     TrailerOrientVector ← (cos(T.EndOrient), sin(T.EndOrient))
6:     Line ← (TrailerOrientVector, G)
7:     Dist ← Line.getDistance(T.EndPos)
8:     setSteeringAngle(S * Dist)
```

gesetzt.

Die Funktion *changeRadius* wird in Algorithmus 6 vorgestellt und ist etwas komplexer als *changeDirection*. Zunächst muss überprüft werden, ob die Gespann-Trajektorie einen Kreis beschreibt oder annähernd eine Gerade. Diese Unterscheidung muss getroffen werden, da eine Gerade einem Kreis mit unendlichem Radius entspricht und somit keinen reellwertigen Mittelpunkt besitzt. Die Berechnung des Lotpunkts auf dem Kreis ist somit nicht möglich. Stattdessen wird der Kreis ab einer festgelegten Größe des Radius als Linie aufgefasst, dementsprechend auch der Abstand **Dist** berechnet wird. In beiden Fällen erfolgt anschließend eine Bestimmung des Lenkwinkels für den nächsten Schritt.

Die dafür zuständige Funktion *setSteeringAngle* ist in Algorithmus 7 zu sehen. Neben der Berechnung des Lenkwinkels aus dem stabilen Winkel sowie dem Distanzterm, erfolgt hier eine Beschränkung des maximalen Lenkwinkels in beide Richtungen hinsichtlich physikalischer Beschränkungen des Fahrzeugs. Zusätzlich wird überprüft, ob die Änderung des Lenkwinkels vom letzten Zeitschritts innerhalb festgelegter Grenzen liegt, welche entweder ebenfalls physikalischen Beschränkungen entsprechen oder aber der Leistungsfähigkeit bzw. Zumutbarkeit des Fahrer entspricht.

Die Funktion *endConditionMet*, zu sehen in Algorithmus 8 zeigt schließlich das Abbruchkriterium, durch das die Ausführung des Controllers zum Erliegen kommt.

Die erste Möglichkeit ist trivialerweise das Erreichen des Zielpunkts bis auf eine hinreichende Genauigkeit, welche vorab definiert wird. Ein Zielpunkt gilt als erreicht,

Algorithm 6 Ausrichten der Assistenz-Trajektorien des Gespanns auf den Zielpunkt

```
1:  procedure changeRadius
2:      T ← Assistance.VehicleTrajectory
3:      G ← GoalPoint
4:      S ← SmoothnessRadius
5:      Δx ← T.StartPos.x - T.EndPos.x
6:      Δy ← T.StartPos.y - T.EndPos.y
7:      r ← getCircleRadius(T.EndPos, T.StartPos, T.StartOrient)
8:      c ← cos(T.StartOrient)
9:      s ← sin(T.StartOrient)
10:     if Δx * c == Δy * s || |r| > 10000 then
11:         Line ← (T.StartPos, T.EndPos)
12:         Dist ← Line.getDistance(G)
13:     else
14:         Center ← computeCenter(T.EndPos, T.StartPos, T.StartOrient)
15:         Circle ← (T.StartPos, T.EndPos, Center)
16:         Dist ← Circle.getDistance(G)
17:     setSteeringAngle(S * Dist)
```

Algorithm 7 Anwendung Lenkwinkels für den nächsten Controllerschritt

```
1: procedure setSteeringAngle(Angle)
2:     C ← CouplingAngle
3:     SteeringAngle ← getStableSteerngAngle(C) + Angle
4:     Adjust SteeringAngle to be in [MinSteering, MaxSteering]
5:     DotSteering ← SteeringAngle - LastSteering
6:     Adjust SteeringAngle so that DotSteering is in [MinDotSteering, MaxDotS-
   teering]
7:     Apply SteeringAngle to vehicle
```

wenn die Anhängerposition das Ziel passiert hat und sich im Umkreis von einer Fahrzeugbreite befindet.

Die zweite Möglichkeit ist der Abbruch, falls sich das Fahrzeug über eine festgelegte Anzahl an sukzessiven Iterationsschritten nicht dem Zielpunkt annähert. Ein Grund hierfür kann sein, dass sich das Fahrzeug in einem illegalen Zustand befindet, etwa durch Überschreitung des maximalen Einknickwinkels, und ohne Vorzusetzen nicht weiterfahren kann. Diese zusätzliche Abfrage sichert den Abbruch des Controller für den Fall, dass unvorhergesehene Fehler auftreten.

Algorithm 8 Abbruchkriterium des Controllers

```
1: procedure endConditionMet
2:     T ← Assistance.VehicleTrajectory
3:     G ← GoalPoint
4:     W ← TrailerWidth
5:     TrailerToGoalVector ← G - T.StartPos
6:     TrailerOrientVector ← (cos(T.StartOrient), sin(T.StartOrient))
7:     Angle ← getAngleBetween(TrailerOrientVector, TrailerToGoalVector)
8:     Dist ← |TrailerToGoalVector|
9:     if Counter ≥ MaxCounter || (Dist ≤ W && |Angle| ≥ π/2) then
10:        Counter ← 0
11:        LastDist ← 999999
12:        return true
13:    else
14:        if Dist ≥ LastDist then
15:            Counter++
16:        else
17:            Counter ← 0
18:        LastDist ← Dist
19:        return false
```

3.4 Bewertung eines Manövers

Im folgenden Kapitel wird auf die im Sampling-Schritt des Annealed Particle Filters genutzte Gewichtungsfunktion eines Manövers eingegangen, welche es erlaubt ein Manöver anhand einer skalaren Kennzahl zu bewerten.

Abschnitt 3.4.1 befasst sich mit der Auswahl geeigneter Gütemaße sowie möglicher Parametrisierungen. In Abschnitt 3.4.2 wird schließlich das Verfahren vorgestellt die ausgewählten Maße zu einer skalaren Kennzahl zu aggregieren.

3.4.1 Auswahl geeigneter Maße

Das **Dynamic Time Warping** erwies sich in Experimenten als ein sehr schnelles und intuitives Maß zwischen zwei Trajektorien. Sie ist der Berechnung der EMD vorzuziehen, da sie ihr gegenüber zahlreiche Vorteile besitzt.

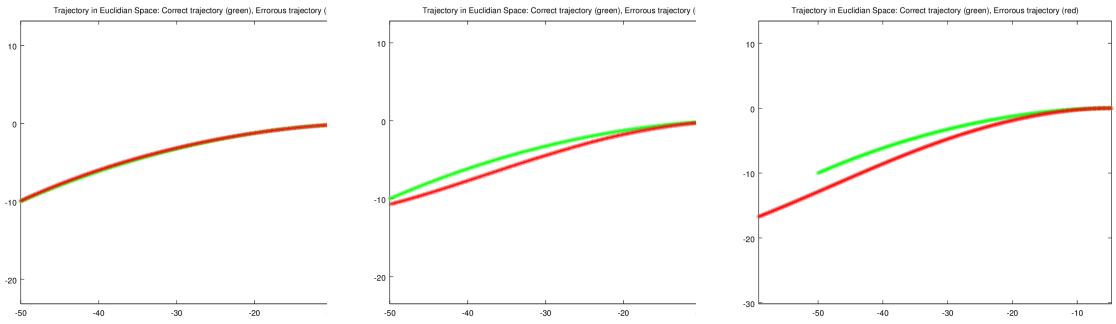


Abbildung 3.18: Zu sehen ist der Vergleich zwischen je zwei Trajektorien-Datensätzen. Der grüne Datensatz ist stets identisch und bildet den Referenz-Datensatz. Er beginnt bei (0,0) und endet bei (-50,-10).

Abbildung 3.18 zeigt den Vergleich zwischen jeweils zwei Trajektorien-Datensätzen. Die grüne Trajektorie entspricht der Referenz-Trajektorie. Die rote soll in der Folge auf Ähnlichkeit mit der grünen verglichen werden. In Tabelle 3.1 sind die jeweiligen Wertigkeiten der beiden Maßzahlen zu sehen. Für die erste und zweite Abbildung scheinen die Wertigkeiten zu korrelieren. So entspricht die EMD annähernd einem Zehntel der DTW, siehe Tabelle 3.2.

Beide Maße geben zudem korrekt wieder, dass die Trajektorien aus Abbildung 3.18 links eine intuitiv höhere Ähnlichkeit besitzen als die in der Mitte. Betrachtet man jedoch die rechte Grafik, so fällt auf, dass der Wert der DTW wie erwartet sehr groß wird, da die Länge der beiden Trajektorien zum einen stark voneinander abweicht, zum anderen die Endpunkte sehr weit voneinander entfernt liegen. Der Wert der EMD jedoch verringert sich interessanterweise in diesem Fall. Dies kann dem Fakt

Vergleich	Punkte	DTW	Zeit	EMD	Zeit
1	425	0.0753042	0.00637026s	0.00676578	0.335387s
2	425	0.815333	0.00555884s	0.0677859	0.419683s
3	525	2.26047	0.00722967s	0.0580879	0.494423s

Tabelle 3.1: Vergleich zwischen Dynamic Time Warping und Earth Mover's Distance für Abbildung 3.18 anhand von Wertigkeiten, Berechnungszeit sowie deren jeweiligem Verhältnis. Für die EMD wurde eine Rastergröße von $0.1m \times 0.1m$ gewählt. Die Anzahl der Punkte der grünen Trajektorie war jeweils 425.

geschuldet sein, dass im Falle von Trajektorien unterschiedlicher Länge Anpassungen in Form von virtuellen Senken und Quellen getroffen werden müssen, siehe Kapitel 2.2.2. In der Folge werden nur die Teilabschnitte der Trajektorie betrachtet, welche einen optimalen Kostenwert erzeugen, während die Teile mit sehr großen Abweichungen ignoriert werden.

Betrachtet man nun zusätzlich die Berechnungszeiten der einzelnen Werte von DTW und EMD, so fällt auf, dass die Berechnung der DTW einen erheblichen Geschwindigkeitsvorteil gegenüber der EMD besitzt, etwa um einen Faktor 50 bis 100. Dies hängt mit der Komplexität des Verfahrens zusammen. Der DTW-Algorithmus rechnet in $\mathcal{O}(n^2)$, während das der EMD zugrunde liegende Simplex-Verfahren bestenfalls polynomiale Laufzeit besitzt.

Die **Steering Reversal Rate** ist das zweite Maß, welches sich in initialen Experimenten bewährt hat. Die Herausforderung bei diesem Maß war es, eine Parametrisierung zu finden, welche zu brauchbaren Ergebnissen führt. Die zu betrachtenden Parameter umfassen zum einen die Gapsize zwischen stationären Punkten, deren Überschreitung ein Reversal anzeigt, siehe 2.2.4. Zum anderen müssen die Parameter des Resamplings mit Bedacht gewählt werden, der hierbei genutzte Frequenz-Filter ist der Butterworth-Filter [Butterworth, 1930]. Es galt eine Balance zu finden, um Ausreißer und Störungen bestmöglich aus dem Signal herauszufiltern, ohne zu einem zu hohen Grad an Informationsverlust zu führen.

Abbildung 3.19 zeigt drei verschiedene Lenkwinkeldatensätze. Der erste Datensatz zeigt eine leicht verrauschte Lenkwinkelfolge. Aufgrund der geringen Standardabweichung des Rauschens besitzt er auch ungefiltert eine geringe Reversal Rate wie

Vergleich	Verhältnis DTW zu EMD	Verhältnis der Zeiten
1	11.13	0.019
2	12.028	0.01325
3	38.915	0.0146

Tabelle 3.2: Die obigen Verhältnisse beziehen sich auf Tabelle 3.1

Tabelle 3.3 zu entnehmen ist. Die Reversal Rate des gefilterten Datensatzes ist daher nur marginal besser. Der zweite Datensatz beschreibt ein sehr viel stärker verrauschtes Signal als der erste. Entsprechend hoch ist die durch Ausreißer induzierte Reversal Rate von 41. Durch Anwendung des Butterworth-Filters konnte die Zahl der Ausreißer um ein Vielfaches reduziert werden, was sich ebenfalls in einem drastischen Rückgang der Reversal Rate auf 3 niederschlägt. Analog verbessert sich auch die Reversal Rate des dritten Datensatzes von 77 auf 3.

Ein Problem, welches sich im Laufe der Experimente herausstellte, war, dass die Filterung zu einer zu starken Glättung des Lenksignals führte. Informationen über bewusste Lenkentscheidungen gingen somit leider verloren. Die resultierenden Reversal Rates der drei Experimente lagen somit in einem ähnlichen Bereich, obwohl die Lenkbewegungen in Abbildung 3.19 unten deutlich ausgeprägter waren als in einer der oberen Grafiken etwa. Letztendlich konnte keine Parametrisierung gefunden werden, welche dieses Problem löst. In der Folge wurde auf eine Filterung des Signals verzichtet.

Die Reversal Rate hängt jedoch noch immer stark von der gewählten Gapsize ab. Wählt man diese zu groß, so werden geringe Richtungsänderungen ignoriert. Wählt man sie zu klein, so wird jedes Zucken des Fahrers als Richtungsänderung interpretiert, was den Wert stark verfälschen kann. Im Kontext dieser Arbeit wurde eine Gapsize von 0.33° für den Lenkwinkel gewählt, was ca. 5° Lenkradwinkel entspricht.

Zusätzlich zu diesen beiden komplexeren Metriken wurde eine Reihe an simplen Metriken ausgewählt, welche einfach zu berechnen sind. Zu diesen gehören:

Datensatz	SRR ohne Filter	SRR mit Filter
1	2 / 176	1 / 176
2	41 / 176	3 / 176
3	77 / 176	3 / 176

Tabelle 3.3: Die Steering Reversal Rates für die drei betrachteten Datensätze. Als Gapsize wurden 1.5° gewählt. Unter Einsatz eines Filters sind die Werte weniger rauschanfällig und geben bewusste Lenkentscheidungen des Fahrers besser wieder.

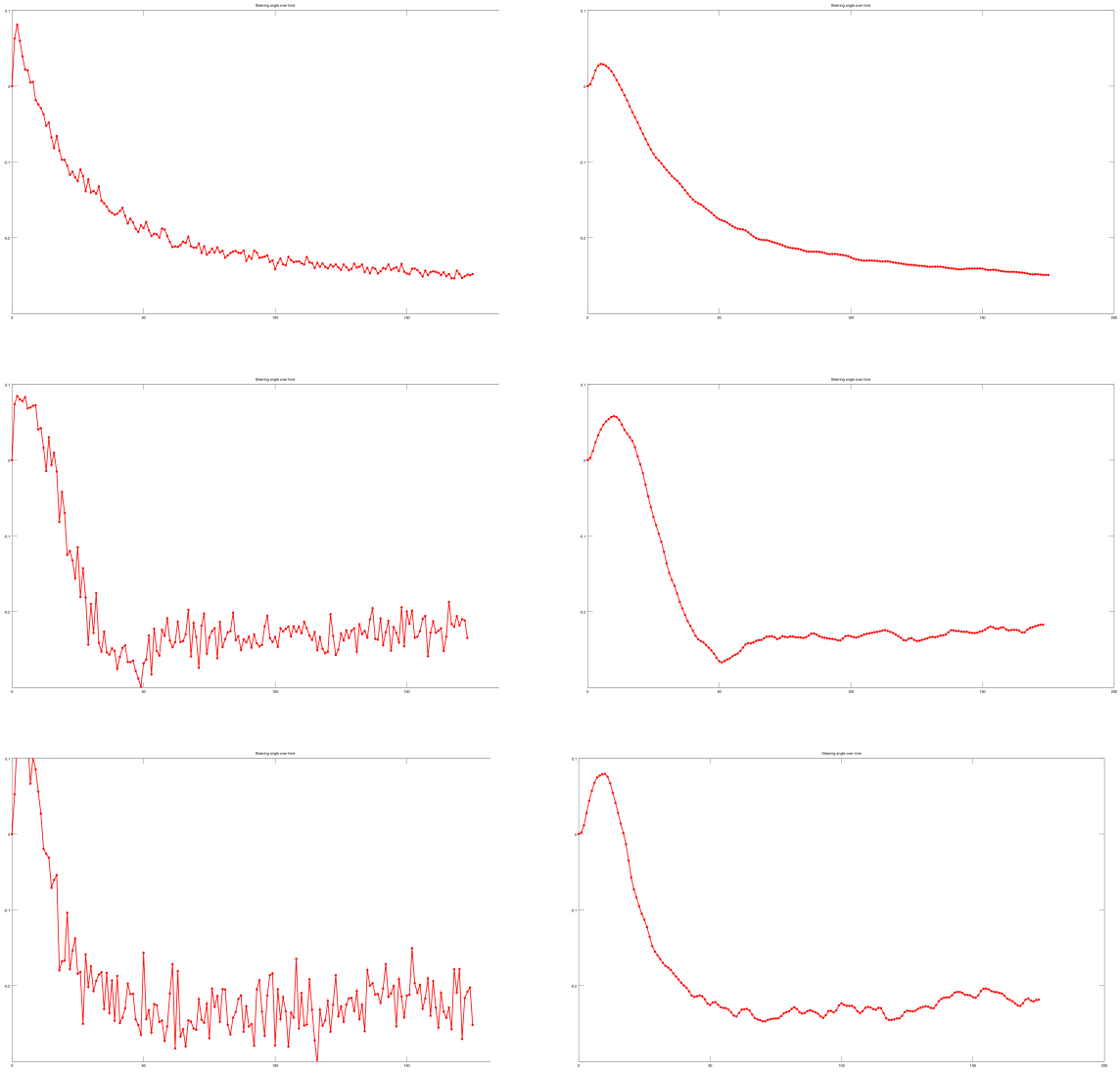


Abbildung 3.19: Zu sehen sind drei unterschiedliche Lenkwinkel-Datensätze, einmal ungefiltert(links) und einmal gefiltert (rechts). Der angewandte Filter ist ein Butterworth-Filter zweiter Ordnung mit Resampling Rate 100Hz und Cutoff-Frequenz 10Hz. Ist das Signal stark verrauscht, so sorgt der Butterworth-Filter für eine Ausdünnung und Glättung.

- **Standardabweichung des Lenkwinkels**: Diese ergibt sich aus der Varianz der Lenkwinkel, siehe Abschnitt 2.2.3, und bildet ein unterstützendes Maß zur Steering Reversal Rate.
- **Positions-Offset**: Die Distanz zwischen Zielposition und tatsächlicher Endposition des Anhängers nach Beendigung des Manövers. Dieses Maß erlaubt es Schlüsse zu ziehen, ob die Assistenz unter dem Einfluss verschiedener Fehler noch in der Lage ist, den Nutzer an die gewünschte Stelle zu navigieren.
- **Winkel-Offsets**: Hierbei werden, analog zum Positions-Offset, die Offsets von Anhängerausrichtung und Einknickwinkel betrachtet. Als Vergleichswert dient ein Referenzdatensatz, in welchem die Position "optimal" angefahren wurde. Dieses Maß ist ebenfalls ein unterstützendes, da es zusätzlich zu der Frage, ob der Zielpunkt erreicht wurde, Hinweise darauf gibt, wie er erreicht wurde.
- **Winkel-Extremwerte**: Hierbei werden die Extremwerte von Lenk- und Einknickwinkel betrachtet. Während hohe Lenkwinkel vor allem auf reduzierten Komfort des Fahrers bei der Nutzung des Assistenz hindeuten, bilden hohe Einknickwinkel vor allem eine Gefahr für die durchgängige Steuerbarkeit des Fahrzeuggespanns und sollten damit vermieden werden.

3.4.2 Aggregation von Maßen

Nachdem eine Auswahl der Metriken stattgefunden hat, wird im folgenden Abschnitt das Verfahren zur Aggregation der einzelnen Kennzahlen vorgestellt.

Die folgenden Vektoren werden zunächst definiert:

- **M**: Dieser Vektor enthält alle oben genannten Maßzahlen zu einem Manöver.
- **O**: Dieser Vektor erlaubt die Definition von Grenzwerten für eine Maßzahl. Viele Maßzahlen besitzen ein natürliches Optimum von 0, etwa der Positions-Offset oder das DTW. Die Definition von Grenzwerten erlaubt es ein Manöver

Algorithm 9 Normalisierung von Maßzahlen in Abhängigkeit von ihren Optima

```
1: Input: M, O
2: Output: normedVals
3: diracDeltaMax ← diracDelta(0)
4: for i = 0; i < size(M); i++ do
5:     percentageVal ← M(i)\O(i)
6:     normedVals(i) ← diracDelta(1 - percentageVal) \diracDeltaMax
```

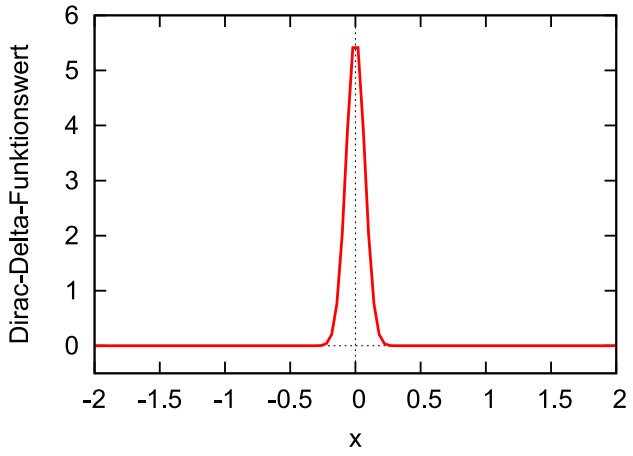


Abbildung 3.20: Naeherungsdarstellung der Dirac-Delta Funktion anhand der Gaussfunktion, entnommen aus [Happe, 2008]. Der Wert für a beträgt 1/200.

nach anderen Gesichtspunkten zu untersuchen, etwa den maximalen Positions-Versatz, welcher unter gewissen Anfangsbedingungen erreicht wird.

- **W**: Dieser Vektor enthält Gewichte, welcher die einzelnen Gütekriterien nach Relevanz bewertet. Je höher das Gewicht einer Maßzahl, desto höher geht ihr Wert in die aggregierte Maßzahl ein.

Um eine Vergleichbarkeit zwischen verschiedenen Maßzahlen herstellen zu können, müssen sie auf eine gemeinsame Domäne normalisiert werden, etwa das Intervall $[0, 1]$. Das Vorgehen wird dabei in Algorithmus 9 beschrieben.

Die Dirac-Delta-Funktion [Dirac] ist hierbei eine Funktion, welche theoretisch an allen Stellen den Funktionswert 0 besitzt außer im Ursprung. Im Rahmen dieser Arbeit wird eine Approximation über die Dichtefunktion der zentrierten Normalverteilung

Algorithm 10 Gewichtung und Aggregation der normalisierten Maßzahlen

```
1: Input: normedVals, W
2: Output: scalar
3: scalar ← 0
4: for i = 0; i < size(normedVals); i++ do
5:     scalar += W(i) ∗ normedVals(i)
```

genutzt:

$$\delta_a(t) = \frac{1}{a\sqrt{\pi}} e^{-x^2/a^2} \tag{3.36}$$

Diese Funktion erreicht ihren Maximalwert im Ursprung. Für $a \to 0$ wird die Funktion immer höher und schmaler, zu sehen in Abbildung 3.20, der Flächeninhalt bleibt jedoch konstant 1. Die für die Aggregation verwendete Variante setzt $a = 0.25$. Ist der Wert von **percentageVal** 1, so erreicht die approximierte Dirac-Delta-Funktion ihr Maximum. Eine Normalisierung über das Maximum bringt den Wert schließlich in ein Intervall von $[0, 1]$.

Eine Anwendung der Gewichte und Bestimmung der aggregierten Maßzahl erfolgt schließlich auf die in Algorithmus 10 beschriebene Variante. Der resultierende Skalar beschreibt schließlich die aus allen Maßzahlen kombinierte Güte des Manövers. Der Maximalwert des Skalars bestimmt sich aus der Summe aller Gewichte. Je nachdem, welche Prioritäten man bei der Bewertung setzt, erhält man ein anderes skalares Maß.

3.5 Realisierung eines Testtools

Im folgenden wird das im Rahmen dieser Arbeit entwickelte Testtool vorgestellt, welche die Ausführung und Bewertung von Manövern basierend auf verschiedenen Fehlereinflüssen ermöglicht.

Die Programmierung dieses Testtools erfolgt in C++. Als Basis für die Kinematik-

Berechnungen wurde die von Christian Eiserloh in seiner Masterarbeit entwickelte Kinematikbibliothek **ezKine** verwendet, welche in Kapitel 2.1.2 vorgestellt wurde. Weitere Software-Bibliotheken umfassen die Template-Bibliothek **Eigen** [Guennebaud u. a., 2010] zur Berechnung und Darstellung von Matrizen und Vektoren sowie die GUI-Bibliothek **Qt4** [Project, 2013] zur Implementierung verschiedener Hilfswerkzeuge, u.a. zur Visualisierung verschiedener Verfahren. Auf letztere wird im Rahmen dieses Kapitels nicht eingegangen.

Die vollständige Dokumentation ist dem Anhang der Ausarbeitung zu entnehmen. Im Rahmen dieses Kapitels wird sich auf die Erläuterung der grundlegenden Funktionsweise der Software beschränkt.

Zunächst werden in Abschnitt 3.5.1 die Anforderungen an das Testtool definiert. Es folgt die Vorstellung der wichtigsten Komponenten der Software in Abschnitt 3.5.2. Die folgenden Abschnitte 3.5.3 bis 3.5.10 beschreiben den Aufbau der einzelnen Komponenten im Detail.

3.5.1 Anforderungen

Die grundlegende Idee des Testtools liegt in der Betrachtung einer Vielzahl verschiedener Fehlerkonfigurationen und deren Auswirkung auf das Manövrieren mithilfe der Fahrerassistenz. Sie soll dabei folgenden Anforderungen genügen:

In Abschnitt 3.2.1 wurden Fehlereinflüsse identifiziert, welche sich auf die Berechnungen der Assistenz-Trajektorien auswirken. Eine Nachbildung dieser Fehler erfordert die Nachbildung des Datenflusses, welchen die von der Assistenz benötigten Werte durchlaufen.

Um die Auswirkungen der Fehler quantifizieren zu können, muss ein Manöver mit den fehlerbehafteten Trajektorien ausgeführt werden. Durch ungenaue Steuerbefehle des menschlichen Nutzers können zusätzliche, unvorhersehbare Fehler auftreten. Das Manöver soll daher mithilfe des in Abschnitt 3.3 entwickelten Controllers ausgeführt

werden.

Die Auswertung eines Manövers soll anhand der Nähe zu Fehlergrenzen erfolgen. Zu diesem Zweck soll die in Abschnitt 3.4 vorgestellte Methode zur Gewichtung von Partikeln genutzt werden.

Um einen möglichst großen Raum verschiedener Fehlerkonfigurationen effektiv abzutasten, soll schließlich der Annealed Particle Filter genutzt werden. Nach jedem Iterationsschritt soll dabei die aktuelle Partikelmenge gespeichert werden.

