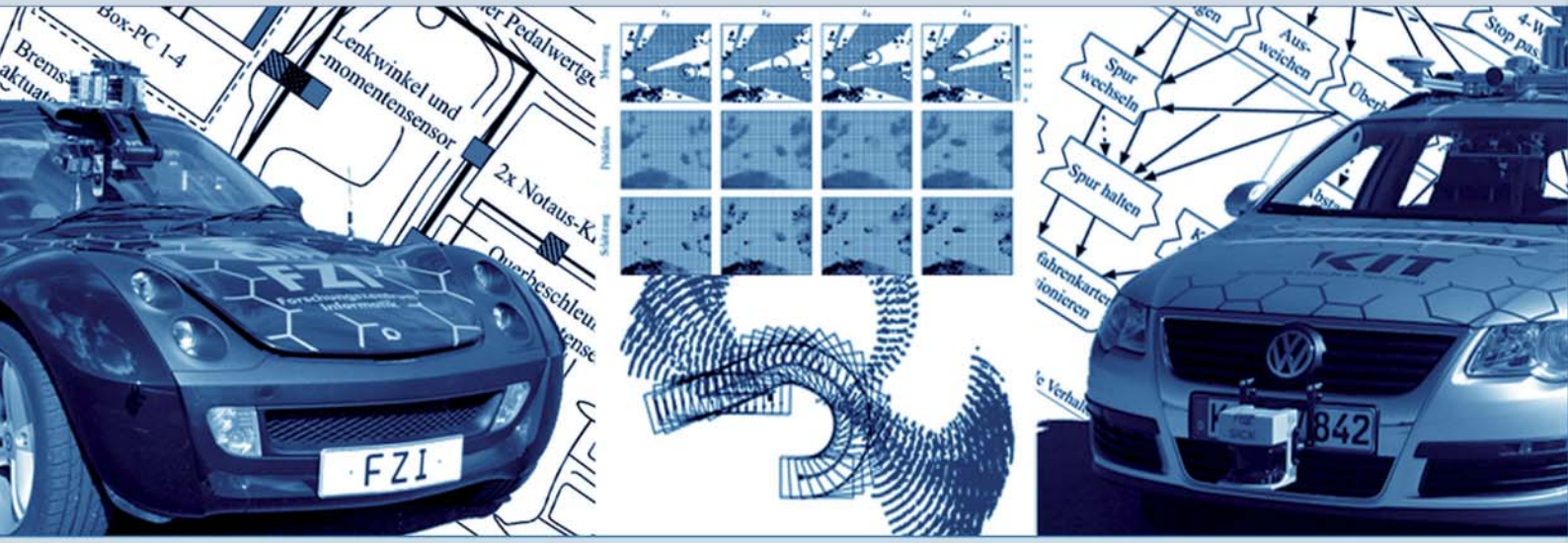
Joachim Schröder



Adaptive Verhaltensentscheidung und Bahnplanung für kognitive Automobile

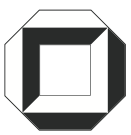
universitätsverlag karlsruhe

Joachim Schröder

Adaptive Verhaltensentscheidung und Bahnplanung für kognitive Automobile

Adaptive Verhaltensentscheidung und Bahnplanung für kognitive Automobile

von
Joachim Schröder



universitätsverlag karlsruhe

Dissertation, Universität Karlsruhe (TH)
Fakultät für Informatik
Tag der mündlichen Prüfung: 29.05.2009

Impressum

Universitätsverlag Karlsruhe
c/o Universitätsbibliothek
Straße am Forum 2
D-76131 Karlsruhe
www.uvka.de

Universitätsverlag Karlsruhe 2009
Print on Demand

ISBN: 978-3-86644-406-5

Adaptive Verhaltensentscheidung und Bahnplanung für kognitive Automobile

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik

der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von

Joachim Schröder

aus Tübingen

Tag der mündlichen Prüfung:	29. Mai 2009
Erster Gutachter:	Prof. Dr.-Ing. Rüdiger Dillmann
Zweiter Gutachter:	Prof. Dr.-Ing. Georg Färber

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Technische Informatik an der Universität Karlsruhe (TH). Allen voran möchte ich meinem Doktorvater, Herrn Prof. Dr.-Ing. Rüdiger Dillmann, für das Vertrauen danken, das er in mich gesetzt hat, für seinen fachlichen Rat und für die wissenschaftliche Freiheit, die er mir während meiner Zeit am Institut gewährt hat. Besonders danke ich ihm für seine Ermutigung und Unterstützung zur Durchführung meiner Diplomarbeit an der Vanderbilt University in Nashville, Tennessee, wo ich außerdem das Glück hatte, meine spätere Ehefrau kennenzulernen. Herrn Prof. Dr.-Ing. Georg Färber bin ich für die bereitwillige Übernahme des Korreferats, das Interesse an meiner Arbeit und die wertvollen Hinweise dankbar.

Meinem Mentor und langjährigen Kollegen Tilo Gockel danke ich für seine wertvollen Tipps in allen Lebenslagen, die gute Zusammenarbeit in vielen Industrieprojekten und seine ständig neuen Ideen und Vorhaben, an denen er auch andere gerne teilhaben lässt. Ohne seine Unterstützung wäre mein Buchprojekt nie realisiert worden.

Steven Wieland hat sich bereits bei seiner Studienarbeit am Institut als hilfsbereiter Kollege und *Macher* präsentiert. Ohne ihn würde dem Institut ein echter Leistungsträger fehlen, und auch der Smart Roadster hätte jetzt noch keine automatische Bremse. Meinen Kollegen Dr. Töff und Dr. Azad danke ich für erfrischende Unternehmungen abseits des Institutsalltags, für Versuche, einem Maschinenbauer die Bildverarbeitung näher zu bringen und natürlich auch für den Hafer- und Bananen-Blues. Mein Kollege Alex Bierbaum verdient Dank für die bereitwillige Hilfe bei elektronischen Problemen und die Durchführung der gemeinsamen Lehrveranstaltungen.

Mit den Kollegen aus der Automobilgruppe und dem Sonderforschungsbereich SFB/TR28 habe ich die letzten drei Jahre intensiv zusammengearbeitet. Ihnen gebührt deshalb ein besonders großes Dankeschön. Stefan Vacek möchte ich für lange Diskussionen über kognitive Fähigkeiten von Automobilen danken, für die Unterstützung bei der Schärfung meines Themas und für die vielen gemeinsamen Fahrten zu den SFB-Treffen nach München. Als Nachfolger von Stefan und mir bringen Tobias Gindele und Sebastian Brechtel neuen Wind in die Automobilgruppe. Beide sind mir in ihren Fachbereichen mittlerweile weit voraus und es besteht kein Zweifel daran, dass es in den nächsten Jahren gelingen wird, ein intelligentes Fahrzeug mithilfe eines Fahrlehrers zu trainieren.

Daniel Jagszent hat als Studienarbeiter, Hiwi und Diplomand in der Automobilgruppe mitgearbeitet und wurde durch seine Fach- und Programmierkenntnisse schnell unersetzlich. Vielen Dank für den großen Einsatz, ohne den wir das Finale der Urban Challenge 2007 sicher nicht erreicht hätten. Die Teilnahme an diesem Wettbwerb mit Team AnnieWay war sicherlich das Highlight meiner Arbeit am ITEC. Ich danke allen Beteiligten für drei verrückte Monate in Mountain View, eine kollegiale Atmosphäre trotz des großen Zeitdrucks und für nächtliche Testfahrten bei Nob Hill. An dieser Stelle sind neben meinen

Institutskollegen stellvertretend für das gesamte Team auch Moritz Werling, Sören Kammel und Julius Ziegler zu nennen. Auch den Kollegen aus den anderen Gruppen möchte ich für die gute Zusammenarbeit danken und für die Chance, über den Tellerrand des eigenen Themas hinaus zu blicken. In alphabetischer Reihenfolge sind dies Tamim Asfour, Martin Do, Dominik Fritz, Dilana Hazer, Paul Holz, Martin Lösch, Sascha Seifert, Steffi Speidel, Peter Steinhaus, Gunther Sudra, Roland Unterhinninghofen, Niko Vahrenkamp und Kai Welke. Danken möchte ich auch meinen FZI-Kollegen Marius Zöllner, Thilo Kerscher, Thomas & Thomas, Clemens Birkenhofer und Kristian Regenstein. Ihr habt den Smart wieder zurückbekommen, bitte sorgt gut für ihn!

Besonderer Dank gebührt den Damen unseres Sekretariats, Nela Redzovic, Christine Brand, Isabelle Löber und Beatrix Jancic für die professionelle Organisation und Führung unseres Institutes. Sie helfen den Mitarbeitern und Studenten zahlreiche organisatorische Hürden zu meistern, haben immer ein offenes Ohr für kleine und große Probleme und engagieren sich weit über das erforderliche Maß hinaus. Auch die „Alteingesessenen“ sollen nicht unerwähnt bleiben. Sie haben das Insitut teilweise schon lange verlassen, waren zu Beginn meiner Mitarbeit jedoch eine große Stütze: Björn Giesler, Oliver Rogalla, Markus Ehrenmann, Raoul Zöllner und Tobias Salb.

Bedanken möchte ich mich auch ganz herzlich bei allen meinen Hiwis, Studienarbeitern und Diplomanden, ohne die die Umsetzung dieser Arbeit nicht möglich gewesen wäre. Stellvertretend für viele andere danke ich Stephan Riedel, Ergin Gündüz, Kosta Papadopoulos und Felix Mayer. Meinem langjährigen Hiwi Udo Müller danke ich insbesondere für die nächtlichen Schraubeinheiten in der FZI-Tiefgarage.

Meinen Eltern Susanne und Detlef Schröder danke ich dafür, dass sie mir eine sehr gute Ausbildung ermöglicht haben und mich bei allen meinen Entscheidungen, ob beruflicher oder privater Natur, mit großer Selbstverständlichkeit unterstützt haben. Meinen Schwiegereltern und meinem Schwager danke ich für die herzliche Aufnahme in ihre Familie, für wundervolle Erlebnisse in Almaty und für eine unvergessliche Hochzeitsfeier.

Der Karlsruher Gang danke ich für tolle Studienjahre, die enge Freundschaft, die sich daraus entwickelt hat, die jährlichen Gardasee-Touren und letztendlich die Geduld, die sie für mich aufgebracht hat. Auch nach vielen Absagen meinerseits an gemeinsame Radrunden wurden die Jungs nie müde, mich wieder und wieder zu fragen.

Meiner Ehefrau Damira danke ich dafür, dass sie mich insbesondere während der Promotionszeit mit viel Liebe, Geduld und Verständnis unterstützt und immer wieder motiviert hat. Ohne Sie hätte ich diese Ausarbeitung wahrscheinlich nie fertig bekommen. Dafür widme ich ihr die vorliegende Arbeit.

Karlsruhe,
den 28. Juli 2009

Joachim Schröder

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Beschreibung und Motivation

Fahrerassistenzsysteme wie ABS oder ESP haben dazu beigetragen, die Anzahl an Verkehrsunfällen zu verringern und die Sicherheit im Straßenverkehr nachhaltig zu erhöhen. Vor dem Hintergrund eines steigenden Verkehrsaufkommens wird nun der Wunsch erkennbar, nicht nur in Notsituationen Fahrer zu unterstützen, sondern in mittlerer bis ferner Zukunft Fahrzeuge (teil-)autonom zu führen. Dies kann dazu beitragen, die Sicherheit weiter zu erhöhen und die Effizienz im Verkehrsfluss durch eine bessere Abstimmung erheblich zu steigern. Wettbewerbe mit autonomen Roboterfahrzeugen in Europa und den USA haben gezeigt, dass diese Vision in der Forschung bereits teilweise umgesetzt werden kann.

Für einen Einsatz im Straßenverkehr muss ein Verständnis für komplexe Verkehrssituationen vorhanden sein, auf dessen Grundlage Entscheidungen herbeigeführt werden. Aus dem Forschungsgebiet der mobilen Robotik sind Erfahrungen vorhanden, wie solche Entscheidungen in kognitiven Systemen getroffen und wie Fahrmanöver ausgeführt werden können. Es ist zu untersuchen, inwieweit sich diese Methoden auch für kognitive Automobile verwenden lassen, speziell im Hinblick auf die besonderen Sicherheitsanforderungen im Straßenverkehr. Die Frage, wie Sicherheit für ein kognitives Automobil realisiert und nachgewiesen werden kann, ist ebenfalls bislang unbeantwortet. Sie ist von zentraler Bedeutung, um ein solches System am Markt anbieten zu können.

Diese Arbeit ist im Rahmen des Sonderforschungsbereiches SFB/TR28 *Kognitive Automobile* entstanden. Sie setzt unter anderem die Aufgabenstellung des Teilprojektes B2 *Verhaltensentscheidung und Bahnplanung* um und ist eng an die Interpretationskomponente des Teilprojektes B1 *Situationsinterpretation und Verhaltenserkennung* gekoppelt (vgl. Abb. 1.1). Nach Analyse und Bewertung der von den Wahrnehmungskomponenten bereitgestellten Daten wird in der Situationsinterpretation durch einen Schlussfolgerungsprozess ein Abbild der aktuellen Situation erzeugt [Vacek 07]. Die um Objektrelationen und erkannte Verhalten von Verkehrsteilnehmern erweiterten Daten werden an die in dieser Arbeit vorgestellten Komponenten weitergereicht und bilden die Grundlage für eine Entscheidung.

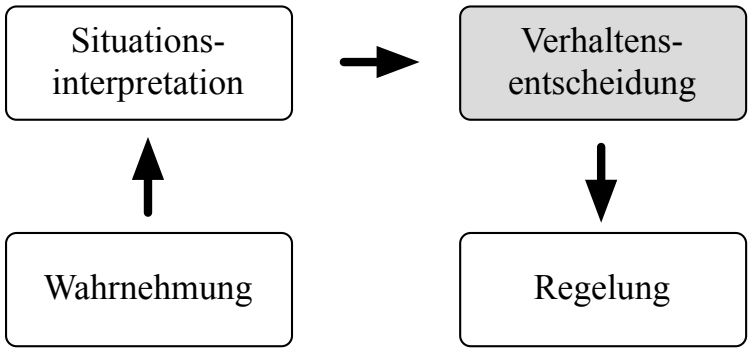


Abbildung 1.1: Einordnung der Verhaltensentscheidung innerhalb der Systemarchitektur des Sonderforschungsbereiches SFB/TR 28.

Die Fragestellungen, welche sich im Anschluss an eine Interpretation ergeben, lassen sich wie folgt formulieren:

1. Welches Verhalten ist in der gegenwärtigen Situation zulässig und am geeignetsten?
2. Wie kann aus den Vorgaben aktiver Einzelverhalten eine optimale Bahn geplant werden?
3. Wie kann die sichere Ausführung auf mehreren Ebenen gewährleistet werden?

Zunächst sind zur Beantwortung der ersten Frage Vorbedingungen der Einzelverhalten zu überprüfen, welche im Grunde die StVO abbilden. Es erfolgt eine Auswahl der besten der aktuell zulässigen Fahrhandlungen. Die Vorgaben mehrerer Einzelverhalten müssen nun in geeigneter Weise fusioniert werden, um dann die zweite Frage beantworten zu können und eine optimale Bahn zu erzeugen. Durch die Einführung einer reaktiven Ebene, sowie einer geeigneten Beschränkung der Bahnplanung wird eine Gewährleistung von unmittelbarer Sicherheit ermöglicht, um eine Antwort auf die dritte Frage zu geben.

Aufgrund der hohen Situationsvielfalt im Straßenverkehr ist langfristig ein Hinzulernen, und damit eine kontinuierliche Verbesserung, wünschenswert. Obgleich Lernverfahren nicht Inhalt dieser Arbeit sind, so ist eine spätere Möglichkeit zur Adaption bereits bei der Auswahl der Verhaltensarchitektur zu berücksichtigen. Ferner sollen interne Daten der Verhaltensausführung in geeigneter Form hinterlegt werden und als Grundlage für eine Bewertung dienen.

Die grundlegenden Herausforderungen dieser Arbeit sind wie folgt:

- Modellierung von Fahrverhalten in einer für den Menschen nachvollziehbaren Struktur aus Gründen der Transparenz und Erweiterbarkeit.
- Korrekte Umsetzung von Vorgaben der Verhaltenssteuerung.

- Umsetzung hoher Sicherheitsanforderungen bei Auswahl und Ausführung von Verhalten auf verschiedenen Systemebenen.
- Möglichkeit zur Anhäufung von Erfahrungswissen für stetige Verbesserungen des Systems.

1.2 Zielsetzung und wissenschaftlicher Beitrag

Die Zielsetzung dieser Arbeit besteht darin, eine Verhaltenssteuerung im Sinne eines kognitiven Automobils zu entwickeln. Aus Gründen der Nachvollziehbarkeit, dem besseren Verständnis und eventuell der späteren Möglichkeit eines Belehrens durch einen Fahrlehrer soll sich die vorgeschlagene Steuerungsarchitektur mit geeigneten Repräsentationen an die Bedürfnisse des Menschen anlehnen. Weiterhin soll eine universell einsetzbare Bahnplanungskomponente entwickelt werden, welche sowohl im städtischen Verkehr, als auch auf Überlandstraßen und Autobahnen verwendet werden kann. Der Schnittstelle zwischen Verhaltenssteuerung und Bahnplanung kommt dabei eine besondere Bedeutung zu. Bei der Schnittstellenwahl ist insbesondere den Fragen der Sicherheit, der dynamischen Umfeldrepräsentation und der Fusion von Verhaltensausgaben Rechnung zu tragen. Für die Wissenserweiterung ist eine geeignete Form zu finden, um abstrakte Messdaten als Erfahrungswissen abzulegen.

Der wissenschaftliche Beitrag besteht darin, eine biologisch motivierte, verhaltensbasierte Architektur zur Steuerung eines kognitiven Automobils zu verwenden und die im Straßenverkehr geforderte Sicherheit auf verschiedenen Ebenen der Architektur zu gewährleisten. Durch das Prinzip der virtuellen Sensoren wird ein hohes Maß an Rückmeldung erreicht und damit die Grundlage gebildet, um zukünftig Lernverfahren zu integrieren. Ein Fahrzeug hätte damit zum ersten Mal die Möglichkeit, eigene Fahrmanöver zu beurteilen.

Mit der Repräsentationsform einer Gefahrenkarte wird ein neuartiges Verfahren vorgestellt, um sowohl das Fahrzeugumfeld als auch die Intentionen der Verhaltenssteuerung darzustellen. Das Fahrzeugumfeld wird dabei zum ersten Mal sowohl durch sensorische Fragmente mit Geschwindigkeitsinformation als auch durch korrekt erkannte Objekte und deren Bewegungsmuster zusammengesetzt. Diese dynamische Repräsentation erlaubt eine effektive Nutzung aller vorhandenen Informationen und eine Prädiktion des gesamten Fahrzeugumfeldes.

1.3 Lösungsansätze

Die wichtigsten erarbeiteten Lösungsansätze sind:

- **Modellierung mit Verhaltensnetzwerken**
  Für die Modellierung der Fahrverhalten werden sogenannte Verhaltensnetzwerke verwendet. Durch die generischen Schnittstellen wird die Interaktion zwischen den

Verhalten von der Physik entkoppelt und eine spätere Adaption im Rahmen von *Hinzulernen* vereinfacht. Ein zwischen die strategische und taktische Ebene der Ausführung geschaltetes Modul *Szenario Manager* integriert Regelprimitive, welche sich aus den Vorbedingungen und der StVO ergeben und dient der Nachvollziehbarkeit von Entscheidungen. Auf unterer Ebene des Verhaltensnetzwerkes setzen reaktive Verhalten die momentanen Fahrhandlungen, aber auch Notfunktionen wie Kollisionsvermeidung um und erzeugen Vorgaben für die Bahnplanung.

- **Szenario Monitor**
  Als Schnittstelle zur Situationsinterpretation wird ein sogenannter *Szenario Monitor* entwickelt. Dieser dient für Verhalten der strategischen Ebene als Fassade zur Interpretation und bereitet deren Informationen auf. Der Szenario Monitor dient als zentrale Instanz zur Hinterlegung der Verkehrsregeln und stellt eine Abstraktionsebene zwischen strategischen Verhalten und den tatsächlichen Fahrhandlungen (taktischen Verhalten) dar. Die vor der Ausführung einer Fahrhandlung notwendigen Vorbedingungen werden vom Szenario Monitor in Form von *Durchführbarkeitstests* überprüft und es werden mögliche Alternativen bereitgehalten. Durch die zentrale und vorherrschende Stellung dieser Komponente können Entscheidungen im Detail nachvollzogen werden.

- **Dynamische Umfeldrepräsentation**
  In der Verarbeitungskette an erster Stelle werten Wahrnehmungskomponenten Sensorinformationen aus und liefern erkannte Objekte an eine symbolische Ebene. Diese Daten dienen – neben einer Verwendung in der Situationsinterpretation und Verhaltensentscheidung – auch als Grundlage für die Bahnplanung. Eine dynamische Repräsentation der Umgebung mit Schätzung der Objektbewegungen für einen geringen Zeithorizont ist auf dieser Grundlage möglich. Aufgrund widriger Umgebungsbedingungen, ungünstiger Konstellationen oder fehlerhafter Wahrnehmungsalgorithmen kann keine Garantie für eine Objektrepräsentation und -klassifikation aller Verkehrsobjekte gegeben werden.

  Die hier entwickelte dynamische Umfeldrepräsentation kombiniert Informationen aus der symbolischen Ebene mit Belegtheitsinformationen der subsymbolischen Zellebene. Letztere werden durch ein *Bayesian Occupancy Filter (BOF)* erzeugt, das neben der Belegtheit auch eine Geschwindigkeitsverteilung für einzelne Zellen schätzt und damit die Möglichkeit zur Prädiktion von Zellbewegungen eröffnet. Mit dieser neuartigen Methode werden alle zur Verfügung stehenden Informationen bestmöglichst genutzt, sodass auch im Falle einer unzureichenden symbolischen Darstellung Bahnen auf Grundlage der Zellprädiktion sicher geplant werden können.

- **Bahnplanung auf Gefahrenkarten**
  Die Bahnplanung erfolgt mithilfe einer Graphensuche auf einer Gefahrenkarte, wobei

eine Verwendung von Fahrzeugmodellen sicherstellt, dass die erzeugte Bahn fahrbar ist. Die Gefahrenkarte entsteht durch die im vorigen Punkt aufgelistete Methode der dynamischen Umfeldrepräsentation. Neben einer effektiven Nutzung der Umfeldinformationen erlauben Gefahrenkarten die Restriktion auf zulässige Bereiche und dadurch einen Ausschluss bestimmter Fahrmanöver.

Eine Erweiterung um den Zustand *Geschwindigkeit* erlaubt das Planen der Trajektorien. Diese zusätzliche Dimension wird in ihrer Vielfalt durch geeignete Mechanismen eingeschränkt, indem Geschwindigkeitsprofile anhand markanter Kartenpunkte erzeugt werden oder oftmals auftretende Verläufe hinterlegt werden. Aufgrund der Sicherheitsanforderungen erfolgt eine mehrstufige Trajektorienplanung. Erst nachdem eine sichere Trajektorie geplant wurde, kommen Optimierungskriterien wie bspw. eine Minimierung der Fahrzeit zum Einsatz.

1.4 Aufbau und Kapitelübersicht

Die Arbeit ist entsprechend dem Informationsfluss in der Steuerungsarchitektur in Abbildung 1.1 strukturiert. Zunächst wird in Kapitel 2 der Stand der Forschung im Bereich des *Autonomen Fahrens* dargelegt. Neben den aktiven Forschergruppen liegt das Hauptaugenmerk in diesem Kapitel auf vorhandenen Arbeiten zu Systemarchitekturen, Verhaltensentscheidung und Bahnplanung für kognitive Automobile. Ein Fazit fasst den Stand zusammen.

In Kapitel 3 wird eine Verhaltenssteuerung für ein kognitives Automobil entworfen und umgesetzt. Zunächst wird auf die Anforderungen an eine solche Steuerung eingegangen. Es folgt der Entwurf der Verhaltenssteuerung unter Berücksichtigung vorhandener Schnittstellen. Das für die Umsetzung benötigte Software-Framework wird im zweiten Teil des Kapitels näher betrachtet. Abschließend wird ein Konzept zur Anhäufung von Erfahrungswissen vorgestellt.

Kapitel 4 befasst sich mit der Bahnplanungskomponente für das kognitive Automobil. Im ersten Teil wird zunächst das Konzept der Gefahrenkarten vorgestellt. Weiterhin werden die Methode der Objektprädiktion sowie das zellbasierte Schätzverfahren im Detail erläutert. Im zweiten Teil wird die Bahnplanung in mehreren Schritten umgesetzt und erweitert. Nach einer Kürzester-Weg-Suche mit geeigneter Schätzfunktion werden Gefahrenkarten in die Optimierungsfunktion aufgenommen, um Umgebungsinformationen berücksichtigen zu können. Es folgt eine Erweiterung hin zur Trajektorienplanung mit Verwendung einer für einen bestimmten Zeitraum vorhergesagten, dynamischen Gefahrenkarte. Am Ende des Kapitels wird das für die Urban Challenge 2007 entwickelte Bahnplanungskonzept vorgestellt, welches dort erfolgreich zum Einsatz kam.

In Kapitel 5 werden die für eine Umsetzung einer Bahn oder Trajektorie notwendigen Verfahren und Regelalgorithmen entwickelt. Nach dem Sicherheits- und Regelungskonzept wird

der Umbau eines Smart-Roadster-Fahrzeuges und eine Drive-by-wire-Umrüstung erklärt. Es folgen Reglerentwürfe zur Längs- und Querregelung des Fahrzeuges und Ergebnisse realer Fahrversuche.

Die Ergebnisse des Gesamtsystems *Verhaltensentscheidung–Bahnplanung–Regelung* werden in Kapitel 6 anhand verschiedener Fahrmanöver dargelegt und diskutiert. Kapitel 7 beschließt die Arbeit mit einer Schlussbetrachtung und einem Ausblick.

In den Anhängen A bis D kann der interessierte Leser Grundlagen zu verschiedenen Systemarchitekturen, Entscheidungs- und Wissensmodellen sowie Kartierungs- und Bahnplanungsverfahren nachlesen. Die Verfahren werden erläutert, in einem Fazit gegenübergestellt und hinsichtlich ihrer Anwendbarkeit im Automobil bewertet. Einige davon wurden als Varianten in die in dieser Arbeit vorgestellten Lösungsansätze integriert. Anhang E liefert eine Übersicht über die im Sonderforschungsbereich SFB/TR28 verwendete Systemarchitektur und deren Komponenten.

Kapitel 2

Stand der Forschung – Autonomes Fahren

2.1 Einführung

Das folgende Kapitel listet zunächst Forschergruppen auf, die wegweisende Arbeiten im Bezug auf autonomes Fahren geleistet haben, welche de facto den Stand der Kunst widerspiegeln. Auf einzelne Methoden zur Entscheidungsfindung, die ihre jeweilige Funktionsfähigkeit in der Praxis unter Beweis stellen konnten, wird in Abschnitt 2.3 eingegangen. Dies sind im Einzelnen eine Verhaltensentscheidung mittels endlichen Automaten sowie eine Auswertung durch Situationsgraphenbäume und mithilfe von Fähigkeitennetzen.

Verschiedene Möglichkeiten, um aus aktiven Fahrverhalten eine fahrbare Bahn zu erzeugen, werden in Abschnitt 2.4 aufgezeigt. Dies geschieht für die vorgestellten Methoden zur Verhaltensentscheidung teilweise implizit über eine Vorgabe von Steuerverläufen, sodass Bahnplanungsverfahren hier gesondert betrachtet werden. Im letzten Abschnitt 2.5 werden die Ansätze gegenübergestellt und es wird ein Fazit gezogen.

2.2 Forschungsgruppen

2.2.1 Universität der Bundeswehr, München

Die von der Forschergruppe um Prof. E. D. Dickmanns entwickelten Fahrzeuge VaMoRs[1] und VaMP (VaMoRs PKW) setzten Maßstäbe im autonomen Fahren und wurden lange Zeit in dem Leistungsverhalten autonomer Fahrzeuge nicht übertroffen [Dickmanns 87, Pellkofer 02, Dickmanns 02] (vgl. Abbildung 2.1). VaMoRs konnte bereits 1987 erste Erfolge verzeichnen und auf einer Strecke von ca. 20 km, mithilfe eines Bildverarbeitungssystems zur Fahrspur- und Hindernisdetektion sowie einem automatisierten Zugriff auf Lenkung, Bremse und Gas, autonom fahren. Im Rahmen des EUREKA[2]-PROMETHEUS[3]-Projektes wurden die Fähig-

[1] Abk. *Versuchsfahrzeug für autonome Mobilität und Rechnersehen.*
[2] Europaweites Netzwerk für marktorientierte, industrielle Forschung und Entwicklung.
[3] Abk. *Programme for a European traffic of highest efficiency and unprecedented safety.*

keiten weiter verbessert [Kemeny 90]. Mit dem 1994 entwickelten Fahrzeug VaMP wurden die Fähigkeiten zusätzlich um autonome Spurwechsel und autonome Überholmanöver erweitert. Durch eindrucksvolle Demonstrationen von Autobahnfahrten um Paris über eine Distanz von mehr als 1 000 km und einer Fahrt von München nach Kopenhagen bei mehr als 95 % autonomem Anteil wurden die Ergebnisse auch einer breiten Öffentlichkeit bekannt.

Abbildung 2.1: Versuchsfahrzeuge VaMoRs und VaMP der Universität der Bundeswehr in München. Bildquelle: [Pellkofer 03].

Ein wichtiger Bestandteil der Fahrzeuge ist das EMS[4]-Vision-System [Gregor 00b, Gregor 00a, Gregor 02]. Hierbei wird, basierend auf Erkenntnissen menschlichen Sehprinzips, *sakkadisches Sehen* verwendet, um bei schnellen Blickrichtungswechseln Perioden der Ruhe vorzusehen und so Bildinformationen auslesen zu können. Der von E. D. Dickmanns entwickelte 4D-Ansatz stellt die Kernkomponente der verwendeten Architektur dar [Dickmanns 87]. In diesem Verfahren zur Umfeldrepräsentation kommt der Zeit eine elementare Bedeutung zu, da alle Objekte in zeitlichem Zusammenhang gespeichert werden. Damit steht eine Historie zur Verfügung, welche durch Schätzverfahren genutzt werden kann, um Vorhersagen über die zu erwartenden Ereignisse zu generieren.

Zur Verhaltensausführung wird ein sogenanntes Fähigkeitennetz verwendet [Pellkofer 03]. Dies ist ein unidirektionaler, zyklenfreier, gerichteter Graph, der die einzelnen Fähigkeiten und Fertigkeiten beinhaltet. In Abschnitt 2.3.3 wird diese Möglichkeit der Verhaltensentscheidung näher betrachtet.

2.2.2 Universität Karlsruhe, Fraunhofer IITB, Karlsruhe

Am Fraunhofer Institut für Informations- und Datenverarbeitung in Karlsruhe steht mit Darvin[5] nunmehr die dritte Generation von Versuchsfahrzeugen zur Verfügung (vgl. Abbildung 2.2). Unter Leitung von Prof. Nagel wurden mit dieser Plattform Algorithmen entwickelt, um im Rahmen von Fahrerassistenzsystemen eine Unterstützung insbesondere

[4] Abk. *Expectation-based Multi-Focal Saccadic Vision.*
[5] Abk. *Driver Assistance using Realtime Vision for Innercity Areas.*

für Behinderte bereitzustellen. An vollständig autonome Fahrzeugführung wurde zunächst noch nicht gedacht, vielmehr sind eine sprachliche Ausgabe von Warnmeldungen oder Empfehlungen sowie ein unterstützender Eingriff in die Fahrzeugführung geplant. Um dieses Ziel zu erreichen, werden Verfahren zur modellbasierten Kreuzungs- und Fahrzeugverfolgung sowie Beschreibungsformen und Analysemöglichkeiten für Verkehrssituationen untersucht [Heimes 98a, Heimes 98b].

Abbildung 2.2: Versuchsfahrzeug Darvin des Faunhofer Instituts für Informations- und Datenverarbeitung in Karlsruhe. Bildquelle: [IITB 08].

Über eine Kombination von Bildfolgenauswertung mit logischer Auswertung auf begrifflicher Ebene soll dem System ein Verständnis für die aktuelle Verkehrssituation vermittelt werden [Darvin 08]. Als Auswertungsverfahren kommen Situationsgraphenbäume zum Einsatz, welche in Abschnitt 2.3.2 näher betrachtet werden [Gerber 00, Nagel 03, H.-H. Nagel und M. Arens 05]. Die gewonnenen Erkenntnisse können dazu verwendet werden, um einen Fahrer bei der Fahrzeugführung zu entlasten – hier steht insbesondere das Halten der Spur und die Abstandsregelung im Fokus. Als Referenz dienen innerstädtische Verkehrssituationen, da diese durch die hohe Komplexität und Vielfalt hohe Anforderungen an das System stellen. Zur Untersuchung modellbasierter Tracking-Verfahren in Bildsequenzen wurde am IITB ein eigenes Framework entwickelt [Dahlkamp 04].

2.2.3 Universitá di Parma, Artificial Vision and Intelligent Systems Laboratory (VisLab), Parma, Italien

Ebenfalls im Rahmen des Prometheus-Projekts [Kemeny 90] wurde am VisLab [VisLab 08] 1989 das Fahrzeug MOB-LAB[6] aufgebaut und von vielen italienischen Forschergruppen für Datenaufzeichnungen und der Realisierung einfacher Warnsysteme verwendet. Nach dem Abschluss des Projektes 1994 wurde im Folgeprojekt ARGO (vgl. Abbildung 2.3) auf Basis

[6] Abk. *Mobile Laboratory*.

eines Lancia Thema aufgebaut [Broggi 99, Broggi 01]. Ziel von ARGO war es, ein Serienmodell für autonome Versuchsfahren umzurüsten. Durch die Verwendung von Standardkomponenten aus dem PC- und Videobereich konnte dies vergleichsweise kostengünstig erreicht werden.

Abbildung 2.3: Versuchsfahrzeuge des Artificial Vision and Intelligent Systems Laboratory der Universitá di Parma. ARGO (links) und The Brains des Teams TerraMax (rechts). Bildquelle: [VisLab 08] und [Team Terramax 08].

Die Umwelterfassung erfolgte bei ARGO ausschließlich mit Videokameras. Für die Auswertung wurde das Sensordatenverarbeitungssystem GOLD[7] entwickelt, das auch Daten anderer Sensorquellen wie Radarsysteme oder Laserscanner verarbeitet [Bertozzi 08]. Die Systemsteuerung bietet drei verschiedene Operationsmodi an: Manuelles Fahren mit Überwachung der Fahrzeugführung und Warnmeldungen in gefährlichen Situationen, überwachtes Fahren, bei welchem in Notsituationen ein Stelleingriff erfolgt und vollständig automatisches Fahren rein nach Wahrnehmungsdaten.

Dem Versuchsfahrzeug wurden Basisverhalten implementiert, um einfache Fertigkeiten wie *Spur wechseln*, *Abstand halten* oder *Spur halten* zu realisieren. Übergeordnete Verhalten auf Planungsebene mit Nutzung von Hintergrundwissen kamen jedoch nicht zum Einsatz. Die Zielsetzung war bei ARGO eher im Forschungsfeld der Bildverarbeitung zu sehen, weniger im kognitiven Bereich.

In jüngster Zeit setzt sich die Forschung im Bereich der vollständig autonomen Fahrzeugführung und der Entwicklung von Entscheidungskomponenten fort. Nach einer erfolgreichen Teilnahme an der Grand Challenge 2005 konnte mit Team TerraMax bei der Urban Challenge 2007 die Finalteilnahme zusammen mit nur zehn anderen Gruppen erreicht werden [Broggi 07].

[7] Abk. *Generic Obstacle and Lane Detection.*

2.2.4 Carnegie Mellon University, Pittsburgh, USA

An der Carnegie Mellon University beschäftigen sich die Abteilungen NavLab[8] unter Leitung von Prof. Chuck Thorpe sowie das Field Robotics Center unter Prof. Red Whittaker mit autonom fahrenden Straßenfahrzeugen. Am NavLab wurden seit 1984 nicht weniger als elf Fahrzeuge mit dem Ziel entwickelt, Verfahren für die Off-Road-Navigation, das autonome Fahren auf Autobahnen, Spurverlassenswarner und Manövrierunterstützung in städtischen Umgebungen zu erforschen [Thorpe 93]. Der Schwerpunkt bei der Umsetzung liegt am NavLab auf der Verwendung neuronaler Netze.

Im Project Alvinn[9] wurde ein neuronales Netz durch Beobachtung eines menschlichen Fahrers trainiert, um einem Bildverarbeitungssystem die Fahrzeugsteuerung beizubringen [Pomerleau 94, Pomerleau 95a]. Hierfür wurden später alternative Methoden zum Backpropagation-Training untersucht [Batavia 96]. Die mit Alvinn gewonnenen Erkenntnisse wurden im Projekt Ralph[10] verwendet, um einen Spurverlassenswarner zu realisieren. Dazu wird nach bildbasierter Bestimmung der Kurvenkrümmung durch einen Abgleich zwischen errechneter Soll-Lenkradwinkelstellung und Ist-Ausführung des Fahrers die Notwendigkeit für eine Warnung ermittelt [Pomerleau 95b]. Momentan laufen mit der neuesten Plattform NavLab 11 Forschungsarbeiten zu den Themen SLAM in dynamischer Umgebung mit Objektverfolgung [Wang 07] und probabilistischen Planungsverfahren [Gonzalez 07, Gonzalez 08].

Abbildung 2.4: Versuchsfahrzeug Boss der Carnegie Mellon University, Gewinner der Urban Challenge 2007. Bildquelle: [Tartan Racing 08].

Die Abteilung Field Robotics Center des Robotics Institute der CMU hat unter Führung von Red Whittaker erfolgreich mehrere Roboterwettbewerbe der DARPA[11] absolviert. Unter

[8] Abk. *Navigation Laboratory*.
[9] Abk. *Autonomous Land Vehicle in a Neural Network*.
[10] Abk. *Rapidly Adapting Lateral Position Handler*.
[11] Abk. *Defense Advanced Research Projects Agency*. Forschungsabteilung des US-amerikanischen Verteidigungsministeriums.

dem Namen *Team Red* wurde beim ersten Grand Challenge Rennen 2004 mit dem Fahrzeug Sandstorm die längste Strecke zurückgelegt – in diesem Rennen erreichte noch kein Teilnehmer das Ziel. Im folgenden Jahr wurde mit zwei Fahrzeugen die komplette Strecke zurückgelegt und es wurden die Plätze zwei und drei belegt. Unter dem neuen Namen *Tartan Racing* [Tartan Racing 08] konnte dann bei der Urban Challenge 2007 [Urban Challenge 07] mit dem Fahrzeug Boss (vgl. Abbildung 2.4) der lang ersehnte Sieg eingefahren werden.

2.2.5 Stanford University, Stanford, USA

Unter der Leitung von Sebastian Thrun gewann das *Stanford Racing Team* [Stanford Racing 08] die Urban Challenge 2005 mit dem Fahrzeug Stanley, einem Versuchsfahrzeug auf Basis eines umgebauten VW Tuareg. Bei diesem Wettbewerb ging es hauptsächlich darum, in schwierigem Gelänge mithilfe von GPS-Informationen eine Wegstrecke abzufahren und lokale Korrekturen vorzunehmen. Der Schwerpunkt lag auf der robusten Sensordatenverarbeitung und auf der Bewegungsausführung, weniger auf einer komplexen Verhaltensentscheidung bzw. Berücksichtigung von Verkehrsregeln. Der Erfahrungsgewinn mit solch großen mobilen Systemen bei den ersten beiden Wettbewerben spiegelte sich insbesondere in der hohen Zuverlässigkeit von Hard- und Softwarekomponenten bei den A-Teams in der Urban Challenge 2007 wider. Hier wurde mit dem Fahrzeug Junior der zweite Platz hinter dem Team *Tartan Racing* belegt (vgl. Abbildung 2.5).

Abbildung 2.5: Versuchsfahrzeuge Junior (links) und Stanley (rechts) des Stanford Racing Teams der Stanford University, Gewinner der Grand Challenge 2005. Bildquelle: [Stanford Racing 08].

Am Stanford AI Lab beschäftigt sich die Gruppe *Robotics and Machine Learning* schwerpunktmäßig mit probabilistischer Entscheidungsfindung für Robotersysteme [Thrun 05] sowie simultaner Lokalisation und Kartenerstellung (SLAM). Im dort entwickelten FastSLAM-Algorithmus [Montemerlo 07] werden sog. Partikelfilter auf das SLAM-Problem angewandt. Diese Algorithmen kamen unter anderem bei der Urban Challenge zum Einsatz [Stanford Racing 07]. Ein weiteres Forschungsfeld sind Lernverfahren [Thrun 96,

Thrun 98b]. So wurde bereits bei der Grand Challenge 2005 ein Ansatz verwendet, um Geschwindigkeit und Fahrstrategie an die Sichtweite zu koppeln. Die menschlichen Fähigkeiten zur Klassifikation von Hindernissen wurden genutzt, um mithilfe aufgezeichneter Videodaten Bildverarbeitungsparameter zu lernen und dem Fahrzeug die gleichen Fähigkeiten zu vermitteln. Diese Methode hat sich insbesondere für die Klassifikation von Büschen und Sträuchern bei der Grand Challenge 2005 als sinnvoll erwiesen.

2.2.6 Weitere Gruppen

Neben den genannten Forschergruppen existieren zahlreiche weitere, kleinere Institutionen und Einrichtungen, die im Bereich des autonomen Fahrens forschen.

Die Gruppe um Prof. Paul Levy am *Institut für Parallele und Verteilte Systeme*[12] beschäftigt sich im Projekt ASSET-Road[13] mit der Fragestellung, wie die Ressourcen der Transportindustrie besser genutzt werden können. Die zentrale Forschungsaufgabe ist die Entwicklung eines Multi-Agenten-Systems, das Beobachtungen von Fahrzeugen verwendet, um Regelverstöße zu identifizieren. Grundlage dafür bildet eine Verkehrsregel-Datenbank, welche im Projektrahmen entwickelt wird.

In der Arbeitsgruppe *Wirtschaftsinformatik und Simulation* von Prof. Timm[14] liegt der Forschungsschwerpunkt unter anderem auf Maschinellem Lernen, Multiagentensystemen und der Wissensrepräsentation von autonomen Systemen in dynamischen Umgebungen. Zahlreiche Veröffentlichungen beschäftigen sich mit der Anwendung der entwickelten Methoden auf intelligente Automobile [Lattner 05b, Lattner 05a, Miene 04].

Am *Intelligent Control Systems Laboratory*[15] werden unter der Leitung von Prof. L. Vlacic intelligente Sensoren, Elektronikkomponenten und Algorithmen zur Regelung und Entscheidungsfindung für Fahrzeuge entwickelt. Im Verbund mit anderen Partnern werden insbesondere im Projekt *CyberCars2*[16] neue Methoden im Hinblick auf Fahrzeug-Fahrzeug- und Fahrzeug-Infrastruktur-Kommunikation erforscht [Kolodko 03, Myers 05, Baber 05].

Das Projektteam E-MOTION um Prof. C. Laugier am INRIA in Grenoble[17] befasst sich mit Geometrie und Wahrscheinlichkeit in Bewegung und Aktion, unter anderem für intelligente Fahrzeuge. Das Forschungsfeld umfasst neben dem Schwerpunkt der multimodalen und inkrementellen Raum- und Bewegungsmodellierung die weiteren Aspekte der Bewegungsplanung und der probabilistischen Entscheidungsfindung. In jüngster Zeit finden sich zahlreiche Veröffentlichungen zum Thema der *Bayes'schen Programmierung* [Yguel 05, Chen 06].

[12] IPVS, vgl. auch `http://www.ipvs.uni-stuttgart.de/`.

[13] Abk. Advanced Safety and Driver Support for Essential Road Transport.

[14] Goethe-Universität, Frankfurt am Main. Vgl. auch `http://www.uni-frankfurt.de/fb/fb12/informatik/forschung/arbeitsgruppen/is/`.

[15] Vgl. `http://www.griffith.edu.au/science/intelligent-control-systems-laboratory/`.

[16] Vgl. `http://www-c.inria.fr/cybercars2/`.

[17] Vgl. `http://www.inrialpes.fr`.

2.3 Verhaltensentscheidung

2.3.1 Verhaltensentscheidung mittels endlicher Zustandsautomaten

Um die Aufgaben der Urban Challenge zu bewältigen, wurde im Team Tartan Racing ein *Reasoning Framework* entwickelt [Urmson 08, Ferguson 08]. Ziel war es, damit Teilaufgaben wie Routenplanung, kontextabhängige Entscheidungsfindung und Bewegungsplanung zu integrieren, um Fahrmanöver sicher und schnell durchzuführen. Abbildung 2.6 zeigt die verwendete Systemarchitektur, welche mit den drei Planungsebenen für Route, Verhalten und Bewegung an jene herkömmlicher Robotersysteme erinnert.

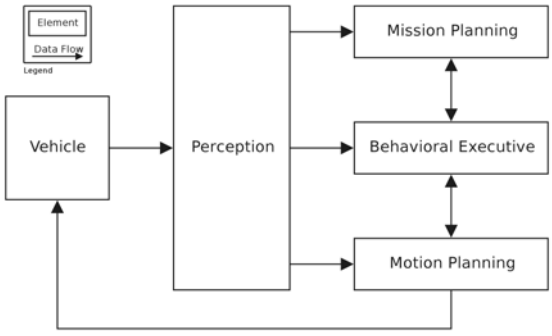


Abbildung 2.6: Software-Systemarchitektur auf dem Fahrzeug Boss der Carnegie Mellon University. Bildquelle: [Ferguson 08].

Von der Missionsplanung wird auf Grundlage der hinterlegten Straßenkarte und einer Kostenfunktion eine Route geplant und an die Verhaltensausführung weitergereicht. Zusammen mit lokalen Informationen der Wahrnehmungskomponente erzeugt die Verhaltensausführung Zwischenziele für den Bewegungsplaner. Die Verhaltensausführung identifiziert zunächst den Fahrkontext, der den Wert *on-road*, *at-intersection* oder *in-zone* annehmen kann. Abhängig von dieser Situationsinterpretation wird ein zugehöriges Metaverhalten aktiviert, das sich wiederum unterlagerter Verhalten der jeweiligen Kontextgruppe bedient (vgl. Abbildung 2.7).

Verhalten wie *StateEstimator* übernehmen dabei weitere Interpretationsaufgaben, um bspw. die logische Position des Eigenfahrzeuges zu bestimmen. Meta- und Basisverhalten sind in einem endlichen Zustandsautomaten zusammengefasst. Die Auswahl der Verhalten erfolgt in Abhängigkeit von den Fortschrittsangaben des Bewegungsplaners und dem aktuellen Missionsziel [Tartan Racing 07].

Um auf unvorhergesehene Situationen angemessen reagieren zu können, und bspw. eine Neuplanung bei abgesperrten Straßen oder ein Umfahren blockierender Fahrzeuge durchzuführen, wurden weitere Mechanismen entwickelt, die in [Baker 08] detailliert beschrieben

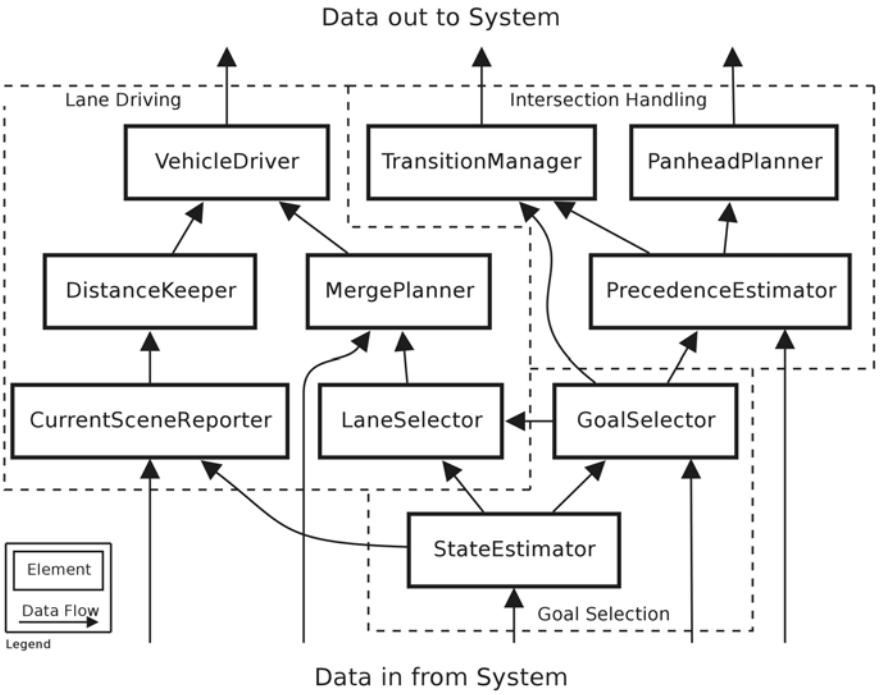


Abbildung 2.7: Komponenten und Schnittstellen auf der Ebene der Verhaltensausführung in Boss. Bildquelle: [Ferguson 08].

werden. Die relativ konventionelle Herangehensweise und das sehr gute Abschneiden zeigt, dass die Aufgaben der Urban Challenge mit herkömmlichen Methoden bewältigt werden konnten und die Schwierigkeiten weniger in der Komplexität, sondern vielmehr in der Zuverlässigkeit der Algorithmen zu suchen waren.

2.3.2 Verhaltensentscheidung mittels Situationsgraphenbäumen

Ein Situationsgraphenbaum stellt eine Gesamtheit von Situationsgraphen und deren Detaillierungen dar und wird in [Arens 01] als Analyseverfahren für eine Verkehrssituation auf Basis von Bildfolgen verwendet [Arens 02b, Arens 02a, Nagel 03]. In [Gerber 00] werden textuelle Beschreibungen aus Situationen mithilfe von Situationsgraphenbäumen generiert. Durch einen hohen Detaillierungsgrad lässt sich eine aus der exakt analysierten Situation geeignete Handlung ableiten, sodass diese Beschreibung in einem bestimmten Verhalten des Eigenfahrzeugs resultiert. Situationsgraphenbäume können damit Interpretations- und Entscheidungsaufgaben übernehmen.

Ein Situationsgraph entsteht durch die Verkettung einzelner Situationen über Prädiktionskanten und stellt die zeitliche Entwicklung einer Situation dar. Es handelt sich um einen gerichteten Graphen mit Start- und Endpunkt. Soll eine Situation abgearbeitet (analysiert) werden, so müssen die zugehörigen spezialisierten Graphen von Start- bis Endpunkt durchlaufen werden. Prädiktionskanten besitzen eine Übergangswahrscheinlichkeit, um

eine Aussage über die Wahrscheinlichkeit möglicher Nachfolgesituationen anzugeben. Abbildung 2.8 zeigt eine schematische Darstellung eines Situationsgraphenbaumes.

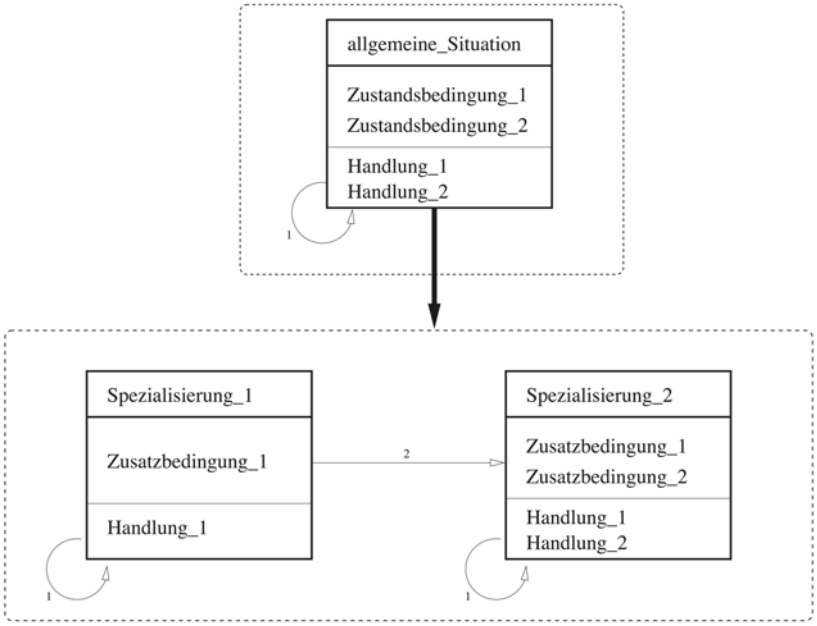


Abbildung 2.8: Aufbau eines Situationsgraphenbaumes. Eine Situationsbeschreibung wird durch Zustands- und Handlungsschema beschrieben (gerahmter Kasten). Aus einer allgemeinen Darstellung (oben) werden einzelne Situationen spezialisiert (unten). Die Abhängigkeit wird durch den großen Pfeil dargestellt. Kleine Pfeile zeigen Nachfolgebeziehungen, Nummern die Priorität dieser Beziehungen. Bildquelle: [Arens 01].

Durch Spezialisierung der allgemeinen Situation, dargestellt durch den großen Pfeil, entsteht ein neuer Situationsgraph mit Situationen und Prädiktionskanten (kleine Pfeile). Im oberen Teil einer Situation steht dabei der Name, unten eine Handlung als Konsequenz der Situation. In der Mitte sind die Prädikate aufgelistet, welche für die jeweilige Situation zutreffen müssen und in ihrer Gesamtheit eine Aussage darstellen. Jede Aussage innerhalb einer Situation wird als Prädikat der unscharfen, metrisch-temporalen Horn-Logik [Kreuzer 06] gespeichert. Situationen können auch in sich selbst übergehen, wie die kreisförmigen Pfeile zeigen. Durch eine weitere Detaillierung der Situationen in neue Situationsgraphen entsteht die Baumstruktur.

2.3.3 Verhaltensentscheidung mittels Fähigkeitennetzen

Das von Pellkofer vorgestellte Verfahren zur Verhaltensentscheidung mittels Fähigkeitennetzen [Pellkofer 03] wurde in der Gruppe um Prof. E. D. Dickmanns entwickelt und integriert die Ideen des 4D-Ansatzes sowie des EMS-Vision-Systems [Dickmanns 87, Gregor 00a, Gregor 00b, Gregor 01, Pellkofer 00, Siedersberger 00].

Die Situationsanalyse ist auch hier eng an die Verhaltensentscheidung gekoppelt (vgl. Abbildung 2.9).

Mithilfe von Situationsaspekten wird eine Situation analysiert und durch linguistische Variablen beschrieben. Es handelt sich dabei um unscharfe Werte im Sinne der Fuzzy-Theorie im Wertebereich $[0; 1]$. Diese Form der Darstellung wurde aus Gründen der Nachvollziehbarkeit und Lesbarkeit der Situationsbeschreibung gewählt und erhöht die Transparenz im Prozess. Eine Situationsanalyse erfolgt iterativ und verfeinert die Situationsbeschreibung in jedem Durchlauf, beginnend mit einem vordefinierten Satz auszuwertender Situationsaspekte.

Da es aus Gründen des Rechenaufwandes nicht möglich ist alle existierenden Situationsaspekte zu überprüfen, werden im weiteren Verlauf nur die angeforderten, relevanten Situationsaspekte in die Analyse aufgenommen. Auf Basis der linguistischen Beschreibung werden mithilfe von unscharfen Regeln Fähigkeiten ausgewählt und parametriert. Um in Notsituationen sofort reagieren und auf andere Fähigkeiten umschalten zu können, werden zu einer Fähigkeit auch mögliche Alternativen ausgegeben, die zunächst weniger hoch priorisiert sind. Die Iteration wird beendet, sobald sich das ausgewählte Verhalten nicht mehr ändert und alle angeforderten Situationsaspekte ausgewertet wurden.

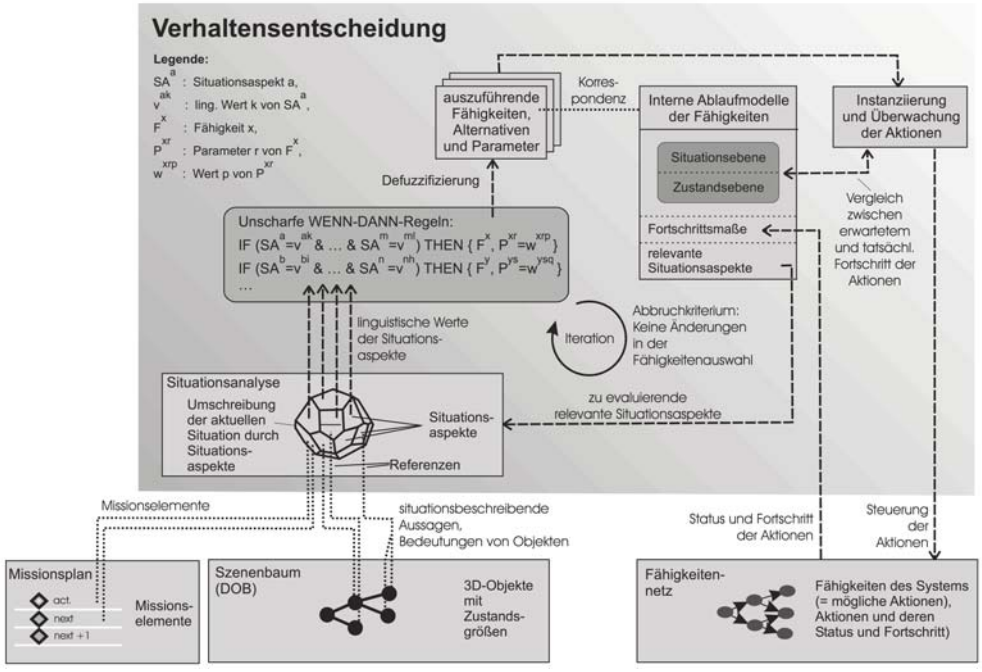


Abbildung 2.9: Komponenten der Situationsinterpretation und Verhaltensentscheidung sowie deren Einbettung in das Gesamtsystem. Die Fähigkeiten des Gesamtsystems sind in Form von Fähigkeitennetzen hinterlegt. Bildquelle: [Pellkofer 03].

Zur Verhaltensausführung wird ein sogenanntes Fähigkeitennetz verwendet. Dies ist ein unidirektionaler, zyklenfreier, gerichteter Graph, welcher die einzelnen Fähigkeiten und

Fertigkeiten beinhaltet. Die Abhängigkeiten innerhalb des Fähigkeitennetzes zwischen Komponenten auf verschiedenen Ebenen sind von statischer Natur. Zwischen Komponenten auf gleicher Ebene existieren keine Wechselwirkungen. Aktionen auf unterster Ebene laufen nacheinander ab. Es handelt sich folglich um ein konkurrierendes Modell. Dies widerspricht der menschlichen Vorgehensweise, bei welcher teilweise unbewusst Handlungen parallel ausgeführt und zu einem Gesamtverhalten kombiniert werden. Der Fortschritt von Einzelaktionen wird über die verschiedenen Ebenen rückgekoppelt, um auf etwaige Fehler angemessen reagieren zu können. Die Fortschrittsmaße müssen so aussagekräftig gewählt sein, dass auf oberster Ebene ein symbolisches Bild von der sich entwickelnden Situation erstellt werden kann.

Durch die hohe Kapselung und die geringe Interaktion zwischen einzelnen Verhalten ist dieser Ansatz nur bedingt für den Einsatz von Lernverfahren geeignet, konnte jedoch in eindrucksvollen Demonstrationen seine Funktionstüchtigkeit beweisen (vgl. Abschnitt 2.2.1).

2.4 Bahnplanung für Automobile

Während in Anhang D allgemeine Ansätze zur Bahnplanung aus dem Gebiet der mobilen Robotik vorgestellt werden, soll in diesem Abschnitt der Stand der Technik zur Bahnplanung speziell für Automobile beleuchtet werden. Unter dem Begriff *Automobil* wird im Folgenden ein Fahrzeug verstanden, welches eine wie in Abbildung 2.10 dargestellte Kinematik besitzt.[18] Aufgrund der vielen Forschungsarbeiten zum Schwerpunkt *Einparken* werden diese Ansätze separat aufgelistet.

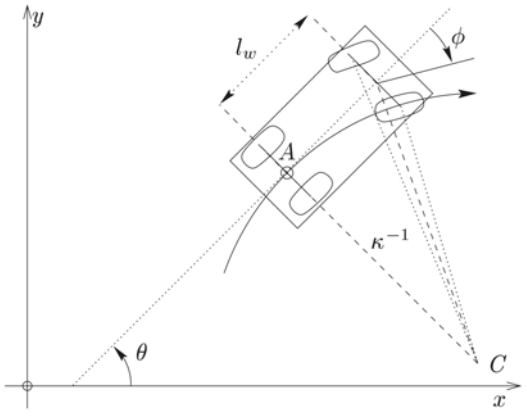


Abbildung 2.10: Modell eines Fahrzeuges mit Automobil-Kinematik. Der Zustand enthält Position (x, y), Orientierung θ, Geschwindigkeit v und Lenkwinkel ϕ. Der maximale Lenkausschlag ϕ_{max} begrenzt die Kurvenkrümmung auf $\kappa_{max} = {l_w}^{-1} tan\phi_{max}$. Bildquelle: [Fraichard 04].

[18] Im Englischen oftmals zutreffend als *Car-like-Vehicle* bezeichnet.

Grundsätzlich dienen die unterschiedlichen Bahnplanungsalgorithmen dazu, ein Automobil aus einer Anfangskonfiguration (x_s, y_s, θ_s) in eine Endkonfiguration (x_g, y_g, θ_g) zu überführen und dabei insbesondere die nicht holonomen Eigenschaften zu berücksichtigen. Nicht-Holonomität entsteht in der Regel dann, wenn ein System weniger Steuerungsparameter als Konfigurationsparameter aufweist. Im Falle eines Automobils ist dies durch die drei Konfigurationsparameter (Position x, y und Orientierung θ) und nur zwei Steuerungsparameter (Geschwindigkeit v und Lenkwinkel ϕ) gegeben.

Die verschiedenen Bahnplanungsverfahren unterscheiden sich durch die Art und Weise, wie Nicht-Holonomität berücksichtigt wird und wieviele Randbedingungen in die Planung einfließen. Im Folgenden werden zwei unterschiedliche Modelle betrachtet.

Reeds & Shepp (RS) Modell

Durch die Beschränkung des Lenkwinkels $|\phi| \leqslant \phi_{max}$ wird eine Randbedingung der Form

$$|\kappa| \leqslant \kappa_{max} = {l_w}^{-1} \tan\phi_{max} \tag{2.1}$$

vorgegeben. Daraus folgt ein Fahrzeugmodell in Zustandsraumbeschreibung

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\,\theta \\ \sin\,\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \omega \tag{2.2}$$

mit Geschwindigkeit $|v| \leqslant v_{max}$ und mit Drehgeschwindigkeit $\omega = v\kappa$ ($|\kappa| \leqslant \kappa_{max}$). Die Änderung der Krümmung erfährt dabei keine explizite Berücksichtigung in der Zustandsbeschreibung. Dieses Modell ist in der Literatur aufgrund seiner Einfachheit sehr weit verbreitet und auch unter dem Namen *Reeds & Shepp Car (**RS Car**)* bekannt.[19] Die Orientierungsänderung des Fahrzeugs ergibt sich aus dem aktuell eingestellten Lenkwinkel zu $\dot{\theta} = \omega = \frac{v}{l_w}\tan\phi$. Dieser wird für das RS-Car üblicherweise als $\phi \in M, M = \{\phi_{min}, 0, \phi_{max}\}$ gewählt. Aus dieser Formulierung folgt, dass mit dem RS-Car Kreisbögen und Geraden gefahren werden können, um von einer Startkonfiguration (x_s, y_s, θ_s) in eine Endkonfiguration (x_g, y_g, θ_g) zu wechseln (vgl. Abbildung 2.11).

Die Diskontinuität beim Übergang zwischen Kreis und Gerade oder beim Übergang zwischen Kreis und Kreis stellt allerdings in der Realität ein Problem dar und würde eine ruckartige Lenkbewegung erforderlich machen. Eine solche ist zum einen weder durch einen Fahrer noch durch Aktorik umzusetzen. Zum anderen stellen plötzlich auftretende Spitzen in der Querbeschleunigung ein Sicherheitsrisiko dar. Die Bahn kann letztendlich nur durch einen vollständigen Stop am Übergang exakt abgefahren werden. Für Parkmanöver ist diese Einschränkung keine wesentliche Beeinträchtigung der Bewegung, weshalb in vielen Algorithmen zum Einparken auf dieses Modell zurückgegriffen wird. Für eine schnelle Fahrt allerdings ist

[19] Benannt nach den gleichnamigen Forschern [Reeds 90].

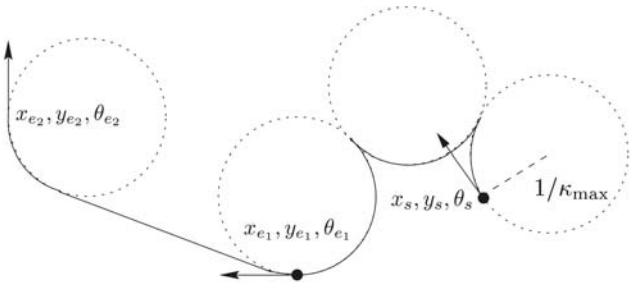


Abbildung 2.11: Übergang aus einer Startkonfiguration (x_s, y_s, θ_s) in die Endkonfigurationen $(x_{e_1}, y_{e_1}, \theta_{e_1})$ und $(x_{e_2}, y_{e_2}, \theta_{e_2})$ mithilfe von Kreisbögen und Geraden des RS-Modells. Bildquelle: [Reeds 90].

dies nicht hinnehmbar, sodass oftmals das im nächsten Abschnitt beschriebene Modell zum Einsatz kommt.

Continuous Curvature (CC) Modell

Neben der Randbedingung für die maximale Krümmung gem. Formel 2.1 kann zusätzlich eine Bedingung für die maximale Krümmungsänderung gesetzt werden:

$$|\sigma| \leqslant \sigma_{max} = \dot{\kappa}_{max} \tag{2.3}$$

Hierin beträgt die Winkelbeschleunigung $\sigma = \dot{\kappa} = \frac{\dot{\phi}}{\cos^2\phi}$ beträgt. Mit diesen Randbedingungen lässt sich die Zustandsraumbeschreibung nun abhängig von den Eingängen v und σ formulieren wie folgt:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\kappa} \end{pmatrix} = \begin{pmatrix} \cos\,\theta \\ \sin\,\theta \\ \kappa \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \sigma \tag{2.4}$$

Hierbei gilt $|v| \leqslant v_{max}, |\kappa| \leqslant \kappa_{max}$ und $|\sigma| \leqslant \sigma_{max}$. Dieses Modell ist auch unter dem Namen CC Car[20] bekannt und erfüllt zusätzlich die Bedingung für kontinuierliche Krümmungsänderung. Der mit diesem Modell zu fahrende Pfad kann sich aus Geradenstücken und Kreisbögen mit Radius $r = 1/\kappa \geqslant 1/\kappa_{max}$ zusammensetzen, wobei im Übergang Klothoidensegmente mit einer Winkelbeschleunigung von höchstens $\pm\sigma_{max}$ liegen. Die Geschwindigkeit muss zur Bestimmung der notwendigen Parameter zu Beginn angegeben werden und stellt eine obere Schranke dar. Für eine Bewegungsplanung können die Eingangsparameter für bestimmte Ausrichtungen variiert werden.

Der Vorteil gegenüber dem RS-Modell ist die Beschreibung der tatsächlich fahrbaren Pfade ohne ein Anhalten im Übergang zwischen Kreis und Gerade. Dagegen steht die aufwändigere

[20] Abk. *Continuous-Curvature Car.*

Berechnung des CC-Modells. Wird lediglich eine ungefähre Abschätzung der Streckenlänge benötigt, so liefert das RS-Modell bereits gute Näherungswerte.

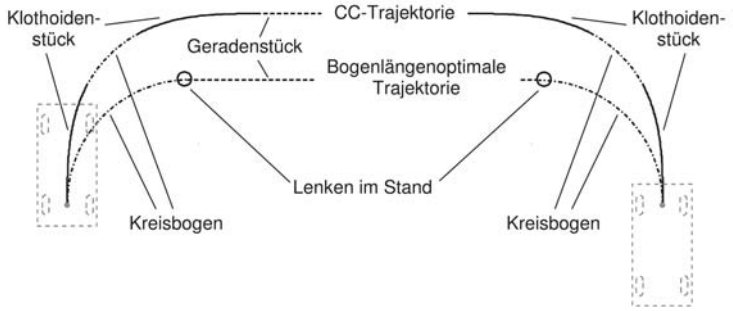


Abbildung 2.12: Veranschaulichung der unterschiedlichen Modelle: Während der mit dem RS-Modell gefundene Weg etwas kürzer ist, weist der CC-Pfad keine Sprünge der Krümmungsänderung auf und ist deshalb ohne ein Anhalten im Übergang fahrbar. Bildquelle: [Müller 06].

2.4.1 Allgemeine Ansätze zur Bahnplanung

Mit dem Continuous-Curvature-Path-Planner (CCPP) [Scheuer 96, Scheuer 97] wird eine Möglichkeit vorgestellt, um für ein Fahrzeug mit Automobil-Kinematik Bahnen mit kontinuierlicher Krümmungsänderung zu planen. Ausgehend von einer Startkonfiguration (x_s, y_s, θ_s) wird ein kollisionsfreier Pfad zur Endkonfiguration (x_g, y_g, θ_g) errechnet, welcher sich aus Klothoidenstücken zusammensetzt und damit die geforderte Bedingung nach sprungfreien Soll-Lenkwinkeln erfüllt. Die obere Schranke für die Krümmung ergibt sich für die gesamte Bahn aus dem minimalen Radius r_{min} zu $|\kappa(s)| \leqslant \frac{1}{r_{min}}$.

In [Divelbiss 97] wird eine numerische Methode zur Pfadplanung für nicht holonome Fahrzeuge vorgestellt. Das Planungsproblem wird zunächst in ein nichtlineares Problem kleinster Fehlerquadrate in einem erweiterten Zustandsraum überführt und dort iterativ gelöst. Mit diesem Ansatz ist eine Pfadplanung für mehrgelenkige Fahrzeuge, wie bspw. Lastwagen mit einem oder mehreren Anhängern, möglich. Obwohl der Ansatz nicht speziell für Parkmanöver ausgelegt ist, wurde dieses Problem dennoch repräsentativ in der Simulation gelöst.

Ein mehrstufiges Verfahren, um Bahnen für nicht holonome Fahrzeuge zu erzeugen, wird in [Laumond 94] vorgestellt. Zunächst wird ein kollisionsfreier Pfad ohne Berücksichtigung der nicht holonomen Eigenschaften geplant. Dieser Pfad wird im nächsten Schritt rekursiv unterteilt, bis alle Endpunkte mit einem Minimalpfad verbunden werden können, welcher die notwendigen Randbedingungen berücksichtigt. In einem Optimierungsschritt werden nicht benötigte Pfadsegmente gelöscht. Als Minimalpfade kommen die in [Reeds 90] vorgestellten *optimalen Pfade* zum Einsatz.

Ein *Best-First*-Suchalgorithmus wird in [J. Barraquand 89] angewandt, um eine Bahn für ein nicht holonomes Fahrzeug zu planen. Im Gegensatz zu anderen Planungsverfahren kommt kein Bahnmodell zum Einsatz, sondern es wird ein diskreter Konfigurationsraum erzeugt. Dieser wird mit dem Arbeitsraum abgeglichen und während der Suche auf gültige Konfigurationen überprüft. Um einen Pfad zu erzeugen und gleichzeitig dessen Fahrbarkeit sicherzustellen, wird ein Fahrzeugmodell verwendet, das die Konfigurationen verkettet. Die Funktionsfähigkeit wurde mit Planungsaufgaben für Parallelparken und Navigation in unstrukturierter Umgebung gezeigt.

2.4.2 Bahnplanung für Parkmanöver

Fast alle bekannten Arbeiten zum Thema Einparken versuchen geschlossene Lösungen zu präsentieren, um ein Fahrzeug bei Kenntnis bestimmter Parameter wie Position, Ausdehnung der Parklücke, Freiraum und Hindernispositionen aus der Start- in die Zielkonfiguration zu überführen. Das Hauptaugenmerk liegt dabei auf Parallelpark-Manövern, da diese Aufgabe im Gegensatz zum Vorwärtseinparken der schwierigere Fall scheint. Ein großer Teil der Arbeiten unterscheidet sich nur hinsichtlich des Typs der Bahnkurve bzw. deren elementarer Bausteine.

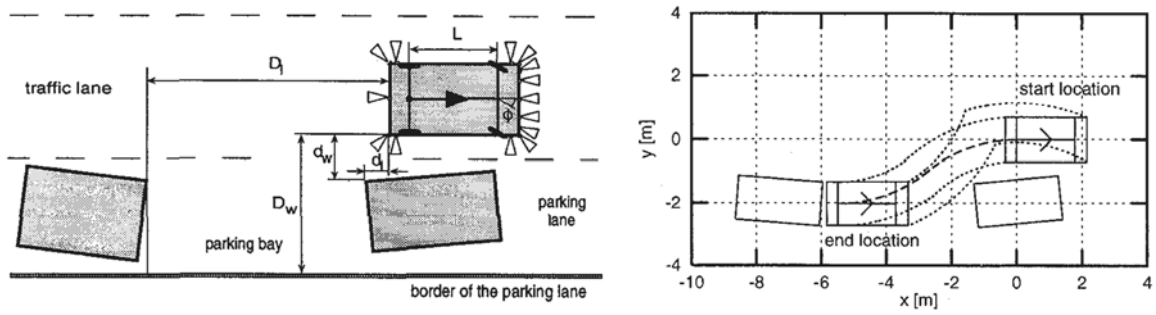


Abbildung 2.13: Ausgangssituation mit den notwendigen Parametern zur Beschreibung von Parklücke und relativer Fahrzeugposition (links). Ausführung des Parallelparkvorgangs mithilfe sinusförmiger Bahnkurven (rechts). Bildquelle: [Paromtchik 96a].

[Paromtchik 96b, Paromtchik 96a] und [Murray 90] verwenden einen iterativen Einparkalgorithmus auf Basis sinusförmiger Kurven, um ein nicht holonomes Fahrzeug zu steuern. Aus Sensordaten müssen die hintere und seitliche Begrenzung der Parklücke (D_l, D_W), sowie ein nach vorn begrenzendes Hindernis (d_w, d_l) bestimmt werden (vgl. Abbildung 2.13). Der Vorgang kann dabei mehrere Stufen rückwärts-vorwärts umfassen, bis die notwendige laterale Abweichung erreicht ist. Für die Periodendauer T einer solchen iterativen Bewegung wurden die folgenden Kontrollfunktionen definiert, um Verläufe für die Lenkwinkelstellung ψ und die Geschwindigkeit v vorzugeben:

$$\phi(t) = \phi_{max} k_\phi A(t), \quad 0 \leqslant t \leqslant T \tag{2.5}$$

$$v(t) = v_{max} k_v B(t), \quad 0 \leqslant t \leqslant T \tag{2.6}$$

Mithilfe des Fahrzeugmodells werden der Umschaltzeitpunkt t' und der maximale Lenkwinkelausschlag ϕ_{max} bestimmt, für die eine kollisionsfreie Fahrt innerhalb des Freiraumes mit folgenden Parametrierfunktionen möglich ist:

$$A(t) = \begin{cases} 1 & ,0 \leqslant t \leqslant t' \\ \cos\frac{\pi(t-t')}{T} & ,t' \leqslant t \leqslant T-t' \\ -1 & ,T-t' \leqslant t \leqslant T \end{cases} \tag{2.7}$$

$$B(t) = 0,5(1-\cos\frac{4\pi t}{T}), 0 \leqslant t \leqslant T \tag{2.8}$$

Die entstehenden Kontrollfunktionen werden dann von der Regelung ausgeführt. Experimente mit einem realen Elektrofahrzeug konnten die Funktionsfähigkeit belegen.

In [Lyon 92] wurden Polynome 5. Ordnung als Bahnkurven verwendet, um einen Parallelparkvorgang zu planen. Als Randbedingungen dienten neben der Position in Start- und Zielpunkten zusätzlich die Steigungen $y'_a = y'_e = 0$ und die Krümmungen in den Endpunkten $\kappa_e = \kappa_a = 0$.

Eine Methode, um autonom in Tiefgaragen einzuparken, wird in [Seignez 05] vorgestellt. Nach dem Abstellen des Fahrzeuges vor der Garage erfolgt die Suche nach einem Parkplatz durch Abfahren einer zuvor aufgenommenen Trajektorie. Ist ein freier Platz identifiziert, dann wird ähnlich wie in [Paromtchik 96a] eine sinusförmige Bahn parametriert und parallel eingeparkt. Dafür wird zunächst eine schnelle Abschätzung der zu fahrenden Pfadlänge mithilfe von Kreisbögen durchgeführt.

Ein zweistufiges Verfahren zur Planung von Einparkvorgängen wird in [Müller 06, Müller 07] vorgestellt. Darin wird der in [Laumond 94] entwickelte Bahnplanungsansatz aufgegriffen und auf Parkmanöver angewendet. Die Methode ist modellbasiert und lässt sich durch Anpassung geometrischer und dynamischer Fahrzeugparameter auf unterschiedlichste Fahrzeugtypen anwenden. Zunächst wird eine kollisionsfreie Bahn zwischen Start- und Zielpunkt geplant und diese dann in eine fahrbare Trajektorie umgewandelt. Im Gegensatz zu vielen anderen Ansätzen wird dadurch auch die Bewegung vorgegeben. Die Trajektorie kann von einer beliebigen Ausgangsposition geplant werden und erlaubt sowohl das Einparken senkrecht bzw. schräg zur Fahrtrichtung sowie das parallele Einparken. Im Prinzip kombiniert diese Arbeit Forschungsarbeiten wie den kollisionsfreien Bahnplanungsalgorithmus aus [Mirtich 92] mit dem Planer für Trajektorien mit kontinuierlicher Krümmungsänderung aus [Fraichard 04].

In einigen Forschungsarbeiten werden KI-Methoden auf das Einparkproblem angewendet. So wird dies in [Sugeno 84] und [Li 03] mithilfe von Fuzzy-Regeln gelöst, in [Gorinevsky 96] werden neuronale Netze trainiert, um Trajektorien für Parkmanöver zu erzeugen.

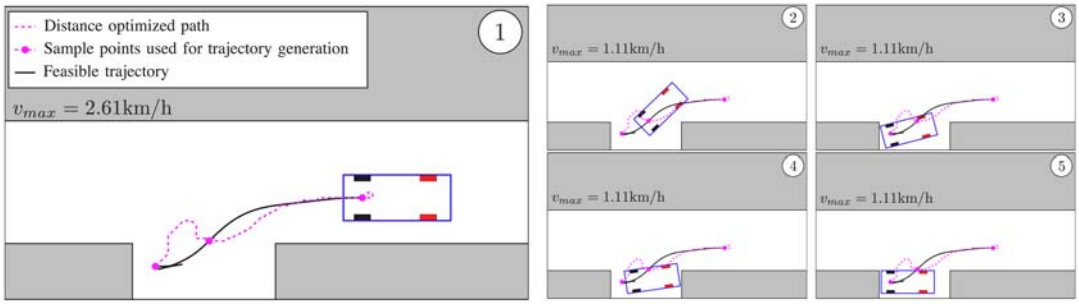


Abbildung 2.14: Verfahren nach [Müller 06]: Resultierende Trajektorie für ein paralleles Einparkmanöver. Die Parklücke ist lediglich 0,4 m breiter und 1,4 m länger als das Fahrzeug selbst. Bildquelle: [Müller 06].

Alle hier vorgestellten Verfahren haben gemein, dass sie auf a priori vorhandene Umgebungsparameter angewiesen sind. Um diese Angaben zu bestimmen, ist eine strukturierte Darstellung auf Objektebene erforderlich. Auf Basis einer gerasterten Karte sind diese Daten nur mit einem großen zusätzlichen Aufwand zu bestimmen.

2.5 Fazit

Im Bezug auf die Systemarchitektur lässt sich sagen, dass mit endlichen Automaten zumindest für spezielle und kompakte Aufgabenstellungen hervorragende Ergebnisse erzielt werden. Die Skalierbarkeit auf einen größeren Diskursbereich kann infrage gestellt werden, sodass dort Vorteile bei Systemen mit ausgeprägter Interpretationskomponente zum Tragen kommen können (Situationsgraphenbäume, Fähigkeitennetze). Für eine abschließende Beurteilung fehlen jedoch weitere Anwendungsbeispiele.

Es scheint Einigkeit darüber zu bestehen, dass die Komponente zur Entscheidungsfindung regelbasiert aufgebaut werden sollte und nicht mithilfe wenig transparenter Methoden wie bspw. künstlichen neuronalen Netzen. Nur so lässt sich explizit vorhandenes Wissen aus der StVO oder von einem Fahrlehrer modellieren. Wie genau diese Regeln repräsentiert werden und in welchem Maß sich die Logik an sprachlichen Elementen orientiert, scheint dabei weniger wichtig. Für die Modellierung der Verhalten in der Ablaufsteuerung lässt sich kein Favorit angeben, vor dem Hintergrund einer späteren Anwendung von Lernverfahren ist jedoch eine Architektur mit geeigneten Ansatzpunkten, bspw. durch Zugang zu den Ausführungsparametern, sinnvoll.

