Kartenbasierte Verhaltensentscheidung und Manöverplanung für automatische Fahrzeuge

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften
(Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau
des Karlsruher Instituts für Technologie (KIT)

angenommene

Dissertation

von

Dipl.-Ing. Philipp Bender

Tag der mündlichen Prüfung:	19. November 2021
Hauptreferent:	Prof. Dr.-Ing. Christoph Stiller
Korreferent:	Prof. Dr.-Ing. Marcus Geimer

Vorwort

Die dieser Dissertation zugrunde liegenden Forschungsarbeiten entstanden wärend meiner Zeit am FZI Forschungszentrum Informatik in Zusammenarbeit mit dem Institut für Mess- und Regelungstechnik am Karlsruher Institut für Technologie.

Mein besonderer Dank gilt Herrn Prof. Stiller für die wissenschaftliche Betreuung und die sehr gute Zusammenarbeit in zahlreichen Forschungsprojekten.

Herrn Prof. Geimer danke ich herzlich für das Interesse an meiner Arbeit und die Übernahme des Korreferats.

Ich wurde nach meiner Einstellung am FZI direkt Teil des Teams, welches zwei Jahre später die Bertha-Benz-Fahrt erfolgreich durchführen sollte. Diese Fahrt war eine Kooperation zwischen der Daimler AG, dem FZI und dem MRT. Ich bedanke mich bei allen Beteiligten, dass ich Teil dieses Teams und einmaligen Projekts sein durfte. Die Daimler AG hat dieses Projekt maßgeblich getragen – stellvertretend bedanke ich mich bei Frau Dr. Breuel und Herrn Prof. Dang für die stets ausgezeichnete Zusammenarbeit.

Herrn Dr. Naumann gilt mein Dank für die inhaltlichen Anmerkungen und die Unterstützung während der Fertigstellung dieser Arbeit.

Das Bild auf der linken Seite ist eine mögliche Lösung eines kombinatorischen Optimierungsproblems, des Problems des Handlungsreisenden. Die Städte sind dabei aus einer Grauwertverteilung eines Portraits von Bertha Benz gezogen.

Kurzfassung

Assistenzsysteme erschließen sich immer größere Teile des öffentlichen Verkehrsraumes und die aktuelle Entwicklung steuert in Richtung automatischer Fahrzeuge. Diese Systeme nehmen die Position des Fahrers ein, müssen Entscheidungen treffen und Aktionen auslösen. Hierfür ist es notwendig, die Situation zu verstehen, Entscheidungsmöglichkeiten zu identifizieren und für eine getroffene Entscheidung Steuersignale zu generieren.

In dieser Arbeit wird eine Kartenrepräsentation (*Lanelet*-Karten) und eine deterministische Verhaltensgenerierung vorgestellt. Die Verhaltensgenerierung bezieht die für das Verhalten relevante Information aus der Karte und setzt sie mit Beobachtungen aus der Sensorik in Beziehung. Das gewünschte Verhalten wird in Form von Nebenbedingungen an einen dem Stand der Technik entliehenen Trajektorienplaner übergeben.

Die Verhaltensentscheidung wird anhand einer vollautomatisierten Fahrt im realen Straßenverkehr erprobt. Basierend auf den Ergebnissen wird untersucht, wie ein lokales Lösungsverfahren zur Trajektorienplanung so erweitert werden kann, dass auch Situationen beherrschbar sind, in denen sich mehrere – aber nicht ineinander überführbare – Optionen eröffnen. Es werden zwei Problemformulierungen und -lösungen vorgestellt und simulativ evaluiert.

Schlüsselworte: Trajektorienplanung – Verhaltensgenerierung – Karten zum automatischen Fahren – Kombinatorische Aspekte der Bewegungsplanung

Abstract

Driver assistance systems become more and more a reality and the current development is towards automated vehicles. These systems aim to replace a human driver and need to make decisions and take actions. The systems generate control signals based on decisions they take after analyzing and comprehending the current traffic situation.

In this work, *lanelets maps* will be introduced along with a behaviour generation module which uses these maps as an input and correlates objects from the perception layer of an automated vehicle with them. The desired behaviour will be passed downstream to a planning unit in form of constraints.

The behaviour generation module is evaluated in real traffic. Based on the derived insights, extensions of local planning schemes are discussed. The aim is to also master scenes with several solutions that cannot be converted continuously into each other. This leads to two possible problem formulations which are evaluated in a simulation.

Keywords: trajectory planning – behaviour generation – maps for automated driving – combinatorial aspects of motion planning

Inhaltsverzeichnis

Kurzfassung iii

Abstract v

Abkürzungen ix

Symbolverzeichnis xi

1 Bitte einsteigen! – Einleitung 1
1.1 Zielsetzung und Beiträge der Arbeit 2
1.1.1 Gliederung 3

2 Grundlagen und Stand der Forschung und Technik 5
2.1 Wie soll ich mich bewegen? – Trajektorienplanung 5
2.1.1 Lokale Verfahren 6
2.1.2 Globale Verfahren 7
2.1.3 Randomisierte Verfahren 8
2.1.4 Kombinatorische Verfahren 9
2.1.5 Lernbasierte Verfahren 10
2.1.6 Trajektorienplanung für Bertha 11
2.2 Was soll ich tun? – Verhaltensentscheidung 14
2.2.1 Verhaltensmodellierung und -Entscheidung allgemein . 14
2.2.2 Ansätze aus der Robotik 16
2.2.3 Ansätze für das automatische Fahren 17
2.3 Wie sieht es um mich herum aus? – Umfeldrepräsentation . . . 19
2.3.1 Geometrische Umfeldmodellierung 19
2.3.2 Umfeldmodellierung mit Karten 22

3 Kartengetriebene Verhaltensentscheidung 27
3.1 Lanelets als universelle Kartendarstellung 27
3.1.1 Geometrische Modellierung 28
3.1.2 Semantische Modellierung 28
3.1.3 Einbettung der Lanelet-Karten in OpenStreetMap 33
3.1.4 Die Karte für die Bertha-Benz-Fahrt 34
3.2 Verhaltensentscheidung für Bertha 37
3.2.1 Architektur 37
3.2.2 Protokolle zur Ereignisbehandlung 39
3.2.3 Einflussnahme auf die Planung: Nebenbedingungen . . 43
3.3 Implementierung in Bertha 46

4 Manöver: Kombinatorische Aspekte der kontinuierlichen Optimierung 49
4.1 Homotopie 50
4.2 Dynamische Objekte in der statischen Welt 53
4.2.1 Problemstellung 53
4.2.2 Lösung 53
4.2.3 Experimente und Ergebnisse 63
4.3 Dynamische Objekte auf der Straße 66
4.3.1 Betrachtungen im Weg-Zeit-Diagramm 67
4.3.2 Homotopie: Begegnungsreihenfolge 70
4.3.3 Konstruktion der Nebenbedingungen 71
4.3.4 Zeitliche Konsistenz der Entscheidung für eine Homotopie 74
4.3.5 Experimente und Ergebnisse 74

5 Zusammenfassung und Ausblick 83

Literaturverzeichnis 85

Abkürzungen

UDP User Datagram Protocol

BBF Bertha-Benz-Fahrt

PVD Path-Velocity-Decomposition

FAS Fahrerassistenzsystem

RRT Rapidly Exploring Random Tree

MDP Markov Decision Process

RL Reinforcement Learning

SLSQP Sequential Least-Squares Programming

CRG Curved Regular Grid

DARPA Defense Advanced Research Projects Agency

RNDF Route Network Definition File

OSM OpenStreetMap

StVO Straßenverkehrsordnung

LSA Lichtsignalanlage

Symbolverzeichnis

κ	Krümmung (einer Kurve)
$\mathbf{x}$	Vektor
$\{a_0, v_a, l_a\}$	Fahrzeug mit der Ausgangsposition a_0, der Geschwindigkeit v_a und der Länge l_a
a^u, a^l	Längsposition des ausgedehnten (nicht als punktförmig angenommenen) Fahrzeugs a, relativ zum Ego-Fahrzeug gesehen nähere (l) und weiter entfernte (u) Koordinate)
B^u, B^l	Nebenbedingung auf Parametern (***B**ound Constraints, **l**ower, **u**pper*)
s	(an Koordinatenachsen) Longitudinale Position entlang einer Referenzlinie
x, y	(an Koordinatenachsen) Position in kartesischen Koordinaten
(●, ●, ●)	Begegnungsreihenfolge
●–●	Durch Nebenbedingungen erzwungene Paarung

1 Bitte einsteigen! – Einleitung

Das *automatisierte Fahren* ist einer der großen Trends im automobilen Umfeld. Verkehrsplaner erhoffen sich einen positiven Einfluss auf urbanen Verkehr, neue Anbieter tummeln sich im Wettstreit um zukunftsweisende Mobilitätskonzepte und Automobilhersteller wecken in den Zielgruppen zusätzlich die Bedürfnisse nach Komfort und Sicherheit. Dabei lässt sich der Grad der Automatisierung eines Kraftfahrzeugs in fünf Stufen einteilen, von *keine automatischen Funktionen* bis hin zum *fahrerlosen Fahrzeug* [GW12]. Diese Stufen sind:

Keine Automatisierung Der Mensch führt alle Fahraufgaben selbst aus.

Assistiert Der Fahrer gibt einzelne Aufgaben an das Fahrzeug ab (z. B. Längs- oder Querführung).

Teilautomatisiert Das Fahrzeug übernimmt Längs- und Querführung, wird dabei aber vom Fahrer ständig überwacht und kann die Kontrolle jederzeit an den Fahrer abgeben.

Hochautomatisiert Das Fahrzeug übernimmt die Längs- und Querführung und entlässt den Fahrer aus der Pflicht zur Überwachung, er muss jedoch im Fahrzeug sein und innerhalb eines bestimmten Zeithorizonts die Kontolle übernehmen können.

Vollautomatisiert Das Fahrzeug übernimmt die Längs- und Querführung ohne die Rückfallebene eines Fahrers. Wird einer Übernahmeaufforderung nicht nachgekommen, kann sich das Fahrzeug selbst in einen sicheren Zustand bringen.

Innerhalb der Stufen selbst gibt es mindestens zwei Aspekte, unter denen das System betrachtet werden muss: *Welche grundsätzlichen technischen Fähigkeiten*

hat das System, und *wie zuverlässig* sichert es diese zu? Diese beiden Dimensionen sind in ihrer Entwicklung nicht auf dem gleichen Niveau. Es gibt viele technische Demonstrationen des grundsätzlich Machbaren, aber nur wenige dieser Funktionen haben bisher einen Reifegrad erreicht, der es erlauben würde, ein Fahrzeug in einer der höheren Stufen einzuordnen.

In Gesellschaft und Politik sind die Entwicklungen bereits angekommen. Die Politik ist bestrebt, der Industrie einen entsprechenden Rahmen zur Innovation zu bieten, gleichzeitig ist sie aber in der Pflicht, die Bürger vor eventuellen Gefahren zu schützen.

1.1 Zielsetzung und Beiträge der Arbeit

In dieser Arbeit werden Komponenten untersucht, welche Entscheidungen treffen und welche somit unmittelbaren Einfluss auf das Verhalten des Fahrzeugs haben. Daher werden diese Komponenten *Verhaltensentscheidungs-* oder *Verhaltensgenerierungs*-Module genannt. Um Entscheidungen treffen zu können, ist eine bestimmte Informationsbasis notwendig. Hier lassen sich zwei Arten von Information unterscheiden, die sich mit *statische* Information und *dynamische* Information ganz treffend beschreiben lassen. Als illustrierendes Beispiel dient eine kleine Kreuzung, an welcher der Fahrzeugführer, egal ob Mensch oder Maschine, entscheiden muss, einem anderen Fahrzeug Vorrang zu gewähren oder nicht. Das andere Fahrzeug kommt von rechts, die Kreuzung ist nicht beschildert. Laut Straßenverkehrsordnung (StVO) ist also Vorrang zu gewähren. Das Fahrzeug ist aber recht weit entfernt und scheint außerdem zu verzögern. Der Fahrer trifft also die Entscheidung, keinen Vorrang zu gewähren. Die statische Information ist in diesem Fall die Kreuzung selbst mit ihrer Geometrie, daraus abgeleiteten Besonderheiten und mit ihrer Beschilderung. Weiterhin statisch im Kontext dieser Arbeit sollen die anzuwenden Regeln sein, wie sie von der StVO vorgegeben sind. Dynamisch hingegen ist alles, was der Fahrer erst kurz vor der Entscheidung kennt. Dazu gehört der Zustand des eigenen Fahrzeugs, die Anwesenheit anderer Verkehrsteilnehmer sowie deren Absicht.

Die wesentlichen Beiträge der Arbeit sind wie folgt:

1. Die geometrische und semantische Repräsentation der statischen Welt durch *Lanelet-Karten* als Basis für die kartengetriebene Verhaltensgenerierung beim automatischen Fahren.

2. Die auf Zustandsautomaten basierende Verhaltensgenerierung für *Bertha*, das Fahrzeug, mit welchem 2013 die erste vollautomatische Befahrung der historischen Bertha-Benz-Route gelang.

3. Eine Methode zur Segmentierung des für die Planung zulässigen Zustandsraumes, sodass kontinuierliche Planungsverfahren innerhalb diskreter Manöver initialisiert werden können und dann Lösungen finden können, die ohne eine entsprechende Initialisierung nicht auffindbar wären.