3.5.2 Komponenten der Software

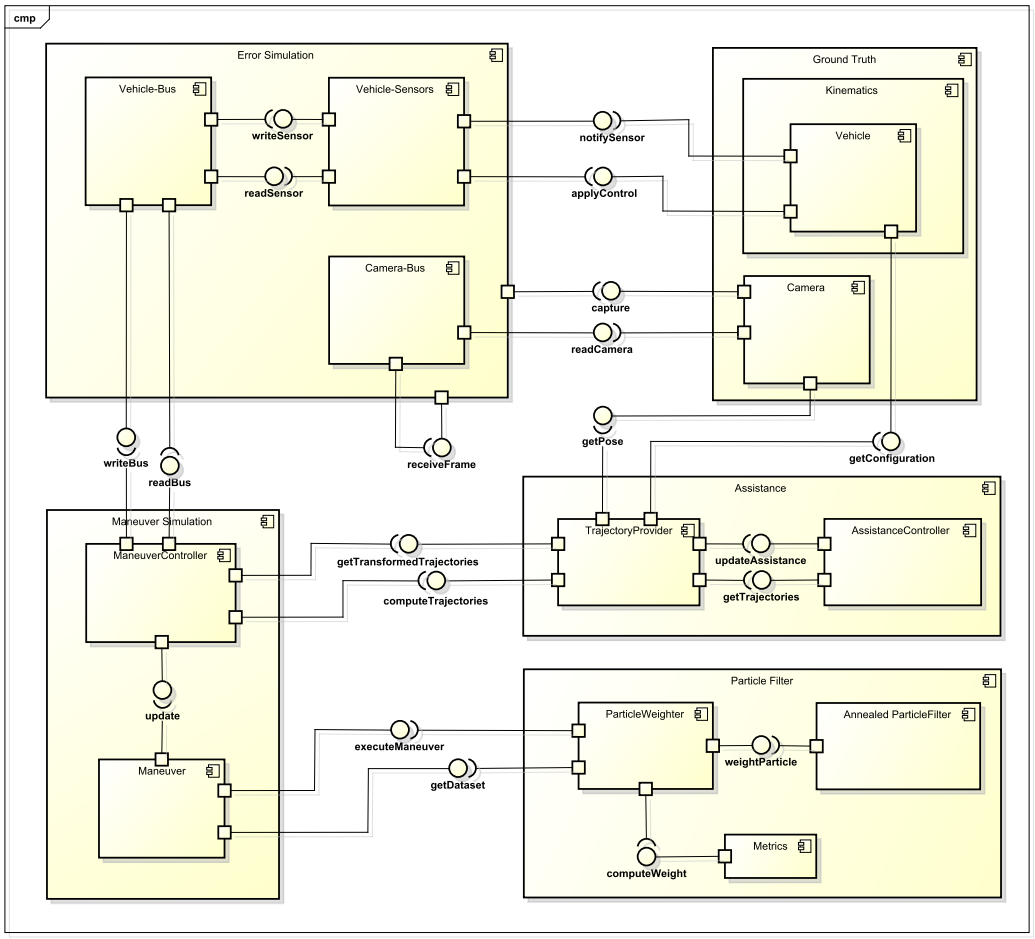


Abbildung 3.21: Die Komponenten der Testumgebung.

Abbildung 3.21 zeigt ein reduziertes Komponentendiagramm, welches die Interaktion und die Abhängigkeiten der verschiedenen Komponenten darstellt. Der Aufbau und die Funktion der Komponenten wird im Folgenden näher betrachtet. Wenn nicht weiter spezifiziert, enthält jede Komponente genau eine Klasse mit identischem Namen.

Komponente Particle Filter

Die Komponente **Particle Filter** enthält eine Implementierung des in Abschnitt 2.3.5 vorgestellten **Annealed Particle Filter**s.

Zur Gewichtung der einzelnen Partikel im **Importance-Sampling**-Schritt steht die Komponente **ParticleWeighter** zur Verfügung. In dieser Komponente erfolgt die Ausführung des entsprechend des Partikels konfigurierten fehlerhaften Manövers. Die Bewertung des Manövers erfolgt anhand der in Abschnitt 3.4.2 vorgestellten Gewichtungsfunktion.

Komponente Maneuver Simulation

Die Komponente **Maneuver Simulation** enthält Strukturen zu Ausführung und Aufzeichnung eines Manövers zwischen einem vordefinierten Start- und Zielpunkt.

Die Sub-Komponente **ManeuverController** enthält eine Implementierung des in Abschnitt 3.3.4 vorgestellten Controller-Algorithmus. In jedem Aktualisierungsschritt empfängt der Controller neue Sensorwerte, aus denen er die von der Assistenz benötigten Winkel extrahiert.

Die Sensorwerte werden von der Komponente **Error Simulation** verschickt und entsprechen einer modifizierten Version der Grundwahrheit.

In jedem Durchlauf werden die auf diese Weise empfangenen Winkel der **Assistance**-Komponente zur Verfügung gestellt, welche daraufhin zu den Winkeln passende

Trajektorien berechnet. Die resultierenden Trajektorien werden abgefragt und bilden die Grundlage für die Generierung eines neuen Steuerbefehls für das Fahrzeug.

Der Steuerbefehl wird schließlich wieder an die **Error Simulation** zurückgeschickt und sorgt für eine Bewegung des Fahrzeugs.

Die Ausführung des Controllers wird gesteuert von der zweiten Sub-Komponente **Maneuver**, welche den Controller in regelmäßigen Abständen aktualisiert. Das Manöver endet, wenn der Zielpunkt bis auf eine tolerierbare Abweichung erreicht wurde, oder wenn sich das Fahrzeug über einen längeren Zeitraum nicht mehr dem Ziel annähert (etwa durch einen verbotenen Fahrzeugzustand).

Der aktuelle Fahrzeugzustand wird nach jedem Ausführungsschritt des Controllers aufgezeichnet. Die vollständige Historie aller Fahrzeugzustände lässt sich nach Beendigung des Manövers vom **ParticleWeighter** auslesen und weiterverarbeiten.

Komponente Assistance

Über die **Assistance**-Komponente lassen sich die Hilfs-Trajektorien der Assistenz in jedem beliebigen Koordinatensystem berechnen. Sie besteht aus der Klasse **AssistanceController**, welche einer Nachbildung in der im Prototyp des realen Fahrerassistenzsystems zum Einsatz kommenden Klasse entspricht, sowie der Proxy-Komponente **TrajectoryProvider**, welche eine Transformation der Trajektorien zwischen verschiedenen Koordinatensystemen ermöglicht.

Die im **AssistanceController** berechneten Trajektorien liegen zunächst im Anhängerkoordinatensystem des Fahrzeugs vor, welches zur Berechnung verwendet wurde. Der Controller jedoch benötigt eine Repräsentation der Trajektorien im Weltkoordinatensystem, welche eine Transformation der Trajektorien mit absoluter Position und Ausrichtung des Zugfahrzeugs erfordert.

Eine Bestimmung des Versatzes durch eine ungenaue Vermessung von Fahrzeug- oder Kamera-Parametern (siehe Abschnitt 3.2.2) erfordert zusätzlich die Transformation

vom Koordinatensystem des fehlerbehafteten Systems in das des fehlerfreien. Dazu greift der **TrajectoryProvider** auf die Grundwahrheiten von Fahrzeug und Kamera zu, welche durch die Komponente **Ground Truth** zur Verfügung gestellt werden.

Anhand der aktuellen Fehlerkonfiguration bestimmt er die fehlerhaften Repräsentationen von Fahrzeug und Kamera und leitet daraus schließlich die benötigten relativen Transformationen her. Die transformierten Trajektorien lassen sich daraufhin vom Controller auslesen.

Komponente Error Simulation

Die Hauptkomponente zur Simulation von Fehlern nennt sich **Error Simulation**. Sie ist eng verbunden mit der Komponente **Ground Truth**, da sie die nötigen Strukturen bereitstellt, um Fehler in der Kommunikation mit der Grundwahrheit zu simulieren.

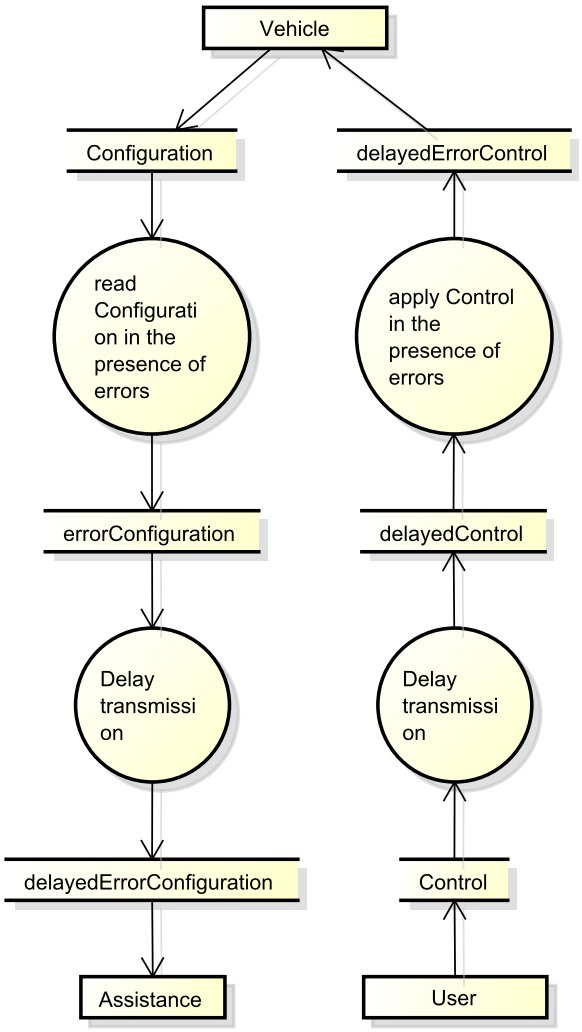


Abbildung 3.22: Das Diagramm zeigt den Datenfluss zwischen Nutzer und Assistenz, welcher über die Komponente **Error Simulation** gesteuert wird. Diese Komponente simuliert den Einfluss verschiedener Fehlerarten, welche auf dem Kommunikationsweg auftreten können.

Die erste modellierte Fehlerkomponente ist die virtuelle Sensorik-Komponente **Vehicle-Sensors**, welche direkt mit dem Fahrzeug verbunden ist. Über diese Sensoren lassen sich der aktuelle Fahrzeugzustand auslesen oder ein Steuerbefehl auf das Fahrzeug anwenden. Sobald ein Sensor ein neues Datum erhält, modifiziert er es anhand vordefinierter Datenfilter, welche dem Wert beispielsweise additives Rauschen hinzufügen, bevor er weitergeleitet wird.

Eine zweite Komponente ist der **Vehicle-Bus**, welcher Strukturen zur zeitverzögerten Weiterleitung von Daten enthält. Diese simulieren den Datenverkehr im realen Fahrzeug, indem sie Daten für eine gewisse Zeit zurückhalten, bevor sie weitergeleitet werden. Diese Datenbusse bilden die Schnittstelle zum zuvor vorgestellten **ManeuverController**, welcher nicht direkt mit dem Fahrzeug kommuniziert, sondern den Umweg über das Bus-System nehmen muss.

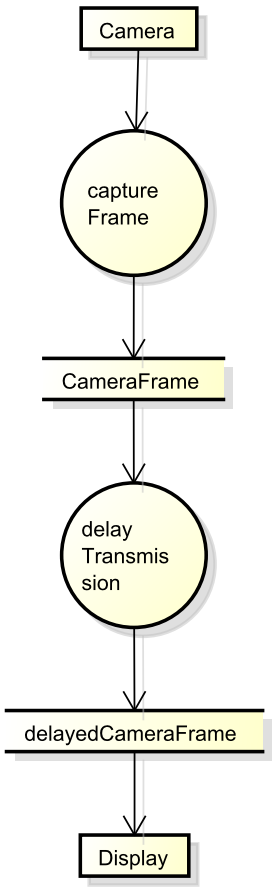


Abbildung 3.23: Das Diagramm zeigt den Datenfluss zwischen Kamera und Display, welcher ebenfalls durch die **Error Simulation** gesteuert wird. Nach der Aufnahme wird das Kamerabild erst mit einer Verzögerung weitergeleitet.

Die **Error Simulation** ist zudem dafür zuständig mithilfe der **Camera**-Komponente in regelmäßigen Abständen ein Kamerabild aufzunehmen und nach einer gewissen Zeitverzögerung als letzten Kameraframe zu speichern. Dafür steht auch in diesem

Fall ein Bus-System zur Verfügung, genannt **Camera-Bus**. Dieser nimmt die letzte Aufnahme der Kamera entgegen und leitet sie nach einer dem Fehler entsprechenden Zeitdauer an die **Error Simulation** weiter. Diese hat somit stets Zugriff auf das zuletzt empfangene Bild der Kamera und kann es für weitere Berechnungen verwenden.

Komponente Ground Truth

Die Grundwahrheit von Fahrzeug und Kamera werden in der Komponente **Ground Truth** gespeichert. Sie dienen als Referenzpunkt für den Vergleich mit fehlerbehafteten Werten. Über die Komponente **Error Simulation** kann ein fehlerbehafteter Zugriff simuliert werden. Benötigt man die exakten Werte, so sollte man direkt auf die Komponente zugreifen.

3.5.3 Die Klasse ErrorConfiguration

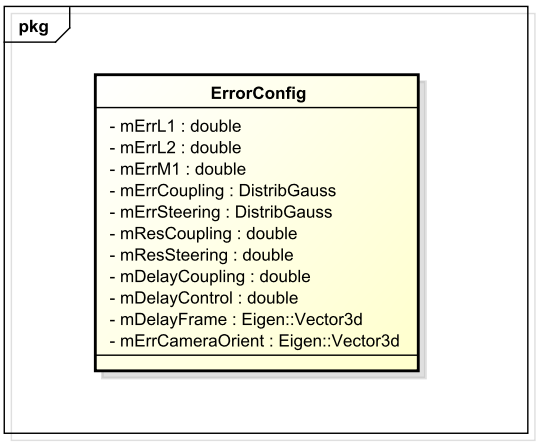


Abbildung 3.24: Die Klasse ErrorConfig.

Die Klasse *ErrorConfig* vereint die Parameter aller in der Modellierung betrachteten Fehler zu einer Fehlerkonfiguration. Die modellierten Fehler umfassen

- **mErrL1**: Der Fehler in *L1* in Metern

- **mErrL2**: Der Fehler in *L2* in Metern
- **mErrM1**: Der Fehler in *M1* in Metern
- **mErrCoupling**: Die Parametrisierung des normalverteilten Rauschens auf dem Einknickwinkel in Rad
- **mErrSteering**: Die Parametrisierung des normalverteilten Rauschens auf dem Lenkwinkel in Rad
- **mResCoupling**: Die Sensorauflösung des Einknickwinkelsensors in Rad
- **mResSteering**: Die Sensorauflösung des (virtuellen) Lenkwinkelsensors in Rad. In der Praxis wird der Lenkradwinkel gemessen. Durch die Annahme einer linearen Übersetzung kann die Auflösung dieses Sensors direkt auf den Lenkwinkel übertragen werden.
- **mDelayCoupling**: Die Verzögerung in der Übertragung des Einknickwinkels in Sekunden
- **mDelayControl**: Die Verzögerung in der Anwendung des nächsten Steuerbefehls. Dies ist ein experimentelles Feld, welches Reaktionsverzögerungen des Menschen simulieren soll. In der Durchführung der Experimente hatte dieses Feld jedoch keine Relevanz.
- **mDelayFrame**: Die Verzögerung in der Übertragung des Kamerabilds in Sekunden
- **mCameraPos**: Die dreidimensionale Position der Kamera im Anhängerkoordinatensystem. Alle Angaben in Metern.
- **mCameraOrient**: Die dreidimensionale Ausrichtung der Kamera im Anhängerkoordinatensystem in Eulerwinkeln. Alle Angaben in Rad.

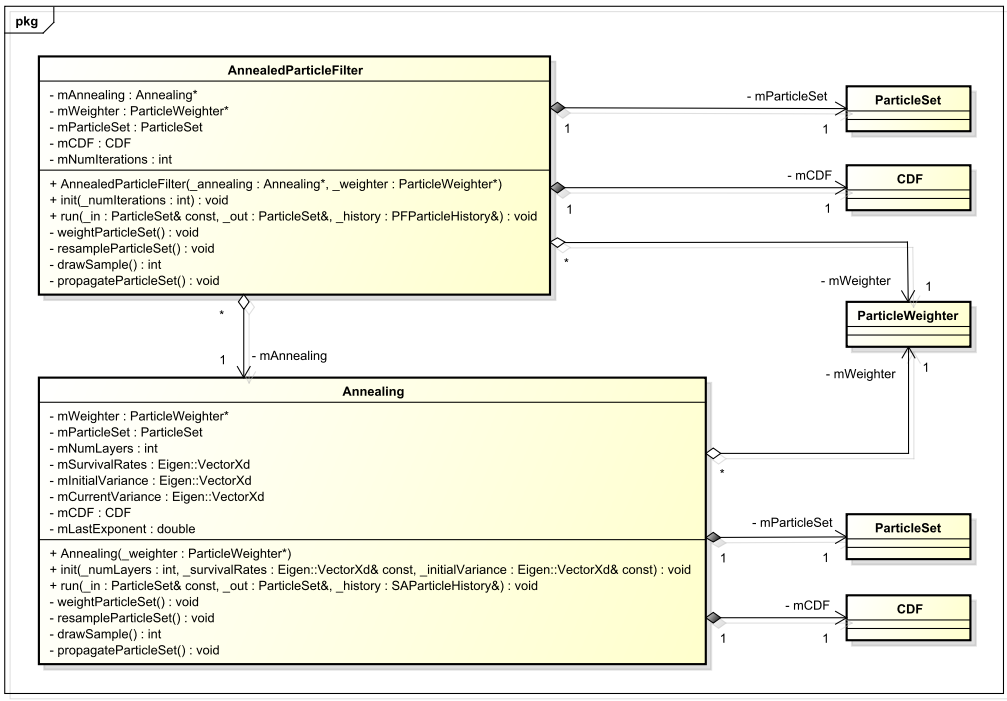


Abbildung 3.25: Die Klassen AnnealedParticleFilter und Annealing.

3.5.4 Die Klassen AnnealedParticleFilter und Annealing

Diese beiden Klassen, zu sehen in Abbildung 3.25, werden gemeinsam betrachtet, da sie sich in vielen Gesichtspunkten ähneln oder komplett übereinstimmen. Jede der beiden Klassen implementiert die drei Schritte eines **Sequential Importance Resampling**-Filters (SIR-Filter), siehe Kapitel 2.3.4, welche den **Importance-Sampling**-Schritt, den **Resampling**-Schritt und den **Diffusions**-Schritt umfassen.

Für jeden Schritt steht ein Hilfsobjekt zur Verfügung, welches die erforderlichen Berechnungen durchführt. Die relevanten Member sind:

- **mParticleSet**: Repräsentiert eine Menge an Partikeln. Ein Partikel entspricht wie zuvor erwähnt einer Fehlerkonfiguration. Diese Klasse erlaubt eine komfortable Verwaltung sowie Berechnungen über die Gesamtheit aller Partikel, etwa die Normalisierung der Gewichte.

- **mParticleWeighter**: Repräsentiert den **Importance-Sampling**-Schritt und wird verwendet, um einen Partikel entsprechend der vorgestellten Methode zu gewichten.
- **mCDF**: Repräsentiert die kumulative Wahrscheinlichkeitsfunktiion, welche im **Resampling**-Schritt verwendet wird. Diese Funktion wird verwendet um Stichproben aus der Partikelmenge zu ziehen entsprechend der vorab bestimmten, normalisierten Wahrscheinlichkeiten aller Partikel.
- **mAnnealing** (nur AnnealedParticleFilter): Repräsentiert den **Diffusions**-Schritt, welcher für eine Bewegung der Partikelmenge im Zustandsraum sorgt. In der **AnnealedParticleFilter**-Klasse entspricht dieser Schritt einem Durchlauf der simulierten Abkühlung, in der **Annealing**-Klasse selbst erfolgt in diesem Schritt die Addition einer multivariaten-normalverteilten Zufallsvariablen auf die Partikel.

Nach jedem Iterationsschritt bzw. Layer von **AnnealedParticleFilter**- bzw. **Annealing**-Klasse erfolgt eine Speicherung der derzeitigen Partikelmenge, sodass das Verfahren jederzeit unterbrochen und wieder aufgenommen werden kann.

3.5.5 Die Klasse Maneuver

Die Klasse **Maneuver** ist die zentrale Klasse zur Steuerung und Verwaltung der Durchführung eines Manövers, zu sehen in Abbildung 3.26. Statische Parameter dieser Klasse umfassen:

- **mStartConfig**: Der initiale Fahrzeugzustand, von dem aus das Manöver begonnen wird.
- **mGoalPosition**: Der anvisierte Zielpunkt in Weltkoordinaten.
- **mDriverReactionTime**: Das Zeitintervall zwischen zwei konsekutiven Entscheidungen des Fahrers. Der Entscheidungsprozess, welchen Lenkwinkel der

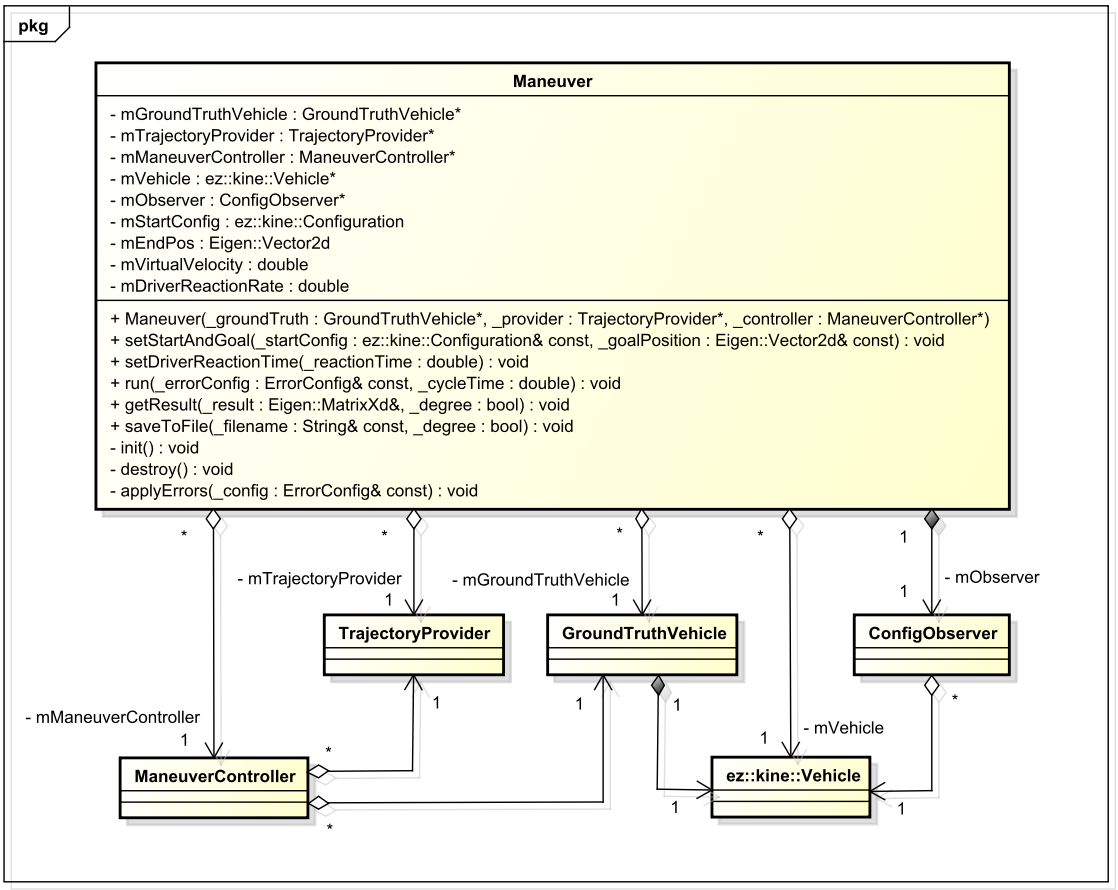


Abbildung 3.26: Die Klasse Maneuver.

Fahrer einstellen will, wird hier als nicht kontinuierlich vorausgesetzt und in der Software als diskrete Zeitschritte modelliert.

- **mTimeScalingFactor**: Ein Faktor, welcher die Beschleunigung der Ausführungsrate zweier nachfolgender Zeitschritte ermöglicht. Ein Wert größer als 1 bedeutet eine langsamere Ausführung, ein Wert kleiner als 1 entsprechend eine schnellere Ausführung, ein Wert von exakt 1 bedeutet eine Ausführung in Echtzeit.

Die folgenden dynamischen Objekte werden zur Ausführung des Manövers genutzt:

- **mGroundTruthVehicle**: Eine Referenz auf die Grundwahrheit von Fahrzeug

und Kamera, welche es erlaubt die Parameter der Fahrzeugkommunikation aus der **Error Simulation** mithilfe einer Fehlerkonfiguration zu aktualisieren.

- **mTrajectoryProvider**: Eine Referenz auf die Komponente, welche die Berechnung der Trajektorien steuert. Dieser kann ebenfalls eine Fehlerkonfiguration übergeben werden, was die internen Transformationen zwischen fehlerbehafteten und fehlerfreien Koordinatensystemen aktualisiert.
- **mManeuverController**: Eine Referenz auf den Controller zur Steuerung des Manövers. Die Klasse **Maneuver** verwaltet die Ausführung einzelner Schritte des Controllers. Das Zeitintervall zwischen den Schritten wird durch das Produkt von Reaktionsintervall des Fahrers und Beschleunigungsfaktor festgelegt: **mDriverReaction** $\times$ **mTimeScalingFactor**.
- **mObserver**: Eine Logger-Klasse, welche über das **Observer**-Pattern mit dem Fahrzeug der GroundTruthVehicle-Klasse verbunden ist. Ändert sich der Fahrzeugzustand, wird der neue Zustand ausgelesen und gespeichert. Die vollständige Historie kann später über eine Anfrage an die **Maneuver**-Klasse ausgelesen werden.

Die **Maneuver**-Klasse wurde so konzipiert, dass man sie ohne weiteres mehrfach hintereinander mit verschiedenen Fehlerkonfigurationen ausführen kann.