Für hinreichend glatte und in der Praxis fahrbare Bahnen muss ein Fahrzeugmodell in Zustandsraumbeschreibung gemäß des CC-Fahrzeugs verwendet werden (vgl. Formel 2.4). Als Schätzfunktion für einen Best-First-Suchalgorithmus hingegen ist das vereinfachte RS-Modell denkbar (vgl. Formel 2.2). Neben dem verwendeten Modell unterscheiden sich Bahnplanungsmethoden in der Frage, wie der Arbeitsraum beschrieben werden kann. Im Gegensatz zu einer symbolischen Repräsentation arbeiten viele Planer auch auf Basis von

Belegtheitskarten. Eine Kombination beider Repräsentationsformen wäre als Grundlage für eine Bahnplanung sinnvoll und würde, erweitert um die Fähigkeiten zur Prädiktion, auch eine Bewegungsplanung zulassen. Zur Lösung von Einparkproblemen existieren viele Ansätze auf Basis geschlossener mathematischer Beschreibungen. Hierfür müssen allerdings die dafür notwendigen Parameter identifiziert werden, was bei Nichtexistenz einer symbolischen Beschreibung schwierig ist. Die Verwendung einer gitterbasierten Beschreibung des Arbeitsraumes könnte als Ansatz für einen universellen Bahnplanungsalgorithmus dienen, mit dem sowohl Bahnen zur schnellen Fortbewegung, als auch spezielle Manöver zum Einparken oder Wenden geplant werden können.

Kapitel 3

Verhaltenssteuerung

3.1 Einführung

Im vorliegenden Kapitel wird die Verhaltenssteuerung für ein kognitives Automobil diskutiert, entworfen und implementiert. Die Verhaltenssteuerung wird als biologisch motiviertes Verhaltensnetzwerk modelliert (vgl. Anhang A.3 und [Albiez 06]). Zunächst werden in Abschnitt 3.2 die Anforderungen an eine solche Steuerung identifiziert. In Abschnitt 3.3 werden Gründe für die Wahl zugunsten von Verhaltensnetzwerken dargelegt, bevor die Verhaltenssteuerung entworfen wird und die Schnittstellen zur Integration in das Gesamtsystem erläutert werden. Das für die Umsetzung der Steuerung entwickelte Softwaresystem wird in Abschnitt 3.4 näher betrachtet. Abschließend geht Abschnitt 3.5 auf Möglichkeiten zur Sammlung von Erfahrungswissen ein, das die Grundlage für spätere Lernverfahren bildet.

3.2 Anforderungen an die Verhaltenssteuerung

Die von Prof. Rodney Brooks in [Brooks 86] formulierten Anforderungen an eine Robotersteuerung gelten ebenfalls für die Verhaltenssteuerung eines kognitiven Automobils und müssen beim Entwurf sowohl des Verhaltensnetzwerkes als auch des Software-Systems entsprechende Beachtung finden (vgl. Abschnitt A.2). Angewandt auf die spezielle Problematik lassen sich die Anforderungen wie folgt formulieren:

1. Anforderung: Unterstützung vielfältiger Ziele. *Fahrentscheidungen im Straßenverkehr setzen sich üblicherweise aus mehreren Zielen zusammen, die unterschiedliche Zeitskalen besitzen und sich teilweise widersprechen. Wird bspw. das Ziel verfolgt, an der nächsten Kreuzung abzubiegen, so existieren parallel dazu weitere Ziele, um Abstände und Geschwindigkeiten einzuhalten oder Vorfahrtsregeln zu überprüfen. Die Verhaltenssteuerung muss in der Lage sein, Art und Wichtigkeit der Ziele zu beurteilen und sich auf dieser Grundlage für ein Ziel zu entscheiden.*

2. Anforderung: Unterstützung vielfältiger Sensoren. *Die Sensordatenverarbeitung und -fusion erfolgt innerhalb der Systemarchitektur an anderer Stelle und ist ein der Verhaltenssteuerung zuarbeitendes Teilsystem. Die Situationsinterpretation erweitert die*

Sensordatenverarbeitung um eine logische Komponente, sodass für die Verhaltenssteuerung abstrakte Daten zur Verfügung stehen (vgl. Abschnitt 3.3.1 und Anhang E.1). Den Forderungen nach Sensordatenfusion und Redundanz wird in den genannten Komponenten entsprochen.

3. Anforderung: Robustheit. *Die Forderung nach Robustheit ist für den Betrieb eines kognitiven Automobils im Straßenverkehr von besonderer Relevanz, weshalb dafür eine Unterteilung in drei wichtige Aspekte erfolgt:*

Datenfehlertoleranz: *Die Verhaltenssteuerung bezieht Sensordaten nicht unmittelbar, sondern lediglich über Wahrnehmungs- und Interpretationskomponenten. Dennoch soll bei fehlerhaften oder unvollständigen Eingangsdaten auf dieser Ebene eine Entscheidungsfindung möglich sein, um das System in einen sicheren Zustand zu überführen.*

Rechenfehlertoleranz: *Die Verhaltenssteuerung soll in der Lage sein, auch bei einem Ausfall interner Software-Komponenten (bspw. einzelner Verhalten) Entscheidungen zu treffen. Eine Absicherung gegenüber Hardware-Ausfällen wird auf Ebene der Regelungsschicht realisiert und ist daher für den Entwurf der Verhaltenssteuerung irrelevant.*

Anpassungsvermögen: *Der Straßenverkehr unterliegt besonders im innerstädtischen Bereich einer großen Dynamik. In dieser Umgebung muss das Fahrzeug in der Lage sein, sich den schnell wechselnden Bedingungen umgehend anzupassen. Dies setzt entsprechend eine schnelle Aktualisierung und Verarbeitung der Daten in der Verhaltenssteuerung voraus.*

4. Anforderung: Erweiterbarkeit & Skalierbarkeit. *Diese Forderung wurde von Brooks im Bezug auf zusätzliche Sensorik und die für eine Auswertung notwendige Steigerung der Rechenleistung gestellt. Im Hinblick auf aktuelle Versuchsträger und deren technische Voraussetzungen ist diese Forderung kaum mehr relevant. In dieser Arbeit wird diese Anforderung deshalb so verstanden, dass es möglich sein muss, die Verhaltenssteuerung um weitere Verhalten zu erweitern und darüber hinaus deren Möglichkeiten auszudehnen.*

Über diese vier Brooks'schen Anforderungen hinaus werden an ein kognitives Automobil für einen sicheren Betrieb im Straßenverkehr weitere spezielle Anforderungen gestellt:

5. Anforderung: Parametrierbarkeit. *Wie menschliche Verkehrsteilnehmer, so muss sich auch die Verhaltenssteuerung an der Straßenverkehrsordnung (StVO) orientieren. Diese Verordnung muss als rechtliche Grundlage mit Regeln und Vorschriften in das System aufgenommen werden. Die Verhaltenssteuerung muss dafür geeignete Möglichkeiten bereitstellen, um diese Regeln zu hinterlegen.*

6. Anforderung: Nachvollziehbarkeit. *Für die Entwicklung und Verifikation solcher Systeme ist eine Möglichkeit zur detaillierten Nachvollziehbarkeit der Entscheidungen unabdingbar. Diese Anforderung besitzt große rechtliche Relevanz, da nur so ein zur Markt-*

freigabe notwendiger Sicherheitsnachweis erbracht werden kann. Die Verhaltenssteuerung muss auch dafür Möglichkeiten bieten, z. B. mit einer Erklärungskomponente.

7. Anforderung: Testbarkeit und Simulationsunterstützung. *Für die Entwicklung und Erweiterung muss die Verhaltenssteuerung Möglichkeiten zum Test einzelner Komponenten vorsehen. Dies schließt zudem die Verfügbarkeit einer Simulationsumgebung für den Test des Gesamtsystems unabhängig vom realen Versuchsträger ein. Die Verhaltenssteuerung soll sich in der Simulation so verhalten wie im realen Betrieb. Von der Echtzeit abweichende Simulationsgeschwindigkeiten oder auch das Pausieren einer Simulation dürfen auf die Funktionsfähigkeit keinen Einfluss haben.*

Insbesondere die letztgenannte Anforderung hat eine große Auswirkung auf das zu entwickelnde Software-System (vgl. Abschnitt 3.4).

3.3 Entwurf

Menschliche Autofahrer besitzen keine lückenlose Beschreibung der Umwelt, können sich durch geschickte Verhaltensweisen und Reaktionen aber offensichtlich gut an dynamische Umgebungen anpassen. Die Fahrhandlungen setzen sich beim Menschen üblicherweise aus mehreren, teilweise unbewusst ausgeführten Unterhandlungen zusammen. Diese besitzen je nach Art und Wichtigkeit unterschiedlichen Einfluss auf die Gesamtausführung. In Notfallsituationen erfolgt eine Fokussierung auf bestimmte vorgefertigte Handlungen, die besonders schnell ausgeführt werden (bspw. um eine Kollision zu vermeiden). Ein menschlicher Fahrer verfügt demzufolge über die Fähigkeit, Entscheidungen schnell über mehrere Hierarchieebenen hinweg zu treffen.

Bei der Auswahl der Architektur für eine Verhaltenssteuerung ist eine Orientierung am menschlichen Fahrer als Vorbild insofern sinnvoll, als die negativen Aspekte menschlichen Fahrens wie Unachtsamkeit, Unvernunft und Selbstüberschätzung nicht modelliert werden und in diesem Zusammenhang keine Rolle spielen. Letztendlich müssen die Modelle für die Verhaltenssteuerung von menschlichen Fahrern hinterlegt und von diesen nachvollzogen und bewertet werden. Menschliche Entscheidungs- und Fahrverhalten sollen aus den folgenden Gründen die Modellierung der Verhaltenssteuerung bestimmen:

- Menschliche Fahrer beherrschen komplexe dynamische Verkehrssituationen.
- Modellwissen ist durch die gegebene Komplexität und Situationsvielfalt nie vollständig vorhanden. Ein menschlicher Fahrer macht dies durch reaktive Verhalten wett.
- Menschliche Entwickler müssen die Entscheidungen des Automobils nachvollziehen und bestätigen können – sie fungieren als Lehrer und Evaluator für das System.

Das Ziel ist dabei nicht die Nachmodellierung eines menschlichen Fahrers, sondern eine Zerlegung von Fahrverhalten und -handlungen in eine für den Menschen verständliche und nachvollziehbare Form [Schröder 07, Hoffmann 06]. Damit würde direkt der Anforderung 6.)

Nachvollziehbarkeit Rechnung getragen. Die von J. Albiez [Albiez 06] entwickelten biologisch motivierten Verhaltensnetzwerke sind in Anhang A.3 ausführlich beschrieben, an dieser Stelle soll zunächst eine kurze Einführung ausreichen.

Verhaltensnetzwerke

Verhaltensnetzwerke sind der Gruppe der biologisch motivierten Steuerungsarchitekturen zuzuordnen [Albiez 06]. Diese Form der Verhaltensmodellierung wurde ursprünglich zur Steuerung von Laufmaschinen entworfen und orientiert sich an der Struktur des zentralen Nervensystems [Albiez 01b, Albiez 01a]. Mit Verhaltensnetzwerken lassen sich schwach-temporalhierarchische Verhaltensteuerungen aufbauen. Die schwache Hierarchie erlaubt eine Verbindung zwischen Verhalten über Zwischenschichten hinweg, die temporalen Eigenschaften hingegen resultieren aus einer stetigen Änderung der aktiven Struktur zur Laufzeit.

Der Kern eines Verhaltensbausteins ist seine Übertragungsfunktion $F(e)$, welche die Sensordaten e in Ausgaben u umwandelt (vgl. Abbildung 3.1). Für die Art der Übertragungsfunktion werden keine Einschränkungen vorgenommen – es ist das für diesen Baustein geeignetste Verfahren zu wählen. Während auf höheren Ebenen neuronale Netze oder endliche Automaten zum Einsatz kommen, so sind klassische PID-Regler oder andere Regelungsstrategien auf niederen Ebenen denkbar. Die Platzierung von Verhalten in den jeweiligen Schichten erfolgt gemäß den in der Natur zu beobachtenden Arten von Verhalten. Planende Verhalten mit mittel- oder langfristigem Horizont sind in höheren Schichten angesiedelt, reflexive Verhalten mit schnellem Antwortverhalten und kurzer Aktivitätsdauer in niederen Schichten, dazwischen existieren Mischformen.

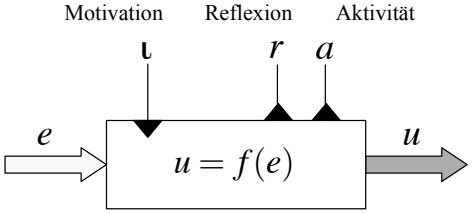


Abbildung 3.1: Schema eines Verhaltensbausteins mit Ein- und Ausgabedaten e und u. Die Steuersignale ι, r und a dienen der Abstimmung mit anderen Verhaltensbausteinen.

Eine wichtige und im Vergleich zu anderen Architekturen besondere Eigenschaft ist das Prinzip der virtuellen Sensoren. Neben der Motivation, welche als Steuereingang dient, werden von jedem Verhalten Werte für Reflexion und Aktivität als Steuersignale ausgegeben. Diese beiden Signale spiegeln den internen Zustand des Verhaltens wider: Die Reflexion eine Art Zufriedenheit, mit welcher die Abweichung von einem Zielzustand angegeben wird. Die Aktivität ist eine Maßzahl für die Leistung, die der Baustein momentan, relativ zu einer Maximalleistung, erbringt. Motivation, Reflexion und Aktivität sind als $\iota, r, a \in [0, 1]$

definiert. Durch die standardisierten Schnittstellen bleiben die Signale austauschbar und sind für alle Verhaltensbausteine verwendbar, ohne Wissen über die interne Struktur des Signalerzeugers zu benötigen.

Verhaltensnetzwerke sind für eine solche Modellierung aus den folgenden Gründen besonders gut geeignet:

1. Verhaltensbasierte Systeme besitzen durch deliberative und reaktive Komponenten die geforderten Eigenschaften nach Entscheidungskompetenz und Robustheit (Anforderungen 1 und 3).

2. Biologisch motivierte Verhaltensnetzwerke liefern mit Reflexionswerten eine stetige Bewertung der Verhaltensausführung. Diese können eine Grundlage für Lernverfahren bilden (Anforderung 4).

3. Verhaltensnetzwerke besitzen eine schwache Hierarchie. Dies bedeutet, dass Sicherheits- oder Notfallverhalten jederzeit über Ausführungsschichten hinweg ausgeführt werden können (Anforderung 3).

4. Verhaltensnetzwerke erlauben eine anforderungsabhängige Rekonfiguration ihrer Struktur zur Laufzeit durch gezielte Motivation bestimmter Verhalten (Anforderung 3).

5. Es bestehen vielfältige Möglichkeiten der Parametrierung des Netzwerkes durch Festlegung der Motivations-, Aktivitäts- und Reflexionsfunktionen (Anforderung 5).

Bevor auf den Entwurf des Netzwerkes eingegangen wird, sollen im folgenden Abschnitt zunächst die Schnittstellen zu den Komponenten der Situationsinterpretation und der Bahnplanung betrachtet werden.

3.3.1 Schnittstellen

Situationsinterpretation

Die Situationsinterpretation führt eine Analyse der aktuellen Verkehrslage auf Basis des vorliegenden Sensorhorizonts durch und bereitet die von der Wahrnehmung gelieferten Daten für die Entscheidungskomponente in einem Weltmodell auf. Sie vervollständigt fehlende Wahrnehmungsdaten, wie z. B. nicht erkannte Fahrspuren, mit Hintergrundwissen, identifiziert logische Beziehungen zwischen erkannten Objekten (bspw. eine Zuordnung von Fahrzeugen zu Spuren) und führt zudem für andere Verkehrsteilnehmer eine Verhaltenserkennung durch. Die Missionsplanung findet ebenfalls in der Interpretationskomponente statt, sodass der zu fahrende Weg durch eine gezielte Markierung der Fahrspuren dargestellt wird.

Die Gründe für eine Trennung der Interpretations- und Entscheidungskomponenten sind die bessere Testbarkeit und Nachvollziehbarkeit der Einzelkomponenten, aber auch eine unabhängige Verwendung derselben. Eine Situationsinterpretation könnte bspw. im Rahmen eines Fahrerassistenzsystems Warnmeldungen ausgeben, ohne wie in der vorliegenden Arbeit

aktive Fahrentscheidungen zu beeinflussen. Die Situationsinterpretation wird im Teilprojekt B1: *Situationsinterpretation und Verhaltenserkennung* des Sonderforschungsbereiches entwickelt [Vacek 07].

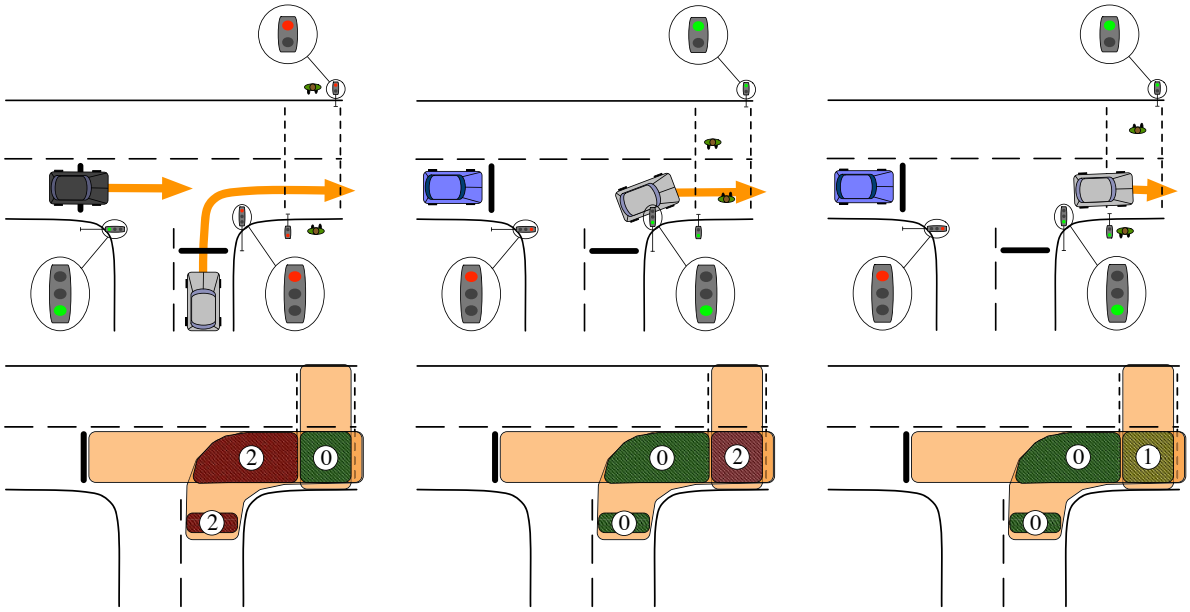
Die Daten der Situationsinterpretation werden, wie alle Daten innerhalb der Systemarchitektur, über die Echtzeitdatenbank (RTDB) verteilt. Als Informationscontainer werden sogenannte Datenbankobjekte definiert, die technisch als Strukturtypen der Programmiersprache C realisiert sind und von allen Anwendungen gelesen und geschrieben werden können. Ein von der Situationsinterpretation aufbereitetes Datenbankobjekt mit Spurinformation enthält bspw. neben den eigentlichen Spurdaten auch Informationen über Nachbarspuren und zugeordnete Verkehrsteilnehmer. Das folgende Programmbeispiel zeigt die C-Struktur, wie sie für den Eintrag einer (interpretierten) Fahrspur in der RTDB definiert wurde:

```
 typedef struct {
  //! Fahrtrichtung dieser Fahrspur
  B1_Enum_Driving_Direction m_driving_direction;
  //! rechte Begrenzung der Fahrspur
  B1_Enum_Lane_Border m_right_border;
  //! linke Begrenzung der Fahrspur
  B1_Enum_Lane_Border m_left_border;
  //! rechte Nachbarspur
  kogmo_rtdb_objid_t m_right_neighbour_lane;
  //! linke Nachbarspur
  kogmo_rtdb_objid_t m_left_neighbour_lane;
...
  //! Liste von Hindernissen auf der Fahrspur
  kogmo_rtdb_objid_t m_list_of_obstacles[B1_MAX_OBSTACLES_IN_LANE];
  //! Anzahl der Hindernisse
  unsigned int m_num_obstacles;
  //! Liste von Verkehrsteilnehmern auf der Fahrspur
  kogmo_rtdb_objid_t m_list_of_participants[B1_MAX_PARTICIPANTS_IN_LANE];
  //! Anzahl der Verkehrsteilnehmer
  unsigned int m_num_participants;
  //! zulaessige Hoechstgeschwindigkeit, -1: unbekannt
  double m_max_topspeed;
...
  //! Geometrie der Fahrspur
  kogmo_rtdb_subobj_b1_lane_geometry m_geometry;
} kogmo_rtdb_subobj_b1_lane_t;
```

Die Spurgeometrie selbst ist in `kogmo_rtdb_subobj_b1_lane_geometry` enthalten und besteht aus einer Anzahl von Punktpaaren. Diese Objektdefinition soll als beispielhafte Veranschaulichung der technischen Schnittstelle zur Interpretationskomponente genügen. Um die Interpretationsdaten auf Seite der Verhaltenssteuerung auszuwerten, dient die im Abschnitt 3.3.2 beschriebene Komponente *Szenario Monitor* [Jagszent 07a].

Neben den Informationen über Einzelobjekte wie Fahrspuren oder Verkehrsteilnehmer liefert die Situationsinterpretation sogenannte Konfliktflächen. Diese stellen potenzielle Gefahrenbereiche dar und werden vor der Freigabe einer Fahrhandlung von der Verhaltenssteuerung überprüft (vgl. den sog. *Szenario Monitor* in Abschnitt 3.3.2). Neben regulären Fahrspu-

ren werden zur Identifikation von Konfliktflächen sog. *virtuelle* Spuren betrachtet. Diese sind als erkannte oder vermutete Spuren eines beweglichen Hindernisses definiert, welches Auswirkungen auf das Fahrverhalten haben kann. So spannen bspw. Fußgänger beim Überqueren einer Straße virtuelle Spuren auf. Zudem werden virtuelle Spuren durch Objekte mit besonderem Einfluss auf das Verkehrsgeschehen definiert, z. B. durch Ampeln.



(a) Zeitpunkt T_1 (b) Zeitpunkt T_2 (c) Zeitpunkt T_3

Abbildung 3.2: Beispielszenario eines Abbiegevorganges an einer Kreuzung. Zum Zeitpunkt T_1 existieren zwei nicht überfahrbare Konfliktflächen für das Eigenfahrzeug (hellgrau), hervorgerufen durch eine Ampel und die Bewegung des schwarzen Fahrzeuges. Zum zweiten Zeitpunkt T_2 sind diese Konfliktflächen befahrbar, die Überschneidung der virtuellen (belegten) Spur *Fußgängerüberweg* führt jedoch zu einer weiteren, nicht befahrbaren Konfliktfläche. Zum Zeitpunkt T_3 ist diese Konfliktfläche frei. Da Fußgänger in der nahen Umgebung sind, darf diese jedoch nur langsam überfahren werden.

Eine Konfliktfläche definiert sich als geometrische Beschreibung einer Überschneidung der eigenen (geplanten) Fahrspur mit jeder anderen regulären oder virtuellen Spur. Konfliktflächen werden anhand ihres aktuellen Gefahrenpotenzials unterschieden. Sie sind dazu in drei Grade eingeteilt:

Grad 0: Es besteht momentan kein Konflikt, der Bereich kann ohne Einschränkungen passiert werden. Konfliktflächen mit Grad 0 können durch äußere Einflüsse in einen höheren Grad wechseln (bspw. durch eine rote Ampel).

Grad 1: Die Fläche stellt unter Umständen einen Konflikt dar und darf nur mit Einschränkungen überfahren werden, bspw. mit reduzierter Geschwindigkeit. Freie Fußgängerüberwege stellen Konfliktflächen dieses Typs dar.

Grad 2: Es besteht ein Konflikt, die Fläche darf keinesfalls überfahren werden. Eine rote Ampel oder ein Fußgängerüberweg mit Fußgänger sind Vertreter dieses Typs.

Abbildung 3.2 veranschaulicht die Klassifikation von Konfliktflächen am Beispiel eines Abbiegevorganges an einer Kreuzung.

Bahnplanung

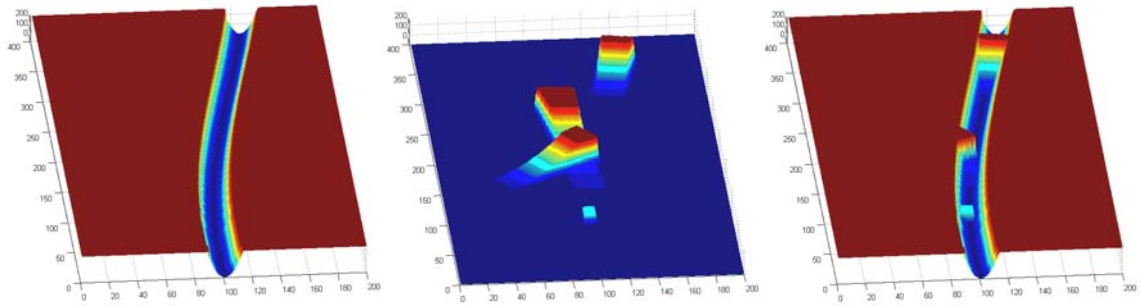
Das Bahnplanungsmodul ist eine separate Systemkomponente und übernimmt vielfältige Aufgabenstellungen wie die Planung von Einparkvorgängen oder Bahnen in unstrukturierter Umgebung sowie das Planen von Fahrstrecken auf Fahrspuren oder in Notsituationen (vgl. Kapitel 4). Diese Abtrennung von der Verhaltensentscheidung eröffnet Möglichkeiten der separaten Entwicklung und Testbarkeit sowie den Einsatz in vielfältigen Anwendungen. Sie erfordert aber zugleich eine Schnittstelle, die mehreren wichtigen Anforderungen genügen muss:

1. Gefährliche oder unerwünschte Bereiche müssen für die Bahnplanung ausgeschlossen werden.
2. Als Ausgabe der Verhaltenssteuerung muss eine gemeinsame Fahrintention aller Verhalten ausgedrückt werden. Dabei darf diese Information nicht zu eng gefasst sein und die Bahnplanung muss Spielraum erhalten.
3. Der Bahnplanung muss eine Prädiktion des Verkehrsflusses auf symbolischer Ebene übermittelt werden, um eine Planung mit Vorausschau durchführen zu können.

Die Berücksichtigung dieser Anforderungen ist mit der Einführung sog. *dynamischer Gefahrenkarten* möglich. Diese werden als Grundlage für die Bahnplanung in Abschnitt 4.2 ausführlich erklärt. An dieser Stelle steht die Erzeugung und Fusion der dynamischen Gefahrenkarten durch die Verhalten der reaktiven Ausgabeschicht im Vordergrund. Eine kurze Erklärung soll zunächst genügen:

Dynamische Gefahrenkarten sind ein Werkzeug, um Ausgaben reaktiver Verhalten der Quersteuerung zu einer gemeinsamen Antwort zu fusionieren. Die reaktiven Verhalten erzeugen zunächst individuelle Gefahrenkarten, die je nach Typ des Verhaltens Fahrintentionen oder Hindernisse beschreiben (vgl. Abbildung 3.3).

Eine Gefahrenkarte ist technisch als Graustufen-Rastergrafik implementiert, bei der jeder Pixel einen bestimmten Bereich der Umgebung hinsichtlich der Gefahr (bzw. dem Risiko) beschreibt, diesen Bereich zu befahren. Die geringste Gefahr wird durch den Wert 0 repräsentiert, die höchste durch 255. Durch Vereinbarungen mit der Bahnplanung können Bereiche mit Gefahrenwert größer einem Grenzwert von der Planung vollständig ausgeschlossen werden. Über ein solches Grauwertbild wird die Gefahr zu einem bestimmten Zeitpunkt beschrieben. Ein zeitlicher Verlauf von Gefahrenwerten ergibt sich durch die Kombination mehrerer Gefahrenkarten zu einer dynamischen Gefahrenkarte. So können bestimmte Ereignisse in die Zukunft prädiziert werden. Das Verhalten *Kollision vermeiden* macht von dieser Möglichkeit Gebrauch, indem es die Bewegungen anderer Verkehrsobjekte schätzt.



(a) (b) (c)

Abbildung 3.3: Gefahrenkarten repräsentieren entweder Fahrintentionen wie *Spur halten* (a) oder beschreiben Hindernisse durch Verhalten wie *Kollision vermeiden* (b). Das Bahnplanungsmodul erhält die Kombination aller entstandenen Gefahrenkarten (c).

Die Erzeugung und Fusion von Gefahrenkarten erfolgt in drei Schritten, die für die jeweiligen Schichten mit gleichen Zeitpunkten durchgeführt werden (vgl. Abbildung 3.4). Nach der Erzeugung im ersten Schritt wird ein zweistufiger Fusionsprozess durchgeführt, um alle Ausgangsdaten der kooperativ ablaufenden Verhalten zu berücksichtigen. Verhalten wie *Auf Straße bleiben* oder *Spur halten* beschreiben Fahrintentionen oder fahrbare Bereiche. Die Ausgaben dieser Verhalten werden zunächst durch Minimum-Fusion in eine gemeinsame Darstellung überführt. Die resultierende Gefahrenkarte wird in einem zweiten Fusionsschritt mit den Ausgaben der Verhalten maximumfusioniert, welche Hindernisse oder nicht befahrbare Bereiche beschreiben. Bislang beschränkt sich der letztgenannte Typ auf das Verhalten *Kollision vermeiden*. Als Ergebnis entsteht eine dynamische Gefahrenkarte, die es der Bahnplanung erlaubt, Fahrzeugkonfigurationen auf Gültigkeit zu überprüfen und einen Gefahrenwert in die Bewertungsfunktion zu integrieren.

Neben der Gefahrenkarte existiert ein weiteres Datenbankobjekt, um Informationen über die zulässige Höchstgeschwindigkeit, Sollwerte für Geschwindigkeit und Beschleunigung, Fahrtrichtung, etwaige Zielpositionen und den gewünschten Planungsmodus zu übermitteln. Abschnitt 4.2 erläutert weitere Details zu Gefahrenkarten und zur Erzeugung von Gefahrenpotenzialen aus Objektinformationen.

3.3.2 Aufbau und Strukturierung des Verhaltensnetzwerkes

Ein grundlegender Unterschied bei der Verwendung von Verhaltensnetzwerken für kognitive Automobile – im Gegensatz zur ursprünglichen Anwendung auf Laufmaschinen – besteht beim Entwurf des Netzes. Ein Top-Down-Entwurf mit der Definition von übergeordneten Metaverhalten ist zwar auch hier möglich, die Unterteilung der Verhalten in kinematotopische Gruppen lässt sich jedoch für Fahrzeuge kaum anwenden. Stattdessen soll eine semantische Hierarchisierung der Verhalten anhand ihres individuellen Zeithorizontes erfolgen. Eine Unterscheidung in drei Gruppen (Schichten) soll dazu genügen. Die Verhalten mit ähnlichem Zeithorizont finden sich innerhalb derselben Schicht wieder. Aus diesen drei Schichten ergibt

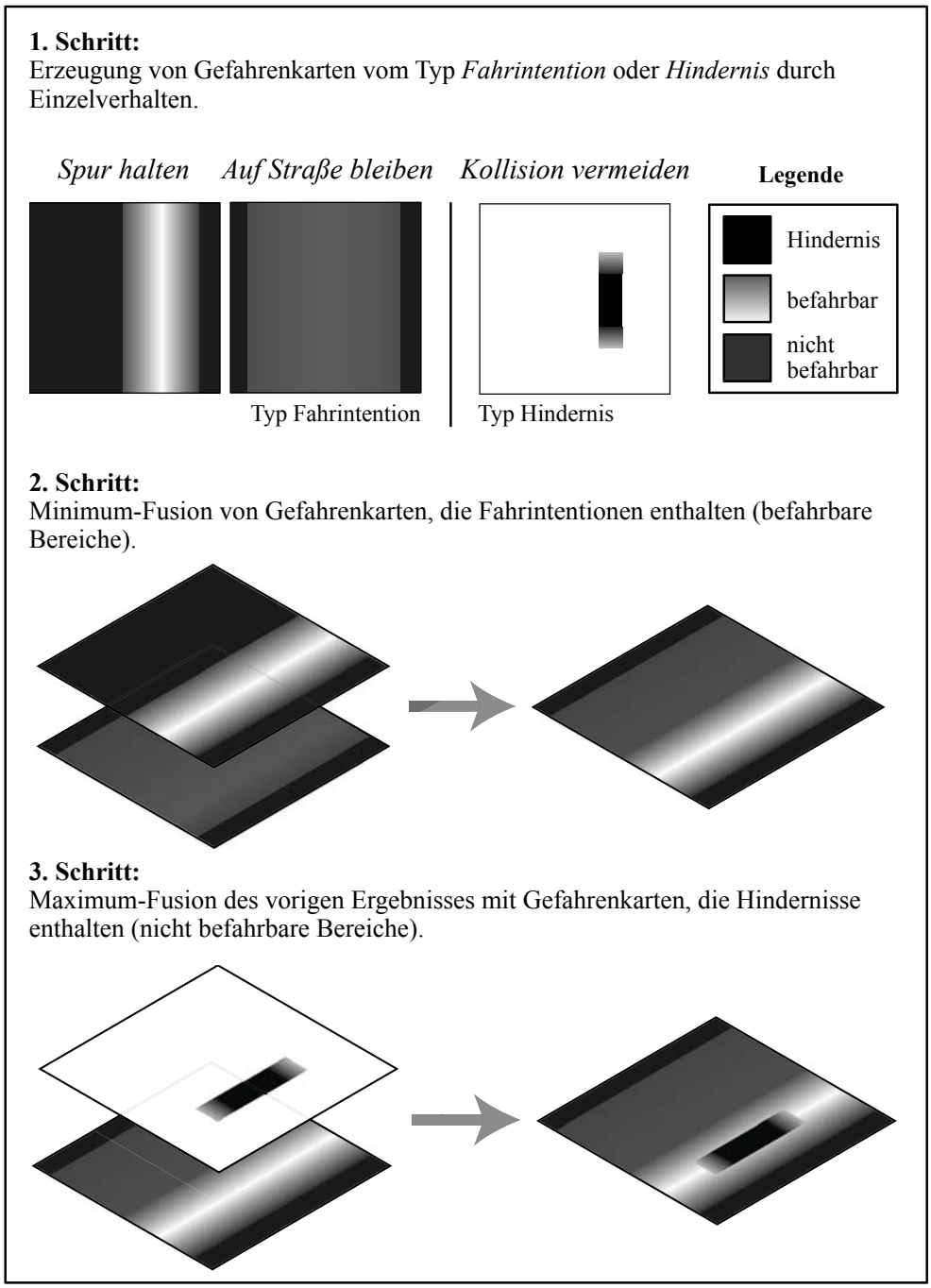


Abbildung 3.4: Mehrstufiger Prozess zur Erzeugung und Fusion von Gefahrenkarten. Bildquelle: [Jagszent 08].

sich die Struktur des Verhaltensnetzwerkes, wie sie unter anderem in der Verhaltensmodellierung von Autofahrern verwendet wird [Michon 85]:

- **Strategische Schicht**
  Die Verhalten dieser Schicht werden als Verhaltensfähigkeiten oder Verhaltensweisen bezeichnet. Sie besitzen stark planenden Charakter, einen großen Zeithorizont und agieren auf hoher Abstraktionsstufe. Verhalten auf dieser Stufe sind vergleichbar mit Anweisungen eines Fahrlehrers wie bspw. *Straße folgen*. Sie setzen den Fokus für unterlagerte Verhalten und koordinieren diese. Zu jedem Zeitpunkt ist immer nur eine dieser Anweisungen möglich. Die Verhalten auf dieser Stufe sind deshalb komplementär und nur wechselseitig aktiv.

- **Taktische Schicht**
  Die in dieser Schicht angesiedelten Verhalten stellen real ausführbare *Handlungsfähigkeiten* oder *Handlungen* dar. Es handelt sich hierbei weniger um langfristige strategische Entscheidungen, sondern um bewusst durchgeführte Fahrhandlungen wie bspw. *Überholen*, die jedoch auch aus mehreren Phasen bestehen können. Taktische Verhalten greifen auf Verhalten der reaktiven Schicht zurück, um Fahrhandlungen umzusetzen.

- **Reaktive Schicht**
  Die Verhalten dieser Schicht können auch als *Fertigkeiten* oder *Handlungsprimitive* angesehen werden. Reaktive Verhalten sind vergleichbar mit den unbewusst ausgeführten Verhalten eines menschlichen Fahrers. Sie besitzen einen kurzen Zeithorizont und setzen eine eng gefasste Aufgabe um, wie bspw. Spur halten oder einen Spurwechsel. Die im Gegensatz zu den Verhalten der anderen Schichten reflexartigen Eigenschaften prädestinieren diese Verhalten zudem für die Umsetzung elementarer Sicherheitskriterien.

Anhand dieser Unterscheidungsmerkmale werden die Einzelverhalten in der Netzwerkstruktur platziert. Die Identifikation möglicher Einzelverhalten erfolgt dabei mit dem Beantworten der folgenden Fragestellungen:

1. Strategisch: Welche Vorgaben auf Meta-Ebene können für das Fahrzeug gemacht werden (vergleichbar mit den Anweisungen eines Fahrlehrers)?

2. Taktisch: Welche Fahrhandlungen führt ein menschlicher Fahrer (bewusst) aus?

3. Reaktiv: Welche Fahrverhalten erfolgen unterbewusst oder in Ausnahmesituationen?

Neben einer vertikalen Zuordnung zu den drei Schichten lassen sich die Verhalten der reaktiven Schicht zudem entsprechend ihrem Einfluss auf die Längs- bzw. Quersteuerung des Fahrzeuges einteilen. Als Ergebnis folgt ein Verhaltensnetzwerk gemäß Abbildung 3.5, das alle für eine grundlegende Fahrzeugsteuerung notwendigen Verhalten enthält. In den folgenden Abschnitten werden detaillierte Erläuterungen zu den einzelnen Verhalten gegeben, zunächst wird jedoch der *Szenario Monitor* als Verbindungsglied zwischen strategischer und taktischer Ebene vorgestellt.

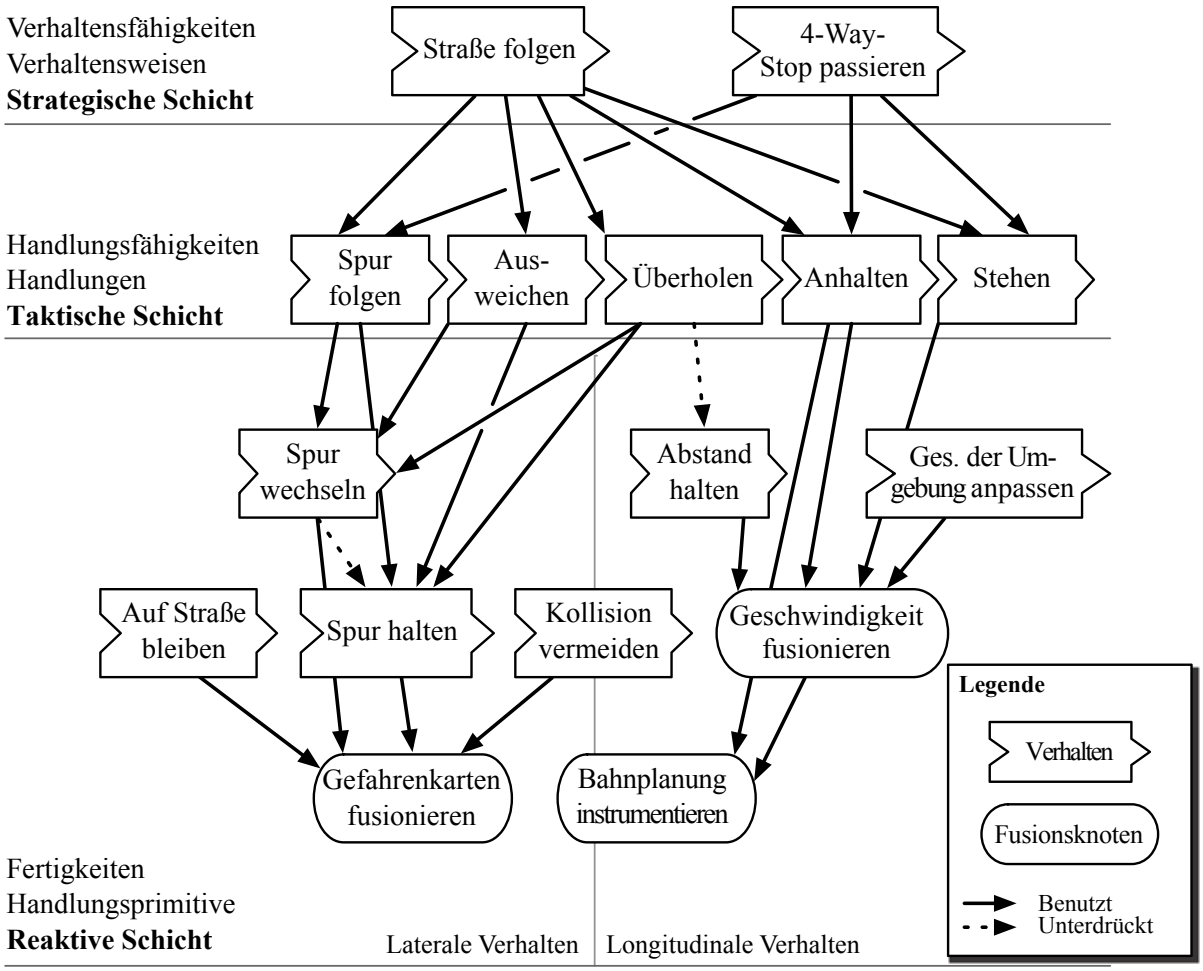


Abbildung 3.5: Verhaltensnetzwerk mit drei Schichten und Verhalten zur Umsetzung grundlegender Fahrmanöver. Die Verhalten der reaktiven Schicht haben Einfluss auf die Längs- oder auf die Quersteuerung des Fahrzeuges.

Szenario Monitor

Der *Szenario Monitor* wurde als weitere Komponente an der Schnittstelle zwischen Situationsinterpretation und Verhaltenssteuerung entwickelt [Jagszent 07a, Jagszent 07b]. Er wertet die Anfragen der strategischen Verhalten nach Zulässigkeit der Ausführung eines taktischen Verhaltens aus (vgl. Abbildung 3.6). Diese Anfrage erfolgt bei jedem Durchlauf der strategischen Ebene für das aktuell ausgeführte und geplante taktische Verhalten. Der Szenario Monitor kennt die für das jeweilige taktische Verhalten notwendigen Bedingungen und löst diese mithilfe der Interpretationsdaten auf. Nach dieser Überprüfung wird ein einzelnes oder eine Menge von Verhalten freigegeben, die mit eventuellen Einschränkungen (bspw. reduzierte Geschwindigkeit) ausgeführt werden dürfen.

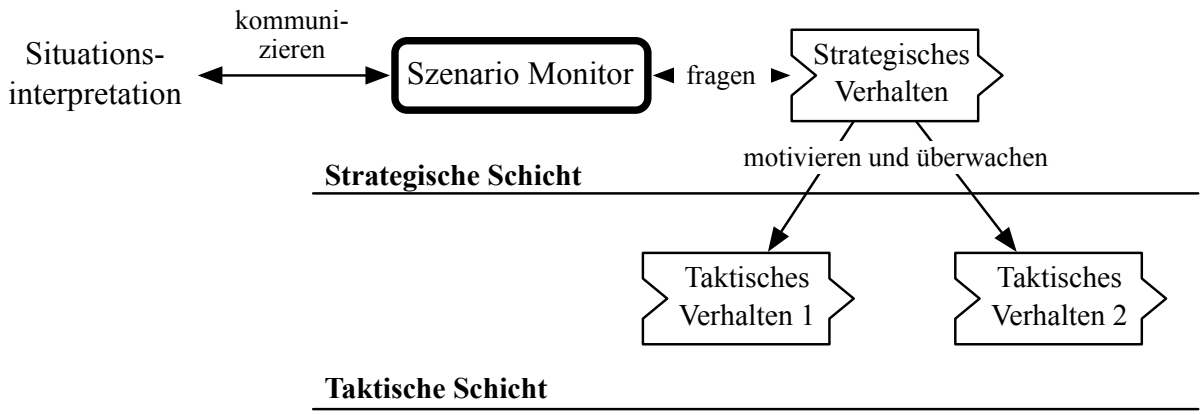


Abbildung 3.6: Schnittstelle zwischen Situationsinterpretation und Verhaltenssteuerung. Der Szenario Monitor nimmt Interpretationsdaten entgegen und wertet mit deren Hilfe die von den strategischen Verhalten gestellten Anfragen aus.

Die Auslagerung der Überprüfungsroutinen in den Szenario Monitor hat den Grund, dass sich ein Großteil dieser Zulässigkeitstests auf hinterlegte Verkehrsregeln bezieht, die sich für verschiedene Verhalten oft nicht unterscheiden. Die Überprüfungsroutinen werden deshalb in sog. Überprüfungsprimitive zerteilt. Für jedes angefragte taktische Verhalten ist eine Liste solcher elementarer Regeln abzuarbeiten. Aktuell sind die folgenden Überprüfungsprimitive implementiert:

laneAvoidObstacle
: Überprüft die Befahrbarkeit der aktuellen Spur und der Gegenspur.

zoneTestTrafficLight
: Überprüft, ob Ampeln die Fahrt beeinflussen.

zoneTestStopSignAndLine
: Lässt das Auto erst weiterfahren, nachdem es an dieser Konfliktfläche angehalten hat.

zoneTestPedestrianCrossing
: Überprüft, ob sich Fußgänger auf einem oder nahe einem Fußgängerüberweg befinden und ob ein Passieren des Fußgängerüberwegs erlaubt und möglich ist.

Bei einem Durchführbarkeitstest für ein taktisches Verhalten werden die Überprüfungsprimitive nach dem in Algorithmus 1 dargestellten Schema ausgewertet. Aufgrund der großen Redundanz von Überprüfungsprimitiven in den taktischen Verhalten wird ein Caching-Mechanismus eingesetzt, der sicherstellt, dass komplexe Berechnungen nicht mehrfach ausgeführt werden und der Test effizient gestaltet wird. Nach jedem Durchlauf der strategischen Schicht wird der Cache geleert, sodass die Überprüfungsroutinen stets mit den aktuellen Daten arbeiten.

Algorithm 1 Vereinfachte Darstellung des Durchführbarkeitstests im *Szenario Monitor*

```
1: sort the conflictzones by distance to the car
2: primitives ← list of check-primitives for the currently checked behavior
3: for all zone in conflictzones do
4:    for all check-primitive in primitives do
5:       test check-primitive with current zone
6:       if test returned false then
7:          return false
8:       end if
9:    end for
10: end for
11: return true
```

3.3.3 Verhalten der einzelnen Schichten

Strategische Schicht

Der von der Situationsinterpretation übermittelte Missionsbefehl resultiert direkt in der Selektion eines zugeordneten strategischen Verhaltens. Den beiden Missionsbefehlen *normales Fahren* und *4-Way-Stop passieren* entsprechen dabei die strategischen Verhalten *Straße folgen* und *4-Way-Stop passieren*. Nur eines dieser Verhalten kann zu jedem Zeitpunkt aktiv sein; eine Koppelung von Verhalten ist auf dieser Ebene nicht möglich.

Die sich aus der Topographie des Straßennetzes ergebenden logischen Zusammenhänge werden durch die allgemeine Darstellung der Konfliktflächen modelliert und gelöst (vgl. Abschnitt 3.3.1). Das strategische Verhalten *Straße folgen* fragt die Informationen über Konfliktflächen beim Szenario Monitor an. Auf eine explizite Behandlung verschiedener Verkehrssituationen kann durch die Konfliktflächendarstellung verzichtet werden. Eine Unterscheidung in mehrere Meta-Verhalten wie bspw. *in x Metern rechts/links abbiegen* oder *dem Straßenverlauf folgen* ist insofern nicht notwendig.

Die strategischen Verhalten bedienen sich der Handlungsfähigkeiten der taktischen Sicht und motivieren diese. Das Zusammenspiel sieht am Beispiel von *4-Way-Stop passieren* folgendermaßen aus:

1. Zunächst wird beim Anfahren an die Kreuzung das Verhalten *Spur folgen* motiviert.
2. Bei Annäherung an die Kreuzung wird zudem das Verhalten *Anhalten* motiviert und angewiesen, an der Stopplinie der Kreuzung zu halten.
3. Beim Erreichen der Stopplinie wird das Fahrzeug in die zeitliche Kette der schon vorhandenen Fahrzeuge eingereiht. Zusätzlich wird das Verhalten *Stehen* so lange motiviert, bis das Eigenfahrzeug Vorrang erhält.
4. Zum Passieren der Kreuzung werden die Verhalten *Anhalten* und *Stehen* demotiviert, und das Verhalten *Spur folgen* übernimmt wieder die alleinige Kontrolle auf taktischer Schicht.

Taktische Schicht

Die fünf Verhalten *Spur folgen*, *Ausweichen*, *Überholen*, *Anhalten* und *Stehen* bilden zusammen die Realisierung der taktischen Schicht. Die Verhalten übernehmen im Einzelnen die folgenden Funktionen:

Das Verhalten *Spur folgen* hat zum Ziel, der von der Situationsinterpretation entsprechend dem Missionsplan vorgegebenen Spur zu folgen. Es motiviert die reaktiven Verhalten *Spur halten*, um auf der aktuell gefahrenen Spur zu bleiben oder *Spur wechseln*, um von einer Nachbarspur auf die Zielspur zu gelangen.

Das Verhalten *Ausweichen* dient dazu, statische Hindernisse zu umfahren. Dies sind üblicherweise parkende Autos, können aber auch generell Objekte anderer Art auf der Fahrspur sein. Wenn möglich, wird das Verhalten *Spur halten* derart instruiert, dass das Hindernis innerhalb der zur Verfügung stehenden Eigenspur umfahren werden kann. Sollte es notwendig sein, die Gegenspur zu benutzen, so kommen ähnlich wie beim Überholen mehrere Phasen des Ausweichens mit vorheriger Überprüfung der Verkehrslage zum Einsatz (vgl. Verhalten *Überholen*).

Statische und dynamische Hindernisse, welche die Eigenspur blockieren, können mit dem Verhalten *Überholen* passiert werden. Dabei werden zunächst die für einen Überholvorgang notwendigen Bedingungen wie Verkehrslage und infrastrukturelle Eigenschaften geprüft und gegebenenfalls ein Überholvorgang eingeleitet. Während der verschiedenen Phasen des Überholens werden nacheinander die reaktiven Verhalten *Spur wechseln*, *Spur halten* und wieder *Spur wechseln* motiviert, um zunächst auf die Gegenspur zu wechseln, das zu überholende Fahrzeug zu passieren und wieder auf die Eigenspur zu gelangen. Das zu Beginn eines Überholvorganges motivierte Verhalten *Abstand halten* muss zunächst deaktiviert werden, um während des Ausscherens auf das vorausfahrende Fahrzeug auffahren zu können.

Das Verhalten *Anhalten* dient dazu, an einer bestimmten Stelle einer Fahrspur zum Stehen zu kommen. Die Stoppposition wird dazu an die Bahnplanung weitergereicht und die Sollgeschwindigkeit entsprechend der verbleibenden Strecke reduziert. Das Verhalten stellt eine bewusst ausgeführte Fahrhandlung dar, die einen (geringen) Planungsanteil erfordert. Es ist deshalb in der taktischen Ebene angesiedelt und nicht, wie bei oberflächlicher Betrachtung nahe liegen könnte, in der reaktiven Schicht.

Im Gegensatz zum Verhalten *Anhalten*, das versucht an einer bestimmten Zielposition zum Stehen zu kommen, bewirkt *Stehen* die Umsetzung der bewusst ausgeführten Handlung des Stillstehens. Das Fahrzeug wird damit zum Stehen gebracht und in dieser Position gehalten. Dieses Verhalten kommt bspw. während des Wartens an einem 4-Way-Stop zum Einsatz.

Reaktive Schicht

Die in der reaktiven Schicht angesiedelten Handlungsprimitive übernehmen Regelungsaufgaben und Sicherheitsfunktionen. Entsprechend ihrem Einfluss auf die Ausgabedaten

lassen sich die Verhalten in zwei Gruppen mit *lateralem* und *longitudinalem* Charakter einordnen. Eine Ausnahme bildet das Verhalten *Kollision vermeiden*, welches Einfluss auf beide Gruppen besitzt und deshalb nicht eindeutig zugeordnet werden kann. Die Verhalten der lateralen Gruppe erzeugen Gefahrenkarten, die vom Fusionsknoten *Gefahrenkarte fusionieren* zu einem Gesamtergebnis kombiniert werden. Das Verfahren, mit dem die Karten fusioniert werden, ist in Abschnitt 3.3.1 erläutert.

Der Fusionsknoten *Geschwindigkeit fusionieren* erzeugt aus den Vorgaben von Verhalten der *longitudinalen* Gruppe eine einheitliche Ausgabe. Das Ergebnis stellt eine Richtgeschwindigkeit dar und wird über den Fusionsknoten *Bahnplanung instrumentieren* an die Bahnplanung weitergereicht. Letzterer übermittelt neben einer Richtgeschwindigkeit weitere Parameter wie bspw. den zeitlichen Planungshorizont, begrenzende Anhaltelinien oder verschiedene Planungsmodi.

Das Verhalten *Spur wechseln* erzeugt ein Gefahrenprofil, welches das Fahrzeug von der Eigenspur auf die rechte oder linke Nachbarspur leitet. Die Parametrierung der Gefahrenfunktionen legt dabei fest, wie schnell der Spurwechsel erfolgt (vgl. *Bahnplanung* in Abschnitt 3.3.1). Dieses Handlungsprimitiv ist gekoppelt mit dem Verhalten *Spur halten*, welches dann automatisch demotiviert wird, um einen wechselseitigen Ausschluss zu garantieren.

Eine Einschränkung des Konfigurationsraumes der Bahnplanung auf tolerierte Bereiche wird vom Verhalten *Auf Straße bleiben* durchgeführt. Dieses Sicherheitsverhalten schließt nicht befahrbare Bereiche vollständig von einer Planung aus und legt für befahrbare Bereiche eine hohe Gefahr fest. Die dadurch markierten Bereiche sind grundsätzlich nutzbar, finden jedoch üblicherweise nach einer Fusion mit Gefahrenkarten anderer Verhalten keine Verwendung.

Unabhängig von der von der Missionsplanung identifizierten Sollspur liefert das Verhalten *Spur halten* eine Gefahrenkarte für die aktuell verwendete Fahrspur. Als weitere Eingabe für das Verhalten dient das Verhältnis zwischen der in der Spur einzunehmenden lateralen Position und der Spurbreite.

Das Verhalten *Kollision vermeiden* markiert in einer Gefahrenkarte alle bekannten Hindernisse und Fremdfahrzeuge, indem es auf deren Abmessungen eine Sicherheitsfunktion anwendet. Diese enthält einen von der Objektklasse abhängigen Sicherheitsabstand. Es erfolgt zudem eine Prädiktion der Objektbewegungen entlang ihrer Fahrspuren. Im Gegensatz zu den anderen Verhalten erzeugt dieses Verhalten dazu mehrere über die Zeit veränderte Gefahrenkarten.

Das Sicherheitsverhalten *Abstand halten* reduziert die Geschwindigkeit des Eigenfahrzeuges abhängig vom Abstand und der Differenzgeschwindigkeit zum vorausfahrenden Fahrzeug. Die Motivation des Verhaltens hat direkten Einfluss auf den Sicherheitsabstand, wobei ein minimaler Wert jederzeit garantiert wird. *Abstand halten* ist immer aktiv und wird nur vom

taktischen Verhalten *Überholen* lediglich für den Zeitraum des Ausscherens deaktiviert.

Das Verhalten *Geschwindigkeit anpassen* reduziert die an die Bahnplanung übermittelte Richtgeschwindigkeit anhand festgelegter Kriterien. Eine erste Einschränkung erfolgt durch die zulässige Höchstgeschwindigkeit im Straßenabschnitt. Als zweites Kriterium dient die Begrenzung der Querbeschleunigung, welche die Geschwindigkeit in Abhängigkeit vom aktuellen Straßenverlauf einschränkt. Als weitere Kriterien könnten die Wetterlage oder die Straßenbeschaffenheit herangezogen werden.

3.3.4 Virtuelle Sensoren und Koppelung von Verhalten

Eine detaillierte Vorstellung der Berechnungsfunktionen für die Reflexion aller Verhalten würde den Rahmen dieses Kapitels sprengen. Stattdessen sollen in den folgenden beiden Beispielen exemplarisch die Berechnungsfunktionen für das reaktive Verhalten *Spur wechseln* und das taktische Verhalten *Überholen* erklärt werden. Weiterhin wird anhand des zweiten Beispiels erläutert, wie eine Koppelung von Verhalten zur Fortschrittsbestimmung verwendet werden kann. Abschnitt 3.3.5 listet alle Berechnungsfunktionen in einer Übersicht auf.

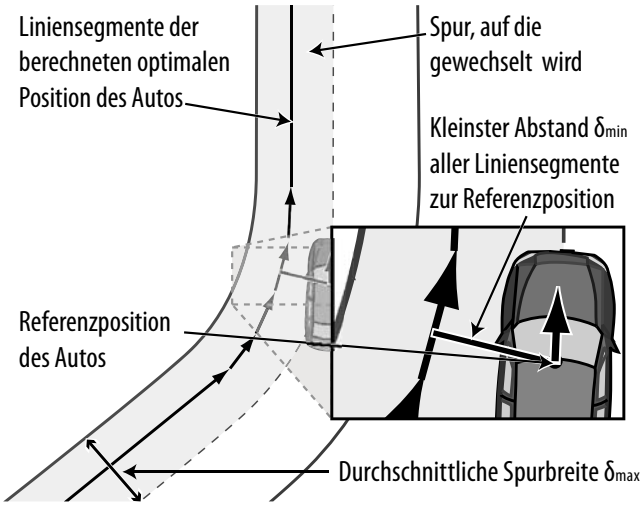


Abbildung 3.7: Relevante physikalische Größen für die Berechnung der Reflexion des Verhaltens *Spur halten*. Bildquelle: [Jagszent 08].

Beispiel *Spur wechseln*

Die Berechnung der Reflexion von Verhalten der reaktiven Ebene beruht üblicherweise auf einer Auswertung physikalischer Größen, um eine Übereinstimmung (in [Albiez 06] auch als *Zufriedenheit* bezeichnet) mit der aktuellen Situation auszudrücken. Analog zum menschlichen Verständnis von *Spur wechseln* ist das Verhalten zufrieden, wenn das Fahrzeug die Mitte der Spur erreicht hat, auf die es wechseln möchte. Befindet sich das Fahrzeug am Rand der Zielspur oder sogar auf einer Nachbarspur, so ist das Verhalten mit der

Situation unzufrieden. Die Reflexion wird anhand dieser beiden Randbedingungen auf das zur Verfügung stehende Intervall zwischen 0 und 1 skaliert.

Als Maß für die Abweichung wird ein Wert $\epsilon = \frac{\delta_{min}}{\delta_{max}}$ berechnet, in den die minimale Distanz δ_{min} des Fahrzeugreferenzpunktes zur optimalen Position eingeht sowie der maximal tolerierte Abstand δ_{max}. Letzterer beträgt üblicherweise die Hälfte der zur Verfügung stehenden Fahrspurbreite, kann aber auch kleiner gewählt werden. Als optimale Position gilt die Mitte der Spur, die anhand von Liniensegmenten der Kartendaten berechnet wird. Abbildung 3.7 veranschaulicht die Größen. Die Reflexion berechnet sich für *Spur wechseln* folgendermaßen:

$$r_{Spur\ wechseln} = \begin{cases} 0, & \text{wenn} \quad \epsilon \leqslant 0 \\ 1, & \text{wenn} \quad \epsilon \geqslant 1 \\ \epsilon, & \text{wenn} \quad 0 < \epsilon < 1 \end{cases}$$

Beispiel *Überholen*

Im Gegensatz zu den Verhalten der reaktiven Ebene stellt die Reflexion bei den Verhalten der taktischen Ebene keine physikalische Größe in Form von Abweichungen oder Regeldifferenzen dar. Die Verhalten der taktischen Ebene besitzen einen höheren Planungsanteil und fokussieren üblicherweise auf einen längeren Zeitraum. Eine Orientierung am zeitlichen Fortschritt, oder auch an einem internen Zustand, liefert deshalb bessere Ansatzpunkte für aussagekräftige Reflexionswerte.

Der vom Verhalten *Überholen* durchgeführte Vorgang lässt sich in mehrere Phasen einteilen, die über die Reflexion kommuniziert werden. Aus der folgenden Unterteilung ist für übergeordnete Verhalten ersichtlich, ob Bedarf zum Überholen besteht und in welcher Phase sich ein möglicher Vorgang befindet:

$$r_{Überholen} = \begin{cases} 0{,}9 & \text{wenn nicht überholt wird, aber Bedarf besteht} \\ 0{,}5 & \text{wenn der laufende Vorgang abgebrochen werden muss} \\ 0{,}2 & \text{wenn auf die Überholspur gewechselt wird} \\ 0{,}15 & \text{wenn überholt wird (passieren)} \\ 0{,}125 & \text{wenn auf die ursprüngliche Spur zurückgewechselt wird} \\ 0{,}0 & \text{wenn nicht überholt wird und kein Bedarf besteht} \end{cases}$$

Das steuernde strategische Verhalten hat zu überprüfen, ob die Reflexion von *Überholen* über der Zeit abnimmt, ob entsprechend die *Zufriedenheit* zunimmt und ein Fortschritt erzielt wird. Bleibt die Reflexion längere Zeit konstant, so hat das strategische Verhalten die Möglichkeit, den Vorgang aktiv abzubrechen. Dies könnte bspw. in einem Demotivieren von *Überholen* und gleichzeitigem Motivieren von *Spur halten* resultieren.

Das Verhalten *Überholen* bedient sich ebenfalls der Reflexionen reaktiver Verhalten, um einen Fortschritt innerhalb der einzelnen Phasen zu bestimmen und eine nächste Phase einzuleiten.

Dies gilt insbesondere für die Reflexion des Verhaltens *Spur wechseln*: Anhand dieses Wertes wird überprüft, ob ein Spurwechsel Fortschritt erzielt und wann er abgeschlossen ist. Die verschiedenen Phasen des Verhaltens *Überholen* werden im Folgenden näher erläutert:

1. *Kein Bedarf*
   Das Verhalten verweilt in dieser Phase solange kein Bedarf zum Überholen besteht oder ein Überholen aufgrund des Verkehrs oder der Straßenstruktur nicht möglich ist. Bedarf zum Überholen besteht dann, wenn ein vorausfahrendes Fahrzeug das Eigenfahrzeug nennenswert aufhält, bzw. über einen gewissen Zeitraum eine Verminderung der gewünschten Geschwindigkeit erzwingt.

2. *Bedarf*
   In dieser Phase wird dem strategischen Verhalten durch entsprechende Reflexion mitgeteilt, dass Überholbedarf besteht und ein Überholvorgang möglich ist. Erteilt das strategische Verhalten durch Setzen der Motivation die Freigabe, dann wird die Phase *Spurwechsel auf Gegenspur* eingeleitet. Andernfalls wird die Phase beibehalten bis der Überholvorgang nicht mehr möglich ist, dann wird nach *Abbruch* gewechselt.

3. *Spurwechsel auf Gegenspur*
   Diese Phase wechselt von der Ausgangs- auf die Gegenspur und überwacht den Fortschritt anhand der Reflexion von *Spur wechseln*. Fällt dessen Reflexion unter einen eingestellten Wert, so gilt die Überholspur als erreicht und es wird die nächste Phase *Überholvorgang* eingeleitet. Im Fehlerfall folgt ein Wechsel nach *Abbruch*.

4. *Überholvorgang (Passieren)*
   Während dieser Phase wird fortlaufend überprüft, ob das zu überholende Fahrzeug vollständig passiert wurde. Ist dies der Fall, wird in die Phase *Spurwechsel auf Ausgangsspur* gewechselt. Im Fehlerfall, hervorgerufen durch ein Beschleunigen des zu überholenden Fahrzeuges und einer (nicht durchführbaren) Verlängerung des Manövers, wird in die Phase *Abbruch* gewechselt.

5. *Spurwechsel auf Ausgangsspur*
   In dieser Phase wird, analog zu *Spurwechsel auf Gegenspur*, wieder zurück auf die Ausgangsspur gewechselt. Sobald der Vorgang abgeschlossen ist, wird die ursprüngliche Phase *Kein Bedarf* eingeleitet.

6. *Abbruch*
   Diese Phase kann jederzeit aktiviert werden und leitet eine Fehlerbehandlungsroutine ein. Diese besteht darin, zurück auf die ursprüngliche Spur zu gelangen und anschließend in die Phase *Kein Bedarf* zu wechseln.

3.3.5 Berechnungsmethoden in der Übersicht

In den vorangegangenen Abschnitten wurden exemplarisch die Berechnungsmethoden für die Verhalten *Spur wechseln* und *Überholen* dargelegt. Tabelle 3.2 liefert eine Übersicht über die Berechnungsvorschriften für Reflexion und Aktivität aller reaktiven und taktischen

Verhalten. Wie bereits angesprochen sind diese Signale für Verhalten der strategischen Ebene nicht relevant, da hier keine weitere Koppelung an andere Verhalten besteht. Es werden die in Tabelle 3.1 aufgelisteten Notationen und Abkürzungen verwendet:

$clamp(x)$	Beschränkt den Wert x auf das Intervall [0,1]
m	Motivation eines Verhaltens
δ_{min}	Minimaler Abstand zwischen der optimalen und der tatsächlichen Fahrzeugposition
δ_{max}	Durchschnittliche Spurbreite
δ_n	Distanz zum nächsten dynamischen Objekt
δ_s	Distanz zur Stopplinie
v_c	Aktuelle Geschwindigkeit des Eigenfahrzeuges
v_d	Angestrebte Geschwindigkeit des Verhaltens
s_s	Zum vorausfahrenden Fahrzeug einzuhaltender Sicherheitsabstand
s_b	Strecke, die notwendig ist, um das Eigenfahrzeug mit Nominalbeschleunigung von 1 $\frac{m}{s^2}$ auf die Geschwindigkeit des vorausfahrenden Fahrzeuges zu bringen
s_r	Strecke, die notwendig ist, um das Eigenfahrzeug mit einer Nominalbeschleunigung von 0,5 $\frac{m}{s^2}$ auf die Geschwindigkeit des vorausfahrenden Fahrzeuges zu bringen

Tabelle 3.1: Notationen und Abkürzungen für die Berechnungsmethoden der virtuellen Sensoren.

3.4 Software-System

Für die Umsetzung der in Abschnitt 3.3 entworfenen Verhaltenssteuerung wird ein geeignetes Software-System (Framework) benötigt. Die Schnittstellen des Frameworks sind teilweise durch die Software-Architektur im SFB (vgl. Anhang E) und vorhandenen Komponenten wie bspw. die Echtzeitdatenbank (RTDB) vorgegeben. Die Umsetzung kann individuell erfolgen und wird für dieses Framework zur Ausführung des Verhaltensnetzwerkes unter Verwendung der Qt4-Bibliothek durchgeführt.

Eine knappe Erläuterung der Konzepte und Entwurfsentscheidungen des Software-Systems soll an dieser Stelle genügen. Für eine detaillierte Beschreibung einzelner Komponenten sei auf [Jagszent 08] verwiesen.

3.4.1 Übersicht

Zunächst wird festgestellt, dass zur Umsetzung der Verhaltenssteuerung drei Schichten mit unterschiedlichen Ausführungsintervallen benötigt werden. Reaktive Verhalten der

Reaktive Verhalten		
Verhalten	**Reflexion**	**Aktivität**
Spur halten	$clamp\left(\frac{\delta_{min}}{\delta_{max}}\right)$	$clamp\left[\frac{1}{2} + clamp\left(\frac{\delta_{min}}{\delta_{max}} \cdot \frac{1}{2}\right)\right] \cdot m$
Spur wechseln	$clamp\left(\frac{\delta_{min}}{\delta_{max}}\right)$	$clamp\left[\frac{1}{2} + clamp\left(\frac{\delta_{min}}{\delta_{max}} \cdot \frac{1}{2}\right)\right] \cdot m$
Kollision vermeiden	$1 - clamp\left(\frac{\delta_n}{10}\right)$	$clamp\left[1 - clamp\left(\frac{\delta_n}{10}\right)\right] \cdot m$
Auf Straße bleiben	$clamp\left(\frac{\delta_{min}}{\delta_{max}}\right)$	$clamp\left[\frac{1}{2} + clamp\left(\frac{\delta_{min}}{\delta_{max}} \cdot \frac{1}{2}\right)\right] \cdot m$
Abstand halten	$1 - clamp\left(\frac{\sqrt{\delta_n - s_s - s_b}}{\sqrt{s_r - s_b}}\right)$	$\left[1 - clamp\left(\frac{\sqrt{\delta_n - s_s - s_b}}{\sqrt{s_r - s_b}}\right)\right] \cdot m$
Geschwindigkeit der Umgebung anpassen	$clamp\left(\frac{v_c - v_d + \frac{1}{4}}{5\frac{1}{4}}\right)$	$clamp\left(\frac{1}{2} + \left[clamp\left(\frac{v_c - v_d + \frac{1}{4}}{5\frac{1}{4}}\right)\right] \cdot \frac{1}{2}\right) \cdot m$
Geschwindigkeit fusionieren	$clamp\left(\frac{\lvert v_c - v_d \rvert}{10}\right)$	$1 \cdot m$

Taktische Verhalten		
Verhalten	**Reflexion**	**Aktivität**
Spur folgen	$clamp\left(\frac{\delta_{min}}{\delta_{max} \cdot \frac{6}{5}}\right)$	$clamp\left[\frac{1}{2} + clamp\left(\frac{\delta_{min}}{\delta_{max} \cdot \frac{6}{5}} \cdot \frac{1}{2}\right)\right] \cdot m$
Überholen	siehe Abschnitt "*Beispiel Überholen*"	
Stehen	$clamp\left(\frac{v_c}{2}\right)$	$1 \cdot m$
Anhalten	$clamp\left(\delta_s\right)$	$clamp\left(\delta_s\right) \cdot m$

Tabelle 3.2: Berechnungsmethoden für Reflexion und Aktivität der Einzelverhalten.

untersten Ebene müssen öfter ausgeführt werden als die Verhalten der oberen Schichten, welche zunehmend überwachenden und planenden Charakter besitzen. Weiterhin erfolgt keine zentrale zeitliche Synchronisierung von Daten über die RTDB mit der Folge, dass auf bestimmte Datenobjekte blockierend gewartet werden muss. Die Umsetzung dieser verschiedenen Aufgaben verlangt nach mehreren Ausführungsfäden, die in Abbildung 3.8 in einer Übersicht zusammengestellt sind.

Die Hauptbestandteile des Software-Systems zur Verhaltenssteuerung werden im Folgenden näher erläutert:

Manager
Der Manager-Ausführungsfaden koordiniert den gesamten Ablauf der Verhaltenssteuerung. Er verwaltet sämtliche Ressourcen, ist für ihre Initialisierung zuständig und überwacht die anderen Komponenten. Die grundlegende Konfiguration wird zentral über den Manager vorgenommen. So ist es möglich, durch alleinigen Austausch des Managers die Funktionsweise bspw. für einen Testlauf anzupassen (vgl. Abschnitt 3.4.3).

Signalemitter
Die Signalemitter-Ausführungsfäden warten auf Ereignisse wie bspw. die Aktualisierungen von Datenbankobjekten. Sie dienen zudem als Zeitgeber für die Umsetzung von Ausführungsintervallen. Die Verwendung eigener Ausführungsfäden verhindert ein rechenintensives *aktives Warten* auf die jeweilige Bedingung. Eintretende Ereignisse werden an den Signalprozessor weitergereicht.

Signalprozessor
Der Signalprozessor verarbeitet die von den Signalemittern eintreffenden Sensordaten und aktiviert die Ausführung einzelner Schichten des Verhaltensnetzwerkes. Im Signalprozessor hinterlegte Regeln stellen eine Einhaltung der Intervalle und zulässiger Toleranzen sicher und berücksichtigen darüber hinaus Abhängigkeiten der einzelnen Schichten voneinander.

Ausführungsschichten des Verhaltensnetzwerkes
Die Schichten des Verhaltensnetzwerkes stellen eigene Ausführungsfäden mit unterschiedlichen Intervallen dar. Sie dienen als Container für die in dieser Schicht angesiedelten Verhalten. Bei der Ausführung einer Schicht werden zunächst alle Daten aus der RTDB geladen, die von Verhalten dieser Ebene benötigt werden. Nach der Berechnung der virtuellen Sensorwerte aller Verhalten werden diese ausgeführt und die veränderten Datenbankobjekte wieder in der RTDB abgelegt. Verhaltensschichten verwalten die in ihnen enthaltenen Verhalten. Dies beinhaltet bspw. auch die Zuordnung von Referenzen auf benötigte Datenabstraktionen.

Im nächsten Abschnitt soll anhand eines Ablaufbeispiels das Zusammenwirken einzelner Komponenten verdeutlicht werden.

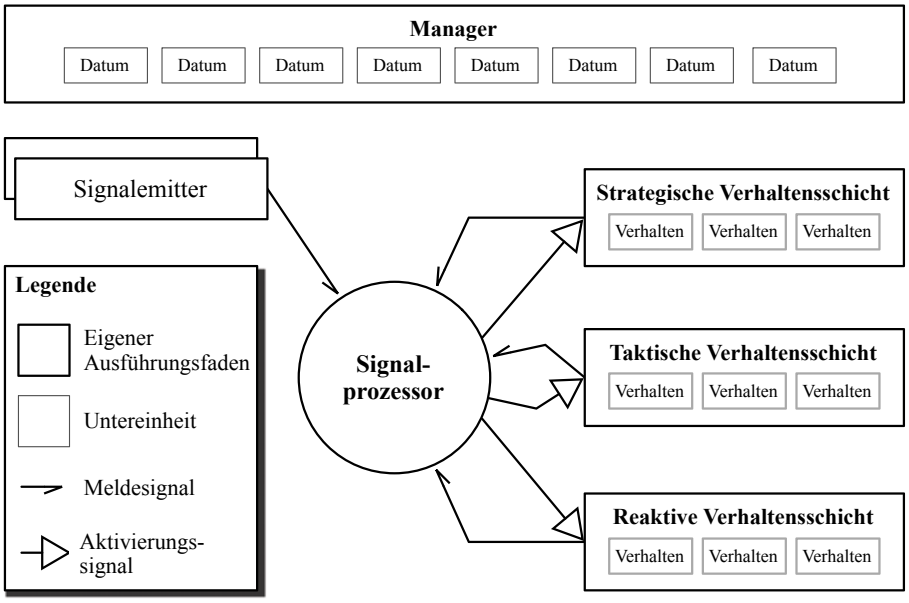


Abbildung 3.8: Architektur des Software-Systems, seine Hauptbestandteile und die Beziehungen zwischen einzelnen Ausführungsfäden.

3.4.2 Koordination von Ausführungsfäden

Bevor die Ausführungsfäden gestartet werden, nimmt der Manager zunächst eine Initialisierung derselben vor. Dies geschieht entsprechend den Abhängigkeiten zwischen den einzelnen Komponenten: Es werden zuerst die Datenabstraktionen initialisiert, dann folgen die Verhaltensschichten, der Signalprozessor und zuletzt die Signalemitter. Nach der Initialisierungsphase werden die Ausführungsfäden in derselben Reihenfolge gestartet und kommunizieren mit asynchronen Nachrichten untereinander. Abbildung 3.9 zeigt das Beispiel einer Nachrichtenfolge zur Koordination der beteiligten Komponenten.

Die verschiedenen Signalemitter melden das Eintreffen bestimmter Ereignisse an den Signalprozessor, danach wird umgehend wieder auf die zugewiesene Bedingung gewartet. Bei den zum Signalprozessor gesandten Nachrichten handelt es sich um Meldungen, die bspw. besagen, dass Datenbankobjekte aktualisiert oder Zeitgeber abgelaufen sind. Der Signalprozessor sammelt und filtert die eintreffenden Signale entsprechend den hinterlegten Regeln. Die reaktive Ausführungsschicht wird z. B. aktiviert, falls seit der letzten Ausführung bereits mehr als 0,1 Sekunden verstrichen sind oder falls neue Daten seitens der Interpretation oder Wahrnehmung vorliegen.

Die Ausführungsschichten der Verhalten melden dem Signalprozessor das Ende eines Berechnungszyklus und stehen dann wieder für eine Aktivierung bereit. Die Rückmeldungen werden vom Signalprozessor gesammelt, der wiederum dem Manager einen vollständig durchlaufenen Zyklus aller Schichten signalisiert.

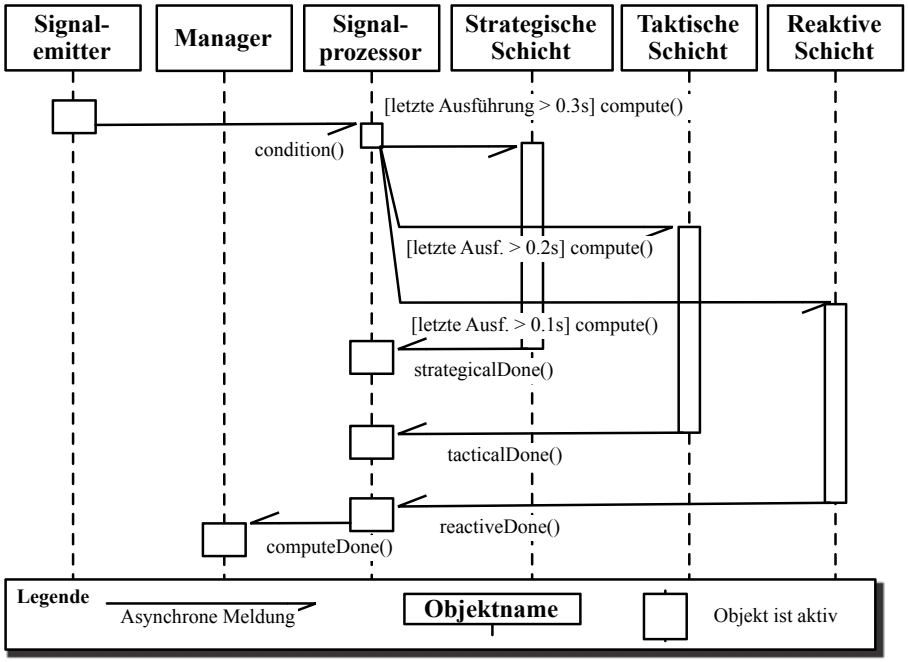


Abbildung 3.9: Beispiel zur Koordination von Ausführungsfäden mithilfe asynchron gesendeter Nachrichten. Die Zeitachse ist in vertikaler Richtung aufgespannt.

3.4.3 Testumgebung

Die Testumgebung dient dazu, die Funktionsfähigkeit einzelner Verhalten und deren Ausgaben zu testen. Damit wird sichergestellt, dass die Verhalten sich gemäß ihrer Spezifikation verhalten und im Verbund mit anderen Verhalten eingesetzt werden können. Für einen Test des Gesamtsystems mit Wahrnehmungs- und Simulationskomponenten steht die im SFB/TR28 entwickelte Simulationsumgebung zur Verfügung (vgl. Abschnitt E.2).

Wie in Abschnitt 3.4.1 beschrieben koordiniert der Manager-Ausführungsfaden den gesamten Ablauf der Verhaltenssteuerung. Ein Test-Manager mit erweiterter Funktionalität ist in der Lage, gezielte Testfälle auszuführen, die die Verhalten mit synthetischen oder zuvor abgespeicherten Daten versorgen (vgl. Abbildung 3.10). Die Daten eines Testfalls liegen in Form einer sog. Fixtur vor. Dies ist ein zu einem bestimmten Zeitpunkt in einer Datei hinterlegter Datenbankinhalt. Hierbei kann es sich um eine Aufnahme früherer Daten oder künstlich generierter Daten handeln. Die RTDB stellt entsprechende Werkzeuge für das Aufzeichnen und Einspielen dieser Datenbankinhalte zur Verfügung.

Nach dem Laden einer Fixtur führt der Testfall die zu testenden Verhalten aus. Der Benutzer hat im Test-Manager erweiterte Möglichkeiten, um die Ausführung von Verhalten zu unterbrechen und schrittweise durchzuführen. Die Ausgaben der Verhalten lassen sich so einfach mit den erwarteten Ausgaben vergleichen.

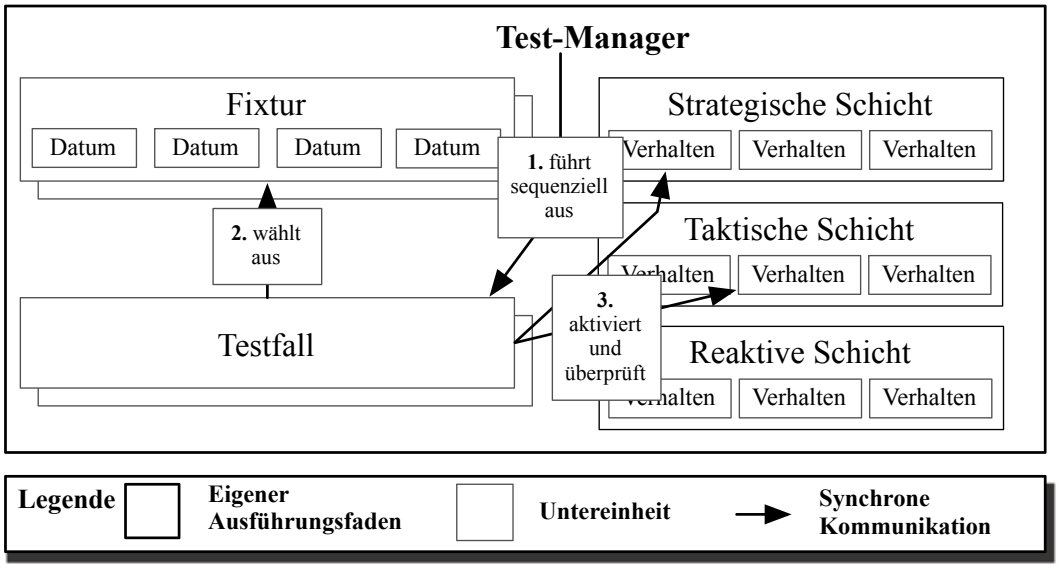


Abbildung 3.10: Test-Manager für das gezielte Testen einzelner Verhaltensaspekte.