1.1.1 Gliederung

Nach den Grundlagen in den Kapiteln 2.1, 2.2 und 2.3 wird in Kapitel 3 die Kartenrepräsentation und die auf dieser basierende Verhaltensentscheidung entwickelt. Kapitel 4 richtet den Blick dann auf das Problem, dass es je nach Verkehrssituation mehrere Handlungsoptionen geben kann. Es wird eine Lösung aufgezeigt, mit der sich die kombinatorischen Aspekte vom Planungsproblem entkoppeln lassen. Dabei werden Nebenbedingungen für ein kontinuierliches Lösungsverfahren generiert und für eine Situation mehrere jeweils innerhalb ihres Manövers optimale Trajektorien erzeugt.

2 Grundlagen und Stand der Forschung und Technik

2.1 Wie soll ich mich bewegen? – Trajektorienplanung

Wird im Bereich des automatischen Fahrzeugs von *Trajektorien* gesprochen, so ist üblicherweise der Zustand eines Verkehrsteilnehmers (z. B. Fahrzeug, Fußgänger) über der Zeit gemeint. Hier soll der Begriff auf Fahrzeuge eingeschränkt werden. Der Zustand eines Fahrzeugs zu einem bestimmten Zeitpunkt entspricht einem Punkt im *Phasenraum*, also dem Raum der möglichen Zustände des Fahrzeugs. Die Trajektorie ist jetzt also die Bewegung im Phasenraum über der Zeit. Da die Trajektorie beschreibt, wie sich das Fahrzeug bewegt hat, bewegen wird oder bewegen soll (je nach Perspektive), wird die Planung derselben gelegentlich auch *Bewegungsplanung* genannt. In diesem Abschnitt werden Verfahren vorgestellt, die auch im Kontext automatischer Fahrzeuge untersucht wurden. Einen allgemeinen Überblick zum Thema Planung gibt Steven LaValle [LaV06].

Fahrzeuge sind keine holonomen Systeme, sondern unterliegen gewissen kinematischen und dynamischen Randbedingungen. Außerdem gibt es nur bestimmte Steuergrößen, mit welchen das System *Fahrzeug* beeinflusst werden kann. Üblicherweise werden Lenkwinkel und Beschleunigung als Systemeingänge gewählt. Mit einer Folge von Steuergrößen zu diskreten Zeitpunkten (Steuerfolge), einem Ausgangszustand und einem geeigneten Modell kann der Fahrzeugzustand in der Zukunft vorhergesagt und beurteilt werden. Diese Beurteilung erfolgt mit Hilfe von *Kosten-* oder *Gütefunktionen*, welche je nach Anwendungszweck formuliert werden. Eine Gütefunktion J belegt die mit Hilfe der Steuerfolge prädizierte Fahrzeugbewegung mit Kosten. Das Ziel der Optimierung einer Trajektorie ist also die Ermittlung einer Steuerfolge, welche

möglichst geringe Kosten verursacht. Diese Vorgehensweise nennt sich *modellprädiktive Regelung* [Raw00]. Dieses Grundprinzip bildet die Basis vieler Planungsverfahren für die Trajektorien automatischer Fahrzeuge.

2.1.1 Lokale Verfahren

Lokale Verfahren nutzen Eigenschaften der Zielfunktion aus: Durch iterative Suche entlang der Gradienten der Zielfunktion oder direktes, analytisches Lösen werden auch große Probleme effizient beherrscht. Die Lösungen sind jedoch *lokal*, das heißt, dass sie nicht unbedingt dem globalen Optimum der Funktion entsprechen müssen. Lokale Verfahren lassen sich mit Nebenbedingungen belegen (Abb. 2.1). Hierdurch werden bestimmte Bereiche aus dem Lösungsraum entfernt. Eine typische Problemstellung hat die Form

$$\hat{x} = \underset{x}{\operatorname{argmin}} f(x) \tag{2.1}$$

sodass

$$g_i(x) = 0, \qquad i = 1 \dots I \tag{2.2}$$

$$h_j(x) \geq 0, \qquad j = 1 \dots J \tag{2.3}$$

$$B_n^l \leq x_n \leq B_n^u, \qquad n = 1 \dots N \tag{2.4}$$

mit der zu minimierenden Funktion $f : \mathbb{R}^N \to \mathbb{R}^1$ unter den I Gleichheitsnebenbedingungen (*equality constraints*) $g_i : \mathbb{R}^N \to \mathbb{R}^m$ und den J Ungleichheitsnebenbedingungen (*inequality constraints*) $h_i : \mathbb{R}^N \to \mathbb{R}^m$. Für die zu optimierenden Variablen x gelten dabei die N Grenzen B^u (*upper*) und B^l (*lower*). Eine verbreitete Implementierung ist Sequential Least-Squares Programming (SLSQP) von Kraft [Kra94].

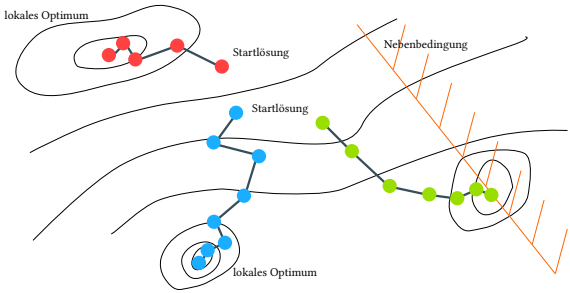


Abbildung 2.1: Lokale Verfahren können schnell konvergieren, aber (allgemein) nur zum nächstgelegenen Optimum. Es ist möglich, den Lösungsraum durch Nebenbedingungen einzuschränken.

Einen sehr guten Überblick über lokale Verfahren gibt Ziegler [Zie15]. In seiner Arbeit werden verschiedene Möglichkeiten diskutiert, wie diese Lösungsverfahren im automatischen Fahrzeug angewendet werden können. Dabei werden die Grundlagen für die später vorgestellte Planung für die Bertha-Benz-Fahrt hergeleitet.

Die Konvergenz ist in großem Maße von der Problemstellung abhängig. Es ist leicht möglich, ein Problem so zu stellen, dass ein iteratives Lösungsverfahren nie konvergieren wird oder dass keine Lösung gefunden wird, bei der alle Nebenbedingungen erfüllt sind.

Gutjahr *et al.* [GPGW16] trennen Längs- und Querführung und lösen beide Probleme mit einem quadratischen Programm. Hierdurch werden die Nebenbedingungen unter Umständen gutmütiger und es wird eine zuverlässigere Konvergenz erreicht.

2.1.2 Globale Verfahren

Globale Verfahren versuchen, die Lokalitätseigenschaft der lokalen Verfahren zu umgehen. Ein möglicher Weg ist die Diskretisierung des Phasenraums. Diskrete Gitterpunkte repräsentieren die möglichen Zustände des Systems. Die Zeit wird ebenfalls diskretisiert und der Phasenraum entsprechend erweitert. Nun werden je zwei Punkte (A, B) durch Kanten verbunden, wenn B ein Folgezustand

von A sein kann. Die Kante wird mit Kosten belegt und es wird ein möglichst günstiger Weg durch den entstandenen Graphen gesucht. Das Problem dieser Methoden ist, dass durch die Diskretisierung des Suchraums die Anzahl möglicher Knoten sehr schnell steigt. Bei einer Diskretisierung von 100 örtlichen Zuständen und je fünf für Geschwindigkeit und Beschleunigung werden pro Zeitschritt bereits 2500 Knoten benötigt – das sind über 6 Millionen potentieller Kanten *pro Zeitschritt*, welche auf Anwendbarkeit und gegebenenfalls Kosten untersucht werden müssen. Es wurden bereits effiziente Ansätze vorgestellt, die durch geschicktes Diskretisieren und Vorberechnen von Graphen schnelle Neuberechnungen ermöglichen. Allerdings werden entweder die Fahrzeugzustände sehr stark reduziert, sodass die vorgestellten Verfahren nur auf quasi-statische Anwendungen wie Einparken oder langsames Manövrieren in beengter Umgebung anwendbar sind, oder aber die zeitliche Diskretisierung wird angepasst mit dem Nachteil, dass die Lösungen recht weit von den jeweiligen Optima entfernt liegen können [Zie15, ZS09].

2.1.3 Randomisierte Verfahren

Die *randomisierten Verfahren* sollen durch geschickte, teils zufällige Variation von Zwischenlösungen Nachteile beider Verfahren überwinden und einen Kompromiss zwischen lokalen und globalen Verfahren darstellen. Von der starren örtlichen Diskretisierung wird abgewichen, was die Anzahl der Knoten stark reduziert. Dagegen werden Erfolg versprechende Gebiete stärker abgetastet (lokale Konvergenz). Globale Konvergenz wird hergestellt, indem immer ein Anteil zufälliger Lösungen gezogen wird. Beispiele solcher Methoden sind die Rapidly Exploring Random Trees (RRTs) von Stephen LaValle [LKJ00, LaV06] sowie ihre Erweiterungen RRT* und RRT*-Smart [NIM$^+$13]. Erfolgreich eingesetzt wurden RRTs zum Beispiel bei der Urban Challenge [KFT$^+$08].

Ein anderer Vertreter dieser Gruppe ist der *Particle Swarm*-Algorithmus [Ken11, PKB07]. Ein Partikel repräsentiert eine Lösung (Trajektorie) und wird anhand einer Gütefunktion bewertet. Zur Lösung des Problems werden viele Partikel erzeugt: der Partikelschwarm. Dieser Schwarm kennt die beste jemals gefundene Lösung über alle Partikel hinweg, jeder Partikel kennt seine beste jemals gefundene Lösung. Die Partikel werden iterativ angeregt und neu ausgewertet. Die Anregung (Verschiebung) setzt sich aus drei zueinander gewichteten

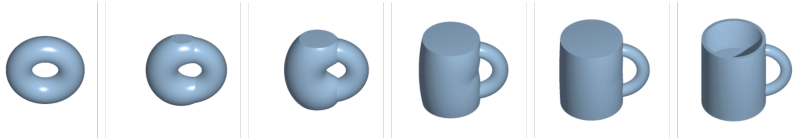


Abbildung 2.2: Es existiert eine kontinuierliche Vorschrift, welche einen Donut in eine Kaffeetasse überführt. Beide Körper sind also Teil einer Homotopie.

Teilen zusammen: einer zufälligen Verschiebung, einer Verschiebung in Richtung der besten Lösung aus Sicht des Partikels und einer Verschiebung in Richtung der besten Lösung aus Sicht des Schwarms. Die Anzahl der Partikel und deren zufällige Verteilung bestimmt die Abdeckung des Lösungsraums, die Anregung der Partikel und die Anzahl der Iterationen bestimmen das Konvergenzverhalten.

2.1.4 Kombinatorische Verfahren

Bei der Beschreibung der lokalen und globalen Verfahren klang es bereits durch: Bei der Bewegungsplanung tauchen kombinatorische Aspekte auf. Das heißt, dass lokale Optima sich nicht nur auf mathematische Eigenschaften wie Monotonie und Konvexität bestimmter Funktionen zurückführen lassen, sondern auf einer viel höheren Ebene abzählbar werden: Ein Hindernis kann links oder rechts umfahren werden. Ein Fahrzeug kann vor oder nach einem Fußgänger die Kreuzung passieren. Solche Fragestellungen, die später im Kapitel 4 ausführlich behandelt werden, sind nicht neu. Der Begriff der *Homotopie*, der auch in dieser Arbeit verwendet wird, wurde z. B. von Bhattacharya auf die Trajektorienplanung übertragen [BKL10, BLK11]. Zwei Trajektorien sind Teil einer Homotopie (man sagt: sie sind homotop zueinander), wenn eine kontinuierliche Vorschrift existiert, welche die eine in die andere überführt. Die Idee wird in Abb. 2.2 anhand einer Kaffeetasse verdeutlicht.

2.1.5 Lernbasierte Verfahren

Es ist nicht notwendig, eine Trajektorie als Schnittstelle zu verwenden. Es können auch direkt Steuerbefehle oder Steuerfolgen ermittelt werden. Lauer [Lau11] stellt eine Studie vor, bei der basierend auf der Position von Fahrstreifenrändern (aus einem Kamerabild) ein Querregler eingelernt wird. Hierfür wird Reinforcement Learning (RL) eingesetzt. Das Ziel ist das optimale Zuordnen einer Aktion (Steuerbefehl, hier: Lenkwinkel) zu einem Zustand. Dabei kann das System komplett in einem Simulator trainiert werden.

Ein anderer Ansatz ist durch den raschen Fortschritt im Bereich des Deep Learning möglich: Forscher eines Chipherstellers lernten eine Abbildung von Kamerabildfolgen auf Steuerbefehle [BDTD+16], ohne aus dem Bild weitere Information (Hindernisse, Fahrstreifen) zu extrahieren. In dieser Arbeit wurde ein Convolutional Neural Network (CNN) verwendet, welches im Gegensatz zu gewöhnlichen neuronalen Netzen räumliche Zusammenhänge von Merkmalen auswerten kann. Dies ist möglich, da Faltungsmasken trainiert werden, mit welchen die Bildinformation immer weiter zusammengefasst wird. Die theoretischen Überlegungen in diesem Bereich sind nicht neu, es ist aber so, dass die praktische Anwendung erst seit wenigen Jahren möglich ist. Der Grund ist die rechentechnisch aufwändige Trainingsphase, um die in diesem Beispiel verwendeten 250.000 Parameter zu erlernen. Der Vorteil des Ansatzes ist, dass die Sammlung der Trainingsdaten sehr einfach war. Es ist lediglich notwendig, die durch einen erfahrenen Fahrer aufgeprägten Steuerbefehle und die zugehörigen Kamerabildfolgen auszuleiten. Um jedoch in größerem Maßstab eingesetzt zu werden, muss weitere Information hinzugefügt werden, wie etwa eine grobe topologische Karte, um auch routingfähig zu werden. Das vorgestellte System kann zwar eingriffsfrei auf der Straße fahren, das Erreichen von Zielen wie das korrekte Abbiegen an Kreuzungen wurde nicht berücksichtigt.