3.5.6 Die Klasse ManeuverController

Die abstrakte Klasse **ManeuverController** enthält ein Codegerüst für verschiedene Arten von Controllern, welche die Assistenz-Trajektorien zur Bestimmung von Steuerbefehlen benutzen. Abbildung 3.27 zeigt die Klasse zusammen mit der Unterklasse **VehicleManeuverController**, welche schließlich die Implementierung des in Abschnitt 3.3.4 vorgestellten Algorithmus enthält. Folgende statische Datenfelder können in der Superklasse gesetzt werden:

- **mStartConfig**, **mGoalPosition** und **mDriverReactionTime**: Analog zur

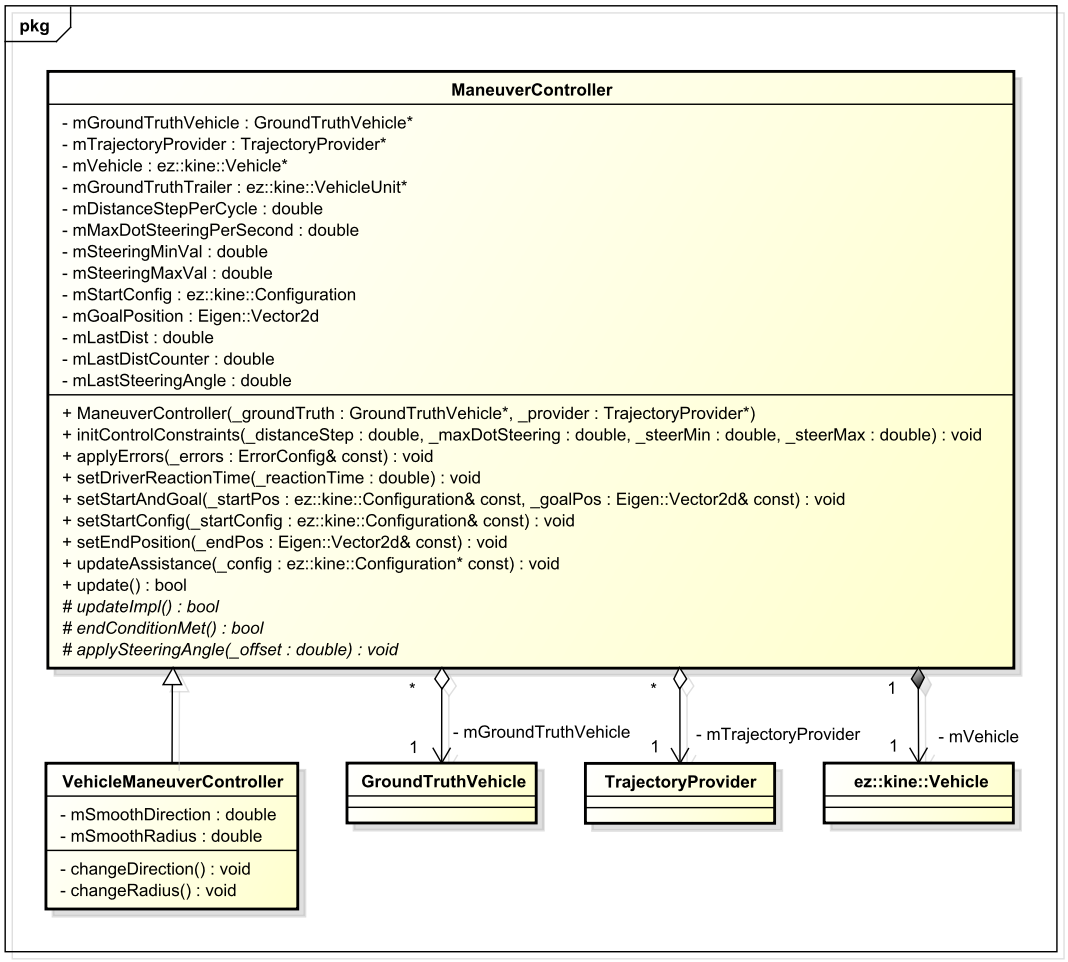


Abbildung 3.27: Die Klassen ManeuverController und VehicleManeuverController.

Maneuver-Klasse.

- **mDistanceStepPerCycle**: Die in jeder Iteration zurückgelegte Strecke. Dies ist der konstante Teil des Steuerbefehls, welcher in jedem Durchlauf auf das Fahrzeug angewendet wird, wobei der neu berechnete Lenkwinkel den variablen Teil darstellt.

- **mMaxDotSteeringPerSecond**: Entspricht der maximalen Lenkwinkeländerung pro Sekunde. Da ein Iterationsschritt genau einem Zeitintervall von **mDriverReactionTime** entspricht, erhält man die Lenkwinkeländerung pro Ite-

rationsschritt durch **mMaxDotSteeringPerSecond** × **mDriverReactionTime**. Der neu berechnete Lenkwinkel darf nicht mehr als dieser Wert vom Lenkwinkel des vorangegangenen Schritts abweichen. Ansonsten wird der neue Lenkwinkel auf den maximalen Wert gesetzt, welcher durch die Beschränkung erreichbar ist.

- **mSteeringMinVal**: Der minimale Wert, den der Lenkwinkel annehmen darf, sei es durch physikalische Beschränkung oder aus anderen Gründen.
- **mSteeringMaxVal**: Der analog maximale Wert, den der Lenkwinkel annehmen darf.

Neben allgemeinen Informationen über das Manöver benötigt der in **VehicleManeuverController** implementierte Algorithmus zudem Informationen über die Assistenz-Trajektorien, möglichst in absoluten Koordinaten, sowie Information über den tatsächlichen derzeitigen Fahrzeugzustand. Diese Informationen werden über die dynamischen Parameter der Klasse geliefert:

- **mGroundTruthVehicle**: Dient als Referenz für die Grundwahrheit, aus der alle Positionen und Ausrichtungen einzelner Fahrzeuggliedern fehlerfrei ausgelesen werden können.
- **mTrajectoryProvider**: Liefert die fehlerhaften Assistenz-Trajektorien in absoluten Koordinaten. Nach jedem Durchlauf des **ManeuverController**s werden aktualisierte (fehlerhafte) Einknick- und Lenkwinkelwerte empfangen, welche sogleich an die Trajektorienberechnung weitergegeben werden.

Am Ende jedes Zyklus wird entweder ein neues Steuerkommando mit dem berechneten Lenkwinkel an das Fahrzeug geschickt oder das Manöver wird beendet. Das Manöver wird erst dann beendet, wenn sich das Fahrzeug in einer vorab definierten Umgebung des Ziels aufhält und es mindestens einmal passiert hat. Alternativ wird das Manöver beendet, wenn sich das Fahrzeug über 10 Iterationen in Folge nicht mehr dem Ziel nähert. Dieser Fall kann auftreten, wenn sich das Fahrzeug in einem

ungültigen Zustand befindet, aus dem es sich ohne vorzusetzen nicht mehr befreien kann, oder wenn des Fahrzeug das Ziel überschießt und eine durch den Controller nicht mehr korrigierbare Abweichung zum Ziel aufbaut.

3.5.7 Die Klasse TrajectoryProvider

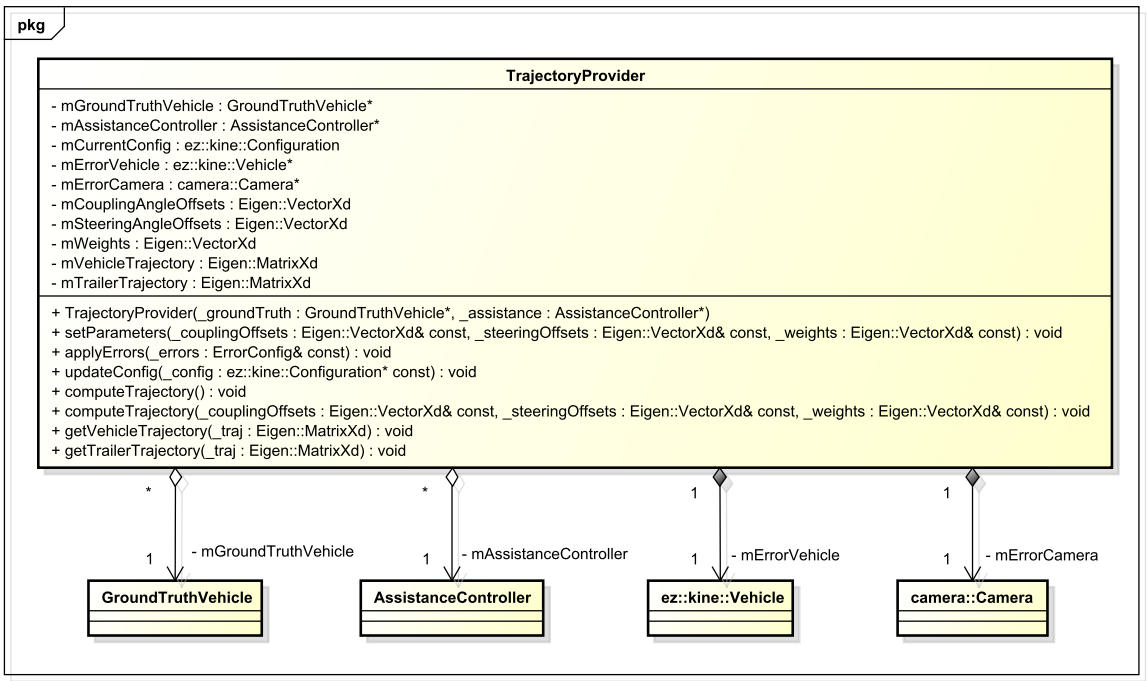


Abbildung 3.28: Die Klasse TrajectoryProvider.

Die in Abbildung 3.28 zu sehende Klasse **TrajectoryProvider** vereinfacht den Zugriff auf den **AssistanceController** im Sinne des **Facade**-Patterns. Sie hat drei grundlegende Funktionalitäten, welche im Folgenden näher erläutert werden.

Die Hauptaufgabe liegt in der Berechnung der Assistenz-Trajektorien. Folgende Datenmember stehen zu diesem Zweck zur Verfügung:

- **mAssistanceController**: Eine Referenz auf die Hauptklasse der Trajektorien-Berechnung wie sie auch (bis auf einige API-Anpassungen) im Prototyp des

Fahrerassistenzsystems gebraucht wird. Die Trajektorien, die diese Klasse produziert, liegen im Anhänger-Koordinatensystem vor.

- **mCurrentConfig**: Der Startzustand des Fahrzeugs, von dem aus die Trajektorienberechnung stattfindet.

Um die Auswirkung von Fehlern im Anfangszustand zu analysieren, steht dem Nutzer die Möglichkeit offen, gemittelte Assistenz-Trajektorien zu berechnen. Eine Anwendung findet sich beispielsweise in der Unscented Transform. Gesteuert wird das Verfahren über die folgenden Felder:

- **mCouplingAngleOffsets**: Die Fehler auf dem Einknickwinkel, welche betrachtet werden sollen.
- **mSteeringAngleOffsets**: Die Fehler auf dem Lenkwinkel, welche betrachtet werden sollen.
- **mWeights**: Gibt die Gewichtung der einzelnen Trajektorien vor. Die Summe der Gewichte muss 1 ergeben.

Diese drei Vektoren müssen dieselbe Größe besitzen. Die Anzahl der Elemente in jedem Vektor gibt vor, über wie viele Trajektorien gemittelt werden soll.

Nachdem die (gemittelten) Trajektorien berechnet wurden, liegen sie in den Datenfeldern **mVehicleTrajectory** und **mTrailerTrajectory** vor. Erfolgt eine Anfrage an diese Datenfelder, so werden zunächst diverse Transformationen angewendet, um die Trajektorien in Weltkoordinaten zu überführen. Die folgenden Datenfelder steuern die Transformationen:

- **mGroundTruthVehicle**: Dieses Objekt erlaubt den Zugriff auf die Grundwahrheiten von Fahrzeug und Kamera. Aus den Pose-Informationen lassen sich so zwei Koordinatensysteme ableiten, welche als Ziel-Koordinatensysteme der folgenden Transformationen dienen.
- **mErrorVehicle**: Ein anhand einer Fehlerkonfiguration erzeugtes, fehlerbehaf-

tetes Fahrzeug-Objekt. Dieses Fahrzeug wird im **AssistanceController** zur Trajektorien-Berechnung genutzt. Die Trajektorien liegen im Koordinatensystem des Anhängers dieses Fahrzeugs vor. Die erste Transformation erfolgt daher von diesem Koordinatensystem ins Anhänger-Koordinatensystem der Grundwahrheit.

- **mErrorCamera**: Ein anhand der Fehlerkonfiguration erzeugtes, fehlerbehaftetes Kamera-Objekt. Die Pose dieses Objekts bestimmt, aus welchem Blickwinkel die Trajektorien wahrgenommen werden. Ist die angenommene Pose fehlerhaft, so ist auch der Blickwinkel fehlerhaft. Die zweite Transformation erfolgt daher vom fehlerhaften Kamera-Koordinatensystem ins entsprechende Koordinatensystem der Grundwahrheit.

Das Vorgehen bei der Transformation zwischen verschiedenen Koordinatensystemen basiert dabei auf der in Abschnitt 3.2.3 vorgestellten Methode.

3.5.8 Modifikationen der Klasse Vehicle

Die ursprünglich geplante Nutzung der Bibliothek *ezKine* sah es vor, dass Fahrzeuge von Hand konstruiert werden müssen. Das heißt, dass nacheinander Zugfahrzeug und Anhänger durch die in Kapitel 2.1.2 vorgestellten Konstruktionsmethoden hinzugefügt werden mussten. Eine Neuerung besteht in der Nutzung des Builder-Patterns zur Erstellung eines komplexen Fahrzeugs auf Basis einer Fahrzeugbeschreibung. Die original implementierte Klasse **VehicleBuilder** enthält eine statische Methode, welche das Fahrzeug aus den übergebenen Fahrzeugparametern zusammenbaut. Diese Klasse wurde im Verlauf der Ausarbeitung durch die nahezu äquivalente Klasse **VehicleFactory** eines Projektmitarbeiters ersetzt, um Konsistenz innerhalb der Arbeitsgruppe zu wahren.

Über ein Observer-Pattern wurde der Klasse **Vehicle** zudem im Laufe der Arbeit Funktionalität hinzugefügt, welche es externen Klassen ermöglicht, direkt über Ände-

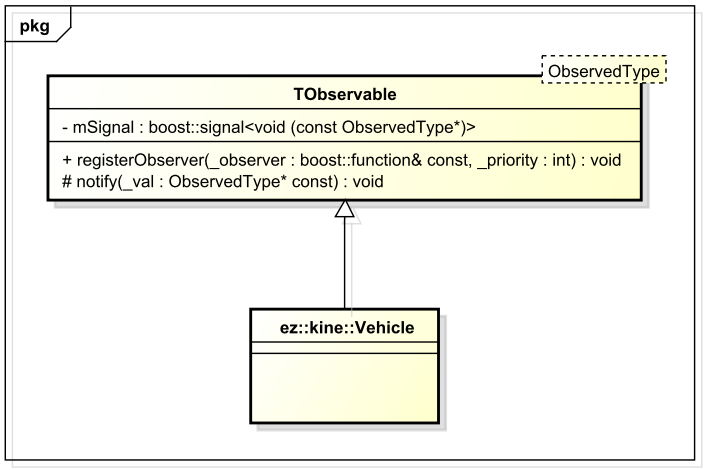


Abbildung 3.29: Die Template-Klasse TObservable.

rungen des Fahrzeugzustands informiert zu werden. Die Grundlage für das Observer-Pattern bildet die Template-Klasse **TObservable**, welche in Abbildung 3.29 zu sehen ist. Unterklassen, welche von **TObservable** erben, erlauben es beliebigen Objekten, sich über die Bereitstellung einer Callback-Funktion als Beobachter zu registrieren. Jederzeit kann in der Unterklasse die Methode **notify** aufgerufen werden, um alle Beobachter über Änderungen am derzeitigen Zustand der Klasse zu informieren. Über den Template-Parameter kann ein Objekt-Typ festgelegt werden, welcher zusätzlich bei der Benachrichtigung übergeben wird. Im Falle der **Vehicle**-Klasse werden zusätzlich Zeitstempel und die durch den letzte Steuerbefehl zurückgelegte Distanz übergeben.

3.5.9 Die Klasse Camera

Die Klasse **Camera** repräsentiert die am Fahrzeug angebrachte Rückfahrkamera, siehe Abbildung 3.30. Sie verwaltet Pose-Informationen und erlaubt es Kamerabilder aufzunehmen und mittels des Observer-Patterns weiterzuleiten. Die folgenden Datenfelder können gesetzt werden:

- **mPosition**: Die dreidimensionale Position der Kamera im Koordinatensystem

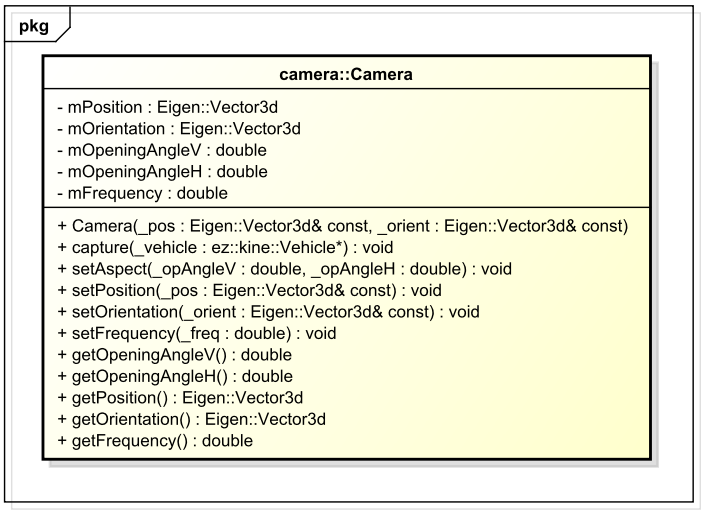


Abbildung 3.30: Die Klasse Camera.

des Anhängers

- **mOrientation**: Die Rotation in Euler-Winkeln um die drei Koordinatenachsen
- **mOpeningAngleV**: Der vertikale Öffnungswinkel der Kamera
- **mOpeningAngleH**: Der horizontale Öffnungswinkel der Kamera
- **mFrequency**: Die Frequenz, mit der Bilder aufgenommen werden

Zusätzlich steht eine Methode namens **capture** zur Verfügung, über die der Befehl zur Aufnahme eines Kamerabilds gegeben werden kann. Wie oft ein Bild aufgenommen werden kann, wird durch die Frequenz beschränkt. Nimmt man die Umgebung als gegeben und statisch an, so kann die Information eines Kamerabildes in der Position und Ausrichtung des Anhängers zum Zeitpunkt der Aufnahme kodiert werden. Mit jedem **capture**-Aufruf wird daher die aktuelle Anhängerpose ausgelesen und an alle Beobachter verschickt.
Analog zur **VehicleFactory** existiert eine Klasse namens **CameraFactory**, welche aus einer Kamerabeschreibung ein **Camera**-Objekt erzeugt.

3.5.10 Die Klasse GroundTruthVehicle

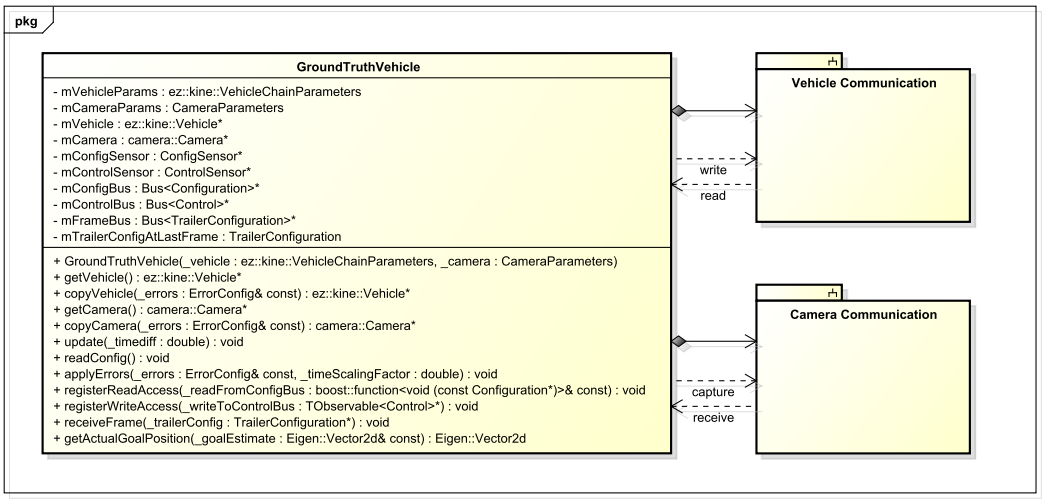


Abbildung 3.31: Die Klasse GroundTruthVehicle.

Abbildung 3.31 zeigt die Klasse **GroundTruthVehicle**, welche die beiden Komponenten **Error Simulation** und **Ground Truth** in einer Klasse vereint.

Diese Klasse verwaltet einerseits die Grundwahrheiten für Fahrzeug und Kamera, andererseits ermöglicht es eine Simulation der fehlerbehafteten Kommunikation zwischen Nutzer und Grundwahrheit. Eine solche Simulation erfordert Strukturen, welche es möglich machen, Werte anhand eines Fehlermodells zu modifizieren oder künstlichen Verzögerungen in der Weiterleitung zu unterziehen. Diese Strukturen finden sich in den beiden Subsystemen **Vehicle Communication** und **Camera Communication**.

Im Folgenden werden zunächst zwei Klassen betrachtet, auf denen diese Strukturen aufbauen, die Klasse **Sensor**, welche die Sensorik im realen Fahrzeug simuliert, sowie die Klasse **Bus**, welche die Datenkommunikation im Fahrzeug simuliert. Danach wird auf die eigentliche Funktionalität der Subsysteme eingegangen.

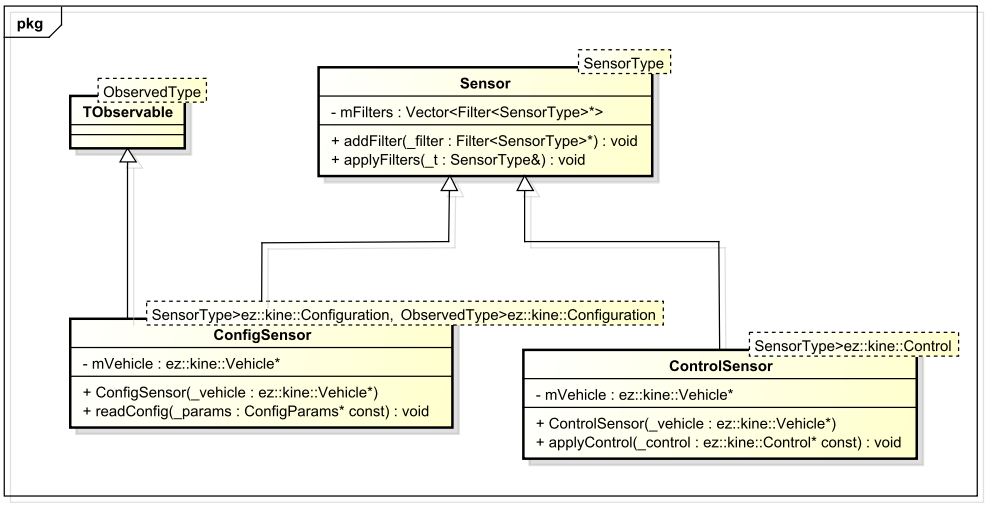


Abbildung 3.32: Die Template-Klasse Sensor.

Die Klasse Sensor

Abbildung 3.32 zeigt die Template-Klasse **Sensor**, welche ein **Proxy**-Objekt für die Klasse **Vehicle** darstellt. Sie steuert, auf welche Art und Weise Datenfelder gesetzt oder ausgelesen werden können. Der Template-Typ gibt den Typ des betrachteten Datenfelds an. Die folgenden Unterklassen werden in der Folge betrachtet:

- **ConfigSensor**: Diese Klasse liest den aktuellen Zustand **mCurrentConfiguration** des Fahrzeugs aus und modifiziert ihn anhand des Fehlermodells, bevor er ihn über das **Observer**-Pattern an alle Beobachter weiterleitet.
- **ControlSensor**: Diese Klasse bekommt einen Steuerbefehl übergeben, modifiziert diesen ebenfalls entsprechend des Fehlermodells, und wendet den Befehl schließlich über **applyControl** auf das Fahrzeug an.

Diese beiden Abstraktionen erlauben die Modellierung von Fehlern im Einknickwinkel, dessen fehlerhafte Repräsentation über den **ConfigSensor** an die Assistenz geliefert wird, sowie im Lenkwinkel, da der **ControlSensor** die Unsicherheit in der Einstellung der Anhänger-Trajektorie simuliert.

Zur Anwendung verschiedener Modifikationen auf die übergebenen Werte steht die abstrakte Klasse **AbstractFilter** zur Verfügung. Einem Sensor-Objekt können eine beliebige Anzahl an Filter-Objekten übergeben werden, welche jeweils einen Teil des Fehlermodells repräsentieren. Die entsprechend Abschnitt 3.2.1 betrachteten Fehler für die Sensorik wurden in den folgenden Filterklassen implementiert:

- **Gauss-Filter**: Fügt dem Sensorwert additives Rauschen hinzu
- **Resolution-Filter**: Beschränkt die Genauigkeit des Sensorwertes auf eine vordefinierte Auflösung

Der erste Filter modelliert die Präsenz von weißem Rauschen in realer Sensorik. Sei x der fehlerfreie Wert. Dann ergibt sich der verrauschte Sensorwert aus:

$$x' = x + \epsilon \tag{3.37}$$

wobei $\epsilon \sim \mathcal{N}(\mu, \sigma)$ eine normalverteilte Zufallsvariable beschreibt.

Der zweite Filter modelliert die limitierte Auflösung eines Sensors. Sei x erneut der fehlerfreie Wert. Sei r zudem die Auflösung des Sensors, d.h. der minimale Abstand, der zwischen zwei Sensorwerten gemessen werden kann. Dann gilt für den zurückgelieferten Sensorwert:

$$toFloor = x \ \% \ r \tag{3.38}$$

$$toRoof = r - toFloor \tag{3.39}$$

$$x' = \begin{cases} x - toFloor, & \text{falls } toFloor < toRoof \\ x + toRoof, & \text{sonst} \end{cases} \tag{3.40}$$

Der zurückgelieferte Sensorwerte ist eine auf das nächste ganzzahlige Vielfache der Auflösung gerundete Repräsentation des tatsächlichen Wertes.

Die Filter-Implementierungen für die beiden Sensor-Unterklassen unterscheiden sich lediglich im Sensortyp, der modifiziert wird.

Eine Besonderheit der **ControlSensor**-Implementierung des **Resolution-Filter** ist, dass nicht nur eine Rundung des Wertes stattfindet, sondern zusätzlich ein gleichverteilter Offset $\delta \in [-\frac{r}{2}, \frac{r}{2}]$ aufaddiert wird. Der Hintergrund ist, dass der Nutzer über feiner als r aufgelöste Winkel kein Feedback aus der Assistenz bekommt. Zwischen einem Lenkwinkel von 10.1° und 10.4° sieht er in der Assistenz keinen Unterschied, wenn der Lenkwinkelsensor lediglich auf 1° aufgelöste Werte liefern kann. Er hat folglich einen Handlungsspielraum von 1°, bevor er eine Veränderung in der Anhänger-Trajektorie bemerkt. Dieser Spielraum wird durch die gleichverteilte Zufallsvariable simuliert.