Mit dieser speziellen Testumgebung wird die an die Verhaltenssteuerung gestellte Anforderung 7 nach *Testbarkeit* erfüllt. Die Simulationsunterstützung ist bereits durch die im SFB/TR28 existierenden Werkzeuge erfüllt.

3.5 Akkumulation von Erfahrungswissen

Aufgrund der hohen Situationsvielfalt im Straßenverkehr ist langfristig ein Hinzulernen, und damit eine Verbesserung der Verhaltensausführung, wünschenswert. Wenngleich Lernverfahren nicht Gegenstand dieser Arbeit sind, so war die Möglichkeit, Erfahrung zu sammeln, mit ein Grund für die Entscheidung zugunsten von Verhaltensnetzwerken. Die durch interne Verarbeitungsschritte in jedem Verhalten anfallenden virtuellen Sensordaten stellen Erfahrungswerte dar, welche die Grundlage für eine Beurteilung der Verhaltenssteuerung bilden.

Diese Beurteilung kann durch menschliche Entwickler, Fahrlehrer oder automatisierte Verfahren erfolgen, um das System durch nachgeschaltete Lernverfahren zu verbessern. Auch wenn der Begriff des *Lernens* an dieser Stelle mit Vorsicht zu genießen ist, so ist zumindest eine Realisierung in gewissen Grenzen denkbar. Für den Einsatz in einer Umgebung mit hohen Sicherheitsanforderungen wie dem öffentlichen Straßenverkehr ist das Lernen von Strukturen, wie bspw. vollständigen Fahrmanövern, nicht erwünscht. Eine Adaption von Ausführungsparametern hingegen (innerhalb verifizierter Bereiche) könnte ein Ansatz für einfache Lernverfahren zum Zwecke einer Optimierung sein. Die Optimierung dieser Fahrmanöver sollte Offline stattfinden, um eine Auswertung umfangreicher Datensätze zu ermöglichen, oder auch, um Vergleiche mit einer Fallbasis anstellen zu können. Neben einer Verwendung für

Lernverfahren bietet die Hinterlegung von Erfahrungswerten aber auch für Entwickler interessante Ansatzpunkte für Nachbesserungen im System.

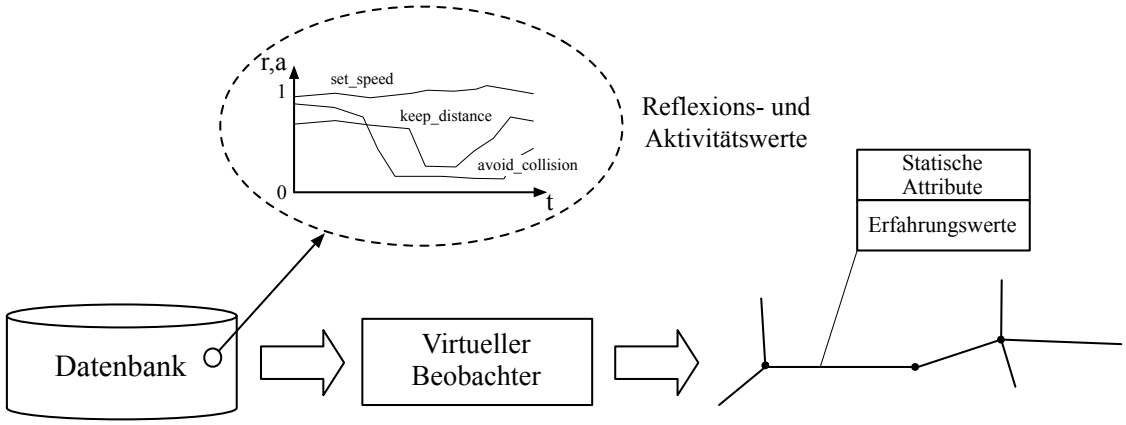


Abbildung 3.11: Ablegen von Erfahrungswerten: Die in der Datenbank hinterlegten Reflexions- und Aktivitätswerte der Verhalten werden durch einen virtuellen Beobachter gefiltert. Hinsichtlich bestimmter Kriterien wie Extrem- oder Durchschnittswerte beurteilt dieser die eingehenden Daten und speichert Auffälligkeiten als Erfahrungswerte in Form von Kantenattributen des Graphen.

Die Reflexionen der Einzelverhalten weisen durch Extremwerte der Verhalten *Spur halten* oder *Kollision vermeiden* auf besondere oder gefährliche Situationen hin. Die Aktivitäten hingegen lassen sich für eine Bewertung hinsichtlich Effizienz oder Komfort heranziehen (bspw. *Geschwindigkeit anpassen*). Abbildung 3.11 zeigt einen Vorschlag, diese virtuellen Sensorwerte mithilfe von sog. *Beobachtern* zu filtern und nur prägnante Werte ortsbezogen in einer semantischen Karte abzulegen. Dies kann in Form von Kantenattributen des Graphen der semantischen Karte zusätzlich zu bereits bestehenden statischen Informationen geschehen. Durch das Abfahren einer Strecke würden so Attribute der Kategorie *Erfahrungswerte* im Graphen abgelegt und könnten von Entwicklern zur Evaluation oder für (automatisierte) Lernverfahren verwendet werden. Die ortsbezogene Darstellung besitzt darüber hinaus den großen Vorteil, dass Erfahrungswissen mit anderen Fahrzeugen ausgetauscht und kombiniert werden kann. Etwaige Problemfälle wie bspw. eine schlecht einzusehende Kreuzung (mit hoher Chance für eine Aktivität von *Kollision vermeiden*) ließen sich durch eine statistische Häufung markanter Werte so schneller identifizieren.

Kapitel 4

Bahnplanung

4.1 Einführung

Das Bahnplanungsmodul hat die Vorgaben der Verhaltenssteuerung unter Berücksichtigung aktueller Wahrnehmungsdaten umzusetzen. Als Schnittstelle dienen sogenannte Gefahrenkarten, über welche dem Bahnplaner einerseits Informationen über Hindernisse übermittelt werden, in der andererseits aber auch Fahrintentionen abgelegt sind. Abschnitt 4.2 gibt zunächst eine Einführung in *Dynamische Gefahrenkarten* und beschreibt deren Aufbau. Abschnitt 4.3 geht näher auf ein im Rahmen dieser Arbeit entwickeltes gitterbasiertes Schätzverfahren zur Vorhersage von Objektbewegungen ein. Das Bahnplanungsmodul selbst mit Anforderungen, Suchalgorithmus und Schätzfunktion wird in Abschnitt 4.4 behandelt. Eine Erweiterung um eine Geschwindigkeitskomponente hin zu wirklicher Bewegungsplanung wird in Abschnitt 4.5 diskutiert, bevor im letzten Abschnitt 4.6 die Anpassungen für die in der Urban Challenge verwendete Bahnplanungsvariante erklärt wird.

4.2 Dynamische Gefahrenkarten

Eine Herausforderung bei der Informationsweitergabe zwischen der Entscheidungskomponente auf hoher Ebene, bis hin zur Regelungsebene auf niederer Ebene besteht darin, die Informationsfülle sukzessive zu reduzieren, um letztendlich mit Stellwerten für die Aktorik eine klare Fahrvorgabe zu erhalten. Dabei besteht die Gefahr, dass die Daten zu schnell reduziert werden, also relevantes Wissen für untere Ebenen verloren geht, oder dass eine Übermittelung von nicht notwendigem Wissen die darunter liegenden Komponenten unnötig komplex werden lässt.

Dynamische Gefahrenkarten sollen helfen, das Bahnplanungsmodul exakt mit den Informationen zu versorgen, die für die Erzeugung einer Bahn notwendig sind [Schröder 08]. Dabei soll soweit abstrahiert werden, dass auf dieser Stufe keine Objektinformationen mehr enthalten sind. Die grundlegende Idee besteht darin, in einem zweidimensionalen Gitter für jede Zelle Z einen Gefahrenwert anzugeben, welcher repräsentativ Gefahren der folgenden Form ausdrückt:

- **Hindernisse** stellen immer eine Gefahr dar und werden mit einem Gefahrenwert $\lambda = 1$ versehen. Eine Klassifikation kann dazu beitragen, ein hinterlegtes Gefahrenmodell anzuwenden. Informationen über Hindernisse werden von der Perzeption und von der Interpretation bezogen.
- **Fahrintentionen** werden von der Verhaltensentscheidung markiert und repräsentieren das zu fahrende Manöver im Groben. Hierbei werden gefährliche Bereiche (Abhang, Böschung) oder auch unerwünschte Bereiche (Gegenspur, Seitenstreifen) mit hohen Gefahrenwerten versehen und dadurch kaum oder gar nicht befahren. Je nach Einschätzung wird hierfür ein Gefahrenwert im Intervall $\lambda = [0, 1]$ gewählt.

Um eine verlässliche Bahnplanung mit Zeithorizont T durchführen zu können, müssen Informationen über Gefahrenwerte für den Zeitraum $0 \leqslant t \leqslant T$ vorhanden sein. Die Gefahrenkarte wird dazu um die Zeit als vierte Dimension erweitert. Dynamische Gefahrenkarten werden aus der Kombination einer (dynamischen) Hinderniskarte G_O und einer (statischen) Karte G_I, welche die Fahrintentionen enthält, gebildet. Sie beschreiben den Gefahrenwert im Bereich 0..255 an einer Stelle x, y zum Zeitpunkt t und werden wie folgt definiert:

$$G(x, y, t) = max(G_O(y, x, t), G_I(x, y)) \tag{4.1}$$

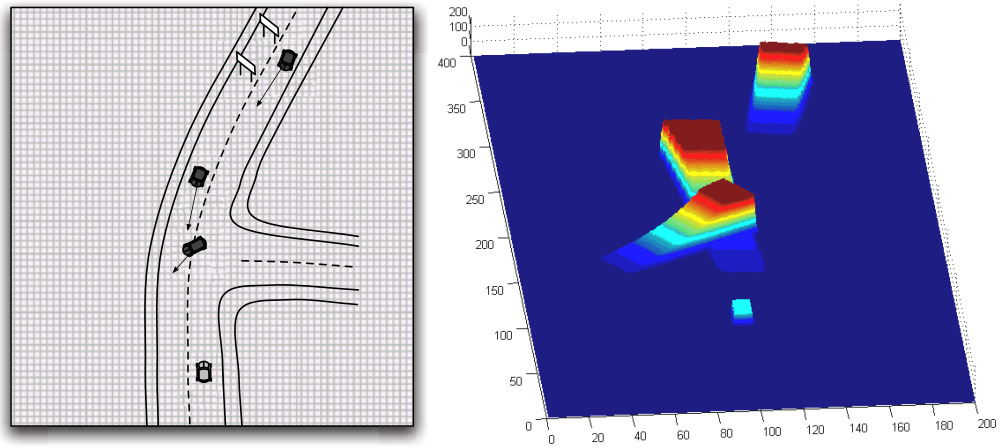
Die Karte ist für einen Bereich mit Ausdehnung m, n um die Position des Eigenfahrzeugs x_e, y_e definiert, so dass weiterhin gilt $x_e - m < x < x_e + m$ und $y_e - n < y < y_e + n$. Anschaulich kann die Gefahrenkarte als 8-Bit-Graustufen-Rastergrafik aufgefasst werden, bei der 0 eine geringe Gefahr darstellt und 255 eine sehr hohe Gefahr. Diese Darstellung besitzt den Vorteil, dass eine technische Verarbeitung mit Werkzeugen und Methoden aus der Bildverarbeitung möglich wird.

4.2.1 Hindernisrepräsentation

Abbildung 4.1 (links) zeigt schematisch dargestellt ein Verkehrsszenario mit drei sich bewegenden Kraftfahrzeugen. Ziel ist es, die Objekte der Szene und deren Bewegungen in Form von Gefahrenwerten in einer Karte G_O auszudrücken. Für jedes Objekt k liefert

$$\vec{x}_k(t) = (x_k(t), y_k(t), \theta_k(t), v)^T \tag{4.2}$$

den zu einem Zeitpunkt t gültigen Zustand in der Ebene. Die Objektgeschwindigkeit wird als konstant angenommen. Die Lage ergibt sich entweder aus der linear angenommenen Bewegung (für Hindernisse abseits der Spuren) oder aus dem Spurverlauf (für Hindernisse auf Spuren). Mit dieser Information kann, entsprechend der Klassifizierung, ein 3D-Gefahrenmodell angewandt werden, um die zu einen Zeitpunkt t von einem Objekt ausgehende Gefahr darzustellen. Zur Darstellung dieses Modells bieten sich in der 2D-Ebene stückweise definierte Funktionen an, welche über Längs- und Querachse des Objektes definiert und jeweils über der anderen Achse aufgetragen werden. Die Funktionen unterscheiden sich je nach Klasse des Objektes bspw. durch die Wahl des Sicherheitsabstandes oder durch die Übergangsfunktion.



(a) (b)

Abbildung 4.1: Schematische Darstellung einer Verkehrsszene mit Eigenfahrzeug (hell) und drei dynamischen Objekten (a). Durch Anwendung eines Gefahrenmodells wird die von den Hindernissen ausgehende Gefahr als Potentialfeld beschrieben (b). Eine Projektion des Gefahrenmodells entlang des Bewegungsvektors soll den zeitlichen Verlauf der Gefahrenwerte andeuten.

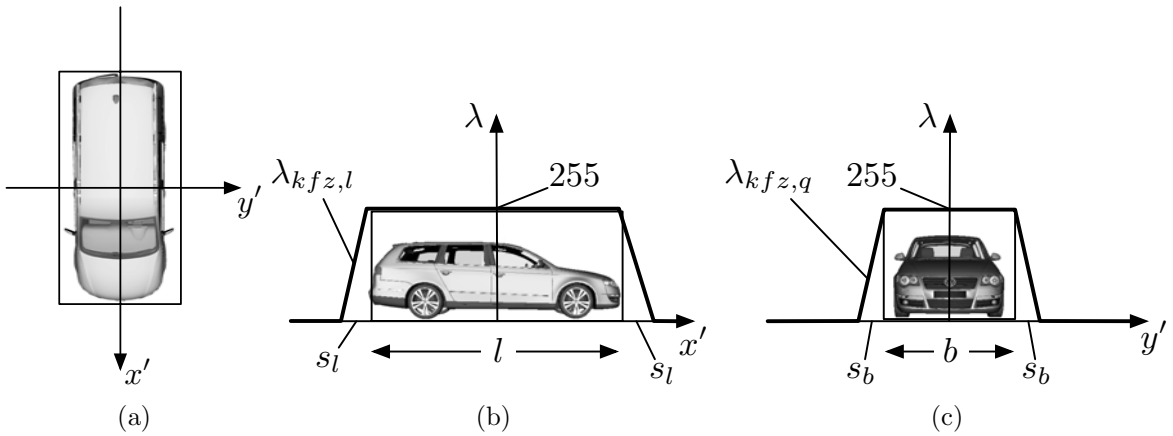
Für ein Objekt vom Typ Kraftfahrzeug seien folgende Funktionen zur Beschreibung eines Gefahrenmodells über Längs- ($\lambda_{kfz,l}$) und Querachse ($\lambda_{kfz,q}$) gewählt:

$$\lambda_{kfz,l}(x') = \begin{cases} 255 & , -\frac{l}{2} \leqslant x' \leqslant \frac{l}{2} \\ 0 & , x' > \frac{l}{2} + s_l, x' < -\frac{l}{2} - s_l \\ \frac{255}{s_l}x' + \frac{l}{2s_l} + 1 & , -\frac{l}{2} - s_l \leqslant x' < -\frac{l}{2} \\ -\frac{255}{s_l}x' + \frac{l}{2s_l} + 1 & , \frac{l}{2} < x' \leqslant \frac{l}{2} + s_l \end{cases} \tag{4.3}$$

$$\lambda_{kfz,q}(y') = \begin{cases} 255 & , -\frac{b}{2} \leqslant y' \leqslant \frac{b}{2} \\ 0 & , y' > \frac{b}{2} + s_b, y' < -\frac{b}{2} - s_b \\ \frac{255}{s_b}y' + \frac{b}{2s_b} + 1 & , -\frac{b}{2} - s_b \leqslant y' < -\frac{b}{2} \\ -\frac{255}{s_b}y' + \frac{b}{2s_b} + 1 & , \frac{b}{2} < y' \leqslant \frac{b}{2} + s_b \end{cases} \tag{4.4}$$

Als Parameter dienen die Höhe des Fahrzeuges h, die Breite b und ein Sicherheitsabstand s_l=2 m nach vorn und nach hinten, sowie s_b=1 m zur Seite. Abbildung 4.2 veranschaulicht die für die lokalen Koordinatenachsen definierten Gefahrenfunktionen. Aus diesen beiden Funktionen resultiert für jedes Objekt k ein lokales Gefahrenmodell $\lambda_{kfz,k}(x', y')$, das durch Transformation auf dessen Zustandsverlauf $\vec{x}_k(t)$ in die globale Darstellung $G_{O,k}(x, y, t)$ überführt werden kann. Die im Bereich $(x_e \pm m, y_e \pm n)$ enthaltenen Objekte liefern einen Beitrag zur gesamten Hindernisrepräsentation aller K Objekte:

$$G_O = max(G_{O,k}), k = 1..K \tag{4.5}$$



(a) (b) (c)

Abbildung 4.2: Lokales Koordinatensystem des Hindernisses (a) und über Längs- und Querachse (b,c) definierte Gefahrenfunktionen $\lambda_{kfz,l}$ und $\lambda_{kfz,q}$. Das 3D-Modell entsteht durch ein Auftragen der Funktionen über der jeweils anderen Achse und Minimums-Fusion der beiden Ergebnisse.

Die dynamische Hinderniskarte G_O wird in der Verhaltenssteuerung vom Verhalten *Kollision vermeiden* erzeugt.

4.2.2 Fahrintentionen

Die von der Verhaltenssteuerung geäußerten Fahrwünsche werden in Form einer Gefahrenkarte G_I übertragen, die diese Fahrintentionen enthält. Die in G_I als gefährlich markierten Bereiche werden je nach Grad von der Bahnplanung ausgeschlossen oder nach Möglichkeit vermieden. Um die Fahrintention *Spur halten* auszudrücken wird bspw. ein geringes Potential auf die gewünschte Fahrspur gelegt, andere Bereiche sind nicht zulässig. Soll ein Spurwechsel ausgeführt werden, dann sind die aktuelle Spur als auch die Zielspur befahrbar – letztere weist jedoch einen geringeren Gefahrenwert auf. Ähnlich wie bei den im vorigen Abschnitt beschriebenen Hindernissen werden Gefahrenmodelle für Fahrintentionen mit einer Potentialfunktion und einer Basisfunktion beschrieben, über welcher erstere aufgetragen wird.

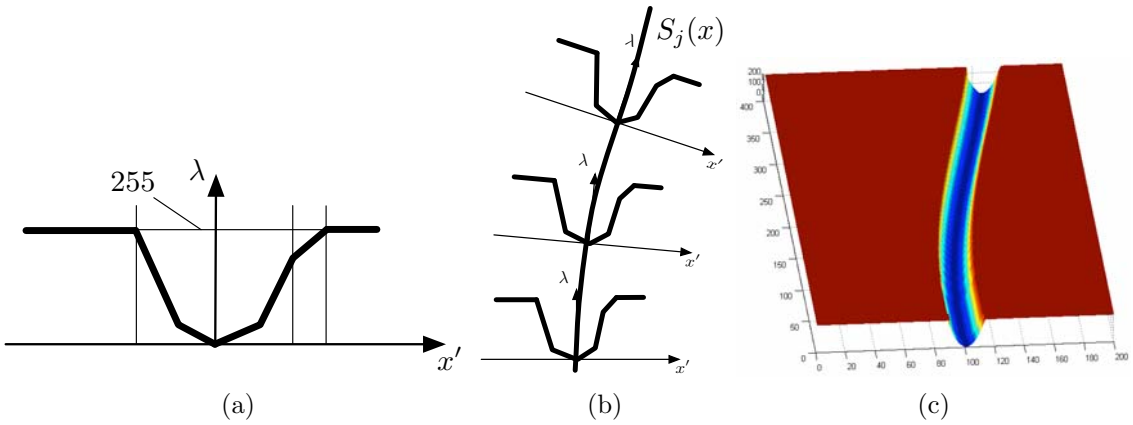
Die Formulierung einer Potentialfunktion $\lambda(x')$ erlaubt es, neben dem bevorzugten Bereich auch Randbereiche wie z. B. Grünstreifen zu modellieren. Dadurch wird eine Abstufung zwischen Bereichen möglich, die im Normalfall nicht verwendet werden sollten, in kritischen Situationen jedoch verwendet werden dürfen. Die Basisfunktion S beschreibt die zu fahrende Bahn, über welche die Potentialfunktion aufzutragen ist. Als Gerüst für die Basisfunktion dient eine kubische Spline-Funktion:

$$S_j(x) = a_j(x - x_j)^3 + b_j(x - x_j)^2 + c_j(x - x_j) + d_j, j = 0, 1, ..., n - 1 \tag{4.6}$$

Abbildung 4.3 veranschaulicht Potential- und Basisfunktionen für die Fahrintention *Spur halten*. Durch die globale Darstellung der Basisfunktion resultiert direkt das Gefahrenmodell $G_{I,i}$ für die jeweilige Fahrintention. Im Unterschied zur Hindernisrepräsentation entsteht das

Gesamtergebnis nicht durch Maximums-, sondern durch Minimumsfusion der Einzelvorgaben:

$$G_I = min(G_{O,i}), i = 1..I \tag{4.7}$$



(a) (b) (c)

Abbildung 4.3: Gefahrenfunktion $\lambda(x')$ zur Beschreibung der Fahrintention *Spur halten* (a). Aufgetragen über einer Basisfunktion $Sj(x)$ (b) ergibt sich das Gefahrenmodell G_I für eine Fahrintention (c).

4.3 Bewegungsschätzung nicht klassifizierter Objekte

Werden Objekte von den Wahrnehmungskomponenten korrekt erkannt, so können diese leicht auf symbolischer Ebene prädiziert werden, wie in Abschnitt 4.2.1 beschrieben. In vielen Fällen ist eine Klassifikation jedoch schwer möglich und damit auch eine Objektverfolgung ausgeschlossen. Die in Form von Rohdaten eines Laserscanners oder anderer Sensorsysteme vorliegenden Hindernisinformationen müssen in der Bahnplanungskomponente aber in jedem Fall berücksichtigt werden. Neben der Schnittstelle zur Verhaltenssteuerung, welche Fahrintentionen und Hindernisse der symbolischen Ebene in einer Gefahrenkarte repräsentiert, muss demnach eine zweite Schnittstelle zur Wahrnehmung existieren (vgl. Abbildung 4.4). Durch Integration der Daten aus diesen zwei Quellen wird eine Unabhängigkeit von der alleinigen symbolischen Repräsentation erreicht und die Sicherheit maßgeblich erhöht. Auch bei einem Ausfall der logischen Schicht könnte das Bahnplanungsmodul weiterhin mit Sensordaten versorgt werden. Problematisch ist die Repräsentationsform, mit welcher die Wahrnehmung Daten liefert. Üblicherweise erfolgt dies in Form von sog. Belegtheitskarten. Diese stellen zwar statistisch aufbereitete Umfeldinformationen bereit, lassen aber dynamische Informationen vermissen. Abbildung 4.5 verdeutlicht das Problem anhand einer Bahnplanungsaufgabe auf Basis einer rein statischen Gefahrenkarte.

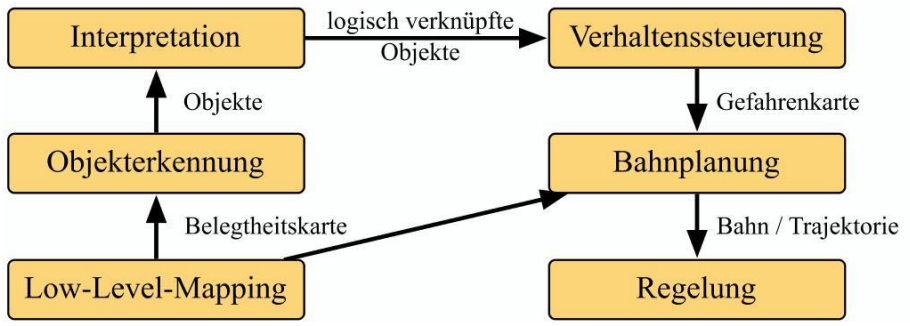


Abbildung 4.4: Umfeldinformationen für die Bahnplanung fallen auf verschiedenen Ebenen der Systemarchitektur an: Die Verhaltenssteuerung liefert auf symbolischer Ebene Objekte und Fahrintentionen in Form von Gefahrenkarten. Nicht klassifizierte Daten auf Zellebene werden direkt von der Wahrnehmung bezogen.

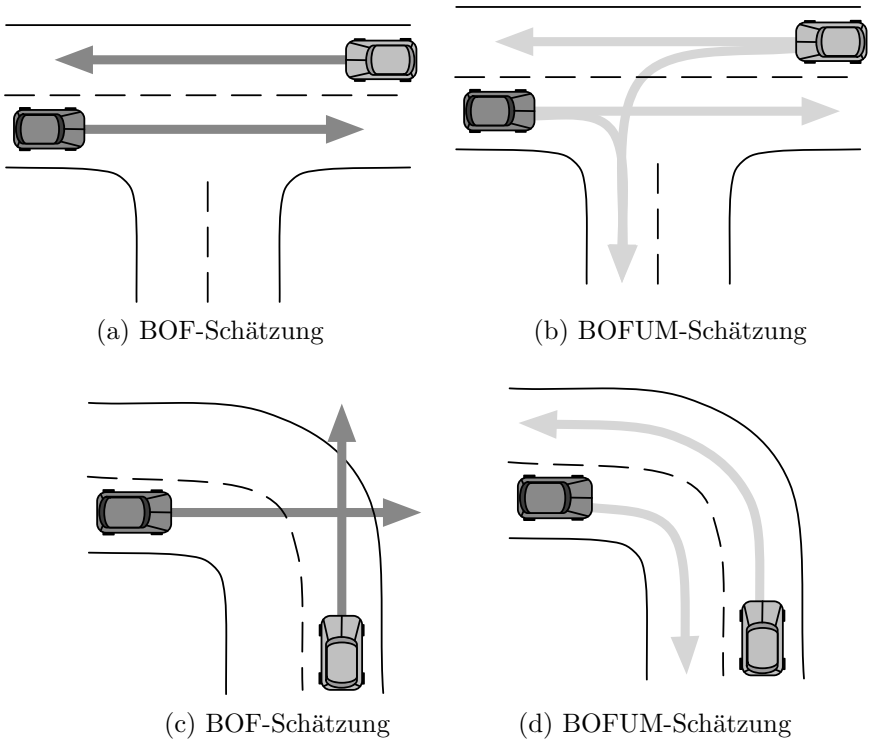
In diesem Abschnitt wird ein Ansatz auf Basis eines *Bayesian Occupancy Filter (BOF)* vorgestellt, um das Problem der Bewegung von Gitterzellen, die keinem Objekt zugeordnet sind, zu lösen. Die Berücksichtigung von symbolischen Informationen findet in diesem Ansatz nicht statt. Er ist vielmehr als eine Ergänzung zur Objektrepräsentation in Abschnitt 4.2.1 zu sehen. Im Gegensatz zu der in [Chen 06] vorgestellten BOF-Implementierung wird die Zellbewegung nicht als linear angenommen, sondern mithilfe von Karteninformationen angepasst[1]. Die Anpassung des Bewegungsmodells basiert auf der Annahme, dass bestimmte Fahrmanöver im Verkehr wahrscheinlicher als andere und an die geometrische Straßenstruktur gebunden sind. So wird ein Fahrzeug bspw. eher auf seiner Spur bleiben als auf

[1]Daraus resultiert die Bezeichnung BOFUM (Bayesian Occupancy Filter using Map Knowledge).

(a) (b) (c)

Abbildung 4.5: Herkömmliche Repräsentation der Belegtheit von Zellen ohne Geschwindigkeitsinformationen. Nicht klassifizierte Sensordaten werden auf Zellebene nicht prädiziert. Dem Bahnplanungsmodul steht folglich nur eine statische Gefahrenkarte zur Verfügung (belegte Zellen sind rot markiert). Im Beispiel kann die Bewegung der Zellen in der Bahnplanung nicht berücksichtigt werden und wird erst spät bei einer Gültigkeitsüberprüfung des geplanten Pfades erkannt.

die Gegenspur wechseln und sicher nicht auf den Gehweg fahren. Mit der Identifikation von *Erreichbarkeitwahrscheinlichkeiten* von Zellen lässt sich eine reale Prädiktion durchführen, was besonders im Falle von Verdeckungen über einen längeren Zeitraum essentiell ist. Der vorgestellte Ansatz basiert auf der *Erhaltung von Belegtheit*, angelehnt an die Masseerhaltung in der Physik.



(a) BOF-Schätzung (b) BOFUM-Schätzung

(c) BOF-Schätzung (d) BOFUM-Schätzung

Abbildung 4.6: Ein konventionell implementierter BOF mit linearer Bewegungsschätzung übersieht die Möglichkeit zur Kollision (a), oder sagt diese fälschlicherweise voraus (c). Unter Zuhilfenahme von Hintergrundwissen über den Straßenverlauf lässt sich eine korrekte Schätzung durchführen (b,d).

4.3.1 Variablendefinitionen

N**:** Menge aller n Zellindizes. Das zweidimensionale Gitter besitzt die Dimension $\sqrt{n} \times \sqrt{n}$.

$$N = \{1, \ldots, n\} \tag{4.8}$$

O : Vektor, der die Belegtheit aller Zellen beschreibt und dessen Elemente die Werte *'occupied'* oder *'not occupied'* annehmen können.

$$O = \begin{pmatrix} O_1 \\ \vdots \\ O_n \end{pmatrix} \in \{occ, nocc\}^n \tag{4.9}$$

V: Vektor zur Beschreibung der Geschwindigkeit der Belegtheit einer Zelle (nicht der Zelle selbst) in x- und y-Richtung. $V_i = (\dot{x}_i, \dot{y}_i)$ ist diskretisiert in Zellen je Zeitschritt Δt. Leere Zellen besitzen keine Geschwindigkeit.

$$V = \begin{pmatrix} V_1 \\ \vdots \\ V_n \end{pmatrix} \in \{\mathbb{Z} \times \mathbb{Z}\}^n \tag{4.10}$$

X: Zellzustand zum Zeitpunkt t, bestehend aus Belegtheit und Geschwindigkeit.

$$X = (O, V) \tag{4.11}$$

X^-: Zellzustand zum Zeitpunkt $t-1$.

$$X^- = (O^-, V^-) \tag{4.12}$$

R: Matrix zur Beschreibung der Erreichbarkeit einer Zelle c von Zelle a aus. Diese Informationen werden mithilfe topographischer Karten gewonnen (vgl. Abschnitt 4.3.5).

$$R \in \{reach, nreach\}^{n \times n} \tag{4.13}$$

T: Transitionsvektor. Die Transition für eine Zelle T_i entspricht j, falls sich die Belegtheit in Zelle i im nächsten Zeitschritt nach j bewegt. Diese abstrakte Beschreibung fasst Geschwindigkeit, Erreichbarkeit und alles vorhandene Wissen über eine Zellbewegung zusammen und erlaubt die Verwendung von Kontextwissen.

$$T = \begin{pmatrix} T_1 \\ \vdots \\ T_n \end{pmatrix} \in N^n \tag{4.14}$$

Z: Messvektor mit Sensorwerten über Belegtheit und Geschwindigkeit jeder Zelle. Die Geschwindigkeit ist der Vollständigkeit halber enthalten, wird jedoch vom hier verwendeten Laserscanner nicht geliefert.

$$Z = (Z_O, Z_V),\ Z_O \in \{occ, nocc\}^n, Z_V \in \{\mathbb{Z} \times \mathbb{Z}\}^n \tag{4.15}$$

4.3.2 Dekomposition der Verbundwahrscheinlichkeit

Mit der bedingten Verteilung aller Variablen lässt sich durch Marginalisierung auf den a posteriori Zustand schließen.

$$\begin{aligned} P(X, X^-, R, T, Z) &= P(X, X^-, R, T) P(Z|X, X^-, R, T) \\ &= P(X, X^-, R, T) P(Z|X) \\ &= \prod_{c \in N} \underbrace{P(X_c, X^-, R, T)}_{\text{Prädiktion}} \underbrace{P(Z_c|X_c)}_{\text{Korrektur}} \end{aligned} \tag{4.16}$$

Die Zellen werden als unabhängig voneinander betrachtet. $P(Z_c|X_c)$ beschreibt das Beobachtungsmodell. Die Prädiktion wird folgendermaßen zerlegt:

$$P(X_c, X^-, R, T) = P(X^-, R, T)P(X_c|X^-, R, T) \tag{4.17}$$

- $P(X^-, R, T)$ stellt die bedingte Verteilung der Transition dar und beschreibt den Zusammenhang zwischen Transition, Erreichbarkeit und a priori Zustand. Transition und Belegtheit werden als unabhängig voneinander angenommen.

$$\begin{aligned} P(X^-, R, T) &= P(O^-, V^-, R, T) \\ &= \prod_{i \in N}(O_i^-) \prod_{j \in N} P(T_j, V^-, R) \end{aligned} \tag{4.18}$$

  $P(O_i^-)$ bezeichnet die a priori Wahrscheinlichkeit einer Belegtheit für Zelle i, $P(T_j, V^-, R)$ die Verbundwahrscheinlichkeit für Transition, Erreichbarkeit und a priori Geschwindigkeit der Zelle j.

- $P(X_c|X^-, R, T)$ beschreibt das Prozess- oder Bewegungsmodell für die Belegtheit in Gitterzellen. Aufgrund der Tatsache, dass die Transition sowohl Geschwindigkeit als auch Erreichbarkeit enthält, gilt der folgende Zusammenhang:

$$P(X_c|X^-, R, T) = P(O_c|O^-, T)P(V_c|O^-, T) \tag{4.19}$$

  $P(O_c|O^-, T)$ beschreibt die Belegtheit einer Zelle c in Abhängigkeit von der a priori Belegtheit und der Transition. $P(V_c|O^-, T)$ stellt die a posteriori Geschwindigkeit einer Zelle c bei gegebener a priori Belegtheit und Transition dar.

4.3.3 Filtermodelle

Nachdem die Verbundwahrscheinlichkeit im vorigen Abschnitt unter Betrachtung von bedingten Unabhängigkeiten zerlegt wurde, werden im folgenden Abschnitt Filtermodelle formuliert, die das Verhalten der Belegtheit von Zellen beschreiben.

Beobachtungsmodell

Das Beobachtungsmodell beschreibt, welche Messwerte bei gegebenem Zustand zu erwarten sind. Aus Gründen der Unabhängigkeit von Sensortechniken wird eine direkte Messung der Belegtheit von Zellen, sowie der Unsicherheit ρ in der Belegtheitsmessung, angenommen. Bei dem verwendeten Laserscanner wird eine Vorverarbeitung von Daten durchgeführt, bei der die 3D-Messpunkte auf eine zweidimensionale Ebene projiziert werden (siehe auch [Kammel 08]).

$$\begin{aligned} P(Z_{O,c} = z_c|X_c) &= P(Z_{O,c} = z_c|O_c = o_c) \\ &= \begin{cases} 1-\rho, & z_c = o_c \\ \rho, & \text{else} \end{cases}, \ \rho \in [0,1] \end{aligned} \tag{4.20}$$

Prozessmodell

Das Prozessmodell wird unter der Annahme aufgestellt, dass sich nur belegte Zellen ausbreiten können. Für die Belegtheit einer Zelle c reicht es aus, wenn mindestens eine belegte Vorgängerzelle a auf sie abgebildet wird. Folglich werden nicht belegte Vorgängerzellen im Prädiktionsschritt ignoriert. Um die Belegtheit einer Zelle zu berechnen wird eine logische ODER-Verknüpfung aller Vorgänger durchgeführt. Werden mehrere Vorgängerzellen auf eine Zielzelle abgebildet, so spiegelt sich dies in der Belegtheit und in deren Geschwindigkeitsvektor wider.

$$\begin{aligned} & P(O_c = occ | O^-, T) \\ = & \begin{cases} 1, & \vee_{a \in N}((O_a = occ) \wedge (T_a = c)) \\ 0, & \text{sonst} \end{cases} \\ = & \; 1 - P(O_c = nocc | O^-, T) \end{aligned} \tag{4.21}$$

Die bedingte Wahrscheinlichkeit dafür, dass die Zelle nicht belegt ist, lässt sich effizienter berechnen und wird deshalb bevorzugt:

$$\begin{aligned} & P(O_c = nocc | O^-, T) \\ = & \begin{cases} 1, & \wedge_{a \in N}((O_a = nocc) \vee (T_a \neq c)) \\ 0, & \text{sonst} \end{cases} \\ = & \prod_{a \in N} (1 - P(O_c = occ | O_a^-, T_a)) \end{aligned} \tag{4.22}$$

Eine Zielzelle c mit Vorgänger a ist belegt, falls die Transition von a der Zelle c entspricht und a belegt war:

$$P(O_c = occ | O_a^-, T_a) = \begin{cases} 1, & (O_a = occ) \wedge (T_a = c) \\ 0, & \text{sonst} \end{cases} \tag{4.23}$$

Die Prädiktion der Geschwindigkeit erfolgt in ähnlicher Art und Weise. Auch hier besitzen nur belegte Vorgängerzellen a einen Einfluss auf die Geschwindigkeit der Zielzelle c. Die Hilfsfunktion $v : N \times N \rightarrow \mathbb{Z} \times \mathbb{Z}$ enthält diese Zuordnung. Sie ordnet eine Kombination von Zellindizes einer Geschwindigkeit zu und bedient sich wiederum einer zweiten Hilfsfunktion $pos : N \rightarrow \mathbb{Z} \times \mathbb{Z}$. Letztere wandelt Indizes in Zellpositionen um.

$$v(a, c) = \frac{pos(c) - pos(a)}{\Delta t}, \; pos(i) = (i \mod n, i \div n) \tag{4.24}$$

Aufgrund der Annahme einer *sicheren* Geschwindigkeit kommt nur eine Vorgängerzelle zur Erfüllung dieser Bedingung in Frage. Die Verbundverteilung des Erwartungswertes der Geschwindigkeit $P(\hat{V})$ kann daher folgendermaßen definiert werden:

$$P(\hat{V}_c = v(a, c) | O^-, T) = P(O_c = occ | O_a^-, T_a) \tag{4.25}$$

Die in die Beschleunigung eingebrachte Unsicherheit wird nach der Geschwindigkeits-Inferenz angewandt.

Transitionsmodell

Zum Erhalt der Belegtheitsmenge wird die Transitionsverteilung für jede Vorgängerzelle a berechnet. Dies beinhaltet eine Normalisierung über alle Nachfolgezellen m, die mit dem Normalisierungsfaktor μ durchgeführt wird.

$$\begin{aligned} & P(T_a = c, V^-, R) \\ = \; & \frac{P(T_a = c, V_a^-, R_{a,1}, \ldots, R_{a,n})}{\sum_{m \in N} P(T_a = m, V_a^-, R_{a,1}, \ldots, R_{a,n})} \\ = \; & \mu_a P(V_a^-) P(R_{a,c}) P(T_a = c | V_a^-, R_{a,c}) \end{aligned} \tag{4.26}$$

$$P(T_a = c | V_a^-, R_{a,c}) = \begin{cases} 1, & V_a^- = v(a,c) \wedge R_{a,c} = \mathit{reach} \\ 0, & \text{sonst} \end{cases} \tag{4.27}$$

4.3.4 Berechnung

Die Berechnung des a posteriori Zustandes einer Zelle c erfolgt durch Marginalisieren:

$$P(X_c|Z) = \frac{\sum_{X^-,R,T} P(X_c, X^-, R, T, Z)}{\sum_{X^-,R,T,X_c} P(X_c, X^-, R, T, Z)} \tag{4.28}$$

$$\propto \sum_{X^-,R,T} P(X_c, X^-, R, T, Z) \tag{4.29}$$

Nach der in Abschnitt 4.3.2 vorgestellten Dekomposition teilt sich die Berechnung in einen Prädiktions- und einen Korrekturschritt auf. Die Transitionsberechnung kann dabei als Teil der Prädiktion isoliert betrachtet werden:

$$\begin{aligned} & \sum_{O^-,V^-,R,T} P(X_c, X^-, R, T, Z) \\ = \; & \underbrace{P(Z_c|O_c)}_{\text{Korrektur}} \\ & \underbrace{\sum_{O^-,V^-,R,T} \underbrace{P(V^-, R, T)}_{\text{Transition}} P(O^-) P(O_c|O^-, T) P(V_c|O^-, T)}_{\text{Prädiktion}} \end{aligned} \tag{4.30}$$

Transition

Die Transitionswahrscheinlichkeit wird als Teil der Prädiktion benötigt und wie folgt berechnet:

$$\begin{aligned} & P(T = t) \\ = \; & \sum_{V^-,R} \prod_{a \in N} \mu_a P(V_a^-) P(R_{a,t_a}) P(T_a = t_a | V_a^-, R_{a,t_a}) \\ = \; & \prod_{a \in N} \mu_a P(V_a^- = v(a, t_a)) P(R_{a,t_a} = \mathit{reach}) \end{aligned} \tag{4.31}$$

Geschwindigkeitsschätzung

Da Unsicherheit nur in der Beschleunigung, nicht aber in der Geschwindigkeit angenommen wird, kann nur eine Vorgängerzelle a die Bedingung $P(\hat{V}_c = v(a,c)|O_a^-, T_a)$ erfüllen:

$$\begin{aligned} & P(\hat{V}_c = v(a,c)) \\ &= \sum_{O^-,T} P(O^-)P(T)P(\hat{V}_c = v(a,c)|O^-,T) \\ &= P(O_a^- = occ)P(T_a = c) \end{aligned} \tag{4.32}$$

Die Geschwindigkeitsschätzung wird durch ein Verrauschen der inferierten Geschwindigkeit $\hat{V}$ erzeugt:

$$P(V_c) = \sum_{\hat{V}_c} P(V_c|\hat{V}_c)P(\hat{V}_c) \tag{4.33}$$

Der Fehler in der Beschleunigung wird als normalverteilt mit Mittelwert null und Kovarianz Σ_t angenommen:

$$v_c = \hat{v}_c + w_t \Delta t,\ w_t \sim \mathcal{N}(0, \Sigma_t) \tag{4.34}$$

$P(V_c|\hat{V}_c)$ ergibt sich durch eine Integration über die resultierende Normalverteilung entsprechend der Zelldiskretisierung.

Belegtheitsschätzung

Zur Berechnung der a posteriori Wahrscheinlichkeit für die Belegtheit einer Zelle ist es sinnvoll, zuerst die Wahrscheinlichkeit dafür zu bestimmen, dass die Zelle nicht belegt ist. Dieser Fall wird nur durch eine einzige a priori Konfiguration verursacht, nämlich genau dann, wenn keine der möglichen belegten Vorgängerzellen auf die Zielzelle abgebildet wird.

$$\begin{aligned} & P(O_c = nocc) \\ &= \sum_{O^-,T} P(O^-)P(T)P(O_c = nocc|O^-,T) \\ &= \sum_{O^-,T} \prod_{a \in N} P(O_a^-)P(T_a)P(O_c = nocc|O_a^-,T_a) \\ &= \prod_{a \in N} \sum_{O_a^-,T_a} P(O_a^-)P(T_a)P(O_c = nocc|O_a^-,T_a) \\ &= \prod_{a \in N} \left[1 - P(O_a^- = occ)P(T_a = c)\right] \end{aligned} \tag{4.35}$$

Die Belegtheitswahrscheinlichkeit einer Zelle errechnet sich dann wie folgt:

$$\begin{aligned} & P(O_c = occ) \\ &= 1 - P(O_c = nocc) \\ &= 1 - \prod_{a \in N} \left(1 - P(O_a^- = occ)P(T_a = c)\right) \end{aligned} \tag{4.36}$$

4.3.5 Berücksichtigung von Hintergrundwissen

Wie zu Beginn in Abschnitt 4.3 angesprochen besteht ein Hauptunterschied zu anderen BOF-Implementierungen in der Verwendung von Hintergrundwissen. Die Annahme, dass das Verhalten von Verkehrsteilnehmern eng mit der Straßenstruktur verbunden ist, führt zu einer Koppelung von topographischer Karte und Erreichbarkeitswahrscheinlichkeiten von Zellen.

Um dieses Wissen in die Prädiktion einfließen zu lassen, wurde eine Erreichbarkeitsmatrix R mit den Elementen $R_{a,c} \in \{reach, nreach\}$ eingeführt, die angibt, wie wahrscheinlich die Bewegung eines Objektes von Zelle a zu Zelle c ist. Zur Unterscheidung von logischen Bereichen wird für jede Zelle die Klasse des Untergrunds angegeben. Zunächst soll eine Einteilung in die folgenden Untergrundklassen genügen: $U = \{lane, sidewalk, unknown\}$. Die Klasse einer Zelle wird durch die Hilfsfunktion $u : N \rightarrow U$ aufgelöst. Die Erreichbarkeitswahrscheinlichkeit errechnet sich nach folgender Vorschrift:

$$P(R_{a,c} = reach) = S_{u(a),u(c)} w(a, c). \tag{4.37}$$

Eine Matrix $S \in [0, 1]^{U \times U}$ und eine Gewichtungsfunktion $w : N \times N \rightarrow [0, 1]$ dienen dazu, die folgenden Annahmen über Erreichbarkeit auszudrücken:

- Wechseln der Untergrundklasse: Objekte bleiben üblicherweise auf ihrer Art von Untergrund. Dieser Zusammenhang wird durch S beschrieben. $S_{u(a),u(c)}$ definiert die Wahrscheinlichkeit, dass Zellen der Untergrundklasse u_a von Zellen der Klasse u_c aus erreicht werden.

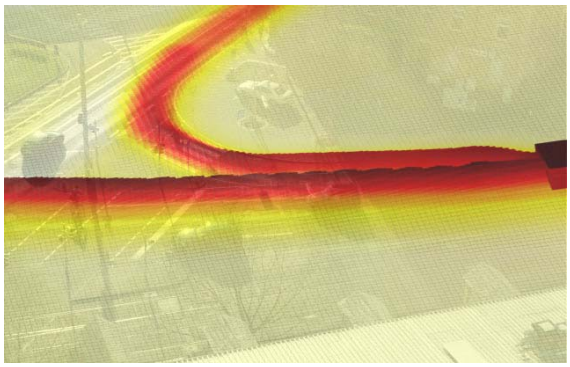
- Bewegung abseits von Spuren: Wenn sich Vorgänger- oder Zielzelle abseits der Spuren befinden, wird von einem Fußgänger ausgegangen – eine Vorzugsrichtung kann in diesem Fall nicht angegeben werden. Die Gewichtsfunktion beträgt dann $w(a, c) = 1$.

- Bewegung auf einer Spur: Fahrzeuge auf Spuren bewegen sich üblicherweise entlang der Spurrichtung. Eine Bewegung entgegen der Fahrtrichtung, quer zur Fahrtrichtung oder auf Nachbarspuren ist möglich, aber unwahrscheinlicher. $w(a, c)$ stellt den jeweiligen Zusammenhang dar.

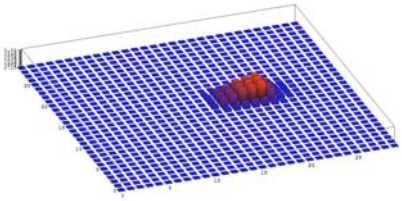
Abbildung 4.7 zeigt Ergebnisse der Erreichbarkeitsanalyse für eine Verkehrsszene am Durlacher Tor in Karlsruhe (vgl. Abb. 4.7a). Ausgehend von einem Fahrzeug (dunkle Zellen) am rechten Rand von Abbildung 4.7b sind die Zielzellen mit hoher Erreichbarkeitswahrscheinlichkeit rot markiert. Aufgrund der Topologie des Straßenverlaufs werden zwei Hypothesen für die Bewegung des Fahrzeuges erstellt. Eine laterale Bewegung innerhalb der Spuren ist möglich (gelb dargestellt), ein Verlassen der Spur wird hingegen ausgeschlossen (helle Bereiche).

Die Abbildungen 4.7c-e zeigen eine zuverlässige Prädiktion der Fahrzeugbewegung unter Verwendung von Erreichbarkeitswahrscheinlichkeiten. Eine Korrektur mit Messdaten erfolgt

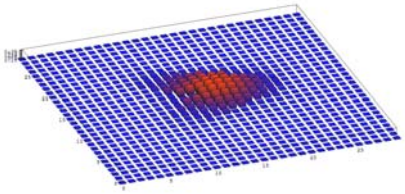
(a) Straßenszene



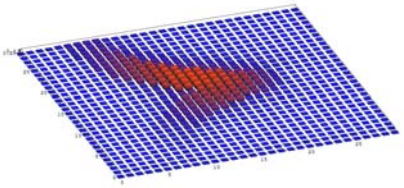
(b) Überlagerte Erreichbarkeit



(c) Prädiktion für $T_0 = 0\,\mathrm{s}$



(d) Prädiktion für $T_1 = 1{,}2\,\mathrm{s}$



(e) Prädiktion für $T_2 = 2{,}4\,\mathrm{s}$

Abbildung 4.7: Beispiel für die Anwendung der Erreichbarkeitsanalyse in einem Kreuzungsszenario (a). Ausgehend von einem Fahrzeug am rechten Rand wird in (b) die Erreichbarkeit der Verkehrsszene überlagert. Dargestellt ist die Summe der Erreichbarkeiten aller Ursprungszellen. Rot bedeutet hohe Erreichbarkeit, farblose Zellen sind nicht erreichbar. Die Abbildungen (c-e) zeigen Prädiktionsschritte des Filters ohne Korrektur unter Verwendung der Erreichbarkeitsinformationen. Durch die ungleiche Spurzuordnung wird nach zwei Prädiktionsschritten eine Tendenz zum Abbiegen erkennbar.

nicht. Das in der Simulation platzierte Fahrzeug mit Initialgeschwindigkeit $v = 8\,\mathrm{m/s}$ und einer Unsicherheit in der Beschleunigung von $\sigma = 2\,\mathrm{m/s^2}$ befindet sich auf geradeaus führender und abbiegender Spur zugleich. Daraus ergeben sich zwei Hauptrichtungen für die Zellbewegungen. Das Beispiel zeigt die grundlegende Fähigkeit des Filters, Hintergrundwissen zu verwenden, um die Prädiktion maßgeblich zu verbessern. Nur mit einer robusten Prädiktion kann den Problemen von verrauschten Messdaten und Verdeckungen begegnet werden.

4.3.6 Filterergebnisse

Das vollständige Filter wurde mit realen Messdaten eines Laserscanners in einer Straßenszene getestet, wie sie Abbildung 4.8a darstellt. Der Versuchsträger mit Sensor nimmt ein abbiegendes Fahrzeug auf, welches während des Abbiegevorganges für ca. 0,5 s vollständig verdeckt wird. Der Vorgang wird zu bestimmten Zeitpunkten $t_1..t_4$ näher betrachtet. Die zum Zeitpunkt t_1 aufgenommenen Messdaten des Laserscanners sind in Abbildung 4.8b dargestellt. Abbildung 4.8 zeigt für vier Zeitpunkte die Messmatrix und die Ergebnisse von Prädiktion und Schätzung. Die Messung liefert lediglich binäre Angaben über Zellbelegtheit und darüber, ob Informationen gewonnen wurden. An den weiss markierten Stellen der

Messmatrix konnte aufgrund von Verdeckung keine Messung erfolgen. Die Zellgeschwindigkeiten wurden ausschließlich über Inferenz bestimmt. Das abbiegende Fahrzeug ist in der Messung durch die gelbe Umrandung gekennzeichnet. Zu den vier Zeitpunkten lassen sich folgende Beobachtungen anstellen:

Zum Zeitpunkt t_1 nähert sich das Fahrzeug der Kreuzung. Aus der Darstellung des Fahrzeuges im Prädiktionsschritt geht hervor, dass sich die Zellgeschwindigkeiten dem bewegten Fahrzeug angepasst haben. Das Fahrzeug fährt zum Zeitpunkt t_2 in den verdeckten Bereich ein. Obwohl die Front des Fahrzeuges in den Messdaten bereits verschwunden ist, zeigt die Schätzung noch eine Belegtheit für diese Zellen.

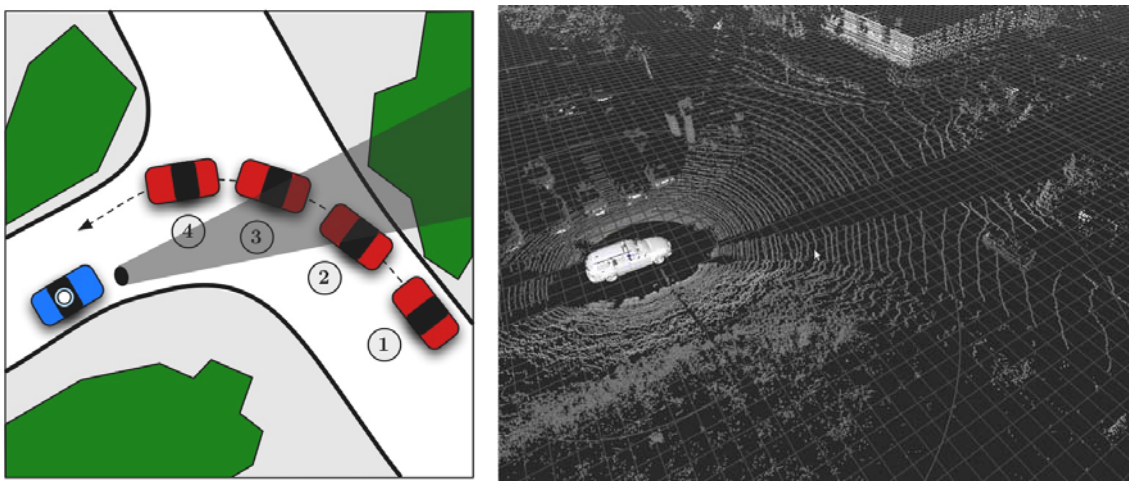
Der verdeckte Bereich wird zum Zeitpunkt t_3 wieder verlassen. Das Filter passt sich den neuen Messwerten schnell an, da diese gut mit der Prädiktion übereinstimmen. Durch die Unsicherheit in der Beschleunigung profitieren Zellen, für die neue Messwerte vorliegen unmittelbar von Geschwindigkeitsinformationen, die vor dem Verschwinden erlangt wurden. Nach dem Abbiegevorgang ist das Fahrzeug zum Zeitpunkt t_4 wieder komplett sichtbar, was sich in genaueren Schätzungen äußert. Erkennbar sind noch leichte Belegtheiten rechts vom eigentlichen Fahrzeug, die aus der zweiten Hypothese *geradeaus fahren* resultieren.

Diese Formulierung eines Bayesian Occupancy Filter zeigt, dass es möglich ist, Hintergrundwissen zu verwenden, um eine wesentliche Verbesserung bei der Schätzung von Bewegungen auf Zellebene zu erzielen. Die Funktionsfähigkeit wird insbesondere durch die Tatsache unterstrichen, dass eine Bestimmung von Geschwindigkeiten ohne eigentliche Messung nur durch Inferenz möglich ist. Für die folgenden Abschnitte wird davon ausgegangen, dass eine dynamische Gefahrenkarte existiert, die sich aus Objektprädiktionen und/oder einer Schätzung von Zellbewegungen zusammensetzt.

4.4 Bahnplanung

Hauptziel der Bahnplanung ist es, einen Planer bereitzustellen, der in einer Vielzahl von Situationen anwendbar ist. Es sollen einerseits komplexe geometrische Bahnen geplant werden, wie sie für Park- oder Wendemanöver notwendig sind. Andererseits ist auch eine Planung weitreichender, glatter Pfade für hohe Fahrgeschwindigkeiten wünschenswert, die sich für Überlandfahrten eignen. Es soll weiterhin möglich sein, Hintergrundwissen über die vorliegende Planungsaufgabe aus dem Situationskontext zu verwenden, um die Suche zu beschleunigen.

Geschlossene Lösungen zur Bahnplanung werden in Anhang D ausführlich vorgestellt und diskutiert. Sie sind oftmals für spezielle Aufgabenstellungen optimiert und lassen sich schwer mit dem Konzept der Gefahrenkarten kombinieren. Ein inkrementeller Graphensuchealgorithmus bietet hier mehr Möglichkeiten zur Kostenabschätzung und Gewichtung einzelner Planungskriterien. Eine Gefahrenkarte soll neben Start- und Zielkonfiguration als Vorgabe für den Planer dienen, um unzulässige Konfigurationen zu kennzeichnen und einen Gefah-



(a) Schematisch dargestellter Vorgang (b) Messdaten des Laserscanners zum Zeitpunkt t_1

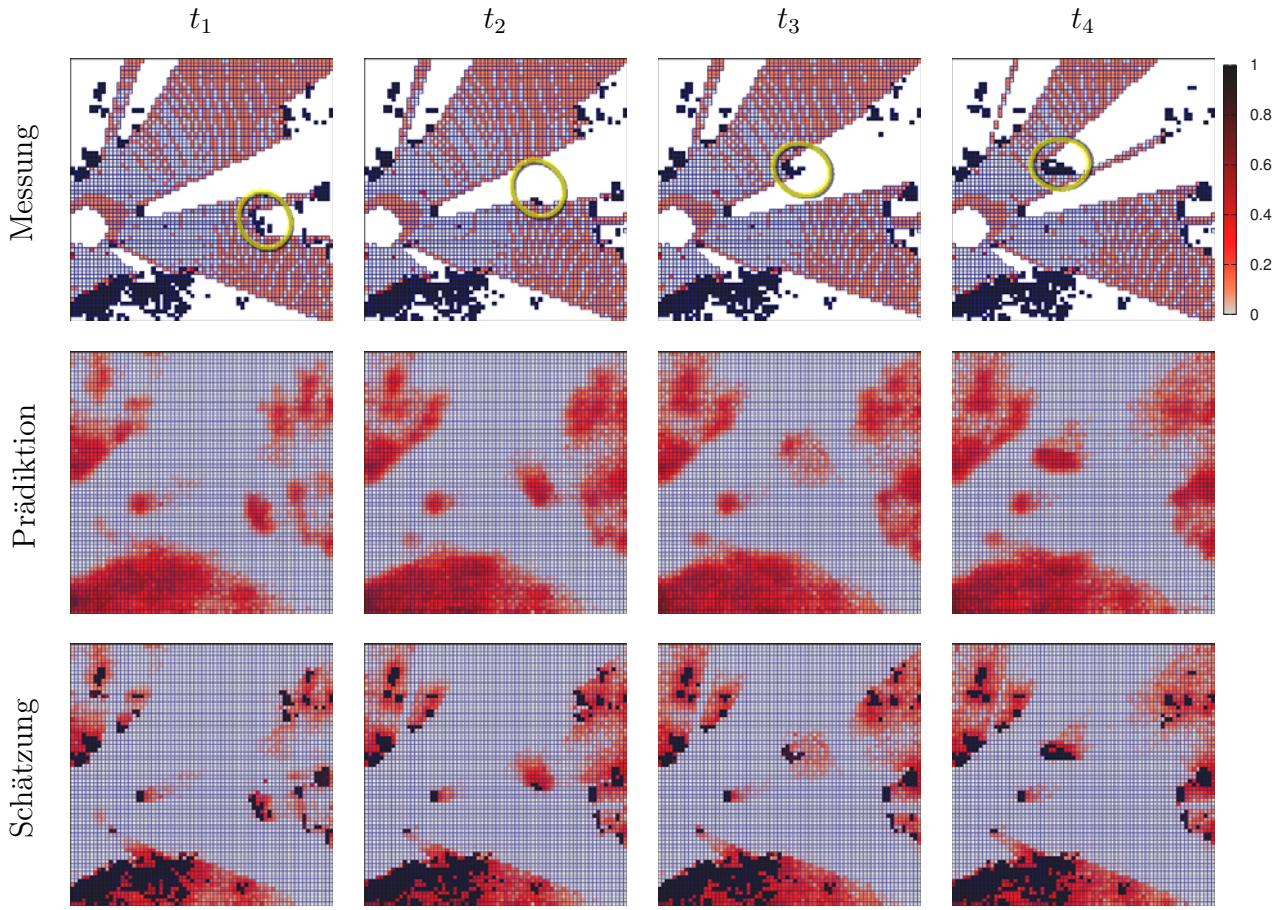


Abbildung 4.8: Schematische Darstellung des Abbiegevorganges (a) und Messdaten des Laserscanners zum Zeitpunkt t_1 (b). Zur Veranschaulichung des Filterprozesses sind im unteren Teil zu vier Zeitpunkten $t_1..t_4$ die real gewonnenen Messdaten sowie Ergebnisse nach Prädiktion und Schätzung gezeigt. Das abbiegende Fahrzeug ist in den Messdaten gelb umrandet.

renwert in der Planung zu berücksichtigen. Die Anforderungen an das Bahnplanungsmodul lassen sich wie folgt zusammenfassen:

- Flexible Pfaderzeugung für dynamische (Stadt- und Überlandfahrten) und geometrisch komplexe Manöver (Park- und Wendemanöver).
- Pfadplanung in Echtzeit.
- Berechnung glatter Pfade mit kontinuierlicher Krümmungsänderung.
- Berücksichtigung eines Gefahrenwertes in der Planung, hervorgerufen durch Hindernisse oder unerwünschte Bereiche.

Vormalige Überlegungen, durch horizontalen Schnitt der Gefahrenkarte einen zweidimensionalen Fahrkorridor zu erhalten und diesen als Toleranzbereich an die Bahnplanung zu reichen, wurden verworfen (siehe Abbildung 4.9). Grund dafür war eine zu starke Reduzierung der Informationen: Wäre der aktuelle Fahrkorridor nicht fahrbar, so müsste von Seiten der Bahnplanung ein Korridor mit höherem Gefahrenpotential angefordert werden. Für solch einen iterativen Prozess ist jedoch in kritischen Situationen keine Zeit. Die Bahnplanung soll deshalb alle Informationen über Gefahrenwerte erhalten, um diese optimal nutzen zu können.

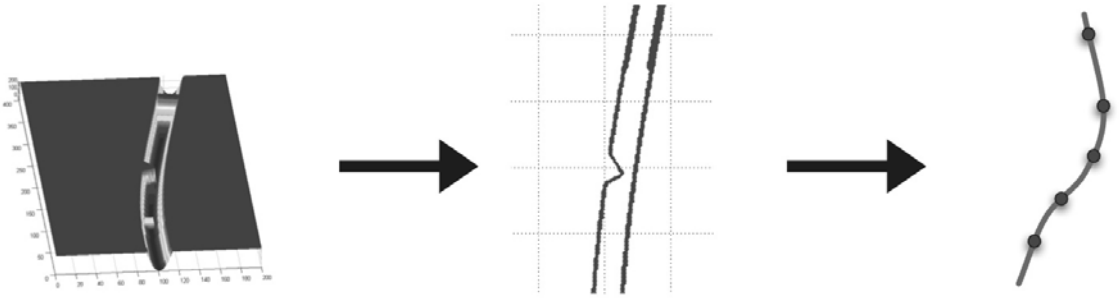


Abbildung 4.9: Eine Reduktion der Gefahrenkarte zu einem zweidimensionalen Korridor vor der eigentlichen Bahnplanung kann zu mehreren Iterationen führen, bis eine Gefahrenstufe gefunden wurde, für die eine Bahnplanung innerhalb des Korridors möglich ist. Um diesen Abstimmungsprozess zu vermeiden, soll stattdessen die gesamte Gefahrenkarte in der Bahnplanung berücksichtigt werden.

4.4.1 Kürzeste-Weg-Suche

Auf das Problem der Planung geometrisch komplexer Pfade für Einparkvorgänge oder Wendemanöver soll ein A*-Graphensuchealgorithmus angesetzt werden, um einen kürzesten Pfad, ausgehend von einer Startkonfiguration $\vec{x}_s = \{x_s, y_s, \theta_s, \phi_s\}$ zu einer Zielkonfiguration $\vec{x}_g = \{x_g, y_g, \theta_g, \phi_g\}$ zu finden. In der üblichen A*-Terminologie werden die Kosten, um vom Startpunkt zu einem beliebigen Punkt n zu gelangen, mit g_n bezeichnet, die geschätzten

Kosten von Knoten n zum Zielpunkt mit h_n. Die Gesamtkosten für einen Knoten setzen sich aus den Einzelsummen zusammen:

$$f_n = g_n + h_n \tag{4.38}$$

Die beiden Anteile gleichen sich bei der Bewegung vom Start zum Ziel aus. Ein wesentlicher Punkt bei der Wahl der Schätzfunktion ist die Zulässigkeit derselben. Eine Schätzfunktion ist zulässig, falls sie die tatsächlichen Kosten nicht überschätzt. Wäre dies der Fall, so könnte eine optimale Lösung versehentlich zu schlecht bewertet und damit verworfen werden. Die Suchfunktion muss weitere Kriterien berücksichtigen, um nicht nur fahrbare sondern auch effiziente Pfade zu erzeugen. Bei Aufstellung der Bewegungskosten g_n und der Schätzfunktion h_n sind folgende Forderungen zu berücksichtigen:

- Ein Vorwärtsfahren wird dem Rückwärtsfahren vorgezogen.
- Richtungswechsel sind als zusätzlicher Aufwand zu werten.

Um sicherzustellen, dass der resultierende Pfad fahrbar ist, wird zur Exploration neuer Knotenpunkte ein kinematisches Fahrzeugmodell gemäß Abbildung 4.10 verwendet. Der Suchgraph wird mit diesem Modell zur Laufzeit aufgebaut. Die resultierenden Pfadpunkte dienen als Ausgleichspunkte für eine B-Spline-Kurve, welche von der Regelung abgefahren wird (vgl. Abschnitt 5).

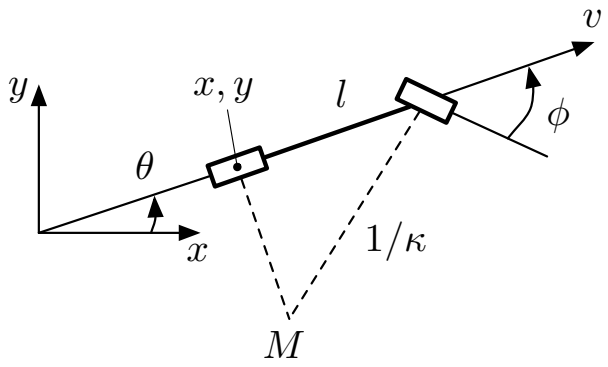


Abbildung 4.10: Kinematisches Fahrzeugmodell mit Geschwindigkeit v, Position (x, y), Ausrichtung θ und Achsabstand l. Die Krümmung κ wird durch die Beschränkung des Lenkwinkels ϕ begrenzt.

Schätzfunktion

Mit der in [Dubins 57] beschriebenen Methode kann eine kürzeste Strecke zwischen Start- und Zielpunkt mit vorgegebener Orientierung berechnet werden, wobei sich die Strecke aus zwei Kreisbögen mit Radius r und einer Tangente dazwischen zusammensetzt (*Circle-Tangent-Circle, CTC*). Diese Methode ist sehr ähnlich zu der in [Reeds 90] vorgestellten, aber etwas schneller zu berechnen. Der Hauptunterschied liegt in der Verbindung beider

Kreisbögen durch entweder eine Tangente oder einen weiteren Kreis. Auch wenn die zweite Möglichkeit etwas kürzer ist, so fällt dieser Unterschied praktisch kaum ins Gewicht.

Die hier vorgestellte Variante ist an [Dubins 57] angelehnt, verwendet jedoch unterschiedliche Radien r_s und r_g in den Start- und Zielpunkten. Abbildung 4.11 zeigt Start- und Zielkreise, auf welchen die Kreisbögen liegen können und eine Möglichkeit der Verbindung über die Kreise B und D. Verschiedene Kombinationen AC, AD, BC und BD sind möglich, allerdings können aufgrund der Drehrichtung im Fall von AC und BD nur äußere Tangenten, bei den Verbindungen AD und BC nur innere Tangenten (mit Richtungswechsel) verwendet werden, um die Zielorientierung zu erreichen. Von vier Möglichkeiten, zwei Kreise miteinander zu verbinden, sind also nur jeweils zwei gültig.

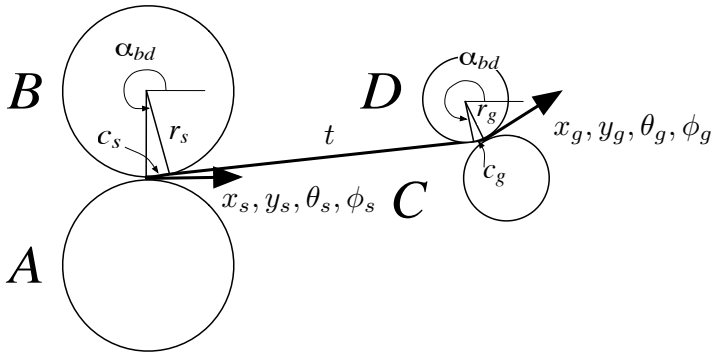


Abbildung 4.11: CTC-Verbindung zwischen Startkonfiguration $x_s, y_s, \theta_s, \phi_s$ und Zielkonfiguration $x_g, y_g, \theta_g, \phi_g$ unter Verwendung der Kreiskombination BD und Typ $dir_{1,1,1}$ (Vorwärtsfahrt sowohl auf erstem Kreisbogen als auch auf Tangente und auf zweitem Kreisbogen). c_s und c_g sind die Längen der Kreisbögen in Start- und Zielpunkten, t ist die Länge der Tangente. Die Radien r_s und r_g ergeben sich aus den geforderten Lenkwinkelstellungen.

Die Gesamtlänge $L = c_s + t + c_g$ ist nicht exakt die Strecke, die das Fahrzeug fahren kann, da ein Anhalten an den Übergängen mit sprunghafter Krümmungsänderung vermieden werden soll. Diese Art der Abschätzung ist dennoch zulässig, da die tatsächliche Strecke größer ist und die Schätzfunktion eine untere Schranke bildet. Der Suchraum wird durch diese Ungenauigkeit geringfügig vergrößert. Um die exakte Strecke abzuschätzen, könnte das *Continuous-Curvature*-Verfahren [Scheuer 96, Scheuer 97] verwendet werden. Die Berechnung ist allerdings weit aufwändiger und lässt sich nicht als schnelle Schätzfunktion auf viele Konfigurationen anwenden. Im hindernisfreien Raum müssten bei guter Schätzfunktion zwar kaum Knotenpunkte exploriert werden. Sobald sich jedoch Hindernisse im Weg befinden, vergrößert sich der Suchraum signifikant und eine schnelle Abschätzung bringt dann Vorteile.

Zieht man alle möglichen Kombinationen von Vorwärts- und Rückwärtsfahren auf den drei Verbindungsteilen in Betracht, so resultieren 32 Lösungen, falls sich die Kreise nicht schneiden. Schneiden sich BC oder AD, fallen jeweils 8 Lösungen weg. Um die Wegstrecke abschätzen zu können, sind die Einzellängen der Verbindungsstücke zu berechnen. Das be-

deutet, dass zunächst die vier Winkel für jeden Kreis bekannt sein müssen, unter welchen die Tangenten diesen berühren. Um zwischen diesen Winkeln zu unterscheiden, wird eine Annotation der Form $\alpha_{CIRCLE,TYPE,DIR}$ verwendet, wobei *CIRLE* einen Kreis aus {A,B,C,D} bezeichnet, *TYPE* mit $\{\|, \times\}$ angibt, ob eine äußere oder innere (kreuzende) Tangente unter diesem Winkel anliegt, und *DIR* mit $\{f, r\}$ die Fahrtrichtung auf der berührenden Tangente als *vorwärts* oder *rückwärts* beschreibt.

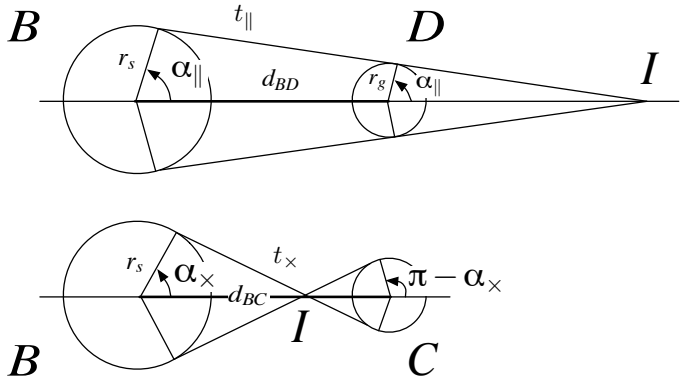


Abbildung 4.12: Verschiedene Möglichkeiten, Kreise über äußere (oben) und innere (unten) Tangenten $t_\|$ und $t_\times$ zu verbinden. Je nach Kombination sind nur äußere oder nur innere Tangenten möglich, um die korrekte Zielausrichtung zu erhalten. d stellt den Abstand der Kreismittelpunkte dar.

$\alpha_{C\times r}$ ist damit ein Winkel des Kreises C, unter welchem das Fahrzeug auf einer inneren Tangente in Rückwärtsrichtung ankommt, zwangsläufig von Kreis B. Zunächst lassen sich mit dem Strahlensatz Hilfswinkel $\alpha_\|$ und $\alpha_\times$ und Längen der äußeren bzw. inneren Tangenten $t_\|$ und $t_\times$ berechnen (siehe Abildung 4.12):

$$\alpha_\| = \pm acos(\tfrac{r_s - r_g}{d}), t_\| = (r_s - r_g)tan(\alpha_\|) \tag{4.39}$$

$$\alpha_\times = \pm acos(\tfrac{r_s + r_g}{d}), t_\times = (r_s + r_g)tan(\alpha_\times) \tag{4.40}$$

Die Anwendung der verschiedenen Variationen von Kreiskombinationen und Fahrtrichtungen führt zu insgesamt 16 Winkeln, welche in Tabelle 4.1 zusammengefasst sind. α_{BD} steht für die Orientierung der Verbindungslinie d_{BD} zwischen den Mittelpunkten der Kreise B und D. Jeder in Tabelle 4.1 aufgelistete Winkel kann mit oder gegen den Uhrzeigersinn erreicht werden. Dies muss bei der Berechnung einer der 32 Lösungen berücksichtigt werden. Eine Möglichkeit bei Verwendung der Kreise BD wäre, rückwärts auf B zu fahren, vorwärts auf der verbindenden Tangente, und wiederum rückwärts auf dem Zielkreis D. Exemplarisch für diesen Fall bedeutet das für die Längen von Kreisbögen und Tangente:

$$c_s = (\delta_s - \pi + \alpha_{B\|f})r_s \tag{4.41}$$

$$c_g = (\delta_g + \pi + \alpha_{D\|f})r_g \tag{4.42}$$

$$t = (r_s - r_g)tan(\alpha) \tag{4.43}$$

	fwd	rev
$\alpha_{A\parallel}$	$\frac{acos(r_s - r_g)}{d_{AC}} + \alpha_{AC}$	$-\frac{acos(r_s - r_g)}{d_{AC}} + \alpha_{AC}$
$\alpha_{A\times}$	$\frac{acos(r_s + r_g)}{d_{AD}} + \alpha_{AD}$	$-\frac{acos(r_s + r_g)}{d_{AD}} + \alpha_{AD}$
$\alpha_{B\parallel}$	$-\frac{acos(r_s - r_g)}{d_{BD}} + \alpha_{BD}$	$\frac{acos(r_s - r_g)}{d_{BD}} + \alpha_{BD}$
$\alpha_{B\times}$	$-\frac{acos(r_s + r_g)}{d_{BC}} + \alpha_{BC}$	$\frac{acos(r_s + r_g)}{d_{BC}} + \alpha_{BC}$
$\alpha_{C\parallel}$	$\frac{acos(r_s - r_g)}{d_{AC}} + \alpha_{AC}$	$-\frac{acos(r_s - r_g)}{d_{AC}} + \alpha_{AC}$
$\alpha_{C\times}$	$-\frac{acos(r_s + r_g)}{d_{BC}} + \alpha_{BC} + \pi$	$\frac{acos(r_s + r_g)}{d_{BC}} + \alpha_{BC} + \pi$
$\alpha_{D\parallel}$	$-\frac{acos(r_s - r_g)}{d_{BD}} + \alpha_{BD}$	$\frac{acos(r_s - r_g)}{d_{BD}} + \alpha_{BD}$
$\alpha_{D\times}$	$\frac{acos(r_s + r_g)}{d_{AD}} + \alpha_{AD} + \pi$	$-\frac{acos(r_s + r_g)}{d_{AD}} + \alpha_{AD} + \pi$

Tabelle 4.1: Absolute Winkelstellungen, unter welchen die jeweiligen Tangenten die Kreise berühren.

Die Notation $dir_{c_s,t,c_g} = (-1, 1, -1)$ beschreibt die Lösung für eine bestimmte Kombination unter Angabe der Laufrichtung auf Startkreis, Tangente und Zielkreis, um später die Richtungsinformation zur Kostenberechnung nutzen zu können. Die weiteren Lösungen werden analog zum vorgestellten Beispiel berechnet und sollen hier nicht explizit aufgelistet werden. Damit sind alle Längen der Verbindungselemente für 32 Lösungen bekannt und können mit Wissen über die Fahrtrichtung für eine Kostenabschätzung verwendet werden.

Suchalgorithmus

In der Bahnplanung wird per Definition keine Geschwindigkeit errechnet. Sie ist deshalb im Zustandsvektor des Fahrzeuges nicht enthalten. Die Festlegung einer Obergrenze v_{max} ist jedoch notwendig, um daraus Größen wie die maximal erlaubte Krümmung $\kappa = \frac{a_{lat,max}}{v_{max}}$ zu berechnen, wobei $a_{lat,max}$ die maximale zulässige Querbeschleunigung darstellt. Der Fahrzeugzustand wird in jedem Knoten n beschrieben durch

$$\vec{x_n} = (x_n, y_n, \theta_n, \phi_n)^T \tag{4.44}$$

und wird als Teil des Knotenzustandes

$$\vec{N_n} = (\vec{x_n}, g_n, h_n, dir_n)^T, dir_n \in \{-1, 1\} \tag{4.45}$$

gespeichert. Im Zustand eines Knotens sind weiterhin Informationen über die aktuellen Bewegungskosten des Knotens g_n, der geschätzten Kosten h_n und der Bewegungsrichtung dir_n enthalten. Ausgehend von einem Vaterknoten werden beim Aufbau des Graphen Nachfolgeknoten erzeugt. Als Vaterknoten wird in jedem Durchlauf der Knoten mit den geringsten Gesamtkosten f_n aus der offenen Liste ausgesucht. Zu Beginn einer Suche ist der Startknoten alleiniger Inhalt der Liste und damit Ausgangspunkt für die Exploration. Mit dem

in Abbildung 4.10 vorgestellten Fahrzeugmodell ergeben sich die Fahrzeugzustände in den Nachfolgeknoten zu:

$$\vec{x_i} = \begin{pmatrix} x_i \\ y_i \\ \theta_i \\ \phi_i \end{pmatrix} = \begin{pmatrix} x_p + cos(\theta_i)\epsilon \\ y_p + sin(\theta_i)\epsilon \\ \theta_p + \frac{\epsilon}{l} sin\phi_i \\ \phi_i \end{pmatrix} \tag{4.46}$$

wobei ϵ die Schrittweite als Abstand zwischen zwei Knoten enthält und i die Anzahl der Diskretisierungsschritte des Lenkwinkels darstellt. ϕ_i durchläuft den Wertebereich von ϕ_{min} bis ϕ_{max} mit Schrittweite $\frac{\phi_{max}-\phi_{min}}{i}$, muss aber gleichzeitig die Forderung nach Krümmungsbeschränkung $|\phi_i| \leq atan(l\kappa)$ erfüllen. Weiterhin ist die Krümmungsänderung $\dot{\kappa}$ aufgrund der begrenzten Lenkwinkeländerung $\dot{\phi}_{max}$ ebenfalls beschränkt, so dass $|\phi_i - \phi_p| \leq \Delta\phi$ mit $\Delta\phi = \frac{\epsilon}{v_{max}}\dot{\phi}_{max}$ eingehalten werden muss, um Klothoiden-ähnliche Pfade zu erzeugen. Position und Orientierung im Zustandsvektor sollten aus Gründen der benötigten Genauigkeit und der geforderten Glattheit der resultierenden Bahn nicht (wie der Lenkwinkel) diskretisiert werden. Aufgrund der beschränkten Rechenkapazität ist es allerdings sinnvoll, sehr nahe beieinander liegende Knoten mit ähnlicher Orientierung nicht gesondert zu betrachten. In der Implementierung wird daher ein Boole'sches Gitter mit der Dimension des Zustandsvektors verwendet, um die Konfigurationen zu diskretisieren und um festzuhalten, ob eine Konfiguration bereits besucht wurde.