Hubschneider *et al.* [HBD+17] erweitern diese Idee durch eine Hinderniskarte, welche bei der Fahrt aufgebaut wird. In weiteren Experimenten wird ein Fahrstreifenwechsel durchgeführt [HBWZ17]. Das so trainierte Modell wird unter dem Namen DriveNET vorgestellt.

2.1.6 Trajektorienplanung für Bertha

Die Bertha-Benz-Fahrt (BBF), die erste automatische Befahrung der Bertha-Benz-Route, war eine gemeinsame Anstrengung der Daimler AG und des FZI Forschungszentrum Informatik und des KIT im Jahr 2013. Ziel war die Befahrung der historischen Route von Mannheim nach Pforzheim, auf welcher Bertha Benz 1888 die erste Langstreckenfahrt in einem Automobil unternahm (Abb. 2.3).

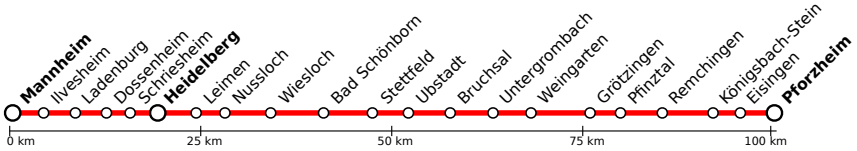


Abbildung 2.3: Ortschaften entlang der Bertha-Benz-Strecke.

Die Trajektorienplanung für dieses Projekt ist ein Vertreter der lokalen, kontinuierlichen Verfahren. Die Trajektorie wird durch zeitlich äquidistant abgetastete Orte repräsentiert, die Fahrzeugdynamik wird mit Hilfe finiter Differenzen aus diesen Orten ermittelt [ZBDS14, ZBS+14]. Die Planung arbeitet nicht direkt mit einem Fahrzeugmodell, sondern nutzt die Flachheit [FSR05] des kinematischen Einspurmodells. Durch die Wahl des Hinterachspunkts als Ausgang des Systems und die Flachheit ist es möglich, die Steuer- und Zustandsgrößen aus dem Systemausgang zu berechnen.

Die Güte der Trajektorie setzt sich dabei aus mehreren Komponenten zusammen:

- *Die Fahrt soll mit einer bestimmten Wunschgeschwindigkeit erfolgen.* Hierfür wird die Geschwindigkeit längs zum Fahrstreifen bestimmt, ihre Abweichung zur Sollgeschwindigkeit geht in das Gütekriterium ein.

- *Die Fahrt soll möglichst komfortabel sein.* Dies wird durch Summation der Beschleunigungs- und Ruckterme, die sich aus den diskreten Differenzen der Optimierungsparametern ergeben, erreicht.

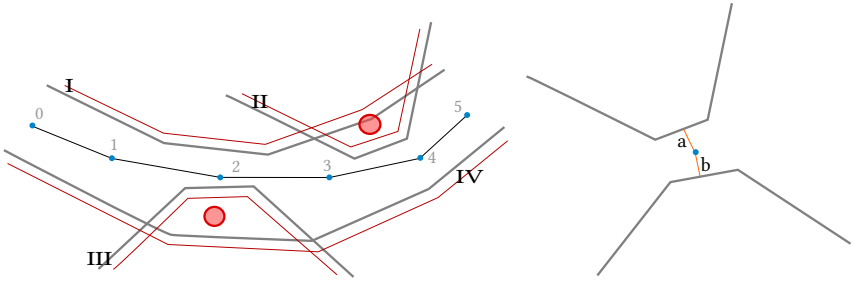


Abbildung 2.4: Modellierung des Planungsproblems durch Kostenfunktionen und Nebenbedingungen. Links eingezeichnet ist ein Fahrstreifen mit zwei Hindernissen (rote Kreise). Die Hindernisse werden durch Polygone approximiert. Die rot markierte Seite eines Polygons ist für die Planung ausgeschlossen, die Abstände zu den Polygonen gehen in das Gütekriterium mit ein (rechter Teil).

- *Das Fahrzeug soll möglichst weit entfernt von Rändern und Hindernissen fahren.* Hierfür werden die Abstände zu den Fahrstreifenrändern und Hindernissen quadriert und aufsummiert. Das Minimum liegt in der Mitte.

Außerdem werden der Trajektorie Randbedingungen auferlegt:

- *Kein Punkt der Trajektorie darf im negativen Bereich einer Abstandsfunktion liegen.* Die Abstandsfunktion ist dabei mit einem Vorzeichen versehen.

Eine für Optimierungszwecke geeignete Abstandsfunktion zu einem Polygonzug wird in den referenzierten Arbeiten ausführlich hergeleitet. Hierbei wird besonders darauf geachtet, dass nicht nur die Distanz kontinuierlich ist, sondern auch der Gradient. Durch Einführung einer Richtung des Polygonzuges ist es möglich, einen Punkt der linken oder rechten Seite des Polygonzuges zuzuordnen und entsprechend ein Vorzeichen zu vergeben.

Abbildung 2.4 zeigt diese Elemente. Die Optimierungsparameter $x_{0\ldots2N-1}$ repräsentieren eine Trajektorie der Länge N. Die Trajektorie wird durch die vier (vorzeichenbehafteten) Abstandsfunktionen I, II, III, IV beeinflusst. Negative

Abstände sind nicht erlaubt (rot markiert). Die sich aus den Abständen ergebenden Kosten sind dann klein, wenn sich das Fahrzeug von den Rändern gleich weit entfernt aufhält. Die Abstandsfunktionen I und IV modellieren den Verlauf des Fahrstreifens, die Funktionen II und III sind durch Hindernisse motiviert (rote Kreise). Die Hindernisse werden dabei heuristisch durch einen Linienzug angenähert. Zuerst wird ein Hindernis entweder dem linken oder rechten Rand zugeordnet, und dann wird ein Linienzug konstruiert, der mit Blick auf das Optimierungsverfahren möglichst gutmütig ist.

Auch eine Behandlung dynamischer Objekte ist möglich. Im Beispiel der Abbildung 2.4 gelten die Abstandsfunktionen II und III für alle Zeitpunkte, da die Hindernisse statisch sind. Im Falle einer Bewegung würden diese geeignet für die N Zeitpunkte prädiziert, und für jeden Zeitpunkt würde eine eigene Abstandsfunktion hergeleitet werden. Dies ist die zentrale Schnittstelle für die Verhaltensentscheidung, welche in Kapitel 3.2 auf Seite 37 beschrieben wird.

Diese Modellierung führt in der Praxis zu hervorragenden Ergebnissen [ZBL+14, ZBS+14, DLB+15].

2.2 Was soll ich tun? – Verhaltensentscheidung

Im Kapitel zur Trajektorienplanung wurde gezeigt, wie auf Basis eines Umfeldmodells eine optimale Trajektorie geplant werden soll. Der Sollzustand besteht aus Fahrstreifen, Geschwindigkeiten, Hindernissen, und so weiter. Der Planer nimmt diese Information entgegen, stellt keine Fragen und plant eine Trajektorie.

Doch woher kommt diese Information? Wer bestimmt, ob eine Kreuzung *vor* oder *nach* einem anderen Fahrzeug zu passieren ist, woher kennt der Planer den Fahrstreifen, wie kann er Planer die Information erhalten, die er braucht, um sich an einer Kreuzung möglichst weich einzufädeln, an einem Stoppschild diese Aktion aber erst vollzieht, nachdem regelgemäß eine Sekunde lang angehalten wurde?

In dieser Arbeit kommt diese Rolle der *Verhaltensentscheidung* zu. Stellt man sich einen Regelzyklus so vor, dass zu Beginn ersteinmal Information gesichtet und verarbeitet wird, dann ist die Verhaltensgenerierung der Punkt, ab welchem nicht gemessen, sondern reagiert wird. In Abb. 2.5 ist dies so dargestellt.

2.2.1 Verhaltensmodellierung und -Entscheidung allgemein

Eine Möglichkeit, Verhalten abhängig von einem Zustand zu modellieren, sind *Zustandsautomaten* oder auch *endliche Automaten*. Diese bestehen aus Zuständen und Zustandsübergängen, welche durch *Ereignisse* hervorgerufen werden. Ein einfaches Beispiel ist der Flaneur aus Abbildung 2.6, der nach dem Smalltalk

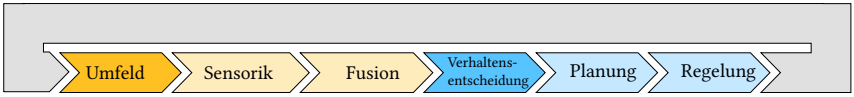


Abbildung 2.5: Mögliche Einbettung der Verhaltensentscheidung in die Fahrzeugarchitektur. Die Verhaltensentscheidung fusioniert die Wahrnehmung der Sensorik mit dem Wissen, wie Straßenverkehr „funktioniert“ und transformiert beides so, dass nachfolgende Planer und Regler arbeiten können.

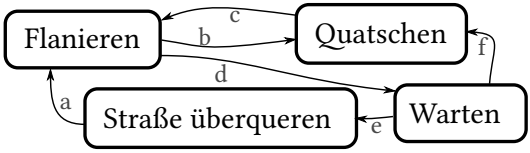


Abbildung 2.6: Endlicher Automat mit vier Zuständen und sechs Zustandsübergängen, die von Ereignissen ausgelöst werden (Ereignisse *a*, … , *f*).

vergisst, dass er eigentlich an einer Ampel wartete. In der Praxis werden solche Modelle zur Programmstrukturierung eingesetzt, indem etwa beim Betreten eines Zustandes bestimmte Routinen ausgeführt werden oder Sensoren nur in bestimmten Zuständen verfügbar sind, die Einparkhilfe hinten zum Beispiel nur bei eingelegtem Rückwärtsgang.

David Harel hatte 1987 die Idee, Zustandsautomaten hierarchisch anzuordnen. Sein Konzept nannte er *Statecharts* [Har87]. Die Bestandteile von Statecharts werden in seiner Veröffentlichung heruntergebrochen auf:

- Zustandsdiagramme,
- Tiefe (Hierarchie),
- Orthogonalität und
- Nachrichtenkanäle.

Neu ist bei Harel, dass ein Zustand durch Unterzustände verfeinert werden kann. Das System ist also in mehreren Zuständen gleichzeitig. Durch das Konzept der *Orthogonalität* ist es möglich, dass Kombinationen aus Verhaltensbausteinen getrennt voneinander modelliert werden können. In Abbildung 2.7 auf der nächsten Seite ist ein System abgebildet, welches gleichzeitig in den Zuständen A und D ist. Beide Zustände sind weiter verfeinert und die Zustandsübergänge sind voneinander unabhängig. Ohne das Konzept der Orthogonalität müssten die Zustände (B, E), (B, F), … eingeführt werden, um das gleiche Verhalten zu erzielen. In der Grafik lösen die – griechisch gesetzten – Ereignisse Zustandsübergänge

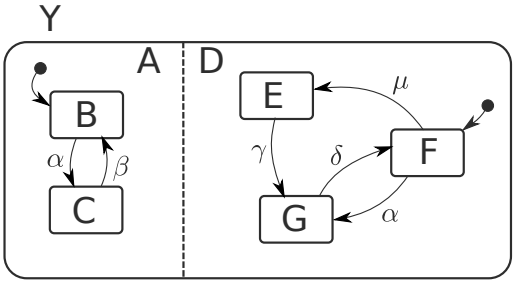


Abbildung 2.7: Orthogonalität in Statecharts, Nachbildung einer Grafik aus [Har87]. Ist das System in Zustand Y, so ist es auch gleichzeitig in A *und* D. Da es unterhalb von A und D weitere Zustände gibt, besteht der Raum möglicher Zustände aus sechs Komponenten, dem *Kreuzprodukt* der Unterzustände von A und D, $\{B, C\} \times \{E, F, G\}$ (Abbildung rechts).

aus. Ereignisse können aber auch bestimmte Verhaltensweisen innerhalb eines Zustandes auslösen und so zur eigentlichen Aufgabe, dem Entwurf reaktiver Systeme, beitragen. Eine sehr gute Implementierung der Statecharts findet sich in den C++-*Boost*-Bibliotheken.