Die Klasse Bus

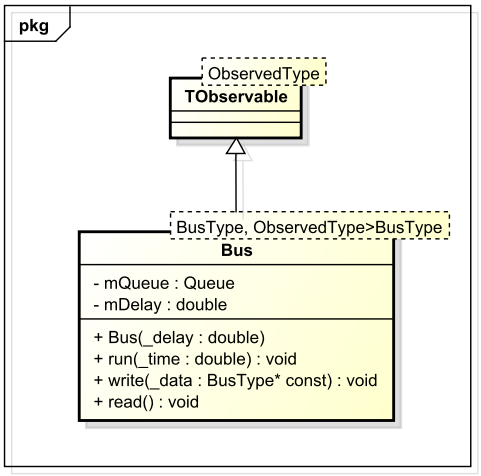


Abbildung 3.33: Die Template-Klasse Bus.

Abbildung 3.33 zeigt die Template-Klasse **Bus**, welche einen Datenbus im realen Fahrzeug modelliert, der (Sensor-)Daten lediglich mit einer gewissen Verzögerung weiterleitet. Über den Template-Parameter wird der Datentyp der zu übertragenen Werte festgelegt. Die Weiterleitung der Werte wird dabei über zwei Methoden gesteuert:

- **write**: Legt ein Objekt des Template-Typs auf den **Bus**. Das Objekt wird in die

interne Schlange **mQueue** eingereiht und bekommt einen Timer zugewiesen. Die Zeit, bis der Timer eines Objekts abläuft, richtet sich nach der Verzögerung **mDelay**, mit der der Bus initialisiert wurde.

- **read**: Nach Ablauf eines Timers wird diese Methode aufgerufen, welche das zu übertragende Objekt über das **Observer**-Pattern an alle Beobachter des Busses versendet.

Es bedarf dabei einer regelmäßigen Aktualisierung aller Timer der Queue (**run**), um sicherzustellen, dass sie rechtzeitig ablaufen und die Werte schließlich weitergeleitet werden.

Entwurf einer Struktur zur Kommunikation mit dem Fahrzeug

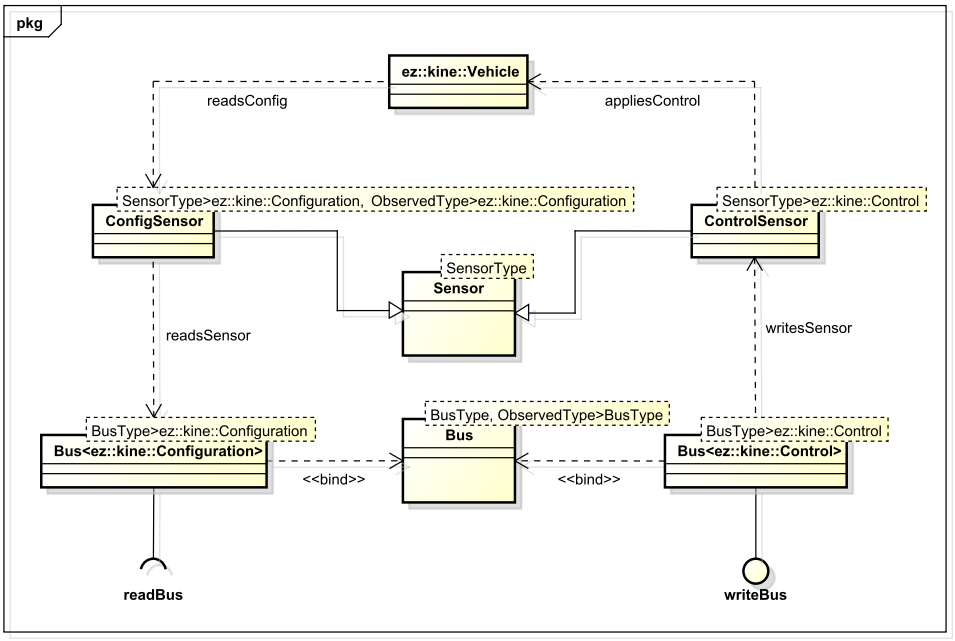


Abbildung 3.34: Das Subsystem zur Kommunikation zwischen Nutzer und Fahrzeug.

Das Subsystem zur Kommunikation zwischen Nutzer und Fahrzeug ist in Abbildung 3.34 zu sehen. Dem Nutzer steht es nun nicht mehr frei, direkt mit dem Fahrzeug zu kommunizieren, er kennt nur noch das Konzept des Datenbusses, auf den er Da-

ten schreibt oder von dem er Daten empfängt. Er braucht keine Kenntnis davon zu besitzen, woher die Daten kommen oder wohin sie versendet werden.

Will der Nutzer eine Bewegung des Fahrzeugs initiieren, so legt er einen Steuerbefehl (**Control**) auf den **Control-Bus**. Über den **Config-Bus** erhält er als Antwort den aktualisierten Fahrzeugzustand (**Configuration**). Über eine Fehlerkonfiguration lässt sich die Größenordnung der auf Sensorik und Datenbussen liegenden Fehler und damit die Eigenschaften der Kommunikation steuern.

Die Verbindung der einzelnen Objekte erfolgt erneut über das Observer-Pattern wie in den vorangegangenen Kapiteln dargestellt.

Entwurf einer Struktur zur Kommunikation mit der Kamera

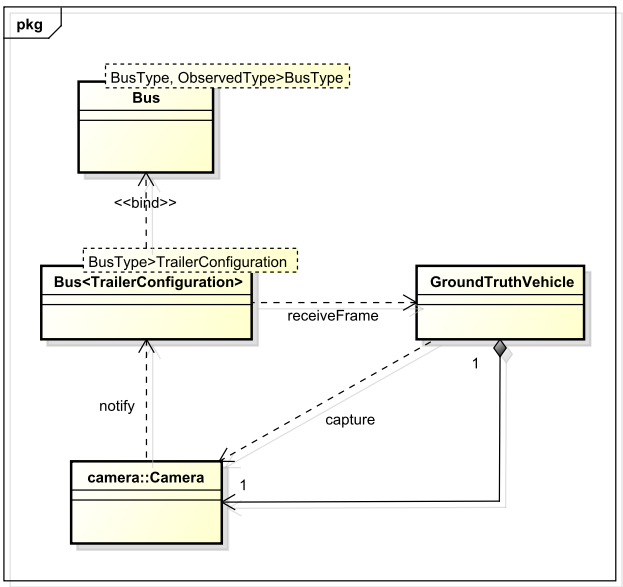


Abbildung 3.35: Das Subsystem zur Kommunikation zwischen Fahrzeug und Kamera.

Aus Abschnitt 3.2.3 ist bekannt, dass sich ein Kamerabild über eine Anhänger-Pose repräsentieren lässt. Um die zeitliche Verzögerung zwischen Aufnahme und Anzeige eines Kamerabildes zu simulieren, wird ebenfalls eine **Bus**-Instanz verwendet, zu sehen in Abbildung 3.35.

Aus der Klasse **GroundTruthVehicle** heraus erfolgt zunächst der Aufruf der **capture**-Methode des Kameraobjekts. Daraufhin wird die ausgelesene Anhänger-Pose über den Datenbus wieder zurück geleitet an das **GroundTruthVehicle**, welches die jeweils zuletzt empfangene Anhänger-Pose zwischenspeichert.

Schnittstellen nach außen

Die oben beschriebenen Strukturen werden automatisch erstellt, sobald das **GroundTruthVehicle** initialisiert wird. Externen Modulen werden die folgenden Schnittstellen zur Verfügung gestellt, um mit den Grundwahrheiten zu kommunizieren:

- **copyVehicle**: Erzeugt eine fehlerbehaftete Kopie der Fahrzeug-Grundwahrheit auf Basis einer Fehlerkonfiguration.
- **copyCamera**: Erzeugt eine fehlerbehaftete Kopie der Kamera-Grundwahrheit auf Basis einer Fehlerkonfiguration.
- **applyErrors**: Aktualisiert die Fehlerwerte aller internen Komponenten auf Basis einer Fehlerkonfiguration.
- **registerWriteAccess**: Ermöglicht die Initiierung von Bewegungen des Fahrzeugs.
- **registerReadAccess**: Ermöglicht die automatische Benachrichtigung über Änderungen im Fahrzeugzustand.
- **getActualGoalPosition**: Berechnet die absolute Position eines Punkts im zuletzt empfangenen Kamerabild, siehe Abschnitt 3.2.3.

Zusätzlich stehen Getter zur Verfügung, über die man direkt auf die Grundwahrheiten zugreifen kann. Damit stellt die Klasse **GroundTruthVehicle** einen Fixpunkt dar im Zugriff auf Fahrzeug und Kamera, sei es fehlerbehaftet oder fehlerfrei.

3.6 Portierung von Kinematik und Assistenz auf RODOS®

Das folgende Kapitel widmet sich der Vorstellung der im Rahmen der Zusammenarbeit mit dem Fraunhofer ITWM entwickelten Softwarekomponenten.

In Abschnitt 3.6.1 erfolgt zunächst eine Analyse bestehender Schnittstellen, um die benötigten Komponenten im System einzubinden. In Abschnitt 3.6.2 werden die Anforderungen an die Software definiert. Die Vorstellung des Gesamtkonzepts zur Einbindung der Assistenz in RODOS® wird in Abschnitt 3.6.3 erläutert. Das kinematische Fahrzeugmodell wird danach in Abschnitt 3.6.4 vorgestellt. Einem kurzen Abschnitt 3.6.5 zum Logging folgt eine Beschreibung der Sotwarekomponente, welche die Kommunikation zwischen Assistenz und der Simulationsumgebung steuert, in Abschnitt 3.6.6. Den Abschluss bildet Abschnitt 3.6.7, welches eine Bewertung der Komponenten sowie des Gesamtsystems enthält.

3.6.1 Analyse bestehender Schnittstellen

Wie in Abschnitt 2.4.2 erläutert wurde, folgt das RODOS-System einem modularen Aufbau. Die einzelnen Komponenten kommunizieren untereinander über UDP-Sockets. Dadurch ist es möglich, ohne großen Aufwand weitere Komponenten hinzuzufügen. Die einzige Voraussetzung ist die Bereitstellung einer Netzwerkschnittstelle. Dabei muss darauf geachtet werden, dass nur die nötigsten Daten zwischen Komponenten kommuniziert werden. So wird sichergestellt, dass der Netzwerkverkehr gering gehalten wird und das System echtzeitfähig bleibt.

Auf einem Echtzeit-Rechner (**dSpace**) des in Abschnitt 2.4.2 beschriebenen Software-Layer läuft das Fahrzeugmodell, welches in regelmäßigen Abständen den Fahrzeugzustand verschickt. Die in diesem Fall genutzte Simulationssoftware ist MATLAB Simulink®. Der Fahrzeugzustand liegt dabei zunächst als Vektor von Float-Werten vor. Die Kommunikation mit anderen Komponenten des Systems erfolgt über den Umweg eines Python-Skripts, welches eine Serialisierung des empfangenen Fahrzeug-

zustands vornimmt. Dies erlaubt die Standardisierung der Schnittstelle zwischen dem Fahrzeugmodell und allen Komponenten, welche auf die Ausgabedaten des Fahrzeugmodells angewiesen sind, etwa der Render-Engine.

3.6.2 Anforderungen an die Software

Die folgenden Anforderungen werden an das System gestellt:

Zunächst soll eine Portierung der 3D-Modelle der Arbeitsgruppe auf die Unity3d-Engine erfolgen, welche vom Fraunhofer ITWM als Grafik-Engine genutzt wird. Zur Verfügung stehen das Modell eines Aufliegers mit Anhänger sowie zwei Modelle eines Mercedes Actros-LKW mit Einachs- und Zweiachsanhänger.

Daraufhin muss ein Fahrzeugmodell in Simulink entwickelt werden. Das entwickelte Modell soll dabei auf **ezKine** basieren. Zusätzlich soll es möglich sein, jeden Fahrzeugzustand einem eindeutigen Zeitpunkt zuordnen zu können.

Es soll möglich sein Fahrzeugzustände mit den zugehörigen Zeitstempeln in einem Logfile zu sichern.

Die Assistenz soll als Blackbox auf einem externen Rechner in der Fahrerkabine laufen und über UDP-Sockets mit RODOS kommunizieren. Es soll daher eine Netzwerkschnittstelle implementiert werden, welche die Weiterleitung von Daten des Fahrzeugmodells an das Fahrerassistenz-System ermöglicht. Im Speziellen soll es möglich sein, UDP-Pakete in ezCom-Nachrichten zu konvertieren.

Die Assistenz benötigt neben den Fahrzeugdaten die Rückfahransicht einer am Heck des Anhängers angebrachten Kamera. Es muss daher eine Möglichkeit gefunden werden, das in der Render-Engine zur Verfügung gestellte Kamerabild abzugreifen und schließlich an den Assistenzrechner weiterzuleiten.

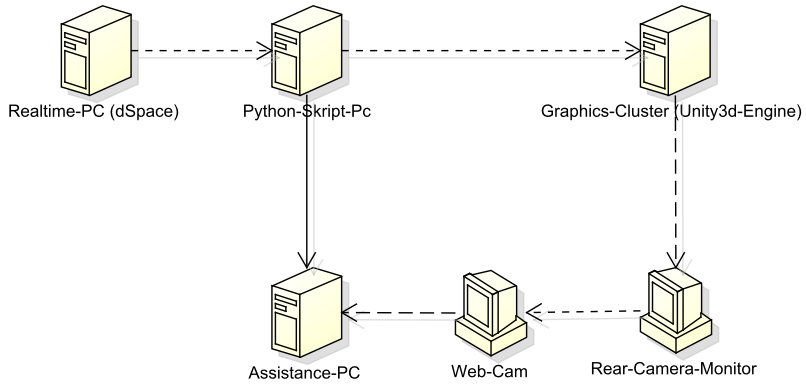


Abbildung 3.36: Visualisierung des Konzepts zur Einbindung der Assistenz.

3.6.3 Vorstellung des Konzepts zur Einbindung der Assistenz

Der in Abbildung 3.36 beschriebene Aufbau wird in den folgenden Paragraphen näher betrachtet. Die kurz gestrichelten Linien repräsentieren eine asynchrone Datenkommunikation, entweder über UDP oder durch Abfilmen des Monitors. Durchgezogene Linien entsprechen einer synchronen Kommunikation, etwa einer TCP-Verbindung oder einer physikalischen Verbindung zwischen zwei Geräten.

Zu sehen ist einerseits die Verbindung zwischen Echtzeit-Rechner und Grafik-Cluster, auf dem die Render-Engine läuft. Diese Verbindung geht den Umweg über ein Python-Skript wie im vorangegangenen Abschnitt beschrieben. Besagtes Skript wird in der Folge um einen zusätzlichen Zielport erweitert, unter dem schließlich die Assistenz erreichbar sein wird.

Da externe Rechner nicht ohne größeren Aufwand in das Netzwerk des Fraunhofer ITWM integriert werden können, ist es nicht möglich, die UDP-Pakete direkt an den Assistenz-Rechner zu verschicken. Eine Weiterreichung des Source-Codes an das Fraunhofer Institut war zudem aufgrund der Rechtslage nicht möglich. Die Kompilierung auf einem bereits im Netzwerk integrierten Rechner war somit nicht möglich. Auch eine Portierung des fertigen Kompilats war aufgrund gewisser Architektur- und OS-spezifischer Abhängigkeiten von Systembibliotheken nicht möglich.

Eine Lösung des Problems wird im Folgenden vorgestellt. Die vom Python-Skript ausgehenden Pakete werden zunächst an eine Applikation weitergeleitet, welche auf einem der bereits im Netzwerk integrierten Rechner läuft, beispielsweise dem **Python-Skript-PC** selbst.

Diese Applikation, im folgenden **UDP-Listener** genannt, empfängt die UPD-Pakete und extrahiert die für die Assistenz benötigten Werte für Einknick- und Lenkwinkel. Auf demselben Rechner wird zudem eine Instanz des ezCom-**Nachrichtenservers** gestartet, welche eine Kommunikation der ezCom-Clients über TCP-Nachrichten ermöglicht. Der **UDP-Listener** registriert sich zu Beginn als ezCom-Client. Dies erlaubt es ihm, die aus den UDP-Paketen extrahierten Winkel in separate ezCom-Nachrichten zu packen und mit dem entsprechenden Nachrichtentyp zu versenden, den die Assistenz für diese Werte erwartet.

Auf dem Assistenz-Rechner läuft in der Zwischenzeit die Assistenz-Applikation, ebenfalls als Instanz eines ezCom-Clients, und empfängt die zuvor erstellten Nachrichten des gewünschten Typs. Aufgrund der Kommunikation über das TCP-Protokoll ist es nicht notwendig, dass dem Assistenz-Rechner - im Gegensatz zur Kommunikation über UDP - umfassende Netzwerkrechte eingeräumt werden.

Ein weiterer Anknüpfungspunkt für die Assistenz ist die Akquirierung des Kamerabilds der Rückfahrkamera. Über ein virtuelles Kamera-Objekt in der Unity3d-Engine wird zunächst eine Rückfahrsicht bereitgestellt. Zur Weiterleitung des Kamerabilds an den Assistenz-Rechner gab es zwei verschiedene Möglichkeiten:

- **Direktes Auslesen des Frame-Buffers aus Unity3d:** Die ist die präferierte Variante. Die Idee besteht in dem Auslesen des low-level Frame-Buffers der Render-Engine. Unity3d stellt dabei in der Professional-Version Methoden bereit, mit der sich das aktuell angezeigte Bild in einer Textur speichern lässt, ähnlich einem Screenshot. Die Textur lässt sich in das **.png**-Format konvertieren und dann über einen UDP-Socket verschicken. Ein Vorteil dieser Variante ist, dass sie eine Simulation verschiedener Kamerasysteme ermöglicht, da sich

die Parameter der virtuellen Kamera ganz einfach Software-seitig ändern lassen. Auch hier müsste - analog zum **UDP-Listener** - der Umweg über eine Applikation gegangen werden, welche das Kamerabild zunächst aus einer UDP-Paket in eine TCP-Nachricht umwandelt, bevor sie zur Assistenz geschickt wird.

- **Abfilmen eines externen Monitors, der die Kameraansicht zeigt:** Dies stellt die alternative Variante dar, falls die erste nicht funktionieren sollte oder aufgrund von zeitlichen Beschränkungen nicht realisierbar sei. Die Idee hierbei ist, dass ein zusätzlicher Monitor abseits der Kabine zur Verfügung gestellt wird, welcher von einer auf einem Stativ angebrachten Kamera, etwa einer Web-Cam, abgefilmt wird. Das abgefilmte Kamerabild kann dabei direkt an die Assistenz-Applikation übertragen werden, sofern das Aufnahmeprogramm auf dem Assistenz-Rechner ausgeführt wird. Ein Problem mit diesem Ansatz umfasst die Kalibrierung und Anpassung der Perspektive der physikalischen Kamera an die Perspektive der virtuellen Kamera. Zudem ist man an das Modell der physikalischen Kamera gebunden. Der Aufwand, das Kameramodell auszutauschen, ist dabei ungleich höher als bei der ersten Variante.

Zu Testzwecken stand leider nur die kostenlose Standard-Version von Unity3d zur Verfügung, in welcher die benötigten Methoden zum Auslesen des Kamerabilds nicht enthalten sind. Die Implementierung hätte daher vor Ort in Kaiserslautern stattfinden müssen. Im letzten dortigen Besuch traten jedoch unvorhergesehene Software-Probleme auf, deren Lösung eine höhere Priorität besaß. Letztlich war es aufgrund von Zeitmangel im Rahmen dieser Arbeit nicht möglich die erste Variante zu implementieren.

3.6.4 Das Fahrzeugmodell

Die Simulink-Implementierung des Fahrzeugmodells erfolgte auf Basis der Kinematikberechnungen, wie sie auch *ezKine* nutzt. Abbildung 3.37 zeigt den Funktionsblock *calculateSteeringStep*, welcher die Übergangsfunktion zwischen zwei Fahrzeugkonfigu-

rationen enthält. Als Eingabeparameter stehen ein Steuervektor (*Control*) sowie der ursprüngliche Fahrzeugzustand (*Initial Configuration*) zur Verfügung, mit deren Hilfe die Berechnung des neuen Fahrzeugzustands erfolgt. Die Blöcke *SAMPLE_TIME* und *STEP_SIZE* stellen Konstanten dar. Ersterer wird zur Integration der aktuellen Geschwindigkeit genutzt, um die zurückzulegende Distanz zu bestimmen, letzterer beschreibt die minimale Distanz, die das Fahrzeug zurücklegen kann.

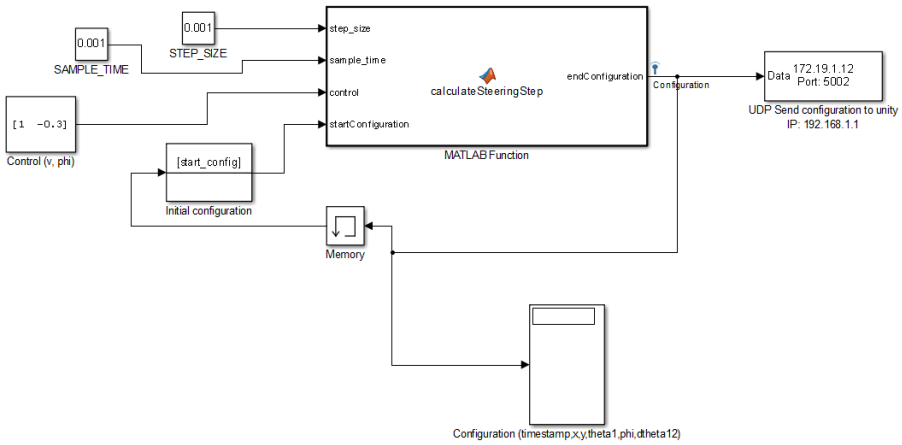


Abbildung 3.37: Das Fahrzeugmodell in Simulink.

Über einen UDP-Block wird der Fahrzeugzustand nach jeder Iteration des Modells an den Pythonskript-Rechner geschickt und an interessierte Komponenten weitergeleitet.

3.6.5 Logging von Fahrzeug-Daten

Das Logging der Daten wird durch eine Erweiterung des Python-Skripts vorgenommen, welche die Daten nach Erhalt in eine Textdatei schreibt. Dieser Vorgang stellte sich als nicht zeitkritisch heraus, sodass auch bei einer Frequenz des Fahrzeugmodells von 1000Hz jede Iteration geloggt werden konnte.

3.6.6 Schnittstellen zwischen RODOS und Fahrerassistenz

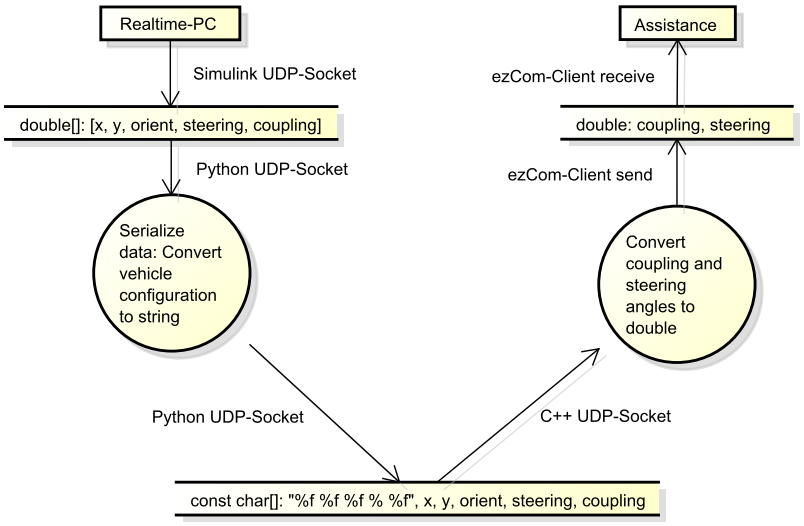


Abbildung 3.38: Datenfluss zwischen dem dSpace-Rechner und der Assistenz.

Wie in Abschnitt 3.6.2 beschrieben werden Netzwerkschnittstellen benötigt, welche eine Weiterleitung von Daten über den TCP-Nachrichtenserver von ezCom ermöglichen. Dazu werden im Folgenden zwei Applikationen vorgestellt. Die erste Applikation sorgt für die Weiterleitung von Einknick- und Lenkwinkeln, welche als UDP-Paket empfangen werden. Die zweite Applikation sorgt für eine Übertragung des Kamerabilds.

Abbildung 3.38 zeigt den Datenfluss zwischen dem Fahrzeugmodell, welches auf dem Echtzeit-Rechner läuft, und der optischen Fahrerassistenz, die auf einem Rechner außerhalb des RODOS-Netzwerks läuft. Der erste Schritt umfasst die in 3.6.1 beschriebene Serialisierung des Fahrzeugzustands (linke Seite). Danach erfolgt die Konvertierung in eine ezCom-Nachricht und die Weiterleitung an die Assistenz (rechte Seite). Da die linke Seite bereits als Python-Skript vorliegt, wird im folgenden die rechte Seite näher betrachtet.

Abbildung 3.39 zeigt ein reduziertes Klassendiagramm der Klasse **UdpListener**. Diese erbt von der in der ezCom-Bibliothek zur Verfügung gestellten Klasse **Tcp-**

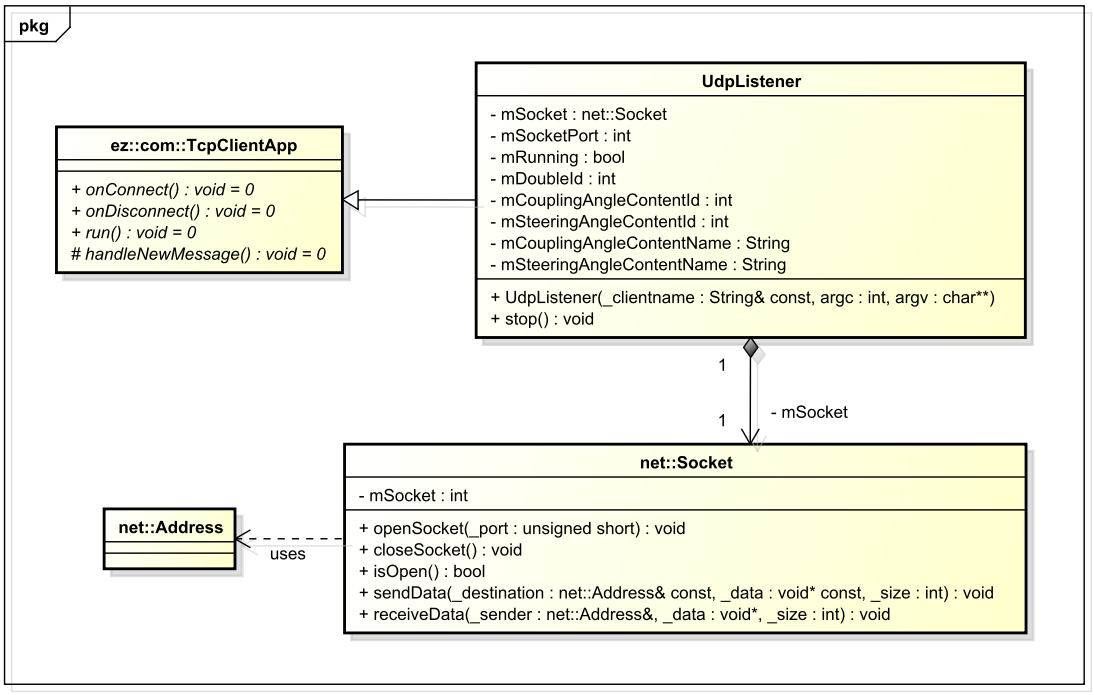


Abbildung 3.39: Die Klasse UdpListener.