Nach der Berechnung der Nachfolgeknoten werden die zugehörigen Kosten g_n und h_n berechnet. Die Bewegungskosten für einen Folgeknoten ergeben sich aus den Bewegungskosten g_p des Vaterknotens und der dazwischen zurückgelegten Strecke ϵ unter Berücksichtigung der Fahrtrichtung. Etwaige Kosten für Richtungsänderungen gehen separat ein, so dass sich für die Gesamtkosten ergibt:

$$g_n = g_p + \epsilon \cdot g_{dir} + g_{chdir} \tag{4.47}$$

Die Kosten sind in den Koeffizienten

$$g_{dir} = \begin{cases} 1 & , dir_n = 1 \\ P_{rev} & , dir_n = -1 \end{cases} \tag{4.48}$$

$$g_{chdir} = \begin{cases} 0 & , dir_n = dir_p \\ P_{chdir} & , dir_n \neq dir_p \end{cases} \tag{4.49}$$

enthalten, wobei g_{dir} die Zusatzkosten P_{rev} für Rückwärtsfahren beinhaltet, g_{chdir} die für einen Richtungswechsel veranschlagten Kosten. Als geschätzte Kosten h_n wird das Minimum aller 32 (oder 16) verfügbaren Lösungen verwendet. Die jeweiligen Lösungen berechnen sich aus den Einzellängen von Kreisbögen und Tangente (vgl. Abschnitt *Schätzfunktion*) multipliziert mit ihren Gewichtungskoeffizienten. Weiterhin gehen die veranschlagten Richtungswechsel in die Kosten ein, so dass für die abgeschätzten Kosten resultiert:

$$\begin{aligned} h_n &= min(h_{n1} \cdots h_{n32}) \\ &= c_s h_{dir_{c_s}} + t h_{dir_s} + c_g h_{dir_{c_g}} + h_{chdir} \end{aligned} \tag{4.50}$$

Als Koeffizienten dienen

$$h_{dir_{c_s,t,c_g}} = \begin{cases} 1 & , dir_{\{c_s,t,c_g\}} = 1 \\ P_{rev} & , dir_{\{c_s,t,c_g\}} = -1 \end{cases} \tag{4.51}$$

$$h_{chdir} = \begin{cases} 0 & , dir_n = dir_{c_s} = dir_t = dir_{c_g} \\ 1,2,3 \cdot P_{chdir} & , \sum dir_{\{n,c_s,t\}} \neq dir_{\{c_s,t,c_g\}} \end{cases} \tag{4.52}$$

Im Beispiel in Abbildung 4.11 wäre die Lösung mit den geringsten Schätzkosten eine Fahrt in Vorwärtsrichtung über alle Segmente mit $dir_{c_s,t,c_g} = (1,1,1)$, was bei entsprechender Richtung im Knoten $dir_n = 1$ zum vereinfachten Ausdruck führt:

$$h_n = c_s + t + c_g \tag{4.53}$$

Nach der Erzeugung von Folgeknoten und deren Kostenberechnung werden diese in der offenen Liste platziert und stellen die Blätter des Suchbaumes dar. Gleichzeitig wird der Vaterknoten in die geschlossene Liste verschoben. Der Knoten mit den geringsten Kosten f_n aus der offenen Liste bildet nun den neuen Vaterknoten für den nächsten Durchlauf. Die Suche ist beendet, sobald ein Folgeknoten die Endbedingung erfüllt, welche üblicherweise durch Erreichen eines gewissen Toleranzbereiches um die Zielkonfiguration gegeben ist. Über die Verkettung von Folge- zu Vaterknoten kann der resultierende Pfad innerhalb der geschlossenen Liste zurückverfolgt werden.

Abbildung 4.13 zeigt drei Suchergebnisse mit unterschiedlichen Start- und Zielkonfigurationen $\vec{x_s}$ und $\vec{x_g}$. Die Fahrzeugabmessungen sind als schwarzes Rechteck dargestellt, die Knotenpunkte des Pfades sind durch graue Segmente verbunden. Die grau dargestellten Konfigurationen zeigen alle Vaterknoten und stellen damit die geschlossene Liste dar. Abbildung 4.13 (a) zeigt eine Planung mit geringer Orientierungsdifferenz von $\vec{x_s}$ nach $\vec{x_g}$. Es zeigt sich, dass manche Knoten im Kurveninneren zwischenzeitlich beste Knoten waren, aber letztendlich nicht zur Zielkonfiguration führen konnten. Dieser Effekt entsteht durch die Diskrepanz zwischen geschätzter und tatsächlicher Strecke und der damit verbundenen unterschätzenden Heuristik.

Abbildung 4.13 (b) zeigt eine Dreipunktwende als komplexeres Manöver. Die Einstellungen der Parameter P_{dir} und P_{chdir} bestimmen die Bereitschaft zum Rückwärtsfahren oder für einen Richtungswechsel und entscheiden letztendlich darüber, ob eine Dreipunktwende oder ein vollständiger Kreis ohne Richtungswechsel gefahren wird. Abbildung 4.13 (c) veranschaulicht das Ergebnis derselben Zielvorgabe wie oben rechts, allerdings mit erhöhten Kosten für einen Richtungswechsel. Die Berücksichtigung der menschlichen Einschätzung, wann ein Richtungswechsel einer längeren Fahrstrecke vorzuziehen ist, kann helfen, sinnvolle Kosten für die Parameter in Form der physikalischen Einheit *Meter* zu finden.

Verwendung von statischen Gefahrenkarten

Wie in Abschnitt 4.2 beschrieben wird die Bahnplanung nicht nur durch Objekte in der Umgebung eingeschränkt, sondern auch durch potentiell gefährliche Bereiche wie bspw. Spuren

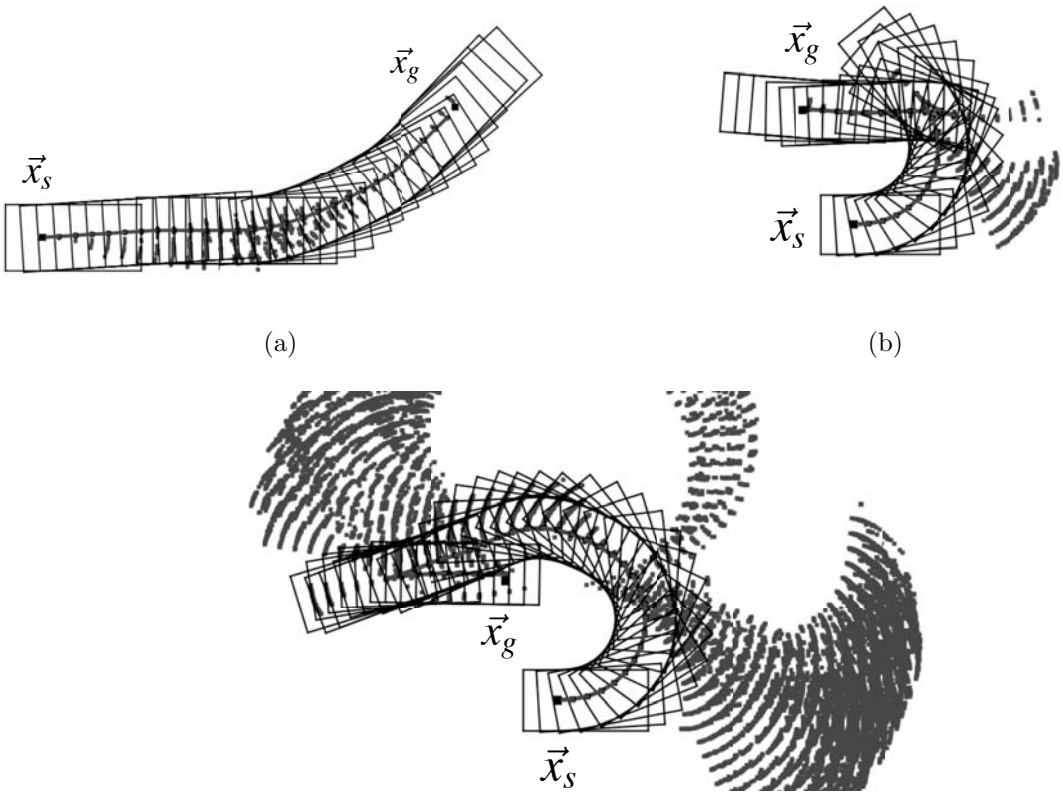


Abbildung 4.13: Ergebnisse der Pfadplanung für verschiedene Start- und Zielkonfigurationen $\vec{x_s}$ und $\vec{x_g}$. Die Schrittweite ϵ wurde auf 0,75 m gesetzt, die Kosten für Rückwärtsfahren betragen $P_{rev} = 1,5$. Die Pixel in der Grafik stellen Knotenpunkte der geschlossenen Liste dar. Für eine einfache Bahnplanung (a) werden 5265 Knotenpunkte benötigt, bei Kosten für einen Richtungswechsel von $P_{chdir} = 5$. Für die Dreipunktwende (b) werden mit demselben Parametersatz 5451 Knoten expandiert. Eine Erhöhung der Kosten für Richtungswechsel auf $P_{chdir} = 15$ resultiert in nur noch einem Richtungswechsel für das Wendemanöver (c). Es werden jedoch 23305 Knotenpunkte benötigt, um einen Pfad zu finden. Die Berechnungszeit betrug ca. 0,1 s für 10000 Knotenpunkte.

des Gegenverkehrs. Gefahrenkarten berücksichtigen beide Aspekte gleichermaßen und nehmen zudem eine Unterteilung hinsichtlich der Höhe der Gefahr vor. In diesem Abschnitt werden zunächst statische Gefahrenkarten betrachtet, welche für den gesamten Planungszeitraum konstant sind und keine Informationen über die zukünftige Umfeldentwicklung beinhalten. Eine statische Gefahrenkarte ist vergleichbar mit der ersten Schicht einer dynamischen Gefahrenkarte. Jede Gefahrenkarte verursacht zusätzliche Kosten r_n, die in die Kostenberechnung eines Knotens einfliessen:

$$f_n = g_n + h_n + r_n \tag{4.54}$$

Die zusätzlichen Kosten für eine Konfiguration berechnen sich wie folgt:

$$r_n = \sum_{i=1}^{N} \frac{cell[i]}{N} P_O \tag{4.55}$$

wobei die Variable i über alle Zellen N läuft, die vom Fahrzeug in der akuellen Konfiguration überdeckt werden (vgl. Abbildung 4.14). Der Gefahrenwert einer Zelle $cell[i]$ liegt im Bereich $\{0 \cdots 255\}$. Die von der Gefahr verursachten Kosten entstehen durch Mittelwertbildung und Gewichtung mit einem Faktor P_O. Letzterer dient dazu, eine Relation zu Bewegungs- und Schätzkosten herzustellen, welche in Form einer Distanz ausgedrückt werden. Ist eine einzelne Zelle $cell[i]$ belegt, so wird der Knoten sofort verworfen.

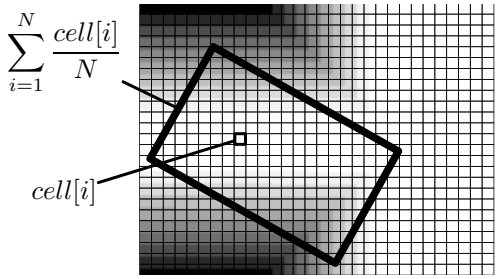
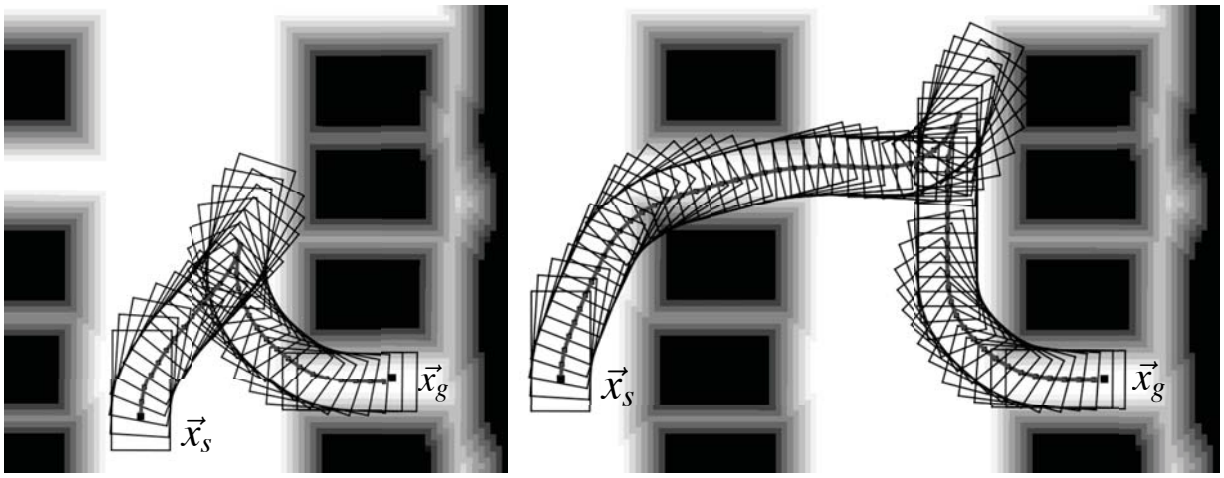


Abbildung 4.14: Der Gefahrenwert einer Konfiguration berechnet sich als Mittelwert der Gefahrenwerte jener Zellen, die das Fahrzeug überdeckt. Eine Multiplikation mit P_O setzt den Gefahrenwert in Relation zu Bewegungs- und Schätzkosten.

Das Bahnplanungsmodul wurde in dieser Form in einer Simulationsumgebung mit synthetischen Gefahrenkarten getestet, in welche statische Hindernisse eingetragen wurden. Eine lineare Gefahrenfunktion wurde gewählt, um den Übergang zwischen Hindernis (schwarz) und Freiraum (weiss) zu modellieren. Abbildung 4.15 (a) zeigt ein rückwärtiges Einparkmanöver in eine Parklücke zwischen anderen Fahrzeugen. Es wird deutlich, dass graue Bereiche mit erhöhtem Gefahrenpotential zwar überfahren werden können, eine großflächige Überdeckung allerdings vermieden wird. Der Parameter P_O legt die Toleranz gegenüber Gefahrenwerten fest und definiert so bspw. den Abstand zu Hindernissen. Für ein einfaches Einparkmanöver zwischen anderen Fahrzeugen war eine Exploration von 3481 Knotenpunkten notwendig. Die Schrittweite ϵ wurde hierzu auf 0,75 m gesetzt.

Eine etwas schwierigere Planungsaufgabe mit längerer Wegstrecke und zusätzlichen Hindernissen zeigt die rechte Grafik (b) in Abbildung 4.15. Da Hindernisse in der Schätzfunktion nicht berücksichtigt werden läuft die Suche zunächst auf Hindernisse auf, bevor ein freier Pfad gefunden wird. Dies führt zu einer wesentlichen Vergrößerung des Suchbaumes gegenüber vorigem Beispiel und insgesamt 33047 Knotenpunkten.

Abbildung 4.16 zeigt als Planungsaufgabe ein Parallelpark-Manöver. Wieder lässt sich beobachten, dass ein Überfahren der grauen Bereiche zwar möglich ist, ein gewisser



(a) (b)

Abbildung 4.15: Simulation eines rückwärtigen Einparkmanövers mit Startknoten vor dem Parkplatz (a) oder hinter einer zweiten Reihe von Fahrzeugen (b). 3481 bzw. 33047 Knotenpunkte werden expandiert, um eine Lösung zu finden. Die Berechnungszeiten betragen mit der aufwändigeren Kostenfunktion für diese Beispiele 0,5 s und 5,0 s.

Sicherheitsabstand jedoch eingehalten wird. Der Planungsvorgang erfordert eine Expandierung von nur 2431 Knotenpunkten. Ein wesentlicher Vorteil dieser Methode gegenüber geschlossenen Lösungen ist, dass im Voraus keine Umgebungsparameter wie verfügbarer Platz oder Abmessungen der Parklücke bestimmt werden müssen, die oftmals schwer aus einem Pixelbild zu bestimmen sind.

In Abbildung 4.17 wurde eine Bahnplanung innerhalb einer Spur mit und ohne Hindernis ausgeführt. Wenngleich diese Bahnplanungsvariante der *kürzesten Pfade* auf geometrisch komplexe Manöver, und weniger auf Bahnen für reguläre Fahrmanöver, optimiert wurde, so

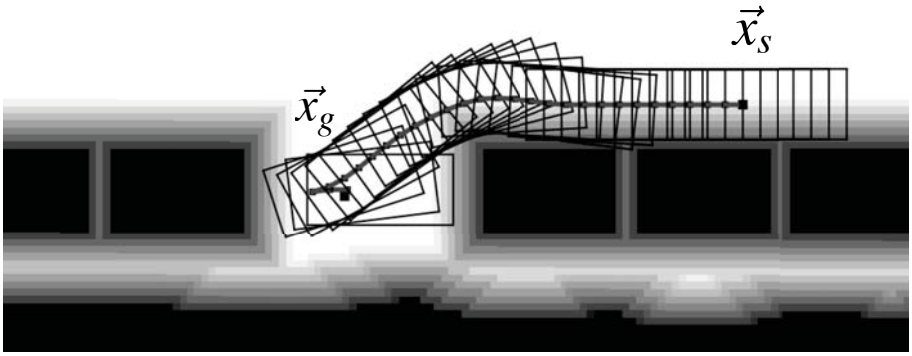
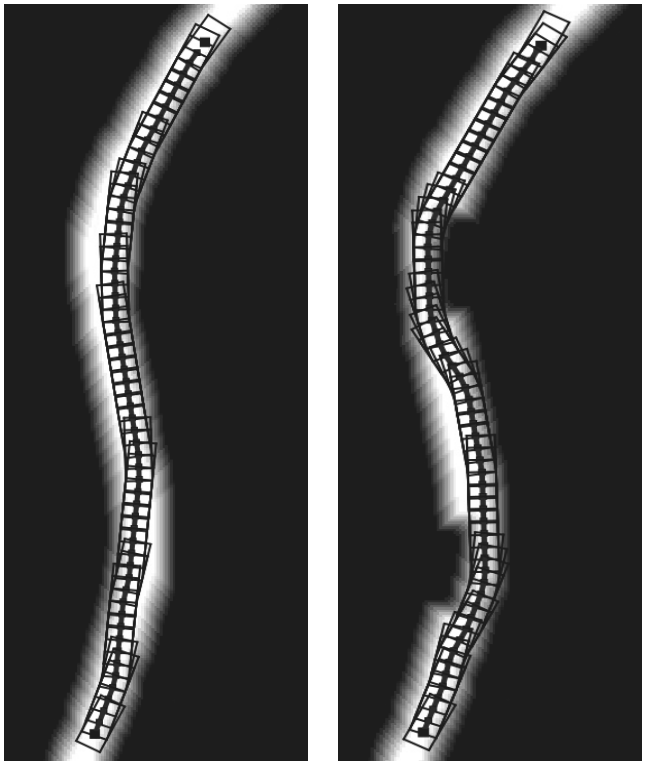


Abbildung 4.16: Für die Planung eines Parallelpark-Manövers werden 2431 Knotenpunkte expandiert. Im Gegensatz zu geschlossenen Lösungen müssen neben den Zielkoordinaten keine weiteren Umgebungsparameter bestimmt werden.

liefert sie dennoch zufriedenstellende Lösungen. Der aufgrund der übermittelten Fahrintention in Form einer Gefahrenkarte stark beschränkte Konfigurationsraum führt zu schnellen Planungszeiten unter 0,5 s und exploriert lediglich 781 bzw. 1495 Knoten. Komponenten wie das Verhaltensnetzwerk legen die Fahrintentionen fest, indem Gefahrenfunktionen parametriert werden, um bevorzugte Bereiche zu kennzeichnen (siehe Abschnitt 4.2.2).



(a) (b)

Abbildung 4.17: Pfadplanung innerhalb einer Fahrspur mit stark eingeschränktem Konfigurationsraum. Lediglich 781 Knotenpunkte werden expandiert um eine Lösung bei hindernisfreier Bahn zu finden (a). Mit zu umfahrenden Hindernissen steigt die Anzahl auf 1495 Knotenpunkte (b).

Hintergrundwissen über die durchzuführende Planungsaufgabe kann für eine situationsabhängige Parametrierung des Planers verwendet werden, um bspw. die Kosten für einen Richtungswechsel oder die Explorationsstrategie einzustellen. Mit angepassten Parametersätzen für die jeweiligen Aufgaben wie Dreipunktwende, Einparken oder Spurfolgen kann die Planungszeit weiter verkürzt werden. Im Falle der Spurplanung ist z. B. eine Exploration rückwärts nicht sinnvoll, so dass durch Beschränkung auf Vorwärtsfahren die Hälfte der möglichen Konfigurationen nicht untersucht werden muss.

Die vorgestellte Methode der *Kürzesten-Weg-Suche* mit genauer Angabe von Start- und Zielpunkt ist für kontinuierliches Fahren entlang von Spuren weniger geeignet. Werden, wie in Abbildung 4.17 gezeigt, die Intentionen der Verhaltenssteuerung hinreichend genau in

der Gefahrenkarte modelliert, so ist eher eine Risikominimierung entlang der vorgegebenen Potentialrinne erwünscht als das Anfahren einer exakten Zielposition. Für diesen Anwendungsfall wird ein weiterer Planungsmodus mit anderen Optimierungskriterien entworfen, wie im folgenden Abschnitt dargestellt.

4.4.2 Gefahren-Minimierung und Verwendung dynamischer Gefahrenkarten

Bei einer Modellierung der Fahrintention *Spur halten* in der Gefahrenkarte steht weniger die Berechnung kürzester Pfade im Mittelpunkt als vielmehr die Erzeugung einer gefahren- oder risikooptimierten Bahn. Die Angabe einer genauen Zielposition ist für diesen Anwendungsfall oftmals schwer möglich und nicht von Interesse. Für eine zweite Variante des Bahnplaners wird die Kostenfunktion f_n für einen Knoten n folgendermaßen abgeändert:

$$f_n = r_{g,n} + r_{h,n} \tag{4.56}$$

Die Gesamtkosten setzen sich aus einem Wert $r_{g,n}$, der die durch vorherige Knotenpunkte verursachten Kosten durch Gefahr darstellt, und den geschätzten verbleibenden Gefahrenkosten $r_{h,n}$, zusammen. Für die beiden Anteile gilt ausserdem:

$$r_{g,n} = r_p + r \tag{4.57}$$

$$r_{h,n} = \sum_{N-n}^{N} r_{max} \tag{4.58}$$

Die durch Gefahr verursachten Kosten $r_{g,n}$, um zu einem Knoten n zu gelangen, setzen sich aus dem Gefahrenwert r_p der Vaterkonfiguration und dem Wert r der Eigenkonfiguration zusammen. Der Gefahrenwert einer Konfiguration wird berechnet wie in Abbildung 4.14 veranschaulicht. Eine Abschätzung des verbleibenden Risikos ist in $r_{h,n}$ enthalten. Für eine konservative Abschätzung wird die Gefahr für alle verbleibenden Konfigurationen als maximal angenommen. Durch diese Maßnahme wird zudem ein Gefälle in Richtung des Zieles erzeugt und die Suche beschleunigt. Die Suchtiefe (Gesamtzahl der Konfigurationen in einem Zweig) richtet sich nach der geforderten Strecke oder der gewünschten Planungsvorausschau. Letzteres ist für die Einordnung einer Konfiguration ein besseres Maß als die Streckenlänge, auch vor dem Hintergrund der dynamischen Gefahrenkarten. Im Gegensatz zu Abschnitt 4.4.1 wird deshalb nicht in äquidistanten Schrittlängen ϵ geplant, sondern in konstanten Zeitscheiben Δt. Der Zustand eines Knotens wird entsprechend um die Zeit erweitert:

$$\vec{N_n} = (\vec{x_n}, g_n, h_n, dir_n, t)^T, dir_n \in \{-1, 1\} \tag{4.59}$$

Als Abbruchbedingung dient die geforderte Vorausschauzeit T_P. Die Geschwindigkeit des Eigenfahrzeugs wird während eines Planungsdurchlaufs auch hier als konstant angenommen und durch die Startkonfiguration x_s, y_s, ϕ_s, v_s vorgegeben. In Abschnitt 4.5 wird die Er-

weiterung bezüglich der Planung von Bewegungen diskutiert. Die Fahrzeugzustände in den Folgeknoten ändern sich im Vergleich zu Gleichung 4.46 zu:

$$\vec{x_i} = \begin{pmatrix} x_i \\ y_i \\ \theta_i \\ \phi_i \end{pmatrix} = \begin{pmatrix} x_p + cos(\theta_i) v_s \Delta t \\ y_p + sin(\theta_i) v_s \Delta t \\ \theta_p + \frac{v_s \Delta t}{l} sin\phi_i \\ \phi_i \end{pmatrix} \tag{4.60}$$

Während im vorigen Abschnitt zur Planung von Park- und Wendemanövern die Angabe einer Maximalgeschwindigkeit v_{max} ausreichend war, so ist hier die Angabe der tatsächlichen Geschwindigkeit v_s zur korrekten Synchronisierung von Zeit und Ort wichtig. Weicht die tatsächliche Geschwindigkeit nennenswert ab, so stimmt der zu einem Planungszeitpunkt t veranschlagte Ort nicht mit der zu erwartenden Position überein. Als Folge würde mithilfe der zu diesem Zeitpunkt t gültigen Gefahrenkarte eine Konfiguration überprüft, die dem Fahrzeug dann nicht entspricht.

No	T_1	T_2	T_3	T_4
1	0	0	0	0
2	1	0	0	-1
3	1	1	-1	-1
4	1	-1	0	0
5	1	1	1	1
6	0	0	-	-
7	1	0	-	-
8	1	-1	-	-
9	1	1	-	-
10	2	1	-	-
11	2	2	-	-

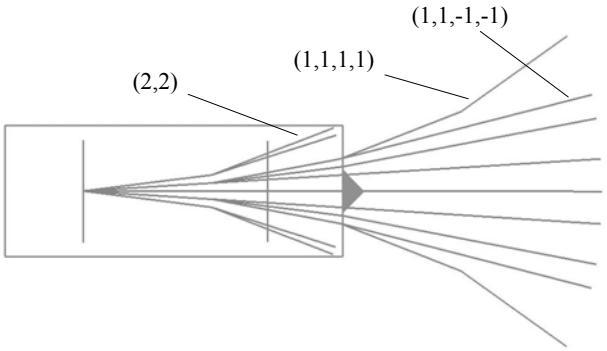
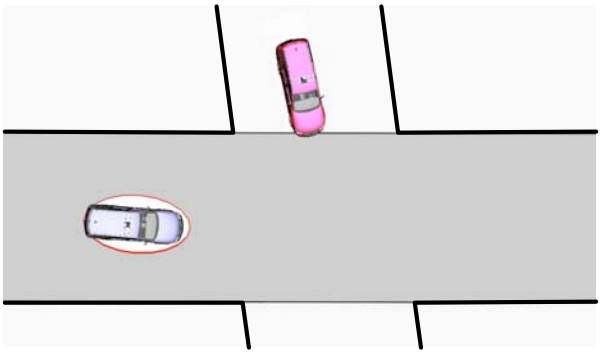


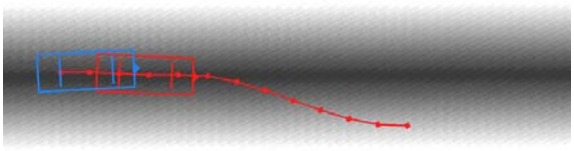
Abbildung 4.18: Tabelle zur Beschreibung vorgefertigter Teilsegmente des Pfades (links), und resultierende Pfadsegmente (rechts).

Die Erzeugung aller in einer Elternkonfiguration möglichen Kindkonfigurationen führt zu einem großen Suchraum und liefert fahrbare, aber in Einzelfällen sehr unregelmäßige Pfade. Beim Menschen lassen sich Lenkmanöver beobachten, die aus unterschiedlichen Geschwindigkeiten und Winkelstellungen resultieren, aber ein bestimmtes Bewegungsmuster aufweisen. Orientiert man sich an diesem Beispiel, dann können als Lösung statt einzelner Kindkonfigurationen ganze Teilsegmente des Pfades erzeugt werden. Diese sind zwar gemäß der in Gleichung 4.60 beschriebenen Gesetzmäßigkeit gebildet, durch hinterlegte Lenkwinkelverläufe über mehrere Zeitschritte wird die Anzahl der Möglichkeiten jedoch eingeschränkt. Abbildung 4.18 zeigt tabellarisch Bewegungsmuster und resultierende Pfade für eine Längsgeschwindigkeit von v=5 m/s. In der Tabelle sind nur Segmente in positiver Lenkrichtung verzeichnet – Segmente in negativer Richtung entstehen durch Multiplikation mit -1. Ausgehend von der aktuellen Konfiguration werden Nachfolgeknoten durch Variation des Lenkwinkels erzeugt. Dabei werden diskretisierte Lenkwinkel der Einheit 0,0875 rad schrittweise

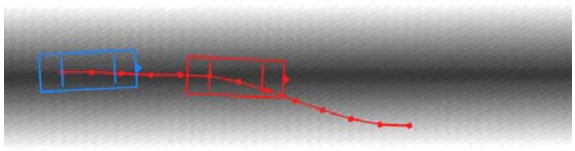
addiert oder subtrahiert. Beträgt der Lenkwinkel zu Beginn 0,2 rad, so resultiert nach Anwendung der dritten Vorschrift $\{1, 1, 1, 1\}$ nach dem vierten Zeitschritt eine Konfiguration mit Lenkwinkel 0,55 rad.



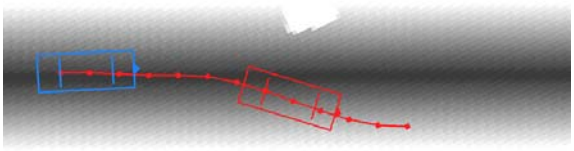
(a) Kreuzung mit Rechts-vor-Links-Situation: Das von links kommende Fahrzeug übersieht die Vorfahrt. Als Folge findet ein reaktives Ausweichmanöver statt.



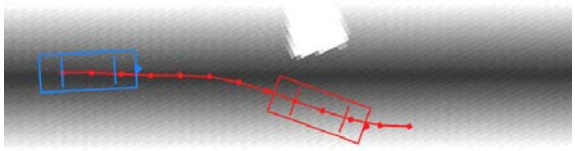
(b) $T_0 = 0,4\,\mathrm{s}$



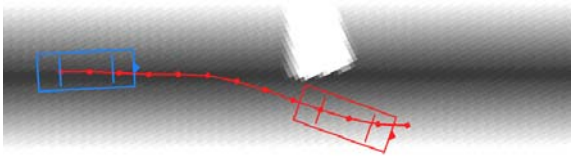
(c) $T_1 = 1,0\,\mathrm{s}$



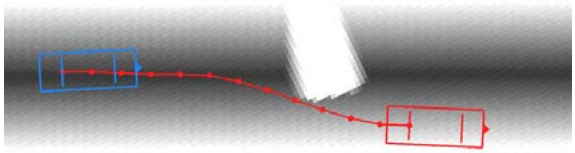
(d) $T_2 = 1,4\,\mathrm{s}$



(e) $T_3 = 1,6\,\mathrm{s}$



(f) $T_4 = 1,8\,\mathrm{s}$



(g) $T_5 = 2,4\,\mathrm{s}$

Abbildung 4.19: Gefahren- oder risikooptimierte Pfadplanung mit dynamischer Gefahrenkarte. Gezeigt sind mehrere Zeitschritte $T_0..T_5$ des Planungsvorganges mit den jeweils gültigen Gefahrenkarten und der geplanten Position des Eigenfahrzeuges (rot markiert). Die zum Ausweichen zur Verfügung stehende Spur ist dunkel dargestellt, nicht befahrbare Bereiche oder Hindernisse sind weiss.

Durch den eingeschränkten Konfigurationsraum verkürzt sich die Suche erheblich, so dass Planungszeiten von unter 100 ms erreicht werden. Abbildung 4.19 zeigt eine Planung mit dynamischer Gefahrenkarte zu verschiedenen Zeitpunkten. Für jeden Knotenpunkt wird zur Überprüfung die zu diesem Zeitpunkt gültige Schicht der Gefahrenkarte verwendet. Das gezeigte Ausweichmanöver resultiert damit automatisch aus der Gefahrenminimierung. Als Abbruchkriterum wurde eine Vorausschauzeit von 2,5 s gewählt.

Für ein kontinuierliches Fahren ist eine fortlaufende Planung mehrerer Pfadsegmente notwendig. Diese beginnen, mit Ausnahme des initialen Pfades, nicht am Eigenfahrzeug, sondern werden an einem Knotenpunkt des bisherigen Pfades *angeheftet*. Nur so kann die vorhandene Regelabweichung beibehalten werden – bei einem Nachführen des Pfades und sprunghafter Reduzierung des Regelfehlers würden die in Kapitel 5 vorgestellten Regler nicht korrekt arbeiten. Abbildung 4.20 veranschaulicht das Zusammensetzen eines kontinuierlichen Pfades aus mehreren Teilstücken.

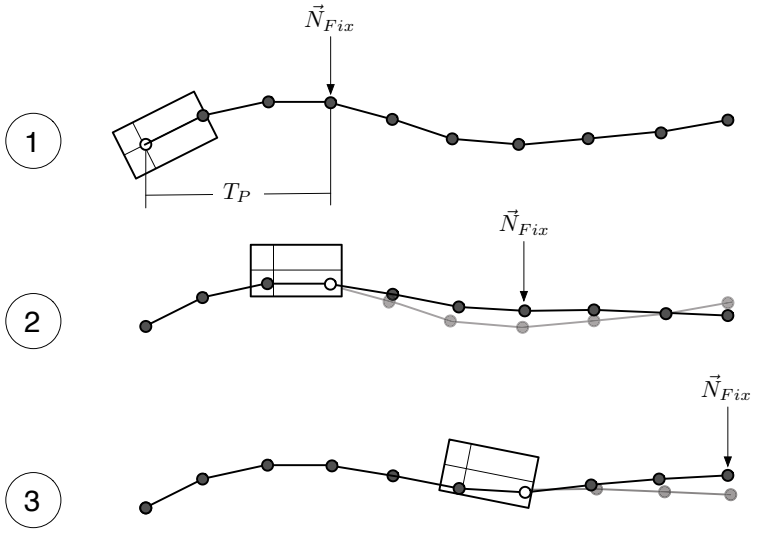


Abbildung 4.20: Ein kontinuierliches Fahren geplanter (Teil-)Pfade wird durch ein Zusammensetzen mehrerer Planungsergebnisse möglich. Die Abbildung zeigt drei aufeinander folgende Planungen. Nach einer initial mit der Eigenposition durchgeführten Planung darf der Pfad für die Dauer T_P der kommenden Planungsphase nicht geändert werden. Daraus ergibt sich ein Fixknoten $\vec{N}_{Fix}$. Dieser Knoten bildet den Startknoten für den jeweils nächsten Planungsschritt.

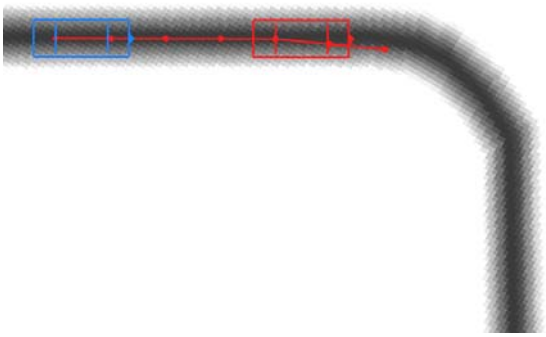
4.5 Bewegungsplanung

In den bislang vorgestellten Ansätzen wurde die Geschwindigkeit als konstant angenommen. Zur Planung von Einpark- oder Wendemanövern ist dies hinnehmbar, für eine dynamische

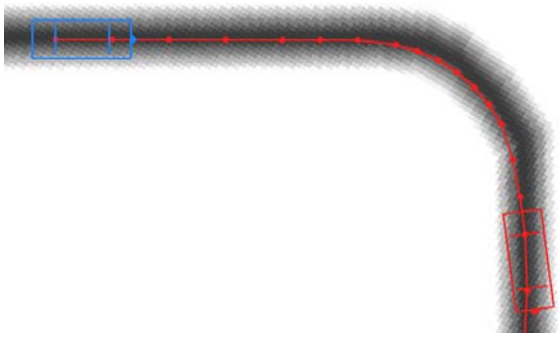
Planung bspw. auf Landstraßen jedoch nicht. Für die Berücksichtigung der Geschwindigkeit wird die Zustandsbeschreibung für einen Knoten n erweitert:

$$\vec{x_n} = (x_n, y_n, \theta_n, \phi_n, v_n)^T \tag{4.61}$$

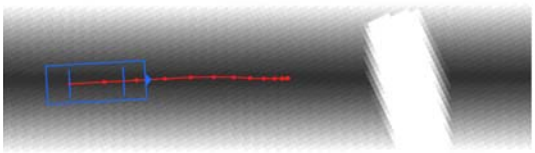
An dieser Stelle würde es naheliegen, den Suchraum um eine weitere Dimension aufzuspannen und die Beschleunigung als zusätzlichen Steuereingang zu verwenden. Der Aufwand für eine Planung würde sich bei einer Suche über alle möglichen Geschwindigkeitskonfigurationen signifikant erhöhen und muss daher eingeschränkt werden. Besonders bei der Dimension der Geschwindigkeit ist die Integration von Hintergrundwissen wünschenswert und einem *Ausprobieren* von Konfigurationen vorzuziehen. So steht bspw. der Kurvenverlauf einer Strecke bereits vor der Planung fest und sollte dem Bahnplaner in geeigneter Weise vermittelt werden. Die Übermittelung eines Geschwindigkeitsprofils von Seiten der Verhaltenssteuerung ist eine mögliche Lösung des Problems. Die Geschwindigkeit v_n eines Knotens n wird während des Aufspannens des Suchraumes aus dem hinterlegten Profil $v(t)$ errechnet.



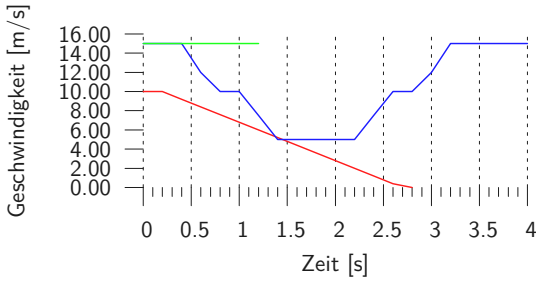
(a) Konstante Geschwindigkeit v=15 m/s



(b) Geschwindigkeitsverlauf mit v=5-15 m/s



(c) Notbremsvorgang mit a=-4 m/s²



(d) Geschwindigkeitsprofile der Beispiele in (a,b,c) (grün, blau, rot)

Abbildung 4.21: Die Abbildungen (a) und (b) zeigen das Durchfahren einer engen Kurve mit konstanter Geschwindigkeit bzw. mit vorberechnetem Geschwindigkeitsprofil, welches aus Karteninformationen erzeugt wurde. In Abbildung (c) ist ein Notbremsvorgang zu sehen, der bei fehlgeschlagener Planung eingeleitet wird. Die in der Planung verwendeten Geschwindigkeitsverläufe sind in Abbildung (d) aufgetragen.

Abbildung 4.21 (d) zeigt Geschwindigkeitsprofile, die einerseits aus der Verhaltenssteuerung vorgegeben werden und andererseits fest in die Bahnplanung integriert sind. Letztere

Möglichkeit dient dazu, in Notfallsituationen das Fahrzeug schnell kollisionsfrei zum Stehen zu bringen. Als Fallback-Lösung können solche Notfallprofile, die sehr schnell zu berechnen sind, vor jedem Planungszyklus durchgeführt werden. Reguläre Geschwindigkeitsprofile werden von der Verhaltenssteuerung, genauer dem Fusionsknoten *Geschwindigkeit fusionieren*, vorgegeben (vgl. Kapitel 3). Anhand der Abbildungen 4.21 (a) und (b) wird deutlich, dass der Planer auf Hintergrundinformationen über den Straßenverlauf angewiesen ist, um im Beispiel die enge Kurve durchfahren zu können.

Ein alternativer Ansatz ist die Hinterlegung einzelner Beschleunigungsprofile. Ähnlich wie bei den vorgefertigten Teilsegmenten des Pfades (vgl. Abbildung 4.18) würde davon ausgegangen, dass ein Beschleunigungs- oder Abbremsvorgang eine gewisse Kontinuität besitzt und meist über einen längeren Zeitraum erfolgt. Eine Implementierung und Überprüfung dieses Ansatzes war aus Zeitgründen in dieser Arbeit nicht mehr möglich. Die in diesem Abschnitt vorgestellte Bahnplanungsvariante der Gefahren-Minimierung wurde für die abschließenden Versuche in Kapitel 6 verwendet.

4.6 Bahnplanung für die Urban Challenge 2007

Der in Abschnitt 4.4.1 beschriebene Algorithmus wurde für die Urban Challenge 2007 in Zusammenarbeit mit dem Institut für Mess- und Regelungstechnik (MRT) der Universität Karlsruhe weiterentwickelt [Ziegler 08] und dort für die Navigation in unstrukturierter Umgebung eingesetzt. Eine Aufgabe war das Durchfahren sogenannter *Zonen* und die Ausführung von Einparkmanövern an gekennzeichneten Stellen. Diese Bereiche konnten bis zu 200×200 m groß sein und sowohl statische als auch dynamische Hindernisse enthalten. Abbildung 4.22 zeigt eine solche Zone mit Parkmarkierungen.

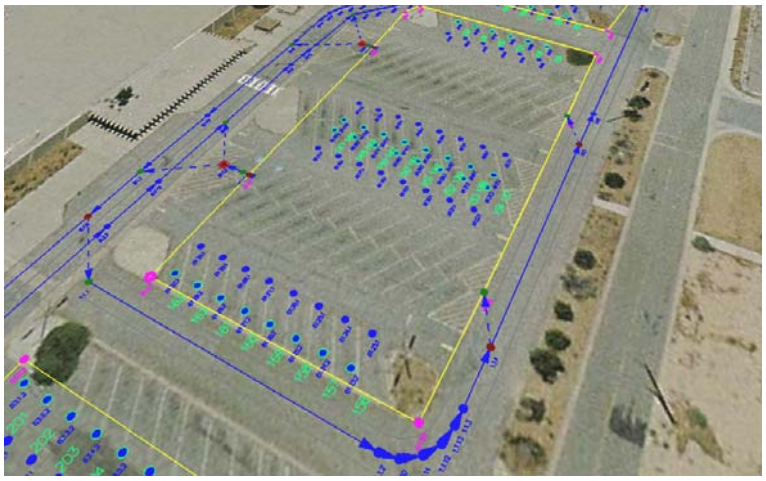


Abbildung 4.22: Das Einparken auf Markierungen in sogenannten *Zonen* (gelb umrandet) war eine Aufgabe bei der Urban Challenge 2007.

Bei Verwendung einer Schätzfunktion vom Typ *Kürzester Weg* (*lokale* Schätzfunktion) werden Hindernisse durch ein Austesten verschiedener Kombinationen umfahren, vergleichbar mit einem Auffüllen des Konfigurationsraumes um das Hindernis. Je nach Anzahl und Anordnung der Hindernisse sowie der geplanten Streckenlänge kann eine Suche mit dieser Heuristik sehr aufwändig sein. Für diesen Anwendungsfall wurde deshalb eine zweite Schätzfunktion verwendet, welche mithilfe von Voronoi-Linien die verbleibende Strecke zum Ziel errechnet und so Hintergrundinformationen über vorhandene Hindernisse berücksichtigt. Abbildung 4.23 zeigt einen aus einer Belegtheitskarte generierten Voronoi-Graphen.

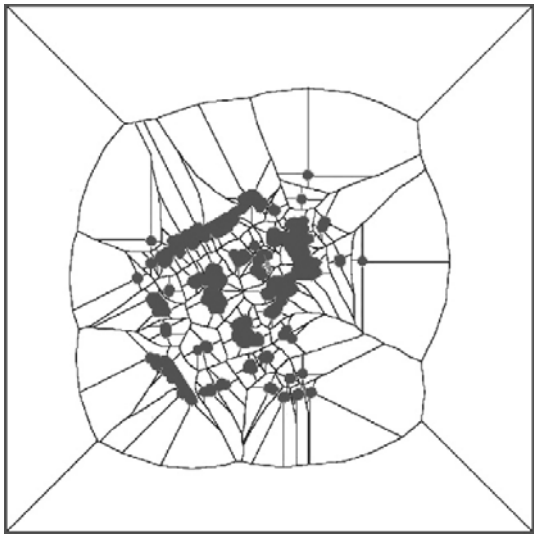
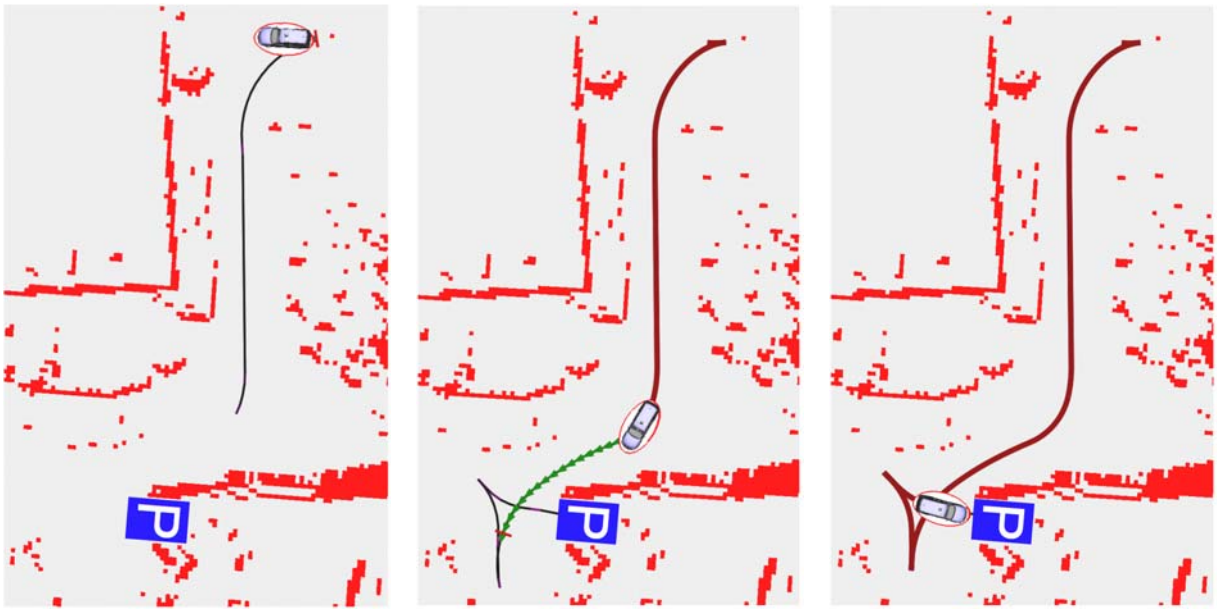


Abbildung 4.23: Voronoi-Graph, erstellt aus Umgebungsinformationen einer Belegtheitskarte. Voronoi-Kanten dienen der besseren Abschätzung tatsächlicher Wegstrecken bei der Berechnung langer Pfade.

Für die Abschätzung mittels der Voronoi-basierten Kostenfunktion wurden zunächst Start- und Zielpunkt auf den Voronoi-Graphen projiziert, und dann mittels einer Dijkstra-Suche der kürzeste Weg zwischen beiden Punkten entlang des Graphen bestimmt. Auch wenn durch die Verwendung des Fahrzeugmodells zur Erzeugung des Graphen auch hier ein fahrbarer Pfad resultierte, so war durch das Matching-Verfahren prinzipbedingt nur eine Annäherung an den Zielpunkt, aber kein Erreichen desselben möglich. Mit der Voronoi-Abschätzung kann zu Beginn einer Fahrt über weite Strecken ein guter Anfangspfad berechnet werden. Für die genaue Planung in einen Zielpunkt abseits der Voronoi-Linien ist jedoch bei Annäherung an den Zielpunkt eine Planung mit lokaler Schätzfunktion notwendig. Abhängig von der Entfernung des Zielpunktes wurde die Planung mit einer der beiden Schätzfunktionen durchgeführt. Ein Zusammensetzen des Pfades aus mehreren Planungsschritten ist bei einer Entfernung von mehr als 40 m ohnehin notwendig, da ausserhalb dieser Entfernung kaum Sensorinformationen vorliegen.



(a) (b) (c)

Abbildung 4.24: Suchalgorithmus mit verschiedenen Schätzfunktionen: Für weite Strecken mit Hindernissen liefert eine Abschätzung auf Basis eines Voronoi-Graphs zunächst schnellere Ergebnisse (a), bei Annäherung an das Ziel findet eine lokale Kostenfunktion genauere Lösungen (b,c).

Abbildung 4.24 zeigt Ergebnisse zu verschiedenen Zeitpunkten: Während zu Beginn (a) zunächst die Voronoi-basierte Abschätzung verwendet wurde, so liefert zu einem späteren Zeitpunkt bei Annäherung an das Ziel die lokale Schätzfunktion die gewünschte Lösung (b,c). Die aus den verschiedenen Planungsschritten resultierenden Teilpfade werden zu einer Gesamtlösung kombiniert. Neben der Verwendung für Einparkaufgaben in Zonen wurde der Planer auf der Urban Challenge ausserdem verwendet, um Dreipunktwenden durchzuführen oder um in undurchsichtigen Kreuzungssituationen über einen *Recovery-Modus* eine alternative Route zu finden.

Kapitel 5

Regelung

5.1 Einführung

Zur Umsetzung der von der Bahnplanung erzeugten Vorgaben wurde ein Regelungsmodul entwickelt, das die für die Aktorik notwendigen Stellgrößen berechnet. Zunächst wird in Abschnitt 5.2 das Regelungskonzept vorgestellt und die Schnittstelle zu höheren Ebenen definiert. Als Versuchsplattform für reale Fahrversuche dient am IAIM ein Smart Roadster. Dieses Fahrzeug wurde von der Smart AG als Testfahrzeug ausgemustert und dem Institut im Serienzustand zur Verfügung gestellt. Die Hardware wird, vor dem Hintergrund eines notwendigen Umbaus auf Drive-by-Wire, in Abschnitt 5.3 erklärt.

Das Fahrzeug wurde im Rahmen dieser Arbeit mit Schnittstellen versehen, um PC-gesteuert auf Lenkungs-, Brems- und Motorsysteme zugreifen zu können und interne Sensordaten auszulesen. Dem Umbau und damit verbundenen Sicherheitsfragen widmet sich Abschnitt 5.4. Auf die einzelnen Systeme zur Längs- und Querregelung mit unterlagerten Regelkreisen gehen die Abschnitte 5.5 und 5.6 näher ein. Für die Umsetzung der Regelung kommt das *Modular Controller Framework MCA2* zum Einsatz. Dieses Framework bietet eine echtzeitfähige Ausführung von Algorithmen durch Nutzung eines Linux-Kernel-Patches, Kommunikationsroutinen für CAN-Bus und Ethernet und darüber hinaus nützliche Werkzeuge zum Visualisieren von Messdaten zur Laufzeit. Die ebenfalls am FZI/IDS entwickelten *Universal Controller Module (UCOM)* sind in dieses System integriert, so dass sich dedizierte Aufgaben leicht auf diese Mikrocontroller auslagern lassen.

Im Rahmen des Sonderforschungsbereiches wird die Fahrzeugregelung im Teilprojekt ZT3 „Invariantes Pfad- bzw. Trajektorientracking“ umgesetzt. Der Vollständigkeit halber sei an dieser Stelle auf laufende Arbeiten insbesondere zu flachheitsbasierten Regelungen verwiesen [Werling 08a],[Werling 08b],[Werling 08c]. Parallel zu den Arbeiten am Smart Roadster wurde innerhalb des Sonderforschungsbereiches ein VW Passat für autonomes Fahren ausgerüstet. Die Schnittstellen und Softwaresysteme sind identisch, der Passat besitzt jedoch eine umfangreichere Sensorausstattung und wurde deshalb für die abschließenden Versuche in Kapitel 6 verwendet.

5.2 Regelungskonzept

Hauptaufgabe der Regelungskomponente ist die Umsetzung der vom Bahnplanungsmodul erzeugten Bahn oder Trajektorie. Die notwendigen Stellgrößen für einzelne Regelkreise sollen dabei vom Regelungsmodul selbst errechnet werden, so dass die vollständige Bahn die Eingangsgröße bildet, nicht etwa ein Geschwindigkeitsprofil mit Lenkwinkelverlauf. Damit wird die Dynamik in der Schnittstelle verringert und es werden zusätzliche Aufgaben in das Regelungsmodul verlagert. Bahnverläufe können so in größeren Abständen vorgegeben werden als etwa Lenkwinkelverläufe. Weiterhin lassen sich Sicherheitsbedingungen integrieren, wie z. B. ein Anhalten am Ende der Bahn.
Zur Gewährleistung von Kontinuität in den Lenkwinkelstellungen eignen sich kubische Polynome, welche darüber hinaus sehr einfach zu berechnen sind:

$$s_j(x) = a_j + b_j \cdot x - x_j + d_j \cdot x - x_j{}^2 + d_j \cdot x - x_j{}^3 \tag{5.1}$$

Um diese Polynome numerisch einfach handhaben zu können, werden sie in Form von kubischen B-Spline-Kurven verwendet. Die vom Bahnplanungsmodul übermittelten Punkte P_k (vgl. Abbildung 5.1) dienen als Kontroll- oder Ausgleichspunkte für einen kubischen B-Spline der Form

$$C(u) = \sum_{k=0}^{n} P_k B_{k,4}(u), n \geqslant 3, \tag{5.2}$$

wobei u den Knotenvektor darstellt, $n+1$ die Anzahl der Kontrollpunkte und $B_{k,4}$ die Gewichtsfunktionen – in diesem Fall Polynome vom Grad 3. Die Gewichtsfunktionen (Basisfunktionen) werden rekursiv nach der de-Boor-Formel berechnet [Dierckx 93]:

$$\begin{aligned} B_{k,1} &= \begin{cases} 1 & : \text{wenn } u_k \leqslant u \leqslant u_{k+1} \\ 0 & : \text{sonst} \end{cases} \\ B_{k,d} &= \frac{u - u_k}{u_{k+d-1} - u_k} B_{k,d-1}(u) + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}} B_{k+1,d-1}(u) \end{aligned} \tag{5.3}$$

Von dieser Ausgleichsfunktion werden Vorgaben für die Querregelung abgeleitet. Die Geschwindigkeit als Vorgabe für die Längsregelung kann von Seiten der Bahnplanung weniger detailliert vorgegeben werden. So reicht es aus, eine Richtgeschwindigkeit v_d zu übermitteln, welche dann von der Längsregelung mit der Beschleunigung a_d ausgeregelt oder unter Berücksichtigung des aktuellen Kurvenverlaufs gegebenenfalls reduziert wird. Als Randbedingung dafür dient die maximal zulässige Querbeschleunigung $a_{lat,max}$. Oftmals unnötige Komplexität, wie sie die Vorgabe eines exakten Geschwindigkeitsprofils erfordern würde, lässt sich damit aus der Bahnplanung fernhalten. Falls das Bahnplanungsmodul jedoch exakte Bewegungen vorgeben möchte, kann das Profil aus Geschwindigkeitsvorgaben der Einzelpunkte erzeugt werden.

Im Straßenverkehr muss oftmals an Haltelinien sehr positionsgenau angehalten werden. Für Vorgaben dieser Art kann zusätzlich zu den Ausgleichspunkten für die Spline-Funktion eine Anhaltedistanz übermittelt werden. Um diese Vorgaben nicht nur zum Anhalten, sondern z.B. auch zum Folgen eines anderen Fahrzeuges verwenden zu können, soll allgemein eine

Frontlinie im Abstand d_F mit Geschwindigkeit v_F dienen (vgl. Abbildung 5.1). Diese darf nicht überfahren werden und ist neben dem letzten verfügbaren Ausgleichspunkt ein weiteres Kriterium, um die Längsregelung mit Vorgaben zu versorgen und gegebenenfalls automatisch die Geschwindigkeit zu reduzieren.

Da dieses Modul Zugriff insbesondere auf Gas und Lenkung des Fahrzeuges hat, muss hier die Sicherheit für andere Verkehrsteilnehmer, und auch für Entwickler, in besonderem Maße gewährleistet werden. Dies umfasst eine Absicherung auf verschiedenen Ebenen gegenüber einem Ausfall von Komponenten, Kommunikations-, Daten- und Bedienerfehlern.

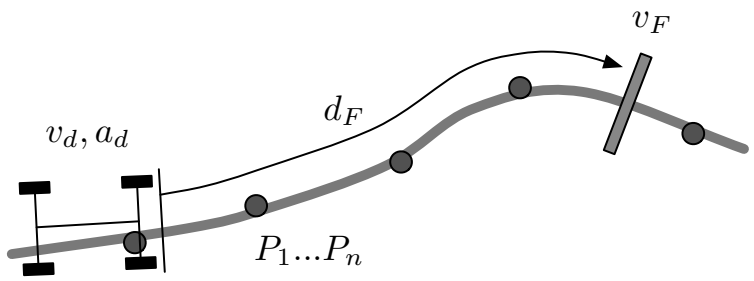


Abbildung 5.1: Eingangsdaten für das Regelungsmodul: Die Ausgleichspunkte P_k definieren eine B-Spline-Kurve, diese dient in erster Linie für Vorgaben der Querregelung. Die Richtgeschwindigkeit v_d und die Richtbeschleunigung a_d werden von der Längsregelung umgesetzt sofern dies mit der zulässigen Querbeschleunigung $a_{lat,max}$ vereinbar ist. Besitzen die Ausgleichspunkte selbst Geschwindigkeiten, so werden daraus Vorgaben für die Längsregelung abgeleitet. Als weitere Einschränkung für die Längsregelung dienen die Position d_F und die Geschwindigkeit v_F einer Frontlinie, die nicht überfahren werden darf.

Weiterhin wird das Regelungsmodul als einzige Komponente Zugriff auf die Hardware des Fahrzeugs haben. Alle Informationen, die von anderen Modulen benötigt werden, müssen in gewissen zeitlichen Abständen bereitgestellt werden. Mit diesen Vorüberlegungen lassen sich die an das Regelungsmodul gestellten Anforderungen und Aufgaben wie folgt definieren:

- Umsetzung von Sicherheit für Verkehrsteilnehmer und Entwickler.
- Abfahren eines in Form von Ausgleichspunkten vorgegebenen B-Splines.
- Umsetzung von Geschwindigkeitsvorgaben (Tempomatfunktion).
- Folgen (oder Anhalten an) einer Frontlinie.
- Versorgung höherer Ebenen mit Sensordaten und Statusinformationen.

Diese Anforderungen werden in den nächsten Abschnitten schrittweise umgesetzt.

5.3 Versuchsplattform *Smart Roadster*

Ein von der Fa. Smart zur Verfügung gestellter Smart Roadster mit Automatikschaltung und 45 kW Leistung dient als Versuchsplattform für autonomes Fahren am IAIM (vgl. Abbildung 5.2). Als Fahrzeug der neueren Generation ist der Smart Roadster serienmäßig mit Anti-Blockier-System (ABS) und Elektronischem Stabilitäts-Programm (ESP) ausgerüstet. Diese Fahrerassistenzsysteme benötigen eine Vielzahl von Informationen wie Fahrgeschwindigkeit, Gaspedalstellung, Motordrehzahl, Gierrate, Querbeschleunigung, aktueller Lenkwinkel und Bremsdruck, so dass die zur Messung dieser Größen notwendige Sensorik bereits vorhanden ist. Zumindest ein Teil dieser Daten wird über einen CAN-Bus zwischen den Komponenten kommuniziert und ist deshalb einfach zugänglich. Die Systeme selbst sollen für Versuchsfahrten aktiv bleiben und stören nicht, da nicht in Grenzbereichen getestet wird. Die Verwendung interner Sensordaten ist elementar und erübrigt größtenteils den Anbau eigener Sensorik zur Messung der genannten Größen.

Abbildung 5.2: Versuchsplattform Smart Roadster (Baujahr 2003).

Als Gaspedal wird ein sogenannter elektronischer Pedalwertgeber verwendet (E-PWG), welcher im Grunde lediglich aus einem Potentiometer besteht und recht problemlos manipuliert werden kann. Die Servolenkung ist nicht, wie sonst üblich, über eine Hydraulik-Unterstützung realisiert, sondern über eine rein elektromechanische Lenkung. Die geringe Achslast des Fahrzeuges macht dies auch mit 12 V Bordspannung möglich. Um die notwendige Unterstützung zu bestimmen, nehmen in der Lenksäule Momentensensoren das vom Fahrer aufgebrachte Lenkmoment auf und bewirken eine geschwindigkeitsabhängige Unterstützung des Servosystems.

Ein ACC-System ist leider nicht vorhanden. Damit entfällt die Möglichkeit elektronisch zu bremsen, so dass diese Aufgabe anderweitig gelöst werden muss. Ein Bremsdrucksensor an der ABS/ESP-Einheit ist allerdings zugänglich, um das Signal analog abzugreifen. Als Stromversorgung steht eine Lichtmaschine mit 900 W zur Verfügung, die zur Versorgung von zahlreichen (im Versuchsbetrieb nicht notwendigen) Zusatzkomponenten wie Nebelleuchten und Sitzheizung für ein Fahrzeug dieser Größe ausreicht.

5.4 Umbau und Fahrzeugbereitstellung

5.4.1 Sicherheitskonzept

Beim autonomen Betrieb eines Straßenfahrzeuges kann aufgrund von Masse und Motorleistung erheblicher Schaden für Personen und Material entstehen. Es liegt daher in der Pflicht der Entwickler und Betreiber, durch geeignete Maßnahmen die Sicherheit so gut wie möglich zu gewährleisten – eine spätere Abnahme durch Sachverständige erfolgt in diesem Fall nicht. Eine wiederholte Straßenzulassung wurde bereits von Seiten der Smart AG bei Bereitstellung des Fahrzeuges ausgeschlossen. Einige der Sicherheitsfragen sind bereits beim Fahrzeugumbau zu berücksichtigen und werden deshalb an dieser Stelle diskutiert.

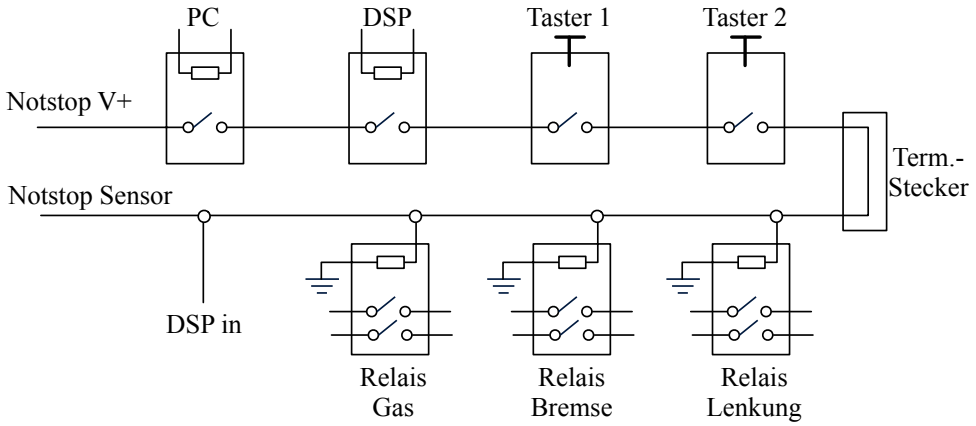


Abbildung 5.3: Eine Sicherheitskette aktiviert alle Komponenten mit Zugriff auf die Fahrzeugsteuerung. Nur bei stromführender Leitung *Notstop Sensor* ist eine automatische Fahrzeugführung möglich. Die Sicherheitskette kann durch Notaus-Schalter, PC oder DSP (UCOM Modul) unterbrochen werden.

Kernpunkt des Sicherheitskonzeptes ist die Forderung nach einer Möglichkeit, alle durch einen Rechner getätigten Eingriffe in die Fahrzeugsteuerung schlagartig zu unterbinden und die Kontrolle einem Sicherheitsfahrer zu übergeben. Um diese Funktionalität umzusetzen, wird über alle Schnittstellenkomponenten mit direktem Zugriff in das Fahrgeschehen eine Sicherheitskette geführt. Diese trennt im Regelfall die Schnittstellenleitungen und leitet nur im Fall einer korrekt geschlossenen Kette die Steuerleitungen des Rechners an die Fahrzeugsteuerung weiter (vgl. Abbildung 5.3). Die Sicherheitskette kann rein elektromechanisch

durch Notaus-Schalter unterbrochen oder auch rechnergestützt durch PC oder DSP (UCOM Modul) ausgelöst werden. Eine Unterbrechung wird vom Rechnersystem erkannt, um einen Stop-Zustand herzustellen und Stellgrößen zurückzusetzen.

Zusammenfassend wird die Sicherheit durch folgende Maßnahmen gewährleistet:

- **Sicherheitsfahrer**
  Ein mit dem Fahrzeug sehr gut vertrauter Fahrer übernimmt bei Testfahrten die Verantwortung für einen sicheren Betrieb. Er dient nur diesem Zweck und beschäftigt sich ausser mit den Fahraufgaben nicht mit den laufenden Tests.

- **Sicherheit auf Hardware-Ebene**
  Über eine Sicherheitskette ist es jederzeit möglich, die Verbindungen zum Rechnersystem zu trennen und dem Fahrer die Kontrolle zurückzugeben. Elementar wichtige Schnittstellen wie jene zur Gas-Steuerung (E-PWG) werden dabei doppelt abgesichert.

- **Sicherheit auf Software-Ebene**
  Eine Plausibilitätsprüfung wird für Eingangswerte durchgeführt – für die Ausgangswerte erfolgt eine Stellgrößenbegrenzung. Über die Drehmomentensensoren im Lenkrad kann ein Soft-Notstop realisiert werden, der durch ein Gegenhalten des Sicherheitsfahrers ausgelöst wird.

Sicherheit auf Software-Ebene garantiert keine Verbesserung der absoluten Sicherheit und dient in erster Linie dazu *harte* Notstops zu vermeiden, um effizienter entwickeln zu können. Viele Fehler lassen sich schwer überprüfen falls sie nicht einfach reproduziert werden können. Eine komplette Notabschaltung, mit Auswirkungen auf den Zustand des Regelsystems, ist dazu eher hinderlich. Die Überwachung der Kommunikation zwischen PC und DSP von Seiten der DSP-Plattform kann dazu beitragen, etwaige Probleme auf PC-Seite früh zu erkennen und zu warnen, bevor ein unangemessenes Fahrzeugverhalten resultiert. Das Ausbleiben von Eingangsdaten wie bspw. Bahnpunkten deutet auf einen solchen Fehler hin.

Erfahrungsgemäß ergeben sich aufgrund der komplexeren und umfangreicheren Software auf dem PC eher Probleme als auf dem DSP mit nur verhältnismäßig wenig Quellcode. Eine Überwachung in dieser Richtung erscheint deshalb sinnvoll. Weniger sicherheitskritisch, und eher aus Komfort-Sicht motiviert, ist eine *sanfte* Umschaltung von manuellem nach automatischem Betrieb. Eine vorherige Stellgrößenanpassung kann dazu beitragen ruckfrei umzuschalten.

5.4.2 Drive-by-Wire-Umsetzung

Abbildung 5.4 gibt einen Überblick über die mit dem Ziel durchgeführten Veränderungen, Zugang zur Fahrzeugsteuerung zu bekommen. Die internen Sensoren werden entweder über den vorhandenen Motor-CAN-Bus oder über separate AD-Wandler ausgelesen. Zusätzlich verbaute Komponenten sind an einem zweiten CAN-Bus, dem sogenannten Control-CAN-Bus, angeschlossen. Parallel zu diesem Bus wird die Sicherheitskette mitgeführt, welche von Handschalter, PC und DSP unterbrochen werden kann. Steuerungsrechner und DSP-Platine

sind im Stauraum des Fahrzeuges verbaut. Für die Ansteuerung der Bremse wurde ein zusätzlicher Aktuator eingebaut, alle anderen Schnittstellen sind rein elektronischer Natur.

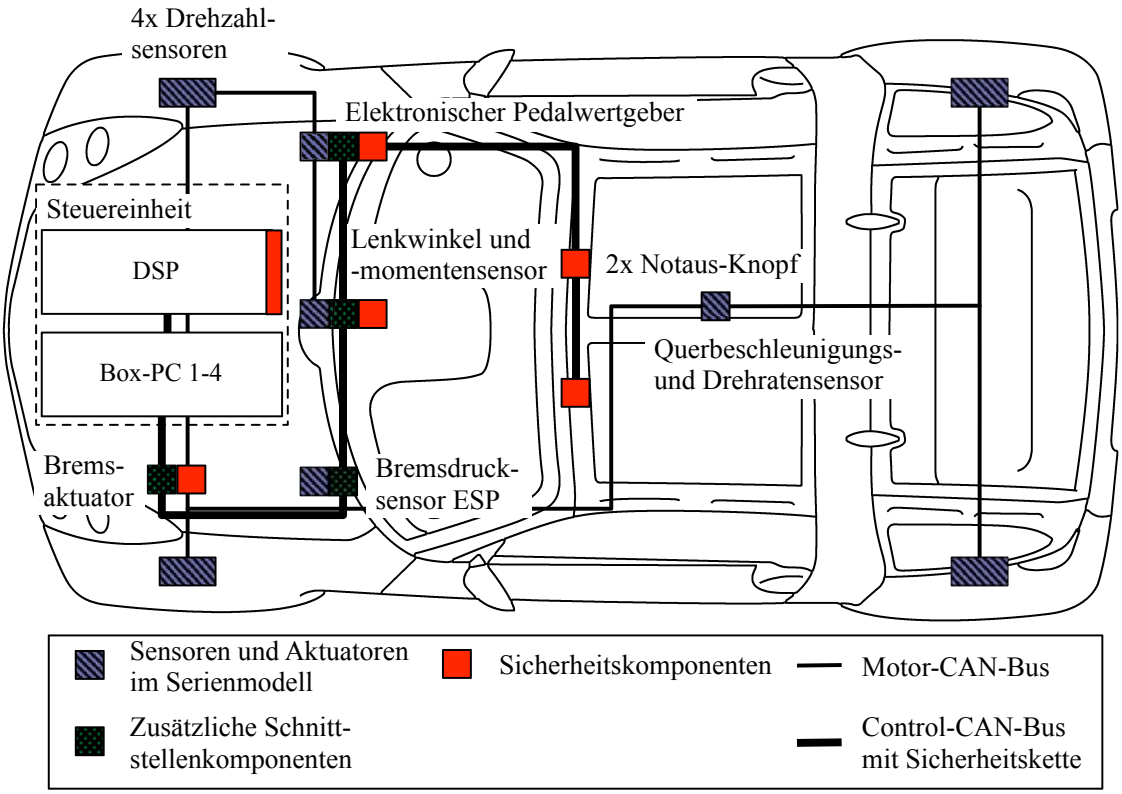


Abbildung 5.4: Übersicht der eingebauten Sensoren, Aktuatoren und Sicherheitseinrichtungen im Fahrzeug. Die Kommunikation zwischen PC, DSP sowie Peripheriegeräten erfolgt über mehrere CAN-Busse. Die Sicherheitskette wird zusammen mit dem Control-CAN-Bus geführt.

Elektronischer Pedalwertgeber

Wie in der Einleitung bereits angesprochen wird im Versuchsfahrzeug ein elektronischer Pedalwertgeber (E-PWG) verwendet. Die in diesem Geber von zwei Potentiometern erzeugten Spannungen können alternativ über einen CAN-Bus-Knoten (CAN-CBX-AO412 [ESD 08]) erzeugt werden, der als DA-Wandler fungiert. Die erzeugten Vorgaben werden über ein Relais in der Sicherheitskette doppelt abgesichert. Ein zusätzlicher AD-Wandler (CAN-CBX-AI814 [ESD 08]) erlaubt es, die Werte des E-PWGs jederzeit auszulesen. Dies hat den Vorteil, dass auch im autonomen Betrieb Vorgaben des Sicherheitsfahrers die aktuellen Regler-Sollwerte überlagern können, und der Regler damit bei unzureichender Gaspedalstellung übersteuert werden kann. Es ist zudem möglich, einen teil-autonomen Testbetrieb zu fahren, bei welchem die Querregelung autonom arbeitet, die Längsführung aber regulär vom Fahrer durchgeführt wird (vgl. auch [Schröder 06]).

Bremsaktuatorik

Obwohl das ESP-System grundsätzlich in der Lage ist einzelne Räder abzubremsen, so reicht die Bremswirkung nicht aus, um das Fahrzeug stark zu verzögern. Für eine voll funktionsfähige Bremsregelung muss demnach zusätzliche Aktuatorik verwendet werden. Eine Modifikation des Hydrauliksystems ist sehr aufwändig und hätte Auswirkungen auf die bestehenden ESP- und ABS-Systeme. Diese sollen jedoch nach wie vor als Black-Box-Systeme genutzt werden, weshalb eine konventionelle Lösung bevorzugt wird. Diese besteht darin, das Bremspedal direkt zu betätigen. Eine vom Fahrer ausgeübte Bremskraft darf durch die Aktuatorik zu keiner Zeit blockiert werden, so dass nur eine flexible Zugkraftübertragung bspw. in Form eines Stahlseiles in Frage kommt. Zur Ansteuerung dient ein bürstenbehafteter DC-Motor, wie er auch zur Leuchtweitenregulierung in Kraftfahrzeugen zum Einsatz kommt. Abbildung 5.5 zeigt schematisch den Aufbau des Bremssystems.

Eine Motorendstufe Maxon ADS_E 50/10 übernimmt die Ansteuerung des Bremskraftverstärkers [MM 08]. Aus Gründen der Zuverlässigkeit soll das System möglichst einfach und robust gestaltet werden. Auf Inkrementalgeber wird deshalb verzichtet, einzig ein magnetischer Endabschalter dient der Wegbegrenzung. Diese Absicherung wird notwendig, falls zeitgleich manuell gebremst wird und der Regler sonst zu weit gegensteuern würde. Die Ansteuerung des Motors erfolgt über i×R-Kompensation. Damit wird die Motorspannung proportional zum angelegten Sollwert ausgegeben und bei steigendem Motorstrom als Folge einer Beanspruchung nachgeführt. De facto ermöglicht dies eine vereinfachte Drehzahlstellung ohne Verwendung eines Sensors.

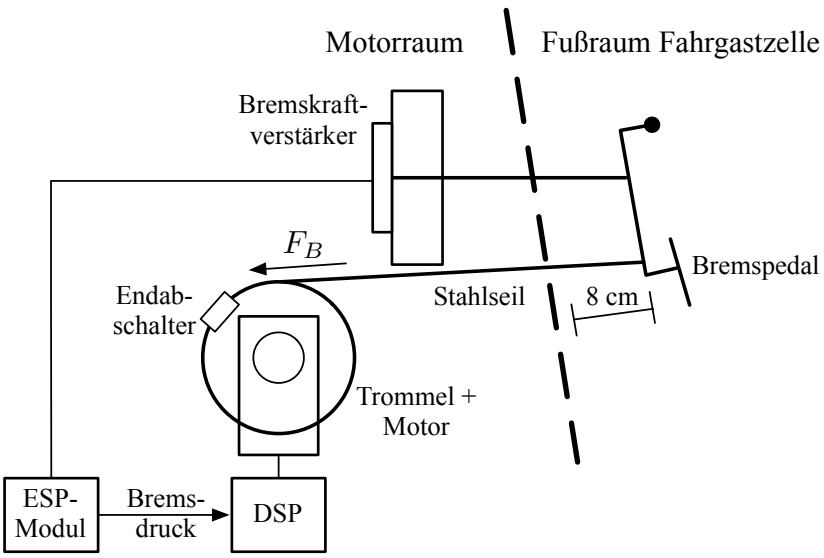


Abbildung 5.5: Schematische Darstellung der Bremsaktuatorik. Ein geforderter Maximalbremsdruck von 50 Bar verlangt eine Seilkraft von $F_B = 200\,\mathrm{N}$, bzw. bei ca. 10 cm Trommeldurchmesser ein Moment von 10 Nm auf der Motorwelle. Der Bremsdrucksensor des ESP-Systems schließt den Kreis für eine Bremsdruckregelung (vgl. Abschnitt 5.5.2).

Lenkungsansteuerung

Im Abschnitt zur Fahrzeug-Hardware wurde bereits das elektromechanische Servosystem zur Lenkunterstützung angesprochen. Bei der Ansteuerung macht man sich die Verbindung zwischen Momentensensoren in der Lenksäule und Lenkungssteuergerät zu Nutze. Über eine Zwischenschaltung wird diese Verbindung getrennt, so dass im autonomen Modus rechnerseitig ein virtuelles Handmoment erzeugt und damit eine Unterstützung angefordert werden kann. Die Umschaltung ist, wie auch bei Gas- und Bremsansteuerung, an die Sicherheitskette gekoppelt.

Der Motor der Lenkunterstützung ist ausreichend stark ausgelegt, um auch ohne Fahrerunterstützung das Fahrzeug lenken zu können. Einziges Hindernis sind die im Seriensteuergerät der Lenkung hinterlegten, geschwindigkeitsabhängigen Kennlinien (vgl. Abbildung 5.6 links). Bei höherer Geschwindigkeit ist der Anteil des vom Fahrer aufgebrachten Momentes ausschlaggebend und die Unterstützung fällt nahezu weg. Eine Trennung des Gerätes vom Bus und die Vorgabe einer virtuellen Geschwindigkeit $v = 0\frac{m}{s}$ würde an einer geeigneten Kennlinie festhalten. Dies stellt jedoch einen zu großen Eingriff in das zentrale Steuerungssystem dar und beeinflusst andere Geräte am CAN-Bus. Als Lösung wurde der Einbau eines modifizierten Steuergerätes mit linearer Kennlinie für alle Geschwindigkeiten vorgenommen, welches freundlicherweise von der Fa. ThyssenKrupp Presta Steering [ThyssenKrupp 08] zur Verfügung gestellt wurde.

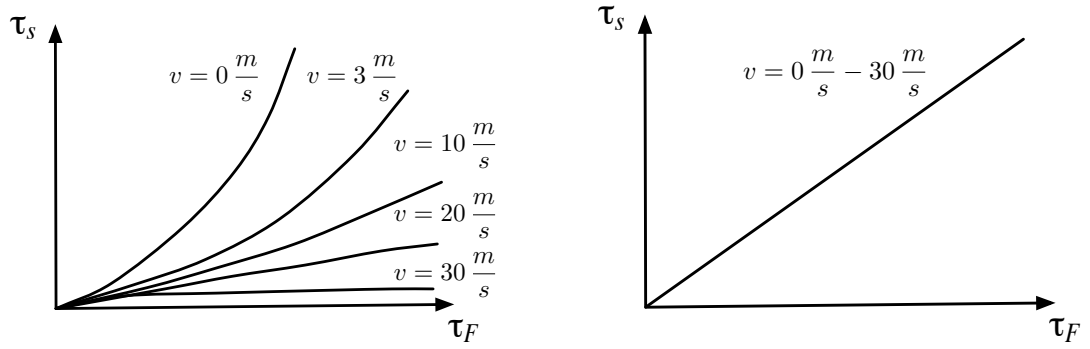


Abbildung 5.6: Schematische Darstellung der geschwindigkeitsabhängigen Unterstützung des Seriensteuergerätes der Lenkung (links) und lineare Kennlinie für alle Geschwindigkeiten im modifizierten Steuergerät (rechts). Aufgetragen ist das Lenkmoment τ_S über dem gemessenen Handmoment τ_F des Fahrers.

Die Verwendung einer linearen Kennlinie bringt den Nachteil mit sich, dass Informationen über nicht-lineare Einflüsse wie z. B. Reifenreibung und Zentrierkräfte verloren gehen und damit auch für einen implementierten Regler nicht mehr zur Verfügung stehen. Die ursprünglich verwendeten Kennlinien werden deshalb als Modell hinterlegt und bei der Stellwerterzeugung weiterhin berücksichtigt (vgl. Abschnitt 5.6). Damit wird das System für einen linearen Regler handhabbarer, was sich im Bezug auf Einregeldauer und Regelgüte vorteilhaft auswirkt.

Rechnersystem und Energieversorgung

Aus Platzgründen können keine weiteren Komponenten, wie zusätzliche Pufferbatterien oder eine zweite Lichtmaschine, zur Energieversorgung eingebaut werden. Durch Verzicht auf größere Verbraucher wie Nebelleuchten und Sitzheizung wird ein Leistungspuffer von ca. 30 A geschaffen, welcher für Rechner und Steuerungskomponenten verwendet werden kann.