2.2.2 Ansätze aus der Robotik

Ende der 1980er Jahre wurde die *Subsumption*-Architektur vorgeschlagen [Bro86]. Hierbei handelt es sich um ein hierarchisches System, in welchem bestimmte Verhaltensoptionen abgelegt sind. Eine Option gruppiert verschiedene Unteroptionen, welche sich alle unter der jeweiligen Option zusammenfassen oder *subsumieren* lassen. Eine Unteroption hat eine bestimmte Ausgabe, welche einerseits aus einer Aktion besteht, gleichzeitig aber auch aus einer Priorität. Diese drückt aus, wie passend die generierte Aktion in der aktuellen Situation ist. Um diese Information zu benutzen wird einer Gruppe von Unteroptionen ein *Arbitrator* zugeordnet, welcher – basierend auf der Priorität – eine Unteroption auswählt und deren Ausgabe an die nächst höhere Schicht weiterleitet. Diese Art der Systemgestaltung hat die Eigenschaft, dass die einzelnen Handlungsoptionen

leicht erweitert und verfeinert werden können und ein System entsteht, welches schnell in den Betrieb gehen und dann sukzessive weiterentwickelt werden kann. Durch die verteilten Entscheidungen der Arbitratoren für bestimmte Optionen sind die Entscheidungswege im Entwicklungsprozess allerdings auch schnell schwer nachzuvollziehen. Anwendungen dieser Architektur sind in [JCB97] zu finden. In [HLLR08] wird eine interessante Abwandlung des Musters vorgeschlagen, allerdings im Kontext strategisch agierender Fußballroboter. Dort werden auch Arbitratoren eingesetzt, die absichtlich unvorhersehbar agieren (Zufallsarbitratoren), um im Wettbewerb einen Vorteil zu erzielen.

2.2.3 Ansätze für das automatische Fahren

Im Gegensatz zur Robotik, die ein sehr breites Spektrum von Anwendungen und Rahmenbedingungen aufweist, ist der Straßenverkehr in ein recht enges Korsett geschnürt. Für jede mögliche Situation lässt sich eine Regel finden, welche im Konfliktfall zur Klärung herangezogen werden kann. Rizaldi und Althoff [RA15] formalisieren die Regeln der Wiener Konvention, sodass ein Fahrzeug eindeutig prüfen kann, ob es sich ordnungsgemäß (im Sinne des Übereinkommens) verhält oder nicht. Während dieser Beitrag sich vor allem um die Absicherung kümmert, schlagen Damm *et al.* [DPRW13] auch Designregeln zum Entwurf von Fahrstrategien vor.

Orzechowski *et al.* [OBLS21, OBL20] übertragen die Idee der Arbitrationsgraphen auf das automatische Fahren. Bausteine eines Verhaltenskataloges werden zu einem Arbitrationsgraphen kombiniert. Arbitratoren wählen Bausteine basierend auf Anwendbarkeit, Priorität oder Kosten aus. Evaluiert wurden Bausteine zum Fahrstreifenwechsel, zum Folgen des eigenen Fahrstreifens und zum sicheren Parken.

Das Team der Cornell-Universität hat für sein Fahrzeug für die Urban Challenge, *Skynet*, einen Ansatz verfolgt, bei welchem Verhalten basierend auf einer Situationsdatenbank erzeugt wird. Eine dem aktuellen Lagebild am ehesten entsprechende Situation wird ausgewählt und die ihr zugeordnete Aktion ausgeführt. Die Aktion ist hierbei über verschiedene Parameter beeinflussbar [FTO+08].

Mirchevska *et al.* [MPW+18] schlagen ein auf RL basiertes Verfahren vor, um einen Fahrstreifenwechsel zu implementieren, ohne dabei ein explizites

Regelwerk zu formulieren. Sie verwenden einen Markov Decision Process (MDP). Der Zustand besteht dabei aus den Positionen der Fahrzeuge vor und hinter dem Egofahrzeug, jeweils auf dem eigenen und den bis zu zwei benachbarten Fahrstreifen. Das System schlägt eine der drei Aktionen *Fahrstreifenwechsel links/rechts* und *beibehalten* vor.

2.3 Wie sieht es um mich herum aus? – Umfeldrepräsentation

Die Verhaltensentscheidung und die Planung operieren auf einem Modell des Zustandes der Welt, dieses Modell soll hier *Umfeldmodell* oder *Umfeldrepräsentation* genannt werden. Es besteht aus statischen und dynamischen Teilen.

Die Welt um das Fahrzeug herum lässt sich durch die Augen der Geometrie und der Topologie betrachten. Die geometrische Betrachtung beschäftigt sich mit Positionen und Abmessungen, während die Topologie diese Information in einen größeren Zusammenhang setzt. Ein vermessenes Stück Asphalt vor dem Fahrzeug bildet mit anderen Asphaltstücken eine zusammenhängende Fläche, eine weiße Linie teilt die Fläche in Fahrstreifen. Zum Fahren werden beide Informationen für unterschiedliche Aspekte benötigt und müssen entsprechend modelliert und repräsentiert werden. Beide Bereiche lassen sich nicht scharf trennen, weshalb dieses Kapitel hauptsächlich die allgemeine Modellierung und die Darstellung in Form einer Karte unterscheidet.

2.3.1 Geometrische Umfeldmodellierung

Wurm *et al.* repräsentieren die 3-D-Welt durch *octrees*, durch deren Einsatz sie beliebige Detailgrade erzielen können [HWB+13]. In ihrer Arbeit präsentieren sie sowohl einen Kartierungsansatz als auch eine speichereffiziente Implementierung. Durch die probabilistische Repräsentation eines Zellzustandes können mit Unsicherheit behaftete Sensormesswerte unkompliziert integriert werden. Ein Octree ist ein Baum, in dem ein Knoten entweder keinen oder acht Nachfolger hat. Ein Knoten ohne Nachfolger, also ein Blatt, repräsentiert den Zustand eines bestimmten Volumenelements in der dreidimensionalen Welt. Ist der Zustand innerhalb einer solchen Zelle nicht homogen, so wird sie pro Dimension jeweils mittig geteilt. Der ursprüngliche Knoten ist nun kein Blatt mehr, sondern ein Knoten mit acht Nachfolgern. Dies wird so lange wiederholt, bis jede Zelle als ausreichend homogen angesehen wird oder eine bestimmte maximale Tiefe erreicht ist. Ein Octree ist kein statisches Gebilde, sondern wird durch Hinzufügen oder Entfernen von Information ständig neu organisiert. Knoten mit Nachfolgern werden in der Graphentheorie auch *innere Knoten* genannt.

Bedarf für die Modellierung von Fahrstreifengeometrien kommt auch aus dem Perzeptionsbereich. Hier werden etwa zur Fahrstreifendetektion *Klothoiden* eingesetzt, um die Ränder zu beschreiben [DM92]. Weitere Ansätze zur Beschreibung von Rändern sind *line snakes* [KCK96] sowie Parabelpaare mit konstanter Krümmung [WDA12].

Klothoiden bieten sich an, weil sie ein häufig verwendetes Element im Straßenbau darstellen. Es handelt sich dabei um Kurven mit konstanter *Krümmungsänderung* relativ zum zurückgelegten Weg. Diese konstante Krümmungsänderung führt zu einer stetigen Veränderung des Lenkwinkels und damit einer ruckarmen Fahrdynamik. Eine Klothoide ist eine parametrische Kurve mit Krümmungsradius $R = \frac{A^2}{l}$, einer positiven Konstante A und der Länge des Kurvenbogens l. Die Berechnung einer durch Klothoiden modellierten Strecke ist sehr einfach, da sich Kreisbögen und Geradenstücke durch Klothoiden mit konstanter Krümmung darstellen lassen. Durch Integration der Differentialgleichung

$$\kappa(l) = c_1 + c_2 l \tag{2.5}$$

$$\mathbf{y} = (x, y, \psi)^\top \tag{2.6}$$

$$\dot{\mathbf{y}}(l) = (\cos\psi, \sin\psi, \kappa(l))^\top \tag{2.7}$$

$$\mathbf{y}(l) = \int_0^l \dot{\mathbf{y}}(\tilde{l})\mathrm{d}\tilde{l} \tag{2.8}$$

lassen sich – nach Vorgabe eines Krümmungsprofils $\kappa(l)$ über die Streckenlänge l – die euklidischen Koordinaten zu jeder Streckenlänge berechnen. Nachteilig an dieser Art der Modellierung ist, dass Abstandsberechnungen nur numerisch zu bewerkstelligen sind.

Eine weitere Art der parametrischen Modellierung sind kubische Splines. Das sind Kurven, bei denen die x- und y-Komponenten abschnittweise als Polynome dritten Grades vorliegen. Der Laufparameter bezeichnet die Weglänge entlang der Achse der Straße. Vorteilhaft bei dieser Modellierung ist, dass mit wenigen Kontrollpunkten große Streckenabschnitte approximiert werden können. Die Polynome können dabei, im Gegensatz zu den Klothoiden, analytisch ausge-

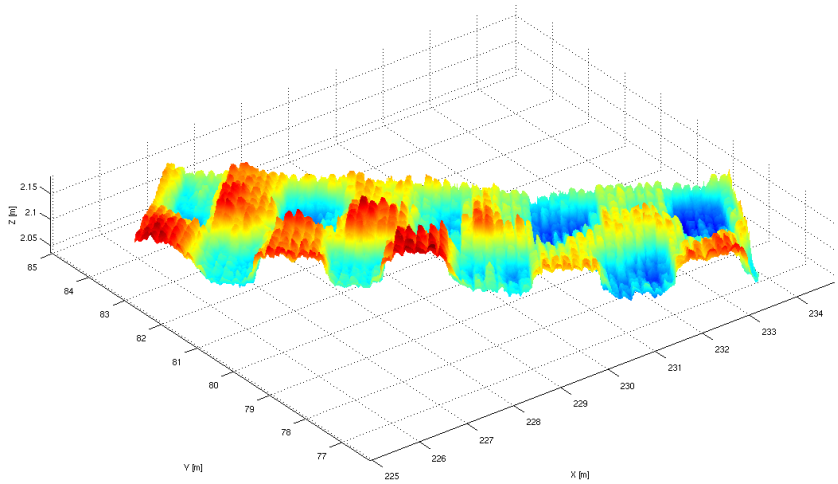


Abbildung 2.8: Modellierung der Oberfläche einer Reifenprüfstrecke. Der sog. *Belgian Block* ist eine bestimmte Art von Kopfsteinpflasterung, die auch auf Reifenprüfstrecken zum Einsatz kommt. Das Relief lässt sich im gezeigten Oberflächenverlauf erkennen. In der Abbildung sind die Punkte aus dem Straßenkoordinatensystem in ein kartesisches Koordinatensystem transformiert. (Bildquelle: VIRES GmbH)

wertet werden. Bezüglich der Abstandsberechnungen haben sie die gleichen Nachteile wie die Klothoiden.

Die bisherigen Ansätze beschreiben die Geometrie in der Fahrstreifen- oder Straßenebene. Mit OpenCRG existiert ein Ansatz, die *Oberfläche* der Straße sehr genau zu beschreiben [ASA20]. Ein Curved Regular Grid (CRG) ist ein regelmäßiges Gitter, welches jedoch nicht in kartesischen Koordinaten, sondern entlang einer Referenzlinie aufgespannt ist. Anwendung findet das Modell in der Fahrwerks- und Reifensimulation. Abb. 2.8 zeigt das Modell eines Abschnitts einer Reifenprüfstrecke.

2.3.2 Umfeldmodellierung mit Karten

Bei der *Grand Cooperative Driving Challenge* 2011 repräsentierte das Siegerteam *AnnieWAY* den Fahrstreifen durch einen Polygonzug mit einem Punktabstand von 0,5 Metern [GLM+12]. Durch die Nutzung eines *k-d-Baums* wurde eine schnelle örtliche Kartenabfrage ermöglicht.

k-d-Bäume (und andere Baumstrukturen wie *R*-Bäume und *Quad-Trees*) adressieren das Problem, dass in einer Karte mit räumlich etwa gleichverteilten Objekten die Datenmenge quadratisch mit der Ausdehnung der Karte in beide Richtungen wächst. Gleichzeitig soll die räumliche Abfrage idealerweise von konstanter Zeit sein. Information zu einem Kartenausschnitt soll also in immer der gleichen Zeit, unabhängig von der Datenmenge, zur Verfügung stehen. Dies ist nicht möglich, doch geeignete Baumstrukturen erreichen etwa logarithmisches Zeitverhalten bei der Abfrage. Die Konstruktion von *k-d*-Bäumen ist in 2.9 dargestellt.

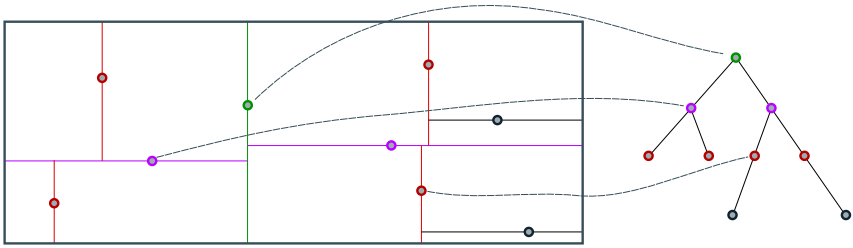


Abbildung 2.9: Konstruktion von *k-d*-Bäumen, hier für $k = 2$, also zwei Dimensionen. In diesem Beispiel werden die Punktmengen abwechselnd in beiden Dimensionen aufgeteilt. Bei der Aufteilung wird ein Pivotpunkt so ausgewählt, dass zu beiden Seiten in der jeweiligen Dimension gleich viele Punkte vorhanden sind. Im gezeigten Fall ist dies der grüne Punkt. Anschließend wird in der anderen Dimension geteilt, hier sind die beiden Pivotpunkte lila dargestellt. Die Prozedur wiederholt sich rekursiv für alle Punkte. Rechts ist der resultierende Baum dargestellt.