ClientApp, welche den Rahmen für einen generischen ezCom-Client darstellt. Die Klasse **Socket** bildet einen Wrapper, welcher den Zugriff und die Nutzung gewöhnlicher Unix-Sockets erleichtert. Zur Repräsentation einer IP-Adresse wird zudem die Klasse **Address** zur Verfügung gestellt.

Abbildung 3.40 zeigt den Datenfluss zwischen der Kamera, welche den externen Monitor der Rückfahransicht filmt, und der Fahrerassistenz. Als Kamera wurde eine handelsübliche Webcam genutzt. Da die Applikation, welche für die Aufnahme der Kamerabilder zuständig ist, auf demselben Rechner wie die Assistenz läuft, kann das Kamerabild direkt in eine TCP-Nachricht gepackt und versendet werden.

Die ezCom-Bibliothek stellt bereits eine entsprechende Applikation namens **ImageProvider** zur Verfügung, welche die oben genannten Anforderungen erfüllt.

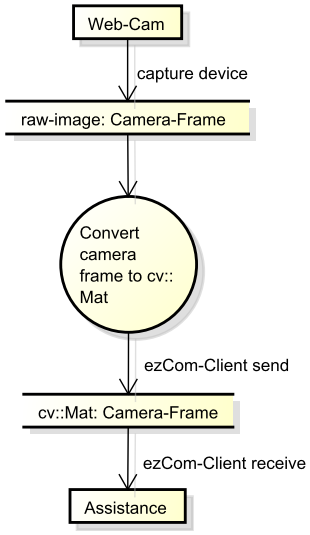


Abbildung 3.40: Datenfluss von der Webcam zur Assistenz.

3.6.7 Bewertung und Probleme

Obwohl die Funktion der Komponenten unabhängig voneinander nachgewiesen werden konnte, war es zum Ende der Arbeit nicht möglich ein funktionsfähiges Gesamtsystem zu stellen. Dies lag vor allem in der Restriktion, welche die Weitergabe des Sourcecodes der Assistenz an das Fraunhofer Institut untersagte. Bei der Portierung statisch gelinkter Applikationen traten Betriebssystem-spezifische Fehler auf, welche die Weiterarbeit behinderten. Betroffen war vor allem die Komponente *UdpListener*. Aufgrund von Zeitmangel war es nicht möglich den Fehler vor Abgabefrist der Arbeit zu beheben. Allerdings ist geplant, die Kooperation nach Abgabe der Ausarbeitung zu einem zufriedenstellenden Ende zu bringen.

4 Experimente

Die in diesem Kapitel durchgeführten Experimente sollen die in Kapitel 1.2 formulierten Leitfragen untersuchen. Aus Zeitgründen konnten die Experimente nicht im geplanten Umgang mit einer systematischen Vorgehensweise durchgeführt werden. Erste qualitative Ergebnisse konnten jedoch mithilfe des in Abschnitt 3.5 vorgestellten Testtools erzielt werden.

In Abschnitt 4.1 wird zunächst das experimentelle Setup sowie die Parametrisierung des Testtools beschrieben. In Abschnitt 4.2 erfolgt eine Beschreibung der durchgeführten Experimente.

4.1 Experimentelles Setup

Im vorliegenden Kapitel wird auf das experimentelle Setup und die Parametrisierung des Testtools näher eingegangen.

In Abschnitt 4.1.1 wird zunächst das grundlegende Vorgehen bei der Durchführung von Experimenten erläutert. In Abschnitt 4.1.2 wird schließlich die Wahl der Laufzeitparameter der Testumgebung betrachtet.

4.1.1 Grundlegendes Vorgehen

Das in Kapitel 3.5 vorgestellte **Testtool** beinhaltet eine Implementierung des aus Abschnitt 2.3.5 bekannten **Annealed Particle Filters**, welcher die Grundlage des

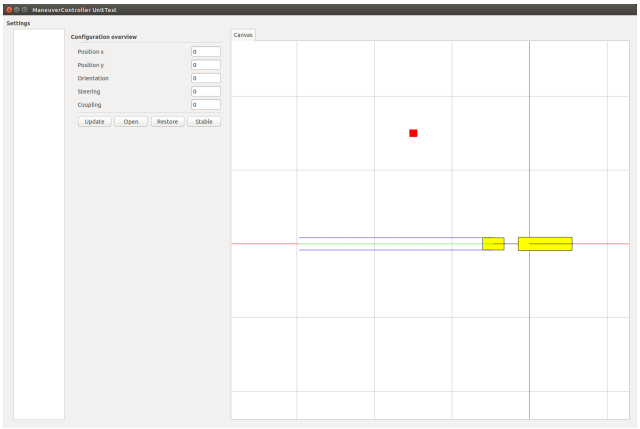


Abbildung 4.1: Exemplarische Darstellung eines Fahrszenarios.

experimentellen Vorgehens bildet. Dieser wird mit einer Menge zufällig generierter Eingabepartikel initialisiert, welche jeweils durch einem 18-dimensionalen Fehlervektor beschrieben werden, entsprechend der Beschreibung aus Kapitel 3.5.3.

Das in der Gewichtung der Partikel auszuführende Manöver entspricht einem im Vorfeld festgelegten Fahrszenario, welches sich im Verlauf des Optimierungsprozesses nicht ändert. Das Szenario hat einen festen Start- und Zielpunkt. Die Umgebung wird dabei als hindernisfrei vorausgesetzt. Abbildung 4.1 zeigt beispielhaft eine solche Fahraufgabe. Das Fahrzeug befindet sich im Ursprung, Einknickwinkel und Ausrichtung betragen 0°.

Die Parameter des hier genutzten Fahrzeugmodells finden sich in Tabelle 4.1. Am Anhänger angezeigt werden die Hilfslinien der Assistenz, welche zum Navigieren genutzt werden. Zusätzlich zum Fahrzeugmodell wird ein Modell der Kamera bereitgestellt, zu sehen in Tabelle 4.2, welches bestimmt, wo im Bild die Assistenz-Trajektorien angezeigt werden. Ziel des Manövers ist es schließlich, den roten Zielpunkt anzufahren.

truck-length	6.94
truck-width	1.80
truck-rear-overhang	1.48
truck-frontcoupling-offset	4.63
coupling-front-offset	-1.7
coupling-min	-90°
coupling-max	90°
trailer-length	2.74
trailer-width	1.66
trailer-rear-overhang	1.35
trailer-frontcoupling-offset	3.0

Tabelle 4.1: Parameter der Fahrzeugbeschreibung

camera-x	0
camera-y	0.58
camera-z	1.25
camera-α	0
camera-β	0
camera-γ	0
camera-fovh	120°
camera-fovv	90°

Tabelle 4.2: Parameter der Kamerabeschreibung

4.1.2 Wahl der Laufzeitparameter

Durch die Kombination verschiedener Verfahren entstand eine Vielzahl möglicher Parametrisierungen des Testtools. Eine geeignete Wahl der Laufzeitparameter war dabei kritisch für den Erfolg des Optimierungsprozesses. Viele der angenommenen Parameter konnten nur über den experimentellen Weg bestimmt werden, da die Zielsetzung dieser Arbeit nach Kenntnis des Autors im wissenschaftlichen Kontext bislang nicht erforscht wurde. In der Literatur fanden sich daher keine Beispiele, aus denen Referenzkonfigurationen generiert werden konnten. Das letztlich genutzte Setup lieferte die besten Ergebnisse.

Im Folgenden werden zum einen die Parameter des Annealed Particle Filter näher betrachtet, welche den Optimierungsprozess auf oberster Ebene steuern. Eine Ebene darunter sorgt die Ausführung des Manövers für eine korrekte Gewichtung der Partikel. Die Ausführung unterteilt sich dabei weiter in Parameter des Controllers, welcher Steuerkommandos produziert, sowie Parameter der Assistenz, welche die Generierung der Steuerbefehle beeinflusst. Der folgende Abschnitt stellt die grundlegenden Parametrisierungen der einzelnen Verfahren vor und beschreibt Probleme, welche bei der Wahl der Werte aufgetreten sind.

Annealing Layer und Partikelzahl

Anzahl der Partikel	25-30
Anzahl der Annealing-Layer	10
Iterationen des Partikelfilters	3-5
Überlebensrate pro Layer	0.25

Tabelle 4.3: Parameter des Annealed Particle Filters

Die Effizienz des **Annealed Particle Filters** hängt sehr stark von der Wahl seiner Laufzeitparameter ab. Tabelle 4.3 zeigt die Parametrisierung des Annealed Particle Filters, welche im Kontext der Experimente genutzt wurde.

Bisherige Anwendungen des Annealed Particle Filter erfolgten im Kontext des Pose-Trackings anhand von Volumendaten. Deutscher beschreibt in [Deutscher u. a., 2000], dass für ein Optimierungsproblem mit 30 Dimensionen eine Partikelzahl von 200 gute Resultate erzielt wurden. Eine Festlegung der Annealing-Layer auf 10 sorgte zudem für die nötige Exaktheit der Ergebnisse.

Das im Rahmen dieser Arbeit betrachtete Optimierungsproblem ist maximal 18-dimensional. Bewährt hat sich schließlich eine Reduktion der Partikelmenge auf 25 bis 30. Die Anzahl der Annealing-Layer wurde beibehalten. Tests mit 5, 10 und 15 Layern führten zu der Beobachtung, dass eine erhebliche Verbesserung der Ergebnisse durch die Erhöhung von 5 auf 10 auftrat, wohingegen die Erhöhung von 10 auf 15 Layer keine augenscheinliche Änderung verursachte.

Die Iterationen des Partikelfilters gestaltete sich variabel, da der eigentliche Optimierungsprozess durch die simulierte Abkühlung durchgeführt wurde. In vielen Experimenten stellte sich nach der dritten Iteration des Partikelfilters keine entscheidenden Verbesserungen mehr ein.

Die Wahl der Überlebensrate beeinflusst schließlich die Glätte bzw. Schärfe der Gewichtungsfunktion. Sie bestimmt wie viele Partikel effektiv den Resampling-Schritt überleben. Eine niedrige Überlebensrate sorgt dafür, dass nur Partikel mit relativ

hoher Wahrscheinlichkeit überleben, während eine hohe Überlebensrate dafür sorgt, dass die gezogenen Partikel gleichmäßiger verteilt sind. Deutscher schlägt als Referenz eine konstante Rate von 0.5 vor. In [Knopp, 2010] wurde gezeigt, dass eine niedrige Überlebensrate zu einem schnelleren Anstieg der Partikel-Gewichte während der ersten Iterationen führt. Ein solches Verhalten ist gerade deshalb von Vorteil, da die Wahrscheinlichkeit weitere, neue Optima zu finden mit der sinkenden Partikelstreuung der darauffolgenden Iterationen abnimmt. Als Überlebensrate wurde daher ein Wert von 0.25 gewählt.

Fehlerart	Standardabweichung
mErrL1	0.02
mErrL2	0.02
mErrM1	0.02
mErrCoupling-μ	0.006rad $\widehat{=}$ 0.34377°
mErrCoupling-σ	0.003rad $\widehat{=}$ 0.171887°
mErrSteering-μ	0.006rad $\widehat{=}$ 0.34377°
mErrCoupling-σ	0.003rad $\widehat{=}$ 0.171887°
mResCoupling	0.006rad $\widehat{=}$ 0.34377°
mResSteering	0.006rad $\widehat{=}$ 0.34377°
mDelayCoupling	50ms
mDelayControl	0ms
mDelayFrame	50ms
mErrCameraPos-x	0.02
mErrCameraPos-y	0.02
mErrCameraPos-z	0.02
mErrCameraOrient-x	0.0123rad $\widehat{=}$ 0.704738°
mErrCameraOrient-y	0.0123rad $\widehat{=}$ 0.704738°
mErrCameraOrient-z	0rad $\widehat{=}$ 0°

Tabelle 4.4: Initiale Standardabweichungen des Annealed Particle Filters

Der letzte Schritt ist die Wahl initialer Varianzen bzw. Standardabweichungen der simulierten Abkühlung. Ebenfalls von Deutscher wurde darauf hingewiesen, dass diese so gewählt werden sollten, dass sie der Hälfte der erwarteten Partikelbewegung im ersten Schritt entsprechen.

Die in Tabelle 4.4 gelisteten Standardabweichungen beziehen sich jeweils auf einen

Wert der Fehlerkonfiguration, welche den Partikel repräsentiert. Zu sehen ist, dass sowohl die Standardabweichung der Steuerverzögerung als auch die der z-Rotation der Kamera auf 0 gesetzt wurde.

Im ersten Fall bedeutet dies, dass Verzögerungen in der Anwendung der Steuerkommandos nicht beachtet werden sollen. Diese Größe wurde eingeführt, um die Verzögerung eines Menschen bei der Bedienung des Lenkrads zu simulieren. Aufgrund der schlechten Ergebnisse, die das Setzen dieses Wertes mit sich brachte, wurde dieser Wert in der Betrachtung vernachlässigt.

Der zweite Fall ist für die Betrachtung von Fehlern irrelevant, da dieser Fehler (noch) nicht implementiert wurde und nur der Vollständigkeit halber in die Betrachtung aufgenommen wurde.

Ausführung des Manövers

Realgeschwindigkeit des Fahrzeugs	$9.0\frac{km}{h}$
Auflösung der Kinematikberechnung	0.01m
Reaktionsrate des Fahrer auf Informationen der Assistenz	25Hz
Speed-Up-Faktor der Ausführung	4-fach

Tabelle 4.5: Parameter des Manövers

Die Ausführung des Manövers bildet die Basis für die Gewichtung der Partikel des Annealed Particle Filters und ist damit verantwortlich dafür, welche Partikel den Resampling-Schritt überstehen. Tabelle 4.5 zeigt eine Übersicht der statischen Parameter des Manövers.

Bei den im Kontext der Evaluation betrachteten Manövern wird mit einem mehrgliedrigen Zugfahrzeug rückwärts rangiert. Die dabei vorherrschenden Geschwindigkeiten sind mitunter sehr gering. Als Realgeschwindigkeit des Fahrzeugs wurde daher ein Wert von weniger als $10\frac{km}{h}$ gewählt, mit dem sich das Fahrzeug konstant rückwärts bewegt.

Die Kinematikberechnung des Fahrzeugs basiert auf ezKine, welche nicht auf Geschwindigkeiten rechnet, sondern auf Distanzen, die das Fahrzeug zurücklegt. Basierend auf der Realgeschwindigkeit können zurückgelegte Distanzen pro Zeitschritt berechnet werden. Die minimale Auflösung der Distanz bestimmt dabei wie genau die Kinematikberechnung durchgeführt wird. Der Standardwert von 0.1m erwies sich von der Auflösung her als zu grob, um verschiedene Realgeschwindigkeiten zu simulieren. Daher wurde eine um den Faktor 10 feinere Auflösung verwendet, was numerische Fehler erheblich reduzierte.

Schließlich muss betrachtet werden, wie regelmäßig eine Reaktion des Fahrers auf aktualisierte Informationen der Assistenz stattfindet. Mit anderen Worten: Wie regelmäßig erfolgt die Generierung eines neuen Steuerbefehls durch den Manöver-Controller? Eine zu hochfrequente Reaktionsrate wäre nicht konform mit der Limitierung der menschlichen Wahrnehmung. Eine zu geringe Reaktionsrate würde den in der Realität kontinuierlichen Entscheidungsprozess nicht adäquat abbilden. Ein Kompromiss wurde schließlich bei einem Wert von 25 erreicht.

Die Ausführungszeit des Manövers bildet ein Bottleneck in der Gewichtung eines Partikels, da es aufgrund der Betrachtung von Datenverzögerungen in Echtzeit ausgeführt werden muss. Aufgrund der daraus resultierenden langen Ausführungszeiten des Annealed Particle Filters wurde nach einer Möglichkeit gesucht, die Ausführungszeit zu reduzieren. Dazu wurde eine zeitliche Skalierung der Ausführung in Betracht gezogen. Die minimale Zeitauflösung, welche zwischen zwei Controller-Schritten zuverlässig gemessen werden konnte, lag bei 1ms. Beschleunigt man in der Folge die Ausführung, wird die Auflösung um denselben Faktor verringert, was zu Ungenauigkeiten in der Betrachtung von Verzögerungen führt.

Tabelle 4.6 zeigt Messwerte für ein von verschiedenen Verzögerungen betroffenes Manöver, jeweils mit einem anderen Beschleunigungsfaktor ausgeführt. Wie man sieht unterscheiden sich die Ausführung in Echtzeit und 4-fach beschleunigte Ausführung kaum bis gar nicht voneinander. Die Auflösung der Verzögerung beträgt

hier 4ms. Erst ab einer 8-fachen Beschleunigung trat ein merklicher Fehler im Ziel-Offset auf. Als maximale Beschleunigung des Manövers wurde daher der Faktor 4 ausgewählt.

Ausführung des Manöver-Controllers

Maximale Änderungsrate des Lenkwinkels	$30\frac{^\circ}{s}$
Glättungsfaktor der Richtungsänderung	0.01
Glättungsfaktor der Radiusänderung	0.01-0.3

Tabelle 4.7: Parameter des Manöver-Controllers

Die gesamte Ausführung des Manövers basiert auf der passenden Konfiguration des Manöver-Controllers, welche von Fahrsituation zu Fahrsituation variieren kann. Wichtig war es hierbei vor allem auf die Grenzen des physikalisch machbaren zu achten. Die Lenkwinkeländerungsrate sollte einem Wert entsprechen, den ein durchschnittlicher Fahrer in einem Auto mit Servolenkung erreichen kann. Durch einen Testvorgang im realen Fahrzeug wurde ein Näherungswert von 540°/s für die Lenkradwinkeländerungsrate bestimmt. Das Übersetzungsverhältnis von Lenkrad- zu Lenkwinkel beträgt etwa 15 zu 1, wie durch Experimente in der Arbeitsgruppe gezeigt werden konnte. Auf diese Weise ließ sich ein Wert von 30°/s als Änderungsrate des Lenkwinkels bestimmen.

Die restlichen Parameter beschreiben wie stark die Lenkbewegungen des Controller

Faktor	Ziel-Offset	DTW
1	0.059	0.043
4	0.06	0.043
8	0.388	0.049
20	0.393	0.055

Faktor	Ziel-Offset	DTW
1	0.233	0.0057
4	0.234	0.0064
8	0.34	0.0094
20	0.89	0.03

Tabelle 4.6: Vergleich der Abweichungen verschiedener Beschleunigungsfaktoren. Die DTW wurde anhand eines Referenzdatensatzes berechnet. Links wurde eine Verzögerung von 115ms in der Weiterleitung des Einknickwinkels, rechts in der Weiterleitung des Kamerabilds simuliert.

ausfallen abhängig von der Distanz zum Ziel, siehe Kapitel 3.3.3. Die vorliegenden Werte wurden experimentell bestimmt und lieferten die besten Ergebnisse für verschiedene Fahrsituationen.

Berechnung der Trajektorien

Iterationen der Assistenz	250
Distanzschritt einer Iteration der Assistenz	0.1
Nutzung der Unscented Transform	false

Tabelle 4.8: Parameter der Trajektorienberechnung

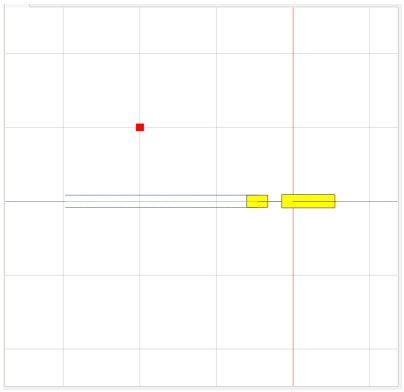
Die Assistenz-Komponente besitzt wenige konfigurierbare Parameter. Zu diesen zählen die Anzahl der Iterationen sowie die zurückgelegte Distanz eines Berechnungsschritts, zu sehen in Tabelle 4.8. Aus dem Produkt dieser beiden Werte ergibt sich die Länge der Trajektorie, welche in diesem Fall auf 25 festgelegt wurde. Dieser Wert wurde aus den Tests des Assistenzprototyps übernommen.

4.2 Experimente

Eine umfassende Durchführung von Experimenten war aus Zeitgründen nicht möglich. Es wurde sich daher auf drei verschiedene Fahrszenarien beschränkt, welche in der Folge anhand verschiedener Fehlerkonfigurationen untersucht wurden. Das erste Experiment beschreibt ein Manöver auf sehr engem Raum mit starker Kurvenfahrt. Das zweite Experiment erfordert ebenfalls eine Kurvenfahrt, diesmal jedoch mit größeren Freiheiten in der Bewegung. Das letzte Experiment betrachtet schließlich die Aufgabe, parallel rückwärts zu rangieren. Die Ergebnisse werden im folgenden Abschnitt vorgestellt.

4.2.1 Experiment 1

Ausgangssituation und Durchführung



Fehlermaß	Gewicht	Grenzwert
DTW	1.0	0.5
Position-Offset	1.0	0.5
Orientation-Offset	0.5	5°
Coupling-Offset	0.5	5°
Steering Reversal Rate	0.5	0.1
Steering Stddev	0.5	10°
Steering Criticality	0.0	0.0
Coupling Criticality	0.0	0.0

Abbildung 4.2: Das Ausgangsszenario für Experiment 1 zusammen mit den Parametern der Gewichtungsfunktion.

Die Ausgangssituation des Manövers ist in Abbildung 4.2 links zu sehen. Der Zielpunkt liegt bei (-20, -10). Die grundlegende Idee hinter der Auswahl dieses Manövers ist zu testen, wie sich die Assistenz beim Rangieren auf relativ engem Raum verhält. Es wurde eine Kurvenfahrt gewählt, da diese kinematisch weniger Spielraum bietet, den Punkt anzufahren.

Die ideale Steuerung des Fahrzeugs würde zwei Lenkbewegungen umfassen, eine zum Einstellen des Einknickwinkels und eine zum Folgen der stabilen Kreisbahn. Da der Punkt nahe am Fahrzeug liegt ist die stabile Kreisbahn schnell erreicht. Bis zu diesem Punkt sollte dem Nutzer freistehen, in welcher Richtung er den Punkt anfährt. Einmal auf der stabilen Kreisbahn folgt er ihr bis der Punkt erreicht wird.

Die Tabelle, zu sehen rechts in Abbildung 4.2, beschreibt die Parameter der Gewichtungsfunktion. Wichtig ist stets, dass das Ziel erreicht wird, als Toleranzwert wurden dabei 0.5m vorgegeben. Unterstützende Größen zum Erreichen des Ziels sind zum einen Abweichungen von der Referenztrajektorie, welche die ”optimale” Navigation zum Ziel repräsentiert sowie Abweichungen von Ausrichtung und Einknickwinkel.

Diese Größen beschreiben wie der Zielpunkt angefahren wurde. Gerade bei besonders engen Manövern mit wenig Spielraum ist es wichtig, dass keine großen Bögen gefahren werden. Für den Lenkwinkel wurden zwei Größen angegeben. Die Reversal Rate und die Steering Variance wurden in Relation zum Referenzdatensatz gewählt und sollen zu große und frequente Lenkbewegungen vermeiden, welche bei einem engen Manöver sehr schnell zu Abweichungen führen können. Die aufsummierten Gewichte ergeben ein Maximalgewicht von 4, welches von einem Partikel erreichen werden konnte.

Die Partikelmenge wurde mit 25 Partikeln initialisiert. Die Anzahl der Iterationen des Partikelfilters betrug 3.

Beobachtung

Abbildung 4.3 zeigt die Entwicklung der Partikelgewichte über die Zeit. Die zunächst zufällig gestreuten Partikel besitzen zu Beginn relativ niedrige Gewichte. Nach den ersten 10 Iterationen, welche einem Partikelfilter-Durchlauf (10 Annealing-Schritte) entsprechen, bildet sich ein erstes Plateau, welches die Gewichte erreichen. Die durchschnittlichen Gewichte pendeln sich um einen Wert von knapp unter 3 ein. Im nächsten Durchlauf erfolgt ein starker Sprung der Gewichte auf fast 3.8. Im letzten Schritt konnte das Gewicht nicht mehr entscheidend verbessert werden und pendelte weiterhin um einen Durchschnittswert von 3.8 von möglichen 4, was eine sehr gute Annäherung bezeichnet.

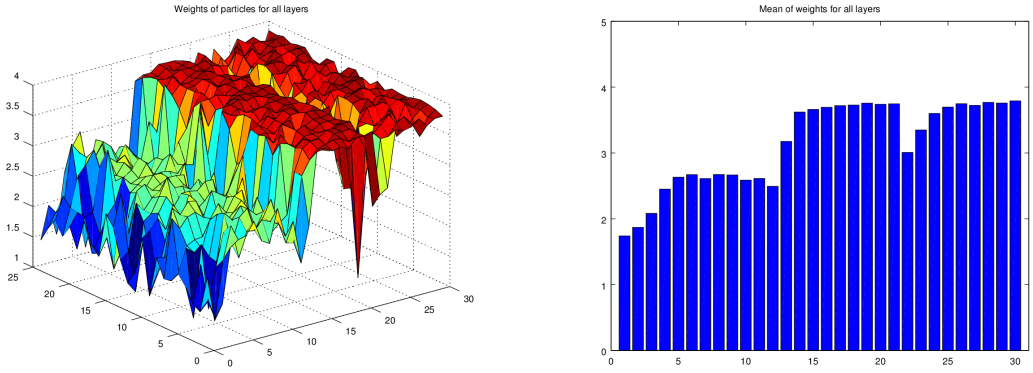


Abbildung 4.3: Links dargestellt ist die Entwicklung der einzelnen Partikelgewichte abhängig von der Anzahl der Iterationen. Die linke Achse beschreibt dabei die Gesamtzahl der Iterationen, die rechte die Anzahl der Partikel. Rechts dargestellt ist die Entwicklung der durchschnittlichen Partikelgewichte über alle Iterationen.

Abbildung 4.4 zeigt die Entwicklung der Partikelmenge anhand der durchschnittlichen Einzelfehler sowie die jeweiligen Standardabweichungen. In jeder Grafik werden die in Abschnitt 3.5.3 definierten Werte von links nach rechts der Reihe nach aufgetragen. Bei der Normalverteilung erfolgt die Nennung von μ vor σ.

Mit jeder Iteration des Partikelfilters erfolgt eine immer stärkere Ausprägung der relevantesten Fehler. Die ausgeprägtesten Fehler der letzten Iteration umfassen *mErrCoupling-*μ sowie *mErrSteering-*μ im Bereich von je 1.3°. Stark ein gehen auch die Verzögerungen *mDelayCoupling* und *mDelayFrame* mit 101ms bzw. 125ms. Der Fehler in der Kamerapose äußert sich am stärksten in *mErrCameraOrient-y* mit 1.95°.