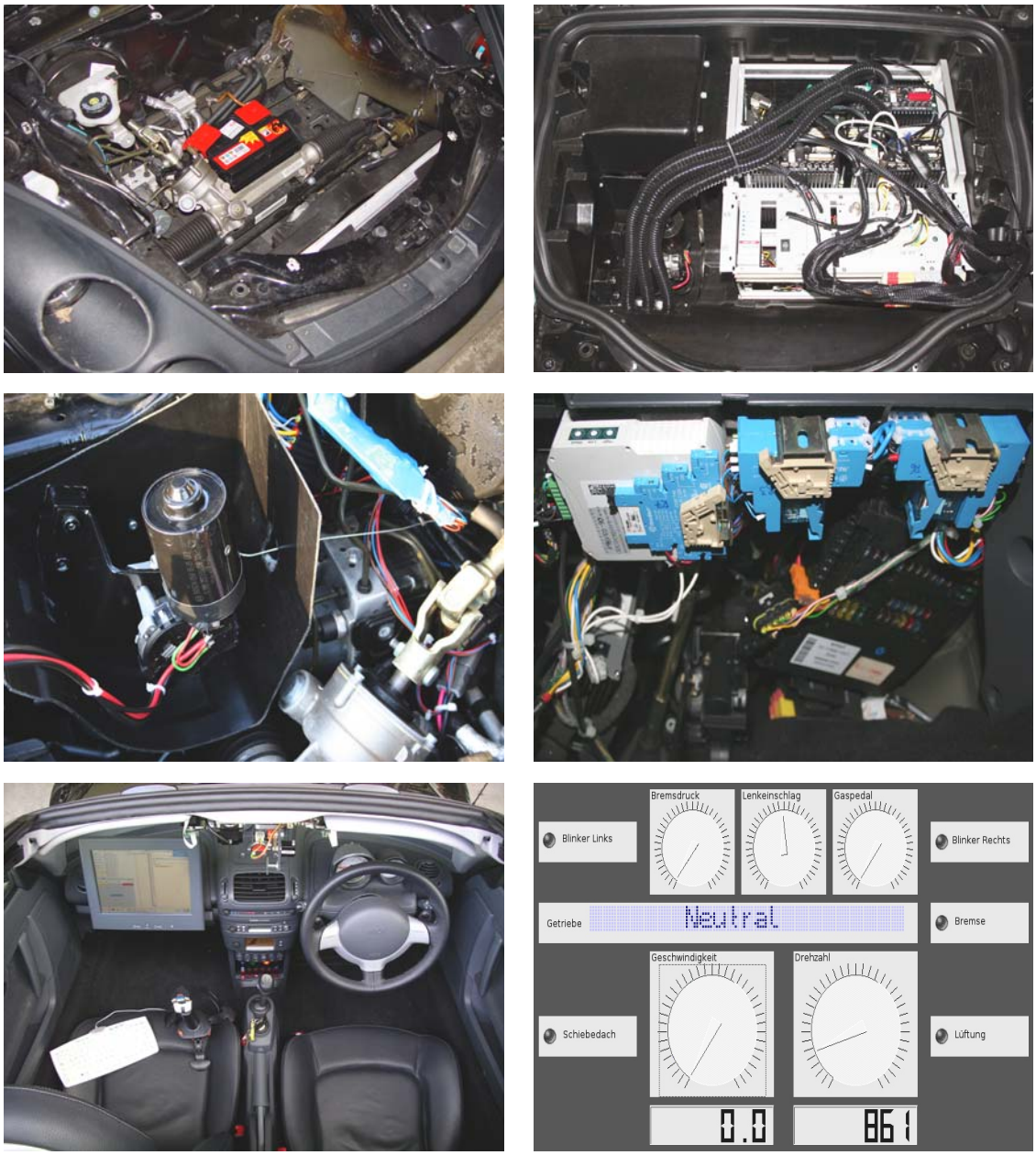


Abbildung 5.7: Ergebnisse des Fahrzeugumbaus: Einschub mit Rechnersystem, DSP und Lüftungsanlage sowie Sicherheitseinrichtungen (oben). Bremsanlage und Schnittstellenkomponenten für Gas- und Lenkungsansteuerung (Mitte). Fahrgastzelle mit Anzeigeelementen und Visualisierung der CAN-Bus-Daten (unten).

Bei der Auswahl des Rechnersystems ist zwangsläufig eine Orientierung an den strengen Beschränkungen bzgl. Stromverbrauch und Platzbedarf notwendig. So kann bspw. kein Serversystem im 19-Zoll-Format eingesetzt werden – stattdessen werden stromsparende Box-Rechner mit Pentium 1,8-GHz-Kern und einer maximalen Leistungsaufnahme von 60 W verwendet. Ein Rechner ist für die Regelung, DSP-Kommunikation und Hardware-Anbindung der CAN-Busse zuständig, drei weitere übernehmen die Aufgaben der Perzeption, Interpretation und Entscheidungsfindung. Eine WLAN-Rundstrahlantenne ermöglicht eine dauerhafte Funknetzverbindung zwischen Fahrzeug und Leitstand auf dem gesamten Testgelände. Über diese Verbindung können von mehreren Personen Messdaten eingesehen sowie Ergebnisse der Einzelkomponenten nachvollzogen werden. Abbildung 5.7 zeigt das Fahrzeug nach dem Umbau mit den wichtigsten Ergebnissen.

5.5 Längsregelung

Die Längsregelung gliedert sich in mehrere Komponenten. Ein *Verlaufsgeber* erzeugt zunächst einen zu fahrenden Geschwindigkeitsverlauf, welcher von einem *Geschwindigkeitsregler* (Tempomat) ausgeregelt wird. Letzterer bedient sich unter anderem eines unterlagerten *Bremsdruckreglers*, der die Verzögerungsvorgaben umsetzt.

Der Verlaufsgeber für die Geschwindigkeit bezieht seine Vorgaben einerseits von der B-Spline-Kurve $C(u)$, aus der sich durch Kombination mit Sollwerten für Geschwindigkeit v_d, Beschleunigung a_d und maximal zulässiger Querbeschleunigung $a_{q_{max}}$ ein Geschwindigkeitsprofil $v_C(t)$ ergibt. Andererseits liefert eine Frontlinie, welche nicht überfahren werden darf, mit Position d_F und Geschwindigkeit v_F zusätzliche Angaben (vgl. Abbildung 5.8). Aus diesen Vorgaben resultieren unterschiedliche Geschwindigkeitsverläufe, wie das mittlere Schaubild der Abbildung zeigt. Aufgabe des Längsreglers ist es, diese unter Umständen widersprüchlichen Vorgaben zu klären und ein einheitliches Geschwindigkeitsprofil zu erzeugen. Dieses wird an ein *Tempomat*-Modul weitergereicht und von selbigem ausgeregelt. Falls eine Trajektorie vorgegeben wird, so bilden die mitgelieferten Geschwindigkeiten in den Ausgleichspunkten weitere Vorgaben und werden in das Profil von $v_C(t)$ übernommen.

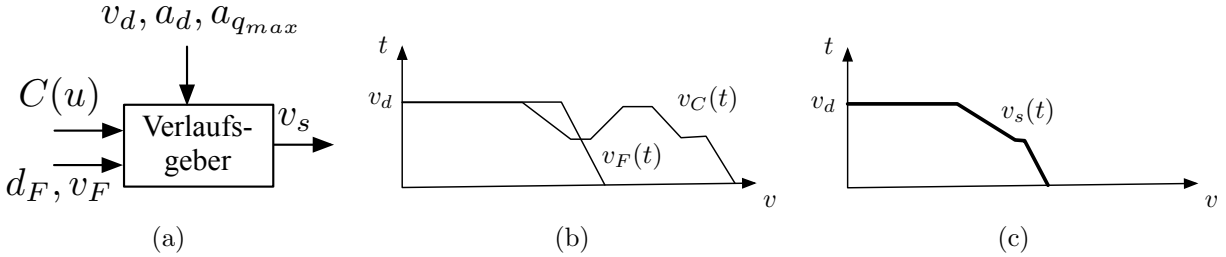


Abbildung 5.8: Ein Verlaufsgeber (a) fusioniert die verschiedenen Vorgaben, indem zunächst einzelne Geschwindigkeitsverläufe $v_C(t)$ und $v_F(t)$ für Bahnkurve (oder Trajektorie) sowie für die Frontlinie erstellt werden (b). Der zu fahrende Verlauf ergibt sich durch Minimumsfusion der Einzelprofile (c).

Der Geschwindigkeitsverlauf $v_C(t)$ der Kurve wird dabei maßgeblich durch die zulässige Querbeschleunigung $a_{max,lat}$ bestimmt, aber auch durch die Notwendigkeit, spätestens am letzten Punkt der Kurve anzuhalten. Das Geschwindigkeitsprofil $v_F(t)$ hängt von der übermittelten Front ab und kann z. B. eine Haltelinie definieren, welche weit vor dem Ende der Kurve liegt (vgl. Abbildung 5.8).

Das zu fahrende Profil ergibt sich durch Minimumsfusion der beiden Profile zu

$$v_s(t) = min(v_F(t), v_C(t)) \tag{5.4}$$

5.5.1 Geschwindigkeitsregler (Tempomat)

Aufgabe des Geschwindigkeitsreglers ist eine zuverlässige Umsetzung der vom Längsregler vorgegebenen Geschwindigkeit v_s. Äußere, teils unbekannte Störgrößen – in Kombination mit nichtlinearen Elementen wie Antriebsstrang und Motorkennfeld – erschweren die Regelung der Geschwindigkeit. Im Folgenden soll ein Versuch unternommen werden, diese Einflüsse in Form von empirisch ermittelten Modellen, oder falls möglich durch Messung zur Laufzeit, im Regelkreis zu berücksichtigen. Abbildung 5.9 zeigt die Geschwindigkeitsregelstrecke mit wichtigen Komponenten.

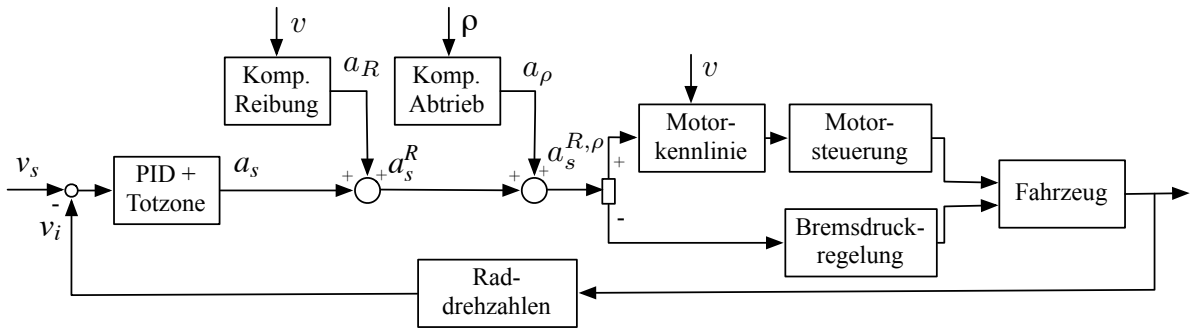


Abbildung 5.9: Schematische Darstellung des Geschwindigkeitsregelkreises mit Totzonenregler und empirisch ermittelten Modellen zur Kompensation von Reibungs- und Abtriebsbeschleunigungen. Es erfolgt eine Aufteilung des Regelkreises zur Vorgabe des korrigierten Sollwertes $a_s^{R,\rho}$ entweder an die Motorsteuerung ($a_s^{R,\rho} \geqslant 0$) oder an die Bremsdruckregelung ($a_s^{R,\rho} < 0$). Die direkte Messung der Beschleunigung ist nicht möglich, weshalb auf einen separaten Beschleunigungsregelkreis verzichtet wird. Die Ausführung des Geschwindigkeitsreglers erfolgt mit einer Periode von 100 ms.

Als Regelungsalgorithmus dient ein herkömmlicher PID-Regler, der um eine einseitige Totzone und Notstop-Funktionalität erweitert wurde. Die Berechnung der Sollbeschleunigung a_s erfolgt nach folgender Vorschrift, wobei die angegebene Reihenfolge eingehalten wird:

$$a_s = \begin{cases} u_{PID}(\Delta v) & \\ 0 & (|v_s - v_i| < v_{gap}, v_i < v_s, v_{i,hist} = v_s) \\ a_{max,brems} & (v_i - v_s > 0,1 \cdot v_s) \end{cases} \tag{5.5}$$

Zunächst wird der PID-Stellwert in Abhängigkeit der Regeldifferenz berechnet. Anschließend erfolgt eine Überprüfung der Totzone. Es erfolgt betragsmäßig keine Beschleunigungsvorgabe, falls sich die Istgeschwindigkeit in einem Bereich v_{gap} unterhalb der Sollgeschwindigkeit befindet, und sie diese zuvor erreicht hatte (also nur im Falle einer Annäherung von oben). Durch die letzte Bedingung $v_{i,hist} = v_s$ ergibt sich diese Hysterese, wodurch ein häufiges Umschalten am Totzonenende vermieden wird. Oberhalb der Sollgeschwindigkeit besteht keine Totzone, so dass notwendige Verzögerungen sofort umgesetzt werden. Die ermittelte Stellgröße kann durch die Notbremsbedingung wiederum überschrieben werden. Diese liegt vor, falls die Istgeschwindigkeit die Sollgeschwindigkeit um mehr als 10% derselben übersteigt. Durch hier nicht weiter erläuterte Maßnahmen, wie bspw. *Anti-Wind-Up* und PT_2-Filter für sprungfreie Übergänge, wird die praktische Verwendbarkeit sichergestellt. Eine Erweiterung der in Gleichung 5.5 beschriebenen Zustände hinsichtlich besonderer Anforderungen (Überholmanöver, maximaler Komfort) ist möglich und könnte durch Parameteradaption erfolgen. An dieser Stelle soll eine allgemein gehaltene Auslegung genügen, die für die meisten Standardmanöver ausreicht.

Durch eine korrekte Kompensation von Reibungseinflüssen, hervorgerufen durch Windwiderstand und Schleppmoment, kann die Regelgüte erhöht und bspw. oftmals ein Betätigen der Bremse vermieden werden. Vergleichbar mit *Fuß vom Gas nehmen* ergibt sich so eine komfortable und materialschonende Fahrweise. Die durch Reibung ausgelöste Verzögerung $a_s^R(v)$ wurde empirisch ermittelt und ist als geschwindigkeitsabhängiges Kennfeld hinterlegt (siehe Abbildung 5.10 a).

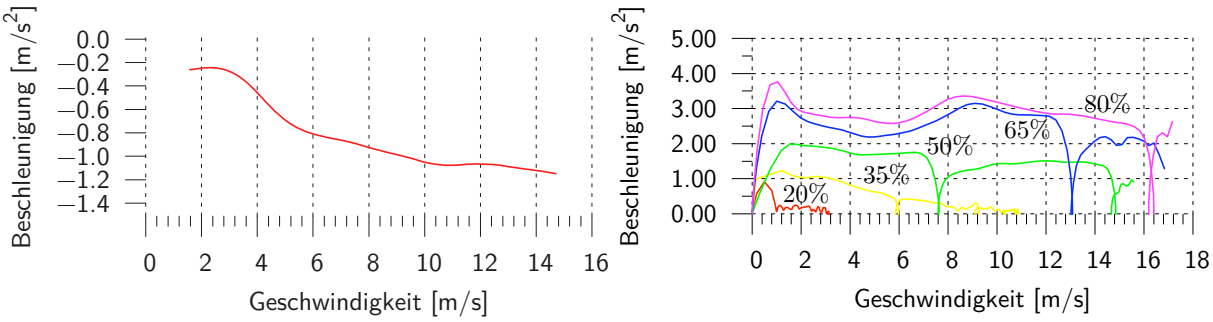


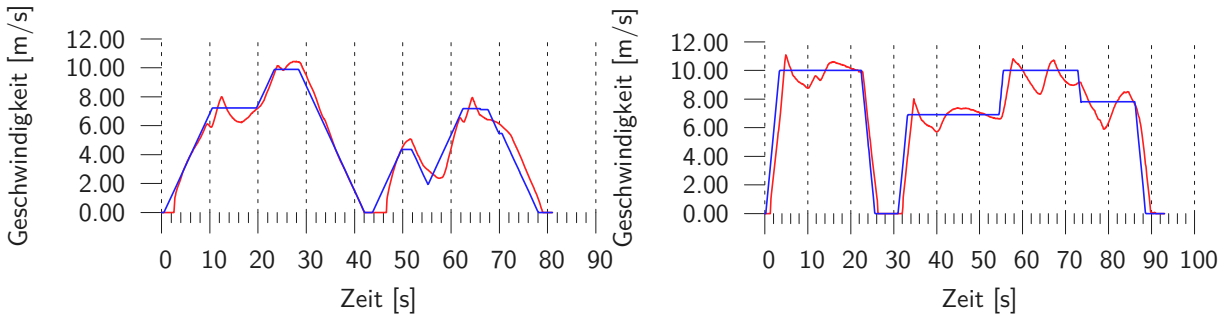
Abbildung 5.10: Zusammenhang zwischen Verzögerung $a_s^R(v)$, hervorgerufen durch Windwiderstand und Schleppmoment, und Geschwindigkeit v (a). Durch Testfahrten ermittelte Motorkennlinien für verschiedene E-PWG-Stellwerte in Prozent (b). Die Unterbrechungen resultieren aus den Schaltvorgängen und werden im Modell nicht berücksichtigt.

Abhängig vom aktuellen Neigungswinkel ρ wirkt eine nicht zu vernachlässigende Abtriebsbeschleunigung a_ρ, welche über einen Neigungswinkelsensor anteilig aus der Erdbeschleunigung zu $g \cdot \sin(\rho)$ berechnet wird. Ein Tiefpass sorgt dafür, dass Bewegungen des Fahrzeugaufbaus nicht als Schwingungen in den gemessenen Abtrieb, und damit in den Antriebsstrang, ein-

fließen. Nach der Kompensation von Reibungs- und Abtriebsanteilen in der Beschleunigung resultiert die korrigierte Beschleunigung $a_s^{R,\rho}$ für das reibungsfreie Fahrzeug in der Ebene:

$$a_s^{R,\rho} = a_s + a_R + a_\rho \tag{5.6}$$

Diese wird, bei positivem Vorzeichen mithilfe der hinterlegten Motorkennlinien (vgl. Abbildung 5.10 b) in Abhängigkeit von der aktuellen Geschwindigkeit in einen Stellwert für das E-PWG umgerechnet. Zwischen den aufgenommenen Kennlinien wird linear interpoliert. Die einzelnen Gänge können im Automatikbetrieb leider nicht abgefragt werden und sind daher hier nicht zu berücksichtigen. Bei negativem Vorzeichen wird die geforderte Verzögerung an den unterlagerten Bremsdruckregelkreis weitergereicht. Die Bremsdruckregelung ist dem Geschwindigkeitsregler unterlagert und für diesen transparent. Durch die Kaskadierung einerseits, und die um den Faktor 10 höhere Taktrate andererseits, wird eine Linearisierung des Bremsdruckregelkreises erreicht und die Regelung für den überlagerten Kreis vereinfacht. Eine gesonderte Behandlung der Bremsdruckregelung erfolgt in Abschnitt 5.5.2.



(a) (b)

Abbildung 5.11: Ergebnisse der Geschwindigkeitsregelung: Folgeverhalten der Regelung bei Vorgaben mit verschiedenen Beschleunigungswerten von 0,8 $\frac{m}{s^2}$ (a) und 2,5 $\frac{m}{s^2}$ (b). Die Sollwerte sind blau dargestellt, die Istwerte rot. Der Verlaufsgeber ist für die Rampenerzeugung zuständig (vgl. Abb. 5.8).

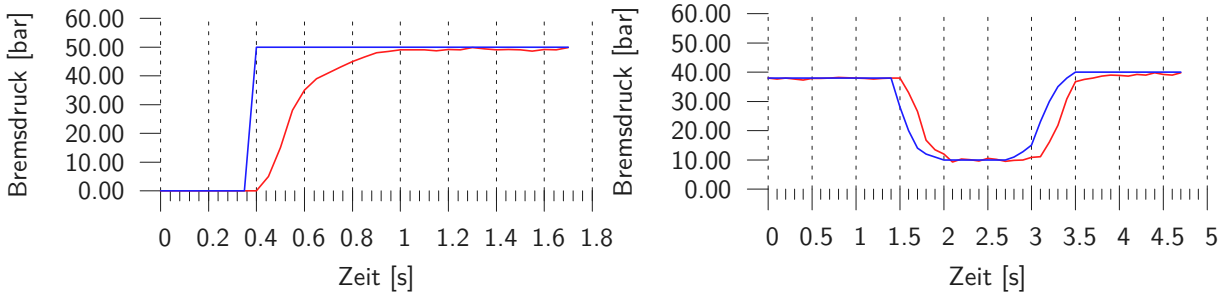
Abbildung 5.11 zeigt Ergebnisse der Geschwindigkeitsregelung. Durch die modellgestützte Regelung kann den Verläufen mit verschiedenen Beschleunigungswerten innerhalb einer Toleranz von 1-1,5 $\frac{m}{s}$ gefolgt werden. Bedingt durch die hinterlegte Totzone mit $v_{gap} = 1,2\,\frac{m}{s}$ wird der Stellwert nach dem Erreichen der Sollgeschwindigkeit auf Null gesetzt, und es erfolgt eine Annäherung an die untere Totzonengrenze. Sobald diese unterschritten wird setzt der Regler wieder ein. Aus diesem Ablauf ergeben sich die charakteristischen Verläufe um einen konstanten Geschwindigkeits-Sollwert. Einzelne, kurze Unterbrechungen im Istverlauf sind auf die Schaltvorgänge zurückzuführen.

5.5.2 Bremsdruckregelung

Aufgabe der Bremsdruckregelung ist es, den Gleichstrommotor über die Motorendstufe so einzustellen, dass sich ein vorgegebener Bremsdruck p_s ergibt. Um dem überlagerten Regelkreis effizient zuzuarbeiten, wird die Regelung mit einem Takt von 10 ms aufgerufen, dem zehnfachen des überlagerten Regelkreises. Zunächst wird die Vorgabe in Form der Verzögerung $a_s^{R,\rho}$ in einen Soll-Bremsdruck p_s umgerechnet. Dabei gilt der folgende Zusammenhang

$$p_s = \lambda \cdot a_s^{R,\rho} \tag{5.7}$$

mit Verstärkungsfaktor $\lambda = -12,5$. Der Bremsdruck bildet nun die Führungsgröße für einen PID-Regelkreis. Weitere Modelle kommen nicht zum Einsatz. Es wird direkt ein Stellwert für die I×R-Vorgabe an die Motorendstufe gesandt. Abbildung 5.12 zeigt Ergebnisse zweier Tests. Das Diagramm (b) zeigt eine Notbremsung mit sprungförmiger Vorgabe des Maximaldrucks von 50 Bar. Dieser Druck wird innerhalb von 0,5 s erreicht. Das Diagramm (b) zeigt die Vorgabe eines Bremsdruckverlaufs mit schnellen Wechseln zwischen 40 und 10 Bar.



(a) (b)

Abbildung 5.12: Notbremsung mit sprungförmiger Vorgabe des Maximaldrucks von 50 Bar (a) und folgen eines Bremsdruckverlaufs (b). Der Soll-Bremsdruck ist blau dargestellt, der Ist-Bremsdruck rot. Die zeitliche Verzögerung zwischen Soll- und Istwerten ist hauptsächlich dem elektromechanischen Ansteuerungsprinzip des Servomotors geschuldet, liegt aber mit ca. 0,5 s noch innerhalb der Reaktionszeit eines menschlichen Fahrers.

5.6 Querregelung

Die Querregelung dient dazu, das Fahrzeug entlang der vorgegebenen Spline-Kurve zu führen. Dieses Modul ist von der Längsregelung vollständig entkoppelt. Zunächst wird der Sollwert ϕ_s für den Lenkwinkel aus der Ablage und der Disorientierung des Fahrzeuges zur Kurve berechnet. Dazu wird mit einem numerischen Verfahren der nächste Punkt zum Referenzpunkt des Fahrzeuges auf dem Spline gesucht (Matching). An dieser Stelle werden die Orientierung θ_C und die Krümmung κ_C abgenommen (siehe Abbildung 5.13).

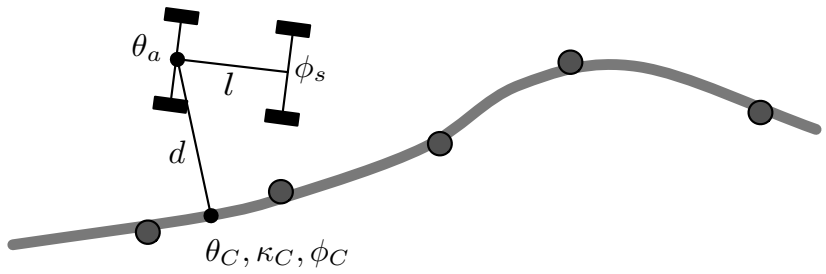


Abbildung 5.13: *Matching* des Fahrzeugreferenzpunktes auf die Bahn durch ein numerisches Iterationsverfahren. Aus dem an dieser Stelle optimalen Lenkwinkel ϕ_C, der Ablage d und der Fehlorientierung $\theta_C - \theta_a$ wird der Soll-Lenkwinkel ϕ_s für den Querregler berechnet.

Daraus resultiert ein an dieser Stelle optimaler Lenkwinkel $\phi_C = \text{atan}(l\kappa_C)$, wobei l der Abstand zwischen Vorder- und Hinterachse des Fahrzeuges ist. Zusammen mit der Ablage d und der aktuellen Fahrzeugorientierung ϕ_a wird mit einem einfachen Regelgesetz in Zustandsraumbeschreibung der Sollwinkel ϕ_s berechnet zu

$$\phi_s = \phi_C + k_d d + k_\theta(\theta_C - \theta_a), \tag{5.8}$$

wobei k_d und k_θ Regelparameter für den Einfluss der Ablage und der Fehlorientierung auf die Spurrückführung sind. Der resultierende Sollwinkel ϕ_s wird vom Regelkreis in Abbildung 5.14 übernommen und ausgeregelt.

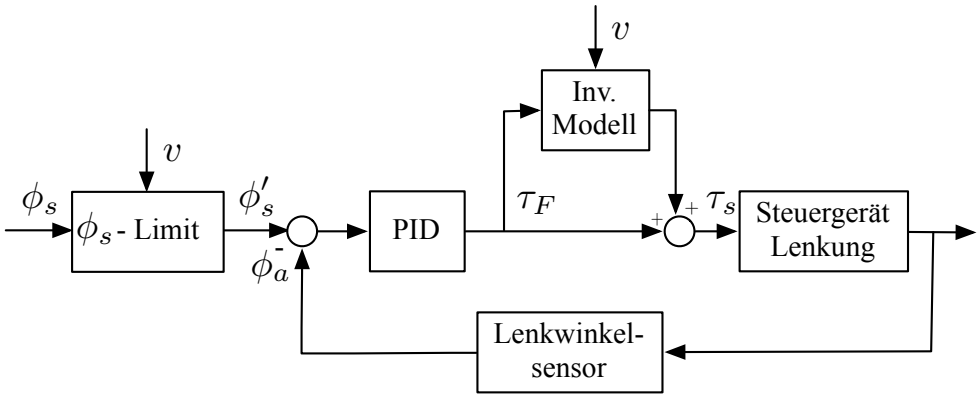


Abbildung 5.14: Regelkreis für die Lenkwinkelregelung. Nach Berechnung aus dem Zustandsregler (Gleichung 5.8) wird der Sollwinkel ϕ_s zunächst geschwindigkeitsabhängig begrenzt und dann von einem PID-Regler ausgeregelt. Ein inverses Modell der Unterstützungskennlinien berücksichtigt nichtlineare Größen wie Reifenreibung und Rückstellmomente.

Um zu schnelle Lenkbewegungen bei Vorgabe ungeeigneter Bahnpunkte zu unterdrücken wird der maximale Lenkwinkelausschlag geschwindigkeitsabhängig begrenzt. Dies hilft Instabilitäten des Fahrzeuges zu vermeiden und dient als weiteres Sicherheits-Plus insbesondere in der Entwicklungsphase. Aus Versuchsfahrten mit einem menschlichen Fahrer wurde eine

Einhüllende aufgenommen, welche als Filter für den veranschlagten Soll-Lenkwinkel dient (siehe Abbildung 5.15).

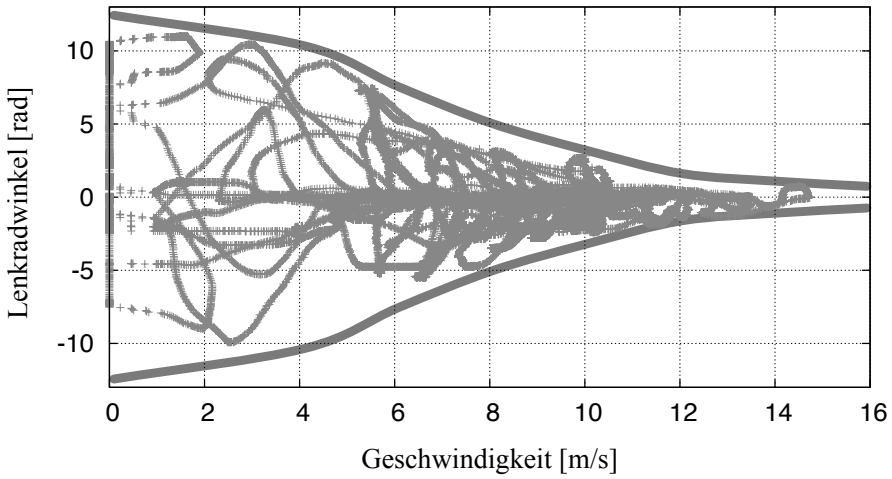
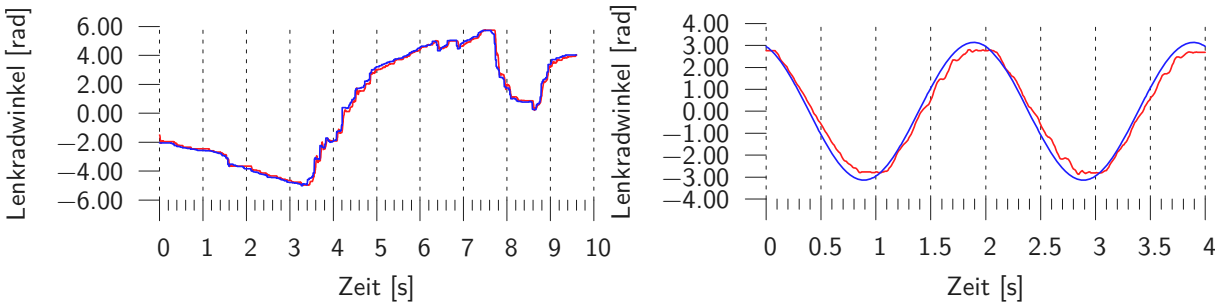


Abbildung 5.15: Die Einhüllende dient der Begrenzung von Sollvorgaben für den Lenkwinkel ϕ in Anhängigkeit von der aktuellen Geschwindigkeit. Im Diagramm ist der Lenkradwinkel aufgetragen. Das Verhältnis zum Lenkwinkel beträgt 19,5:1. Als Vorbild diente ein menschlicher Fahrer.

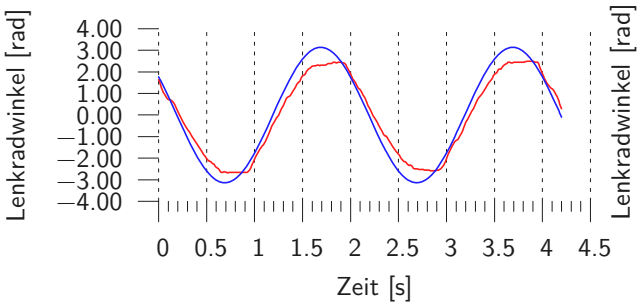
Als Regler wird ein linearer Folgeregler vom Typ PID verwendet. Der Regler folgt der korrigierten Vorgabe ϕ'_s, wobei keine Sprünge in der Führungsgröße auftreten dürfen. Dies ist jedoch üblicherweise durch die verwendeten kubischen B-Splines ausgeschlossen. Als Stellgröße wird ein Lenkmoment τ_F ausgegeben, welches dem Lenkmoment eines virtuellen Fahrers entspricht. Durch Anwendung der invertierten Kennlinien des Lenkungssteuergerätes wird dem Lenkmoment des Fahrers die notwendige Unterstützung hinzugefügt (vgl. *Lenkungsansteuerung* in Abschnitt 5.4.2 und Abbildung 5.6). Die durch diese Kompensation erreichte (teilweise) Linearisierung der Strecke ist aufgrund der starken Geschwindigkeitsabhängigkeit unbedingt notwendig. Abbildung 5.16 zeigt Ergebnisse des Lenkwinkelreglers für die Vorgabe eines manuellen Lenkwinkelverlaufs und eine sinusförmige Anregung bei verschiedenen Geschwindigkeiten.

Häufig wird auch eine sogenannte Sichtpunkt- oder Deichselregelung verwendet, um einen Sollwert für den Lenkwinkel vorzugeben. Dabei läuft ein Sichtpunkt in konstantem Abstand vor dem Fahrzeug auf der gegebenen Kurve. Der einzuschlagende Lenkwinkel ergibt sich nun aus einer gedachten Deichselstellung und zielt damit in Richtung des Sichtpunktes auf der Kurve. Diese Methode hat den Vorteil, dass eine stabile Rückführung auf die Kurve erfolgt. Allerdings müssen dazu nicht unerhebliche Abweichungen, abhängig von der Distanz zum Sichtpunkt, in Kauf genommen werden. Für das genaue Abfahren einer Bahn wie es bspw. zum Einparken verlangt wird, eignet sich diese Methode damit nicht.

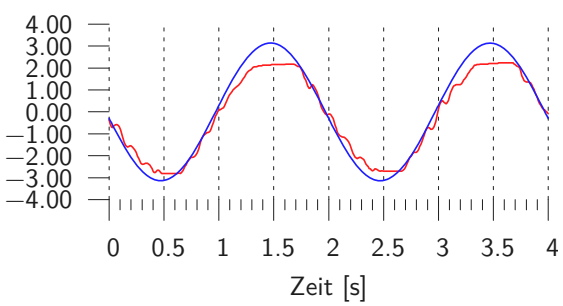


(a) Manuelle Vorgabe bei v=5 $\frac{\text{m}}{\text{s}}$

(b) Sinusförmige Vorgabe bei v=5 $\frac{\text{m}}{\text{s}}$



(c) Sinusförmige Vorgabe bei v=10 $\frac{\text{m}}{\text{s}}$



(d) Sinusförmige Vorgabe bei v=15 $\frac{\text{m}}{\text{s}}$

Abbildung 5.16: Folgeverhalten des Lenkwinkelreglers bei einer manuellen Vorgabe des Lenkradwinkels (a). Der Soll-Lenkradwinkel ist blau dargestellt, der Ist-Lenkradwinkel rot. Eine sinusförmige Anregung des Soll-Lenkradwinkels mit 4 Hz bei verschiedenen Geschwindigkeiten zwischen 5 und 15 $\frac{\text{m}}{\text{s}}$ zeigen die Abbildungen (b-d). Mit höherer Geschwindigkeit sind zunehmende Abweichungen an den Extremstellen zu beobachten. Lenkradwinkelstellungen von mehr als ± 3 rad kommen bei regulären Fahrmanövern für Geschwindigkeiten oberhalb 10 $\frac{\text{m}}{\text{s}}$ nicht mehr vor, sodass Abweichungen in diesen Bereichen toleriert werden können (vgl. *Einhüllende* in Abbildung 5.15).

Kapitel 6

Ergebnisse und Evaluation

In diesem Kapitel werden die Ergebnisse der Verhaltenssteuerung präsentiert und evaluiert. Teilergebnisse der Bahnplanungs- und Regelungskomponenten wurden bereits in den Kapiteln 4 und 5 vorgestellt und diskutiert, so dass hier nur mehr die Ergebnisse des Gesamtsystems überprüft werden. Im ersten Teil dieses Kapitels wird die Verhaltenssteuerung anhand zahlreicher Testfälle überprüft, welche sowohl in der Simulation als auch in der Praxis ausgeführt wurden. Im zweiten Teil wird untersucht, inwieweit die in Abschnitt 3.2 speziell für die Verhaltenssteuerung formulierten Anforderungen erfüllt wurden und ob die in Abschnitt 1.2 geforderte Zielsetzung als Ganzes erreicht wurde.

6.1 Ergebnisse

6.1.1 Testumgebung

Alle im Folgenden durchgeführten Tests wurden auf dem Testgelände der Mackensen-Kaserne in Karlsruhe durchgeführt. Abbildung 6.1 zeigt ein Luftbild des Testgeländes mit überlagertem topografischen Straßennetz. Das gezeigte Gelände ist ca. $250 \times 100\, m^2$ groß. Auf den 150 m langen Teilstücken im linken Bereich der Karte sind Überhol- oder Ausweichmanöver mit geringer Geschwindigkeit möglich. Das Straßennetz ist in Form einer Straßenkarte im System abgelegt und bildet die Grundlage für alle Testszenarien in Simulation und Realität. In der Simulation kommt ein im SFB/TR28 entwickeltes Fahrzeugmodell als Ersatz für den Versuchsträger zum Einsatz. Eine Simulation von Sensordaten erfolgt auf Wahrnehmungsebene nicht, sodass Wahrnehmungskomponenten wie *Spurerkenner* und *Fahrzeug-Verfolger* in der Simulation keine Anwendung finden.

Stattdessen werden die für einen Test benötigten Verkehrsteilnehmer oder Hindernisse direkt auf symbolischer Ebene eingebracht und bspw. mit einem Joystick die gewünschte Bewegung erzeugt. Entsprechende Modelle stellen ein realitätsnahes Verhalten sicher, zudem werden Sensormessfehler durch ein Verrauschen von Positionen und Geschwindigkeiten der Objekte simuliert. Das Straßennetz besteht aus zwei kreisförmig verlaufenden Straßenzügen, die über eine Kreuzung miteinander verbunden sind.

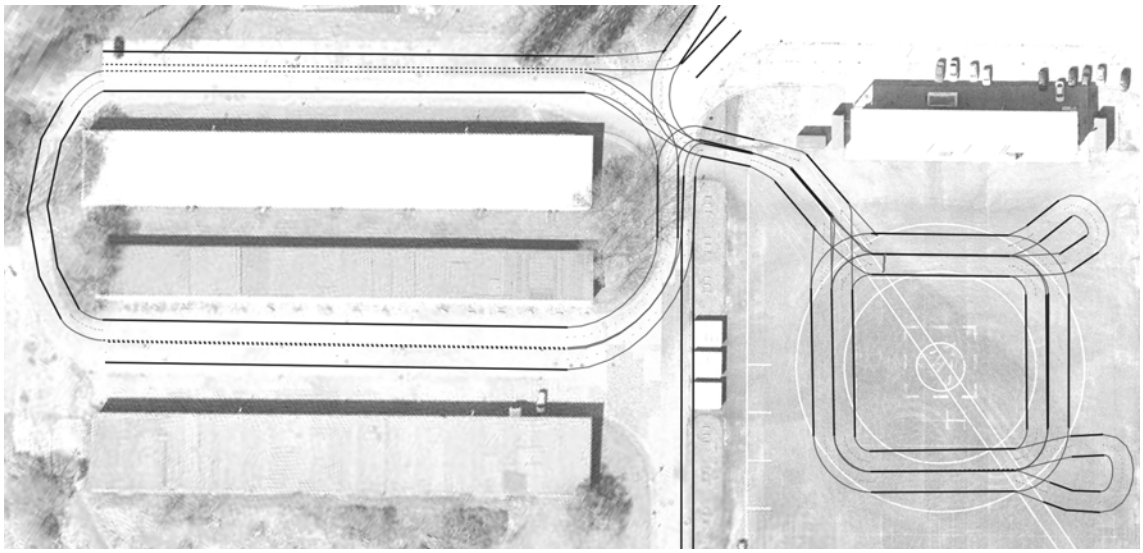


Abbildung 6.1: Luftbild des Testgeländes in der Mackensen-Kaserne in Karlsruhe. Das für die Testszenarien verwendete topografische Straßennetz ist dem Bild überlagert.

6.1.2 Evaluationskriterien und Testszenarien

Zunächst sollen in diesem Abschnitt Kriterien für eine Überprüfung des Systems definiert werden, mit deren Hilfe sich Testszenarien formulieren lassen. Die im Rahmen der Urban Challenge 2007 geforderten und vom Team AnnieWay mithilfe von endlichen Zustandsautomaten gelösten Aufgaben sollen als Evaluationskriterien für dieses System dienen, soweit dies möglich ist. Dies hat den Vorteil, dass ein Vergleich der verschiedenen Herangehensweisen möglich wird. Die für den Wettbewerb vorgegebene Aufgabenstellung ist unter [Urban Challenge 07] detailliert aufgelistet. Die folgenden Fragestellungen werden als Kriterien für eine Bewertung des Systems zugrunde gelegt:

1. Ist eine Durchführung von Aufgaben ähnlichen Umfangs wie bei der Urban Challenge 2007 möglich?

2. Liefert das Verhaltensnetzwerk virtuelle Sensorwerte, die aussagekräftige Erfahrungswerte ausdrücken?

3. Bleibt das System bei Ausfall einzelner Komponenten innerhalb der Verhaltenssteuerung sicher?

In der Aufgabenstellung der Urban Challenge lassen sich Einzelaspekte identifizieren, die gezielt überprüft werden können. Daraus werden die folgenden Testszenarien für eine Evaluation dieser Arbeit abgeleitet:

- *Spur folgen*
- *Überholen eines statischen Hindernisses*

- *Überholen eines fahrenden Fahrzeugs*
- *Fahrzeug folgen*

Parkmanöver wurden bereits im Kapitel 4 Bahnplanung ausgiebig getestet. Sie finden hier keine weitere Betrachtung. Ein Test des Kreuzungsverhaltens *4-Way-Stop* ist zum gegebenen Zeitpunkt leider noch nicht möglich, da die von der Situationsinterpretation benötigten Daten nicht vollständig vorliegen. Das Eigenfahrzeug kommt an der Stopplinie zum Stehen, verfügt jedoch über keine Informationen bezüglich der Reihenfolge wartender Fahrzeuge und kann den Konflikt nicht auflösen. Zur Überprüfung der Anforderungen nach Robustheit und Fehlertoleranz dienen zwei weitere Tests:

- *Ausfall der taktischen und strategischen Schicht*
- *Ausfall eines reaktiven Verhaltens*

Die aufgelisteten Testszenarien werden im folgenden Abschnitt in Simulation und Realität ausgeführt.

6.1.3 Test der Verhaltenssteuerung

Testszenario *Spur folgen*

Beschreibung: In diesem Testszenario soll die grundlegende Funktionalität der Verhaltenssteuerung und das Zusammenspiel von Verhalten überprüft werden. Eine Beteiligung anderer Verkehrsteilnehmer erfolgt nicht. Das kognitive Automobil hat in diesem Szenario einer vorgegebenen Spur zu folgen. Abbildung 6.2 zeigt die Situation. Dabei wird eine enge 180°-Kurve durchfahren, um die Fahrtrichtung umzukehren. Der Test wurde in der Simulation und mit dem realen Fahrzeug ausgeführt.

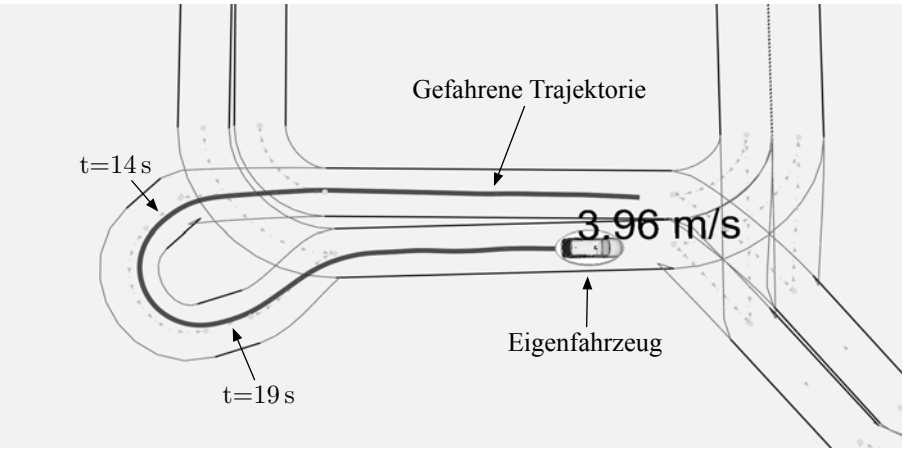
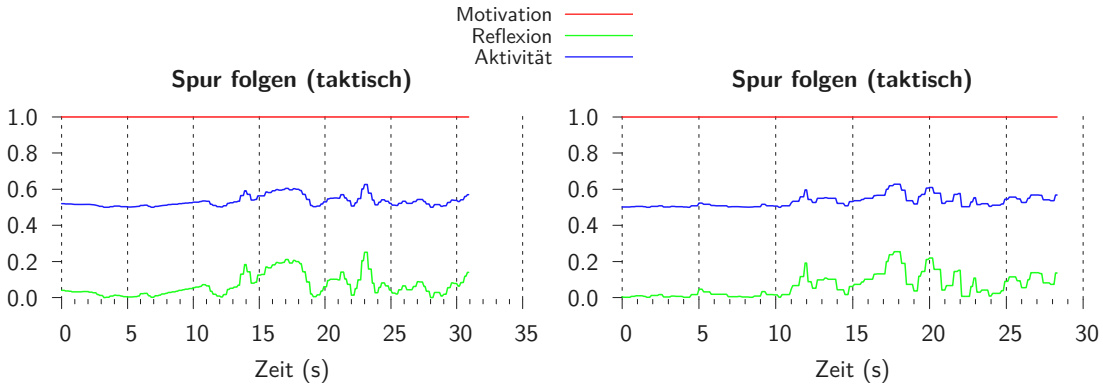


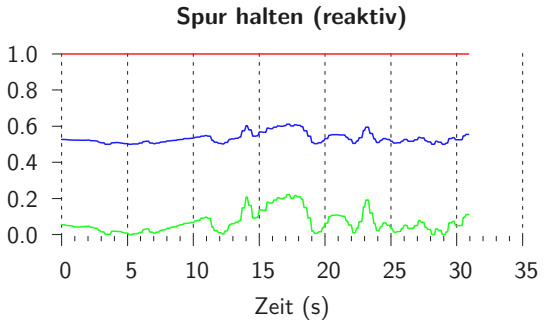
Abbildung 6.2: Bildschirmfoto aus der Simulation nach Ausführung des Testszenarios *Spur folgen*. Dargestellt sind das Straßennetz, die gefahrene Trajektorie und das Eigenfahrzeug.

Ergebnisse: In diesem Testszenario sind lediglich die Verhalten *Spur folgen* und *Spur halten* von Belang. Die Einzelgrafiken in Abbildung 6.3 zeigen Verlaufswerte für Motivation, Reflexion und Aktivität der getesteten Verhalten. Die Abbildungen 6.3a und 6.3c veranschaulichen diese Werte für einen Simulationslauf. Abbildungen 6.3b und 6.3d zeigen die Werte als Ergebnis eines Tests mit dem Versuchsträger. Für die Bedeutung von Motivation, Reflexion und Aktivität sei auf Abschnitt A.3.1 verwiesen. Die Berechnungsfunktionen wurden für alle Verhalten in Abschnitt 3.3.5 aufgelistet.

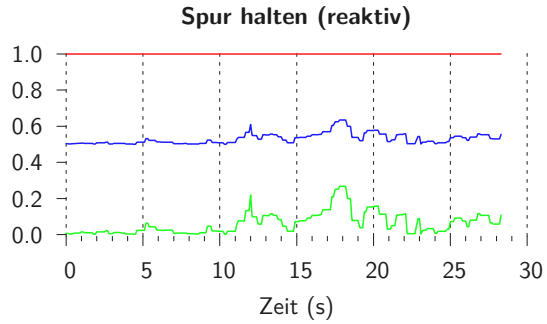


(a) *Spur folgen* in der Simulation.

(b) *Spur folgen* auf dem Fahrzeug.



(c) *Spur halten* in der Simulation.



(d) *Spur halten* auf dem Fahrzeug.

Abbildung 6.3: Ergebnisse des Testfalls *Spur folgen* in Simulation und auf dem Fahrzeug.

Die Ergebnisse aus Simulation und realem Versuch unterscheiden sich nur geringfügig. In beiden Fällen lässt sich durch das Verhaltensnetzwerk eine Gefahrenkarte erstellen, auf deren Grundlage eine Bahn geplant und ausgeregelt wird. Der Test zeigt, dass die Integration der Verhaltenssteuerung und deren Schnittstellen in die Systemarchitektur funktioniert. An manchen Stellen des Parcours sind geringe Abweichungen von der Spurmitte erkennbar, die sich durch erhöhte Reflexionswerte ausdrücken. Tendenziell ist dies auf den geraden Strecken weniger der Fall. Auffällig sind leicht erhöhte Reflexionswerte im Zeitfenster zwischen 14 s und 19 s. Dies hängt damit zusammen, dass die Bahnplanung aufgrund der kinematischen Beschränkungen in diesem engen Kurvenbereich keine exakt in der Spurmitte liegende Bahn errechnen konnte.

Testszenario *Überholen eines statischen Hindernisses*

Beschreibung: Dieses Testszenario demonstriert die Ausführung des taktischen Verhaltens *Überholen* anhand eines statischen Hindernisses auf der Eigenspur. Das in Abbildung 6.4 veranschaulichte Szenario zeigt das statisch platzierte Hindernis und das Eigenfahrzeug, welches unter Zuhilfenahme der Gegenspur überholt. Eine Nutzung der Gegenspur ist aus verkehrsrechtlicher Sicht erlaubt. Dieses Szenario zeigt die Verwendung von Objektinformationen über andere Verkehrsteilnehmer und den Informationsfluss zwischen Situationsinterpretation und Verhaltensentscheidung sowie die korrekte Verarbeitung der Daten. Das taktische Verhalten *Überholen* wird in diesem Szenario unter einfachen Randbedingungen getestet.

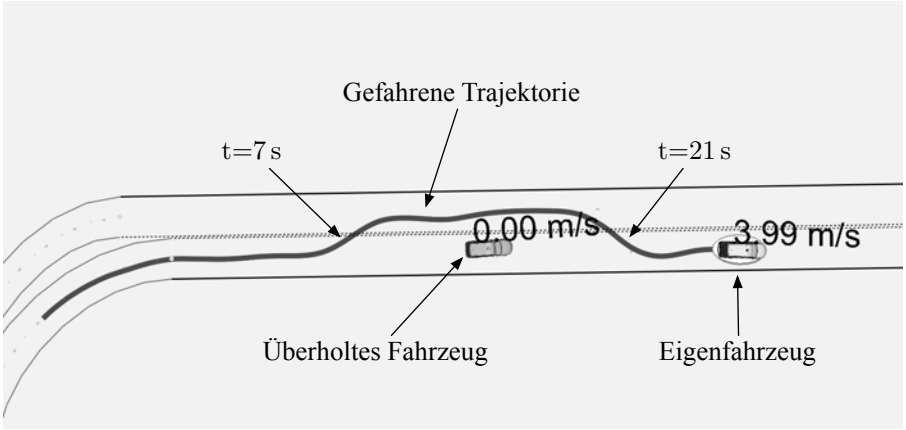
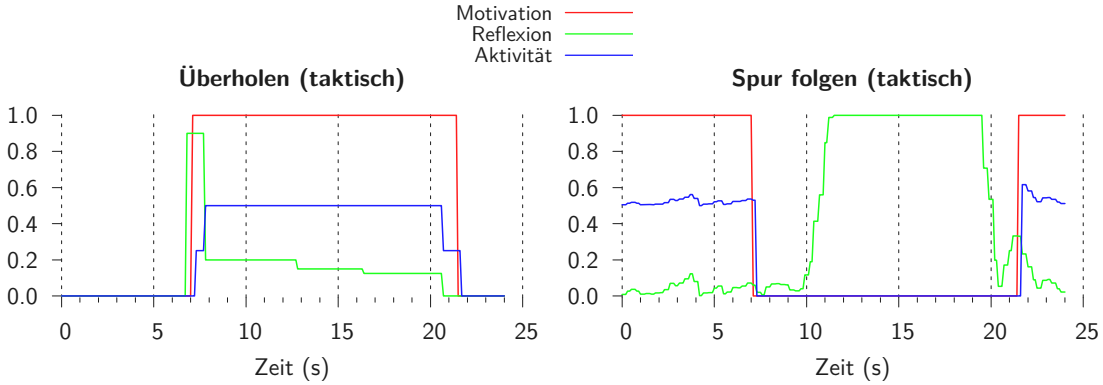


Abbildung 6.4: Bildschirmfoto des Testszenarios *Überholen eines statischen Hindernisses*. Zu erkennen ist das Eigenfahrzeug mit gefahrener Trajektorie sowie ein überholtes Fahrzeug als stehendes Hindernis.

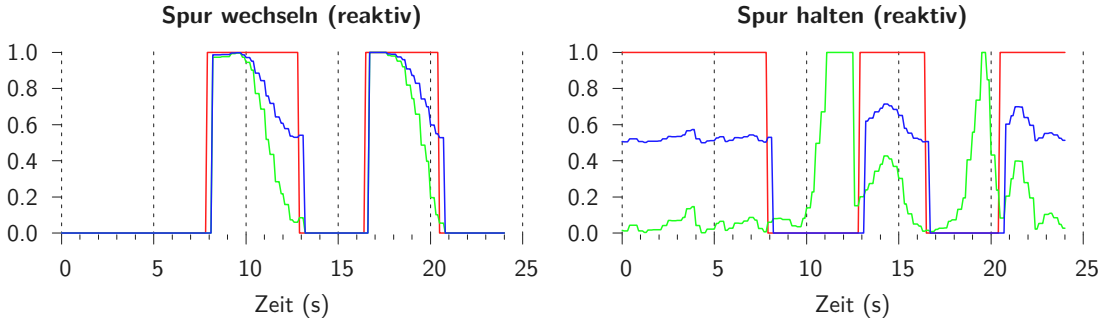
Ergebnisse: Die beiden Abbildungen 6.5 und 6.6 zeigen Verlaufswerte von Motivation, Reflexion und Aktivität der beteiligten Verhalten für eine simulierte Ausführung und einen realen Test auf dem Versuchsträger. Zum besseren Verständnis des Ablaufs sei der interne Ablauf eines Überholvorganges zunächst beschrieben:

Das taktische Verhalten *Überholen* erkennt ein voraus fahrendes Fahrzeug und bestimmt ein Maß für die Behinderung durch dieses Fahrzeug, hervorgerufen aufgrund der Differenzgeschwindigkeiten und des Abstandes. Resultiert daraus der Wunsch zum Überholen, und besteht zudem die Möglichkeit dazu, so wird diese Information für das momentan aktive strategische Verhalten hinterlegt. Sobald das strategische Verhalten *Überholen* motiviert, parametriert dieses die reaktiven Verhalten *Spur halten* und *Spur wechseln* und leitet einen Spurwechsel als erste Phase des Vorganges ein. Bei einem Spurwechsel muss das Verhalten *Abstand halten* demotiviert werden, da der Sicherheitsabstand bei einem Ausscheren nicht mehr eingehalten werden kann und der Spurwechsel damit behindert würde. Nach dem Passieren des statischen Hindernisses mit dem Verhalten *Spur halten* wird ein nochmaliger Spurwechsel ausgeführt, um wieder auf die ursprüngliche Spur zu gelangen.

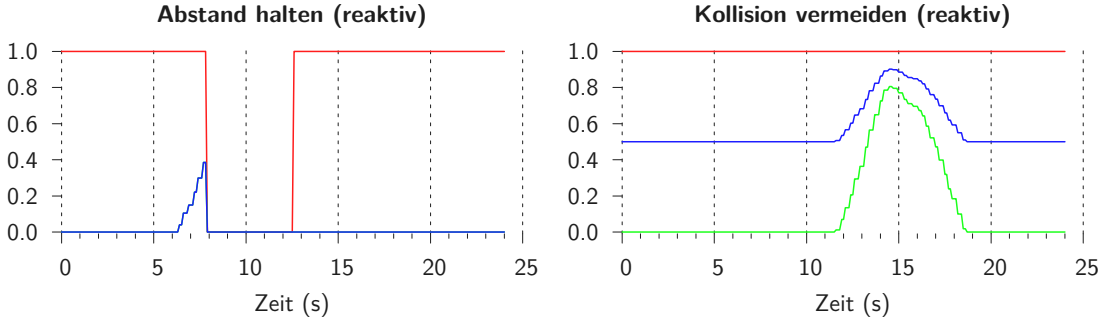
Abhängig vom Abstand zum Hindernis zeigt die Reflexion von *Kollision vermeiden* eine erhöhte Unzufriedenheit. Dieser Wert erreicht ein Maximum zum Zeitpunkt 15 s, dann sind sich die Fahrzeuge am nächsten. Die reaktiven Verhalten *Spur wechseln* und *Spur halten* werden wechselseitig motiviert. Ihre Reflexionswerte repräsentieren die Positionen innerhalb der jeweiligen Spuren. Das als (bewusste) Fahrhandlung ausgeführte taktische Verhalten *Spur folgen*, das während der gesamten Überholphase demotiviert ist, zeigt eine nachvollziehbar hohe Unzufriedenheit, solange sich das Eigenfahrzeug auf der Gegenspur befindet. Würde bspw. der Überholvorgang abgebrochen, dann würde dieses Verhalten versuchen einen Spurwechsel herbeizuführen, um wieder auf die Eigenspur zu gelangen.



(a) *Überholen* in der Simulation. (b) *Spur folgen* in der Simulation.



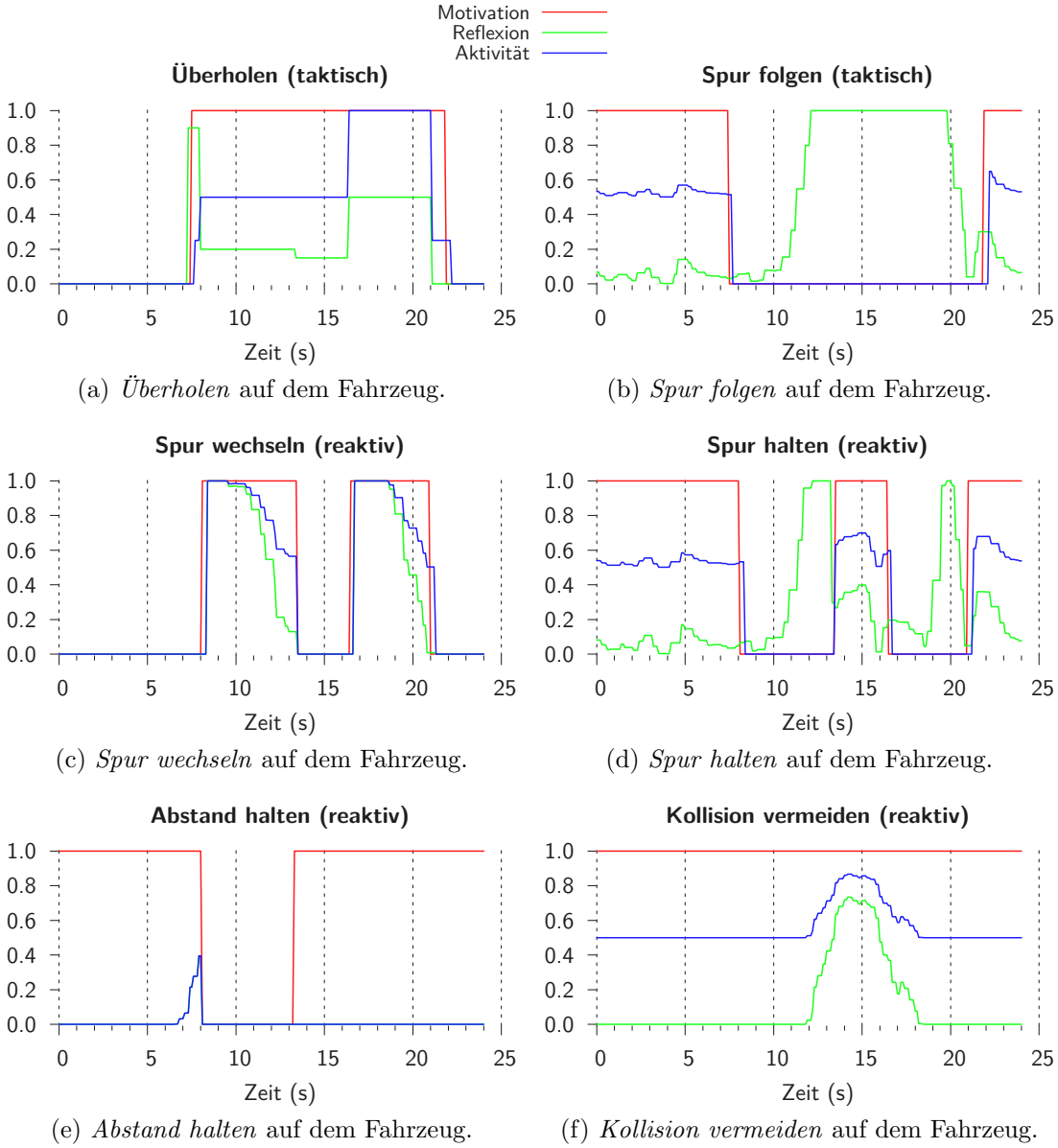
(c) *Spur wechseln* in der Simulation. (d) *Spur halten* in der Simulation.



(e) *Abstand halten* in der Simulation. (f) *Kollision vermeiden* in der Simulation.

Abbildung 6.5: Ergebnisse des Testszenarios *Überholen eines statischen Hindernisses* aus der Simulation.

Bei Betrachtung der Testergebnisse wird eine Diskrepanz zwischen einer Ausführung in der Simulation und auf dem realen Versuchsträger deutlich, die der real schwierigeren Wahrnehmungsaufgabe geschuldet ist: Zum Zeitpunkt T=17 s ist beim Verhalten *Überholen* ein Anstieg von Aktivität und Reflexion zu beobachten. Dies bedeutet einen Abbruch des Überholvorganges (vgl. Beschreibung der Reflexionswerte in Abschnitt 3.3.4). Grund dafür war ein kurzzeitiges Verlieren des zu überholenden Fahrzeuges durch die Wahrnehmungskomponente. Beim Wiedererkennen wurde ein Fahrzeug mit neuer Identifikation übermittelt, so dass der Überholvorgang des ursprünglichen Fahrzeuges nicht fortgesetzt werden konnte. In dieser späten Phase hatte dies jedoch keinen Einfluss mehr auf den Überholvorgang.



(a) *Überholen* auf dem Fahrzeug.

(b) *Spur folgen* auf dem Fahrzeug.

(c) *Spur wechseln* auf dem Fahrzeug.

(d) *Spur halten* auf dem Fahrzeug.

(e) *Abstand halten* auf dem Fahrzeug.

(f) *Kollision vermeiden* auf dem Fahrzeug.

Abbildung 6.6: Ergebnisse des Testszenarios *Überholen eines statischen Hindernisses* auf dem Versuchsträger.

Probleme dieser Art könnten in Zukunft durch eine Filterung auf Interpretationsebene vermieden werden. Bei genauer Betrachtung fällt auf, dass die Werte für die Aktivität einiger Verhalten der Beschreibung in Abschnitt A.3.1 widersprechen, da ein Verhalten nicht aktiv sein kann, wenn es nicht motiviert ist. Diese verlängerte Aktivität beschränkt sich auf einen Ausführungsschritt der jeweiligen Ebene und erklärt sich dadurch, dass Reflexion und Aktivität in der aktuellen Implementierung immer für den letzten Verarbeitungsschritt berechnet werden. Abbildung 6.7 zeigt Aufnahmen zu sechs Zeitpunkten bei der realen Versuchsdurchführung in der Mackensen-Kaserne.

Abbildung 6.7: Versuchsdurchführung des Testszenarios *Überholen eines statischen Hindernisses* in der Mackensen-Kaserne in Karlsruhe. Die farbig markierten Hütchen dienen der Veranschaulichung der in Abbildung 6.4 gezeigten Spur.

Testszenario *Überholen eines fahrenden Fahrzeugs*

Beschreibung: Ähnlich dem zuvor beschriebenen Testszenario wird in diesem Versuch ein fahrendes Fahrzeug überholt (vgl. Abbildung 6.8). Während sich in der Verhaltenssteuerung die einzelnen Phasen des Überholens mit Ausnahme der Zeitdauer kaum vom vorigen Beispiel unterscheiden, so stellt diese Aufgabe erhöhte Anforderungen an die Bahnplanungskomponente bzgl. der Verarbeitung von dynamischen Gefahrenkarten. Mit diesem Versuch soll überprüft werden, ob eine Handhabung dynamischer Situationen durch das Gesamtsystem möglich ist. Der Versuch wurde in der Simulation und auf einem realen Versuchsträger ausgeführt.

Ergebnisse: Die zeitlichen Verläufe der Motivationen, Reflexionen und Aktivitäten sind in den Abbildungen 6.10 und 6.11 für Simulation und realen Test dargestellt. Sie unterscheiden

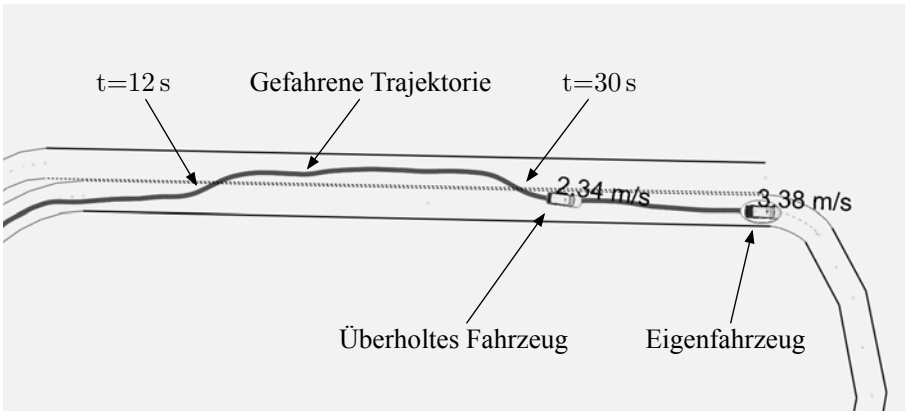


Abbildung 6.8: Bildschirmfoto des Testszenarios *Überholen eines fahrenden Fahrzeugs* ausgeführt auf dem Versuchsträger. Dargestellt sind das Straßennetz, das überholte Fahrzeug am Versuchsende, das Eigenfahrzeug und dessen gefahrene Trajektorie.

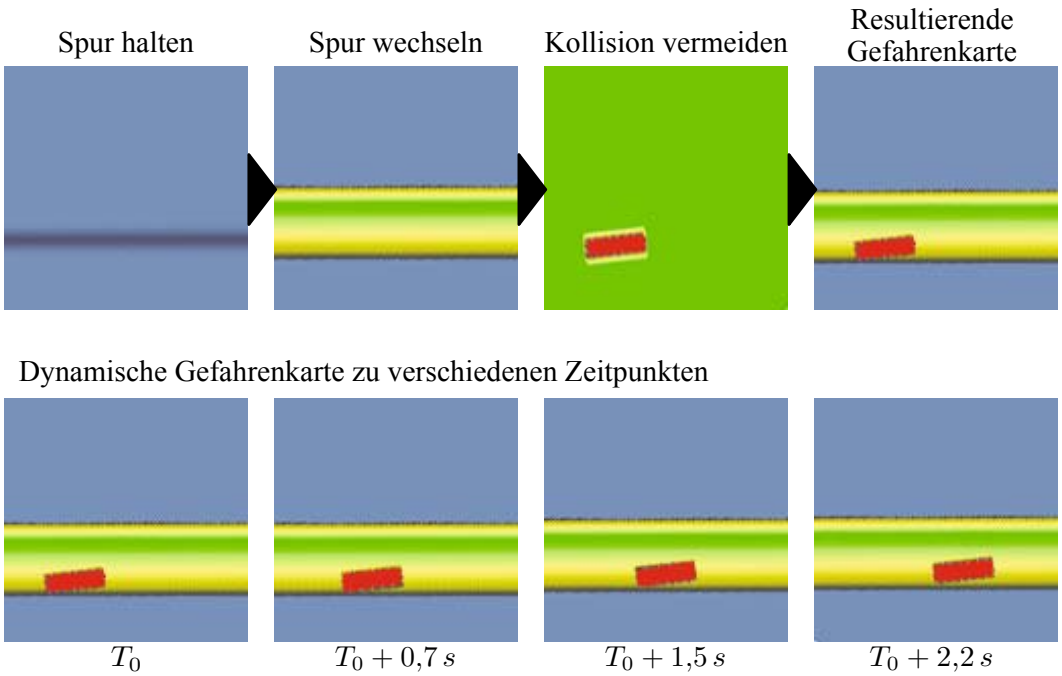
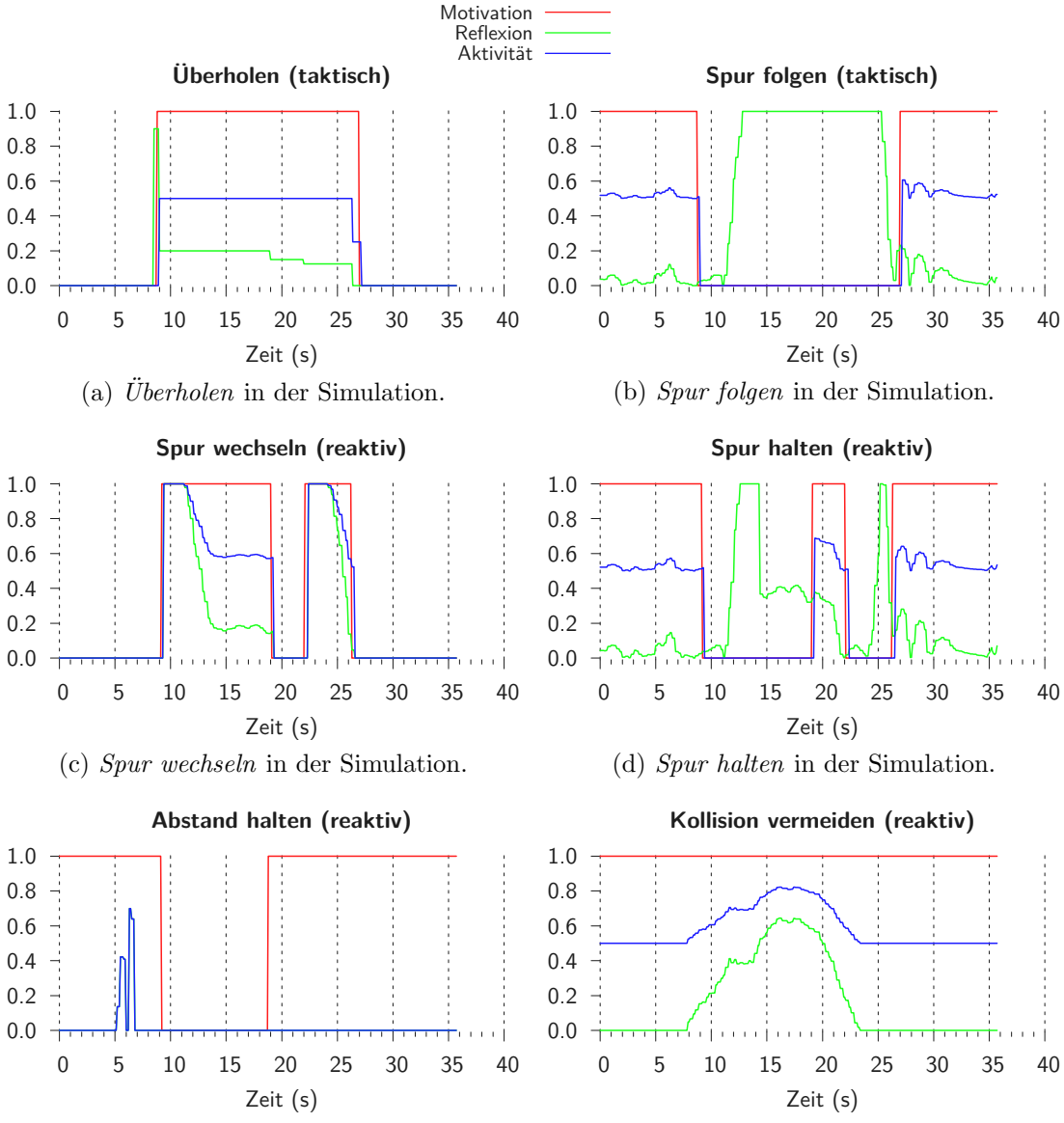


Abbildung 6.9: Zusammensetzung der Gefahrenkarte aus den Einzelkarten der Verhalten *Spur halten*, *Spur wechseln* und *Kollision vermeiden* zum Zeitpunkt T_0=16 s (obere Reihe). Das Verhalten *Kollision vermeiden* prädiziert Hindernisse in die Zukunft, so das sich daraus eine dynamische Gefahrenkarte bilden lässt. Die untere Reihe zeigt die fusionierte Gefahrenkarte zu verschiedenen zukünftigen Zeitpunkten relativ zu T_0. In der gezeigten Darstellung bedeutet grün sehr geringe Gefahr, gelb mittlere, dunkelgrau hohe Gefahr (noch befahrbar). Blaue und rote Bereiche werden von der Bahnplanung ausgeschlossen.

sich von jenen im vorigen Versuch lediglich durch Details. So lässt sich für das Verhalten *Spur folgen* ein wesentlich längerer Zeitraum mit hoher Reflexion beobachten, da sich das Eigenfahrzeug lange auf der Gegenspur befindet. Es ist zu erkennen, dass das Verhalten *Abstand halten* beim Auffahren auf das Fahrzeug höhere Reflexions- und Aktivitätswerte aufweist (diese sind im Diagramm deckungsgleich) und das Eigenfahrzeug durch dieses Verhalten an die Geschwindigkeit des vorausfahrenden Fahrzeuges angepasst wird. Der Überholwunsch wird erst nach diesem Vorgang durch hohe Reflexionswerte von *Überholen* geäußert.



(a) *Überholen* in der Simulation.

(b) *Spur folgen* in der Simulation.

(c) *Spur wechseln* in der Simulation.

(d) *Spur halten* in der Simulation.

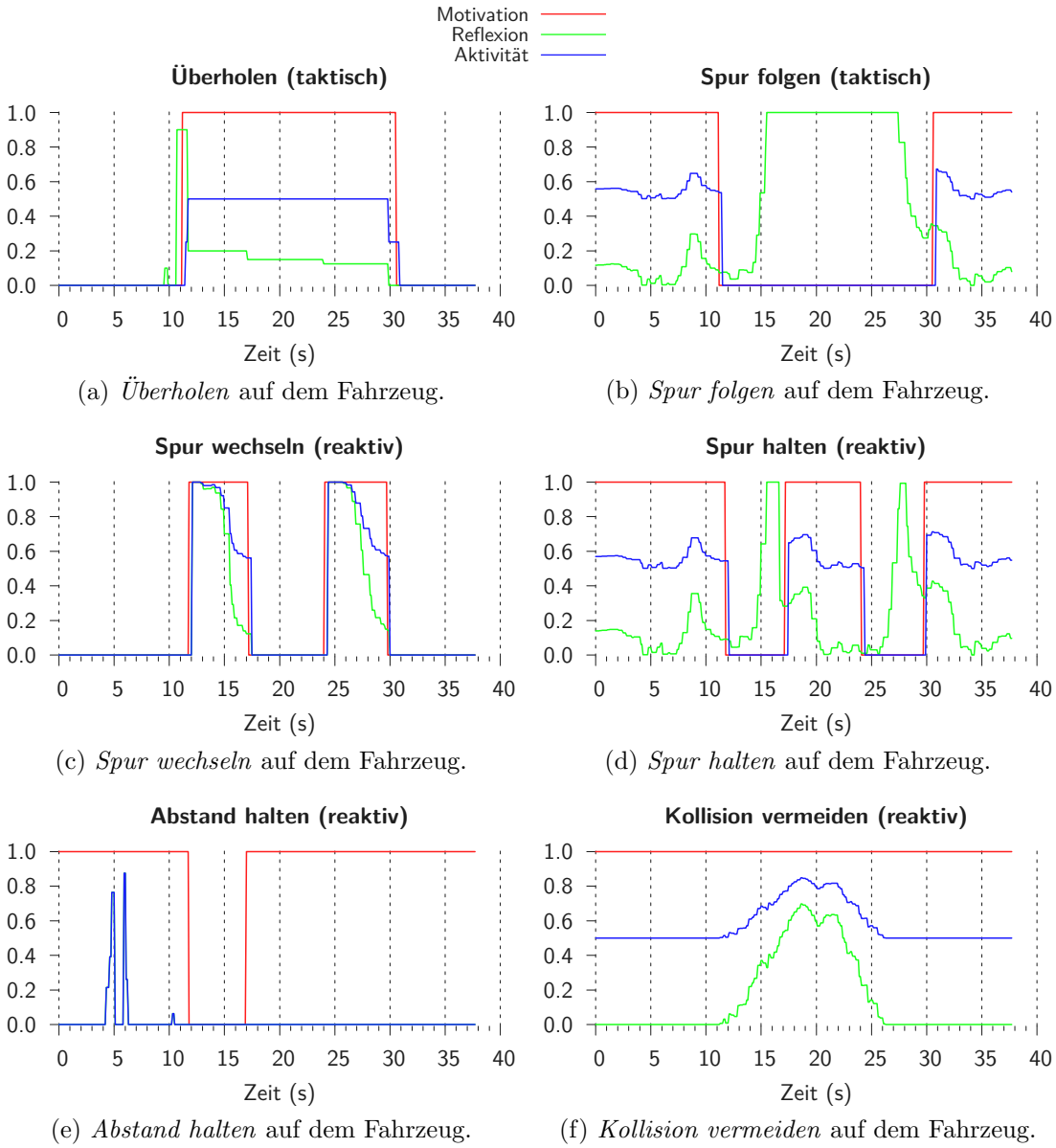
(e) *Abstand halten* in der Simulation.

(f) *Kollision vermeiden* in der Simulation.

Abbildung 6.10: Ergebnisse des Testszenarios *Überholen eines fahrenden Fahrzeugs* aus der Simulation.

Es ist ausserdem zu beobachten, dass der Spurwechselvorgang in der Simulation länger gedauert hat als in der realen Ausführung. Ereignisse dieser Art sind auf die Abstimmung zwi-

schen Verhaltenssteuerung und Bahnplanung zurückzuführen. So hat die Bahnplanung das Eigenfahrzeug mehrere Sekunden auf einer lateralen Position innerhalb der Zielspur geführt, welche als Auslöser für einen vollständig durchgeführten Spurwechsel noch nicht ausreichend war. Die Notwendigkeit, in der Spurmitte zu fahren, wird durch das Gefahrenpotential ausgedrückt, das für diese Spur vergeben wird. Da dies für die Bahnplanung nicht das einzige Optimierungskriterium für die zu planende Bahn ist, besteht hier ein gewisser Spielraum. Die Auflösung der Gefahrenkarte trägt in Kombination mit den Diskretisierungsschritten des Bahnplaners ebenfalls zu solchen Effekten bei.



(a) *Überholen* auf dem Fahrzeug.

(b) *Spur folgen* auf dem Fahrzeug.

(c) *Spur wechseln* auf dem Fahrzeug.

(d) *Spur halten* auf dem Fahrzeug.

(e) *Abstand halten* auf dem Fahrzeug.

(f) *Kollision vermeiden* auf dem Fahrzeug.

Abbildung 6.11: Ergebnisse des Testszenarios *Überholen eines fahrenden Fahrzeugs* auf dem Versuchsträger.

Der Überholvorgang wurde im real durchgeführten Test im Vergleich zum Testszenario *Überholen eines statischen Hindernisses* nicht abgebrochen. Dies hat den Grund, dass die Wahrnehmungskomponente sich bewegende Fahrzeuge wesentlich zuverlässiger erkennt und insbesondere die Ausrichtung besser schätzen kann. Abbildung 6.9 zeigt die Gefahrenkarten verschiedener Verhalten zum Zeitpunkt T=16 s und das Ergebnis, wie es sich zu diesem Zeitpunkt für die Bahnplanung darstellt.

Testszenario *Fahrzeug folgen*

Beschreibung: Der Schwerpunkt bei diesem Szenario liegt auf der Schnittstelle zur Bahnplanung und auf einem Test derselben. Soll einem Fahrzeug mit definiertem Abstand gefolgt werden, so muss sich die durchgeführte Prädiktion korrekt in den Gefahrenkarten widerspiegeln. Dabei ist insbesondere die richtige Zuordnung vom Zeitpunkt der Kartenerstellung zum Zeitpunkt der Verwendung in der Bahnplanung wichtig. Darüber hinaus wird mit diesem Testszenario die korrekte Funktion des Verhaltens *Abstand halten* überprüft. Das in Abbildung 6.12 dargestellte Szenario zeigt ein langsam fahrendes Fahrzeug, welches nicht überholt werden kann und dem das Eigenfahrzeug folgen soll. Etwaige Überholversuche wurden durch entsprechende Parametrierung der Fahrspurberandung unterdrückt. Das voraus fahrende Fahrzeug führt mehrere Abbrems- und Beschleunigungsvorgänge aus.

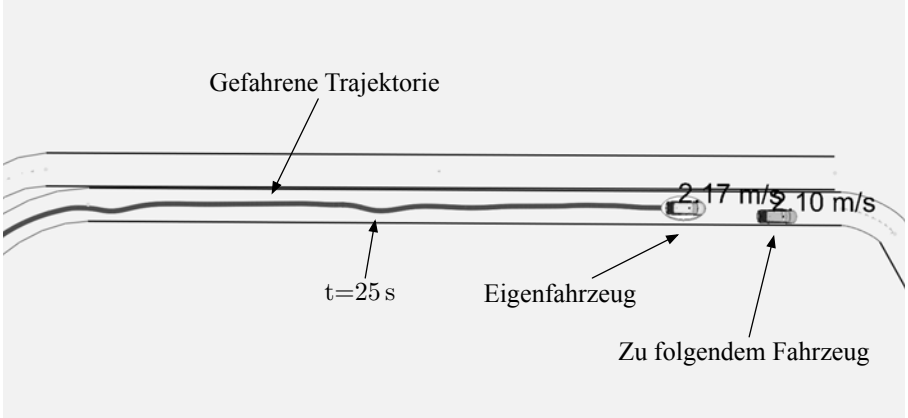
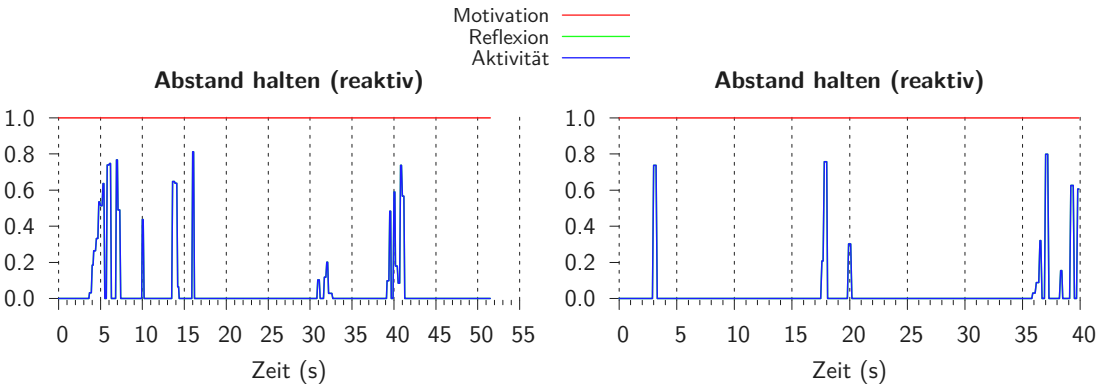


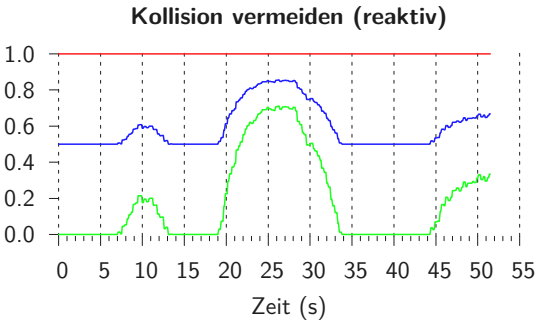
Abbildung 6.12: Bildschirmfoto des Testszenarios *Fahrzeug folgen* in der Simulation. Dargestellt sind das Straßennetz, das Fahrzeug dem gefolgt wird, das Eigenfahrzeug und dessen gefahrene Trajektorie.