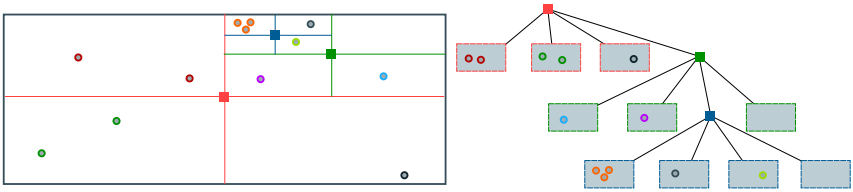


Abbildung 2.10: Konstruktion von *Quad-Trees*. Einem *Quad* wird eine bestimmte Kapazität von Punkten zugewiesen, in diesem Beispiel sind es vier Punkte. Bei Überschreitung der Kapazität wird das Quad in vier weitere Quadranten unterteilt und die enthaltenen Punkte werden den neuen Quadranten zugewiesen.

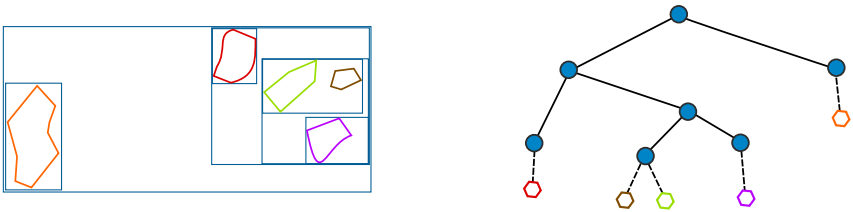


Abbildung 2.11: Die hier dargestellten *R*-Bäume (R für *rectangle*) übertragen die Konzepte der *Quad-Trees* und *k-d*-Bäume auf sich flächig ausbreitende Strukturen wie Linien und Polygone. Zu sehen sind Polygone, welche in eine Hierarchie von Rechtecken eingebettet sind.

Für die *Urban Challenge* [BIS09] der Defense Advanced Research Projects Agency (DARPA) wurde das Format Route Network Definition File (RNDF) eingeführt. Das RNDF teilt Straßen in Segmente ein, welche aus einem oder mehreren Fahrstreifen bestehen. Die Fahrstreifen sind durch Wegpunkte angenähert sowie mit einer Breitenangabe versehen. Zusätzlich werden *Zonen* durch Polygone dargestellt. Zonen modellieren einen befahrbaren Bereich, der nicht direkt zu einem Fahrstreifen gehört, wie etwa ein Parkplatz. Die Repräsentation im RNDF ist recht vollständig was die semantische Verknüpfung der Kartenbestandteile angeht, aber eher einfach, was die geometrische Darstellung der Fahrstreifen betrifft.

Betaille *et al.* nutzen *extended maps*, welche die Welt geometrisch durch Klothoiden beschreiben und semantisch durch Verbindungen zwischen diesen [BTM10]. Hasberg und Hensel nutzen einen probabilistischen Spline-Ansatz um im Kontext des Schienenverkehrs ein gleichzeitiges Kartieren und Lokalisieren zu ermöglichen [HHS12].

Das OpenStreetMap (OSM)-Projekt [RT10] modelliert die Welt aus den drei Primitiven *node* (Knoten, Stützpunkte), *way* (Weg, hier: Polygonzug) und *relation*, einer Inbezugsetzung dieser Primitive. Ähnlich wie im RNDF wird die Semantik sehr exakt modelliert, während die Geometrien der Fahrstreifen nachrangig sind. So werden Punkte üblicherweise ohne Höhe abgespeichert, Straßen werden durch einfache Linien und nicht durch flächige Objekte repräsentiert. Brubaker *et al.* nutzen die Daten aus OSM, um eine globale Lokalisierung rein auf Basis von Eigenbewegungsdaten zu erreichen [BGU13]. Dabei nutzen sie die sehr exakten semantischen Kartendaten und kommen mit einer vergleichsweise groben geometrischen Repräsentation aus.

Ein sehr umfangreicher Ansatz kommt vom OpenDRIVE-Konsortium, welches aus Vertretern der Automobil- und Zulieferindustrie besteht. Bei OpenDRIVE werden Verkehrsnetze semantisch und geometrisch beschrieben [ASA21]. Dabei steht der Begriff der *Straße* im Mittelpunkt. Ihr Verlauf wird durch eine *Referenzlinie* bestimmt, welche sich aus Geradenstücken (keine Krümmung), Klothoiden (konstante Krümmungsänderung), Kreisbögen (konstante Krümmung) und kubischen Polynomen zusammensetzen kann. Entlang dieser Referenzlinien werden nun Fahrstreifen modelliert, welche von der Referenzlinie ausgehend in den positiven und negativen Bereich durchnummeriert werden. Diese Fahrstreifen können mit Eigenschaften wie Breite und Verschiebung gegenüber der Refe-

renzlinie beschrieben werden. Außerdem ist es möglich, das laterale Profil einer Straße zu beschreiben, etwa eine einseitige Überhöhung in einer Kurve oder ein beidseitiges Abfallen zu Zwecken der Entwässerung.

3 Kartengetriebene Verhaltensentscheidung

Im vorangegangenen Kapitel wurden die Grundlagen zur Umfeldrepräsentation in Karten sowie zur modularen Verhaltensentscheidung gelegt. In diesem Kapitel wird auf Basis dieser Grundlagen eine Kartenrepräsentation vorgeschlagen sowie eine auf dieser basierenden Verhaltensentscheidung entwickelt.

In Kapitel 2.1.6 auf Seite 11 wurde die Trajektorienplanung vorgestellt, wie sie bei der BBF zum Einsatz kam. Diese Planung war in der Lage, den durch zwei Ränder beschriebenen Freiraum vollständig zur Planung zu nutzen. Durch ein Verhaltensentscheidungssystem wurden Verkehrsregeln beachtet; ein ordnungsgemäßes Fahren wurde ermöglicht.

In diesem Abschnitt wird das Modul zur Verhaltensgenerierung beschrieben. Dieses Modul bezieht die benötigte Information aus drei Quellen, diese sind die *Karte*, also statische Umgebungsinformation, *dynamische Objekte*, also alles, was nicht in der Karte eingezeichnet ist, und aus *Regeln*, welche auf das sich aus Karte und dynamischen Daten ergebende Lagebild angewandt werden.

Dieses Kapitel basiert in Teilen auf der Veröffentlichung [BZS14], in welcher *Lanelets* erstmalig beschrieben werden. Die wesentlichen Beiträge sind die Verhaltensentscheidung für die BBF und die Beschreibung von Lanelets, der für die BBF verwendeten Kartendarstellung.

3.1 Lanelets als universelle Kartendarstellung

Die hier vorgestellte und im Rahmen dieser Arbeit entwickelte Kartendarstellung nutzt als zentrales Element *Fahrstreifensegmente*, welche *Lanelets* genannt werden sollen.

3.1.1 Geometrische Modellierung

Die geometrische Modellierung ist ein Kernelement der Karte. Die Position der Strukturen wird auf zwei Dimensionen reduziert.

Objekte in der Welt, sichtbar und unsichtbar, werden durch Punkte, geschlossene Polygone und Polygonzüge approximiert. Ein *Polygonzug* ist eine Menge von Punkten, welche durch lineare Interpolation verbunden werden. Der Polygonzug hat zwei wesentliche Eigenschaften. Erstens wird eine beliebig genaue Approximation beliebiger Fahrstreifenkonturen ermöglicht, zweitens existieren sehr effiziente Algorithmen zur Abstandsberechnung eines Punkts zu einem Polygonzug. Es ist klar, dass diese Eigenschaften theoretische Nachteile haben. Wird ein Kreis (x, y, r) durch ein Polygon approximiert, dann werden erheblich mehr Parameter benötigt, um eine gewisse Genauigkeit zu erreichen. Die Abstandsberechnung eines Punkts zu einem Kreis ist direkt möglich, beim Polygonzug muss unter Umständen über mehrere Abschnitte iteriert werden. Diese Nachteile spielen in der Praxis jedoch eine untergeordnete Rolle.

3.1.2 Semantische Modellierung

Die semantische Modellierung setzt geometrische Primitive in Bezug zueinander. Aus einzelnen Primitiven werden *Objekte*, welche wiederum untereinander in Beziehung stehen.

Modellierung des befahrbaren Bereichs

Lanelets sind die kleinste Einheit, in der Fahrstreifenstücke modelliert werden. Ein Lanelet besteht mindestens aus einem linken und einem rechten Rand. Der Rand besteht aus einem oder mehreren Randstücken, welche jeweils durch Polygonzüge beschrieben sind. Durch die Rollen der Randstücke (*links* oder *rechts*) ist gleichzeitig festgelegt, in welcher Richtung das Fahrstreifensegment zu durchfahren ist.

Bereiche, die in beide Richtungen befahrbar sind, werden durch zwei Lanelets modelliert. Diese referenzieren die zum Bereich gehörende Geometrie in jeweils vertauschten Rollen *rechts* und *links*.

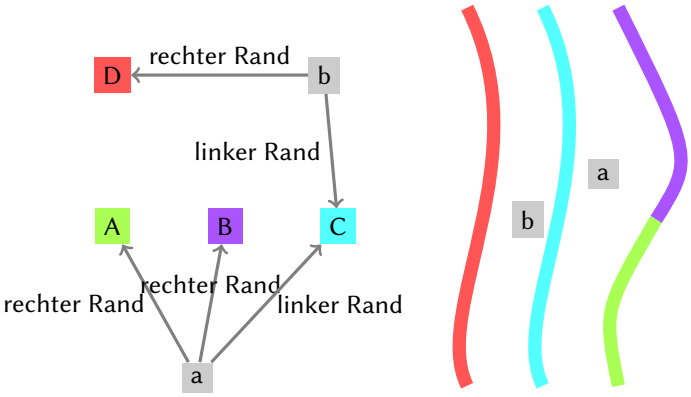
```
def sort_parts(parts):
    result = [parts.pop(0)]
    while parts:
        probe = parts.pop(0)
        if not (insert(probe, result) or \
        insert(probe[::-1],result)):
            parts.append(probe)
    return result
```

(a) `sort_parts`-Routine zum Sortieren von Randstücken.

```
def insert(part, parts):
    if part[1] == parts[0][0]:
        parts.insert(0, part)
        return True
    elif part[0] == parts[-1][1]:
        parts.append(part)
        return True
    return False
```

(b) Hilfsroutine zum Anfügen eines Stückes an die Ergebnisliste.

Abbildung 3.1: Algorithmus zum Sortieren der Randstücke.



(a) Graphendarstellung (b) Geometrische Darstellung

Abbildung 3.2: Geometrische und semantische Darstellung zweier Lanelets *a*, *b*. Die Beziehung zwischen beiden Teilen der Abbildung ist über die Farbe verdeutlicht.

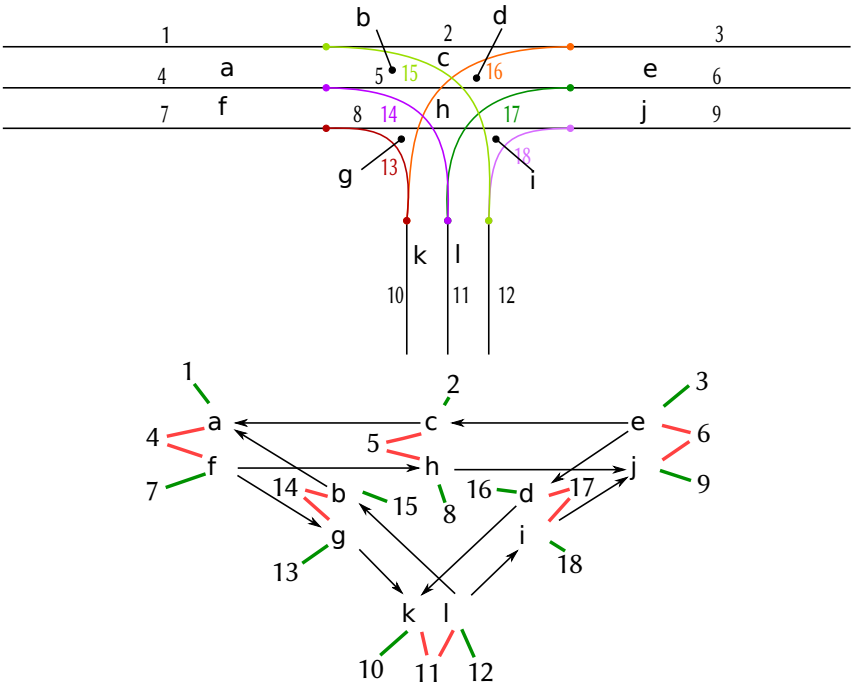


Abbildung 3.3: Weiteres, komplexeres Beispiel für die Darstellung der Topologie und Geometrie einer Einmündung. Arabische Zahlen bezeichnen die Ränder. Lateinische Kleinbuchstaben bezeichnen die Lanelets. Im Unteren Teil der Abbildung ist der Zusammenhang zwischen den Lanelets untereinander dargestellt sowie die Beziehung der Lanelets zu den Rändern. Die Farbe grün steht für die Beziehung *rechter Rand*, die Farbe rot steht für die Beziehung *linker Rand*.

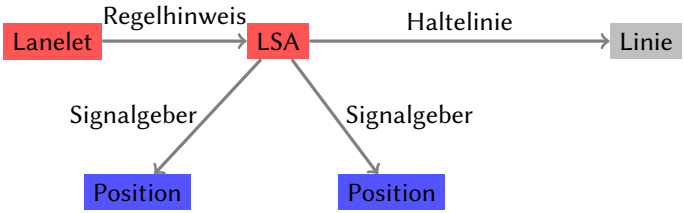


Abbildung 3.4: Graphendarstellung einer Lichtsignalanlage.