Eine hohe Standardabweichung weist auf große Schwankungen der Werte innerhalb der Partikelmenge hin. Je stärker die Standardabweichung eines Fehlertyps, desto inkonsistenter ist er in der Partikelmenge. In Gegenzug ist ein Fehler umso konsistenter in der Fehlermenge, je kleiner die Standardabweichung ist. Es ist gut zu sehen, dass es sich bei der resultierenden Partikelmenge um eine sehr konsistente Menge handelt.

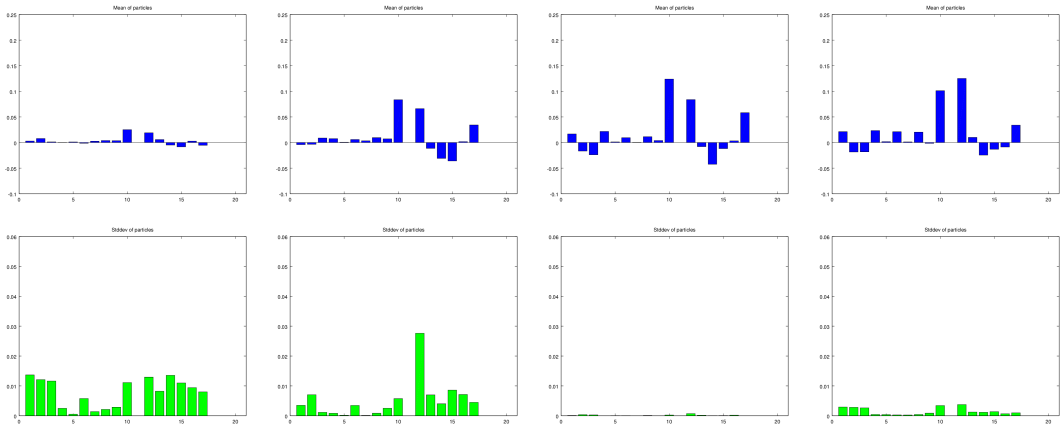
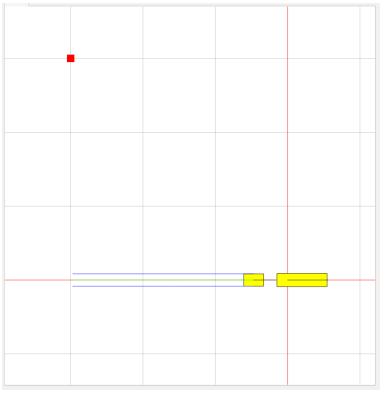


Abbildung 4.4: Oben: Die Entwicklung des durchschnittlichen Fehlerwerte einer Partikelmenge über mehrere Schritte des Partikelfilters. Links ist der Mittelwert der Eingabe-Partikelmenge zu sehen, es folgen die jeweiligen Ausgabe-Partikelmengen der 3 Iterationen des Partikelfilters. Unten: Die entsprechenden Standardabweichungen der Partikelmenge.

4.2.2 Experiment 2

Ausgangssituation und Durchführung



Fehlermaß	Gewicht	Grenzwert
DTW	1.0	0.75
Position-Offset	1.0	0.5
Orientation-Offset	0.75	5°
Coupling-Offset	0.75	5°
Steering Reversal Rate	0.75	0.25
Steering Stddev	0.75	5°
Steering Criticality	0.0	0.0
Coupling Criticality	0.0	0.0

Abbildung 4.5: Das Ausgangsszenario für Experiment 2 zusammen mit den Parametern der Gewichtungsfunktion.

Die Ausgangssituation des Manövers ist in Abbildung 4.5 links zu sehen. Der Zielpunkt liegt dieses Mal bei (-30, -30). Die grundlegende Idee hinter der Auswahl dieses

Manövers ist zu testen, wie sich die Assistenz beim Rangieren auf größeren Raum verhält, wenn das Ziel weiter vom Fahrzeug entfernt ist. Beibehalten wurde die Kurvenfahrt, da bei großen Einknickwinkeln die größten Fehler auftreten, siehe Kapitel 3.2.

Im Gegensatz zu Experiment 1 wurde hier jedoch viel mehr Spielraum gegeben im Hinblick auf die Möglichkeiten den Punkt anzufahren. Durch die geringe Länge der Assistenz-Trajektorien relativ zum Abstand des Punktes benötigt das Fahrzeug einige Zeit um die stabile Fahrt zu erreichen. Auch hier sind theoretisch nicht mehr als zwei Lenkbewegungen vonnöten, um den Punkt anzufahren. Freigestellt ist es dem Nutzer jedoch, wie er die erste Lenkbewegung initiiert, um den Einknickwinkel einzustellen, den er zur Kurvenfahrt benötigt.

Analog zu Experiment 1 beschreibt die Tabelle in Abbildung 4.5 die Parameter der Gewichtungsfunktion des zweiten Experiments. Aufgrund des erhöhten Abstands zum Ziel wurde der Wert für die DTW etwas erhöht. Bei der Wahl der Standardabweichung des Lenkwinkels wurde sich an dem Wert des Referenzdatensatzes orientiert. Die Gesamtgewichtung wurde zudem auf 5 erhöht, um zu überprüfen, wie sich Unterschiede in der Priorisierung der Maße auswirken.

In diesem Durchlauf wurden 30 Partikel betrachtet bei 5 Iterationen des Partikelfilters.

Beobachtung

Abbildung 4.6 zeigt erneut die Entwicklung der Partikelgewichte über die Zeit. Auch hier wurden die Partikel zu Beginn zufällig im Fehlerraum gestreut. Es fällt auf, dass die Partikel relativ schnell in den Gewichten gestiegen sind. Schon nach der ersten Iteration des Partikelfilters (10 Annealing-Schritte) pendelt sich die Menge bei einem Gewicht von ca. 3.8 ein. Nach einer noch geringen Verbesserung zum Ende des zweiten Schritts hielten sich die Durchschnittswerte am Ende eines Partikelfilter-

Schritts stets konstant bei knapp unter 4. Im letzten Schritt war sogar ein leichter Rückgang zu beobachten. Die angepeilte Marke eines Gewichts von 5 wurde somit stark verfehlt.

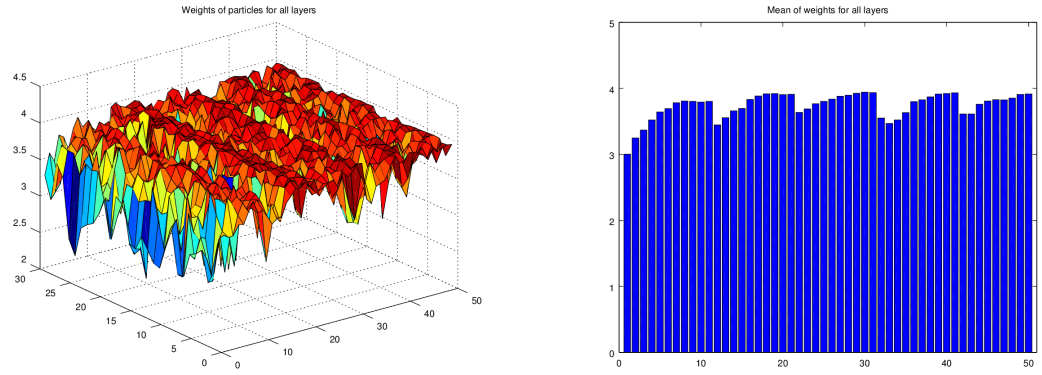


Abbildung 4.6: Links dargestellt ist die Entwicklung der einzelnen Partikelgewichte abhängig von der Anzahl der Iterationen. Die linke Achse beschreibt dabei die Gesamtzahl der Iterationen, die rechte die Anzahl der Partikel. Rechts dargestellt ist die Entwicklung der durchschnittlichen Partikelgewichte über alle Iterationen.

In Abbildung 4.7 sind erneut die Entwicklung der Partikelmenge sowie die jeweiligen Standardabweichungen der einzelnen Fehlerarten zu sehen. Die zweite Grafik von links beschreibt die Partikelmenge zum Ende des ersten Partikelfilterschritts, wo sich die Gewichte bereits auf ein stabilen Niveau eingependelt hatten. Die beteiligten Fehler sind schwach ausgeprägt und inkonsistent. Nach einer weiteren Iteration bildet sich drei Peaks heraus, bei *mDelayFrame*, *mErrL1* sowie *mErrCameraOrient-y*. Die resultierende Partikelmenge ist dabei konsistenter als die vorherige.

Rechts zu sehen ist schließlich die Partikelmenge nach dem letzten Schritt. In dem Versuch das Gewicht weiter zu verbessern, erfolgte die Abtastung neuer Bereiche des Fehlerraums, was sich vor allem in der Fahrzeuggeometrie und der Kameraposition bemerkbar machte. Das Resultat ist ein relativ konsistenter Fehler in *mErrL1* und *mErrL2* um etwa -5cm. *mDelayFrame* liegt bei etwa 180ms, jedoch weist der Wert eine mit 50ms sehr hohe Standardabweichung aus. Sehr konsistent geht *mErrCameraPos-*

z in den Fehler ein. Deren Wert beträgt hier ca. -8.5cm. Wichtig zu erwähnen ist, dass *mDelayCoupling*, obwohl es einen geringen Mittelwert besitzt, ebenfalls starken Schwankungen unterliegt. Es kann davon ausgegangen werden, dass in vielen Partikeln daher auch die Einknickwinkelverzögerung eine wichtige Rolle spielt, während sie in anderen fast gänzlich ignoriert wird.

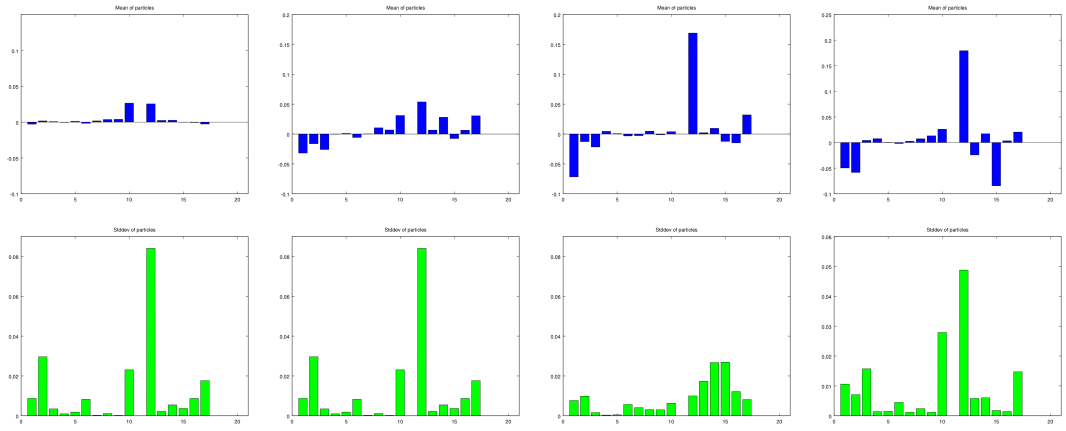
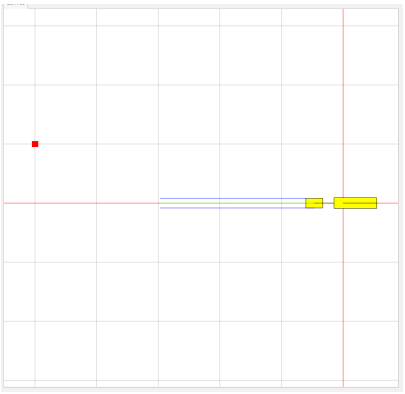


Abbildung 4.7: Oben: Die Entwicklung des durchschnittlichen Fehlerwerte einer Partikelmenge über mehrere Schritte des Partikelfilters. Links ist der Mittelwert der Eingabe-Partikelmenge zu sehen, es folgen die Ausgabe-Partikelmengen der Iterationen 1, 2 und 5 des Partikelfilters. Unten: Die entsprechenden Standardabweichungen der Partikelmenge.

4.2.3 Experiment 3

Ausgangssituation und Durchführung



Fehlermaß	Gewicht	Grenzwert
DTW	1.0	0.75
Position-Offset	1.0	0.5
Orientation-Offset	0.75	5°
Coupling-Offset	0.75	5°
Steering Reversal Rate	0.75	0.25
Steering Stddev	0.75	2°
Steering Criticality	0.0	0.0
Coupling Criticality	0.0	0.0

Abbildung 4.8: Das Ausgangsszenario für Experiment 3 zusammen mit den Parametern der Gewichtungsfunktion.

Die Ausgangssituation des Manövers ist in Abbildung 4.2.3 links zu sehen. Der Zielpunkt liegt dieses Mal bei (-50, -10). Nachdem in den ersten beiden Experimenten das Verhalten der Assistenz bei großen Einknickwinkeln getestet wurde, erfolgt nun die Betrachtung eines Manövers, welches sich durch die Einstellung eines vergleichsweise geringen Einknickwinkels durchführen lässt. Durch den geringen y-Versatz des Ziels und den großen Abstand zum Fahrzeug sind nur geringe Lenkbewegungen vonnöten, um die passenden Lenkwinkel einzustellen. Der große Abstand zum Fahrzeug gibt dem Fahrer zudem genügend Zeit, auf Schwankungen der Trajektorie zu reagieren und seinen Kurs zu korrigieren, was sich bei der Kurvenfahrt der vorangegangenen Experimente als schwierig erweisen würde, da sich große Einknickwinkel nur durch große Lenkbewegungen ausgleichen lassen.

In der Tabelle in Abbildung 4.2.3 finden sich schließlich die Parameter der Gewichtungsfunktion des dritten Experiments. Die einzige Änderung im Gegensatz zum zweiten Experiment besteht auch hier in der Anpassung der Standardabweichung für das Lenkverhalten. Die Gesamtgewichtung beträgt dieses Mal ebenfalls 5.

Auch hier wurden 30 Partikel über 5 Iterationen des Partikelfilters verfolgt.

Beobachtung

Die Entwicklung der Partikelgewichte dieses Experiments ist in Abbildung 4.9 dargestellt. Die zu Beginn zufällig gestreuten Partikel erreichten auch in diesem Fall schnell ein Durchschnittsgewicht von mehr als 4. Doch wie schon im zweiten Experiment begannen die Gewichte ab dem dritten Schritt zu stagnieren und verschlechterten sich gegen Ende hin erneut. Der maximal erreichte Durchschnittswert der Gewichte lag bei ca. 4.4 von möglichen 5, was eine Verbesserung im Vergleich zum letzten Experiment darstellt.

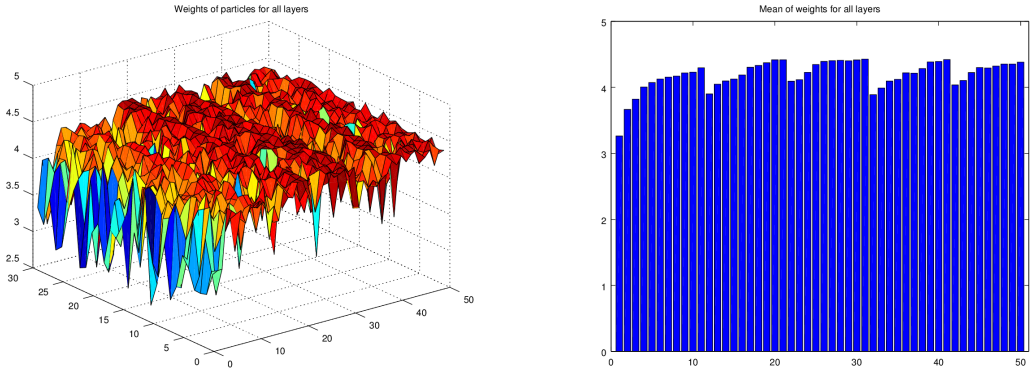


Abbildung 4.9: Links dargestellt ist die Entwicklung der einzelnen Partikelgewichte abhängig von der Anzahl der Iterationen. Die x-Achse beschreibt die 25 Partikel, die y-Achse die 30 Iterationen des Annealed Particle Filters. Rechts dargestellt ist die Entwicklung des Durchschnittswert der Partikel eines Layers. Die x-Achse entspricht erneut den 30 Partikeln.

Die in Abbildung 4.10 dargestellten Grafiken erlauben die Beobachtung desselben Phänomens aus Experiment 2. Nachdem sich nach dem zweiten Schritt (Grafik 3 oben) eine relativ konsistente Fehlermenge gebildet hat, erfolgte im Versuch das Gewicht zu verbessern eine erneute Streuung der Partikel.

Zu den prominentesten Fehlern gehören erneut die beiden Verzögerungen, welche mit 100ms für *mDelayCoupling* und 150ms für *mDelayFrame* eingehen. Besonders

ausgeprägt sind dieses Mal die Sensorik-Fehler, *mErrCoupling-μ* etwa liegt bei fast 1°, *mResCoupling* und *mResSteering* in einem ähnlichen Bereich. Mit jeweils 3 bis 5cm großen Verschiebungen gehen zudem *mErrL2* sowie die Positionsfehler in der Kamera in den Fehler mit ein.

Die allgemeine Konsistenz dieser Partikelmenge ist recht hoch, was die Sensor-Fehler betrifft. Erneut zeigen sich jedoch große Schwankungen in den Verzögerungen. Gleichermaßen verteilt zeigen sich die Fehler in der Fahrzeuggeometrie sowie der Kameraposition.

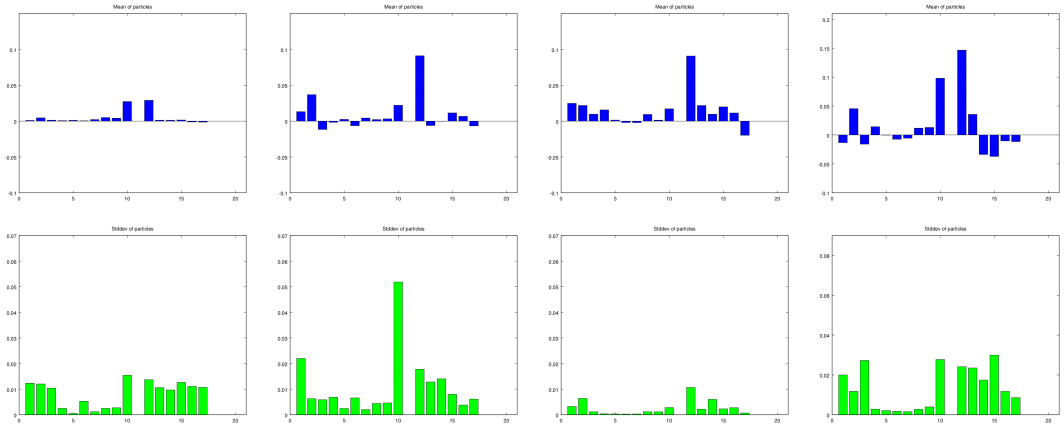


Abbildung 4.10: Oben: Die Entwicklung des durchschnittlichen Fehlerwerte einer Partikelmenge über mehrere Schritte des Partikelfilters. Links ist der Mittelwert der Eingabe-Partikelmenge zu sehen, es folgen die jeweiligen Ausgabe-Partikelmengen der Iterationen 1, 2 und 5 des Partikelfilters. Unten: Die entsprechenden Standardabweichungen der Partikelmenge.

5 Evaluation

Das folgende Kapitel befasst sich mit der Evaluation der erzielten Ergebnisse.

In Abschnitt 5.1 erfolgt die Evaluation des genutzten Verfahrens. Dabei wird vor allem auf die Optimalität der resultierenden Partikelmenge eingegangen. Die durchgeführten Experimente werden schließlich in Abschnitt 5.2 ausgewertet.

5.1 Evaluation des Verfahrens

5.1.1 Optimalität der Ausgabe-Partikel

Zunächst wird die Optimalität des Annealed Particel Filters bewertet. Dazu wurden die aus Kapitel 4.2 bekannten Durchschnittswerte der Ausgabepartikel jedes Experiments betrachtet und mittels der Manöver-Durchführung in eine Menge an Maßzahlen überführt. Tabelle 5.1.1 zeigt exemplarisch die erhaltenen Maße eines Durchlaufs. Zu

Fehlermaß	Exp1	Anteil	Exp2	Anteil	Exp3	Anteil
DTW	0.37	0.763	0.37	0.358	0.5528	0.75
Position-Offset	0.51	0.994	0.487	0.997	0.647	0.708
Orientation-Offset	4.31°	0.365	5.77°	0.68	4.77°	0.744
Coupling-Offset	4.75°	0.495	3.17°	0.439	2.53°	0.282
Steering Reversal Rate	3.89%	0.476	23.95%	0.745	27.74%	0.73
Steering Stddev	9.61°	0.497	4.31°	0.695	1.738°	0.7
Gesamtgewicht	-	3.71 / 4	-	3.912 / 5	-	3.924 / 5

Tabelle 5.1: Anteile der Fehler am Gesamtgewicht des Partikels.

sehen ist zudem der jeweilige Anteil, den ein Maß am Gesamtgewicht des Partikels hatte. Das prozentual beste Ergebnis wurde in Experiment 1 erzielt mit einer Genauigkeit von 92.75% des Maximalgewichts. Die beiden anderen anderen Experimente folgen mit 78.24% sowie 78.48% Genauigkeit.

Die erste Fahrsituation war sehr eingeschränkt in ihrer Ausführung und ließ deshalb wenig Spielraum zum Manövrieren. Wenige Optionen standen zur Verfügung, die Vorgaben der Gewichte und Schranken zu erreichen. Die resultierende Partikelmenge wies folglich eine hohe Konsistenz auf, siehe Abbildung 4.4. Die Genauigkeit der Mittelung entspricht daher fast genau dem in Abbildung 4.3 zu sehenden durchschnittlichen Gewicht der Partikelmenge.

Das Problem der letzten beiden Fahrsituationen bestand in dem größeren Freiraum, der der Ausführung gewährt wurde. Wie man in Abbildungen 4.7 und 4.10 erkennen kann, ist die Konsistenz der resultierenden Partikelmengen deutlich geringer, was auf eine eine größere Variabilität der Fehlerkonfigurationen hindeutet. Im Falle des zweiten Experiments sind Gewicht des Durchschnittsfehlers und Durchschnitt aller Gewichte noch annähernd gleich. Im dritten Experiment verschlechtert sich die Genauigkeit bereits um mehr als 12% bzw. einen Wert von knapp 0.6.

Im Gegensatz zu den bisherigen Anwendungen des Annealed Particle Filters im Kontext des Pose-Trackings besteht hier ein komplexerer Zusammenhang zwischen Eingabefehler und Ausgabefehler. Eine Mittelung der Fehlerkonfigurationen führt deshalb womöglich zu einer leichten Verfälschung der Ergebnisse durch Auslöschungen und Verstärkungen, welche in einzelnen Partikeln nicht vorhanden waren. Die Tatsache, dass Experiment 2 kaum betroffen ist, kann einem zufälligen Umstand geschuldet sein. Experiment 3 zeigt besonders große Inkonsistenzen in der Fahrzeugvermessung und den Kameraparametern. Hier ist eine gegenseitige Beeinflussung der Fehler sehr wahrscheinlich.

Statt des Durchschnitts sollten daher Partikel ausgewählt werden, welche ein besonders hohes Gewicht besitzen und einzeln untersucht werden. Aufgrund von Zeitmangel

gegen Ende der Ausarbeitung war es nicht mehr möglich einzelne Partikel einer genaueren Untersuchung zu unterziehen, daher musste man sich mit der Auswertung der Mittelwertmenge zufrieden geben.

5.1.2 Reproduzierbarkeit der Partikelgewichte

Fehlermaß	Lauf 1	Lauf 2	Lauf 3
DTW	0.517477	0.545	0.55478
Position-Offset	0.661	0.706	0.643
Orientation-Offset	4.54°	4.77°	5.59°
Coupling-Offset	1.85°	2.4°	2.52°
Steering Reversal Rate	28.2%	30.5%	33.1%
Absolute Reversals	121	131	142
Steering Stddev	1.675°	1.7°	1.741°
Min Steering	-4.86°	-4.86°	-4.87°
Max Steering	7.11°	7.83°	7.05°
Steering Criticality	15.81%	17.4%	15.67%
Min Coupling	-3.67°	-3.7°	-3.76°
Max Coupling	0.19°	0.22°	0.35°
Coupling Criticality	4.08%	4.12%	4.18%

Tabelle 5.2: Reproduzierbarkeit eines Partikelgewichts anhand der wiederholten Ausfuehrung eines Manoevers.

Ein weiteres Problem, welches während der Auswertung auftrat, war die Abhängigkeit verschiedener Partikel von probabilistischen Einflüssen der Fehlerkonfiguration. Die Messungen von Einknick- und Lenkwinkel unterliegen jeweils normalverteiltem Rauschen. Je höher das Rauschen, desto geringer war die Reproduzierbarkeit des Partikelgewichts.

Tabelle 5.1.2 zeigt exemplarisch drei verschiedene Durchläufe des dritten Experiments mit der in Tabelle 5.2.1 zu findenden durchschnittlichen Fehlerkonfiguration. Diese besitzt eine relativ hohe Standardabweichung des Lenkwinkels, welche zu Ungenauigkeiten im Endzustand des Fahrzeugs sowie der zurückgelegten Trajektorie führt. Es ist daher möglich, dass die Auswahl von Partikeln im Resampling-Schritt anhand einzelner Ausreißer erfolgte. Wie sehr der Annealed Particle Filter die Präsenz solcher

Ausreißer kompensieren kann, ist nicht bekannt, da der hier präsentierte Ansatz in der Literatur bislang nicht erforscht wurde.

Die in Abschnitt 5.1.1 genutzten Ausgabe-Werte der Experimente wurden über mehrere Iterationen des Manövers mit denselben Startvoraussetzungen bestimmt. Das Ergebnis, welches die beste Bewertung erzielte, wurde schließlich als Referenzwert genommen.

5.1.3 Wahl der Gewichte und Schranken

Aufgrund von Zeitmangel war es nicht möglich eine umfangreiche Betrachtung der Auswirkung verschiedener Gewichte und Schranken auf die einzelnen Fahrsituationen durchzuführen. Verschiedene Experimente mit reduzierter Partikelzahl und Iterationsschritten führten jedoch zu der Annahme, dass einige Zielvorgaben besonders stark mit einer bestimmten Gruppe von Fehlern korrelieren, während andere besonders von einer anderen Gruppe beeinflusst werden.

Eine Erhöhung der DTW beispielsweise resultierte entweder in einem weiteren oder einem engeren Bogen, der bei der Anfahrt auf das Ziel gefahren wurde. Besonders beteiligte Fehler umfassen vor allem Fahrzeugabmessungen und der Bias von Sensoren. Die genauen Zusammenhänge werden in Abschnitt 5.2.1 detaillierter erläutert.