Ergebnisse: Die Schaubilder mit Motivations-, Reflexions- und Aktivitätswerten der Verhalten, aus Simulation sowie realem Versuch, zeigt Abbildung 6.13. Es sind die Verläufe für die in diesem Versuch relevanten Verhalten und Fusionsknoten *Abstand halten*, *Kollision vermeiden* und *Geschwindigkeit fusionieren* dargestellt. Im Gegensatz zu den bisherigen Tests lassen sich die Ergebnisse aus Simulation und realem Versuch nicht direkt vergleichen. Dies hängt insbesondere mit den unterschiedlichen Fahrverhalten des vorausfahrenden

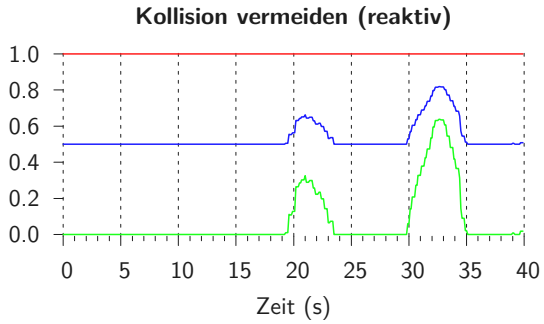


(a) *Abstand halten* in der Simulation.

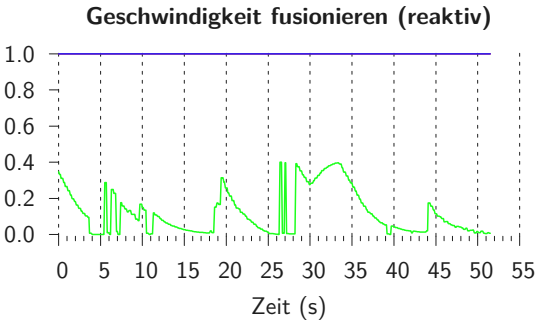
(b) *Abstand halten* auf dem Fahrzeug.



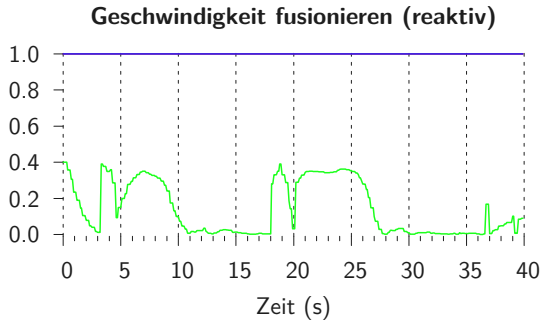
(c) *Kollision vermeiden* in der Simulation.



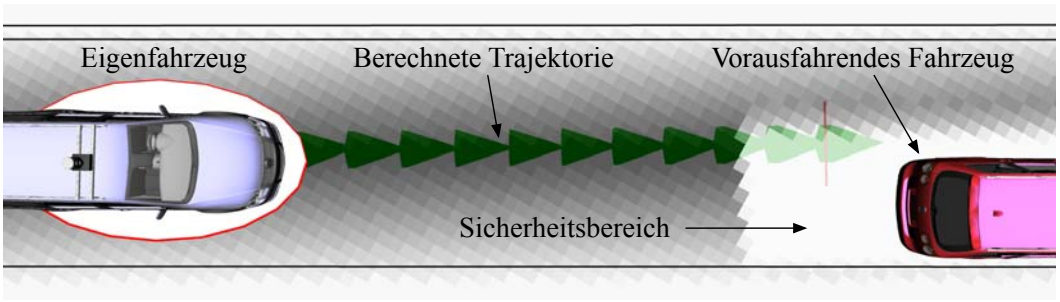
(d) *Kollision vermeiden* auf dem Fahrzeug.



(e) *Geschwindigkeit fusionieren* in der Simulation.



(f) *Geschwindigkeit fusionieren* auf dem Fahrzeug.



(g) Detaillierte Bildschirmaufnahme aus der Visualisierung. Dargestellt sind das Eigenfahrzeug, das vorausfahrende Fahrzeug und die vom Bahnplaner zu diesem Zeitpunkt erzeugte Trajektorie. Die aktuelle Gefahrenkarte ist der Szene hinterlegt. Schwarz bedeutet geringes, weiß bedeutet hohes Gefahrenpotential.

Abbildung 6.13: Ergebnisse des Testszenarios *Fahrzeug folgen*.

Fahrzeuges zusammen, hervorgerufen durch die schwierige Abstimmung mit einem menschlichen Fahrer. Dies wird anhand der unterschiedlichen Reflexionswerte von *Geschwindigkeit fusionieren* in Simulation und realem Versuch deutlich.

Das Verhalten *Kollision vermeiden* zeigt hohe Reflexionswerte in beiden Fällen, sobald sich das Eigenfahrzeug anderen Objekten zu stark nähert. In der aktuellen Simulationsumgebung können Fremdfahrzeuge lediglich pausiert werden, was einem unmittelbaren Anhalten gleichkommt. Als Folge entstehen in der Simulation höhere Reflexionswerte. Dieser Effekt wird in den Abbildungen 6.13e und 6.13f durch die unterschiedlich ausgebildeten Steigungsformen besonders deutlich. Die in Abbildung 6.13g gezeigte Trajektorie zum Zeitpunkt T=40 s führt augenscheinlich in das vorausfahrende Fahrzeug. Dies ist nur möglich, da die Fahrzeugposition mithilfe der dynamischen Gefahrenkarten korrekt in die Zukunft projiziert wird und sich zu den Zeitpunkten an Stützstellen der Bahn (grün markiert) kein Fahrzeug mehr aufhalten wird.

An zwei Stellen der in Abbildung 6.12 eingezeichneten Trajektorie sind kleinere Wellen zu erkennen (bspw. bei T=25 s). An diesen beiden Stellen hatte das vorausfahrende Fahrzeug zuvor angehalten und der Bahnplaner musste noch vor dem gewünschten Planungshorizont abbrechen. Die Wellen erklären sich mit dem aktuell gewählten Abbruchkriterium, bei dem versucht wird, einen möglichst großen kollisionsfreien Planungszeitraum zu erreichen. Hinsichtlich dieser Kriterien ist ein Pfad optimal, der an den Rand der Spur führt und so einen geringfügig längeren Weg aufweist.

Testszenario *Ausfall der taktischen und strategischen Schicht*

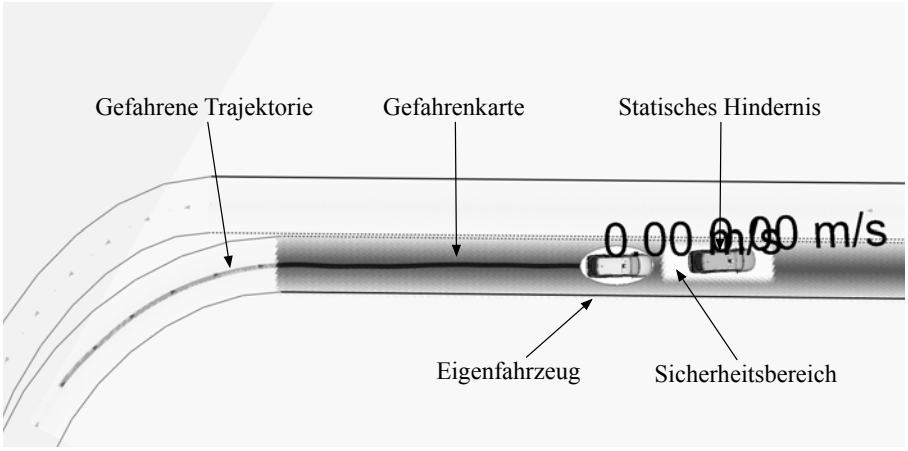
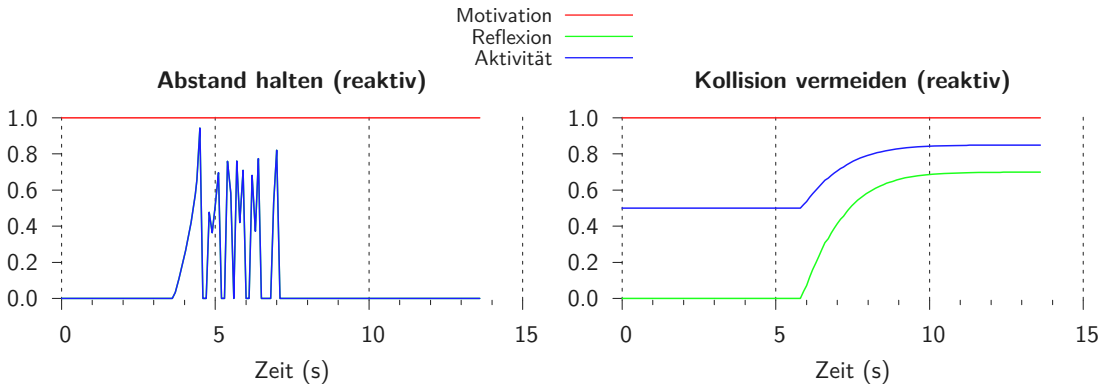


Abbildung 6.14: Bildschirmfoto des Testszenarios *Ausfall der taktischen und strategischen Schicht* in der Simulation.

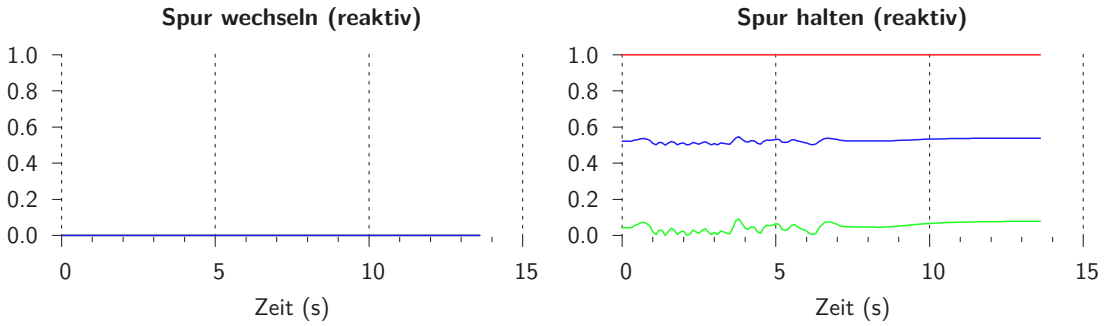
Beschreibung: Mit diesem Testszenario soll die an das Verhaltensnetzwerk gestellte Forderung nach Robustheit (Anforderung 3, vgl. Abschnitt 3.2) demonstriert werden.

Es soll gezeigt werden, dass das Fahrzeug auch bei Ausfall der Entscheidungsebene und wichtiger deliberativer Verhalten noch sicher geführt werden kann. Für diesen Versuch werden die strategische und die taktische Schicht des Verhaltensnetzwerkes vollständig deaktiviert. Das reaktive Verhalten *Spur halten* bleibt motiviert. Dies gilt ebenfalls für die dauerhaft aktivierten Sicherheitsverhalten *Abstand halten* und *Kollision vermeiden*. Auf der Eigenspur blockiert ein Fremdfahrzeug den Weg. Das Szenario ist vergleichbar mit *Überholen eines statischen Hindernisses*.

Ergebnisse: Abbildung 6.15 zeigt die Ergebnisse dieses Testszenarios, welches nur in der Simulation ausgeführt wurde. Eine Durchführung auf dem Versuchsträger wäre möglich, würde jedoch wegen der Loslösung von planenden Schichten für diesen Testfall keine anderen Ergebnisse liefern.



(a) *Abstand halten* in der Simulation. (b) *Kollision vermeiden* in der Simulation.



(c) *Spur wechseln* in der Simulation. (d) *Spur halten* in der Simulation.

Abbildung 6.15: Ergebnisse des Testszenarios *Ausfall der taktischen und strategischen Schicht*. Der Versuch wurde nur in der Simulation durchgeführt.

Aufgrund des Ausfalls des taktischen Verhaltens *Überholen* ist es nicht möglich, einen Spurwechsel auszuführen. Das zuständige Verhalten bleibt demotiviert (vgl. Abbildung 6.15). Das Verhalten *Abstand halten* wird beim Herannahen aktiv und bringt das Eigenfahrzeug in sicherem Abstand vor dem Hindernis zum Stehen. Das reaktive Verhalten *Spur halten* sorgt zudem dafür, dass die aktuelle Spur nicht verlassen wird. Das reaktive Verhalten *Kollision*

vermeiden wird durch die drohende Gefahr aktiv und erzeugt entsprechende Gefahrenkarten. Diese werden jedoch bei einem funktionsfähigen Verhalten *Abstand halten* nicht benötigt.

Testszenario *Ausfall eines reaktiven Verhaltens*

Beschreibung: Mit diesem Test soll gezeigt werden, dass auch bei einem fehlerhaften oder ausgefallenen reaktiven Verhalten noch ein sicheres Gesamtverhalten mit dem Verhaltensnetzwerk erzeugt werden kann. Dies gilt jedoch nicht für Sicherheitsverhalten wie *Kollision vermeiden* oder *Abstand halten*. Ein Ausfall der gesamten reaktiven Ebene oder der daran anschließenden Komponenten kann nicht in Kauf genommen werden, diese müssten für hohe Sicherheitsanforderungen redundant ausgelegt werden. Als Referenzsituation dient auch hier das Testszenario *Überholen eines statischen Hindernisses*. Das reaktive Verhalten *Spur wechseln* ist für die Durchführung des Überholvorganges besonders relevant und wird deshalb als Fehlerquelle gewählt. Es wird im Test derart modifiziert, dass es keine Ausgabe erzeugt. Die höher liegenden Verhalten der taktischen und strategischen Schicht wurden nicht verändert.

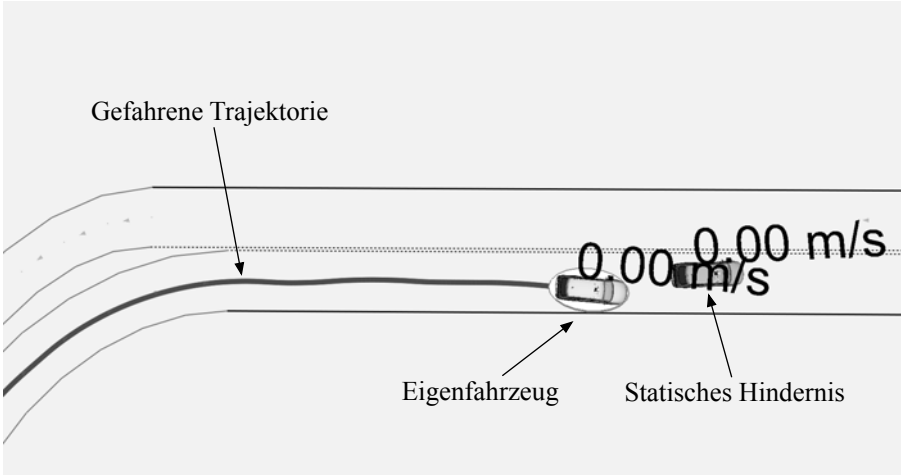
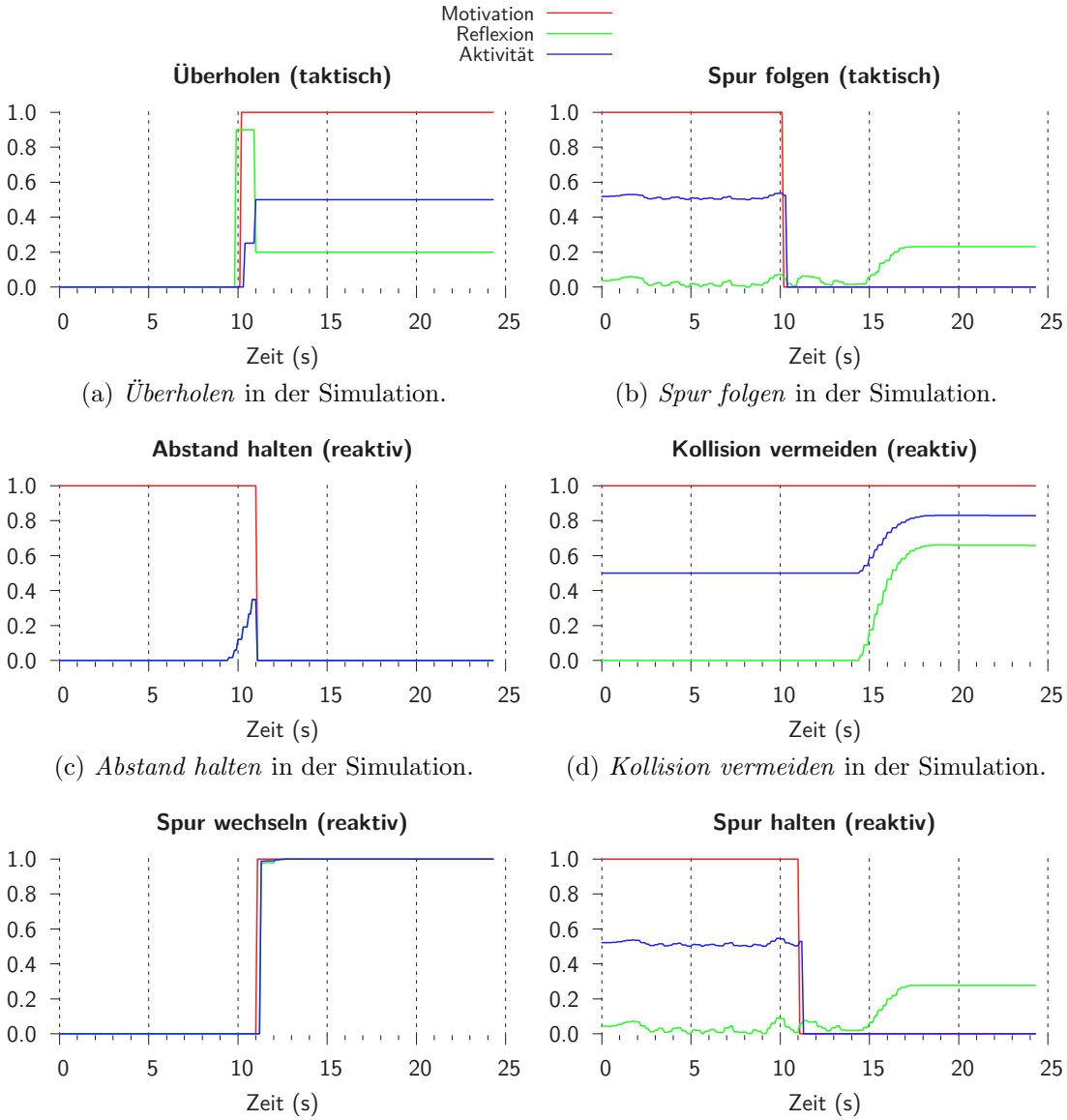


Abbildung 6.16: Szenario des Testfalls *Ausfall eines reaktiven Verhaltens*.

Ergebnisse: Der Testfall wurde nur in der Simulation ausgeführt, Abbildung 6.17 zeigt die Verlaufswerte für Motivation, Reflexion und Aktivität einiger ausgewählter Verhalten. In der ersten Phase des Überholvorganges motiviert das taktische Verhalten *Überholen* das reaktive Verhalten *Spur wechseln*. Letzteres erzeugt jedoch keine Gefahrenkarte, weshalb das Eigenfahrzeug auf der Eigenspur verbleibt. Als Folge des ausbleibenden Spurwechsels wird der Überholvorgang im weiteren Verlauf korrekterweise abgebrochen, wie die Reflexionswerte belegen. Das Verhalten *Abstand halten* wurde zu Beginn des Überholvorganges bereits demotiviert und kann deshalb das Eigenfahrzeug nicht wie im vorigen Testszenario zum Stehen bringen. In letzter Instanz verhindert das Sicherheitsverhalten *Kollision vermeiden*, dass die Fahrzeuge zusammenstoßen. Dieses Verhalten ist immer aktiv und kann von keinem anderen Verhalten unterdrückt werden.



(a) *Überholen* in der Simulation.

(b) *Spur folgen* in der Simulation.

(c) *Abstand halten* in der Simulation.

(d) *Kollision vermeiden* in der Simulation.

(e) *Spur wechseln* in der Simulation.

(f) *Spur halten* in der Simulation.

Abbildung 6.17: Ergebnisse des Testszenarios *Ausfall eines reaktiven Verhaltens*.

6.2 Evaluation

In diesem Abschnitt werden die Ergebnisse der in dieser Arbeit entwickelten Konzepte hinsichtlich der geforderten Aufgabenstellungen überprüft. In Abschnitt 6.2.1 wird dazu zunächst untersucht, inwieweit das Gesamtsystem *Verhaltenssteuerung – Bahnplanung – Regelung* die Zielsetzung dieser Arbeit erfüllt. Es wird auf einzelne Punkte der in Kapitel 1, Abschnitt 1.2 formulierten Zielsetzung Bezug genommen. In Abschnitt 6.2.2 wird speziell auf die Anforderungen eingegangen, welche in Abschnitt 3.2 an die Verhaltenssteuerung gestellt wurden.

6.2.1 Zielsetzung

Die zu Beginn der Arbeit in Abschnitt 1.2 formulierte Zielsetzung wird zur Überprüfung in diesem Abschnitt in ihre Einzelaspekte zerlegt.

> *„Die Zielsetzung dieser Arbeit besteht darin, eine Verhaltenssteuerung im Sinne eines kognitiven Automobils zu entwickeln. Aus Gründen der Nachvollziehbarkeit, dem besseren Verständnis und eventuell der späteren Möglichkeit eines Belehrens durch einen Fahrlehrer soll sich die vorgeschlagene Steuerungsarchitektur mit geeigneten Repräsentationen an die Bedürfnisse des Menschen anlehnen.“*

Die Umsetzung der Verhaltenssteuerung für das kognitive Automobil wurde mit biologisch motivierten Verhaltensnetzwerken durchgeführt. Mit dieser Architektur lassen sich menschliche Ausführungsarten von Verhalten (bewusst und unterbewusst) darstellen. Mechanismen der Datenfusion ermöglichen eine einheitliche Systemantwort als Ergebnis kooperativ arbeitender Verhalten. Weiterhin bieten Verhaltensnetzwerke durch die Parametrierbarkeit der Steuerfunktionen einzelner Verhalten vielfältige Anpassungsmöglichkeiten, die von Lernverfahren verwendet werden können. Um die Nachvollziehbarkeit im Ablauf der Verhaltenssteuerung sicherzustellen, wurden Fahrverhalten in einer dreischichtigen Struktur angesiedelt, wie sie auch für die Verhaltensmodellierung von Autofahrern verwendet wird.

Eine Identifikation von Einzelverhalten wurde anhand einer einfachen Klassifikation mittels Fragestellungen durchgeführt, die ein menschlicher Fahrer einfach beantworten kann. Dazu zählten unter anderem, welche Fahrverhalten bewusst oder unbewusst ausgeführt werden, welche zeitliche Relevanz sie besitzen oder welche Verhalten in Ausnahmesituationen erforderlich sind. Durch Einführung einer zentralen Entscheidungskomponente *Szenario Monitor* lassen sich Entscheidungen gezielt anhand der Wertigkeit von Überprüfungsprimitiven rekapitulieren.

> *„Weiterhin soll eine universell einsetzbare Bahnplanungskomponente entwickelt werden, welche sowohl im städtischen Verkehr, als auch auf Überlandstraßen und Autobahnen verwendet werden kann.“*

Es wurde ein Bahnplaner als separates Modul entwickelt, das in Verbindung mit der Verhaltenssteuerung eingesetzt wird und deren Vorgaben umsetzt. Die Bahnplanung integriert kinematische und dynamische Modelle und erzeugt jederzeit real fahrbare Pfade. Die

Ergebnisse zeigen, dass damit sowohl das Planen von geometrisch komplexen Manövern zur Durchführung einer Dreipunktwende und dem Einparken in verschiedenen Konfigurationen als auch eine Planung von Pfaden in dynamischer Umgebung möglich ist, wie sie bspw. zum Überholen benötigt werden. Eine Verifikation im Diskursbereich *Autobahn* war leider aus verkehrsrechtlichen Gründen nicht möglich. Eine Erweiterung hin zur Bewegungsplanung und der Erzeugung von Trajektorien eröffnet neue Möglichkeiten zur Planung zeitlich genau abgestimmter Manöver. Verschiedene Stufen des Planungszyklus stellen die schnelle Berechnung einer kollisionsfrei befahrbaren Bahn sicher, bevor eine optimierte Lösung erzeugt wird. Die in dieser Arbeit entwickelte Bahnplanungskomponente kam in leicht angepasster Form bei der Teilnahme des Team AnnieWay an der Urban Challenge erfolgreich zum Einsatz.

> *„Der Schnittstelle zwischen Verhaltenssteuerung und Bahnplanung kommt dabei eine besondere Bedeutung zu. Bei der Schnittstellenwahl ist insbesondere den Fragen der Sicherheit, der dynamischen Umfeldrepräsentation und der Fusion von Verhaltensausgaben Rechnung zu tragen.“*

Zur Umsetzung der Schnittstelle zwischen Verhaltenssteuerung und Bahnplanung wurde das Konzept der dynamischen Gefahrenkarten entwickelt. Damit besteht die Möglichkeit, eine zukünftige Schätzung des Fahrzeugumfeldes an die Bahnplanung weiterzureichen, und die darin enthaltene Dynamik angemessen zu repräsentieren. Erfolgreiche Versuche wie bspw. der Überholvorgang eines fahrenden Fahrzeuges haben gezeigt, dass das Konzept realisierbar ist, sich sehr gut für die Fusion kooperativer Verhaltensausgaben eignet und auch bei Ausfall einzelner Komponenten in der Verhaltenssteuerung noch ausreichende Sicherheit bietet. Durch die Rasterdarstellung der Gefahrenkarten lassen sich die Ausgaben der Verhaltenssteuerung leicht mit zellbasierten Schätzverfahren, wie dem *Bayesian Occupancy Filter*, kombinieren. Dies erlaubt es der Bahnplanung, neben einer Anbindung auf symbolischer Ebene, Daten über tieferliegende Schichten direkt von der Wahrnehmung zu beziehen und erhöht damit die Sicherheit des Gesamtsystems.

> *„Für die Wissenserweiterung ist eine geeignete Form zu finden, um abstrakte Messdaten als Erfahrungswissen abzulegen.“*

Eine Grund für die Auswahl von Verhaltensnetzwerken als Architektur der Verhaltenssteuerung war das Konzept der virtuellen Sensorwerte. Diese dienen nicht nur den Verhalten untereinander als Möglichkeit zur Koordination und Kooperation, sondern lassen sich aufgrund ihres bewertenden Charakters sehr gut als Erfahrungswerte heranziehen. Darüber hinaus liefern diese abstrakten Angaben über *Zufriedenheit* oder *Ausnutzung* dem Entwickler wertvolle Hinweise auf notwendige Nachbesserungen. Dazu wurde eine Methode der Datenaufbereitung mithilfe von sog. Beobachtern und dem Ablegen in einer semantischen Karte konzeptuell vorgestellt.

6.2.2 Anforderungen

Die vor dem Entwurf in Abschnitt 3.2 an die Verhaltenssteuerung gestellten Forderungen werden in diesem Abschnitt auf ihre Berücksichtigung hin überprüft.

1. Anforderung: Unterstützung vielfältiger Ziele. Mit der vorgestellten Architektur werden vielfältige Ziele mit unterschiedlichem Abstraktionsgraden unterstützt, indem langfristige Verhaltensweisen (strategische Verhalten), Handlungsfähigkeiten und -primitive (taktische und strategische Verhalten) zeitgleich eingesetzt werden. Des Weiteren werden auf Ebene der reaktiven Schicht mehrere Ziele gleichzeitig verfolgt, wie bspw. *Abstand halten* und *Spur folgen*. Die Fusion der vielfältigen Ziele reaktiver Verhalten wird für laterale Verhalten mithilfe von Gefahrenkarten durchgeführt, um ein gemeinsames Ziel zu bestimmen, das alle Fahrintentionen enthält. Die durchgeführten Tests konnten belegen, dass dieses Konzept tragfähig ist. Die longitudinalen Geschwindigkeitsvorgaben als zweiter Ausdruck einer Zielvorstellung werden mithilfe eines zugehörigen Fusionsknotens durch Minimumsfusion in eine gemeinsame Ausgabe überführt. Dies funktioniert in der momentanen Ausführung, da die reaktiven Verhalten lediglich eine Richtgeschwindigkeit vorgeben. Dynamische Fahrmanöver, wie *schnelle* Überholvorgänge oder aktives Ausweichen, in welchen nicht die minimale Geschwindigkeit gefahren werden kann, sind mit der aktuellen Fusionsmethode noch nicht möglich. Falls Verhalten zukünftig Geschwindigkeitsverläufe für dynamische Fahrmanöver vorgeben, müsste eine Priorisierung der reaktiven Verhalten erfolgen.

2. Anforderung: Unterstützung vielfältiger Sensoren. Aus den in der Einführung genannten Gründen ist eine Unterstützung vielfältiger Sensoren aus Sicht der Verhaltenssteuerung gegeben, so dass diese für eine Systemarchitektur allgemein formulierte Anforderung für die Verhaltenssteuerung nicht von Bedeutung ist.

3. Anforderung: Robustheit.

Datenfehlertoleranz: Eine Toleranz gegenüber fehlerhaften Sensordaten wird durch die vorgeschalteten Wahrnehmungs- und Interpretationskomponenten sichergestellt und muss von der Verhaltenssteuerung nicht in besonderem Maße berücksichtigt werden. Es wird davon ausgegangen, dass die Schnittstelle zur Interpretationskomponente so scharf formuliert ist, dass eine Toleranz bzgl. deren Daten nicht notwendig ist. Zur Durchführung der vorgestellten Testszenarien wurde Datentoleranz gegenüber einer fehlerhaft erkannten Ausrichtung von Fahrzeugen durch die zugehörige Wahrnehmungskomponente implementiert. Nur so konnte ein statisches Hindernis in Längsrichtung auf der Eigenspur korrekt verarbeitet werden. An anderer Stelle war dies nicht notwendig.

Rechenfehlertoleranz: Die Toleranz gegenüber Rechenfehlern ist anhand der Testszenarien *Ausfall der taktischen und strategischen Schicht* und *Ausfall eines reaktiven Verhaltens* ersichtlich. Während bei einem Ausfall der strategischen und taktischen Ebenen eine grundlegende Funktionalität durch reaktive Verhalten sichergestellt und das Fahrzeug weiterhin sicher geführt wird, so kann dies bei einem Ausfall der reaktiven Schicht nicht mehr gewährleistet werden. Der Ausfall eines einzelnen (nicht sicherheitskritischen) reaktiven Verhaltens

konnte jedoch noch toleriert werden. Auch wenn unterlagerte Sicherheitsmechanismen auf Bahnplanungs- und Regelungsebene realisiert wurden, so sollte für einen sicheren Betrieb zumindest die reaktive Ebene redundant ausgeführt werden.

Anpassungsvermögen: Ein hohes Maß an Anpassungsvermögen wurde insbesondere durch die schnell ausgeführten und teilweise dauerhaft aktiven Verhalten der reaktiven Ebene erreicht. In Situationen, in welchen durch fehlerhafte Wahrnehmungsdaten (Testszenario *Überholen eines statischen Hindernisses*) oder Ausfall von Komponenten (Testszenario *Ausfall eines reaktiven Verhaltens*) neue Umgebungsbedingungen aufgetreten sind, konnten diese Verhalten eine schnelle Anpassung vornehmen, um z. B. einen Überholvorgang abzubrechen. Eine ausreichend schnelle Aktualisierung und Verarbeitung von Daten war durch das System auch auf dem realen Versuchsträger jederzeit gegeben.

4. Anforderung: Erweiterbarkeit & Skalierbarkeit. Da das Verhaltensnetzwerk verschiedene Stufen der Entwicklung durchlaufen hat, und Handlungsfähigkeiten wie *Überholen* erst sehr spät integriert wurden, ist die grundsätzliche Erweiterbarkeit gegeben. Die Abhängigkeiten zwischen Verhalten untereinander wurden auf das Notwendigste beschränkt, um geeignete Voraussetzungen für eine Erweiterung zu schaffen. Die Hinterlegung von Überprüfungsprimitiven in Form von Listen im *Szenario Monitor* lassen eine einfache Erweiterung um weitere Verkehrsregeln zu. Die auf reaktiver Ebene entwickelten Repräsentations- und Fusionsmethoden ermöglichen zweifelsohne eine Integration zusätzlicher Verhalten. Ob die Anforderung an Erweiterbarkeit ausreichende Berücksichtigung fand, ist abschließend nur durch eine Nachfolgearbeit zu bewerten.

5. Anforderung: Parametrierbarkeit. Die Verhaltenssteuerung bietet die Möglichkeit, Regeln für die Überprüfung auf Durchführbarkeit von taktischen Verhalten zu hinterlegen. Diese in Form des Szenario Monitors hinterlegten Regeln dienen der Repräsentation der Straßenverkehrsordnung. Die Verhaltenssteuerung bietet darüber hinaus mit der Struktur von Verhaltensnetzwerken zahlreiche Ansatzpunkte, um die Art der Ausführung von Verhalten anzupassen. An erster Stelle seien die Berechnungsmethoden für virtuelle Sensorwerte genannt, an zweiter Stelle die Berechnungsfunktionen der Verhaltensausgaben. Das entworfene System genügt damit der Anforderung nach Parametrierbarkeit.

6. Anforderung: Nachvollziehbarkeit. Die Nachvollziehbarkeit ist erstens durch die transparente Entscheidungskomponente in Form des *Szenario Monitor* gegeben. Das Testen von taktischen Verhalten auf Durchführbarkeit mithilfe von aufgelisteten Überprüfungsprimitiven macht Entscheidungen im Detail ersichtlich. Zweitens dient die an die menschliche Vorstellung angelehnte Klassifikation von Fahrverhalten der besseren Nachvollziehbarkeit im Ablauf. Mithilfe von Aktivitätswerten lässt sich leicht nachvollziehen, welche Verhalten einen Beitrag zum Verhalten des Gesamtsystems leisten.

7. Anforderung: Testbarkeit und Simulationsunterstützung. Zur Erfüllung der Forderung nach Testbarkeit des Software-Systems wurde die in Abschnitt 3.4.3 vorgestellte Testumgebung entwickelt. Damit lassen sich gezielt einzelne Verhalten auf korrekte Funktionsfähigkeit

überprüfen, indem vorgefertigte Testfälle mit zugehörigen Daten in das Verhaltensnetzwerk eingespielt werden. Für einen umfangreichen Test des Verhaltensnetzwerkes in Kombination mit anderen Systemkomponenten wie Interpretation und Bahnplanung steht die im SFB/TR28 entwickelte Simulation zur Verfügung.

Die Verhaltenssteuerung wurde dahingehend an die Simulationsumgebung angepasst, dass Datenbankereignisse als Signalemitter für die Ausführungsschichten verwendet werden (vgl. Abschnitt 3.4.2). Weiterhin kann der interne Ausführungstakt in Form einer *virtuellen Simulationszeit* auch aus einem externen Datenbankobjekt bezogen werden. Dadurch lässt sich die Ausführung einzelner Schichten in einem Simulationslauf verlangsamen, wobei die zeitliche Koordination der Ausführungsfäden erhalten bleibt. Die Versuche dieses Kapitels zeigen, dass sich die Verhaltenssteuerung in der Simulation identisch mit einer realen Ausführung auf dem Versuchsträger verhält.

Zusammenfassend lässt sich feststellen, dass sich mit dem vorgestellten und umgesetzten Konzept Fahrmanöver ausführen lassen, wie sie auch für die Urban Challenge 2007 gefordert waren. Das entwickelte System kann sich demnach mit den existierenden, konventionellen Ansätzen messen, bietet darüber hinaus jedoch insbesondere Vorteile bei der Wissenserweiterung. Die formulierten Anforderungen an das System wurden weitestgehend erfüllt, wenngleich die Anforderung nach Erweiterbarkeit und Skalierbarkeit zu diesem Zeitpunkt noch nicht abschließend bewertet werden kann.

Kapitel 7

Schlussbetrachtungen

7.1 Zusammenfassung

Novität

Mit der Verwendung einer biologisch motivierten Verhaltensarchitektur zur Steuerung eines kognitiven Automobils wurde in dieser Arbeit eine neue Herangehensweise an das Problem der Auswahl und Ausführung von Fahrverhalten vorgestellt. Dieses Verfahren wurde bislang von anderen Forschern nicht veröffentlicht. Das Prinzip der virtuellen Sensoren stellt in Kombination mit semantischen Karten eine neue Grundlage dafür dar, Erfahrungswissen abzulegen und Fahrmanöver zu beurteilen.

Die Einführung dynamischer Gefahrenkarten erlaubt eine Darstellung der Umfeldentwicklung auf subsymbolischer Ebene und stellt ebenfalls eine Neuerung dar. In der Schnittstelle zwischen Verhaltenssteuerung und Bahnplanung wurde damit eine neue Form der Datenrepräsentation geschaffen, um Fahrintentionen und Hindernisinformation zu vereinen und einen Ort-Zeit-bezogenen Gefahrenwert zu repräsentieren. Kritische Bereiche lassen sich mit dieser Methode für die Planungskomponente vollständig sperren.

Mit dem Konzept der dynamischen Gefahrenkarten wurde zudem eine neue Möglichkeit vorgestellt, symbolische Daten der kognitiven Schicht mit Sensordaten der unteren Ebene zu fusionieren. Diese Form der Redundanz wird für einen sicheren Betrieb von kognitiven Automobilen als unbedingt notwendig erachtet. Das entwickelte Schätzverfahren erweitert bestehende Ansätze um eine Erreichbarkeitsbestimmung mittels Karteninformationen, um die Bewegung nicht klassifizierter Objekte zuverlässig vorherzusagen.

Vorteile gegenüber anderen Ansätzen

Ein grundlegender Vorteil des Gesamtsystems gegenüber anderen Systemen besteht in der Trennung von Situationsinterpretation und Verhaltensentscheidung. Während die Situationsinterpretation dadurch einerseits auch für Warnsysteme ohne nachgeschaltete Verhaltensentscheidung verwendet werden kann, so sind andererseits die Ergebnisse der Verhaltensentscheidung durch die präzise formulierte Schnittstelle exakt nachvollziehbar.

Über den aktuellen Zustand des Szenario Monitors und der darin abgelegten Überprüfungsprimitive lassen sich Gründe für eine Ablehnung oder Freigabe von Fahrhandlungen isoliert betrachten und überprüfen.

Die Verwendung biologisch motivierter Verhaltensnetzwerke führt durch das Prinzip der virtuellen Sensoren zu einem hohen Maß an Rückmeldung des Systems. Die Sensorwerte können z.B. dazu dienen, eine Systembewertung als Basis für Lernverfahren zu realisieren. Bei anderen Verhaltensmodellen wie bspw. endlichen Automaten ist dieses Prinzip der Rückmeldung einzelner Verhaltensaspekte nicht vorgesehen und eine Nutzung zum Zweck der Systembewertung deshalb nicht möglich. Die zwischen den Verhaltensbausteinen bestehenden Schnittstellen sind einheitlich definiert und vereinfachen dadurch eine Erweiterung um zusätzliche Verhalten. Bei anderen Systemarchitekturen wird dies durch dedizierte Schnittstellen oftmals erschwert.

Während andere Ansätze auf eine vollständige Repräsentation auf symbolischer Ebene vertrauen, um Fahrentscheidungen auszuführen, so setzt die in dieser Arbeit verwendete Kombination mehrerer Datenquellen dies nicht voraus. Das vorgestellten Verfahren zur Schätzung nicht-klassifizierter Objekte stellt die Planung hindernisfreier Pfade auch bei einer fehlerhaften Objekterkennung sicher.

Einschränkungen

Das Treffen und Ausführen komplexer Fahrentscheidungen ist bislang auf Manöver wie *Spur folgen, Ausweichen* oder *Überholen* beschränkt. Zur Umsetzung weiterer Fahrmanöver müssen zusätzliche strategische Verhalten implementiert und die Schnittstelle zur Situationsinterpretation erweitert werden. Ein Verarbeiten von Unsicherheiten im Szenario Monitor findet bisher nicht statt. Infolgedessen ist die Angabe einer *Sicherheit* bei Freigabe taktischer Verhalten nicht möglich.

Eine Bewegungsplanung ist bislang nur mit Vorgabe eines Geschwindigkeitsverlaufs von Seiten der Verhaltenssteuerung möglich. Die Aufnahme der Geschwindigkeit als weitere Suchdimension in der Planung ist im Rahmen dieser Arbeit nicht erfolgt. Die gegenwärtige Implementierung des BOFUM-Verfahrens ist nur bei Verwendung einer geringen Zellauflösung echtzeitfähig, liefert aber auch dann nur im Intervall von ca. 1 s neue Schätzungen. Für eine Integration in die Bahnplanungskomponente mit einer Ausführungsrate von 5 Hz ist dies noch zu wenig.

Die genannten Einschränkungen lassen sich durch eine Umsetzung der im Ausblick beschriebenen Anknüpfungspunkte auflösen.

7.2 Ausblick

Die vorliegende Arbeit schafft die Grundlage für weiterführende Studien im Bereich der Entscheidungsfindung für kognitive Automobile, der Modellierung von Fahrverhalten und der Anwendung von Lernverfahren. Für nachfolgende Arbeiten bieten sich insbesondere die folgenden Themen als Anknüpfungspunkte an:

- Eine Implementierung grundlegender Verhalten in das Verhaltensnetzwerk ist im Rahmen dieser Arbeit erfolgt. Eine Erweiterung um strategische Verhalten ist notwendig, um komplexe Verkehrssituationen bewältigen zu können und die Erweiterbarkeit des Ansatzes zu zeigen. Parallel muss dazu die Komponente der Situationsinterpretation weiterentwickelt werden, um mittelfristig Unsicherheiten zu liefern und Verhalten von Verkehrsteilnehmern zu identifizieren.

- Die Behandlung von Unsicherheiten wurde bislang in der Entscheidungskomponente ausgeklammert. Für zukünftige Systeme ist die Verarbeitung dieser Daten unbedingt notwendig, um ein Maß für die Sicherheit von Entscheidungen angeben zu können. Der Szenario Monitor muss dazu um die Fähigkeit der probabilistischen Entscheidungsfindung erweitert werden. Eine enge Abstimmung mit der Interpretationskomponente ist eine Voraussetzung dafür.

- Semantische Karten finden im automobilen Umfeld vermehrt Anwendung als metrisch-temporale Datenspeicher. Die Entscheidungskomponente für ein kognitives Automobil sollte verstärkt von dieser Repräsentationsform Gebrauch machen. Ein Schritt in diese Richtung wäre die Ausarbeitung der in dieser Arbeit aufgezeigten Möglichkeiten zur Hinterlegung von Prozessdaten und Erfahrungswissen.

- Lernverfahren werden als notwendig erachtet, um die Sicherheit von kognitiven Automobilen über ein verifiziertes Maß hinaus weiter zu erhöhen und so bspw. das Einsatzgebiet zu erweitern. Vor einer Anwendung von Lernverfahren sind zunächst Kriterien für eine Systembewertung zu erarbeiten. Weiterhin ist zu prüfen, in welchen Grenzen ein Automobil selbstständig lernen darf und welche Verfahren sich dafür eignen.

- Die in dieser Arbeit entwickelten Verfahren zur Bahn- und Bewegungsplanung sind in Kombination mit dem BOFUM-Schätzverfahren zu evaluieren. Für die Bewegungsplanung ist zu klären, ob die Geschwindigkeit als weitere Dimension in die Suche integriert werden kann. Die vorgestellte BOFUM-Implementierung ist bislang noch sehr zeitintensiv und sollte parallelisiert werden, um schnellere Wiederholraten zu ermöglichen.

Die aufgelisteten Vorschläge verfolgen das gemeinsame Ziel, die Sicherheit von Entscheidungen beziffern und nachweisen zu können und Verfahren zu entwickeln, um maschinelles Lernen in kognitiven Automobilen zu ermöglichen. Besonders die mit dem erstgenannten Ziel verbundene Aufgabe steht einer Markteinführung im Wege und ist vor einer möglichen Zulassung autonomer Systeme im Straßenverkehr zu lösen. Erst wenn dieses Problem gelöst ist, können kognitive Automobile dazu beitragen, die Anzahl an Unfällen im Straßenverkehr zu reduzieren.

Anhang A

Paradigmen in der Robotik

In der Robotik herrschte lange Zeit die Meinung vor, dass Systemarchitekturen aus drei funktionalen Elementen bestehen müssen: Einem Sensorsystem, einem Planungssystem und einem ausführenden System. Die Vorstellung, dass diese Elemente hierarchisch in einer festen Abfolge *Messen-Planen-Ausführen* zusammenarbeiten sollten, führte zur Umsetzung sogenannter funktionaler oder auch deliberativer Architekturen. Diese dominierten die Systemarchitekturen bis in die 80er Jahre, als der von Rodney Brooks vorgestellte Ansatz der reaktiven, verhaltensbasierten Robotersteuerungen eine große Begeisterung für dieses neue Konzept auslöste. In seinem Ansatz wurde das hierarchische Paradigma in Frage gestellt und auf einen Planungsschritt in einem reaktiven Paradigma schlichtweg verzichtet. Der Informationsfluss beschränkt sich in diesem Konzept auf *Messen-Ausführen*. Damit erfolgt eine sehr schnelle Systemantwort bei gleichzeitig geringer Komplexität der Steuerung. Abbildung A.1 zeigt den Informationsfluss für klassische, deliberative Systeme im Gegensatz zu reaktiven, verhaltensbasierten Ansätzen.

Obwohl mit dem reaktiven Ansatz erstaunliche Systemverhalten erzeugt wurden, so offenbarte der Verzicht auf die Planungskomponente auch erhebliche Schwächen. Nach dem ersten Begeisterungssturm folgte eine nüchterne Analyse mit dem Ergebnis, die Vorteile beider Architekturen in einer Mischform, den sogenannten hybriden Architekturen, zu vereinen. In diesem Konzept werden Planungsmodelle verwendet, sind aber keine alleinige Grundlage für ein funktionierendes System sondern nehmen reaktive Komponenten zu Hilfe.

Je nach Anwendungsfall finden sich bei heutigen Robotersystemen überwiegend hybride Systemarchitekturen mit reaktiver oder deliberativer Ausprägung. In den folgenden Abschnitten werden aus den jeweiligen Bereichen wichtige Vertreter vorgestellt, die den Stand der Technik wesentlich mitbestimmt haben und die besonders im Hinblick auf die Verwendung in der (auto)mobilen Robotik interessant sind.

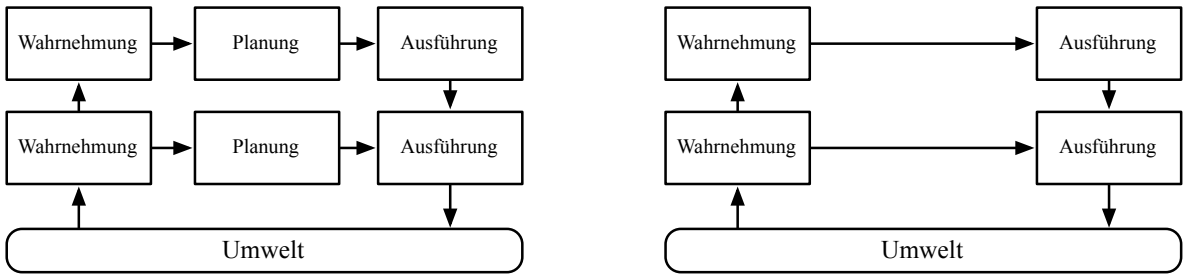


Abbildung A.1: Informationsfluß bei klassischen, deliberativen Kontrollsystemen (links) und reaktiven, verhaltensbasierten Systemen (rechts).

A.1 Funktionsbasierte, deliberative Ansätze

A.1.1 Shakey

Shakey wurde Ende der 60er Jahre am Stanford Research Institute entwickelt und war der erste mobile Roboter, der eigenständige Entscheidungen treffen konnte [Nilsson 69]. Shakey konnte selbst navigieren, farblich markierte Objekte erkennen und diese verschieben. Eine Fernsehkamera in Kombination mit einem Entfernungsmesser fungierte als Sensorik zur Objekterkennung. Die Planungsgrundlage für Entscheidungen war ein lokales Weltmodell, in welches erkannte Objekte eingefügt wurden.

Nach Nilsson benötigt ein Robotersystem bestimmte Fähigkeiten, welche sich in drei Klassen einteilen lassen und in Shakey Anwendung fanden:

1. Problemlösung: Eine Analyse der auszuführenden Aufgabe soll zu einer Sequenz von einfachen Handlungen führen. Die Folgen von Handlungen müssen dem Roboter für eine sinnvolle Problemlösung bekannt sein.

2. Modellierung: Wird zur Planung einer Sequenz von Aktionen benötigt, um ein vorgegebenes Ziel zu erreichen. Die Modelle müssen durch wechselnde Umgebungsbedingungen an neues Wissen angepasst werden.

3. Wahrnehmung: Mithilfe von visuellen Sensoren wird das lokale Weltmodell aufbereitet und eine Ausführung der geplanten Aktionen überwacht.

Zur Integration zahlreicher Fähigkeiten in ein System schlägt Nilsson vor, dieses in mehrere Aktionseinheiten zu zerlegen. Jede Aktionseinheit besitzt einen eigenen Problemlöser und ein Teilmodell. Sie sollte in der Lage sein, dem System eine bestimmte Funktionalität zur Verfügung zu stellen. Dabei ist es zudem möglich, andere Aktionseinheiten in den auszuführenden Plan zu integrieren.

Die Planungsaufgabe konnte dadurch in mehrere kleine Teilprobleme zerlegt werden. Dies war ein erster Ansatz, um die steigende Komplexität von Robotersystemen in den Griff zu bekommen. Die Architektur von Shakey sah vor, eine Einteilung von Funktionen in niedere

und höhere Schichten vorzunehmen. Niedere Funktionen arbeiten direkt auf den Aktoren, während höhere Funktionen komplexere Planungsaufgaben übernehmen und sich dafür einer mächtigen Problembeschreibungssprache bedienen. Die im NASREM-Modell formulierte Schichteneinteilung wurde im Grunde bereits bei Skahey in einfacher Form umgesetzt.

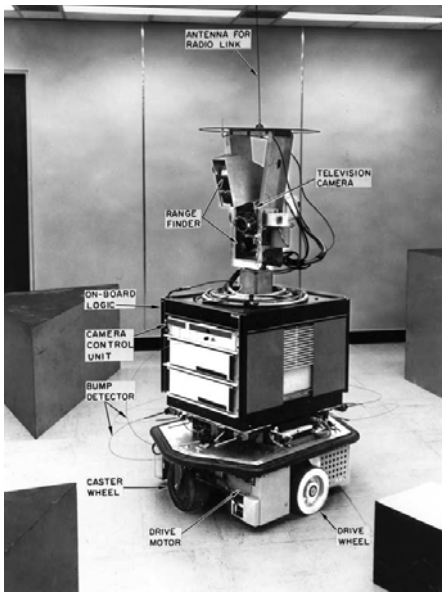


Abbildung A.2: Mobiler Roboter Shakey, Bildquelle: `http://www.sri.com`.

Durch dieses in jener Zeit aufsehenerregende Projekt ließen sich viele andere Forscher inspirieren. Der Entwicklung im Bereich der künstlichen Intelligenz und mobilen Robotik wurde dadurch ein großer Schub versetzt.

A.1.2 NASREM

Das *NASA Standard Reference Model (NASREM)* für die Architektur eines Teleroboter-Kontroll-Systems wurde Mitte der 80er Jahre im Auftrag der US-Regierung entwickelt [Albus 89]. Die Absicht des *National Institute for Standards and Technology (NIST)*[1] war es, für die NASA[2] eine Steuerungsarchitektur zu entwickeln, die verschiedene Zulieferer für das *Flight Telerobot Servicer Project (FTS)* einheitlich verwenden sollten. Das NASREM entwickelte sich als Standardmodell für eine hierarchisch funktionsorientierte Architektur.

Das Modell in Abbildung A.3 zeigt eine hierarchische Architektur mit horizontaler und vertikaler Aufteilung in mehrere Module. Vertikal wird zwischen den drei Stufen *Sensor Processing*, *World Modeling* und *Task Decomposition* unterschieden.

[1] Vgl. `http://www.nist.gov/`

[2] *National Aeronautics and Space Administration (NASA)*. Vgl. `http://www.nasa.gov/`.

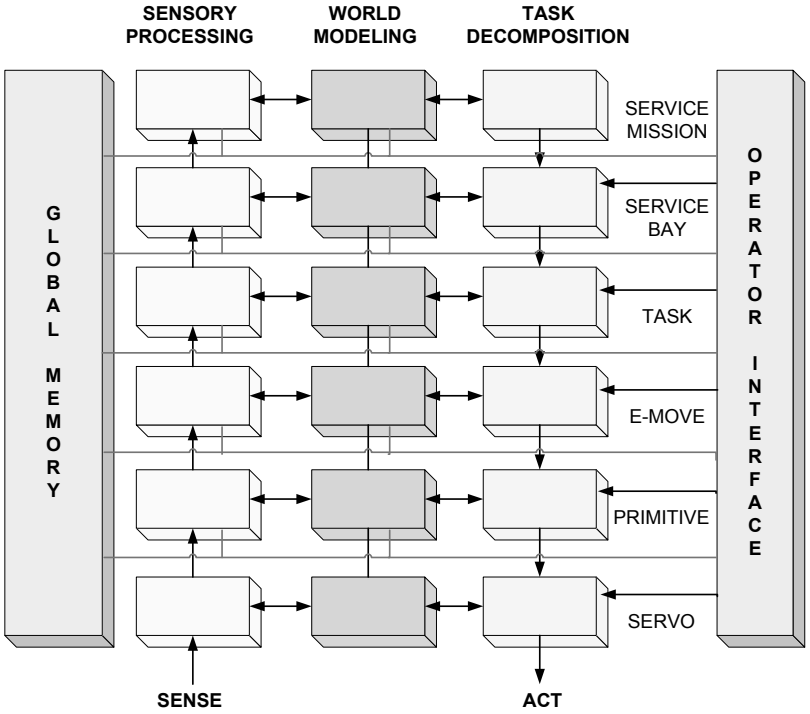


Abbildung A.3: Systemarchitektur des NASA Standard Reference Model (NASREM). Bildquelle: [Hoffmann 06].

Die in der Spalte *Sensory Processing* angeordneten Module haben die Aufgabe, Sensorinformationen aufzunehmen und Schritt für Schritt aufzubereiten. Dies beinhaltet das Melden von bestimmten Ereignissen, die Filterung von Sensordaten und die Bereitstellung eines Gütemaßes der Sensorinformationen – also einer qualitativen Bewertung der gemessenen oder aufbereiteten Informationen. Die *World-Modeling*-Module beziehen diese Sensorinformationen, um die Modelle auf verschiedenen Ebenen konsistent zu halten. Erwartungswerte oder Vorgaben können an die Sensormodule zurückfließen. Neben einem lokalen Weltmodell ist in diesen Modulen auch der Zustand des Systems in zeitlicher Abfolge hinterlegt. Durch prädiktive Fähigkeiten können so nicht nur aktuelle Informationen, sondern auch vergangene oder in gewissem Rahmen zu erwartende Informationen in Form von Modellen bereitgestellt werden. Diese Eigenschaft ist für die *Task-Decomposition*-Module zur Planung und Ausführung von Aktionen wesentlich. Deren Aufgabe ist es, komplexe Aktionen in einfachere Teilschritte zu zerlegen, diese Anweisungen in einem Ausführungsplan zusammenzustellen und dann stückweise bis zur Servoebene durchzureichen. Zur Ausübung dieses Aufgabenkomplexes besitzt jedes Modul der *Task Decomposition* eine Einheit zum Aufgabenmanagement, zur Planung und zur Ausführung.

Ein menschlicher Benutzer hat die Möglichkeit, über das *Operator Interface* auf verschiedenen Hierarchiestufen einzelne *Task-Decomposition*-Module anzusprechen. Damit lassen sich Planung und Ausführung beeinflussen oder für einzelne Module komplett übernehmen. Mit

dieser Möglichkeit können Teilaufgaben ausgelagert oder einzelne Module gezielt getestet werden. Die vertikale Einteilung in sechs Schichten spiegelt die strenge Hierarchie wider, in der kein Informationsfluss über Zwischenschichten hinweg erlaubt ist. In den vertikalen Ebenen werden die Module entsprechend ihrer Eigenschaften wie Antwortzeit, Fähigkeit zur Vorausschau und Abhängigkeit vom Weltmodell eingeordnet.

A.2 Reaktive, verhaltensbasierte Ansätze

Verhaltensbasierte Steuerungsarchitekturen stehen in ihrer Herangehensweise im Gegensatz zu klassischen Steuerungssystemen. Während dort nach der Wahrnehmung eine symbolische Darstellung der Umwelt als Planungsgrundlage erzeugt wird, so wird bei verhaltensbasierten Steuerungsarchitekturen versucht, die Abhängigkeit von einem Umweltmodell zu umgehen oder zumindest gering zu halten. Dies ist in gewissen Grenzen möglich. Je nach Grad der Unabhängigkeit wird oftmals eine weitere Klassifikation in deliberativ-basierte oder reaktiv-basierte verhaltensbasierte Steuerungsarchitekturen unternommen.

Dem Kybernetiker Valentino Braitenberg gelang es in den 80er Jahren, über direkte, hemmende oder treibende Verbindungen zwischen optischen Sensoren und Motoren komplexe Verhalten zu erzeugen. Mit der Veröffentlichung 1984 in seinem Buch *Vehicles* führte er die Senor-Aktor-Koppelung in der Robotik ein [Braitenberg 86]. Seine Experimente werden bis heute oft als Anschauungsbeispiele für verhaltensbasierte Steuerungen herangezogen. Neben W. Grey Walters, der 1953 die *Machina Speculatrix* vorstellte, darf Braitenberg als Vater der verhaltensbasierten Robotik bezeichnet werden. Mit Einführung der *Subsumption Architecture* durch Rodney Brooks erhielten die verhaltensbasierten Steuerungsarchitekturen endgültig ihren festen Platz in der Robotik [Brooks 86, Brooks 87]. Eine sehr gute Übersicht über verhaltensbasierte Steuerungen bietet das Buch von Ronald Arkin [Arkin 98].

Eine Schwierigkeit bei verhaltensbasierten Ansätzen besteht in der Fusion von Verhaltensausgaben. Durch die oftmals schwach ausgeprägte Hierarchie erzeugen mehrere Module Ausgaben auf derselben Ausführungsebene. In vielen verhaltensbasierten Architekturen wird diesem Problem dadurch begegnet, dass eine *kompetitive* oder *kooperative* Ausprägung gewählt wird. Damit wird festgelegt, ob nur eines oder mehrere Verhalten an der Ausführung beteiligt werden (vgl. Abildung A.4).

A.2.1 Subsumption Architecture

Die *Subsumption Architecture* (Subsumptionsarchitektur) wurde von Rodney Brooks Mitte der 80er Jahre eingeführt und hat über die Jahre vielen weiteren Architekturen wie bspw. auch den *Motor Schemas* als Vorlage gedient. Die Einzelverhalten sind verschiedenen Klassen zugeordnet und in einer hierarchischen Struktur angeordnet (vgl. Abbildung A.5). Höhere Schichten können die Verhaltensausgaben niederer Schichten unterdrücken. Es handelt sich daher um eine kompetitive Architektur, die dem *winner-takes-it-all*-Prinzip

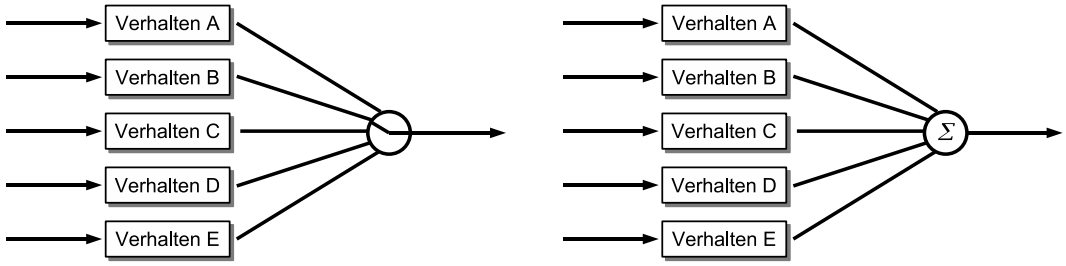


Abbildung A.4: Zusammenführung von Verhaltensausgaben bei einer kompetitiven Verhaltenssteuerung (links) und einer kooperativen Verhaltenssteuerung (rechts). Bildquelle: Albiez [Albiez 06].

folgt. Die Verhaltenscharakteristik wird mithilfe von endlichen Automaten modelliert.

Die Nachteile der Subsumptionsarchitektur liegen in der schlechten Skalierbarkeit. Da immer nur ein Verhalten für die Generierung von Aktorsignalen aus Sensorsignalen zuständig ist, muss in jedem Verhalten das gesamte Wissen über die zu steuernde Roboterarchitektur hinterlegt werden – eine komponentenspezifische Aufgabenteilung ist nicht möglich.

Nach Brooks wird die Subsumptionsarchitektur dennoch den folgenden vier Anforderungen an ein Robotersystem in hohem Maße gerecht:

1. *Multiple Goals*: Ein Roboter kann mehrere Ziele gleichzeitig verfolgen, die zum Teil auch gegensätzlich sein können. Damit ist es denkbar, dass ein Verhalten eine Zielfahrt plant, andere Verhalten jedoch aufgrund einer Hindenisvermeidung andere Zwischenziele anvisieren. Jede Schicht kann parallel ein anderes Ziel verfolgen. Der kompetitive Charakter sorgt letzten Endes dafür, dass eine Schicht zur Ausführung gelangt.

2. *Multiple Sensors*: Die Fusion mehrerer Sensoren stellt in der Robotik oftmals ein Problem dar. In der Subsumptionsarchitektur wird dieses Problem vernachlässigt oder zumindest abgeschwächt, da die Sensorinformationen nicht zentral zusammengeführt werden müssen, sondern jede Schicht nur die zur Ausführung notwendigen Informationen erhält. Ein Sensorwert kann dabei auch als Eingang für mehrere Schichten verwendet werden.

3. *Robustness*: Durch die Unabhängigkeit der einzelnen Schichten und die parallele Ausführung wird Redundanz erreicht und damit die Robustheit erhöht. Fällt eine höhere Schicht aus, so werden die Funktionsfähigkeiten zwar eingeschränkt, zumindest grundlegende Fähigkeiten sind aber durch niedere Schichten weiterhin bereitgestellt.

4. *Extensibility*: Eine Erweiterung um zusätzliche Schichten kann leicht durchgeführt werden ohne existierende Schichten zu beeinflussen. Dieser Umstand ist für das Lösen komplexerer Aufgaben notwendig oder auch, um zusätzliche Sensorik anzusteuern.

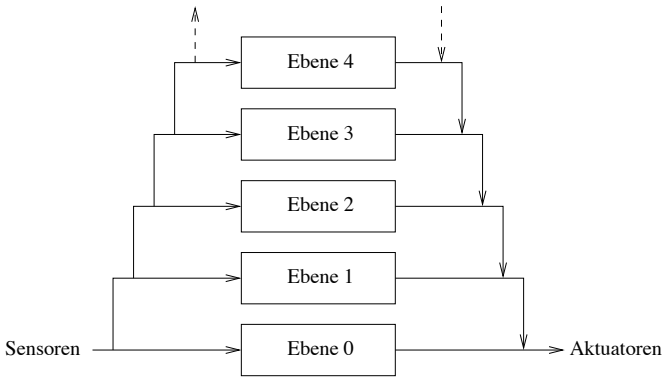


Abbildung A.5: Funktionsweise der Subsumptionsarchitektur: Ausgaben höherer Schichten unterdrücken niedere Verhalten. Bildquelle: Albiez [Albiez 06].

Trotz der bekannten Schwachpunkte konnten mit dieser Steuerungsarchitektur auf den Sechsbeinern *Attila* und *Ghengis* beachtliche Erfolge erzielt werden [Ferrel 95]. Damit wurde letztendlich die bis dahin vorherrschende Meinung widerlegt, dass komplexes Verhalten das Ergebnis eines komplexen Steuerungssystems sein muss und gezeigt, dass nicht zwangsläufig ein akkurates Umweltmodell notwendig ist.

A.2.2 Motor Schemas

Die *Motor Schemas* von Ronald Arkin sind eng mit der Subsumptionsarchitektur verwandt, unterscheiden sich aber über die Art der Fusion von Ausgabedaten [Arkin 87, Arkin 89, Arkin 92]. Verschiedene Einheiten, als Motor Schemas bezeichnet, generieren eine vektorielle Ausgabe, die als Summe an die Aktoren weitergeleitet wird. Eine Unterdrückung von Verhalten findet hier nicht statt – es handelt sich folglich um einen kooperativen Ansatz.

Mehr als die Subsumptionsarchitektur orientieren sich die Motor Schemas an der Biologie, um Paradigmen wie die aktionsgesteuerte Wahrnehmung (*action-oriented perception*) umzusetzen. Mit Verwendung der Schema-Theorie versucht Arkin, die Gemeinsamkeiten von neurobiologisch getriebenen und künstlichen Verhalten in einer Architektur abzubilden. Ziel ist es dabei, durch das Modell Verhaltensprimitive für ein Robotersystem zur Verfügung zu stellen.
Ein Motorschema besteht aus mehreren *Perceptual Schemas* und *Perceptual Subschemas*, die von Sensoren mit Daten versorgt werden und einem mehrdimensionalen Potenzialfeld. In diesem Potenzialfeld wird von den Perceptual Schemas ein Punkt ausgewählt, der den Motorschemas zur Erzeugung der Ausgabe dient. Ein Vorteil von Motorschemas ist die Integration von Hintergrundwissen in die Perceptual Schemas.

Nach Arkin unterscheiden sich Motor Schemas insbesondere durch folgende Punkte von anderen Ansätzen:

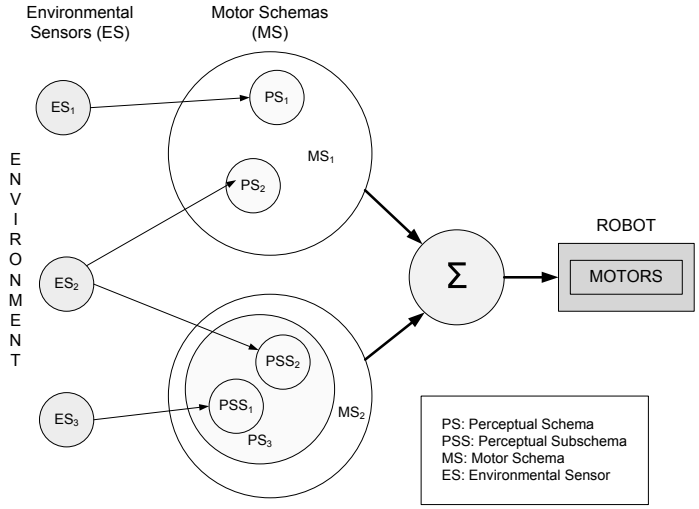


Abbildung A.6: Systemarchitektur der *Motor Schemas*. *Perceptual Schemas* sind den *Motor Schemas* unterlagert und versorgen diese mit Umgebungsinformationen. Bildquelle: [Hoffmann 06].

1. Die Verhaltensausgaben erfolgen in einem einheitlichen Vektorformat.

2. Die Koordination erfolgt durch Vektoraddition.

3. Es existiert keine vordefinierte Hierarchie. Die Verhalten werden zur Laufzeit konfiguriert und bilden eine temporale Struktur. Eine Umordnung zur Laufzeit kann aufgrund von Änderungen in der Umgebung, der Absicht oder verfügbarer Ressourcen geschehen.

4. Es gibt keine Entscheidungsinstanz für die Fusion der Ausgaben. Jedes Verhalten trägt zum Ergebnis bei.

5. Unsicherheiten von Sensordaten können in der Verhaltensausgabe wiedergegeben werden.

Das Konzept von Motor Schemas zeigt deutlich, dass die Architektur zur Steuerung von mobilen Robotern entwickelt wurde und sich deshalb nur bedingt zur Steuerung anderer Systeme eignet. Die Architektur kam in vielen mobilen Robotersystemen am *Georgia Tech Mobile Robot Laboratory* [GT-MRL 08] zum Einsatz, unter anderem in HARV, George [Arkin 90] oder Callisto [Balch 95].

A.3 Hybride Ansätze

A.3.1 Verhaltensnetzwerke

Die sogenannten Verhaltensnetzwerke stellen neben den Motor Schemas eine weitere Variante der biologisch motivierten Steuerungsarchitekturen dar. Allerdings sind durch die Hierarchisierung auch planende Einheiten vorgesehen, wodurch diese Architektur den hybriden Ansätzen zuzuordnen ist. In [Albiez 06] entwirft und verwendet Jan Albiez Verhaltensnetzwerke für die Steuerung einer Laufmaschine und orientiert sich dabei an der Struktur des zentralen Nervensystems [Albiez 01b, Albiez 01a]. Die drei Kernpunkte, die Albiez aus der biologischen Forschung als unbedingt notwendig für eine Architektur zur Steuerung einer Laufmaschine erachtet, sind:

1. Ein hierarchisches System mit direkten Verbindungen zwischen hohen und niederen Schichten.

2. Die kinematotopische Struktur des Roboters muss sich bei der Anordnung von Verhalten widerspiegeln.

3. Ein kompetitiver Antagonismus als grundlegende Regelungsstruktur.

Auf diesen Punkten baut Albiez eine schwach-temporalhierarchische Verhaltenssteuerung auf. Die schwache Hierarchie setzt die erste Forderung um und erlaubt eine übergreifende Verbindung von Verhalten auch über Zwischenschichten hinweg. Die temporalen Eigenschaften resultieren aus einer stetigen Änderung der aktiven Struktur zur Laufzeit. Obgleich die Verbindungen zwischen Verhalten statischer Natur sind, so werden diese über ein Steuersignal motiviert oder de-motiviert. Dies kommt einer strukturellen Anpassung des Verhaltensnetzwerkes gleich.

Der Kern eines Verhaltensbausteins ist seine Übertragungsfunktion $F(e)$, welche die Sensordaten e in Ausgaben u umwandelt (vgl. Abbildung A.7). Für die Art der Übertragungsfunktion werden keine Einschränkungen vorgenommen – es ist das für diesen Baustein geeignetste Verfahren zu wählen. Während auf höheren Ebenen neuronale Netze oder endliche Automaten zum Einsatz kommen, so sind klassische PID-Regler oder andere Regelungsstrategien auf niederen Ebenen denkbar. Die Platzierung von Verhalten in den jeweiligen Schichten erfolgt gemäß den in der Natur zu beobachtenden Arten von Verhalten. Planende Verhalten mit mittel- oder langfristigem Horizont sind in höheren Schichten angesiedelt, reflexive Verhalten mit schnellem Antwortverhalten und kurzer Aktivitätsdauer in niederen Schichten, dazwischen existieren Mischformen.

Eine wichtige und im Vergleich zu anderen Architekturen besondere Eigenschaft ist das Prinzip der virtuellen Sensoren. Neben der Motivation, welche als Steuereingang dient, werden von jedem Verhalten Werte für Reflexion und Aktivität als Steuersignale ausgegeben. Diese beiden Signale spiegeln den internen Zustand des Verhaltens wider: Die Reflexion eine Art Zufriedenheit, mit welcher die Abweichung von einem Zielzustand angegeben wird.

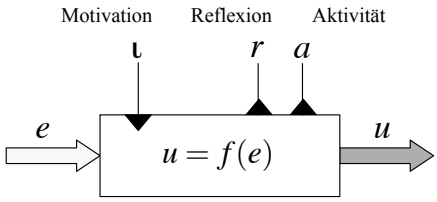


Abbildung A.7: Schema eines Verhaltensbausteins mit Ein- und Ausgabedaten e und u. Die Steuersignale ι, r und a dienen der Abstimmung mit anderen Verhaltensbausteinen.

Die Aktivität ist eine Maßzahl für die Leistung, die der Baustein momentan, relativ zu einer Maximalleistung, erbringt. Motivation, Reflexion und Aktivität sind als $\iota, r, a \in [0, 1]$ definiert. Durch die standardisierten Schnittstellen bleiben die Signale austauschbar und sind für alle Verhaltensbausteine verwendbar, ohne Wissen über die interne Struktur des Signalerzeugers zu benötigen.

Virtuelle Sensoren sind für die Koppelung von Verhalten wichtig, um komplementäre oder redundante Strukturen zu schaffen sowie für die Fusion von Verhaltensausgaben. Abbildung A.8 zeigt einen Fusionsknoten als weiteres Grundelement in Verhaltensnetzwerken. Grundlage für eine Datenfusion kann die Aktivität der Verhalten sein, welche eine Art Dringlichkeit für die Weiterleitung der Daten darstellt. Durch Einsatz entsprechender Fusionsknoten können innerhalb des Verhaltensnetzwerkes kompetitive oder kooperative Gruppen angelegt werden.

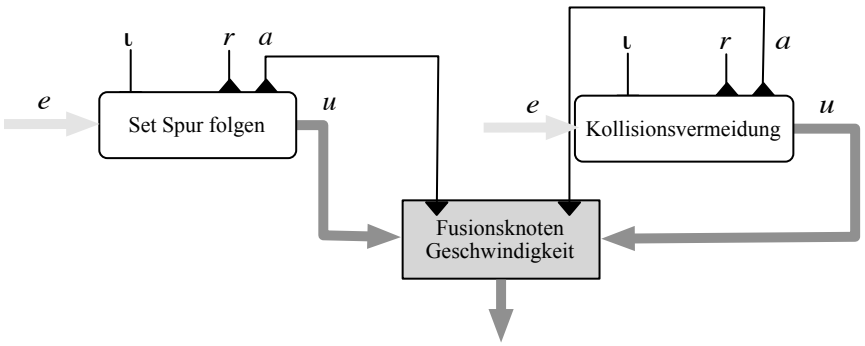


Abbildung A.8: Fusion von Ausgaben verschiedener Verhaltensbausteine. In diesem Fall dient die Aktivität als Kriterium für eine Gewichtung.

Ein weiterer wichtiger Aspekt macht das Prinzip der virtuellen Sensoren besonders wertvoll: Durch die detaillierte Aufschlüsselung der systeminternen Abläufe wird die Nachvollziehbarkeit für den Entwickler vereinfacht, indem nicht nur das Ergebnis selbst sichtbar wird, sondern charakteristische Eigenschaften bereits während der Ausführung. Eine dauerhaft

hohe Unzufriedenheit einzelner Verhalten kann beispielsweise auf ein Sicherheitsdefizit hindeuten. Für das System selbst bieten virtuelle Sensoren die Möglichkeit einer Bewertung der Verhaltensausführung, welche als Grundlage für die Anwendung von Lernverfahren eingesetzt werden kann.

A.3.2 JPL Exploratory Robot Architecture

Bei dieser von E. Gat am JPL[3] entwickelten Architektur handelt es sich um eine Mischform mit deliberativer Ausprägung [Gat 90]. Ursprünglich wurde dieses Steuerungsprinzip entwickelt, um einem auf einem entfernten Planeten befindlichen Roboter ein Maß an Eigenständigkeit zu verleihen – für reine Teleoperation sind die Laufzeiten dorthin zu lang. Abbildung A.9 zeigt die Architektur in der Übersicht mit den wichtigsten Modulen für Wahrnehmung, Pfadplanung, Ausführungs-Kontroll-Planer und dem Ausführungssystem.

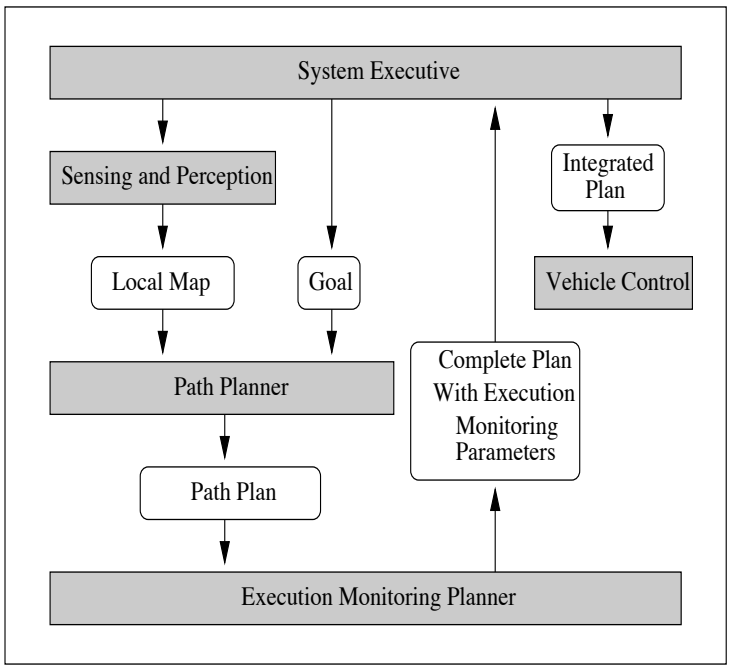


Abbildung A.9: Darstellung der *JPL Exploratory Robot Architecture*. Bildquelle: [Gat 90].

Das Wahrnehmungsmodul hat zur Aufgabe, eine lokale Karte der Umgebung zu erstellen, welche dann genutzt wird, um einen Pfad zu planen. Aus diesem erstellt der Ausführungs-Kontroll-Planer (AKP) wiederum einen parametrisierten Bewegungsplan. Der AKP greift dabei auf eine Simulation zurück, welche mögliche Abweichungen vom festgelegten Pfad antizipiert und korrigierend eingreifen kann, um die Ausführungsparameter bspw. entsprechend der Beschaffenheit des Untergrundes zu variieren.

[3] *Jet Propulsion Laboratory*, Pasadena, Kalifornien.

Die starke Orientierung am *Sense-Model-Plan-Act*-Paradigma mit sequentiellem Informationsfluss liefert als Ausgabe einen vollständigen Plan mit Ausführungsparametern, der nur noch sehr schwach angepasst werden kann. Die Integration von reaktiven Eigenschaften beschränkt sich deshalb auf sehr einfache Funktionen, um den Roboter anzuhalten oder auf eine sichere Position zu fahren.

Die Ideen wurden später in der Architektur ATLANTIS weiterentwickelt [Gat TA], welche auf der Planetenerkundungsplattform *Robby* [Gat 91] und im Rahmen des *NASA Path Finder Program* [Shirley 95] eingesetzt wurde.

A.3.3 AuRA-Architektur

Die AuRA-Architektur von R. Arkin basiert größtenteils auf den von ihm entwickelten *Motor Schemas* [Arkin 89, Arkin 87]. AuRA erweitert dieses reaktive Prinzip um eine hierarchische Komponente zur Abwicklung von Modellierungs- und Planungsaufgaben [Arkin 97]. Abbildung A.10 zeigt eine schematische Darstellung der AuRA-Architektur, die sich aus einem hierarchischen und einem reaktiven Teil zusammensetzt.

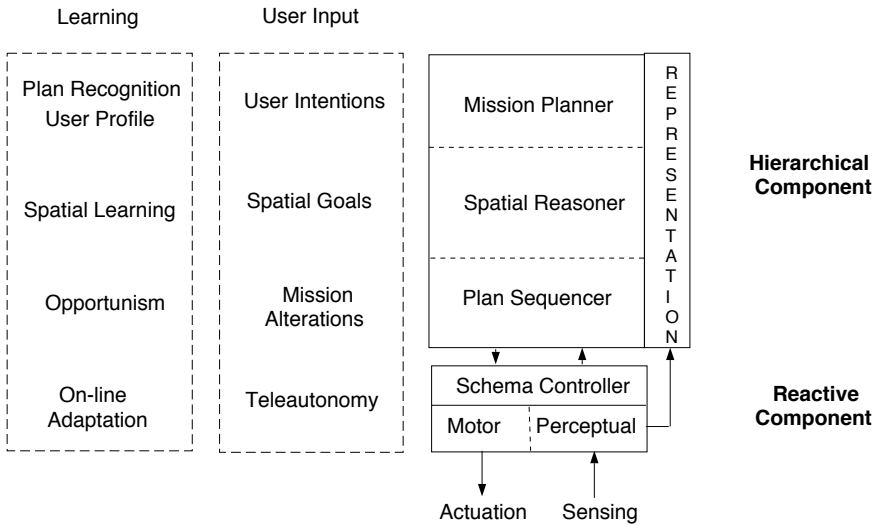


Abbildung A.10: *Aura*-Architektur als Erweiterung der *Motor Schemas*. Bildquelle: [Arkin 97].