Modellierung von Regelhinweisen

Regelhinweise bilden ab, wie sich Fahrzeuge in verschiedenen Situationen zu verhalten haben. Diese Hinweise werden einem oder mehreren Lanelets zugeordnet und beinhalten die notwendige Information, die ein Fahrzeug benötigt, um sich ordnungsgemäß zu verhalten.

Modellierung von Lichtsignalanlagen Eine Lichtsignalanlage (LSA) gibt bestimmte Fahrstreifenstücke durch ein Lichtsignal frei. Ist das Fahrstreifenstück nicht freigegeben, muss an einer Haltelinie gewartet werden. Das Lichtsignal wird dabei synchron über einen oder mehrere Signalgeber kommuniziert. Diese Struktur wird in der Umgebungsrepräsentation wie in Abb. 3.4 dargestellt.

Modellierung von Vorfahrtsregeln Es werden zwei Arten von Vorfahrt unterschieden:

1. Situationen, bei denen sich die Fahrwege kreuzen (Typ *yield*) und
2. Situationen, nach deren Auflösung Vorrangberechtigter und Vorranggewährender den gleichen Fahrweg verfolgen (Typ *merge*).

Ist für eine bestimmte Kreuzungsbefahrung beides vorgesehen, so sind zwei getrennte Regelhinweise abzulegen. Ein Beispiel hierfür ist das Linksabbiegen im Gegenverkehr bei gleichzeitigem Beachten der Rechts-vor-links-Regel, oder eine abbiegende Vorfahrt wie in Abb. 3.5, anhand welcher beide Situationen erläutert werden. Der Typ *merge* bildet die Situation ab, in welcher beide Fahrzeuge

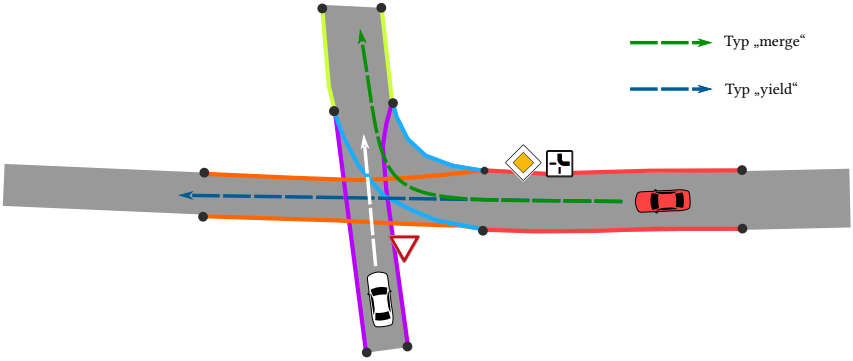


Abbildung 3.5: Abbiegende Vorfahrtstraße, rotes Fahrzeug ist vorrangberechtigt. Je nach Fahrweg des roten Fahrzeugs sind die Typen *merge* oder *yield* erforderlich.

nach Abschluss des Manövers den gleichen Fahrweg verfolgen. Das ist in der Abbildung z. B. dann der Fall, wenn beide Fahrzeuge Lanelet ○ befahren. Fährt das rote Fahrzeug geradeaus, so kreuzen sich die Fahrwege und es entsteht ein Manöver des Typs *yield*.

Stoppschilder werden modelliert, indem einer Vorfahrtssituation eine Mindestanhaltedauer zugeordnet wird. Außerdem kann durch Zuordnung eines Ways in der Rolle *stop line* eine Haltelinie zugeordnet werden.

3.1.3 Einbettung der Lanelet-Karten in OpenStreetMap

OpenStreetMap als Projekt wurde bereits in Abschnitt 2.3 auf Seite 19 vorgestellt. An dieser Stelle soll noch einmal darauf verwiesen werden, dass im Selbstverständnis des Projekts unterschieden wird zwischen der *Datenbasis*, also den Geodaten, und der *Anwendung*, also dem Einsatz der Daten in Navigationsanwendungen oder der visuellen Darstellung von Karten. Eine Rolle in der Mitte fällt dem Datenaustauschformat zu, genauer gesagt der Datenmodellierung. Die Welt wird in OpenStreetMap mit Hilfe dreier Primitive modelliert:

node zur Darstellung eines Punkts,

way zur Darstellung geordneter Listen von *nodes* und

relation zur Darstellung eines Bezugs zwischen den drei Primitiven.

Die englischen Begriffe sollen hier naheliegend mit *Knoten*, *Weg* und *Relation* übersetzt werden, auch wenn sich hinter *Weg* im Kontext dieser Arbeit etwas anderes vermuten lässt.

Jedes Primitivum kann durch Schlüssel-Wert-Paare mit weiterer Information angereichert werden. Außerdem fällt jedem Mitglied einer Relation eine bestimmte *Rolle* zu.

Abgelegt werden die Daten in zweierlei Form: zur effizienten Speicherung und Abfrage werden relationale Datenbanken mit einer Erweiterung für Geodaten eingesetzt [OH15, Mom01], zum unkomplizierten Austausch existiert ein XML-Format.

Die im letzten Abschnitt vorgeschlagene Kartendarstellung lässt sich in die OSM-Primitive überführen. Damit werden die Editierwerkzeuge des OSM-Projekts nutzbar sowie alle Möglichkeiten zur Datenhaltung, welche im Projekt angeboten werden. Die Lanelet-Karten können allerdings *nicht* zur Datenbasis von OSM beitragen, da die Ziele der Modellierung und die Art der Darstellung grundsätzlich abweichen.

3.1.4 Die Karte für die Bertha-Benz-Fahrt

Im Rahmen der Bertha-Benz-Fahrt wurden ca. 100 Kilometer Strecke kartiert. Entlang der Strecke wurden dabei alle Fahrstreifen in die Karte aufgenommen. Die Karte

- bestand aus 78.000 Knoten,
- 8.900 Lanelets und
- war dreieinhalb Megabyte groß (ca. 35 Kilobyte pro Kilometer).

Die Modellierung der Fahrstreifen war ausreichend präzise, um die Fahrt rein nach Karte (und keiner weiteren Fahrstreifenschätzung) durchführen zu können. Bezogen auf das deutsche Straßennetz mit einer Gesamtlänge von ca. 650.000 Kilometern entspräche das einer Karte von 20 Gigabyte, also einer durchaus

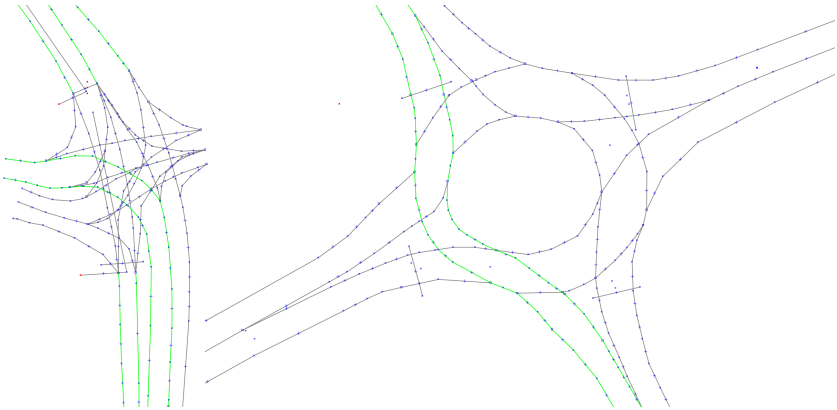


Abbildung 3.6: Kartierung einer Kreuzung (links) und eines Kreisverkehrs (rechts).

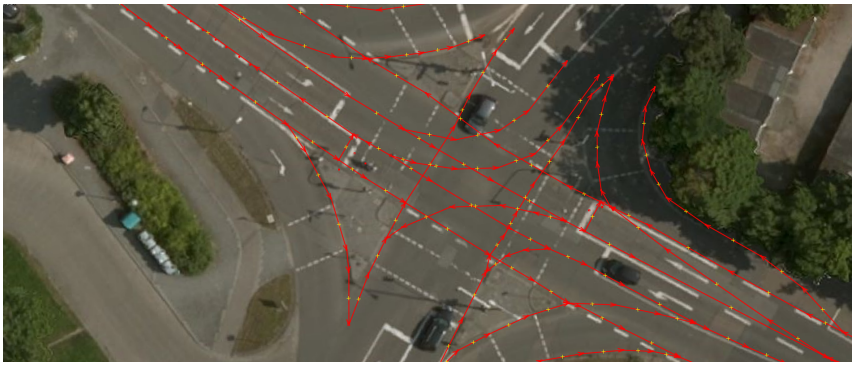


Abbildung 3.7: Lanelet-Repräsentation einer komplexen Kreuzung mit mehreren Fahrstreifen und Fahrstreifenübergängen. Die Kartenerstellung wurde durch präzise referenzierte Luftaufnahmen unterstützt.

handhabbaren Größe. Durch die relativ geringe Datenmenge pro Strecke ist auch ein Bezug nach Bedarf denkbar.

Eine Reimplementierung der Kartenkomponente der Bertha-Benz-Fahrt wurde unter den Bedingungen der *GNU Public License* unter dem Namen *libLanelet* veröffentlicht[1].

Effiziente Online-Abfrage der Karteninhalte

In Abschnitt 2.3.1 wurden räumliche Indizierungsmethoden für Punktdaten vorgestellt, die *Octrees* und *kd-Bäume*. Ein entsprechendes Indizierungsverfahren für ausgedehnte (in der Fläche, räumlich oder in höheren Dimensionen) Daten sind die *R-Bäume* (engl.: *R-Tree*) [Gut84, BKSS90]. In solchen Bäumen repräsentieren Knoten jeweils Intervalle in mehreren Dimensionen, für die Ebene also Rechtecke. Diese Rechtecke können entweder weitere Kindknoten aufweisen oder direkt mit Nutzdaten versehen sein. R-Bäume können Bereichsanfragen sehr schnell beantworten, sind aber in der Konstruktion gegenüber z. B. Quad-Trees aufwändiger. Da die Karteninhalte nur selten aktualisiert werden, fällt diese Eigenschaft hier nicht ins Gewicht.

Vergleich der Lanelet-Karte mit OpenDRIVE

OpenDRIVE ist ein weit verbreitetes Format für die Modellierung von Straßen für Simulationszwecke oder allgemein Zwecke des automatischen Fahrens. Wo liegen die Vorteile und Unterschiede der beiden Formate?

Zunächst haben beide Kartenformate das XML-Format zur Grundlage, sind also ohne proprietäre Werkzeuge zugänglich. OpenDRIVE hat zusätzlich noch die Eigenschaft, dass ein explizites Längskoordinatensystem für Straßen angegeben ist, während diese Referenz bei Lanelet-Karten nur implizit und nicht Fahrstreifen übergreifend vorhanden ist. In der Modellierung hingegen spielen Lanelets ihre Stärken aus: da Fahrstreifenbegrenzungen explizit modelliert werden, müssen sie nicht über einen Referenzlinien- und Breitenverlauf implizit dargestellt werden. Außerdem sind beliebig komplexe Verläufe möglich.

Ein weiterer Vorteil von Lanelet-Karten ist die Werkzeugverfügbarkeit. Zur Bearbeitung der Karten kann zwischen mehreren Editoren gewählt werden,

[1] `https://github.com/phbender/liblanelet`, abgerufen am 20. April 2021.

die größtenteils quelloffen verfügbar sind. Für OpenDRIVE ist nur ein Editor verfügbar[2] und es müssen Lizenzen für jeden Arbeitsplatz bezogen werden. Lanelet-Karten profitieren außerdem von der Software-Infrastruktur des OpenStreetMap-Projekts: die Daten können in GIS-Datenbanken abgelegt und direkt mit den Editoren synchronisiert werden. So ist es möglich, mit einer sehr großen Datenbasis zu arbeiten. Außerdem wird eine Versionierung unterstützt, was die parallele Modifikation durch mehrere Kartenbearbeiter ermöglicht.

Inzwischen gibt es Arbeiten, die den Ersatz von OpenDRIVE durch Lanelets beschreiben [AUK18].

3.2 Verhaltensentscheidung für Bertha

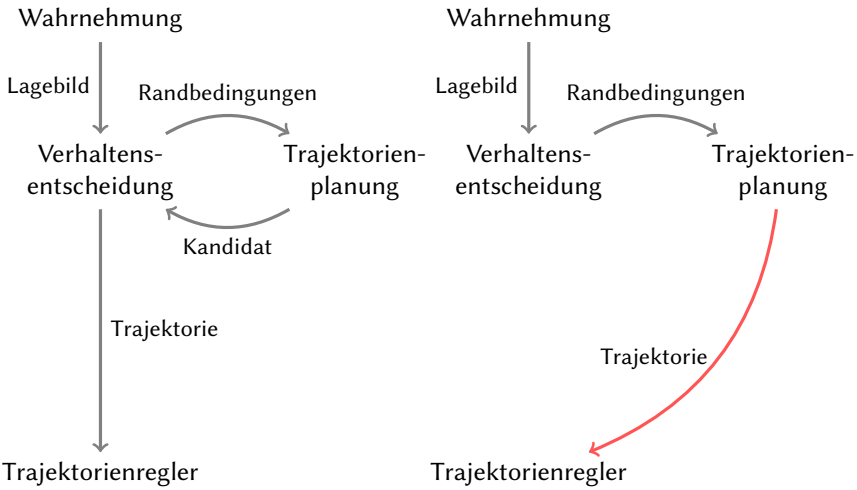
In diesem Abschnitt wird das Verhaltensentscheidungsmodul basierend auf der Bertha-Benz-Fahrt beschrieben.