Eine Erhöhung der Positionsungenauigkeit beeinflusste neben dem tolerierten Abstand zum Ziel auch die Trajektorie, da sie einen Rahmen vorgibt, in welchem die Trajektorie enden darf. Erhöhungen lassen der Trajektorie mehr Spielraum, wodurch generell höhere Werte für die DTW mit einer höheren Positionsungenauigkeit einhergehen. In den präsentierten Experimenten wurde daher versucht eine Balance zu finden, einen möglichst hohen Versatz der Trajektorie zu erhalten, während gleichzeitig bei der Zielungenauigkeit die Grenze von 0.5m nicht überschritten werden sollte.

Das Lenkverhalten wurde vorgegeben durch die abzufahrene Trajektorie. In vielen

Experimenten unterschied sich das fehlerbehaftete Lenkverhalten von der Struktur her kaum vom Lenkverhalten des Referenzdatensatzes. Die Peaks lagen in etwa an derselben Stelle, auch die verschiedenen Lenkphasen waren eindeutig identifizierbar. Beeinflusst wurde das Verhalten vor allem durch Sensor-induzierte Fehler. Beispielsweise führten ungenaue Sensor-Auflösungen zu Sprüngen. Ein hoher Rauschwert im Lenkwinkelsensor sorgte zudem für starke Lenkwinkelschwankungen und Unsicherheiten, welche sich jedoch nicht merklich in der Trajektorie niederschlugen.

5.1.4 Die Gewichtungsfunktion

Die initial vorgeschlagene Gewichtungsfunktion berechnete für jedes Fehlermaß den Abstand zu seinem entsprechenden Grenzwert. Danach erfolgte eine Normalisierung auf das Intervall $[0, 1]$. Ein Wert von 1 bedeutete, dass die Fehlergrenze exakt getroffen wurde, für jede Überschreitung der Schranke wurde der Funktionswert auf 0 gesetzt. Die initiale Argumentation dahinter war, dass die Fehlerschranke eine absolute Grenze darstellen sollte, welche nicht überschritten werden durfte. Gibt man eine Positionsungenauigkeit von 0.5m vor, so sorgt ein Wert von 0.51m schon für eine Invalidierung des Durchlaufs, selbst wenn die restlichen Maßzahlen die Schranken perfekt treffen würde.

Erste Experimente mit der initialen Grenzfunktion führten zu schlechten Ergebnissen, da viele Partikel, welche dem Optimum nahe kamen, verworfen wurden, zum Teil durch die in Abschnitt 5.1.2 beschriebenen probabilistischen Ausreißer. Gerade diese Partikel liefern jedoch die interessantesten Einsichten und würden zu einer Optimierung der Partikelmenge führen. So wurden lediglich Partikel, welche einen konsistent sicheren Abstand zu ihren Grenzen wahrten, in den folgenden Iterationen übernommen. Die resultierenden Partikelmengen besaßen ein entsprechend niedriges Gesamtgewicht und waren weitestgehend unbrauchbar.

Die in Kapitel 3.4.2 vorgeschlagene Dirac-Delta-Funktion umging dieses Problem, indem sie die Fehlergrenze als Peak der Gewichtungsfunktion ansah statt einer starren

Grenze. Die Höhe und die Schmalbandigkeit der Funktion konnten über den Parameter a gesteuert werden.

Ein Wert von 0.5 führte bereits zu guten Ergebnissen in der resultierenden Partikelmenge, was sich in einem zumeist hohen Durchschnittsgewicht und akzeptabler Diversität der Partikel äußerte. Experimente mit einem Wert von 0.25 führten zu einem schnelleren Anstieg der Gewichte der Partikelmenge während der ersten Iterationen, da die Funktion Partikel mit größerer Nähe zu den Fehlergrenzen um ein Vielfaches besser bewertete.

Weiterhin zu beobachten war, dass die Konsistenz der Partikelmenge einem durchweg hohen Niveau entsprach, was zu Lasten der Diversität der Menge ging. Faktoren kleiner als 0.25 führten dazu, dass die zufällig initialisierte Startmenge der Partikel fast gänzlich Gewichte gegen 0 erhielten, da die Funktion zu schmalbandig wurde. Der Anstieg der Gewichte erfolgte daher sehr langsam.

5.1.5 Ungenauigkeiten in der Modellierung

Das zugrunde liegende Fehlermodell erhebt keinen Anspruch auf Vollständigkeit. Es wurde versucht ein weitreichendes Spektrum verschiedener Fehlereinflüsse adäquat in der Software abzubilden. Aufgrund der Komplexität des Anwendungsbereichs der Software konnte jedoch nur ein Teilbereich betrachtet werden.

Umwelteinflüsse, Einflüsse durch Terrain, Wind und Wetter fehlen beispielsweise in der Modellierung. Verschiedene Teilschritte im Datenfluss der Assistenz wurden ebenfalls übersprungen, etwa die Übersetzung des Lenkradwinkels oder die Reaktionszeit des Fahrers.

Das allgemeine Fahrverhalten, welches im Rahmen dieser Arbeit angenommen wurde beinhaltet die sofortige Reaktion des Fahrers auf neue Reize und Informationen. Ziel dieser Arbeit war es nicht, ein kognitives Modell eines Fahrers zu entwickeln. Als Fahrermodell wurde daher ein sehr simples gewählt. Manche Fehlerarten gehen deshalb

etwas stärker ein als in der Realität, etwa die Wahrnehmung von Verzögerungen auf die Millisekunde genau. Eine striktere Betrachtung von Fehlern hat den Vorteil, dass die im tatsächlichen Fahrzeug vorgefundenen Bedingungen zur Nutzung der Assistenz um ein Vielfaches entspannter sind als im Kontext dieser Arbeit. Dies gibt dem Fahrer einerseits einen größeren Spielraum für Fehlerkorrekturen, andererseits aber auch für den falschen Gebrauch der Software.

Ein weiterer wichtiger Punkt ist die Annahme einer konstanten Fahrzeugbewegung. In Kapitel 3.3.2 wurde dies damit begründet, dass es durch den Mehrwert an Sicherheit, den die Assistenz liefert, möglich sein soll, zu navigieren ohne stehen bleiben zu müssen. Diese Annahme gilt in der Realität natürlich nicht. Der Einfluss vieler Verzögerungs-basierter Fehler kann dadurch in der Realität minimiert werden.

5.2 Evaluation der Experimente

Dieses Kapitel widmet sich der Evaluation der Experimente. Zunächst werden die resultierenden Trajektorien anhand ihrer Form und der Positionsgenauigkeit zum Ziel ausgewertet. Danach erfolgt eine Auswertung des Lenkverhaltens. Den Abschluss bildet die Zusammenfassung der Ergebnisse sowie Schlüsse, welche sich daraus ziehen lassen.

5.2.1 Auswertung der Trajektorien

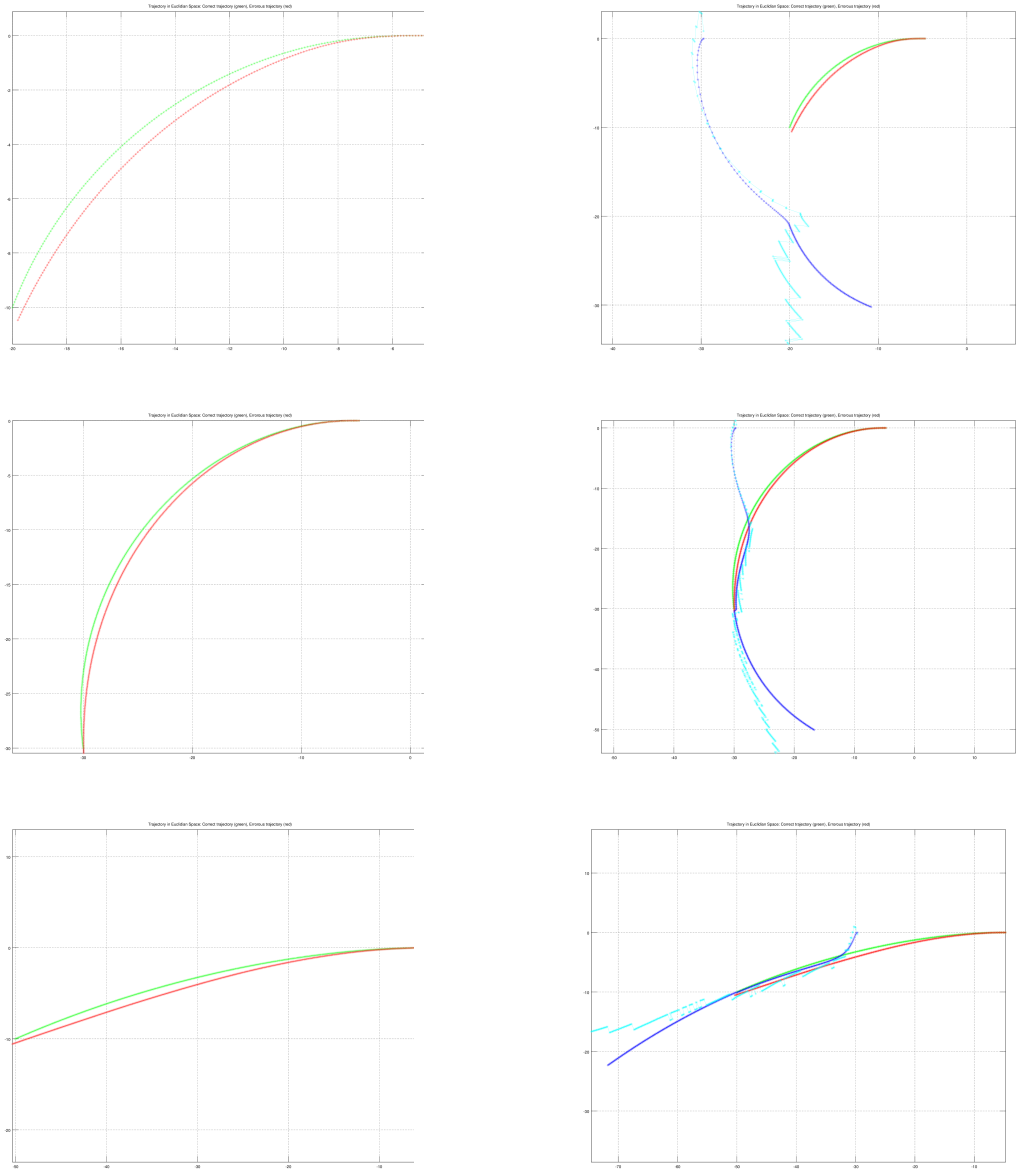


Abbildung 5.1: Die Trajektorien der durchschnittlichen Partikelgewichte. Links der Vergleich zwischen Referenzdatensatz (grün) und Experiment (rot). Rechts zusätzlich die Entwicklung des Endpunkts der Gespann-Trajektorie des Referenzdatensatzes (blau) und des Experiments (cyan).

Abbildung 5.1 zeigt auf der linken Seite die Trajektorien der drei durchgeführten Experimente. Die dargestellten Daten entstammen der in Abschnitt 5.1.1 beschriebenen bestmöglichen Ausführung des jeweiligen Manövers. Links ist die erste Fahraufgabe mit einem Zielpunkt von (-20, -10) zu sehen, der Zielpunkt in der Mitte beträgt (-30, -30), der rechte schließlich (-50, -10). Aufgrund des vom Testtool abweichenden Koordinatensystems von Octave werden die Trajektorien an der x-Achse gespiegelt dargestellt.

Die durchschnittlichen Fehlerkonfigurationen der Experimente finden sich in Tabelle 5.2.1.

Fehlerart	Exp1	Exp2	Exp3
mErrL1	0.02106	-0.05042	-0.01208
mErrL2	-0.01834	-0.05888	0.04481
mErrM1	-0.01779	0.00277	-0.01565
mErrCoupling-μ	1.34°	0.442°	0.7935°
mErrCoupling-σ	0.1°	0.0218°	0.0126°
mErrSteering-μ	1.213°	-0.0928°	-0.418 °
mErrSteering-σ	0.078°	0.1444°	0.3168 °
mResCoupling	1.166°	0.41°	0.673°
mResSteering	0.087°	0.755°	0.749°
mDelayCoupling	101ms	24.3ms	97.6ms
mDelayControl	0ms	0ms	0ms
mDelayFrame	125ms	182.8ms	147ms
mErrCameraPos-x	0.01013	-0.02372	0.03338
mErrCameraPos-y	-0.02427	0.01762	-0.03477
mErrCameraPos-z	-0.01321	-0.08459	-0.04032
mErrCameraOrient-x	-0.493°	0.208°	-0.576°
mErrCameraOrient-y	1.946°	1.122°	-0.611°
mErrCameraOrient-z	0°	0°	0°

Tabelle 5.3: Partikelmengen der drei Experimente.

Form der Trajektorien

Gemeinsam ist den Trajektorien, dass eine Linkskurve gefahren wird. Es ist zu beobachten, dass die fehlerbehafteten Kurven allesamt eine stärkere Krümmung besitzen

als die Kurven der Referenzdatensätze. Dies führt zu den gemessenen DTW-Werten von 0.37, 0.37 und 0.55 für die Ähnlichkeit der Trajektorien.

Grund hierfür ist eine Verschiebung der Endpunkte der Gespann-Trajektorie, welcher der Fahrer zum Zielen nutzt. Ein Vergleich zwischen korrekter und fehlerbehafteter Entwicklung der Endpunkte ist in Abbildung 5.1 rechts zu sehen. Die korrekten Endpunktkurven lassen sich in zwei Abschnitte unterteilen: der Abschnitt vor der stabilen Kreisfahrt und den Abschnitt danach. Letzterer zeichnet sich durch eine höhere Dichte der Punkte sowie die Bewegung auf einer annähernden Kreisbahn aus. Die fehlerbehafteten Endpunktkurven sind zum einen in ihren Start- und Übergangspunkten verschoben, zum anderen weisen sie viele Sprünge auf.

An dieser Stelle wird auf die Verschiebungen näher eingegangen. Diese lassen sich auf zwei Hauptfaktoren zurückführen.

Der erste Faktor ist eine zu geringe Krümmung der Gespann-Trajektorie, wodurch die Endpunkte der Trajektorie nach rechts wandern. Die Folge ist, dass ungewollt ein engeres Manöver gefahren wird. Aus Kapitel 3.2.2 ist bekannt, dass Fehler in der Fahrzeugvermessung sowie dem Einknickwinkel für einen Bias der resultierenden Gespann-Trajektorie in eine Richtung sorgen.

Im Falle einer Linkskurve mit negativem Einknickwinkel ist eine zu geringe Krümmung auf positive Fehler *mErrCoupling-μ* und *mErrL2* sowie negative Fehler in *mErrL1* und *mErrM1* zurückzuführen. Analog führen entgegengesetzte Fehlerwerte zu einer Erhöhung der Krümmung.

Betrachtet man die Fehlerkonfigurationen der Manöver, so fällt auf, dass der erste Datensatz eine sehr hohe Abweichung in *mErrCoupling-μ* von mehr als 1.3° besitzt. Die Fehler für *mErrL2* und *mErrM1* wirken sich mit jeweils knapp -1.8cm entgegen, gehen also kaum in den Endfehler ein. Der Fehler in *mErrL2* schwächt die Abweichung mit 2cm kaum merklich ab.

Datensatz 2 besitzt lediglich ein Drittel der Abweichung des ersten Datensatzes, zu-

dem findet keine entscheidende Verstärkung durch Fehler in der Fahrzeugvermessung statt. Der kombinierte Fehler liegt hier nur bei ca. 1cm.

Im dritten Datensatz verstärken sich erstmals die Fehler in der Fahrzeugvermessung sowie *mErrCoupling-*μ. Der kombinierte Fehler aus der Fahrzeugvermessung beträgt hier etwas mehr als 7cm. Das Resultat ist eine größere Abweichung der Trajektorie, was ein DTW-Wert von 0.55 bestätigt.

Der zweite verstärkende Faktor liegt der y-Ausrichtung der Kamera. In Kapitel 3.2.3 wurde beschrieben, dass eine Abweichung dieser Ausrichtung eins zu eins in den Rotationsfehler der Assistenz-Trajektorien übergeht. Soll eine Linkskurve gefahren werden, sorgt ein wie hier positiver Ausrichtungsfehler dafür, dass die Endpunkte der Gespann-Trajektorie noch weiter nach rechts wandern.

In den Datensätzen äußert sich dieser Fehler wie folgt: In ersten Datensatz erfolgt erneut eine Verstärkung des ohnehin schon stark ausgeprägten Bias nach rechts durch einen im Gegensatz zu den anderen Datensätzen hohen Wert von fast 1.95°. Durch Experimente konnte gezeigt werden, dass der Fehler sogar nahezu verdoppelt wird. Abbildung 5.2.1 zeigt eine Ausführung des ersten Manövers, in welcher lediglich das Vorzeichen der Kameraausrichtung invertiert wurde. Der Unterschied in den Trajektorien ist nunmehr kaum noch merklich.

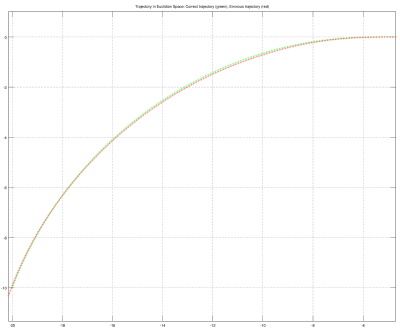


Abbildung 5.2: Abschwaechung von Fehlern in der Partikelmenge.

Auch im zweiten Datensatz erfolgt eine Verstärkung durch einen Wert von 1.1°.

Da der allgemeine Bias der Trajektorie in diesem Datensatz gering ist, wirkt sich gerade dieser Wert hier besonders stark auf die Krümmung der Trajektorie aus. Im dritten Datensatz erfolgt eine leichte Auslöschung durch einen Wert von -0.6°. Der Datensatz erzielt dennoch einen hohen Wert für die DTW, da sich die Fehler der Fahrzeugvermessung sowie des Einknickwinkels hinreichend verstärken.

Zusätzlich zu Verschiebungen wirken sich andere Fehler direkt auf das Lenkverhalten und damit die Form der Trajektorie aus.

Der erste wichtige Faktor ist die Verstellung der Spur, welche durch den Wert *mErrSteering-μ* zum Ausdruck gebracht wird. Eine Verstellung sorgt dafür, dass das Fahrzeug in eine Richtung zieht. Aus Kapitel 3.2.2 ist bekannt, dass positive Abweichungen im Lenkwinkel zu einem negativen Bias in der Einknickwinkeländerung führt. Mit anderen Worten bewirkt ein zu groß angenommener Lenkwinkel, dass der Einknickwinkel weniger stark (im positiven Wertebereich) bzw. stärker (im negativen Bereich) verändert wird als man beabsichtigt. Im Falle einer Linkskurve führt dieses Verhalten zu einer zu großen Einstellung des Einknickwinkels für die Kurvenfahrt und folglich zu der wahrgenommenen Abweichung.

Datensatz 1 besitzt einen sehr hohen Wert für *mErrSteering-μ*, welcher ca. 18.2° Abweichung des Lenkradwinkels entspricht. Das Fahrzeug zieht in diesem Fall stark nach rechts. Die Kurve wird folglich sehr viel enger genommen.

In den anderen beiden Datensätzen findet jeweils eine Abschwächung statt. Die Abweichung in Datensatz 2 ist dabei mit knapp -1.4° Lenkwinkelabweichung zu vernachlässigen. Datensatz 2 besitzt eine Abweichung von ca. -6.3°. Dass das Manöver dennoch eine gute Wertung der DTW erhält ist der Geradlinigkeit des Manövers zu verdanken. Auf diesen Fakt wird an anderer Stelle genauer eingegangen.

Das Lenkverhalten wird zusätzlich durch die Reaktion auf einen verzögerten Einknickwinkel beeinflusst. Ist *mDelayCoupling* zu hoch, reagiert der Nutzer auf veraltete Daten wie in Kapitel 3.2.3 beschrieben. Dieser Fehler tritt vor allem bei der stetigen Änderung des Einknickwinkels auf, etwa bei der Einfahrt in eine Kurve.

Fährt man eine Linkskurve, nähert man sich dem Ziel von rechts. Die Gespann-Trajektorien nähern sich im ersten Teilabschnitt ebenfalls von rechts wie Abbildung 5.1 zeigt. Durch die verzögerte Darstellung der Trajektorien wird der Nutzer dazu angehalten stärkere Lenkwinkel einzustellen, um die verlorengegangene Reaktivität der Gespann-Trajektorie zu kompensieren.

Fehler in *mDelayCoupling* sind nur in den Datensätzen 1 und 3 von Relevanz, wo sie jeweils im Bereich von 100ms liegen. Der Fehler sollte in Datensatz 3 allerdings weniger stark von Bedeutung sein als in Datensatz 1, da hier eine weniger starke Kurve gefahren wird und sich der Einknickwinkel kaum ändert im Laufe des Manövers. Es ist zu beachten, dass das im Rahmen der Experimente angenommene Reaktionsintervall 40ms beträgt. Der Wert des dritten Datensatz liegt daher unter der Wahrnehmungsschwelle des virtuellen Nutzers und kann somit vernachlässigt werden..

Positionsgenauigkeit

Bei der Positionsgenauigkeit können zwei Arten von Abweichungen unterschieden werden: der Versatz in x-Richtung und in y-Richtung des Fahrzeugs. Der Versatz in x-Richtung beschreibt die Längsverschiebung zwischen den beiden Fahrzeugen am Zielpunkt, während der Versatz in y-Richtung die Querverschiebung beschreibt. Die Querverschiebung kann größtenteils durch die Abweichungen in der Trajektorie erklärt werden.

Verantwortlich für die Längsverschiebung ist die Verzögerung des Kamerabilds. Fährt man ohne abzusetzen das Manöver zu Ende, so sieht man die Umgebung, welche durch die Rückfahrkamera aufgezeichnet wird, erst um einen Wert von *mDelayFrame* verzögert. Bei einer Geschwindigkeit von 9km/h entspricht dies einer zusätzlich zurückgelegten Strecke von 25, 56 und 24cm für die drei Manöver.

Abbildung 5.1 zeigt, dass bei allen Manövern ein relativ großer y-Versatz vorherrscht. Der Zielpunkt wurde ”überschossen”. Die Positionsfehler mit *mDelayFrame* lagen bei

0.51, 0.49 und 0.65. Experimente, in welchen die Kameraverzögerung ignoriert wurde bei ansonsten gleichbleibender Fehlerkonfiguration, führten zu Positionsverbesserungen von 51 auf 43cm, von 48.7 auf 5cm und von 64.7 auf 16cm Abstand zum Zielpunkt respektive. Vor allem im zweiten und dritten Manöver konnte somit eine erhebliche Verbesserung erzielt werden.

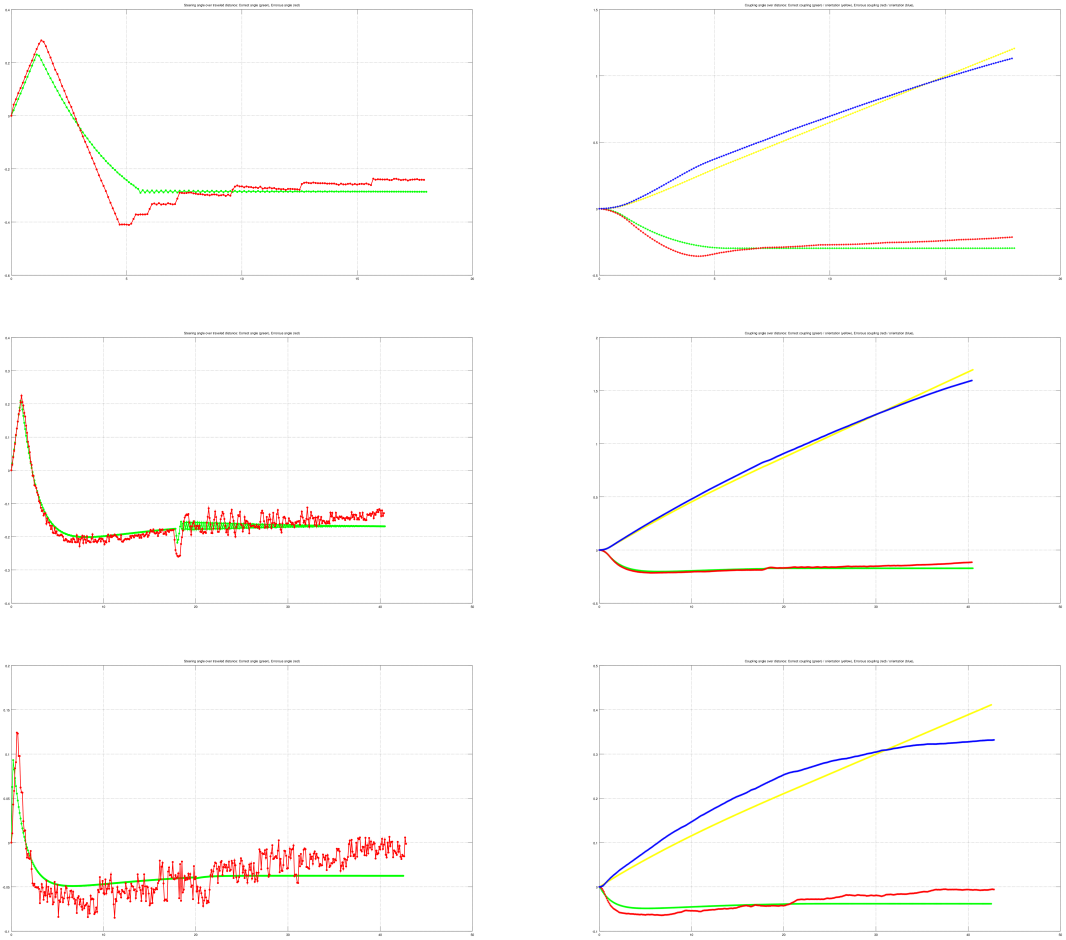
Der Grund hierfür war, dass bei diesen beiden Manövern eine erheblich niedrigere Querverschiebung vorherrschte, bedingt durch eine weniger ausgeprägte Verstärkung der einzelnen Fehler. Aufgrund der Länge der Manöver und der großen Navigationsfreiheiten war es dem Manöver-Controller so möglich die Abweichungen in der Gespann-Trajektorie zu kompensieren, wie man Abbildung 5.1 entnehmen kann. Er konnte schließlich nur durch ein "Überschießen" dazu gezwungen werden eine hohe Positionsungenauigkeit zu erreichen.

5.2.2 Auswertung des Lenkverhaltens

Abbildung 5.2.2 zeigt die zu den Manövern gehörigen Lenkverhalten sowie Einknickwinkel und Ausrichtungen des Anhängers. Bis zu einem gewissen Grad korrelieren die dargestellten Lenkwinkel-Entwicklungen mit den abgefahrenen Trajektorien. Gemeinsam ist den Datensätzen, dass zunächst ein Ausschlag nach rechts erfolgt, um einen geeigneten Lenkwinkel für die Linkskurve einzustellen. Danach wird der Winkel stetig reduziert und versucht auf der stabilen Kreisfahrt gerade zu halten. Es ist zu beobachten, dass in allen drei Experimenten der Lenkwinkel auf der stabilen Kreisbahn stetig ansteigt statt konstant zu bleiben. Dies ist eine Folge der Korrekturen, welche ein Fahrer anbringen muss, um den Bias der Gespann-Trajektorien nach rechts zu kompensieren.