In der hierarchischen Komponente finden sich die drei klassischen Ebenen wieder: Ein Missionsplaner zur Erzeugung einer Sequenz von Zwischenzielen, ein Navigator, welcher unter Zuhilfenahme einer Karte fahrbare Pfade erzeugt und letztlich eine Ablaufsteuerung, welche geeignete Verhalten auswählt und parametriert. Diese Verhalten, hinterlegt in *Motor Schemas*, liefern eine vektorielle Ausgabe, aus der durch gewichtete Fusion die notwendigen Reglerwerte abgeleitet werden. Die ebenfalls von der Ablaufsteuerung aktivierten *Perceptual*

Schemas liefern Informationen an höhere Ebenen zurück, um den Fortschritt der Mission zu überwachen. Bei unvorhergesehenen Ereignissen kann der Navigator den Plan verwerfen und eine neue Folge von Zielpunkten und zugehörigen Verhalten erstellen. Durch die relativ starke Koppelung von Navigator und Verhalten erfordert dies eine vollständige Neuplanung. Eine Abänderung oder Anpassung bestehender Vorgaben ist nicht möglich.

Durch die Verwendung der *Motor Schemas* wird das Problem der Vorhersagbarkeit und der Fusion mehrerer Verhaltensausgaben vererbt. Dies ist wohl der Grund, weshalb sich die Nutzung bislang auf mobile Plattformen mit einfacher Missionskomplexität beschränkt hat. Der Gewinn des *Clean-up-the-Office*-Wettbewerbs mit den Roboterplattformen *Ganymede* und *Callisto* zeigt jedoch, dass das Konzept für typische Aufgabenstellungen der mobilen Robotik einsetzbar ist [Simmons 95].

Sharp-Architektur

Die Sharp-Architektur wurde 1998 von Christian Laugier und Kollegen an INRIA[4] entwickelt, um eine Bewegungsplanung für ein Straßenfahrzeug unter Berücksichtigung der besonderen Kinematik durchzuführen [Laugier 89, Laugier 01]. Sharp besitzt eine Dreischicht-Architektur, welche einen Missionsplaner, einen Trajektorienplaner und einen Bewegungscontroller beinhaltet (vgl. Abbildung A.11).

Um von Seiten des Missionsplaners einen Ablaufplan zu erstellen, wurde das Konzept der *Sensor-Based Maneuver (SMB)* eingeführt. Der Missionsplaner kombiniert dabei geplante Trajektorien mit einzelnen SMBs, welche unterschiedliche Beschreibungen für Fertigkeiten darstellen und durch Parametrierung zur Laufzeit instanziiert werden. Die Erkenntnis, dass in der Realität nicht sehr viele Manöver unterschiedlichen Typs durchgeführt werden und sich der Unterschied oftmals nur auf eine andere Parametrierung beschränkt, führte zu dem Entschluss, SMBs als Meta-Skills für eine Beschreibung zu verwenden.

Ein SBM wird dabei durch ein Skript als Abfolge von einzelnen Wahrnehmungs- und Steuerungsfertigkeiten beschrieben und orientiert sich am von Rich und Knight entwickelten *script*-Paradigma [Rich 83]. Dadurch entstehen zur Laufzeit Verhalten mit reaktiver Ausprägung, welche robust und flexibel sind, um sich neuen Gegebenheiten während einer Handlungsausführung anzupassen. Die Sharp-Architektur wird an INRIA in den Versuchsplattformen *Ligier* und *Cycab* eingesetzt [Laugier 01].

A.4 Fazit

Im Hinblick auf eine mögliche Architektur für ein kognitives Automobil bleibt festzustellen, dass dafür eine stark deliberative Ausprägung mit separaten Planungs- und Entscheidungskomponenten unbedingt notwendig ist. Nur so lassen sich die verschiedenen Formen von Hintergrundwissen (Objektinformationen, Straßenkarten, Verkehrsregeln) bestmöglichst

[4] *Institut National de Recherche en Informatique et en Automatique.* Toulouse, Frankreich.

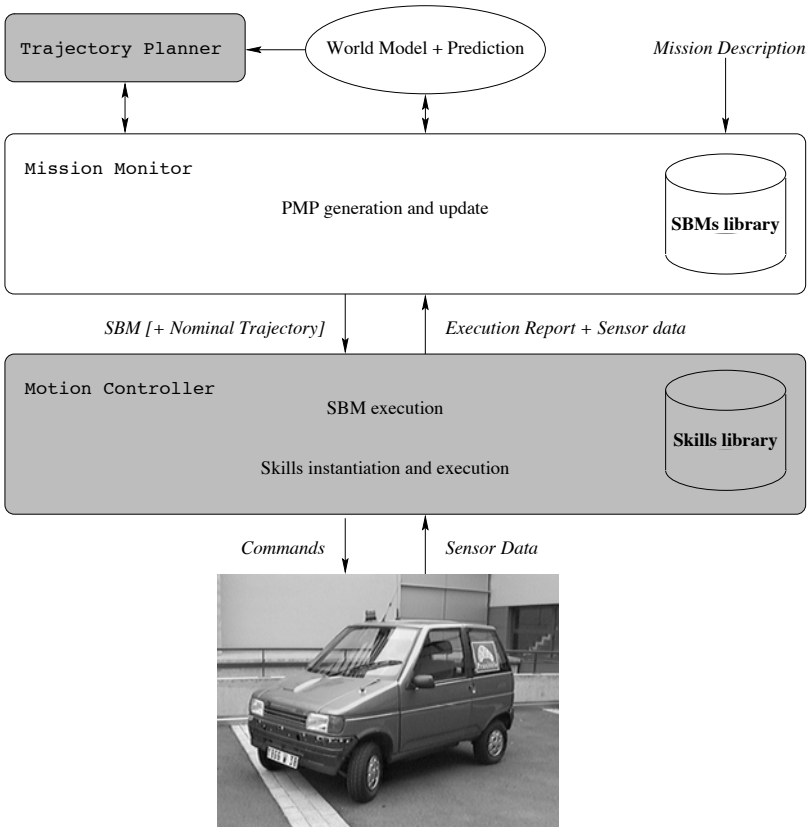


Abbildung A.11: Sharp-Architektur des INRIA. Bildquelle: [Laugier 01].

nutzen. Das Paradigma *Die Welt als Modell* greift in dieser komplexen Umgebung nicht, und die geforderte Zuverlässigkeit und Sicherheit für den Straßenverkehr kann nur unter Berücksichtigung von umfangreichem Modellwissen stattfinden.

Die Nachvollziehbarkeit von Entscheidungen ist ein wichtiger Punkt bei sicherheitskritischen Systemen. Dies legt den Schluss nahe, eine zentrale, hierarchische Entscheidungskomponente vorzusehen, und eine Entscheidung nicht durch konkurrierende Elemente auf einer Ebene herbeizuführen. Die Integration reaktiver Komponenten kann helfen, die Antwortzeit in kritischen Situationen und damit auch die Sicherheit zu erhöhen. Im Gegensatz zur Entscheidungsebene sind in der Ausführungsebene reaktive Elemente mit konkurrierenden oder kooperierenden Ausprägungen denkbar. Ein hybrides Konzept mit deliberativer Ausrichtung unter Verwendung reaktiver Elemente in der Ausführung scheint die zielführende Architektur für ein autonomes Straßenfahrzeug zu sein.

Anhang B

Wissensmodellierung und Entscheidungsfindung

Von einem Robotersystem mit deliberativem Charakter wird erwartet, dass es nicht nur reagiert, sondern wohlüberlegte Entscheidungen trifft. In diesem Zusammenhang kann man auch von einem kognitiven System sprechen, also einem System, welches seine Entscheidungen auf eine Erkenntnis stellt, die aus Beobachtungen und der Verwendung von Wissen resultieren. Solche Systeme werden oftmals auch als *intelligent* bezeichnet, was zeigt, dass versucht wird, die ursprünglich dem Menschen eigene Fähigkeit der Intelligenz auch auf Robotersysteme zu übertragen.

Zunächst herrschte mit dem Aufkommen von schnellen Rechenmaschinen die Meinung vor, Intelligenz wäre alleine durch ein Ausprobieren der kombinatorischen Vielfalt zu realisieren. Dies stellte sich jedoch als Irrtum heraus, da die Anzahl an Möglichkeiten die Rechenleistung bei Weitem übersteigt. Die Vielzahl an Varianten muss durch geeignete Verfahren zunächst reduziert werden, bevor ein Vergleich vorgenommen werden kann. Die Probleme und ihre Lösungsansätze wurden unter dem Begriff der *Künstlichen Intelligenz* zusammengefasst. Als Teilgebiet der Informatik wird in diesem Forschungsfeld versucht, Lösungsmöglichkeiten durch die Verwendung von zusätzlichem Wissen einzuschränken.

Soll Wissen bei der Entscheidungsfindung verwendet werden, so ist zunächst zu untersuchen, in welchen Formen dieses Wissen vorliegen kann. In der Literatur wird oftmals eine Unterscheidung in explizite und implizite Repräsentation von Wissen vorgenommen. Explizites Wissen liegt symbolisch vor und ist direkt verfügbar, während implizites Wissen erst gewonnen werden muss. In klassischen Architekturen wird oft explizites Wissen verwendet. In der verhaltensbasierten Robotik hingegen wird versucht, dies zu vermeiden und implizites Wissen zu verwenden. Arkin nimmt in [Arkin 98] eine etwas anschaulichere Unterteilung von Wissen für einen Roboter vor:

1. Räumliches Wissen über die Umwelt: Ein Verständnis des zur Verfügung stehenden Navigationsraumes und der Strukturen, die den Roboter umgeben.

2. Objektwissen: Kategorien oder Beispiele von verschiedenen Objekttypen in der Welt.

3. Wahrnehmungswissen: Informationen, wie man die Umgebung unter verschiedenen Umständen wahrnimmt.

4. Verhaltenswissen: Verständnis darüber, wie man sich in bestimmten Situationen verhalten sollte.

5. Ego-Wissen: Limitierungen der Roboterfähigkeiten in der umgebenden Welt.

6. Absichtswissen: Informationen, die die Ziele und Absichten des Roboters betreffen.

Die Wissensformen zwei und drei sind insbesondere für Wahrnehmungs- und Interpretationskomponeten interessant und spielen in dieser Arbeit eine eher untergeordnete Rolle. Besondere Aufmerksamkeit für eine Verhaltensentscheidung kommt den Punkten eins, vier, fünf und sechs zu. Im folgenden Abschnitt werden Modellierungsmöglichkeiten vorgestellt, um durch eine Kombination von Beobachtung und Wissen eine Entscheidung herbeizuführen.

B.1 Endliche Automaten

Endlichen Automaten oder Zustandsmaschinen[1] sind Modelle, um Verhalten zu beschreiben, welche aus Zuständen und Zustandsübergängen bestehen. Sie sind Teil der Automatentheorie, die als wichtiges Werkzeug der Berechenbarkeits- und Komplexitätstheorie verwendet wird, um formale Sprachen und formale Grammatiken zu verarbeiten. Von endlichen Automaten spricht man, wenn die Zustandsmenge des Automaten endlich ist. In der Praxis ist nur dieser Typ relevant, da sich die hiermit modellierten Systeme immer in einem wohldefinierten Zustand befinden.

Ein Zustand entspricht dabei einem Prozessschritt, in welchem Informationen gespeichert und verarbeitet werden. Ein Zustandsübergang (Transition) beschreibt eine Änderung von einem Zustand in einen anderen Zustand und definiert logische Bedingungen, welche erfüllt sein müssen, um einen Übergang auszulösen. Aktionen sind situationsabhängige Aktivitäten des Automaten und werden nach [Wikipedia 09] wie folgt unterschieden:

- Eingangsaktion: wird beim Eintreten in einen neuen Zustand ausgelöst.
- Ausgangsaktion: wird beim Verlassen eines Zustandes ausgelöst.
- Eingabeaktion: wird im aktuellen Zustand abhängig von Eingabedaten ausgelöst.
- Übergangsaktion: wird bei einem Zustandsübergang ausgelöst.

Ein endlicher Automat wird in der Regel durch Übergangstabellen oder ein Zustandsübergangsdiagramm dargestellt. Abbildung B.1 zeigt die Darstellung eines einfachen endlichen Automaten, der das Verhalten eines Fahrers an einer Ampel mittels Zustandsübergangstabelle und -diagramm beschreibt.

[1] Engl. *Finite State Machine (FSM).*

Aktueller Zustand / Übergangs-bedingung	Fährt	Steht
Ampel rot	Steht E: Abbremsen	-
Ampel grün	-	Fährt E: Anfahren

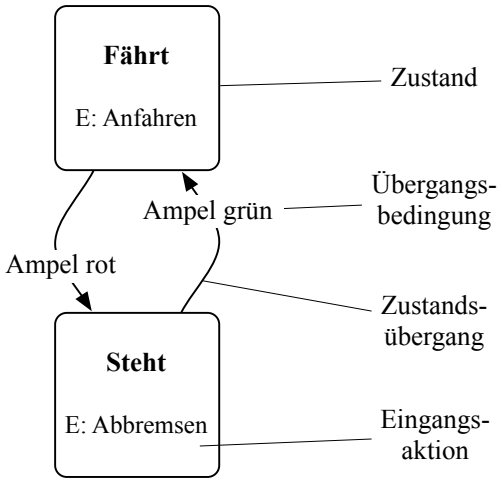


Abbildung B.1: Vereinfachte Darstellung von Fahrzuständen in Abhängigkeit von einer Ampelanlage als Zustandsübergangstabelle (links) und Zustandsübergangsdiagramm mit Eingangsaktionen (rechts).

Grundsätzlich werden zwei Gruppen von Automaten unterschieden: *Akzeptoren* (erkennende Automaten ohne Ausgabe) und *Transduktoren*. Ein typisches Anwendungsgebiet für Akzeptoren sind das Suchen von Mustern in Bild, Sprache und Wort oder das Überprüfen einer Benutzereingabe. Gelangt der endliche Automat nach Verarbeitung der Eingabedaten in den gewünschten Endzustand, so wurden die Eingangsdaten akzeptiert. Führt durch unpassende Eingangsdaten ein Zustandsübergang zu einem anderen Ergebnis, dann wird diese Tatsache über einen abweichenden Endzustand kommuniziert.

Transduktoren hingegen kommunizieren nicht nur über den aktuellen Zustand, sondern liefern selbst Ausgabedaten. Die Ausgabefunktionen werden durch bestimmte Aktionen ausgelöst. Typische Anwendungen für diese Klasse von endlichen Automaten sind Prozesssteuerungen, bei welchen die Kommunikation nicht ausschließlich über Einzelzustände erfolgen kann. Es lassen sich zwei bekannte Typen unterscheiden:

Moore-Automat
In diesem Modell werden lediglich Eingangsaktionen benutzt. Die Ausgabe hängt also nur vom Zustand ab. Der Ablauf im endlichen Automaten ist dadurch leichter nachzuvollziehen als mit dem Mealy-Modell, allerdings sind dafür mehr Zustände notwendig.

Mealy-Automat
Im Mealy-Modell werden Eingabeaktionen benutzt. Die Ausgabe hängt vom Zustand und der Eingabe ab. Der Vorteil dieser Art der Modellierung ist die Verringerung der Anzahl von Zuständen. Dies wird allerdings mit einer höheren Komplexität der Einzelzustände erkauft, welche oftmals schwieriger zu durchschauen sind.

Moore- und Mealy-Automaten unterscheiden sich lediglich in der Darstellung. Bzgl. der Modellierungsmöglichkeiten sind sie gleichwertig und können ineinander überführt werden.

Abbildung B.2 zeigt die Darstellung eines endlichen Automaten zur Steuerung eines Fahrzeuges, ausgeführt als Moore- oder Mealy-Modell. Im Gegensatz zu Abbildung B.1 wurde der Moore-Automat ausformuliert und beinhaltet in den Einzelzuständen keine Unterzustände mehr. Eine sehr gute Einführung in die Automatentheorie bietet [Hopcroft 03]. Tipps zur praktischen Anwendung sind in [Wagner 06] enthalten.

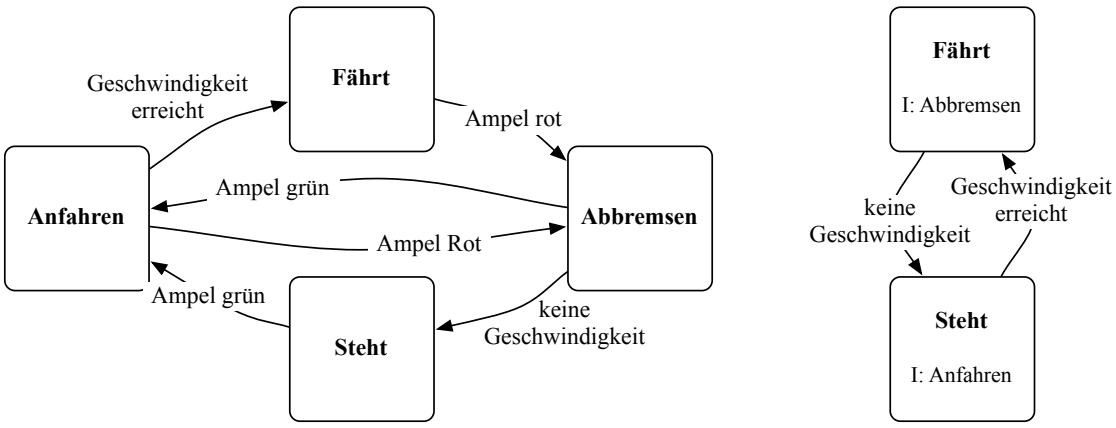


Abbildung B.2: Endlicher Automat zur Steuerung eines Fahrzeuges an einer Ampelanlage, ausgeführt als Moore-Automat (links) und Mealy-Automat (rechts).

B.2 Petri-Netze

Petri-Netze wurden in den 60er Jahren durch Carl Adam Petri formal definiert und können als mathematisches Modell für nebenläufige Sytsteme verwendet werden. Aufgrund dieser Fähigkeit, auch nebenläufige Prozesse abzubilden, stellen Petri-Netze eine Verallgemeinerung der Automatentheorie dar. Sie finden hauptsächlich in der industriellen Automatisierungstechnik Anwendung, werden aber auch als Verfahren zur Entscheidungsfindung in der Robotik oder für Organisationszwecke genutzt. Die mathematischen Grundlagen sind eine Voraussetzung für Analyseverfahren, um eine Validierung und Verifikation von Prozessen durchführen zu können. Hierfür stehen etliche Werkzeuge zur Verfügung [PNW 08].

Ein Petri-Netz ist ein gerichteter Graph, welcher aus Stellen (*Places P*), Transitionen (*Transitions T*), gerichteten Kanten und Marken besteht (vgl. Abbildung B.3). Jede Stelle im Netz kann über eine vorab definierte Kapazität eine bestimmte Anzahl an Marken aufnehmen. Den gerichteten Kanten sind Kosten zugeordnet, welche eine vorgeschaltete Stelle aufbringen muss, um eine Transition aktivieren zu können. Eine Transition feuert, wenn alle Kosten von den eingehenden Kanten aufgebracht werden. Entsprechend den Kantengewichten werden dann aus den Eingängen Marken entnommen und auf der ausgehenden Kante hinzugefügt.

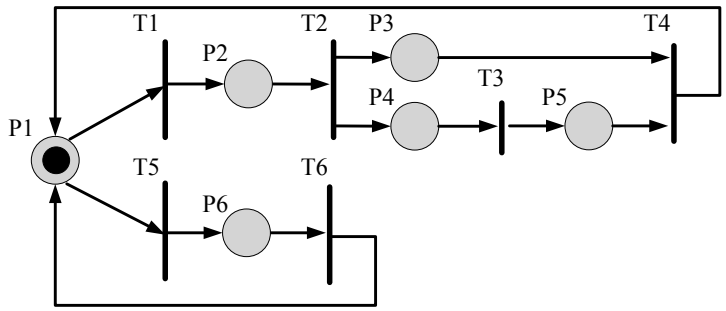


Abbildung B.3: Petri-Netz mit Stellen P, Transitionen T, gerichteten Kanten und Marken (in P_1).

Ein Petri-Netz kann formal als ein 6-Tupel (S, T, F, K, W, m_0) mit folgenden Eigenschaften definiert werden:

- S: nichtleere Menge von Stellen $S = \{s_1, s_2, ..., s_{|S|}\}$
- T: nichtleere Menge von Transitionen $T = \{t_1, t_2, ..., t_{|T|}\}$
- F: nichtleere Menge der Kanten $F \subseteq (S \times T) \cup (T \times S)$
- Kapazitätsfunktion K : $S \rightarrow \mathbb{N} \cup \{\infty\}$
- Gewichtsfunktion $W : F \rightarrow \mathbb{N}$
- Startmarkierung $m_0 : S \rightarrow \mathbb{N}$

Aufgrund der disjunkten Beziehung zwischen S und T werden Petri-Netze auch als bipartiter und gerichteter Graph bezeichnet. Der aktuelle Zustand des Netzes ist jederzeit anhand der Marken in den Stellen ablesbar und wird durch die Markierung m repräsentiert. Da Marken in der ursprünglichen Definition von Petri-Netzen nicht unterscheidbar sind wurden in späteren Weiterentwicklungen farbige Marken eingeführt, um Prozesse noch differenzierter betrachten zu können. Zeiterweiterte Petri-Netze erlauben die Darstellung von zeitverbrauchenden Transitionen für die bessere Abbildung realer Prozesse. Eine gute Einführung in Petri-Netze bietet [Reisig 90]. Anwendungsorientierte Einführungen mit Beispielen finden sich in [Abel 90] und [Baumgarten 96].

B.3 Aussagenlogik und Prädikatenlogik

Um eine Logik aufzubauen, muss zunächst eine konkrete Vorstellung darüber vorhanden sein, was ausgedrückt werden soll, um auf dieser Basis eine formale Repräsentation oder Syntax zu erstellen. Eine Semantik definiert dabei die genaue Bedeutung der Syntax. Mithilfe von Logiken ist es möglich, neues Wissen aus bereits bekanntem Wissen zu folgern. Dies geschieht durch das Einführen von Aussagen und Regeln anhand von bereits bekanntem Wissen. Bspw.

kann aus den folgenden Aussagen geschlussfolgert werden, dass auf einer Straße Autofahrer und Radfahrer vorkommen können:

1. Auf einer Straße kann man fahren.
2. Ein Auto kann fahren.
3. Ein Fahrradfahrer kann fahren.

Mithilfe der Aussagenlogik können Verknüpfungen von und Beziehungen zwischen Urteilen untersucht werden. Die Urteile in der Aussagenlogik sind dabei atomare Einheiten und können entweder *wahr* oder *falsch* sein. Durch Kombination solcher elementarer Einheiten mithilfe verschiedener Arten von Verknüpfungen können zusammengesetzte Aussagen wie bspw. *Das Auto fährt UND der Vogel fliegt* kombiniert und ausgewertet werden. Als Darstellung in der Prädikatenlogik würde diese Aussage folgendermaßen formuliert:

fahren(Auto) $\triangle$ fliegen (Vogel)

Die Prädikatenlogik baut auf der Aussagenlogik auf und ermöglicht eine Untersuchung der elementaren aussagenlogischen Bestandteile hinsichtlich ihrer inneren Struktur [Barwise 05, Mates 97]. Im prädikatenlogischen Sinne ist ein Prädikat, entgegen der grammatikalischen Bedeutung, eine Folge von Wörtern mit Fehlstellen, welche durch das Einsetzen von Termen eine vollständige Aussage mit wahrem oder falschem Gehalt liefert. Das Prädikat *fährt(x)* würde durch Einsetzen des Terms *Das Auto* zu einer Aussage: *fährt(das Auto)*.

Das wohl wichtigste Element der Prädikatenlogik ist der Quantor. Quantoren beschreiben, auf wieviele Objekte ein Prädikat zutrifft. Grundsätzlich werden zwei Arten von Quantoren unterschieden:

- Universelle Quantoren liefern eine Aussage über alle Objekte einer Menge. Sie ist genau dann wahr, wenn die quantifizierte Aussage für alle diese Objekte zutrifft. Die Aussage *_fährt* würde als universalquantifizierte Aussage lauten: *Alle Autos fahren*, in formaler Darstellung $\forall x F x$, wobei $\forall$ den Universalquantor darstellt, $F\_$ das Prädikat *_fährt*.
- Existenzielle Quantoren hingegen liefern eine Aussage über einzelne Elemente und sind genau dann wahr, wenn die Aussage für mindestens ein Element der Menge gilt. Die Aussage *_fährt* würde in existenzquantifizierter Übersetzug lauten: *Es gibt mindestens ein Ding, das fährt.* Die formale Darstellung dazu wäre $\exists x F x$, mit $\exists$ als Existenzquantor.

Die Frage, ob eine unbeschilderte Kreuzung passiert werden darf, könnte durch folgenden prädikatenlogischen Zusammenhang beantwortet werden:

$$erlaubt = istunbeschildertekreuzung(x) \wedge istmöglich(y) \wedge\neg\ istbelegt(s)$$

mit den Termen *x: Kreuzungstyp, s: rechter Zubringer, y: Geradeausfahrt*. Um eine Interpretation der Variablenbelegung vornehmen zu können, muss zuvor der Diskursbereich festgelegt werden. Der Diskursbereich beschreibt formal einen begrenzten Ausschnitt aus einer realen Welt und legt mögliche Prädikate und Terme fest.

Die Prädikatenlogik ist eine Möglichkeit, Wissen zu repräsentieren und bspw. durch die Kombination mit Entscheidungsbäumen auch komplexe Entscheidungen zu treffen. Über prädikatenlogische Ausdrücke kann an Knotenpunkten abhängig vom Ergebnis verzweigt werden. Die Prädikatenlogik bildet die theoretische Grundlage für die Programmiersprache Prolog.[2]

B.4 Fuzzy-Logik

Die Grundlage der *Fuzzy-Mengen-Theorie*, oder der *unscharfen Mengenlehre*, wurde 1965 an der Universität von Berkeley, USA von Lotfi A. Zadeh entwickelt. Durch diese Theorie entstand die Möglichkeit, nicht exakte und unvollständige Datensätze mathematisch zu beschreiben, welche zunächst nur in verbaler Form vorliegen. Aus sprachlich formulierten Regeln kann mittels Fuzzy-Logik eine mathematische Beschreibung erzeugt und diese mit Rechnersystemen verarbeitet werden. Fuzzy-Inferenz kann den Widerspruch lösen, daß die Prämissen mehrerer Regeln „irgendwie“ erfüllt sind, indem mehrere Regeln Gültigkeit besitzen und entsprechend fusioniert werden.

Der Hauptunterschied zwischen Fuzzy Mengen und traditionellen Mengen liegt an der Definition der Zugehörigkeitsfunktion. Bei traditionellen Mengen werden die Zugehörigkeitsaussagen der Elemente eindeutig gemacht – Elemente sind hier in Mengen enthalten oder nicht. Bei Fuzzy-Mengen gibt die Zugehörigkeitsfunktion μ als reelle Zahl zwischen 0 und 1 den Grad der Zugehörigkeit an und beschreibt damit, wieviel Prozent des Elements zu einer Menge gehört.

Somit entsteht die Möglichkeit, Elemente einer Menge linguistisch zu beschreiben und technisch zu verarbeiten. In Abbildung B.4 sind trapez- oder dreiecksförmige Zugehörigkeitsfunktionen dargestellt, welche in ihrer Gesamtheit die *Fuzzyfunktion* ergeben, um eine linguistische Variable *Wassertemperatur* zu beschreiben. Viele Probleme lassen sich in praktischen Anwendungen auf diese Grundformen reduzieren, welche aufgrund der einfachen Berechenbarkeit gerne verwendet werden. Grundsätzlich sind jedoch beliebige Funktionen denkbar die einen Funktionswert im Intervall $[0, 1]$ liefern.
Eine Fuzzy-Menge A ist definiert als eine Menge von Paaren

$$A = \{(x, \mu) | x \in \mathrm{R}, \mu \in [0, 1]\},$$

wobei jeder Zahl x ein Zugehörigkeitsgrad μ zwischen 0 und 1 zugeordnet wird. Die gesamte Zuordnung $x \rightarrow \mu$ ergibt die Zugehörigkeitsfunktion μ_A der Menge A. Analog zur klassischen

[2] Abk. *Programming in Logic*.

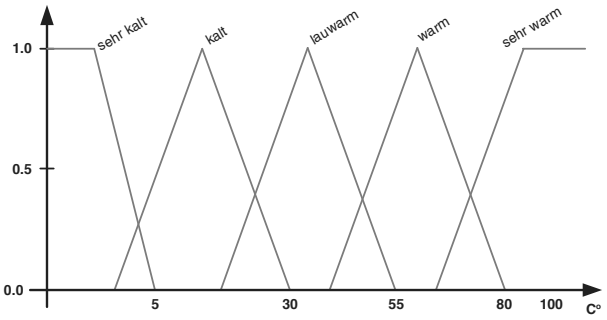


Abbildung B.4: Definition von Fuzzy-Mengen über Dreiecks- und Trapezfunktionen. Die linguistische Variable *Wassertemperatur* wird über die verbalen Terme *sehr kalt, kalt, lauwarm, warm* und *sehr warm* beschrieben.

Logik sind auch in der Fuzzy-Logik Mengenoperationen definiert. Davon sind Methoden für die Berechnung des Durchschnitts und der Vereinigung die am häufigsten verwendeten:

Durchschnitt: $min\{\mu_A(x), \mu_B(x)\}$

Vereinigung: $1 - \mu_A(x)$

Komplement: $max\{\mu_A(x), \mu_B(x)\}$

Mit diesen Operatoren ist es möglich, den Grad der Zugehörigkeit aus mehreren Zugehörigkeitsfunktionen zu bestimmen und dadurch vielfältige Faktoren zu berücksichtigen. Abbildung B.5 zeigt den Datenfluss in einem Fuzzy-System mit Umwandlung scharfer Eingangswerte (Fuzzifizierung), dem Schließen (Fuzzy-Inferenz mit Komposition) und der Rückwandlung in scharfe Ausgangswerte (Defuzzifizierung).

Fuzzifizierung: Die Generierung von unscharfen Daten aus scharfen Informationen wie bspw. Messwerten wird als *Fuzzifizierung bezeichnet*. Im Vorfeld müssen hierfür die Fuzzy-Mengen und Zugehörigkeitsfunktionen auf Basis der linguistischen Variablen definiert werden (vgl. Abbildung B.4). Dabei ist zu beachten, dass die Zugehörigkeitsfunktionen linguistischer Terme sich überlappen. In der Verarbeitung wird ein Eingangswert fuzzifiziert, indem der Grad seiner Zugehörigkeit für alle linguistischen Variablen bestimmt wird.

Fuzzy-Inferenz und Komposition: Dieser Schritt wird auch als unscharfes Schließen bezeichnet und definiert die Wechselwirkungen im System. Dafür werden Regeln der Form

$$\text{wenn } X_1 \text{ ist } x_1 \text{ und } ... \text{ und } X_n \text{ ist } x_n \text{ dann } Y \text{ ist } y$$

definiert, um die Eingangsgrößen durch Operationen wie Vereinigung oder Durchschnitt zu verknüpfen und auf Ausgangsgrößen abzubilden. $X_1, ..., X_n$ und Y stellen linguistische Variablen dar, $x_1, ..., x_n$ und y Fuzzy-Mengen. Die Regeln werden oftmals von einem

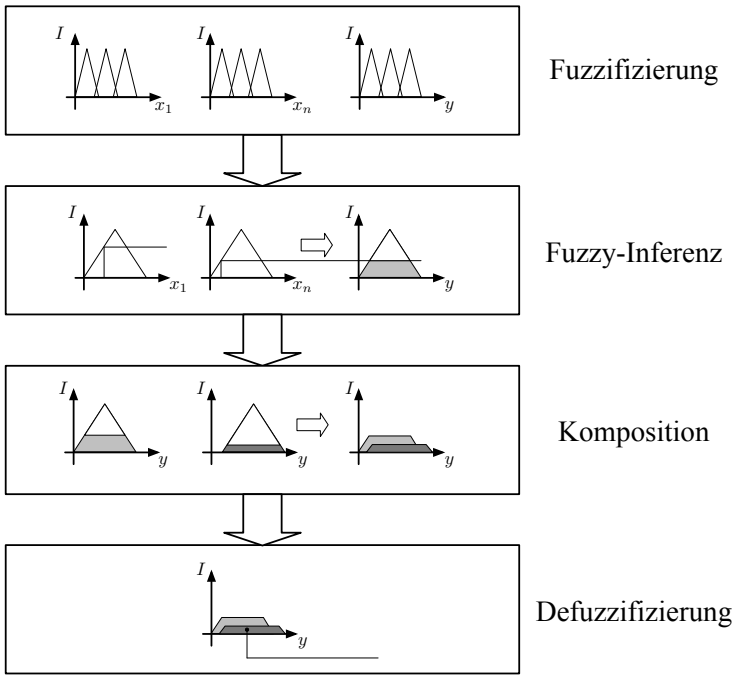


Abbildung B.5: Datenfluss in einem Fuzzy-System: Nach der Fuzzifizierung der scharfen Eingangswerte erfolgt eine Inferenz auf Basis hinterlegter Regeln. Eine Gesamtzugehörigkeitsfunktion wird über die Komposition der Einzelfunktionen gebildet, ehe durch Defuzzifizierung wieder ein scharfer Ausgangswert erzeugt wird.

Experten sprachlich formuliert, ohne dass die mathematischen Zusammenhänge bekannt sind. Die aus den Regeln entstehenden Zugehörigkeitsfunktionen werden zu einer Gesamtzugehörigkeitsfunktion zusammengefasst. Dieser Schritt wird als Komposition bezeichnet und erfolgt meist durch Maximumsfusion.

Defuzzifizierung Die Fuzzy-Inferenz liefert als Ergebnis wiederum eine unscharfe Information. Ziel der Defuzzifizierung ist es, das Ergebnis wieder in scharfe Ausgangswerte umzuwandeln, um die Ausgaben bspw. zur Steuerung eines technischen Systems nutzen zu können. Abhängig von der Anwendung kommen dafür verschiedene Methoden zum Einsatz. Ein häufig angewandtes Verfahren ist die Schwerpunkt-Methode, bei der ein Ausgabewert anhand des Schwerpunkts der Zugehörigkeitsfunktion ermittelt wird.

Die Verwendung von Fuzzy-Logik erstreckt sich über verschiedenste Bereiche. Neben der Automatisierungstechnik und Regelungstechnik kommt sie mittlerweile auch in der Betriebswirtschaft, der Medizintechnik oder zur Sprachbeschreibung zum Einsatz. In vielen Fällen werden die notwendigen Fuzzy-Funktionen aus statistischen Erhebungen erzeugt. Sofern diese Messung im Vorfeld nicht möglich ist, können diese auch erst im laufenden Betrieb erhoben werden. Fuzzy-Logik bietet sich deshalb auch an, um lernende Systeme zu realisieren, bspw. zur Prognose des Energieverbrauchs oder zur Steuerung eines Fahrstuhls. Eine gute

Einführung in die Theorie der Fuzzy-Logik findet sich in [Lowen 96], während der Schwerpunkt in [Zimmermann 01] auf Anwendungen der Fuzzy-Logik liegt.

B.5 Bayes'sche Netze

Mithilfe Bayes'scher Netze kann unsicheres Wissen repräsentiert und daraus geschlussfolgert werden [Jensen 96, Thrun 05]. Es handelt sich dabei um graphische Modelle, die Graphentheorie und Wahrscheinlichkeitstheorie miteinander kombinieren, um die Wahrscheinlichkeitsverteilung verschiedener voneinander abhängiger Ereignisse zu modellieren. Ein Bayes'sches Netz besteht aus einem gerichteten, azyklischen Graph und bedingten Wahrscheinlichkeitsverteilungen. Es besteht aus:

Knoten – Repräsentieren Bayes'sche Zufallsvariablen oder Ereignisse.

Gerichtete Kanten – Beschreiben kausale Abhängigkeiten zwischen verbundenen Knoten und enthalten bedingte Wahrscheinlichkeiten. Das Ereignis an der Pfeilspitze hängt von dem Ereignis am Pfeilende in irgendeiner Weise ab.

Ein Bayes'sches Netz dient dazu, die gemeinsame Wahrscheinlichkeitsverteilung aller beteiligten Variablen unter Ausnutzung bekannter bedingter Unabhängigkeiten möglichst kompakt zu repräsentieren. Sind $A_1, ..., A_n$ Bayes'sche Zufallsvariablen, so stellt sich deren gemeinsame Verteilung, die Verbundwahrscheinlichkeitstafel aller Knoten, als

$$P(A_1, ..., A_n) = \prod_{i=0}^{n} P(A_i|pa(A_i))$$

dar, wobei $pa(A)$ die Menge der Elternknoten repräsentiert. Mithilfe von Baumdiagrammen lassen sich Wahrscheinlichkeitsverteilungen veranschaulichen (vgl. Abbildung B.6). Im Beispiel kann die Wahrscheinlichkeit für das Eintreten eines bestimmten Ereignisses durch die Verknüpfung von Pfaden errechnet werden, dabei gelten folgende Pfadregeln:

1. Die Wahrscheinlichkeit eines Ereignisses ist gleich dem Produkt der Wahrscheinlichkeiten auf dem Pfad, der zu diesem Ereignis führt.

2. Die Wahrscheinlichkeit eines Ereignisses ist gleich der Summe der Wahrscheinlichkeiten der Pfade, die dieses Ereignis bilden.

Um auf das Eintreten eines bestimmten Ereignisses zu schließen und darauf Entscheidungen aufzubauen, werden in der Praxis approximative Verfahren verwendet. Exakte Verfahren sind möglich, spielen aber eine oft nur unwesentliche Rolle, da die exakten Wahrscheinlichkeiten selten benötigt werden. Auch exakte Inferenzverfahren sind bei Verwendung auf Rechnern mit Gleitkommaarithmetik nicht genau. Neben einer Entscheidungsfindung auf Basis bestimmter Ereigniswahrscheinlichkeiten werden Bayes'sche Netze gerne auch für gezielte weitere Abfragen eingesetzt, um zu ermitteln, welche zusätzlichen Informationen notwendig sind, um eine Entscheidung *sicher* treffen zu können. Die Bezifferung einer Eintrittswahrscheinlichkeit als Sicherheitsmaß ist für alle Anwendungen interessant, bei

denen ein Nachweis erbracht werden muss, um eine Zulassung, bspw. für den Betrieb eines Fahrerassistenzsystems, zu erhalten.

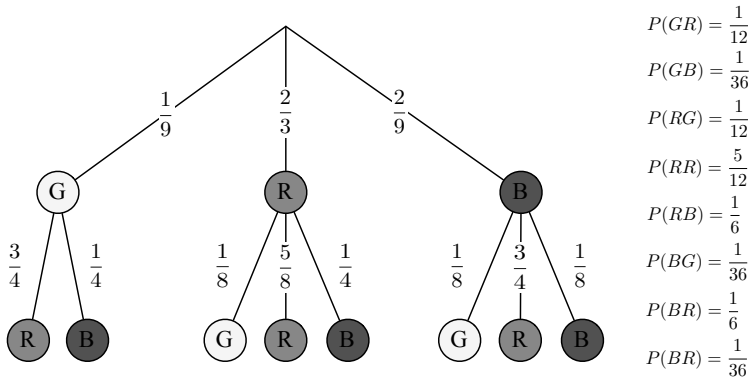


Abbildung B.6: Bayes'sches Netz als Baumdiagramm: Zweimaliges Ziehen ohne Zurücklegen einer Kugel aus einer Urne mit zwei blauen, sechs roten und einer gelben Kugel.

Bayes'sche Netzwerke bieten den großen Vorteil, dass Probleme behandelt werden können, auf die sich wegen ihrer Komplexität oder unvollständiger Daten kaum andere Methoden anwenden lassen. Neben der mathematischen Grundlage stellt die Bayes'sche Wahrscheinlickeitsrechnung eine sehr verständliche Beschreibungsmöglicheit stochastischer Zusammenhänge zur Verfügung. Eine Integration menschlicher Erfahrungswerte in Form von Expertenwissen ist mit einer subjektiven Formulierung von Wahrscheinlichkeiten einfach möglich. Durch die gezielte Änderung bestimmter Wahrscheinlichkeiten lassen sich leicht die Folgen bestimmter Ereignisse abschätzen und simulieren oder eine Sensitivitätsanalyse durchführen.

Im Gegenzug lassen sich Bayes'sche Netzwerke nur schwer auf Korrektheit überprüfen und an veränderte Umweltbedingungen anpassen. Oftmals stellt es eine große Schwierigkeit dar die Wahrscheinlichkeiten zu bestimmen. Zudem ist es nicht möglich, zyklische Abhängigkeiten zu beschreiben, da diese im Graph nicht vorkommen dürfen. Damit können nicht alle vorhandenen Unabhängigkeiten der Ursprungsverteilung in Bayes-Netzen dargestellt werden. Aufgrund der einfachen Verständlichkeit und der Möglichkeit zur Nutzung von Erfahrungswissen, ist die Verwendung jedoch weit verbreitet.

Anwendung finden Bayes'sche Netze überall dort, wo mit unsicherem Wissen gearbeitet wird und unbekannte und zufällige Einflüsse eine wesentliche Rolle spielen. Darunter fällt z. B. in der Medizin die Diagnose von Krankheiten, die Durchführung einer Wettervorhersage in der Meteorologie oder die Wissensrepräsentation, Entscheidungsfindung und Mustererkennung in der Informatik.

B.6 Entscheidungsbäume

Entscheidungsbäume stellen eine weitere Möglichkeit dar, Wissen zu repräsentieren. Sie folgen dem Prinzip *Teile und Herrsche*[3], indem sie Probleme in mehrere Teilprobleme zerlegen, um diese dann vereinfacht lösen zu können. Entscheidungsbäume eignen sich sehr gut, um aufeinanderfolgende Entscheidungen anschaulich darzustellen und den Lösungsweg nachzuvollziehen (vgl. Abbildung B.7). Ein Objekt oder eine Situation dient als Eingabe an der Wurzel des Entscheidungsbaumes. In der Folge werden in jedem Knoten des Entscheidungsbaumes Regeln überprüft, die dann zum nächsten Knoten führen. Gelangt man an einen Endknoten (Blatt) des Baumes, so ist das Ergebnis gefunden und die Ausgangsfrage beantwortet. Auf dem Weg von der Wurzel bis zum Blatt werden die Objektattribute schrittweise konjugiert. Man spricht auch davon, dass das Objekt *klassifiziert* wurde und bezeichnet Entscheidungsbäume aus diesem Grund oftmals als Klassifikationsbäume. Binäre Entscheidungsbäume stellen dahingehend einen Sonderfall dar, dass die Regeln nur aus Boole'schen Ausdrücken bestehen.

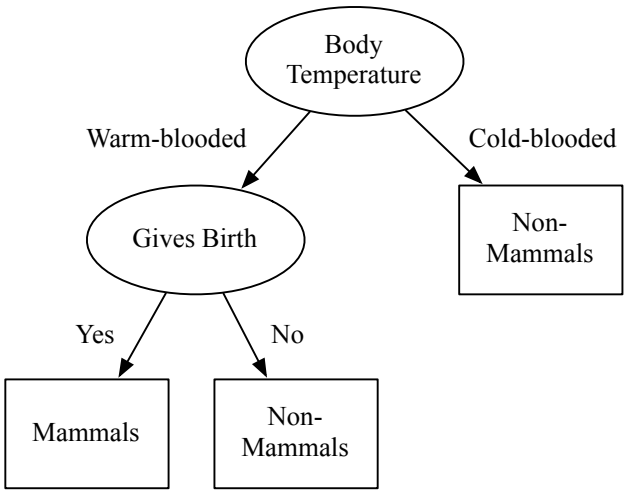


Abbildung B.7: Entscheidungsbaum zur Lösung des Klassifikationsproblems für Säugetiere.

Entscheidungsbäume können über Regelinduktion automatisch aufgebaut oder gelernt werden und sind deshalb ein weit verbreitetes *Data-Mining*-Verfahren. Sind Objekte und zugehörige Klassen bekannt, so kann diese Information genutzt werden, um in einem *Top-Down*-Ansatz sukzessive Einzelregeln zu erstellen und den Baum aufzubauen. Dabei wird an jedem Knoten ein neues Attribut eingesetzt und über statistische Verfahren ermittelt, wie gut es die Trainingsmenge zerlegt.

Die Einzelschritte sind in einem Entscheidungsbaum gut nachzuvollziehen, den Gesamtüberblick zu behalten ist bei komplexen Regelwerken jedoch schwierig. Mit zunehmender

[3]Engl. *Divide and Conquer*.

Komplexität kann zudem die Klassifikationsgüte abnehmen, da die Gefahr einer Überanpassung besteht. Durch ein sogenanntes *Pruning*[4] kann in solchen Fällen eine Vereinfachung stattfinden. Dies kann geschehen, indem gezielt nach Redundanzen in den Regeln gesucht wird. Ändert sich durch das Weglassen einer einzelnen Bedingung nichts am Ergebnis, so wird diese aus der Regel gestrichen und die Menge reduziert.

Entscheidungsbäume werden mittlerweile nicht mehr nur in der KI benutzt, sondern vermehrt auch als Diagnosewerkzeug in der Medizin, für die Generierung von Kundenprofilen oder zur Fehleranalyse. In [French 86] und [Mitchell 97] werden Entscheidungsbäume als Möglichkeit der Entscheidungsfindug ausführlich diskutiert.

B.7 Fallbasiertes Schließen

Fallbasiertes Schließen *(Case-Based-Reasoning, CBR)* bezeichnet ein maschinelles Lernverfahren, bei dem durch Analogiebildung ein bekanntes Lösungsverfahren eines früheren Problems auf ein aktuelles Problem angewandt wird. Als anschauliches Beispiel wird oft ein Automechaniker angeführt, welcher das Problem löst, indem er sich einen ähnlichen Fall eines anderen Fahrzeuges in Erinnerung ruft. Bei dieser Art der Modellierung werden Aufgaben mit zugehöriger Lösung in einer Datenbasis abgelegt. Im Bezug auf ein kognitives Automobil könnten dies auch eine Situationsbeschreibung und ein auszuführendes Verhalten sein. Ist für ein aktuelles Problem keine Lösung bekannt, so können bekannte Fälle auf dieses Problem zurückgeführt werden. Sind mehrere Fälle aus der Datenbank ähnlich, dann kann jeder einzelne durch Fusion zur Gesamtlösung beitragen. Dies ist in der Praxis jedoch schwer zu realisieren.

Fallbasiertes Schließen kann den Lernverfahren der künstlichen Intelligenz zugeordnet werden, da das vorhandene Wissen ständig erweitert wird. Von Vorteil ist, dass eine vergleichsweise kleine Anzahl an Referenzbeispielen ausreicht und durch die Analogiebildung und Rückspeisung von Beispielen stetig anwächst. Eine Schwierigkeit besteht in der Herstellung von Analogien, welche weit genug gefasst sein müssen, um ausreichend abstrahieren zu können, allerdings nicht zu weit, um keine Verbindung zwischen unterschiedlichen Fällen herzustellen. Eine potentielle Gefahr besteht darin, dass eine Veralterung des Wissens stattfindet, da bekannte Lösungsvorschläge auf immer mehr Fälle angewandt werden ohne neue Lösungen einzubringen. Dieser Effekt kann durch gezieltes Training vermindert werden, indem Fälle, welche idealerweise den bekannten Fällen nicht zu ähnlich sind, mit neuen Lösungen eingelernt werden.

Von Agnar Aamodt und Enric Plaza wurde Fallbasiertes Schließen als ein Prozess mit vier Phasen formalisiert (vgl. Abbildung B.8 und [Aamodt 94]):

Retrieve
Passend zur einer Problembeschreibung wird in der Fallbasis ein ähnliches Problem

[4] *Beschneidung.* Bewusstes ignorieren bestimmter Informationen.

gesucht. Die Schwierigkeit besteht dabei in der Defintion der Ähnlichkeiten und damit der Festlegung des zulässigen Abstraktionsgrades.

Reuse
Der ähnlichste Fall wird als möglicher Lösungsweg identifiziert und bildet den Ausgangspunkt für die Lösung des Problems.

Revise
Üblicherweise muss der gefundene Lösungsansatz für das vorliegende Problem angepasst werden. Dies geschieht in der *Revise*-Phase.

Retain
Der neue Fall wird mit Lösungsweg in der Fallbasis abgespeichert und kann auf künftige Problemstellungen angewandt werden. Die Leistungsfähigkeit des Systems wird mit jedem Eintrag weiter verbessert.

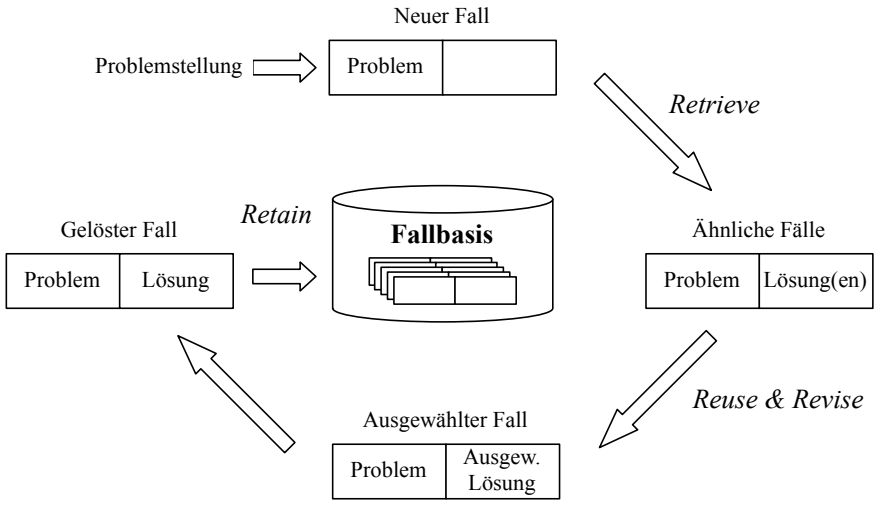


Abbildung B.8: Zyklus des Fallbasierten Schließens mit vier Phasen.

Fallbasiertes Schließen stammt ursprünglich aus dem Teilgebiet der künstlichen Intelligenz, hat aber inzwischen ebenfalls Anwendungsmöglichkeiten in Unternehmen bspw. zu Diagnosezwecken, in Beratungssystemen oder in der Wirtschaftsinformatik gefunden. In [Vacek 07] wird Fallbasiertes Schließen zur Situationsinterpretation für ein kognitives Automobil verwendet.

B.8 Fazit

Mit endlichen Automaten und Petri-Netzen stehen Architekturen für eine sehr einfache Form der Entscheidungsfindung zur Verfügung, die damit relativ leicht zu modellieren und nachzuvollziehen ist. Es ist jedoch fraglich, ob die Situationsvielfalt im Straßenverkehr mit

diesen Methoden zufriedenstellend abgedeckt werden kann, oder eine aufwändige Detaillierung in eine sehr große Menge an Zuständen mit entsprechend vielen Übergängen notwendig ist. In der Aussagen- und Prädikatenlogik lassen sich Regeln sehr einfach umsetzen. Diese Art der Modellierung bietet sich dann an, falls Regeln für eine Entscheidungsfindung nicht anhand mathematischer Modelle, sondern bspw. mithilfe von Angaben eines menschlichen Fahrlehrers erstellt werden sollen. Die von einem aussagen- oder prädikatenlogischen Entscheidungssystem getroffenen Entscheidungen sind wiederum natürlichsprachlicher Natur und insofern sehr gut nachvollziehbar.

Fuzzy-Logik ist im Bezug auf eine sprachliche Formulierung von Regeln ebenfalls interessant. Ein Problem für eine Anwendung auf kognitive Automobile stellt jedoch die Definition der Fuzzy-Funktionen dar. Diese lassen sich für den komplexen Diskursbereich *Straßenverkehr* nicht wie bei einfacheren Systemen aus statistischen Erhebungen gewinnen. Im Gegensatz zu den anderen vorgestellten Verfahren bieten Bayes'sche Netze die Möglichkeit, Unsicherheiten in der Entscheidungsfindung mit zu berücksichtigen, und damit die Sicherheit in der Entscheidung durch eine wahrheitsgetreuere Modellierung zu erhöhen.

Fallbasiertes Schließen dient als Verfahren zur Problemlösung eher der Entscheidungsfindung „im Großen". Es scheint insofern für eine Situationsinterpretation eher geeignet als für eine Entscheidungsfindung in der Verhaltensausführung. Die elementaren Handlungen der Fahrzeugführung, als Reaktion auf eine interpretierte Situation, müssen dem System vorab bekannt sein und lassen sich nicht erst durch Analogiebildung gewinnen.

Anhang C

Umfeldkartierung

Im folgenden Abschnitt werden bekannte approximative und kontinuierliche Methodiken zur Umfeldkartierung aufgeführt. Die Kartierung dient an dieser Stelle ausschließlich dem Zweck der metrischen oder metrisch-temporalen Umfeldrepräsentation als Grundlage für eine Bahnplanung. Topografische Karten, wie sie als Wissensspeicher zum Einsatz kommen, werden an dieser Stelle nicht weiter vertieft.

Die Umfeldkartierung kann, je nach verwendetem Bahnplanungsalgorithmus, variabel oder verbindlich an die Methode gekoppelt sein. Da eine Kartierung in der Regel in einem separaten Schritt erfolgt und z. B. zum Zweck der Datenfusion auch gänzlich für sich stehen kann, werden die verschiedenen Methoden zunächst separat aufgelistet.

C.1 Quad-Trees

Quad-Trees sind neben den gitterbasierten Ansätzen mit konstanter Zellgröße, wie sie auch für die Belegtheitskarte verwendet werden, eine Möglichkeit, Daten unter Verwendung von möglichst wenig Speicherplatz abzulegen [Finkel 74, Samet 84, Kraschl 05]. Sie optimieren die zweidimensionale Gitterdarstellung, indem der Raum zunächst nur in vier Gebiete unterteilt wird und diese je nach Belegtheit bis zu einer Maximalauflösung weiter verfeinert werden (vgl. Abbildung C.1). Als Alternative Darstellung kann eine Baumstruktur dienen, bei der jeder Knoten bis zu vier Kindknoten besitzt. Für den dreidimensionalen Fall werden sogenannte *Oct-Trees* verwendet [Ibaroudene 95]. Auf Basis dieses Zerlegungsverfahrens lassen sich Belegtheitskarten mit variabler Auflösung erstellen.

Eine effiziente Speichernutzung kann ihren Vorteil haben. Dieser hebt sich aber je nach Umgebung wieder auf, wenn keine großen, zusammenhängende Bereiche frei sind. Die laufende Neuorganisation der Baumstruktur, und die damit zusammenhängende Ausführung von Prüfmechanismen, kann sich ebenfalls nachteilig auswirken. Neben der Kartenerstellung zur Bahnplanung oder Kollisionsvermeidung in der Robotik kommen Quad-Trees auch in der Computergrafik für Visualisierungen, zur flächenhaften Strukturierung von Rasterdaten sowie als Zugriffsmechanismus in Datenbanksystemen zum Einsatz.

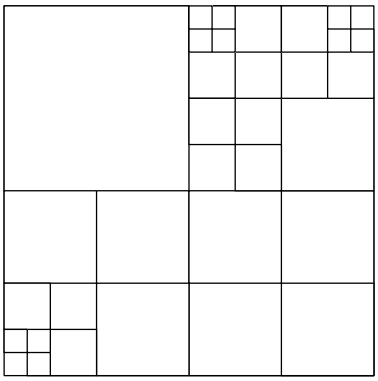


Abbildung C.1: Umgebungskartierung mithilfe von Quad-Trees. Bestimmte Gebiete werden verfeinert repräsentiert.

C.2 Sichtbarkeitsgraph

Ein Sichtbarkeitsgraph oder V-Graph wird durch Verbinden aller Hindernisecken untereinander in einer polygonalen Szene erzeugt. Es sind nur Kanten außerhalb von Hindernissen zulässig, also Kanten auf denen Sichtkontakt herrscht (vgl. Abbildung C.2 links). Die Raumgrenzen zählen ebenfalls zu den Hindernisgrenzen. Sichtbarkeitsgraphen werden zur Bahnplanung von punktförmigen Robotern benutzt, um mithilfe eines entsprechenden Suchalgorithmus den kürzesten Weg vom Start zum Ziel zu finden, welcher immer im Sichtbarkeitsgraphen enthalten ist.

Die Implementierung ist vergleichsweise einfach, jedoch besitzt der V-Graph die Komplexität O^2 und ist dadurch für große Szenen mit erhelblichem Rechen- und Speicherbedarf verbunden. Ein weiterer Nachteil dieser Methode ist die Erzeugung einer Vielzahl von Kanten, welche für die Planung selbst nicht relevant sind, vom Suchalgorithmus aber teilweise mit untersucht werden müssen. In [Arimoto 92] wird die Methode des Tangentengraphen beschrieben, welcher einen Untergraphen des V-Graphen repräsentiert [Doyle 95] (vgl. Abbildung C.2 rechts).

Um einen Tangentengraphen zu erhalten, werden aus einem V-Graphen alle ungültigen bitangenten Kanten entfernt. Eine bitangente Kante ist ungültig, falls die Verlängerung der Kante eines der beiden beteiligten Hindernisse schneidet. Der Tangentengraph ist der kleinste Graph zur Bestimmung des kürzesten Weges zwischen Hindernissen. Die Vorteile des Tangentengraphen kommen erst bei einer Suche zum Tragen, da dieser über den Situationsgraphen erzeugt wird und der Aufwand damit mindestens ebenbürtig ist.

C.3 Voronoi-Diagramme

Voronoi-Diagramme sind nach dem russischen Mathematiker Georgy Voronoi benannt und können angewandt werden, um die Fläche zwischen Objekten (Punkten, Linien, Polygonen)

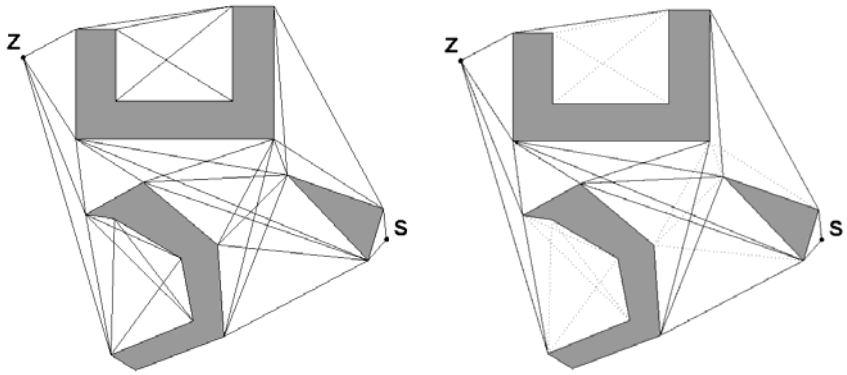


Abbildung C.2: Sichtbarkeitsgraph (links) und Tangentengraph (rechts). Im Tangentengraph werden nur die für eine Pfadsuche relevanten Kanten eingetragen. Bildquelle: [Doyle 95].

in Regionen aufzuteilen, deren Grenzen zu allen Objekten den maximal möglichen Abstand aufweisen. Diese Grenzen werden als Voronoi-Kanten bezeichnet. Handelt es sich bei den Objekten um Punkte, so bestehen die Voronoi-Kanten aus jeweils einem Liniensegment zwischen zwei Voronoi-Knoten, welches auch als *Bisektor* zwischen zwei Anziehungspunkten bezeichnet wird. Jedem Punkt wird dabei ein Raum zugeteilt, in dem der jeweilige Punkt zu allen anderen Punkten in diesem Raum am nächsten liegt. Das Voronoi-Diagramm wird in diesem Fall als *gewöhnlich* bezeichnet.

Das gewöhnliche Voronoi-Diagramm lässt sich laut Kraschl wie folgt definieren [Kraschl 05]:

> Gegeben sei eine endliche, nicht leere Menge $P = \{p_1, p_2, ..., p_n\}$ von Punkten eines Raumes R und eine Distanzfunktion $d(x_i, p_i)$. Dann enthält die Voronoi-Region $V^\circ(p)$ eines ausgezeichneten $p \in P$ alle Punkte, für die bezüglich einer Abstandsfunktion $d(x_i, p_i)$ kein anderer Punkt in P näher liegt als p, also
>
> $$V^\circ = \{x \in R : d(x, p) < d(x, q) \text{ für alle } q \in P\{o\}\} \tag{C.1}$$
>
> Mit dem Begriff *Voronoi-Diagramm* wird die Menge VD aller Voronoi-Regionen bezeichnet
>
> $$VD(P) = \bigcup_{p \in P} \{V^\circ(p)\} \tag{C.2}$$

Werden die Punkte durch Linien ersetzt, so ergibt sich eine verallgemeinerte Version des Voronoi-Diagramms mit Polygonen als Objekte. Die Voronoi-Linien oder Bisektoren werden in diesem Fall als Parabelstücke dargestellt. Auf Basis einer Belegtheitskarte oder erkannter Objekte lassen sich so Wege mit größtmöglichem Abstand zu Hindernissen errechnen, so dass diese Variante insbesondere für die Navigation von mobilen Robotern Anwendung findet.

In der Praxis kommen meist Näherungsverfahren zum Einsatz, um Voronoi-Diagramme auf Basis diskretisierter Punkte zu berechnen. Dazu werden i.d.R. das *Divide-and-Conquer*-Verfahren oder das *Plane-Sweep*-Verfahren verwendet [Shamos 75]. Das Erzeugen des

Graphen ist mit einem Rechenaufwand von $O(n \cdot log(n))$, bei einem Speicheraufwand von $O(n)$, möglich. Abbildung C.3 zeigt ein Voronoi-Diagramm des Bahnplanungsmoduls von Team AnnieWay. In dieser Anwendung diente es der Wegstreckenabschätzung in hindernisreicher Umgebung.

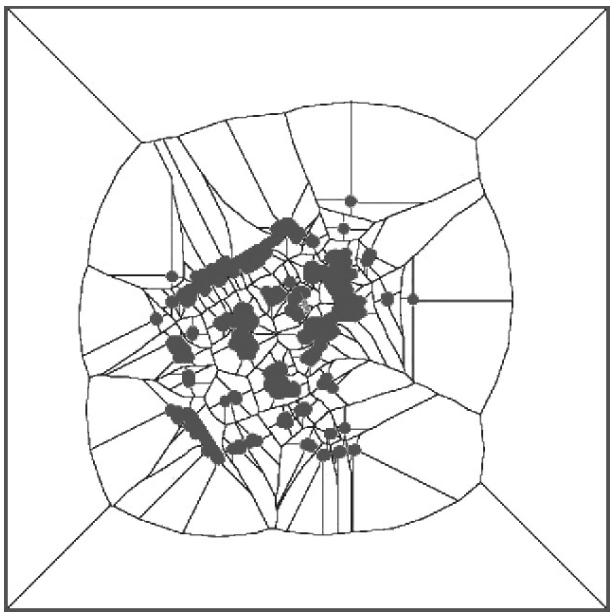


Abbildung C.3: Voronoi-Diagramm aus dem Bahnplanungsmodul von Team AnnieWay. Aufnahme eines Parkplatzes mit Hindernissen (rot) und Wege mit größtmöglichem Abstand (grün) davon. Die Ausdehung des Eigenfahrzeugs ist bereits in den Hindernissen berücksichtigt.

Voronoi-Diagramme werden neben der Verwendung in der mobilen Robotik außerdem zur Schrifterkennung, zur Darstellung von Freiräumen in der Polymerphysik und als Interpolationsverfahren in der Messtechnik verwendet. In der letztgenannten Anwendung werden mithilfe der Polygon-Methode punktförmig aufgenommene Messwerte als Fläche dargestellt und damit der Freiraum zwischen Punkten gefüllt.

C.4 Belegtheitskarten

Belegtheitskarten speichern Hindernisinformationen in einer diskreten Karte (Gitter) und zählen deshalb zu den approximativen Kartierungsverfahren. In englischsprachigen Veröffentlichungen tauchen in diesem Zusammenhang oftmals die Begriffe *Occupancy Grids* oder *Evidence Grids* auf. Letztere stellen den Ursprung dieser probabilistischen Methode zur Umweltkartierung dar und wurden von Martin Martin und Hans Moravec entwickelt, um Beweismaterial über Hindernisinformationen der Umgebung zu sammeln und diese in Form von Wahrscheinlichkeiten abzulegen [Martin 96]. Belegtheitskarten zeichnen sich besonders durch folgende Eigenschaften aus:

1. Sie zerlegen das hochdimensionale Kartierungsproblem in Einzelprobleme, die unabhängig voneinander gelöst werden können.

2. Sie machen sich bekannte, inverse Sensormodelle zunutze.

Belegtheitskarten kommen als Grundlage für eine Bahnplanung und Kartierung zum Einsatz [Konolige 99, Schiele 94, Simmons 96, Thrun 98a] und werden in der Regel aus Sensordaten von Laserscannern [Yu 05], Ultraschallsensoren [Yamauchi 97, Moravec 85, Noykov 07] oder Kameras [Correa 05, Braillon 08] generiert. Abbildung C.4 zeigt eine Belegtheitskarte von Team AnnieWay, wie sie für die Urban Challenge 2007 zur Pfadplanung verwendet wurde.

Abbildung C.4: Belegtheitskarte als Grundlage zur Pfadplanung. Verwendet von Team AnnieWay im Rahmen der Urban Challenge 2007. Freie Bereiche sind weiß dargestellt, Hindernisse schwarz. Für grau markierte Felder liegt keine oder wenig Information vor.

Für die Berechnung von Belegtheitswerten einzelner Zellen wird die Bayes'sche Wahrscheinlichkeitsrechnung verwendet. Sei m die Belegtheitskarte, dann beschreibt $m_{x,y}$ das Ereignis, dass Zelle x, y belegt ist. Belegtheitskarten werden aus mehreren Messungen $z_1 ... z_T$ geschätzt, Position und Orientierung der Sensoren sind zu den jeweiligen Zeitpunkten bekannt. Von Interesse ist nun die Wahrscheinlichkeit für die Belegtheit einer einzelnen Zelle unter Berücksichtigung der durchgeführten Messungen:

$$p(m_{x,y}|z_1, ...z_T) \tag{C.3}$$

Die hierfür zugrunde gelegte Unabhängigkeit zwischen Zellen ist real nicht gegeben und kann zu inkonsistenten Karten führen, wird aber dennoch häufig angenommen. Thrun verwendet in [Thrun 03] ein *Forward Model*, um mit statistischen Informationen widersprüchli-

che Sensordaten besser zu fusionieren. Bei Verwendung eines Evidence-Grids wird nicht die Wahrscheinlichkeit selbst, sondern die Chance

$$M(x,y) = \frac{(m_{x,y}|z_1, ...z_T)}{(\overline{m}_{x,y}|z_1, ...z_T)} \tag{C.4}$$

je Zelle hinterlegt. Aus Optimierungsgründen wird üblicherweise der Logarithmus gebildet, weil sich dadurch die Berechnungen durch Addition statt Multiplikation erheblich vereinfachen:

$$l^T_{x,y} = log\frac{p(m_{x,y}|z_1, ..., z_T)}{1 - p(m_{x,y}|z_1, ..., z_T)} \tag{C.5}$$

Auflaufende Sensordaten können dadurch einfach addiert werden. Um die Belegungswahrscheinlichkeit zu erhalten wird daraus später nur einmal zurückgerechnet:

$$p(m_{x,y}|z_1, ..., z_T) = 1 - [e^{l^T_{x,y}}]^{-1} \tag{C.6}$$

Da die Messungen akkumuliert werden müssen, wird eine rekursive Form verwendet um $l^T_{x,y}$ aktuell zu halten:

$$l^T_{x,y} = log\frac{p(m_{x,y}|z_t)}{1 - p(m_{x,y}|z_t)} + log\frac{1 - p(m_{x,y})}{p(m_{x,y})} + l^{T-1}_{x,y} \tag{C.7}$$

Als Initialisierung dient:

$$l^0_{x,y} = log\frac{1 - p(m_{x,y})}{p(m_{x,y})} \tag{C.8}$$

Die absolute Wahrscheinlichkeit für eine Zellbelegung $p(m_{x,y})$ wird abhängig von der zu erwarteten Hindernisdichte auf einen Wert zwischen 0,3 und 0,5 gesetzt. Weiterhin wird nur noch die bedingte Wahrscheinlichkeit $p(m_{x,y}, z)$ benötigt, welche die Wahrscheinlichkeit für eine Belegtheit von Zelle x, y in Abhängigkeit vom Messwert darstellt. Dieser Wert wird aus dem inversen Sensormodell gewonnen, welches die Zuordnung zwischen Messwert und Ursachen herstellt. Die Möglichkeit, verschiedene Sensorquellen mit individuellen Modellen in die Berechnung einfließen zu lassen, machen Belegtheitskarten als Werkzeug zur Sensordatenfusion besonders interessant. Eine Schwäche von Belegtheitsfeldern ist der Umgang mit dynamischen Daten. Durch die Annahme der *statischen Welt*

$$p(z_t|z_1, ..., z_{t-1}, m) = p(z_t|m) \tag{C.9}$$

wird Unabhängigkeit zwischen allen Messungen bei gleicher Karte m angenommen, was nur in den seltensten Fällen stimmt. Eine Erweiterung hinsichtlich Dynamik in der Kartierung kann mit dem sog. *Bayesian Occupancy Filter (BOF)* erreicht werden (siehe Abschnitt C.5). Je nach verwendeter Auflösung benötigen Belegtheitskarten mit festem Raster unter Umständen sehr viel Speicherplatz. Abhilfe schaffen kleinere, lokale Karten oder variable Zellgrößen abhängig von der Entfernung zum Sensorträger.

C.5 Bayesian Occupancy Filter (BOF)

Die in Abschnitt C.4 vorgestellten Belegtheitskarten haben die Berechnung eines Wahrscheinlichkeitswertes für die Zellbelegung zum Ziel. Um die Methode berechenbar zu halten wurden diese Werte als voneinander unabhängige Zufallsvariablen angenommen und jede Zelle damit gesondert betrachtet. Dieser Ansatz wurde in [Prassler 00] verwendet, um durch *Clustering* von Zellen mehrere bewegte Objekte zu verfolgen. Bei Veränderung der Zellinformationen, bspw. durch Verdeckung, konnte auch der Cluster nicht mehr aufrecht erhalten werden und die Erkennung ging verloren.

Mit dem Bayesian Occupancy Filter (BOF) wurde ein Ansatz vorgestellt, um das Problem der real nicht bestehenden Unabhängigkeit von Zellen anzugehen, und mit der Geschwindigkeit als weiteren Zellzustand einen räumlich-zeitlichen Zusammenhang zwischen Zellen zu beschreiben [Yguel 05, Chen 06]. Damit erhofft man sich eine wesentlich bessere Prädiktion von bewegten Zellen insbesondere bei temporärem Ausfall von Sensordaten durch Verdeckung. Neben der Belegtheitswahrscheinlichkeit wird die Geschwindigeit in Form einer Verteilung für jede Zelle mitgeschätzt.

Für die Aufstellung des Modells werden in [Yguel 05] folgende wahrscheinlichkeitstheoretische Variablen definiert, die im Zusammenhang mit einer einzelen Zelle c gelten:

- C ist ein Index, um die Lage einer Zelle in einem 2D-Gitter zu beschreiben.
- A ist ein Index, der alle möglichen Vorgänger einer Zelle c im 2D-Gitter beschreibt.
- $Z_t \in \mathcal{Z}$, wobei Z_t eine Zufallsvariable der Sensormesswerte relativ zu c ist.
- $V \in \mathcal{V} = \{v_1, ..., v_n\}$, wobei V eine Zufallsvariable der Geschwindigkeit für eine Zelle darstellt. Diese ist in n Schritte diskretisiert.
- $O, O^{-1} \in \mathcal{O} \equiv \{occ, emp\}$. O ist die Zufallsvariable für die Belegtheit einer Zelle c und kann entweder *belegt (occ)* oder *frei (emp)* sein. Entsprechend ist O^{-1} die Zufallsvariable für die Belegtheit des Vorgängers der Zelle c. Für eine gegebene Geschwindigkeit $v_k = (v_x, v_y)$ und einen Zeitschritt δt kann die Vorgängerzelle für $c = (x, y)$ als $c^{-1} = (x - v_x\delta t, y - v_y\delta t)$ definiert werden.

Mit diesen Definitionen ergibt sich die Verbundwahrscheinlichkeit unter Zuhilfenahme der Bayes'schen Regel zu:

$$P(C, A, Z, O, O^{-1}, V) = P(A)P(V|A)P(C|V, A)P(O^{-1}|A)P(O|O^{-1})P(Z|, O, V, C) \quad \text{(C.10)}$$

Die einzelnen Elemente besitzen dabei folgende Bedeutung:

- $P(A)$ ist die Verteilung über alle möglichen Vorgänger von c. Dieser Wert wird oftmals für alle Zellen einheitlich gewählt, um die Erreichbarkeit einer Zelle von allen Zellen aus mit gleicher Wahrscheinlichkeit anzugeben.

- $P(V|A)$ ist die Verteilung über mögliche Geschwindigkeiten eines bestimmten Vorgängers der Zelle c. Die Verteilung ist als Histogramm hinterlegt.
- $P(C|V, A)$ gibt an, ob die Zelle c von einem Vorgänger $[A = a]$ mit Geschwindigkeit $[V = v]$ erreichbar ist.
- $P(O^{-1}, A)$ ist die bedingte Verteilung der Belegtheit eines Vorgängers.
- $P(O, O^{-1})$ ist die bedingte Verteilung der Belegtheit für die aktuelle Zelle.
- $P(Z|O, V, C)$ ist die bedingte Verteilung über die Messwerte.

Ziel ist es, die Verteilung der Belegtheit und die Geschwindigkeitsinformation einer Zelle zu schätzen. Aus der globalen Filtergleichung

$$P(V, O|Z, C) = \frac{\sum_{A,O^{-1}} P(C, A, Z, O, O^{-1}, V)}{\sum_{A,O,O^{-1},V} P(C, A, Z, O, O^{-1}, V)} \tag{C.11}$$

können die notwendigen Berechnungen für Prädiktion und Schätzung abgeleitet werden. Bei bekannter Zellgeschwindigkeit wird zunächst die Wahrscheinlichkeit für *belegt* und *frei* vorausgesagt:

$$\alpha(occ, v_k) = \sum_{A,O^{-1}} P(A)P(v_k|A)P(C|A)P(O^{-1}|A) \cdot P(occ|O^{-1}) \tag{C.12}$$

$$\alpha(emp, v_k) = \sum_{A,O^{-1}} P(A)P(v_k|A)P(C|A)P(O^{-1}|A) \cdot P(emp|O^{-1}) \tag{C.13}$$

In einem zweiten Schritt wird mithilfe der Beobachtung

$$\beta(occ, v_k) = \sum_{A,O^{-1}} P(A)P(v_k|A)P(C|A)P(O^{-1}|A) \cdot P(occ|O^{-1}) \tag{C.14}$$

$$\beta(emp, v_k) = \sum_{A,O^{-1}} P(A)P(v_k|A)P(C|A)P(O^{-1}|A) \cdot P(emp|O^{-1}) \tag{C.15}$$

die Belegtheit für eine Zelle C mit Geschwindigkeit v_k geschätzt:

$$P(occ, v_k, Z, C) = \frac{\beta(occ, v_k)}{l(v_k)} \tag{C.16}$$

wobei $l(v_k) = \beta(occ, v_k) + \beta(emp, v_k)$ die Wahrscheinlichkeit für eine Geschwindigkeit v_k angibt. Ein mögliches Problem kann sich aus der Diskretisierung des Gitters ergeben, wenn sich bewegte Zellen nicht um ganzzahlige Vielfache verschieben und dadurch unterschiedlich verteilte Sprünge benachbarter Zellen entstehen. Die diskretisierten Geschwindigkeiten für V sollten sich daher an einem ganzzahligen Vielfachen einer Zellverschiebung $\frac{dx}{\delta t}$ und $\frac{dy}{\delta t}$ orientieren.

In [Yguel 05] wird die Möglichkeit vorgeschlagen, die Schätzung nicht in jedem Schritt für alle Zellen auszuführen, sondern nur für die mit ganzzahliger Zellverschiebung, also bei n benötigten Zeitschritten für eine Zellenverschiebung nur mit der Frequenz $\frac{1}{n}$. Langsam bewegten Zellen kommt dadurch weniger Aufmerksamkeit zu und entsprechend steht für schnell bewegte Zellen mehr Rechenzeit zur Verfügung.

Durch Schätzen einer Geschwindigkeitsverteilung, zusätzlich zur Belegungswahrscheinlichkeit, ist das BOF ein wichtiges Werkzeug um dynamische Zellinformationen zu prädizieren. Das Verfahren kompensiert insbesondere den großen Nachteil der Belegtheitskarten bei Verdeckungen. Mit dem BOF ist es möglich, die Objektbewegung eine Zeit lang zu prädizieren um bspw. einen Tracking-Algorithmus weiterhin mit Daten versorgen zu können. Durch die Geschwindigeitsverteilung ist zudem eine Klassifikation von Zellen möglich, so dass eine bessere Clusterbildung durch Abgrenzung möglich wird. In Kombination mit einem Kalman-Filter auf Objektebene wurden mit diesem Verfahren in [Chen 06] gute Ergebnisse für eine Personenverfolgung erzielt.

Ein interessanter Aspekt ist die Verwendung als dynamische Belegtheitskarte für eine Bahnplanungskomponente mit Zeithorizont $T \leqslant 2-3\,s$. Da nicht davon ausgegangen werden kann, dass alle relevanten Objekte und Hindernisse auf Objektebene korrekt klassifiziert werden, so kann eine Prädiktion mit BOF zumindest auf Zellebene stattfinden.

C.6 Fazit

Sichtbarkeitsgraphen und Voronoi-Diagramme beinhalten neben einer Hindernisdarstellung bereits kollisionsfreie Pfade im Freiraum und lassen deshalb, neben einer Suche auf dem vorhandenen Graphen, wenig Spielraum für Bahnplanungsmethoden oder die Integration von Straßengeometrien. Eine Verwendung im Straßenverkehr erscheint deshalb zunächst fraglich. In Bereichen ohne Straßengeometrie, wie bspw. auf Parkplätzen, könnten diese Kartierungsverfahren jedoch in Kombination mit einem Suchalgorithmus für eine Abschätzung der verbleibenden Strecke eingesetzt werden.

Quad-Trees stellen eine interessante Möglichkeit dar Speicherplatz einzusparen, allerdings lässt sich dieses Verfahren nicht mit Belegtheitskarten oder BOF kombinieren, bei welchen einheitliche Zellgrößen gefordert werden. Letztere Kartierungsverfahren bieten die Möglichkeit, unsichere Daten zu verarbeiten und Sensormodelle zu verwenden – im Hinblick auf eine möglichst realitätsnahe Abbildung ein großer Vorteil. Mit BOF besteht zudem die Möglichkeit, losgelöst von einer Objektklassifikation eine Prädiktion auf Zellebene durchzuführen. Neben einem Sicherheits-Plus kann eine Unterscheidung von Zellen anhand ihrer Geschwindigkeiten auch für Wahrnehmungskomponenten von Interesse sein.

Anhang D

Bahnplanungsalgorithmen

In diesem Abschnitt sollen grundlegende Bahnplanungsalgorithmen erklärt werden. Diese entstammen vornehmlich dem Bereich der mobilen Robotik und lassen sich mit Einschränkungen auch für Automobile verwenden. Auf spezielle Bahnplanungsverfahren für Automobile wird an dieser Stelle nicht eingegangen. Diese sind im Rahmen des Standes der Technik in Abschnitt 4.4 verzeichnet.

D.1 Potenzialfeldmethode

Die Potenzialfeldmethode [Khatib 86] ist eine besondere Form der Kartierung, die in ihrer Anwendung jedoch stark auf die Bahnplanung abzielt und deshalb eher den Bahnplanungsalgorithmen zuzuordnen ist. Bei dieser Methode wird der Roboter als Teilchen angesehen, welches, vergleichbar mit einem elektrischen Feld in der Physik, in einem Potenzialfeld bestimmten Kräften ausgesetzt ist. Hindernisse erzeugen ein abstoßendes Potenzial, der Zielpunkt ein anziehendes Potenzial. Die aus dem Gesamtpotenzial entstehende Kraft bestimmt die Bewegungsrichtung des Roboters. Das Verfahren wird oftmals mit dem Beispiel einer ins Tal rollenden Kugel veranschaulicht.

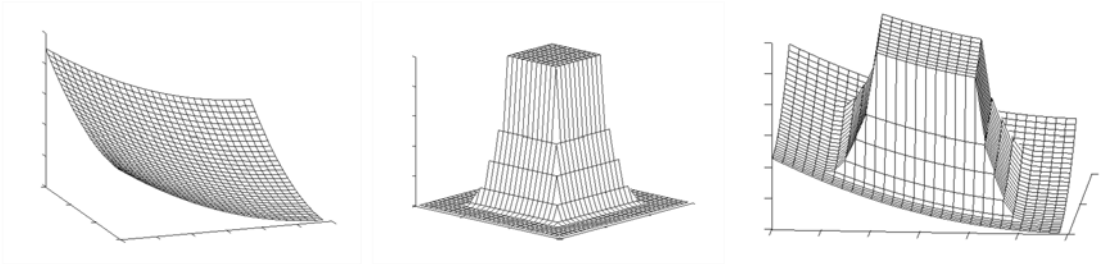


Abbildung D.1: Parabelfunktion als Hauptfeld (links), Hindernisdarstellung in Hindernisfeld (Mitte) und Gesamtfeld durch Maximumsfusion der Einzelpotenziale (rechts). Bildquelle: [Kraschl 05].