Die Verhaltensentscheidung bildet ein System gemeinsam mit der Trajektorienplanung. Während die Trajektorienplanung ein effizientes, aber bezüglich der Umwelt „blindes“ System ist, kann die Verhaltensentscheidung in größeren Zusammenhängen analysieren und entscheiden, selbst aber keine optimale Planung durchführen. Beide Systeme stehen idealerweise in enger Beziehung (siehe Abb. 3.8): die Verhaltensentscheidung bereitet das Lagebild der Perzeption zu Randbedingungen für ein Optimierungsproblem auf. Gibt es mehrere Möglichkeiten dieser Aufbereitung, so präsentiert die Verhaltensentscheidung all diese Möglichkeiten und bekommt von der Trajektorienplanung je einen Trajektorienkandidaten zurück. Aus diesen Kandidaten wählt sie eine Trajektorie zur Weitergabe an den Regler aus. Ein Teil dieser Überlegungen greift Kapitel 4 auf Seite 49 vor: dort wird die Identifikation möglicher Varianten vorgestellt sowie deren Umsetzung durch Nebenbedingungen.

3.2.1 Architektur

Im Straßenverkehr lassen sich Regeln und Zustände identifizieren, sodass sich die Umsetzung mit Hilfe des Statechart-Formalismus anbietet. Ein kleines Detail spricht jedoch gegen diese erste Idee: In Zustandsautomaten – was Statecharts ja

[2] *RoadDesigner* der Firma *VIRES*



(a) Idealisierte Einbindung der Verhaltensentscheidung. Sie bekommt durch Variation der Randbedingungen mehrere Kandidaten geliefert, aus denen sie auswählen kann.

(b) Praktische Umsetzung für die Bertha-Benz-Fahrt. Die Verhaltensentscheidung musste ohne Rückführung der Trajektorie eine möglichst optimale Variation der Randbedingungen liefern.

Abbildung 3.8

im Kern sind – ist die *Anzahl* der Zustände nicht variabel. Im Straßenverkehr ist das hingegen nicht so: wird der Durchlauf eines bestimmten Teils des Automaten als Interaktionsprotokoll angesehen zwischen dem Ego-Fahrzeug und anderen Akteuren, wie etwa einer LSA oder einem anderen Fahrzeug, so wird schnell klar, dass mehrere Instanzen dieses Protokolls zeitgleich aktiv und notwendig sein können. Eine solche Instanz wird im Folgenden *Ereignis* genannt. Um mehrere Ereignisse zeitgleich behandeln zu können, wird das System wie folgt gestaltet.

Ausgangspunkt für die Verhaltensgenerierung ist eine *Mission*. In dieser sind der Reihe nach zu erreichende Orte enthalten. Den Missionspunkten werden Lanelets zugeordnet, anhand derer ein optimales Routing durchgeführt wird. Das Ergebnis ist eine Liste aufeinanderfolgender Lanelets oder – allgemeiner – ein Pfad durch den Kartengraphen.

Ausgehend vom aktuell befahrenen Lanelet hält die Verhaltensgenerierung einen gewissen Horizont vor, also einen Ausschnitt aus dem Pfad. Dieser räumliche Horizont ist so groß zu wählen, dass er jederzeit den Aktionsradius des Fahrzeugs innerhalb des zeitlichen Planungshorizontes des Planungsmoduls abdeckt.

Innerhalb eines solchen Ausschnittes werden die Ereignisse gesammelt (Abb. 3.9). Diese sind eindeutig identifizierbar, sodass überprüft werden kann, ob ein Ereignis bereits bearbeitet wird oder nicht. Mit dem Auftreten eines Ereignisses wird das zur Ereignisbehandlung erforderliche Protokoll aktiv. Dieses Protokoll wird durch Statecharts modelliert.

Auf diese Weise aktivierte Ereignisse werden in eine Warteschlange eingereiht. Ist das zu einem Ereignis gehörende Protokoll abgearbeitet, wird das Ereignis der Warteschlange entnommen und verworfen. Die Ereignisse in der Warteschlange werden zyklisch mit dem aktuellen Lagebild versorgt und geben, abhängig vom Zustand des jeweiligen Protokolls, Anweisungen zurück. Die Anweisungen werden gesammelt, aus Ihnen werden die Randbedingungen für das Planungsproblem generiert.

3.2.2 Protokolle zur Ereignisbehandlung

Zu behandelnde Ereignisse werden hauptsächlich durch die Karte hervorgerufen und haben einen örtlichen Bezug. Da die Behandlung von Ereignissen mit örtlichem Bezug ein sehr häufiger Fall ist, soll zuerst dieser Teil geschildert

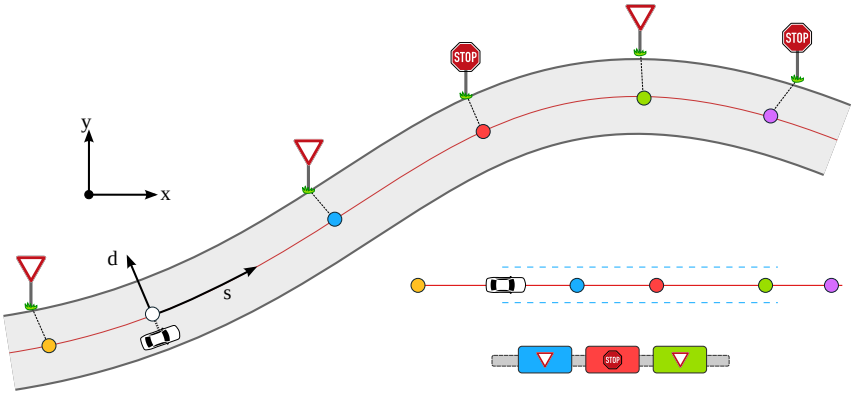


Abbildung 3.9: Behandlung relevanter Ereignisse. Die den Ereignissen durch die Karte zugeordneten Positionen werden in Fahrzeugkoordinaten umgewandelt. Für einen bestimmten Horizont werden die Ereignisse in die Liste aktiver Ereignisse übernommen, die Behandlung startet.

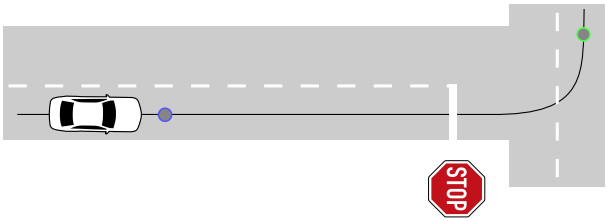


Abbildung 3.10: Startpunkt (blau) und Endpunkt (grün) einer Ereignisbehandlung. Diese Punkte werden aus der Karte auf den Pfad projiziert, beim Überfahren werden jeweils Ereignisse ausgelöst.

werden. In den Protokollen für weitere Ereignisse wird dieser Teil dann zwar der Vollständigkeit halber gezeigt, aber nicht weiter besprochen.

Ein ortsbezogenes Protokoll beginnt mit dem Überfahren eines bestimmten Punkts, dem *Startpunkt* (Abb. 3.10). Diese Überfahrt wird bestimmt, indem die Straßenlängskoordinate der Fahrzeugposition mit der des Ereignisses verglichen wird. Analog zum Startpunkt gibt es den *Endpunkt*, welcher das Protokoll beendet. Der Hintergrund ist, dass das ein Ereignisprotokoll zwar die Rückwirkung seiner Abarbeitung durch Messwerte mitbekommen kann, jedoch nicht über die Planung informiert wird. Am Beispiel einer Lichtsignalanlage bedeutet dies, dass nach dem Übergang vom *Warten bei Rot*-Zustand in den *Grün*-Zustand nicht davon ausgegangen werden kann, dass die Anlage passiert wird, sondern über Beobachtung der Position geprüft werden muss, ob der Endpunkt passiert wurde.

Lichtsignalanlage

Die LSA ist der erste Vertreter eines ortsbezogenen Protokolls. Eine LSA kennt aufgrund der Modellierung in der Karte die Identifikationsnummern der zugehörigen Lichtsignalgeber. Die Lichtsignalgeber werden in der Perzeptionsschicht detektiert und mit einem der Zustände *grün*, *nicht grün* oder *unbekannt* bereitgestellt. Die Signalzustände, welche *gelb* beinhalten, werden nur ihrer Bedeutung nach berücksichtigt: ein strategisches Überfahren trotz der Möglichkeit zum Anhalten findet nicht statt. Der Zustand *unbekannt* wird eingeführt, um mit

fehlenden Beobachtungen umgehen zu können, also Beobachtungen, bei denen der Lichtsignalgeber z. B. wegen Verdeckung durch andere Objekte nicht im Bild gefunden werden konnte. In Abb. 3.13 ist das Protokoll dargestellt. Die Ereignisse zum Zustandsübergang sind wie folgt zu verstehen: *green* bedeutet, dass mindestens eines der Lichtsignale als *grün* und innerhalb eines bestimmten Zeitraums *keines* als *nicht grün* detektiert wurde (etwa: innerhalb der letzten halben Sekunde, dieser Wert muss auf die Perzeption angepasst sein). Der Übergang *red* entspricht dem Gegenteil. Durch diese Vorgehensweise entsteht eine Robustheit durch zeitliche Redundanz (mehrere Beobachtungen des gleichen Signalgebers) sowie eine Redundanz durch mehrere Lichtsignalgeber selbst, soweit vorhanden. Die Ereignisse *passed* · werden ausgelöst, wenn das Fahrzeug den Aktivierungspunkt (*tp*) oder die Haltelinie (*sl*) überfahren hat.

Vorfahrtssituation

Bei der Vorfahrtssituation werden zwei Varianten unterschieden. Die Unterscheidung erfolgt anhand der Fahrwege der Beteiligten. Die Fahrwege können sich *kreuzen* (Typ *yield*) oder sich *vereinen* (Typ *merge*), das wurde in Abschnitt 3.1.2 auf Seite 32 bereits beschrieben.

Gemein ist beiden Varianten, dass die Behandlung – wie auch die der LSA – über bestimmte Punkte getrieben wird, siehe dazu Abb. 3.13. Nach Überfahren des Auslösepunkts geht das Fahrzeug abhängig von der Attributierung in der Karte in den Zustand *Unclear* oder *Stop*. Im *Unclear*-Zustand wird das Fahrzeug wegen einer unzureichenden Informationslage aus der Perzeption ein Anhalten vorbereiten. Dieser Zustand wird verlassen, sobald die Fahrzeugsensorik in der Lage ist, die notwendige Information zur Behandlung der Situation zu erfassen. Im Beispielprotokoll geschieht das durch Unterschreiten eines individuell festgelegten Abstands zur Haltelinie. Im Falle *Stop* hält das Fahrzeug an der Haltelinie an, wartet für eine pro Ereignis einstellbare Zeit, und geht dann in den Zustand *Drive* über. Jetzt werden die Nebenbedingungen in Abhängigkeit der anderen Verkehrsteilnehmer generiert (dazu gleich mehr) und das Fahrzeug kann die Kreuzung befahren. Nach Überschreiten der Haltelinie ist die Ereignisbehandlung abgeschlossen.

Um die Situationen korrekt zu behandeln, ist der Fahrweg der anderen Verkehrsteilnehmer erforderlich. So lange dieser nicht eindeutig klar ist, werden

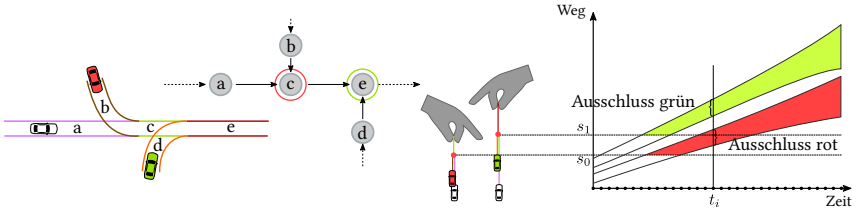


Abbildung 3.11: Entstehung des Weg-Zeit-Diagramms aus einer Verkehrssituation auf Basis der Lanelet-Karten. Das Ego-Fahrzeug (weiß) ordnet die anderen Fahrzeuge Lanelets zu und stellt Hypothesen für deren Trajektorien relativ zur eigenen Trajektorie auf. Dabei hilft die Anordnung der Lanelets in einem Graphen. Die Trajektorie des roten Fahrzeugs kann bei s_0 ein vom Ego-Fahrzeug benutztes Lanelet belegen, entsprechend wird die Trajektorie ab dieser Bogenlänge relevant. Analog gilt das für das grüne Fahrzeug. Das Ego-Fahrzeug hat also für diskrete Zeitpunkte (hier exemplarisch t_i) bestimmte Bereiche, die für die eigene Trajektorie ausgeschlossen sind.

für eine möglichst sichere Planung *alle möglichen* Fahrwege angenommen. In Abb. 3.5 bedeutet dies, dass das rote Fahrzeug sowohl das in orange eingezeichnete Lanelet (geradeaus fahren) als auch das in grün eingezeichnete Lanelet (links abbiegen) erreichen kann und damit beide Möglichkeiten vorgesehen werden müssen.

3.2.3 Einflussnahme auf die Planung: Nebenbedingungen

Bisher wurde gezeigt, wie Wahrnehmung und Karte verschiedene Zustände aktivieren und deaktivieren. Doch wie sieht der Einfluss auf die Fahraktion aus?