Mehrere Beobachtungen zur Unsicherheit des Lenkverhaltens können in der Folge getroffen werden.

Der erste und der dritte Datensatz beschreibt ein höheres Lenkintervall als der jeweili-



ge Referenzdatensatz. Sowohl nach der ersten Lenkbewegung als auch beim Übergang auf die Kreisbahn wird der Lenkwinkel erst mit einer gewissen Verzögerung angepasst. Wie in Abschnitt 5.2.1 beschrieben ist dies das Resultat des Fehlers in *mDelayCoupling*. Eine verzögerte Darstellung führt zu einer verzögerten Lenkbewegung. Dieser Fehler ist bedingt durch die Zielvorgabe einer im Vergleich zum Referenzdatensatz erhöhten Standardabweichung der Lenkwinkel.

Sowohl der zweite als auch der dritte Datensatz sind sehr verrauscht, was eine Folge der im Gegensatz zum ersten Datensatz hohen Standardabweichung des Lenkwinkels ist. Ein Wert von jeweils knapp 0.75° entspricht einer Standardabweichung im Lenkradwinkel von mehr als 11°. Dies ist eine Folge der verlangten Steering Reversal Rate

von knapp 25% in beiden Fällen. Die verlangte Rate des ersten Manövers war deutlich geringer, was sich in der Absenz von unnötigen Lenkbewegungen niederschlägt.

Weiterhin zu beobachten ist der Einfluss von Sensor-Auflösungen auf das allgemeine Lenkverhalten. Aufgrund des Rauschens lassen sich aus Datensatz 2 und 3 wenige Schlüsse ziehen. Datensatz 1 veranschaulicht jedoch den Einfluss von Sprüngen in der Gespann-Trajektorie, bedingt durch einen Fehler von *mResCoupling*. Für jeden Übergang zwischen zwei minimal aufgelösten Sensorwerten des Einknickwinkels entsteht eine plötzliche Lenkbewegung, um die plötzlich auftretende Abweichung zu kompensieren. Statt eines stetigen Anstiegs erfolgt eine stufenweise Erhöhung des Lenkwinkels.

5.2.3 Bewertung

Vorgestellt wurden drei Experimente, welche jeweils unterschiedliche Fahrsituationen darstellten. Die daraus gewonnenen Erkenntnisse werden an dieser Stelle zusammengefasst. Im Rahmen dieser Arbeit wurde lediglich ein einziges Fahrzeugmodell betrachtet. Es wird daher versucht Tendenzen in den vorliegenden Ergebnissen zu erkennen, welche im Zusammenhang mit verschiedenen Aspekten des Fahrzeugmodells gebracht werden können. Dies erlaubt es eindeutige Aussagen zu Auswirkungen von Eingangsfehlern zu treffen. Aussagen zu konkreten Fehlergrenzen können für den konkreten Fall getroffen werden, werden jedoch für den allgemeinen Fall erschwert.

Zusammenfassung der Ergebnisse

Die Fehlergrenzen der Fahrsituationen unterschieden sich lediglich in den Vorgaben für das Lenkverhalten. Somit konnten konsistente Beobachtungen für die Positionsungenauigkeit getroffen werden.

Szenario 1 beschrieb eine enge Kurvenfahrt mit einem sehr kurzen Segment, in der der Punkt anvisiert wurde. Aufgrund des geringen Abstands zum Ziel sowie dem ge-

ringen Kreisradius eines fehlerfreien Manövers, musste eine erhebliche Verstärkung der einzelnen Fehler untereinander stattfinden, um die gewünschte Abweichung in der Trajektorie herbeizuführen. Den größten Anteil hatten hier *mErrCoupling-μ*, *mErrSteering-μ* sowie *mErrCameraOrient-y*. Selbst durch den daraus resultierenden erheblichen Bias der Assistenz-Trajektorien nach rechts konnte noch eine Positionsgenauigkeit von einem halben Meter erreicht werden. Durch die relativ geringe Länge des Manövers ist es dem Nutzer nicht möglich eine große Abweichung aufzubauen bevor er das Ziel erreicht hat. Somit lassen sich auch noch Fehler jenseits von 1° in *mErrCoupling-μ* und *mErrSteering-μ* sowie annähernd 2° *mErrCameraOrient-y* auf kurzer Strecke auffangen. Wie zuvor beschrieben korreliert die mögliche Abweichung der Trajektorie mit der Endposition. Schon die Korrektur eines der Werte, etwa der Spur, würde zu einer weniger starken Krümmung der Trajektorie und damit eine Annäherung an den Zielpunkt führen.

Die Absenz von Lenkwinkelschwankungen konnte durch die Stringenz der Kurvenfahrt erklärt werden, welche keine großen Schwankungen zuließ. Selbst der hohe Wert für *mResCoupling* sorgte lediglich für ein treppenartiges Lenkverhalten, nicht aber für eine Beeinflussung der Zielposition. Je näher man dem Ziel bereits ist, desto weniger gewinnt dieser Wert an Relevanz. Die geringe Länge des Manövers trug dazu bei.

Szenario 2 beschrieb eine weniger enge Kurvenfahrt mit einem längeren Segment, in welchem der Zielpunkt anvisiert werden konnte. Der erhöhte Abstand zum Ziel gab dem Fahrer hier mehr Spielraum in der Anfahrt des Punkts. Im Allgemeinen erfolgte bei diesem Manöver eine geringere Verstärkung der Fehler untereinander, zum Teil erfolgten sogar nachweisbare Abschwächungen, etwa durch *mErrL1* und *mErrL2*. Obwohl sich die Fehler hier weniger ausgeprägt verstärkten, lagen Abweichungen in der Trajektorie im selben Bereich wie beim ersten Manöver. Grund hierfür war, dass sich Abweichungen in der Trajektorie gegen Ende des Manövers korrigieren ließen. Die beobachtete Abweichung von fast 50cm war lediglich das Resultat des Fehlers in *mDelayFrame*. Zusätzlich zu weiteren geringen Fehlereinflüssen ließen sich eine

Abweichung im Einknickwinkel von 0.44° sowie ein Ausrichtungsfehler der Kamera von mehr als einem Grad somit kompensieren.

Selbst eine relativ hohe Auflösung des Lenkwinkels *mResSteering* sowie ein leichtes Rauschen wirkten sich nicht weiter auf die Positionierung aus. Lediglich das Lenkverhalten wurde etwas unsicher.

Szenario 3 beschrieb ein Manöver, in welchem der Zielpunkt annähernd gerade angefahren werden musste. Dieses Manöver ließ dem Fahrer die größten Freiheiten in der Anfahrt des Ziels. Die Verstärkung der Fehler lag in etwa auf einem Niveau mit dem zweiten Experiment. Durch die im Gegensatz zu den ersten beiden Szenarien große Länge des Manövers konnte sich jedoch der Fehler in der Abweichung der Trajektorie stärker aufbauen. Auch hier war es möglich, die Abweichungen der Trajektorie zu kompensieren. Analog zum zweiten Experiment ergab sich der Positionierungsfehler lediglich durch den hohen Wert von *mDelayFrame*. Dies bedeutet, dass selbst eine Abweichung von 0.75° sowie ein kombinierter Fehler in der Fahrzeugvermessung von 7cm noch zu einer guten Positionsgenauigkeit führt.

Schlussbetrachtung

Wie in den vorangegangenen Kapiteln betrachtet, wirken sich verschiedene Fehlerarten auf verschiedene Aspekte des Manövers aus. Die folgenden Schlüsse lassen sich ziehen.

Abschwächung und Verstärkung von Fehlern ist abhängig vom Fahrzeugmodell. Die Auswertung aus Abschnitt 5.2.1 stützt sich auf die in Kapitel 3.2.2 vorgestellte Analyse der Fehler im Fahrzeugmodell. Die Beeinflussung der Fehler untereinander ändert sich je nach Fahrzeugtyp, Achsenlängen und Kupplungspunkten. Aus Abschnitt 5.2.1 lässt sich zudem folgern, dass der Fehler in der Position der Trajektorien-Endpunkte mit zunehmender Länge der Gespann-Trajektorie ansteigt.

Der Wirkungsgrad von Fehlern ist abhängig vom durchgeführten Manöver. Eine Eingabefehlerkonfiguration äußert sich in verschiedenen Situationen auf verschiedene Arten. Wie der Vergleich von Experiment 1 und 2 zeigt, benötigt eine enge Kurve einen viel höheren Verstärkungsgrad der Fehlern als ein weitläufiges Manöver, um dieselben Ausgabe zu erhalten. Dies lässt den Schluss zu, dass stringente und kurze Manöver die Auswirkung von Fehlern einschränken. Im Umkehrschluss bedeutet dies, dass sich Fehler stärker äußern, je mehr Freiraum dem Fahrer zum Manövrieren zugestanden wird. Gleichzeitig zeigten Szenario 2 und 3, dass die Möglichkeit zum korrigieren von Fehlern gleichermaßen erhöht wird, wenn die Länge des Manövers steigt.

Positionierungsfehler und unsicheres Lenkverhalten sind weitestgehend unabhängig voneinander. In den drei Experimenten wurden die allgemeinen Lenkphasen des Manövers durch den Zielpunkt vorgegeben. Schwankungen, sofern nicht unverhältnismäßig, beeinflussten die resultierende Trajektorie und den Zielpunkt kaum. Abweichungen von der Trajektorie kamen durch deterministische Einflüsse wie die Tendenz der Assistenz-Trajektorien zustande, während das Lenkverhalten vor allem durch probabilistische und dynamische Fehler beeinflusst wurde wie durch Sensor-Auflösung bedingte Sprünge und Lenkwinkelrauschen.

Viele Parameter hatten wenig bis keinen Einfluss auf die Positionierung. Auffällig war in den drei Experimenten, dass die Positionsfehler der Kamera kaum relevant waren für den Erfolg des Manövers. Während das Rauschen im Lenkwinkel für ein unsicheres Fahrverhalten sorgte, ging erstaunlicherweise das Rauschen des Einknickwinkels ebenfalls kaum in die Betrachtung der Ausgabefehler mit ein.

Viele Fehler lassen sich auf einfache Art kompensieren bzw. vermeiden. Dieser Punkt soll besonders hervorgehoben werden. In Abschnitt 5.1.5 ist von den Ungenauigkeiten in der Modellierung die Rede, welche zu einer strikteren Betrachtung

einzelner Fehler führen als in der Realität vorhanden. Durch die Tatsache, dass der Fahrer Situationen interpretieren und entsprechend reagieren kann, ist es möglich verschiedene Fehlerarten durch eigene Korrekturen zu umgehen. Positionsungenauigkeiten durch zu große Kamera-Verzögerung können beispielsweise kompensiert werden, indem man regelmäßig anhält um seine Position zu prüfen. Verzögerungen in der Gespann-Trajektorie können auf dieselbe Art ausgeglichen werden. Es leidet lediglich der Fahrkomfort. Durch den Einfluss von menschlicher Reaktionszeit reagiert der Fahrer zudem nicht auf jede einzelne Störung in der Anhänger-Trajektorie, sondern orientiert sich nach der durchschnittlichen Richtung der Hilfslinie. Andere Fehler lassen sich durch Reparaturarbeiten, etwa die korrekte Einstellung der Spur, lösen. Regelmäßig kann zudem eine Neukalibrierung der Kamera erfolgen, um Verschleißerscheinungen entgegenzuwirken.

6 Zusammenfassung

Im Rahmen dieser Arbeit wurde untersucht, wie sich Fehler im Messmodell auf die Positionsgenauigkeit beim Rangieren mit einem Fahrerassistenzsystem auswirken. Dabei wurde besonderen Wert auf die Identifizierung von Fehlergrenzen gelegt.

Zunächst wurde eine quantitative Analyse der Auswirkung von Modellfehlern auf die Komponenten des Assistenz durchgeführt. Dazu gehörten das Fahrzeugmodell, Sensorik sowie der allgemeine Datenfluss der Assistenz-Software. Ein heuristisches Verfahren, genannt Annealed Particle Filter, wurde genutzt um einen großen Raum verschiedener Kombinationen von Fehlereinflüssen zu betrachten. Dazu wurde zunächst ein Controller entwickelt, mit dem sich ein Manöver mithilfe der visuellen Informationen des Assistenz simulieren ließ. In der Folge wurde eine Methode vorgestellt, mit deren Hilfe man das Manöver anhand definierter Fehlergrenzen bewerten konnte. Es wurde schließlich ein Testtool realisiert, mit dem sich Experimente durchführen ließen hinsichtlich der Auswirkung von Modellfehlern und tolerierbarer Fehlergrenzen.

Des Weiteren wurde das in Kooperation mit dem Fraunhofer Institut entwickelte Konzept vorgestellt zur Portierung der Assistenz auf das RODOS® -System. Im Rahmen dessen sind verschiedene Softwarekomponenten entstanden, welche die Einbindung und Kommunikation der Assistenz mit der Simulationsumgebung ermöglichen.

6.1 Bewertung

Der hier vorgestellte Ansatz zur Nutzung des Annealed Particle Filters, um Fehlergrenzen zu finden, stellt eine Neuerung dar. Es kann davon ausgegangen werden, dass im Rahmen dieser Arbeit aufgrund fehlender Referenzen nicht die optimale Konfiguration des Verfahrens genutzt wurde. Die vorgestellte Parametrisierung entstand anhand weniger Experimente. Aus zeitlichen Gründen war es nicht möglich eine ausführliche Betrachtung des Einflusses verschiedener Parametrisierungen durchzuführen. Die allgemeine Funktionalität des Verfahrens konnte jedoch gezeigt werden. Die resultierenden Partikelmengen waren nicht optimal, zeigten jedoch entscheidende Tendenzen, die für den Ausgabefehler verantwortlich zeigten. Dadurch war es möglich Schlüsse zu ziehen hinsichtlich Fehler-Wirkungs-Zusammenhängen sowie tolerierbaren Fehlergrenzen im Messmodell.

Die Wahl der Gewichte und Fehlergrenzen beeinflusste, anhand welcher Kriterien die Partikelmenge optimiert werden sollten. Auch hier war es nicht möglich umfassende Experimente mit verschiedenen Parametrisierungen durchzuführen. Ein weiterer Ansatzpunkt zur Verbesserung ist neben der Parametrisierung die Wahl weiterer oder besserer Qualitätsmaße zur Bewertung des Manövers. Im Allgemeinen zeigten die im Rahmen dieser Arbeit gewählten Maße jedoch gute Resultate.

Kritisch für den Erfolg der Partikelgewichtung war die Ausführung des Manövers anhand der Informationen der Assistenz. Hier mussten einige Vereinfachungen im betrachteten Fahrermodell getroffen wurden, was zu Ungenauigkeiten in der Modellierung führte. Positiv zu sehen war die Stabilität des Manöver-Controllers, welcher es stets schaffte das Ziel zu erreichen.

In der Zusammenarbeit mit dem Fraunhofer Institut war es indes nicht möglich die im Vorfeld gesteckten Ziele zu erreichen. Probleme traten vor allem in der Portierung statischer Bibliotheken auf, welche eine Fertigstellung des Systems verhinderten. Zufriedenstellend lief die Entwicklung der einzelnen Softwarekomponenten, deren

Funktionalität in Einzelversuchen nachgewiesen werden konnte.

6.2 Ausblick

Eine umfassende Evaluierung war aus Zeitgründen nicht möglich. Obwohl die Evaluation letztendlich zufriedenstellend verlief, ist die Notwendigkeit weiterer Testläufe gegeben. Ansatzpunkte hierfür sind das Testen verschiedener Parametrisierungen des heuristischen Verfahrens, verschiedener Trajektorienlängen und Fahrzeugmodelle.

Es kann überlegt werden das bestehende Fahrzeugmodell durch die Modellierung weiterer dynamischer Faktoren zu erweitern. So könnte beispielsweise eine Modellierung von Umwelteinflüssen wie Wetter, Witterung oder Terrain-spezifischen Einflüssen erfolgen, was zu einer realitätsnäheren Evaluation führen würde.

Ein weiterer Ansatzpunkt ist die Simulation von menschlichem Verhalten im Umgang mit der Assistenz. Im Rahmen dieser Arbeit wurde ein sehr simples Fahrermodell genutzt. Ein komplexeres Fahrermodell würde es erlauben Muster in der Nutzung der Assistenz zu entdecken, durch die potentielles Fehlverhalten identifiziert und diesem gegengewirkt werden kann.

Abschließend ist geplant, die im Rahmen dieser Arbeit begonnene Zusammenarbeit mit dem Fraunhofer ITWM zu einem für beide Parteien zufriedenstellenden Abschluss zu bringen. Dazu wird versucht die zum Ende der Arbeit aufgetretenen Probleme zeitnah zu lösen um die Assistenz schließlich in der Fahrerkabine installieren und testen zu können.

Literaturverzeichnis

[pet 2015] *petra - prototyping and evaluation of tractor reverse driving assistance.* http://www.uni-koblenz-landau.de/de/koblenz/fb4/ist/AGZoebel/Projects/petra, 2015

[Alt u. Godau 1995] Alt, Helmut ; Godau, Michael: Computing the Fréchet distance between two polygonal curves. In: *International Journal of Computational Geometry & Applications* 5 (1995), Nr. 01n02, S. 75–91

[Altafini 2001] Altafini, C.: Some properties of the general n-trailer. In: *International Journal of Control* 74 (2001), Nr. 4, S. 409–424

[Aust u. a. 2011] Aust, Mikael L. ; Regan, Michael A. ; Benmimoun, Mohamed: Disentangling the effects of advanced driver assistance system functions in Field Operational Tests: Recommendations from the European "euroFOT" project. (2011)

[Boas 2006] Boas, Mary L.: *Mathematical Methods in the Physical.* John Wiley & Sons., Inc, 2006

[Boem u. a. 2011] Boem, Francesca ; Pellegrino, FA ; Fenu, G ; Parisini, Thomas: Trajectory clustering by means of Earth Mover's Distance. In: *2011 International Federation of Automatic Control World Congress*, 2011, S. 4741–4746

[Bradski] Bradski, G.: In: *Dr. Dobb's Journal of Software Tools*

[Butterworth 1930] Butterworth, Stephen: On the theory of filter amplifiers. In: *Wireless Engineer* 7 (1930), S. 536–541

[Carsten u. Nilsson 2001] Carsten, OMJ ; Nilsson, L: Safety assessment of driver assistance systems. In: *European Journal of Transport and Infrastructure Research* 1 (2001), Nr. 3, S. 225–243

[Chung u. a. 2011] Chung, Woojin ; Park, Myoungkuk ; Yoo, Kwanghyun ; Roh, Jae I. ; Choi, Jongsuk: Backward-motion control of a mobile robot with n passive off-hooked trailers. In: *Journal of mechanical science and technology* 25 (2011), Nr. 11, S. 2895–2905

[Dantzig 1998] Dantzig, George B.: *Linear programming and extensions*. Princeton university press, 1998

[Deutscher u. a. 2000] Deutscher, Jonathan ; Blake, Andrew ; Reid, Ian: Articulated body motion capture by annealed particle filtering. In: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* Bd. 2 IEEE, 2000, S. 126–133

[Deutscher u. Reid 2005] Deutscher, Jonathan ; Reid, Ian: Articulated body motion capture by stochastic search. In: *International Journal of Computer Vision* 61 (2005), Nr. 2, S. 185–205

[Dirac] Dirac, P: A M 1958 The Principles of Quantum Mechanics. In: *Oxford: Clarendon) p* 180, S. 287–313

[Eiserloh 2013] Eiserloh, Christian T.: *Schnell wachsende Suchbäume zur Pfadplanung für allgemeine Gliederfahrzeuge*, Universität Koblenz-Landau, Campus Koblenz, Diplomarbeit, 2013

[Eiter u. Mannila 1994] Eiter, Thomas ; Mannila, Heikki: Computing discrete Fréchet distance / TU Vienna, Austria. 1994. – Forschungsbericht

[Forster 1983] Forster, Otto: *Analysis 1 Differential- und Integralrechnung einer Veränderlichen.* vieweg, 1983. – 87 S.

[Fuchs u. a. 2014] Fuchs, Christian ; Eggert, Simon ; Knopp, Benjamin ; Zobel, Dieter: Pose detection in truck and trailer combinations for advanced driver assistance systems. In: *Intelligent Vehicles Symposium Proceedings, 2014 IEEE* IEEE, 2014, S. 1175–1180

[Gordon u. a. 1993] Gordon, Neil J. ; Salmond, David J. ; Smith, Adrian F.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In: *IEE Proceedings F (Radar and Signal Processing)* Bd. 140 IET, 1993, S. 107–113

[Guennebaud u. a. 2010] Guennebaud, Gaël ; Jacob, Benoît u. a.: *Eigen v3.* http://eigen.tuxfamily.org, 2010

[Happe 2008] Happe, Markus: *Parallelisierung und Hardware-/ Software-Codesign von Partikelfiltern*, Universität Paderborn, Diplomarbeit, 2008

[Hummel u. a. 2011] Hummel, T ; Kühn, M ; Bende, J ; Lang, A: Advanced driver assistance systems: an investigation of their potential safety benefits based on an analysis of insurance claims in Germany. In: *German Insurance Association Insurers Accident Research, Research Report FS* 3 (2011)

[Isard u. Blake 1998] Isard, Michael ; Blake, Andrew: Condensation—conditional density propagation for visual tracking. In: *International journal of computer vision* 29 (1998), Nr. 1, S. 5–28

[Johansson u. a. 2004] Johansson, E ; Engström, J ; Cherri, C ; Nodari, E ; Toffetti, A ; Schindhelm, R ; Gelau, C: Review of existing techniques and metrics for IVIS and ADAS assessment. In: *Adaptive Integrated Driver Vehicle Interface (AIDE) Product number: IST-1-507674-IP* (2004)

[Julier u. Uhlmann 1997] Julier, Simon J. ; Uhlmann, Jeffrey K.: A new extension of the Kalman filter to nonlinear systems. In: *Int. symp. aerospace/defense sensing, simul. and controls* Bd. 3 Orlando, FL, 1997, S. 3–2

[Kalman 1960] Kalman, Rudolph E.: A new approach to linear filtering and prediction problems. In: *Journal of Fluids Engineering* 82 (1960), Nr. 1, S. 35–45

[Kantorovich 1942] Kantorovich, Leonid V.: On the translocation of masses. In: *Dokl. Akad. Nauk SSSR* Bd. 37, 1942, S. 199–201

[Kelly 2002] Kelly, Alonzo: General solution for linearized stochastic error propagation in vehicle odometry. In: *Preprints 15th IFAC World Congress*, 2002

[Kleer u. a. 2014] Kleer, Michael ; Gizatullin, Andrey ; Dressler, Klaus ; Müller, Steffen: Real-time human in the loop MBS simulation in the Fraunhofer Robot-Based Driving Simulator. In: *Archive of Mechanical Engineering* 61 (2014), Nr. 2, S. 270–285

[Knopp 2010] Knopp, Benjamin: *Modellbasierte Poseschätzung von Menschen aus dichten Volumendaten*, Universität Koblenz-Landau, Campus Koblenz, Diplomarbeit, 2010

[Liu u. a. 1999] Liu, Yung-Ching ; Schreiner, Christopher S. ; Dingus, Thomas A.: Development of human factors guidelines for advanced traveler information systems (ATIS) and commercial vehicle operations (CVO): Human factors evaluation of the effectiveness of multi-modality displays in advanced traveler information systems. 1999. – Forschungsbericht

[MacCormick u. Isard 2000] MacCormick, John ; Isard, Michael: Partitioned sampling, articulated objects, and interface-quality hand tracking. In: *Computer Vision—ECCV 2000*. Springer, 2000, S. 3–19

[Macdonald u. Hoffmann 1980] Macdonald, Wendy A. ; Hoffmann, Errol R.: Review of relationships between steering wheel reversal rate and driving task demand. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 22 (1980), Nr. 6, S. 733–739

[MATLAB 2014] MATLAB: *version 8.3.0 (R2014a)*. Natick, Massachusetts : The MathWorks Inc., 2014

[Mclean u. Hoffmann 1971] Mclean, John R. ; Hoffmann, Errol R.: Analysis of drivers' control movements. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 13 (1971), Nr. 5, S. 407–418

[Metropolis u. a. 1953] Metropolis, Nicholas ; Rosenbluth, Arianna W. ; Rosenbluth, Marshall N. ; Teller, Augusta H. ; Teller, Edward: Equation of state calculations by fast computing machines. In: *The journal of chemical physics* 21 (1953), Nr. 6, S. 1087–1092

[Monge 1781] Monge, G: Mémoire sur la théorie des déblais et remblais, Mémoires Acad. In: *Royale Sci. Paris* 3 (1781)

[Orten 2015] *Webseite der Firma.* http://www.orten.com, 2015

[Östlund u. a. 2004] Östlund, J ; Nilsson, L ; Carsten, O ; Merat, N ; Jamson, H ; Jamson, S ; Mouta, S ; Carvalhais, J ; Santos, J ; Anttila, V u. a.: Deliverable 2-HMI and safety-related driver performance. Human Machine Interface And the Safety of Traffic in Europe. In: *Project HASTE GRD1/2000/25361 S* 12 (2004)

[Östlund u. a. 2005] Östlund, Joakim ; Peters, Björn ; Thorslund, Birgitta ; Engström, Johan ; Markkula, Gustav ; Keinath, Andreas ; Horst, Dorit ; Juch, Susann ; Mattes, Stefan ; Foehl, Uli: Driving performance assessment-methods and metrics. (2005)

[Project 2013] Project, Qt: *Qt v4.8.5.* https://qt-project.org, 2013

[Roh u. Chung 2010] Roh, Jaeil ; Chung, Woojin: Reversing control of a car with a trailer using a Driver Assistance System. In: *Advanced Robotics and its Social Impacts (ARSO), 2010 IEEE Workshop on* IEEE, 2010, S. 99–104

[Sang u. a. 2011] Sang, Feng ; Xingbin, Chen ; Shuohua, Chen ; Xiao, Lu: Application and evaluation about obstacle edge extraction technology in the parking assistant system. In: *International Journal of Computational Intelligence Systems* 4 (2011), Nr. 6, S. 1342–1349

[Schwarz 2009] Schwarz, Christian: *Entwicklung eines Regelungsverfahrens zur Pfadverfolgung für ein Modellfahrzeug mit einachsigem Anhänger*, Universität Koblenz-Landau, Diplomarbeit, Februar 2009

[Smith u. a. 1962] Smith, Gerald L. ; Schmidt, Stanley F. ; McGee, Leonard A.: *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle.* National Aeronautics and Space Administration, 1962

[Vestri u. a. 2005] Vestri, Christophe ; Bougnoux, Sylvian ; Bendahan, Reny ; Fintzel, Katia ; Wybo, Seba ; Abad, Fred ; Kakinami, Toshiakai: Evaluation of a vision-based parking assistance system. In: *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE* IEEE, 2005, S. 131–135

[Zöbel 2013] Zöbel, Dieter: *Mathematical Modeling of the Kinematics of Vehicles.* 2013. – unveröffentlicht