Zur Erzeugung eines Potenzialfeldes werden ein Hauptfeld und ein Hindernisfeld berechnet und diese später zu einem Gesamtfeld kombiniert (siehe Abbildung D.1). Das Hauptfeld

erzeugt ein Gefälle vom Start- zum Zielpunkt. Für die Modellierung des Gefälles wird oftmals eine Parabelfunktion angesetzt. Das Feld wird mit aktueller Position x, Zielposition x_d und Gefälle k_p folgendermaßen beschrieben:

$$U_{x_d}(x) = \frac{1}{2} k_p (x - x_d)^2 \tag{D.1}$$

Die anziehende Kraft resultiert zu:

$$\vec{F}_{x_d}(x) = -k_p (x - x_d) \tag{D.2}$$

Durch die Trennung von Haupt- und Hindernisfeld wird eine bessere Effizienz bei bewegten Hindernissen erreicht, so dass sich diese Methode gut für die dynamische Hindernisvermeidung eignet. Zur Berechnung der Hindernispotenziale dient folgende Formel:

$$U_O(x) = \begin{cases} \frac{1}{2}\eta(\frac{1}{\rho(x)} - \frac{1}{\rho_0})^2 & \rho(x) \leqslant \rho_0 \\ 0 & \rho(x) > \rho_0 \end{cases} \tag{D.3}$$

Hierin ist η ein Skalierungsfaktor, ρ_0 der kürzeste Abstand zu einem Hindernis und ρ der maximale Wirkungsabstand. Daraus resultiert die abstoßende Kraft

$$\vec{F_O}(x) = \begin{cases} \eta(\frac{1}{\rho(x)} - \frac{1}{\rho_0})\frac{1}{\rho^2}\frac{\partial \rho}{\partial x} & \rho(x) \leqslant \rho_0 \\ 0 & \rho(x) > \rho_0 \end{cases} \tag{D.4}$$

Um die Trägheit des Roboters zu integrieren wird meist ein weiterer Term F_T veranschlagt, so dass die gesamte auf den Roboter wirkende Kraft F als

$$\vec{F}(x) = \vec{F}_{x_d}(x) + \vec{F}_O(x) + \vec{F}_T(x) \tag{D.5}$$

berechnet wird. Ein bekanntes Problem der Potenzialfeldmethode ist die Enstehung lokaler Minima bei der Überlagerung beider Felder. Diese können bereits bei der Erzeugung des Feldes erkannt und aufgefüllt werden, oder es werden bei der Pfadverfolgung entsprechende Fluchtstrategien angewandt. Bspw. kann über die Verweildauer an einer bestimmten Stelle auf ein lokales Minimum geschlossen und das Umgebungspotenzial sukzessive angehoben werden.

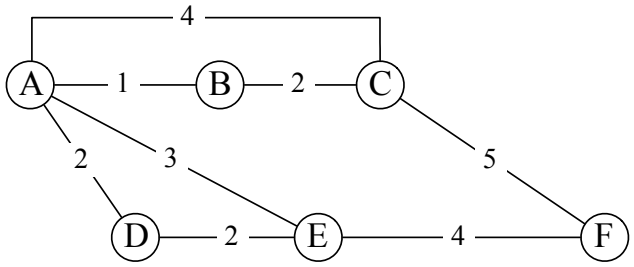
Für die Pfadsuche wird oftmals ein Gradientenabstiegsverfahren verwendet, welches allerdings die Probleme der lokalen Minima nicht löst und diese zunächst anfahren würde. Mit dem Wellenausbreitungsverfahren steht ein Verfahren zur Verfügung, um ein Potenzialfeld gänzlich ohne Minima aufzubauen [Lee 61]. Eine im Zielpunkt gestartete Wellenfront breitet sich nach dem Huygens'schen Prinzip aus und erhöht mit dem zurückgelegten Weg die Kosten bzw. das Potenzial, bis der gesamte Freiraum belegt ist.

Bei der Potenzialfeldmethode besteht neben den lokalen Minima zudem das Problem, dass über die Definition des Hauptfeldes keine weiteren Randbedingungen wie Bewegungsmodelle oder Geschwindigkeitsvorgaben eingebracht werden können. Es ist daher schwer möglich, die Bewegung zu parametrieren. Die Methode der Beschreibung von Umwelteinflüssen mittels Potenzialen eröffnet allerdings in Kombination mit anderen Kartierungsmethoden gute Möglichkeiten, um Kosten oder Risiko für verschiedene Bereiche oder Objekte darzustellen.

D.2 Dijkstra

Der Dijkstra-Algorithmus wurde nach dem niederländischen Informatiker Edsger W. Dijkstra benannt und dient der Berechnung des kürzesten Weges zwischen einem Startknoten und einem Zielknoten in einem kantengewichteten Graphen [Dijkstra 59]. Der Zielknoten kann dabei beliebig gewählt sein, die Kantengewichte müssen positiv sein. Eine typische Anwendung für diesen Algorithmus ist die Pfadsuche in Navigationssystemen auf Basis einer topographischen Karte. Als Kantengewichte werden hierbei abhängig vom Optimierungskriterium z. B. die voraussichtliche Fahrzeit oder die Kantenlänge gewählt [Cormena 04]. Für eine hindernisfreie Bahnplanung in der Robotik wird üblicherweise kein topographischer Pfad als Grundlage verwendet sondern eine Belegtheitskarte. Der Graph für die Dijkstra-Suche wird darauf erst zur Laufzeit erzeugt, wobei jeder Knoten im Suchbaum eine bestimmte Roboterkonfiguration darstellt. Grundsätzlich eignen sich damit alle gitterbasierten Kartierungsverfahren auch für eine Graphensuche, indem zusätzlich eine Vorschrift zur Erzeugung von Graphenknoten hinterlegt ist. Die Hinderniskarte beschreibt letztlich nur noch die Zulässigkeit solcher Knotenpunkte, kann aber auch eine Bewertung hinsichtlich der Güte vornehmen.

In der Regel werden für die Suche zwei Listen verwendet: Eine *offene* Liste $\mathcal{O}$ speichert Nachfolgeknoten, die selbst noch nicht besucht wurden. Die *geschlossene* Liste $\mathcal{C}$ enthält bereits abgearbeitete Knoten. Dies ist vergleichbar mit den Verzweigungen (geschlossene Liste) und Blättern (offene Liste) eines Suchbaumes. Abbildung D.2 zeigt einen Graphen mit gewichteten Kanten (links) und die Inhalte der beiden Listen nach jedem Durchlauf (rechts).



Schritt	geschlossene Liste	offene Liste
0	A	B(1),D(2),E(3),C(4)
1	A,B	D(2),E(3),C(3)
2	A,B,D	C(3),E(3)
3	A,B,D,C	E(3),F(8)
4	A,B,D,C,E	F(7)

Abbildung D.2: Graph mit gewichteten Kanten (links) und Inhalte der offenen und geschlossenen Liste nach jedem Iterationsschritt (rechts). Gesucht werden die geringsten Kosten zwischen A und allen anderen Knotenpunkten.

Mit folgendem Algorithmus können alle Kosten des Startpunktes A zu den anderen Zielpunkten berechnet werden:

1. Auswählen des Startpunktes, setzen aller Kosten in den Knoten auf ∞.

2. In die *offene* Liste werden Nachfolgeknoten mit ihren jeweiligen Kosten eingetragen, welche vom aktuell ausgewählten Knoten aus direkt erreichbar sind. Die Kosten der

Nachfolgeknoten setzen sich zusammen aus den Kosten des Vorgängerknotens und den Kosten der Verbindung. Falls der Nachfolgeknoten bereits in der Liste enthalten ist, wird der Knoten mit den geringeren Kosten beibehalten.

3. Verschieben des aktuellen Knotens in die *geschlossene* Liste.

4. Auswählen des Knotens mit den geringsten Kosten aus der *offenen* Liste, springe zu Schritt 2. Beenden, falls kein weiterer Punkt in der Liste vorhanden ist.

Damit berechnet sich die Liste kürzester Wege von A aus als:

$$A(0), B(1), C(2), D(3), E(3), F(7) \tag{D.6}$$

Da üblicherweise der genaue Pfad benötigt wird, muss zu jedem Knoten der Vorgängerknoten gespeichert werden, um bei Erreichen des Zielknotens den Pfad zurückverfolgen zu können. Ein Nachteil der ungerichteten Suche im Dijkstra-Algorithmus ist, dass in jedem Fall der gesamte Graph durchsucht werden muss. Andernfalls kann nicht garantiert werden, dass der Pfad mit den geringsten Kosten gefunden wird. Eine verwandtes Suchverfahren ist der A*-Algorithmus, der zusätzlich eine Abschätzung der verbleibenden Kosten durchführt und den kürzesten Pfad dadurch schneller findet.

D.3 A*

Der A*-Suchalgorithmus ist ein erweiterter Dijkstra-Algorithmus und wurde von Nilsson 1980 vorgestellt [Nilsson 80]. Dieses Verfahren hat sich in der Robotik zu einem Standard-Suchalgorithmus zur Pfadplanung entwickelt. Der Unterschied gegenüber dem Dijkstra-Verfahren ist die Verwendung einer Heuristikfunktion, um die verbleibenden Kosten von jedem Knoten zum Ziel abzuschätzen. Damit erfolgt eine zielgerichtete Suche, deren Aufwand bei gut gewählter Schätzfunktion erheblich unter dem von Dijkstra liegt.

In der Praxis wird bei einer Wegstreckenoptimierung oftmals der euklidische Abstand als Schätzfunktion verwendet. Je nach Kinematik des Roboters oder Wissen über die Umgebung können aber auch völlig andere Verfahren eingesetzt werden. Entscheidend ist, dass eine *zulässige* Heuristik verwendet wird, bei der die Kosten nicht überschätzt werden. Die Heuristik muss also eine untere Schranke bilden. Ist dies nicht der Fall, so kann nicht garantiert werden, dass der Pfad mit den geringsten Kosten gefunden wird. In der Praxis wird deshalb oftmals eine Heuristik gewählt, die leicht unterschätzt und dadurch den Suchbaum geringfügig vergrößert, aber die geforderte Bedingung erfüllt.

Für jeden Knoten werden bei der A*-Suche die Gesamtkosten $f(x)$ als Summe der Bewegungskosten $g(x)$ und der geschätzten Kosten $h(x)$ hinterlegt:

$$f(x) = g(x) + h(x) \tag{D.7}$$

Die Bewegungskosten müssen dazu in jedem Knoten separat vorgehalten werden. Die Suche selbst läuft nach dem in Abschnitt D.2 beschriebenen Prinzip ab, mit dem Unterschied,

dass bei Erreichen des Zielpunktes sofort abgebrochen werden kann. Abbildung D.3 zeigt die Vorteile des Verfahrens gegenüber der Verwendung des Dijkstra-Algorithmus. Als Heuristik-Funktion wurde für die gerasterte Karte die Manhattan-Distanz gewählt. Die Rechenkomplexität von A* steht in direktem Zusammenhang mit der verwendeten Heuristik. Im ungünstigsten Fall hängt die Menge der untersuchten Knoten exponentiell von der Anzahl der Knoten des kürzesten Pfades ab. Falls von der Schätzfunktion h folgende Bedingung erfüllt wird, ist der Zusammenhang polynomisch:

$$|h(x) - h^*(x)| \leqslant O(log\ h^*(x)) \tag{D.8}$$

h^* gibt dabei die optimale Heuristik an. Der Fehler in der Heuristik sollte demnach nicht größer sein als der Logarithmus der perfekten Heuristik [Russell 03]. Auch wenn der A*-Algorithmus große Vorteile gegenüber Dijkstra bietet, so hängt die Qualität entscheidend von der Wahl der Heuristikfunktion ab. Abbildung D.4 (links) zeigt, dass die Verwendung einer reinen Abstands-Heuristik ohne Einbeziehung von Hindernissen zwangsläufig einen großen Suchraum erfordert. Werden hingegen Hindernisse in der Schätzfunktion berücksichtigt, so fällt der Suchraum wesentlich kleiner aus.

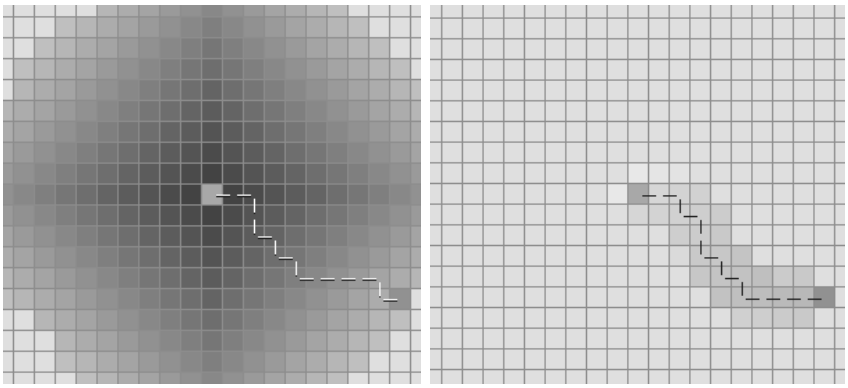


Abbildung D.3: Verwendung des Dijkstra-Algorithmus zur Pfadplanung (links) erfordert einen wesentlich größeren Suchraum (dunkel dargestellt) als das A*-Verfahren mit Manhattan-Distanz als Heuristikfunktion (rechts). Bildquelle: [Patel 08].

In vielen Veröffentlichungen wurden auf Basis des ursprünglichen Verfahrens Erweiterungen oder Optimierungen entwickelt [Trovato 92, Koenig 04] bis hin zu einer dynamischen Variante D* [Stentz 94, Stentz 95]. Neben der Verwendung in der Robotik wird A* in vielen anderen Bereichen der Informatik eingesetzt, so z. B. für die Bewegung von Figuren bei Computerspielen oder zur Routenplanung.

Im *Lifelong-Planning-A* (LPA)* werden optimale Pfade eines vollständigen Graphen berechnet, in welchem sich die Kosten der Kanten im Laufe der Zeit verändern. Bei diesem Verfahren werden zunächst von einer vorherigen Suche alle Kosten übernommen und nur die Zellen mit neuen Informationen und deren Nachfolger berechnet. Das Wissen wird also soweit möglich weitergereicht.

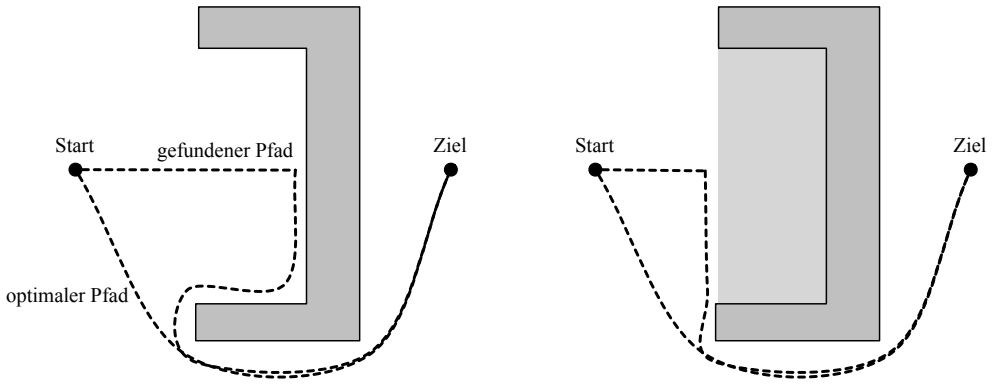


Abbildung D.4: Berücksichtigung von Hindernissen in der Schätzfunktion (rechts) kann zu wesentlich schnelleren Ergebnissen führen als eine reine Abstandsfunktion (links).

D.4 D*

Der A*-Algorithmus wurde von Anthony Stentz zum sogenannten D* [Stentz 94] oder *Dynamic A** erweitert, um einen initial mit A* geplanten Pfad neuen Umgebungsinformationen anzupassen. Dies ist insbesondere dann wünschenswert, wenn ein ursprünglich nicht einsehbares Gebiet während der Fahrt mit Sensoren erfasst wird oder wenn bewegte Objekte in der Szene auftreten. Ungültig gewordene Segmente im Pfad werde inkrementell repariert, indem die beteiligten Knotenpunkte wieder in der offenen Liste platziert werden und so neue Ansatzmöglichkeiten bieten.

Eine Erhöhung der Kosten bei jeder Neuplatzierung vermeidet die Erzeugung von lokalen Minima, so dass sich der Suchbaum über der Zeit verbreitert. Eine zusätzliche Unterscheidung der Knoten in der offenen Liste über die Zustände *Raise* und *Lower* bestimmt, inwieweit Informationen über die Pfadkosten an benachbarte Knoten propagiert werden. Der korrigierte Pfad erfüllt nach wie vor alle Anforderungen an die Optimalität, benötigt in der Regel aber nur einen Bruchteil einer kompletten Neuplanung.

Die von Stentz weiterentwickelte Variante *Focussed Dynamic A** basiert auf D*, orientiert sich aber bei der Exploration neuer Knotenpunkte am bisherigen Pfad [Stentz 95]. Er erzeugt dadurch weniger neue Knotenpunkte für die offene Liste, was wiederum zu einer geringeren Rechenzeit führt. Im sogenannten *Anytime Dynamic A**-Verfahren wird dem Umstand Rechnung getragen, dass üblicherweise nur begrenzte Rechenzeit zur Verfügung steht innerhalb der ein Ergebnis vorliegen muss [Likhachev 05]. Der zunächst sehr schnell berechnete, suboptimale Pfad wird bei diesem Algorithmus iterativ verbessert, solange Zeit bleibt.

Weitere Anpassungen des ursprünglichen D* folgten mit *D* Lite* [Koenig 02] und *Delayed D** [Ferguson 05]. Das Prinzip der initialen Pfadplanung mit inkrementeller Korrektur wird auch hier verwendet, die algorithmische Herangehensweise ist jedoch etwas unterschiedlich.

D.5 Rapidly-exploring Random-Trees (RRTs)

Rapidly-exploring Random-Trees (RRTs) sind ein Verfahren, um schnell nicht-konvexe hochdimensionale Suchbäume zu durchsuchen [LaValle 98, LaValle 01, Yershova 05]. Ausgehend von einem Startpunkt S wird rekursiv ein Baum in zufällige Richtungen aufgebaut. Dies geschieht, indem nach einer Expansionsvorschrift in Richtung eines zufällig ausgewählten Punktes R exploriert wird. Ansatzpunkt am Baum ist der zu diesem Punkt nächstliegende Knoten N. Wird ein Hindernis im Konfigurationsraum erkannt, so werden die beteiligten Zweige gelöscht und die Suche wird an den offenen Knotenpunkten fortgesetzt.

Durch den Einsatz von Expansionsvorschriften können Randbedingungen wie z. B. nicht-holonome Exploration eingebracht werden. Wurde der gesuchte Zielknoten erreicht, so kann der Pfad anhand der Baumstruktur abgelesen werden. Abbildung D.5 zeigt die Exploration eines hindernisfreien Raumes zu verschiedenen Zeitpunkten.

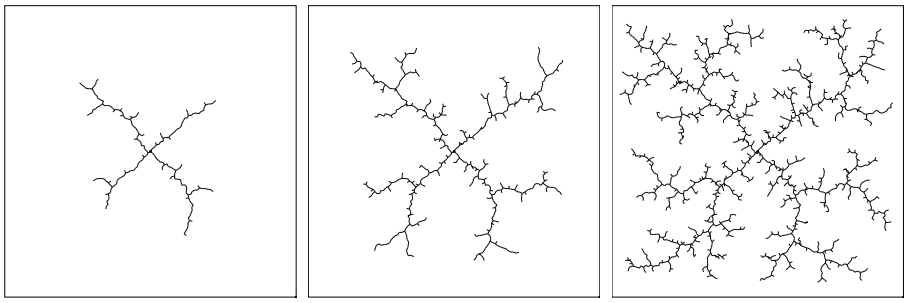


Abbildung D.5: RRT-Exploration eines hindernisfreien Raumes zu verschiedenen Zeitpunkten. Es zeigt sich die besondere Eigenschaft von RRTs, verstärkt in Richtung der unbesuchten Gebiete zu erkunden. Bildquelle: [LaValle 98].

Durch Verwendung einer Heuristik in der Expansionsvorschrift kann der Baum beschleunigt in Richtung eines Zieles aufgebaut werden. RRTs lassen sich, wie auch A*- oder D*-Suchalgorithmen, auf gitter- oder graphenbasierten Karten anwenden. Sie garantieren jedoch nicht, dass ein existierender optimaler Pfad gefunden wird. RRTs eignen sich aufgrund dieser Eigenschaften nur bedingt als eigenständige Bahnplaner sondern vielmehr als Komponente, um Möglichkeiten auszuloten. Eine Pfadbereinigung (bspw. ein Sichtbarkeitstest) ist oftmals notwendig, um glatte und gemäß eines gesetzten Kriteriums optimalere Bahnen zu erhalten, als die RRT-Ergebnisse sie liefern. Aufgrund des nahezu willkürlichen Aufbaus lässt sich für RRTs kein Aufwand für die Berechnung angeben.

Die Vorteile von RRTs werden erst bei der Suche in hochdimensionalen Räumen sichtbar. Während im zwei- oder dreidimensionalem Raum bspw. A* noch mit vertretbarem Aufwand optimale Lösungen berechnen kann, so liefern RRTs für nicht-holonome Systeme im $\mathbb{R}^{12}$ noch schnelle Lösungen. Typische Anwendungsgebiete sind daher weniger die Planung von Fahrstrecken mobiler Plattformen, sondern vielmehr die Bewegungsplanung von seriellen Roboterkinematiken.

D.6 Elastische Bänder

Das *Elastische-Bänder*-Verfahren ist kein eigenständiger Bahnplanungsalgorithmus, sondern es modifiziert einen bestehenden Pfad aufgrund von Veränderungen in der Umgebung [Quinlan 94, Khatib 93, Walther 02]. Das Verfahren wird oftmals in Kombination mit der Potenzialfeldmethode oder A*-Suche eingesetzt, um eine initiale Bahn zu planen und diese ohne komplette Neuberechnung in Echtzeit anzupassen. Der geplante Pfad wird dabei als elastisches Band angenommen, welches durch virtuell wirkende Kräfte seine Form ändern kann, um Hindernissen auszuweichen. Dadurch wird eine globale Bahnplanung mit reaktiven Eigenschaften einer Bewegungssteuerung verknüpft. Abbildung D.6 zeigt die Veränderung eines initial geplanten Pfades durch den Einfluss von Hindernissen.

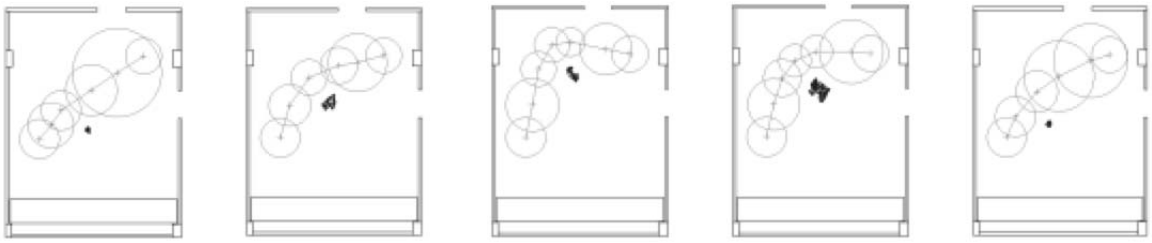


Abbildung D.6: Veränderung des geplanten Pfades unter Einfluss eines beweglichen Hindernisses. Bei höherer Dehnung entstehen zusätzliche Blasen. Bildquelle: [Walther 02].

In der sogenannten *Bubbles*-Implementierung werden Blasen verwendet, um externe Kräfte auf das Band auszuüben. Die einzelnen Blasen werden an Stützstellen des Pfades erzeugt, müssen sich gegenseitig überlappen und stellen den lokal kollisionsfreien Raum dar. Der Grad der Überlappung ist einstellbar. Der Radius einer Blase ergibt sich aus dem Abstand zum nächsten Hindernis. Über eine Mindestgröße lässt sich dadurch zudem ein Sicherheitsabstand garantieren. Kann durch den maximal möglichen Radius die Überlappung nicht mehr aufrecht erhalten werden, so müssen zusätzliche Stützstellen mit neuen Blasen eingefügt werden.

Bei zu großer Dehnung reisst das Band ab und es wird eine globale Neuplanung initialisiert. Durch die Beschränkung der Anpassung auf wenige Bereiche der Bahn ist dieses Verfahren sehr schnell und wurde in Echtzeit mit bis zu 10 Hz eingesetzt [Walther 02]. Die Zuverlässigkeit des Verfahrens hängt jedoch in großem Maße von der Komplexität der Umgebung ab. Bei sehr starken Veränderungen muss in der Regel wieder auf den globalen Planer zurückgegriffen werden.

D.7 Fazit

Die Potenzialfeldmethode liefert, neben einer Darstellung der Hindernisinformationen, durch das Hauptfeld auch eine Vorlage für eine Bahn. Dies kann mit dem Gradientenabstiegsverfahren oder der Wellenausbreitungsmethode gefunden werden. Ein kinematisches Modell

lässt sich im Suchalgorithmus nachträglich überlagern. Allerdings können sich dadurch teilweise ungewünschte Wechselwirkungen mit dem für ein punktförmiges Objekt definierten Hauptfeld ergeben. Die A*- und D*-Verfahren eignen sich gut, um nicht-holonome Kinematiken zu berücksichtigen und um zielgerichtet zu suchen. Letzterer Punkt schränkt den Suchbereich erheblich ein, so dass die Lösung schneller gefunden wird. In Kombination mit globalen Kartierungsverfahren wie Sichtbarkeitsgraph oder Voronoi-Diagramm kann eine Planung über weite Strecken sehr effizient durchgeführt werden. Vor dem Hintergrund einer begrenzten Planungszeit kommt Dijkstra aufgrund der fehlenden Abschätzung für eine schnelle Suche nicht in Frage.

Die Zufälligkeit bei der Suche mit RRTs lässt sich durch Schätzfunktionen einschränken. Es kann jedoch trotzdem nicht garantiert werden, dass ein optimaler Pfad gefunden wird. Die Verwendung von RRTs scheint geeignet, um sich in vollständig unbekannter Umgebung einen Überblick zu verschaffen – nicht jedoch, um vorhandenes Hintergrundwissen zu verwenden. Die Vorteile der schnellen Suche in hochdimensionalen Räumen lassen sich im R^2 gegenüber A* oder D* nicht aufrecht erhalten.

Das Elastische-Bänder-Verfahren eignet sich als Erweiterung für ein herkömmliches Bahnplanungsverfahren. Eine Verwendung im Straßenverkehr ist denkbar, allerdings müsste der Arbeitsbereich auf zulässige Geometrien, wie z. B. einzelne Spuren, begrenzt werden.

Anhang E

Systemarchitektur des Sonderforschungsbereiches

In diesem Kapitel wird in Abschnitt E.1 zunächst die Systemarchitektur des Sonderforschungsbereiches SFB/TR28 *Kognitive Automobile* mit den wichtigsten Komponenten vorgestellt. Abbildung E.1 zeigt eine Übersicht dazu. Es wird zudem in Abschnitt E.2 auf die Simulationsumgebung eingegangen, die im Rahmen des SFBs entwickelt wurde, und die ein elementarer Bestandteil für den Testbetrieb ist. Auf die Hardware des VW Passat von Team AnnieWay wird in Abschnitt E.3 eingegangen.

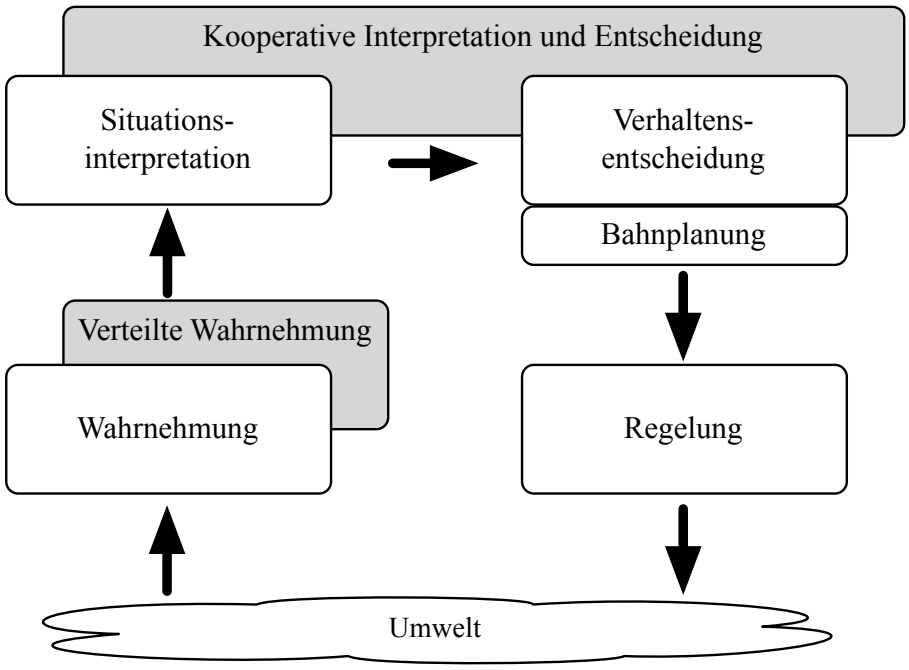


Abbildung E.1: Vereinfachte Systemarchitektur des SFB/TR28. Mit anderen kognitiven Fahrzeugen interagierende Komponenten sind grau dargestellt.

E.1 Systemkomponenten

Aufgabe der Wahrnehmungskomponenten ist es, für die Teilsysteme der kognitiven Ebene symbolische Daten bereitzustellen. Auf Grundlage der von Sensoren wie Laserscannern oder Kameras gewonnenen Daten werden Verfahren zur Klassifikation und Verfolgung von Objekten angewandt. Neben der Objekterkennung für das Eigenfahrzeug existieren auf dieser Ebene auch verteilte Wahrnehmungskomponenten, welche Daten mehrerer Sensorträger auf verschiedenen Abstraktionsstufen fusionieren.

Die von den Wahrnehmungskomponenten bereitgestellten symbolischen Daten werden von der Komponente der Situationsinterpretation miteinander verknüpft. Dabei werden die logischen Beziehungen zwischen Objekten untersucht, um ein Abbild der aktuellen Verkehrssituation zu erzeugen. Dies umfasst bspw. die Zuordnung von Fahrzeugen auf Spuren, die Identifikation von Beziehungen zwischen Verkehrsteilnehmern (bspw. *A fährt vor B* oder *A überholt B*) oder das Herausfinden von Verhalten einzelner Verkehrsteilnehmer.

Die dieserart aufbereiteten Daten können von der Komponente zur Verhaltensentscheidung zur Auswahl eines Fahrmanövers herangezogen werden. Nach einer Überprüfung von Vorbedingungen wird eine Fahrhandlung ausgewählt, die am besten zu der von der Interpretationskomponente übermittelten Teilmission passt. Jede Fahrhandlung bedient sich weiterer reaktiver Verhalten wie *Spur halten* oder *Geschwindigkeit fahren*, welche ihre Fahrintentionen in einer Gefahrenkarte eintragen. Die in der Gefahrenkarte enthaltenen Vorgaben werden durch einen Graphensuchealgorithmus in eine Bahn oder Trajektorie überführt. Dieses Teilsystem kann dabei als eigenständige Komponente betrachtet werden.

Zur globalen Lagebildbestimmung und Koordination innerhalb einer Fahrzeuggruppe dient eine kooperative Interpretations- und Entscheidungskomponente. Damit wird es möglich, in sicherheitskritischen Situationen eine kooperative Entscheidung zu treffen, welcher sich Einzelfahrzeuge unterordnen müssen. Bei einem fehlgeschlagenen Überholversuch würden die Fahrzeuge bspw. so gesteuert, dass das überholende Fahrzeug in eine entstehende Lücke einfädeln kann.

Die Regelung setzt die Vorgaben der Bahnplanung in Stellgrößen um und berücksichtigt dabei insbesondere fahrdynamische Aspekte. Aufgabe der Regelungskomponente ist es zudem, die Einhaltung zusätzlicher sicherheitsrelevanter Vorgaben sicherzustellen. Dazu zählt neben dem Anhalten an einer separat übermittelten Stopplinie und einer geschwindigkeitsabhängigen Lenkwinkelbegrenzung die Überprüfung der Funktionstüchtigkeit der Einzelkomponenten. Bei Vorgabe einer Bahn wird von der Regelung zusätzlich ein Verlauf für die Längsgeschwindigkeit erzeugt.

Neben den Teilkomponenten der Systemarchitektur existieren sogenannte Infrastrukturkomponenten, die nicht unmittelbar in den in Abb. E.1 gezeigten Informationsfluss eingebunden, für die Funktionstüchtigkeit und Nachvollziehbarkeit aber unbedingt notwendig sind:

Eine sog. Echtzeitdatenbank[1] (RTDB) dient zum Austausch von Daten zwischen Systemkomponenten und ist ein integraler Bestandteil des Systems. Die RTDB stellt ein Hilfsmittel zur Verwendung gemeinsam genutzten Arbeitsspeichers dar und bietet anderen Anwendungen eine C/C++-Schnittstelle zur Anbindung. Die Datenbank hält eine Historie aller Objekte vor und bietet die Möglichkeit, Inhalte nach Kriterien wie Namen, Typ oder Zeitstempel zu suchen. Sie ist schnell genug, um auch umfangreiche Sensordaten von Laserscannern und Kameras in Echtzeit zu verwalten.

Verschiedene, parallel zur RTDB entwickelte Werkzeuge erlauben ein Visualisieren, Aufnehmen und Abspielen von Datenbankinhalten. So können im Fahrbetrieb aufgenommene Messdaten anschließend im Labor analysiert und nachvollzogen werden. Die RTDB wurde für den Betrieb auf einem Rechnersystem konzipiert. Für den Datentausch zwischen mehreren Rechnern über Ethernet existiert eine sog. Netzwerkbrücke, die Objekte bestimmten Typs in einer entfernten RTDB erzeugt und aktualisiert.

Ein Kommunikationssystem stellt die Möglichkeit der WLAN-Verbindung zwischen Fahrzeugen untereinander sicher und wird bspw. von den kooperativen Komponenten benötigt, um entfernte Datenbankinhalte auszutauschen oder um Testergebnisse auf einem Laborrechner online nachzuvollziehen.

E.2 Simulationsumgebung

Autonome Fahrversuche mit dem Versuchsträger können nur auf einem speziellen Testgelände durchgeführt werden. Dies ist mit einem hohen zeitlichen Aufwand verbunden und je nach Verfügbarkeit des Fahrzeuges nicht immer durchführbar. Insbesondere zur Vorbereitung auf die Urban Challenge 2007 wurde deshalb eine Simulationskomponente entwickelt, um Komponenten der Systemarchitektur auch ohne das Fahrzeug testen zu können. Die Simulationsumgebung umfasst hauptsächlich die im vorigen Abschnitt angesprochene Visualisierung der RTDB und ein Fahrzeugmodell zur Umsetzung der Regelgrößen. Letzteres schließt den Regelkreis und enthält dieselben Regelalgorithmen wie sie auf dem Versuchsträger zum Einsatz kommen – das simulierte Fahrzeug verhält sich so nahezu wahrheitsgetreu.

Auf eine Simulation von Sensordaten wird aufgrund eines fehlenden Modells der Umgebung verzichtet. Es besteht jedoch die Möglichkeit, auf symbolischer Ebene mit den vorhandenen Werkzeugen synthetische Objekte in der RTDB abzulegen. Dadurch lassen sich z. B. andere Fahrzeuge über einen Joystick steuern und so gezielt Verkehrsszenen nachstellen. Eine Erzeugung mehrerer Fahrzeuginstanzen mit demselben Verhaltensmuster des Einzelfahrzeuges erlaubt eine Simulation von Verkehr und den Test der Interaktion zwischen autonomen Fahrzeugen. Abbildung E.2 zeigt die Komponente zur Visualisierung von Datenbankinhalten, die auch für die Simulation verwendet wird.

[1] Engl. Real-Time Database.

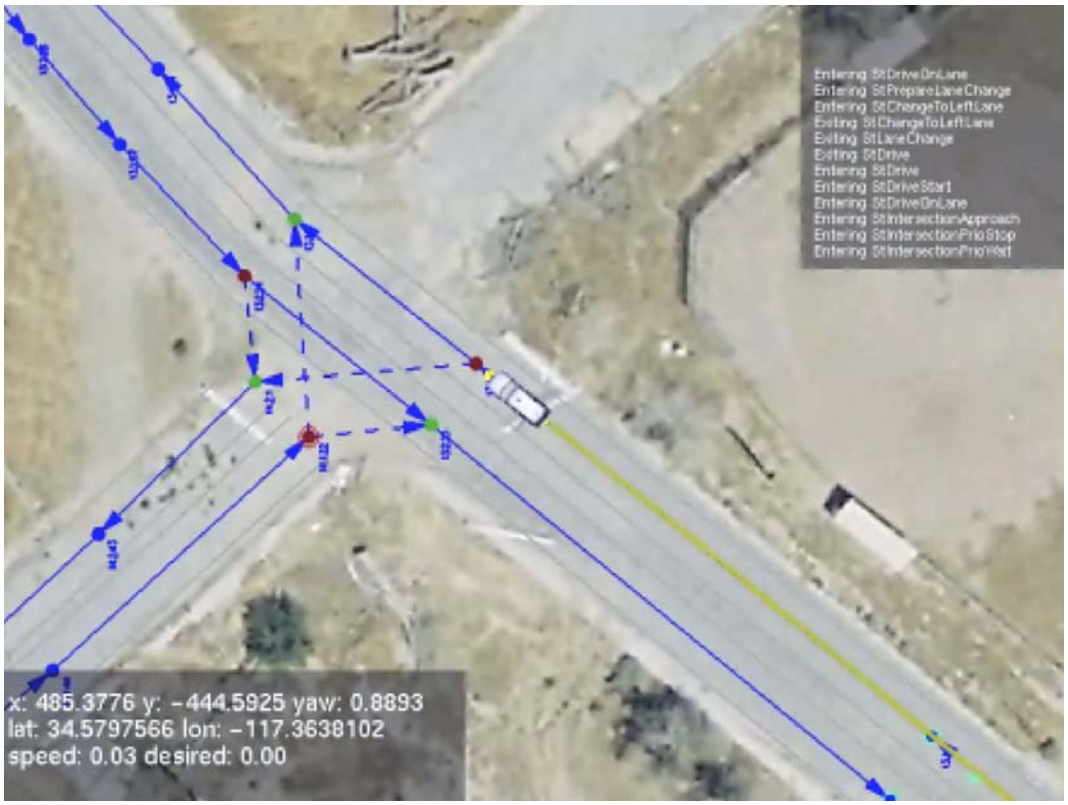


Abbildung E.2: Bildschirmfoto der Datenbankvisualisierung. Bildquelle: [Kammel 08].

E.3 Fahrzeugausstattung

Den Hauptbestandteil der Fahrzeugsteuerung bildet ein AMD-Opteron-Serversystem mit vier Kernen und einer Taktung von 2,2 GHz. Weiterhin steht ein Hauptspeicher von 4 GB zur Verfügung und eine schnelle Grafikkarte, die unter anderem zur Berechnung von Bahnplanungsalgorithmen verwendet wird. Eine *DSpace Autobox* übernimmt als separate Recheneinheit die Ausführung der Regelungsalgorithmen und bindet das Rechnersystem an die Schnittstelle zum Fahrzeug an. Als einzige Komponente besitzt die Autobox Zugriff auf den CAN-Bus des Fahrzeuges und kann Stelleingriffe vorgeben. Die für eine autonome Fahrzeugsteuerung notwendigen Veränderungen an Lenkung, Bremse und Motorsteuerung des VW Passat wurden bereits vom Hersteller vorgenommen.

Als Sensoren dienten im ersten Versuchsstadium lediglich 2D- und 3D-Laserscanner (SICK LMS 291 bzw. Velodyne HDL-64E). Mit diesen beiden Sensortypen werden Messgenauigkeiten von 1 bzw. 5 cm bei einer Entfernung von 80-100 m erreicht. Ein Inertialsystem OXTS RT3003 liefert, in Kombination mit der Fahrzeugodometrie, die Lage mit einer Genauigkeit von 20 cm in der Position und 0,1° in der Ausrichtung. Eine Sicherheitskette ist über die Aktuatorik des Fahrzeuges geführt und lässt sich sowohl über Hand- und Fußschalter, wie auch über eine Funksteuerung auslösen. Die Stromversorgung ist auf einen vollständig autarken Betrieb von bis zu 4 Stunden ausgelegt, bevor eine Versorgung über die Lichtmaschine

des Fahrzeuges oder eine externe Spannungsquelle notwendig wird. Abbildung E.3 zeigt die Ausstattung des Fahrzeuges zum Zeitpunkt der Urban Challenge 2007.

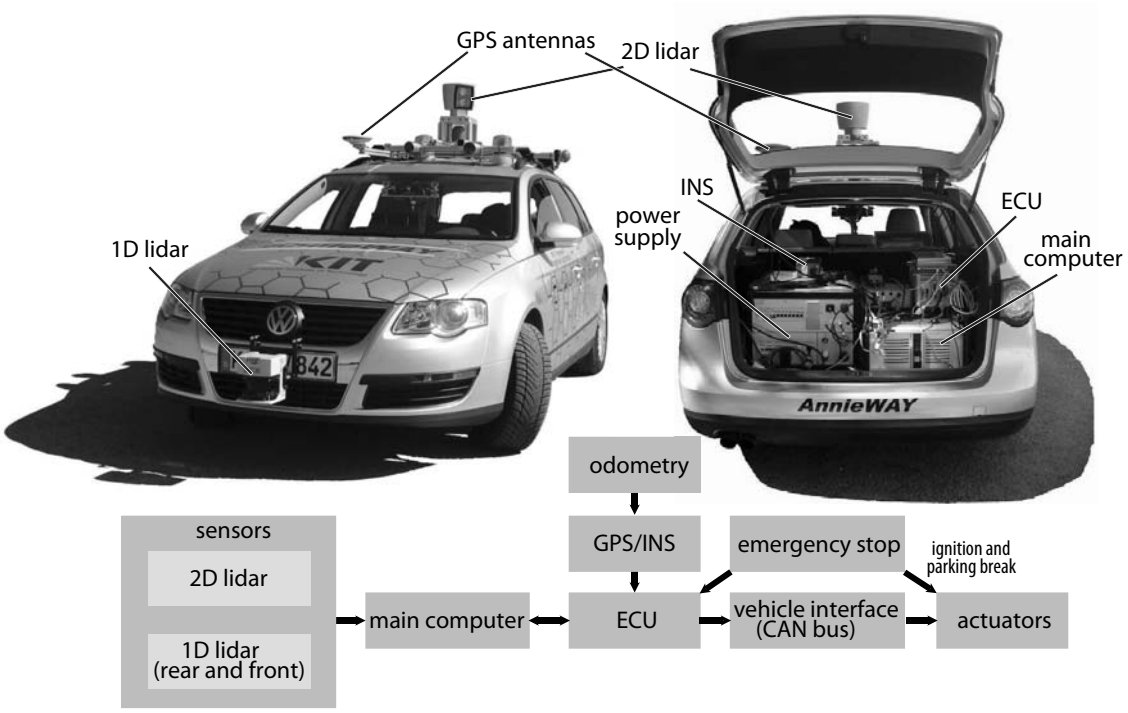


Abbildung E.3: Hardwarearchitektur des VW Passat von Team AnnieWay. Bildquelle: [Kammel 08].

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Aamodt 94] A. Aamodt, E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. *Artificial Intelligence Communications*, 7(1):39–52, 1994.

[Abel 90] D. Abel. *Petri-Netze für Ingenieure – Modellbildung und Analyse diskret gesteuerter Systeme*. Springer-Verlag, Berlin, Heidelberg, 1990.

[Albiez 01a] J. Albiez, W. Ilg, T. Luksch, K. Berns, R. Dillmann. Learning a Reactive Posture Control on the Four-Legged Walking Machine BISAM. Tagungsband: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Maui, Hawaii, USA, Oktober 2001.

[Albiez 01b] J. Albiez, T. Luksch, W. Ilg, K. Berns. Reactive Reflex-based Posture Control for a Four-Legged Walking Machine. Tagungsband: *International Conference on Climbing and Walking Robots (CLAWAR)*, Karlsruhe, September 2001.

[Albiez 06] J. Albiez. *Verhaltensnetzwerke zur adaptiven Steuerung biologisch motivierter Laufmaschinen*. Dissertation, Forschungszentrum Informatik (FZI), Universität Karlsruhe (TH), Mai 2006.

[Albus 89] J.S. Albus, H.G. McCain, R. Lumia. NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM). Technischer Bericht 1235, National Institute for Standards and Technology, Gaithersburg, USA, 1989.

[Arens 01] M. Arens. Prototypische Umsetzung einer natürlichsprachlichen Verhaltensbeschreibung in eine unscharfe metrisch temporale Logikdarstellung. Diplomarbeit, Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe (TH), 2001.

[Arens 02a] M. Arens, H.-H. Nagel. *Representation of Behavioral Knowledge for Planning and Plan-Recognition in a Cognitive Vision System*, Seiten 268–282. Lecture Notes in Computer Science: Springer-Verlag, Berlin, Heidelberg, 2002.

[Arens 02b] M. Arens, A. Ottlik, H. Nagel. Natural Language Texts for a Cognitive Vision System. Tagungsband: *15th European Conference on Artificial Intelligence (ECAI–2002)*, Seiten 21–26, Lyon, France, 21.-26. Juli 2002.

[Arimoto 92] S. Arimoto. Path Planning using a Tangent Graph for Mobile Robots among Polygonal and Curved Obstacles. *The International Journal of Robotics Research*, 11(4):376–382, 1992.

[Arkin 87] R. C. Arkin. Motor Schema-based Navigation for a Mobile Robot: An Approach to Programming by Behavior. Tagungsband: *IEEE International Conference on Robotics and Automation*, Seiten 264–271, Raleigh, North Carolina, USA, März 1987.

[Arkin 89] R. C. Arkin. Motor Schema-based Mobile Robot Navigation. *The International Journal of Robotics Research*, 8(4):92–112, 1989.

[Arkin 90] R. C. Arkin, R. R. Murphy. Autonomous Navigation in a Manufacturing Environment. *IEEE Transactions on Robotics and Automation*, 6(4):45–54, August 1990.

[Arkin 92] R. C. Arkin. Integrating Behavioral, Perceptual and World-Knowledge in reactive Navigation. *Robotics and Autonomous Systems*, 6:105–122, 1992.

[Arkin 97] R. C. Arkin, T. Balch. AuRA: Principles and Practice in Review. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 9:175–189, April 1997.

[Arkin 98] R. C. Arkin. *Behavior-based Robotics*. The MIT Press, Cambridge, Massachusetts, USA, 1998.

[Baber 05] J. Baber, J. Kolodko, T. Noel, M. Parent, L. Vlacic. Cooperative autonomous driving: intelligent vehicles sharing city roads. *IEEE Robotics and Automation Magazine*, 12(1):44–49, 2005.

[Baker 08] C. Baker, D. Ferguson, J. Dolan. Robust Mission Execution for Autonomous Urban Driving. Tagungsband: *10th International Conference on Intelligent Autonomous Systems (IAS)*, Seiten 155–163, Eindhoven, Holland, Juli 2008.

[Balch 95] T. Balch, R. Arkin. Motor Schema-based Formation Control for Multiagent Robot Teams. Tagungsband: *International Conference on Multiagent Systems*, San Francisco, Kalifornien, USA, 1995.

[Barwise 05] J. Barwise, J. Etchemendy. *Aussagen- und Prädikatenlogik*. Mentis-Verlag, Paderborn, 2005.

[Batavia 96] P. Batavia, D. Pomerleau, C. Thorpe. Applying Advanced Learning Algorithms to ALVINN. Technischer Bericht CMU-RI-TR-96-31, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Oktober 1996.

[Baumgarten 96] B. Baumgarten. *Petri-Netze – Grundlagen und Anwendungen*. Spektrum Akademischer Verlag, Heidelberg, 1996.

[Bertozzi 08] M. Bertozzi, L. Bombini, A. Broggi, P. Cerri, P. Grisleri, P. Zani. GOLD: A Complete Framework for Developing Artificial Vision Applications for Intelligent Vehicles. *IEEE Intelligent Systems*, 23(1):69–71, 2008.

[Braillon 08] C. Braillon, C. Pradalier, K. Usher, J.L. Crowley, C. Laugier. Occupancy Grids from Stereo and Optical Flow Data. *Springer Tracts in Advanced Robotics*, 39, 2008.

[Braitenberg 86] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. The MIT Press, Cambridge, USA, 1986.

[Broggi 01] A. Broggi, M. Bertozzi, G. Conte, A. Fascioli. *ARGO Prototype Vehicle*, Kapitel 14: Intelligent Vehicle Technologies. Butterworth-Heinemann, London, England, Juni 2001.

[Broggi 07] A. Broggi, P. Grisleri, C. Yakes, C. Hubert. The Oshkosh-VisLab joint Efforts on UGVs: Architecture, Integration, and Results. Tagungsband: *146th NATO Workshop on Applied Vehicle Technology Panel*, Florenz, Italien, Mai 2007.

[Broggi 99] A. Broggi, M. Bertozzi, A. Fascioli. ARGO and the MilleMiglia in Automatico Tour. *IEEE Intelligent Systems*, 14(1):55–64, Januar 1999.

[Brooks 86] R. A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, 1986.

[Brooks 87] R. A. Brooks. Intelligence without Representation. Technischer Bericht, MIT Artificial Intelligence Laboratory, Cambridge, 1987.

[Chen 06] C. Chen, C. Tay, C. Laugier, K. Mekhnacha. Dynamic Environment Modeling with Gridmap: A Multiple-Object Tracking Approach. Tagungsband: *IEEE International Conference on Control, Automation, Robotics and Vision*, Seiten 1–6, Singapore, 2006.

[Cormena 04] Th. H. Cormena, C. E. Leiserson, R. Rivesta, C. Stein. *Algorithmen – Eine Einführung*. Oldenbourg Verlag, München, 2004.

[Correa 05] F.R. Correa, J. Okamoto. Omnidirectional Stereovision System for Occupancy Grids. Tagungsband: *12th International Conference on Advanced Robotics ICAR*, Band 5, Seiten 628 – 634, Seattle, Washington, USA, 18.-20. Juli 2005.

[Dahlkamp 04] H. Dahlkamp, A. Ottlik, P. Reuter, H.-H. Nagel. Model-based Tracking in Image Sequences (Motris) – A Framework for Experiments. Technischer Bericht, Institut für Algorithmen und Kognitive Systeme, Fakultät für Informatik der Universität Karlsruhe (TH), September 2004.

[Darvin 08] Darvin. Projektbeschreibung. Online-Quelle, 2008. Erreichbar unter: `http://i21www.ira.uka.de/darvin/projektbeschreibung.html`, *Zuletzt besucht am:* 19. Februar 2009.

[Dickmanns 02] E. D. Dickmanns. Forschungsbericht. Technischer Bericht, Institut für Systemdynamik und Flugmechanik, Universität der Bundeswehr, München, 2002.

[Dickmanns 87] E. D. Dickmanns. 4D-Szenenanalyse mit integralen raumzeitlichen Modellen. Tagungsband: *9. DAGM-Symposium Mustererkennung*, London, United Kingdom, 1987.

[Dierckx 93] P. Dierckx. *Curve and surface fitting with splines*. Oxford University Press, Inc., New York, NY, USA, 1993.

[Dijkstra 59] E. W. Dijkstra. A Note on two Problems in Connexion with Graphs. *Numerische Mathematik.*, 1:269–271, 1959.

[Divelbiss 97] A.W. Divelbiss, J.T. Wen. A Path Space Approach to Nonholonomic Motion Planning in the Presence of Obstacles. *IEEE Transactions on Robotics and Automation*, 13(3):443–451, 1997.

[Doyle 95] A. Doyle. *Algorithms and Computational Techniques for Robot Path Planning*. Dissertation, University of Wales, Bangor, UK, 1995.

[Dubins 57] L. E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[ESD 08] ESD. Website ESD Electronics. Online-Quelle, 2008. Erreichbar unter: `http://www.esd-electronics.com`, *Zuletzt besucht am:* 19. Februar 2009.

[Ferguson 05] D. Ferguson, A. Stentz. The Delayed D* Algorithm for Efficient Path Replanning. Tagungsband: *IEEE International Conference on Robotics and Automation*, Seiten 2045–2050, Barcelona, Spanien, April 2005.

[Ferguson 08] D. Ferguson, C. Baker, M. Likhachev, J. Dolan. A Reasoning Framework for Autonomous Urban Driving. Tagungsband: *IEEE Intelligent Vehicles Symposium*, Seiten 775–780, Eindhoven, Holland, Juni 2008.

[Ferrel 95] C. Ferrel. Global Behavior via Cooperative Local Control. *Autonomous Robots*, 2(2):105–125, 1995.

[Finkel 74] R. Finkel, J.L. Bentley. Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4(1):1–9, 1974.

[Fraichard 04] T. Fraichard, A. Scheuer. From Reeds and Shepp's to Continuous-Curvature Paths. *IEEE Transactions on Robotics and Automation*, 20(6):1025–1035, 2004.

[French 86] S. French. *Decision Theory - An Introduction to the Mathematics of Rationality*. Halsted Press, New York, NY, USA, 1986.

[Gat 90] E. Gat, M. G. Slack, D. P. Miller, R. J. Fiby. Path Planning and Execution Monitoring for a Planetary Rover. Tagungsband: *IEEE International Conference on Robotics and Automation*, Seiten 20–25, Cincinnati, Ohio, USA, 1990.

[Gat 91] E. Gat. Integrating Reaction and Planning in a Heterogeneous Asynchronous Architecture for Mobile Robot Navigation. *SIGART Bulletin*, 2(4):70–74, 1991.

[Gat TA] E. Gat. On Three-Layer Architectures. *Artificial Intelligence and Mobile Robots*, MIT/AAAI Press, 1997.

[Gerber 00] R. Gerber. *Natürlichsprachliche Beschreibung von Straßenverkehrsszenen durch Bildfolgenauswertung*. Dissertation, Fakultät für Informatik der Universität Karlsruhe (TH), Januar 2000.

[Gonzalez 07] J. P. Gonzalez, A. Stentz. Planning with Uncertainty in Position Using High-Resolution Maps. Tagungsband: *IEEE International Conference on Robotics and Automation (ICRA)*, Rom, Italien, April 2007.

[Gonzalez 08] J. P. Gonzalez, A. Stentz. Replanning with Uncertainty in Position: Sensor Updates vs. Prior Map Updates. Tagungsband: *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, Kalifornien, USA, Mai 2008.

[Gorinevsky 96] D. Gorinevsky, A. Kapitanovsky, A. Goldenberg. Neural Network Architecture for Trajectory Generation and Control of Automated Car Parking. *IEEE Transactions on Control Systems Technology*, 4(1):50–56, 1996.

[Gregor 00a] R. Gregor, E. D. Dickmanns. EMS-Vision: Mission Performance on Road Networks. Tagungsband: *International Symposium on Intelligent Vehicles (IV)*, Dearborn, Michigan, USA, Oktober 2000.

[Gregor 00b] R. Gregor, M. Luetzeler, M. Pellkofer, K.-H. Siedersberger, E.D. Dickmanns. EMS-Vision: A Perceptual System for Autonomous Vehicles. Tagungsband: *International Symposium on Intelligent Vehicles (IV)*, Dearborn, Michigan, USA, Oktober 2000.

[Gregor 01] R. Gregor, M. Luetzeler, E.D. Dickmanns. EMS-Vision: Combining on- and off-Road Driving. Tagungsband: *SPIE Conference on Unmanned Ground Vehicle Technology III, AeroSense '01*, Orlando, Florida, USA, April 2001.

[Gregor 02] R. Gregor, M. Lützeler, M. Pellkofer, K.-H. Siedersberger, E. D. Dickmanns. EMS-Vision: A Perceptual System for Autonomous Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):48–59, März 2002.

[GT-MRL 08] GT-MRL. Website Georgia Tech Mobile Robot Laboratory. Online-Quelle, 2008. Erreichbar unter: `http://www.cc.gatech.edu/ai/robot-lab/`, *Zuletzt besucht am:* 19. Februar 2009.

[H.-H. Nagel und M. Arens 05] H.-H. Nagel und M. Arens. *Fahrerassistenzsysteme mit maschineller Wahrnehmung*, Kapitel Innervation des Automobils und Formale Logik, Seiten 89–116. Springer-Verlag, Berlin, Heidelberg, 2005.

[Heimes 98a] F. Heimes, H.-H. Nagel. Real-Time Tracking of Intersections in Image Sequences of a Moving Camera. *Special Issue on Machine Vision for Intelligent Vehicles and Autonomous Robots; Int. Journal of Engineering Applications of Artificial Intelligence (EAAI)*, 11:215–227, 1998.

[Heimes 98b] F. Heimes, H.-H. Nagel, T. Frank. Model-Based Tracking of Complex Innercity Road Intersections. *Special Issue on Intelligent Transportation Systems - Traffic Sensing and Management; Mathematical and Computer Modelling*, 9(27):189–203, 1998.

[Hoffmann 06] M. Hoffmann. Biologisch motiviertes Verhaltensnetzwerk zur Steuerung eines autonomen Automobils. Betreuer: J. Schröder. Diplomarbeit, Institut für Technische Informatik (ITEC), Universität Karlsruhe (TH), 2006.

[Hopcroft 03] J. E. Hopcroft, R. Motwani, J. D. Ullman. *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie.* Addision-Wesley, München, 2003.

[Ibaroudene 95] D. Ibaroudene, R. Acharya. Parallel Display of Objects represented by Linear Octrees. *IEEE Transactions on Parallel and Distributed Systems*, 6(1):79–85, 1995.

[IITB 08] IITB. Website Fraunhofer Institut für Informations- und Datenverarbeitung Karlsruhe. Online-Quelle, 2008. Erreichbar unter: `http://i21www.ira.uka.de/`, *Zuletzt besucht am:* 19. Februar 2009.

[J. Barraquand 89] J.-C. Latombe J. Barraquand. On Nonholonomic Mobile Robots and Optimal Maneuvering. Tagungsband: *IEEE International Symposium on Intelligent Control (ISIC89)*, Seiten 340–347, Albany, New York, USA, 25.-26. Sept. 1989.

[Jagszent 07a] D. Jagszent. Betrachtungen zur Verhaltensauswahl am Beispiel eines Abbiegevorganges. Betreuer: J. Schröder. Studienarbeit: Lehrstuhl Industrielle Anwendungen der Informatik und Mikrosystemtechnik (IAIM), Universität Karlsruhe (TH), März 2007.

[Jagszent 07b] D. Jagszent, J. Schröder, M. Zöllner, R. Dillmann. An Approach for Behaviour Selection in an Autonomous Vehicle. Tagungsband: *6th IFAC Symposium on Intelligent Autonomous Vehicles*, Toulouse, Frankreich, September 2007.

[Jagszent 08] D. Jagszent. Verhaltensgenerierung eines kognitiven Automobils mittels eines biologisch motivierten Verhaltensnetzwerks. Betreuer: J. Schröder. Diplomarbeit, Institut für Technische Informatik (ITEC), Universität Karlsruhe (TH), 2008.

[Jensen 96] F. V. Jensen. *An Introduction to Bayesian Networks*. Springer-Verlag, Berlin, Heidelberg, 1996.

[Kammel 08] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. Hundelshausen, O. Pink, C. Frese, C. Stiller. Team AnnieWAY's Autonomous System for the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):615–639, 2008.

[Kemeny 90] A. Kemeny. PROMETHEUS - Design Technics. Tagungsband: *International Congress on Transportation Electronics*, Seiten 201–207, Warrendale, PA, USA, 1990.

[Khatib 86] O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[Khatib 93] O. Khatib, S. Quinlan. Elastic Bands: Connecting Path Planning and Control. Technischer Bericht, Robotics Laboratory, Computer Science Department, Standford University, 1993.

[Koenig 02] S. Koenig, M. Likhachev. D* Lite. Tagungsband: *AAAI Conference of Artificial Intelligence (AAAI)*, Seiten 476–483, Edmonton, Alberta, Kanada, 2002.

[Koenig 04] S. Koenig, M. Likhachev, D. Furcy. Lifelong Planning A*. *Artificial Intelligence Journal*, 155(1):93–146, 2004.

[Kolodko 03] J. Kolodko, L. Vlacic. Cooperative Autonomous Driving at the Intelligent Control Systems Laboratory. *IEEE Intelligent Systems*, 18(4):8–11, 2003.

[Konolige 99] K. Konolige, K. Chou. Markov Localization using Correlation. Tagungsband: *16th International Joint Conference on Artificial Intelligence (IJCAI)*, Seiten 1154–1159, Stockholm, Schweden, 1999.

[Kraschl 05] G. Kraschl. Routenplanung für mobile Roboter. Diplomarbeit, Institut für Navigation und Satellitengeodäsie der Technischen Universität Graz, Graz, Österreich, 2005.

[Kreuzer 06] M. Kreuzer, S. Kühling. *Logik für Informatiker*. Pearson Studium, München, 2006.

[Lattner 05a] A. D. Lattner, J. D. Gehrke, I. J. Timm, O. Herzog. A Knowledge-based Approach to Behavior Decision in Intelligent Vehicles. Tagungsband: *IEEE Intelligent Vehicles Symposium*, Seiten 466–471, Las Vegas, USA, 6.-8. Juni 2005.

[Lattner 05b] A. D. Lattner, I. J. Timm, M. Lorenz, O. Herzog. Knowledge-based Risk Assessment for Intelligent Vehicles. Tagungsband: *IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, Seiten 191–196, Waltham, Massachusetts, USA, 18.-21. April 2005.

[Laugier 01] Christian Laugier, Thierry Fraichard. *Intelligent Vehicle Technologies, Chapter 11: Decisional Architectures for Motion Autonomy*. Butterworth-Heinemann, Oxford, United Kingdom, 2001.

[Laugier 89] C. Laugier, Th. Fraichard, E. Paromtchik, Ph. Garnier. Sensor-based Control Architecture for a Car-like Vehicle. Tagungsband: *IEEE-RSJ International Conference on Intelligent Robots and Systems*, Band 1, Seiten 216–222, Victoria, BC, Kanada, 1989.

[Laumond 94] J.-P. Laumond, P.E. Jacobs, M. Taix, R.M. Murray. A Motion Planner for Nonholonomic Mobile Robots. *IEEE Transactions on Robotics and Automation*, 10(5):577–593, 1994.

[LaValle 01] S. M. LaValle, J. J. Kuffner. *Algorithmic and Computational Robotics: New Directions*. A. K. Peters, Wellesley, MA, 2001.

[LaValle 98] S. M. LaValle. Rapidly-exploring Random Trees: A new Tool for Path Planning. Technical Report 98-11, Computer Science Dept., Iowa State University, Oktober 1998.

[Lee 61] C. Lee. An Algorithm for Path Connection and its Applications. *IRE Transactions on Electonic Computers EC-10*, 25(1):346–365, 1961.

[Li 03] T.-H. Li, S.-J. Chang. Autonomous Fuzzy Parking Control of a Car-like Mobile Robot. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 33(4):451–465, 2003.

[Likhachev 05] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, S. Thrun. Anytime Dynamic A*: An Anytime, Replanning Algorithm. Tagungsband: *International Conference on Automated Planning and Scheduling (ICAPS)*, Monterey, Kalifornien, USA, Juni 2005.

[Lowen 96] R. Lowen. *Fuzzy Set Theory. Basic Concepts, Techniques and Bibliography.* Kluwer Academic Publisher, Norwell, MA, USA, 1996.

[Lyon 92] D. Lyon. Parallel Parking with Curvature and Nonholonomic Constraints. Tagungsband: *IEEE Intelligent Vehicle Symposium*, Seiten 341–346, Detroit, USA, 1992.

[Martin 96] M. C. Martin, H. Moravec. Robot Evidence Grids. Technischer Bericht CMU-RI-TR-96-06, Robotics Institute, Pittsburgh, PA, März 1996.

[Mates 97] B. Mates. *Elementare Logik - Prädikatenlogik der ersten Stufe.* Vandenhoeck und Ruprecht, Göttingen, 1997.

[Michon 85] J. A. Michon. A Critical View of Driver Behavior Models: What do we know, what should we do? In L. Evans, R. Schwing, Hrsg., *Human Behavior and Traffic Safety*, Seiten 485–520. Plenum Press, New York, 1985.

[Miene 04] A. Miene, A. D. Lattner, U. Visser, O. Herzog. Dynamic-Preserving Qualitative Motion Description for Intelligent Vehicles. Tagungsband: *IEEE Intelligent Vehicles Symposium*, Seiten 642–646, Parma, Italien, 14.-17. Juni 2004.

[Mirtich 92] B. Mirtich, J. Canny. Using Skeletons for Nonholonomic Path Planning among Obstacles. Tagungsband: *IEEE International Conference on Robotics and Automation*, Band 3, Seiten 2533–2540, Nizza, Frankreich, 12.-14. Mai 1992.

[Mitchell 97] T. M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[MM 08] MM. Website Maxon Motor. Online-Quelle, 2008. Erreichbar unter: `http://www.maxonmotor.com`, *Zuletzt besucht am:* 19. Februar 2009.

[Montemerlo 07] Michael Montemerlo, Sebastian Thrun. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics.* Springer Tracts in Advanced Robotics, Springer-Verlag, Berlin, Heidelberg, 2007.

[Moravec 85] H. P. Moravec, A. Elfes. High Resolution Maps from Wide Angle Sonar. Tagungsband: *IEEE International Conference on Robotics and Automation*, Seiten 116–121, Nagoya, Japan, 1985.

[Müller 06] B. Müller, J. Deutscher. Zweistufige Trajektorienplanung für das automatische Einparken. Tagungsband: *VDI-Tagung Steuerung und Regelung von Fahrzeugen und Motoren*, Wiesloch, 2006.

[Müller 07] B. Müller, S. Grodde, J. Deutscher. Continuous Curvature Trajectory Design and Feedforward Control for Parking a Car. *IEEE Transactions on Control System Technology*, 15(3):541–553, Mai 2007.

[Murray 90] R. Murray, S. Sastry. Steering Nonholonomic Systems using Sinusoids. Tagungsband: *IEEE International Conference on Decision and Control*, Seiten 2097–2101, Honolulu, Hawaii, Dezember 1990.

[Myers 05] T. Myers, L. Vlacic, T. Noel, M. Parent. Autonomous driving in a time-varying environment. Tagungsband: *IEEE Workshop on Advanced Robotics and its Social Impacts*, Seiten 53–58, 2005.

[Nagel 03] H.-H. Nagel, M. Arens. Zur unscharf-metrisch-temporallogischen Spezifikation von Verkehrssituationen bei der Autobahnfahrt. Tagungsband: *Fahrerassistenzworkshop*, Löwenstein/Hößlinsülz, 2003.

[Nilsson 69] N.J. Nilsson. A Mobile Automaton: An Application of Artificial Intelligence Techniques. Technischer Bericht, AI Center, SRI International, 333 Ravenswood Ave, Menlo Park, CA 94025, März 1969.

[Nilsson 80] N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Company, Wellsboro, PA, USA, 1980.

[Noykov 07] S. Noykov, C. Roumenin. Occupancy Grids building by Sonar and Mobile Robot. *Robotics and Autonomous Systems*, 55(2):162–175, 2007.

[Paromtchik 96a] I. E. Paromtchik, C. Laugier. Autonomous Parallel Parking of a Nonholonomic Vehicle. Tagungsband: *IEEE Intelligent Vehicle Symposium*, Seiten 13–18, Tokyo, Japan, 1996.

[Paromtchik 96b] I. E. Paromtchik, C. Laugier. Motion Generation and Control for Parking an Autonomous Vehicle. Tagungsband: *IEEE International Conference on Robotics and Automation*, Seiten 3117–3122, Minneapolis, USA, 22.-28. April 1996.

[Patel 08] A. Patel. Website Amit's Game Programming Information. Online-Quelle, 2008. Erreichbar unter: `http://www-cs-students.stanford.edu/~amitp/gameprog.html`, *Zuletzt besucht am:* 19. Februar 2009.

[Pellkofer 00] M. Pellkofer, E.D. Dickmanns. EMS-Vision: Gaze Control in Autonomous Vehicles. Tagungsband: *International Symposium on Intelligent Vehicles (IV)*, Seiten 296–301, Dearborn, Michigan, USA, Oktober 2000.

[Pellkofer 02] M. Pellkofer, E.D. Dickmanns. Behavior Decision in Autonomous Vehicles. Tagungsband: *IEEE Intelligent Vehicle Symposium*, Band 2, Seiten 495–500, Versailles, Frankreich, Juni 2002.

[Pellkofer 03] M. Pellkofer. *Verhaltensentscheidung für autonome Fahrzeuge mit Blickrichtungssteuerung*. Dissertation, Fakultät für Luft- und Raumfahrttechnik, Universität der Bundeswehr München, München, Januar 2003.

[PNW 08] PNW. Website Petri Nets World. Online-Quelle, 2008. Erreichbar unter: `http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/`, *Zuletzt besucht am:* 19. Februar 2009.

[Pomerleau 94] D. Pomerleau. Defense and Civilian Applications of the ALVINN Robot Driving System. Tagungsband: *Government Microcircuit Applications Conference*, Seiten 358–362, San Diego, Kalifornien, USA, November 1994.

[Pomerleau 95a] D. Pomerleau. Neural Network Vision for Robot Driving. In M. Arbib, Hrsg., *The Handbook of Brain Theory and Neural Networks*, Seiten 161–181. The MIT Press, 1995.

[Pomerleau 95b] D. Pomerleau. RALPH: Rapidly Adapting Lateral Position Handler. Tagungsband: *IEEE Symposium on Intelligent Vehicles*, Seiten 506–511, Detroit, USA, September 1995.

[Prassler 00] E. Prassler, J. Scholz, A. Elfes. Tracking Multiple Moving Objects for Real-time Robot Navigation. *Autonomous Robots*, 8(2):105–116, 2000.

[Quinlan 94] S. Quinlan. *Real-time Modification of Collision-free Paths*. Dissertation, Standford University, Stanford, CA, USA, 1994.

[Reeds 90] J. A. Reeds, L. A. Shepp. Optimal Paths for a Car that goes both Forwards and Backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[Reisig 90] W. Reisig. *Petrinetze - Eine Einführung*. Springer-Verlag, Berlin, Heidelberg, 1990.

[Rich 83] E. Rich, K. Knight. *Artificial Intelligence*. McGraw-Hill, Columbus, Ohio, USA, 1983.

[Russell 03] S. J. Russell, P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[Samet 84] H. Samet. The Quadtree and Related Hierachical Data Structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260, 1984.

[Scheuer 96] A. Scheuer, Th. Fraichard. Planning Continous-Curvature Paths for Car-Like Robots. Tagungsband: *IEEE International Conference on Intelligent Robots and Systems*, Osaka, Japan, 1996.

[Scheuer 97] A. Scheuer, Th. Fraichard. Continous-Curvature Path Planning for Car-Like Vehicles. Tagungsband: *International Conference on Intelligent Robots and Systems*, Seiten 997–1003, Grenoble, Frankreich, 1997.

[Schiele 94] B. Schiele, J. Crowley. A Comparison of Position Estimation Techniques using Occupancy Grids. Tagungsband: *IEEE International Conference on Robotics and Automation*, Seiten 1628–1634, San Diego, Kalifornien, USA, 1994.

[Schröder 06] J. Schröder, U. Müller, R. Dillmann. Smart Roadster Project: Setting up Drive-by-Wire or How to Remote-Control your Car. Tagungsband: *9th International Conference on Intelligent Autonomous Systems (IAS)*, Seiten 383–390, Tokyo, Japan, 2006.

[Schröder 07] J. Schröder, M. Hoffmann, M. Zöllner, R. Dillmann. Behavior Decision and Path Planning for Cognitive Vehicles using Behavior Networks. Tagungsband: *IEEE Intelligent Vehicle Symposium*, Seiten 710–715, Istanbul, Türkei, Juni 2007.

[Schröder 08] J. Schröder, T. Gindele, D. Jagszent, R. Dillmann. Path Planning for Cognitive Vehicles. Tagungsband: *IEEE Intelligent Vehicles Symposium*, Seiten 1119–1124, Eindhoven, Holland, 2008.

[Seignez 05] E. Seignez, A. Lambert, T. Maurin. Autonomous Parking Carrier for Intelligent Vehicle. Tagungsband: *IEEE Intelligent Vehicle Symposium*, Seiten 411–416, Las Vegas, USA, 2005.

[Shamos 75] M. I. Shamos, D. Hoey. Closest-point Problems. Tagungsband: *16th Annual IEEE Symposium on Foundations of Computer Science*, Seiten 151–162, Marianske Lazne, Tschechien, 1975.

[Shirley 95] D. Shirley, J. Matijevic. Mars Pathfinder Microrover. *Autonomous Robots*, 2(4):283–289, Dezember 1995.

[Siedersberger 00] K.-H. Siedersberger, D. Dickmanns E.˙EMS-Vision: Enhanced Abilities for Locomotion. Tagungsband: *Intelligent Vehicles Symposium*, Dearborn, Michigan, USA, 3.-5. Oktober 2000.

[Simmons 95] R. Simmons. The 1994 AAAI Mobile Robot Competition. *AI Magazine*, 16(2):19–30, 1995.

[Simmons 96] R. Simmons. Where in the World is Xavier, the Robot? *Machine Perception*, 5(1):5–9, 1996.

[Stanford Racing 07] Stanford Racing. Stanford's Robotic Vehicle *Junior*: Interim Report. Technischer Bericht, Standford University, 2007.

[Stanford Racing 08] Stanford Racing. Website Stanford Racing Team. Online-Quelle, 2008. Erreichbar unter: `http://www-cs.stanford.edu/group/roadrunner/`, *Zuletzt besucht am:* 19. Februar 2009.

[Stentz 94] A. Stentz. Optimal and Efficient Path Planning for Partially-Known Environments. Tagungsband: *IEEE International Conference on Robotics and Automation*, Band 4, Seiten 3310–3317, San Diego, CA, USA, Mai 1994.

[Stentz 95] A. Stentz. The Focussed D* Algorithm for Real-Time Replanning. Tagungsband: *International Joint Conference on Artificial Intelligence*, Seiten 1652–1659, Montreal, Quebec, Kanada, August 1995.

[Sugeno 84] M. Sugeno, K. Murakami. Fuzzy Parking Control of Model Car. Tagungsband: *23rd IEEE Conference on Decision and Control*, Band 32, Seiten 902–903, Las Vegas, NV, USA, 1984.

[Tartan Racing 07] Tartan Racing. Techreport. Online-Quelle, 2007. Erreichbar unter: `http://www.darpa.mil/GRANDCHALLENGE/TechPapers/Tartan_Racing.pdf`, *Zuletzt besucht am:* 19. Februar 2009.

[Tartan Racing 08] Tartan Racing. Website Tartan Racing Team. Online-Quelle, 2008. Erreichbar unter: `http://www.tartanracing.org/`, *Zuletzt besucht am:* 19. Februar 2009.

[Team Terramax 08] Team Terramax. Website Team Terramax. Online-Quelle, 2008. Erreichbar unter: `http://www.terramax.com/`, *Zuletzt besucht am:* 19. Februar 2009.

[Thorpe 93] C. Thorpe, R. C. Coulter, M. Hebert, T. Jochem, D. Langer, D. Pomerleau, J. Rosenblatt, W. Ross, A. Stentz. Smart Cars: The CMU Navlab. Tagungsband: *WORLD MED93*, Oktober 1993.

[Thrun 03] S. Thrun. Learning Occupancy Grids with Forward Sensor Models. *Autonomous Robots*, 15:111–127, 2003.

[Thrun 05] Sebastian Thrun, Wolfram Burgard, Dieter Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, MA, USA, 2005.

[Thrun 96] S. Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers, Boston, MA, USA, 1996.

[Thrun 98a] S. Thrun. Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[Thrun 98b] S. Thrun, L.Y. Pratt. *Learning To Learn*. Kluwer Academic Publishers, Boston, MA, USA, 1998.

[ThyssenKrupp 08] ThyssenKrupp. Website ThyssenKrupp Presta Steering. Online-Quelle, 2008. Erreichbar unter: `http://www.thyssenkrupp-presta.com/`, *Zuletzt besucht am:* 19. Februar 2009.

[Trovato 92] K. I. Trovato, L. Dorst. Differential A*. *IEEE Transactions on Knowledge and Data Engineering*, 14(6):1218–1229, 1992.

[Urban Challenge 07] Urban Challenge. Website Darpa Urban Challenge. Online-Quelle, 2007. Erreichbar unter: `http://www.darpa.mil/grandchallenge/`, *Zuletzt besucht am:* 19. Februar 2009.

[Urmson 08] C. Urmson, et al. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(1):425–466, Juni 2008.

[Vacek 07] S. Vacek. *Videogestützte Umfelderfassung zur Interpretation von Verkehrssituationen für kognitive Automobile*. Dissertation, Institut für Technische Informatik (ITEC), Universität Karlsruhe (TH), Karlsruhe, 2007.

[VisLab 08] VisLab. Artificial Vision and Intelligent Systems Laboratory (VisLab), University of Parma, Italien. Online-Quelle, 2008. Erreichbar unter: `http://vislab.it/`, *Zuletzt besucht am:* 19. Februar 2009.

[Wagner 06] F. Wagner, R. Schmuki, P. Wolstenholme, T. Wagner. *Modeling Software with Finite State Machines: A Practical Approach.* Auerbach Publications, London, United Kingdom, 2006.

[Walther 02] M. Walther. Entwicklung eines hierarchischen Echtzeit-Bahnplaners für mobile Plattformen unter Verwendung von 3D-Umweltmodellen. Diplomarbeit, Fakultät für Elektrotechnik und Informationstechnik. Institut für Technik der Informationsverarbeitung. Universität Karlsruhe (TH), 2002.

[Wang 07] C.-C. Wang, C. Thorpe, M. Hebert, S. Thrun, H. Durrant-Whyte. Simultaneous Localization, Mapping and Moving Object Tracking. *The International Journal of Robotics Research*, 26(6), Juni 2007.

[Werling 08a] M. Werling, T. Gindele, D. Jagszent, L. Gröll. A Robust Algorithm for Handling Moving Traffic in Urban Scenarios. Tagungsband: *IEEE Intelligent Vehicles Symposium*, Eindhoven, Holland, 2008.

[Werling 08b] M. Werling, L. Gröll. Low-level Controllers Realizing High-Level Decisions in an Autonomous Vehicle. Tagungsband: *IEEE Intelligent Vehicles Symposium*, Eindhoven, Holland, 2008.

[Werling 08c] M. Werling, L. Gröll, G. Bretthauer. Ein Multiregler zur Erprobung vollautonomen Fahrens. *at- Automatisierungstechnik*, (11):585–591, 2008.

[Wikipedia 09] Wikipedia. Stichwort: Endlicher Zustandsautomat. Online-Quelle, 2009.

[Yamauchi 97] B. Yamauchi. A Frontier-based Approach for Autonomous Exploration. Tagungsband: *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Seiten 146–151, Monterey, Kalifornien, USA, 1997.

[Yershova 05] A. Yershova, L. Jaillet, T. Simeon, S. M. LaValle. Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain. Tagungsband: *IEEE International Conference on Robotics and Automation*, Seiten 3856–3861, Barcelona, Spanien, 2005.

[Yguel 05] M. Yguel, C. Tay, K. Mekhnacha, C. Laugier. Velocity Estimation on the Bayesian Occupancy Filter for Multi-Target Tracking. Technischer Bericht 5836, INRIA Rhone-Alpes, Grenoble, Frankreich, Januar 2005.

[Yu 05] J. Yu, Z. Cai, X. Zou, Z. Duan. *Self-localization of a Mobile Robot by Local Map Matching Using Fuzzy Logic*, Seiten 921–924. Fuzzy Systems in Expert System and Informatics. Lecture Notes in Computer Science, 2005.

[Ziegler 08] J. Ziegler, M. Werling, J. Schröder. Navigating Car-Like Robots in Unstructured Environments Using an Obstacle Sensitive Cost Function. Tagungsband: *IEEE Intelligent Vehicles Symposium*, Eindhoven, Holland, 6.-8. Juni 2008.

[Zimmermann 01] Hans-Jürgen Zimmermann. *Fuzzy Set Theory and its Applications.* Springer-Verlag, Berlin, Heidelberg, 2001.

Vor dem Hintergrund eines steigenden Verkehrsaufkommens wird der Wunsch erkennbar, nicht nur in Notsituationen Fahrer zu unterstützen, sondern in mittlerer bis ferner Zukunft Fahrzeuge autonom zu führen.

Die Existenz einer Beschreibung der Verkehrsszene auf symbolischer Ebene vorausgesetzt, stellt sich die Frage, wie sich daraus ein angemessenes Verhalten ableiten und ausführen lässt.

In der vorliegenden Arbeit wird eine biologisch motivierte Verhaltenssteuerung für ein kognitives Automobil vorgestellt, die ein hohes Maß an Rückmeldung liefert und damit die Grundlage für zukünftige Lernverfahren bildet. Weiterhin wird eine universell einsetzbare Bahnplanungskomponente entwickelt, die sich für Parkaufgaben und reguläre Fahrmanöver eignet. Wesentliches Element des vorgestellten Ansatzes ist die Repräsentation von Fahrintention und Fahrzeugumfeld mithilfe von dynamischen Gefahrenkarten, die es erlauben, Daten verschiedener Abstraktionsstufen zu vereinen.

ISBN: 978-3-86644-406-5

www.uvka.de

SBN 978-3-86644-406-5
9 783866 444065 >