Die Nebenbedingungen werden aus dem Weg-Zeit-Diagramm abgeleitet. Dazu werden der Fahrweg des Ego-Fahrzeugs und die mutmaßlichen Fahrwege der anderen Verkehrsteilnehmer einer einheitlichen Wegkoordinate zugeordnet, siehe Abb. 3.11. Je nach Art der Vorfahrtssituation entstehen Ausschlüsse im Weg-Zeit-Diagramm, welche dann als Nebenbedingungen an die Trajektorienplanung

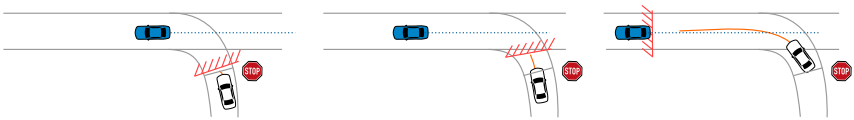
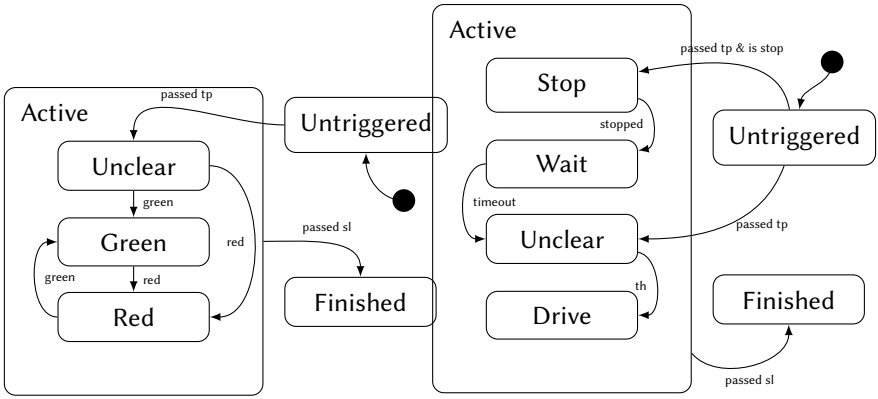


Abbildung 3.12: Die Zustände *Stop* (links), *Unclear* (Mitte) und *Drive* (rechts) gemäß des Protokolls 3.13b. Rot eingezeichnet sind jeweils die an die Planung kommunizierten Nebenbedingungen.

weitergegeben werden. In der Abbildung wird angenommen, dass die anderen Fahrzeuge jeweils Vorrang haben. Außerdem wurde für die Prädiktionen eine leichte positive und negative Beschleunigung angenommen, um ein möglichst konservatives und defensives Verhalten zu erreichen. Das erklärt, warum sich die farbig markierten Bereiche mit fortschreitender Zeit verbreitern.

Abb. 3.12 zeigt eine ähnliche Situation, hier aber mit den Zustandsübergängen im Fokus. Die Haltelinie im Bild habe eine für die Sensorik ungünstige Position. Dies wird kompensiert durch das Anlegen einer weiteren Linie, welcher sich das Fahrzeug nähert, nachdem an der Linie des Stoppschilds angehalten wurde. Nach Erreichen dieser zweiten, künstlich gesetzte Linie wird in den Zustand *Drive* gewechselt und die Nebenbedingungen werden aus den anderen Verkehrsteilnehmern generiert. Die Nebenbedingungen sind als Ungleichheitsnebenbedingungen im Straßenkoordinatensystem formuliert.

Die Verhaltensgenerierung hat keinen direkten Durchgriff auf die Planung. Sie erzeugt lediglich einen Satz an Nebenbedingungen, welchen die Planung folgen muss. Sind die Nebenbedingungen so gestaltet dass sie mehrere Varianten zulassen, so sind zwei Möglichkeiten denkbar: die Architektur von Bertha war so ausgelegt, dass die Verhaltensgenerierung die Nebenbedingungen ohne weitere Prüfung an das Planungmodul weitergeleitet hat. Dieses Planungsmodul hatte einen vorgeschalteten, heuristischen Entscheidungsprozess. Dieser ist jedoch ohne weiteres auch in der Verhaltensentscheidung selbst vorstellbar: die Verhaltensentscheidung präsentiert dem Planungsmodul jede mögliche Variation der Nebenbedingungen und trifft darauf basierend eine Entscheidung indem die Trajektorie an den Trajektorienregler weitergeleitet wird. Die Entscheidung für ein Manöver muss unter zwei Gesichtspunkten erfolgen: die geplante Tra-



(a) Protokoll zur Behandlung einer LSA, umgesetzt als Statechart-Diagramm. `tp` steht für den Aktivierungspunkt, `sl` für die Haltelinie.

(b) Protokoll zur Behandlung einer Vorfahrtssituation, umgesetzt als Statechart-Diagramm. `th` steht für die Unterschreitung eines Abstands, `stopped` und *timeout* werden ausgelöst wenn das Fahrzeug steht bzw. eine bestimmte Zeit gestanden hat.

Abbildung 3.13: Protokolle zur Ereignisbehandlung

jektorie muss aus kurzfristiger Sicht möglichst gut sein. Gleichzeitig soll die Entscheidung aber zeitlich möglichst konstant bleiben. Diese zeitliche Konstanz wird im Abschnitt 4.3.4 auf Seite 74 gesondert behandelt.

3.3 Implementierung in Bertha

Die hier vorgestellte Karte und die auf ihr basierende Verhaltensgenerierung waren unter den Kernkomponenten der Fahrzeuge *S 500 Intelligent Drive*, von denen zwei Exemplare zur Durchführung der Fahrt aufgebaut wurden. Die Rechnerarchitektur ist in Abb. 3.14 zu sehen.

Besonders an der Fahrt war die sehr abwechslungsreiche Umgebung. Alle üblichen Verkehrssituationen waren vertreten: Kreisverkehre, Vorfahrtssituationen, Fahrstreifenwechsel, Schnellstraßen, Stadtzentren (Mannheim, Heidelberg), kleinere Ortschaften mit schmalen Straßen – all diese Situation wurden durch das Verhaltensmodul im Zusammenspiel mit der Trajektorienplanung korrekt behandelt.

Die Implementierung des Planungsrechners bestand aus folgenden Komponenten:

- Einer Lanelet-Karte,
- einem Verhaltensgenerierungsmodul und
- einem effizienten, kontinuierlichen Planer.

Die Karte und das Verhaltensgenerierungsmodul waren gemäß dieses Kapitels gestaltet, das Planungsmodul ist in [Zie15] beschrieben.

Diese Softwarekomponenten konnten über eine Echtzeitdatenbank (RTDB) miteinander kommunizieren. Zwischen den Rechnern wurde über User Datagram Protocol (UDP) kommuniziert.

Die Verhaltensgenerierung lieferte dem Planer folgende Information:

- Fahrstreifenränder (Fahrkorridor)
- Soll-Geschwindigkeit

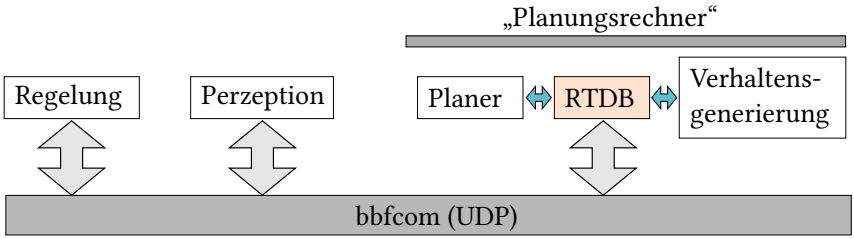


Abbildung 3.14: Architektur für *Bertha*. Über eine UDP-Verbindung wurden Zustands- und Planungsinformationen zwischen den verschiedenen Rechnern kommuniziert. Innerhalb des Planungsrechners konnten das Planungs- und Verhaltensmodul außerdem über eine sog. *Echtzeitdatenbank* (RTDB) kommunizieren. Diese Datenbank fungierte gleichzeitig als Broker zur *bbfcom*.

- Nebenbedingungen resultierend aus Position und Bewegung anderer Verkehrsteilnehmer, bezogen auf den eigenen Fahrkorridor (Einfädeln, Vorfahrt achten)

- Nebenbedingungen unabhängig von anderen Verkehrsteilnehmern (um LSA und Stoppschilder korrekt zu behandeln)

Der Planer war dabei mit Heuristiken versehen, um aus diesen Informationen eine möglichst gute Trajektorie ableiten zu können. In dieser Implementierung war die Verhaltensentscheidung nicht dafür ausgelegt, dem Planer mehrere Möglichkeiten zu eröffnen. Dies muss nicht so sein – die Verhaltensentscheidung kann auch so gestaltet werden, dass sie einen kontinuierlichen Planer dabei unterstützt, eine möglichst global optimale Lösung zu finden. Dies wird im folgenden Kapitel diskutiert.

Insgesamt war die Bertha-Benz-Fahrt ein äußerst erfolgreiches Projekt. Die Strecke wurde mit nur minimalen Eingriffen mehrfach automatisch befahren. Die Ergebnisse wurden breit für die Öffentlichkeit kommuniziert und wissenschaftlich in mehreren Veröffentlichungen aufbereitet. Das Format der Lanelet-Karten wird häufig zitiert und in Implementierungen berücksichtigt, so trägt z. B. D.

Abbildung 3.15: Links: Eine typische Ortsdurchfahrt entlang der *Bertha Benz Memorial Route*. Rechts: Das Fahrzeug *S 500 Intelligent Drive*.

Petrich eine Koordinatensystemtransformation bei [3] und das Forschungszentrum Informatik hat eine Neuimplementierung der Lanelet-Karten quelloffen zur Verfügung gestellt[4] [PPJ+18].

[3] `https://github.com/dspetrich/corridor`, abgerufen am 20. April 2021.

[4] `https://github.com/fzi-forschungszentrum-informatik/Lanelet2`, abgerufen am 20. April 2021.

4 Manöver: Kombinatorische Aspekte der kontinuierlichen Optimierung

In Kapitel 2.1 wurden Verfahren zur Trajektorienplanung vorgestellt. Diese wurden in zwei Kategorien eingeteilt: *lokale, kontinuierliche* Verfahren, welche unter Ausnutzung von Gradienten und Hesse-Matrizen recht schnell konvergieren, und *globale* Verfahren, welche, mit entsprechenden Heuristiken ausgestattet, das globale Minimum finden möchten. Beide Herangehensweisen haben ihre Berechtigung: die lokalen Verfahren stehen auf einer soliden mathematischen Basis, scheitern aber, wenn die Zielfunktion mehrere Nebenminima aufweist; die globalen Verfahren gleichen diesen Nachteil aus, der Preis ist jedoch eine hohe Laufzeit oder eine Reduktion der Optimierungsparameter.

In diesem Kapitel wird ein Verfahren vorgestellt, mit dem sich die Vorzüge beider Methoden kombinieren lassen. Vorarbeiten zu diesem Verfahren wurden in [BTS15] und [BS15] veröffentlicht. Es basiert auf zwei Annahmen:

1) Verschiedene lokale Minima sind Teil verschiedener Homotopien und

2a) ein lokales Verfahren wird diese Minima jeweils finden, wenn die Initialisierung Teil dieser Homotopie ist oder

2b) die Homotopie durch Nebenbedingungen bestimmt wird.

Folgendes Beispiel, illustriert in Abb. 4.1, soll die Annahmen veranschaulichen: Ein Ego-Fahrzeug, in der Abbildung weiß, folgt einem roten Fahrzeug. Auf dem benachbarten Fahrstreifen nähert sich ein grünes Fahrzeug aus der entgegengesetzten Richtung. Dem weißen Fahrzeug stehen zwei Handlungsoptionen offen: es kann das rote Fahrzeug *bevor* oder *nachdem* sich grün und rot begegnet

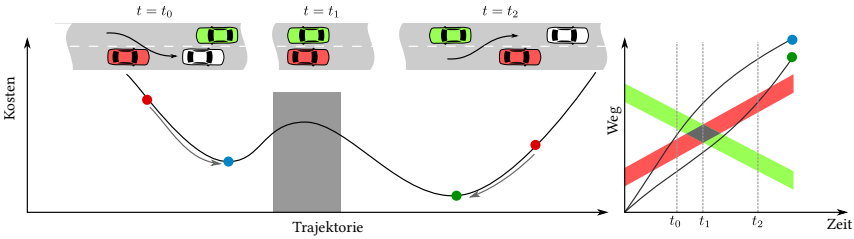


Abbildung 4.1: Veranschaulichung der Idee der Zerlegung des Lösungsraumes. Zwei Minima sind Teil verschiedener Homotopien, die Lösungsräume sind getrennt durch den Bereich, in welchem das rote und grüne Fahrzeug auf gleicher Höhe sind. Dieser Bereich ist im Weg-Zeit-Diagramm rechts als grauer Rhombus gekennzeichnet. Für beide Varianten ist eine Initialisierung (rote Punkte) und eine Lösung eingezeichnet. Die beiden Lösungen (grün und blau) finden sich im Weg-Zeit-Diagramm wieder.

sind überholen. Für beide Varianten wird es jeweils eine beste Trajektorie geben, aber beide Lösungen sind nicht kontinuierlich ineinander überführbar. Das ist links durch eine ausgeschlossene Lösungsmenge und rechts durch einen Rhombus gekennzeichnet. Werden Anfang und das jeweils rechte Ende der beiden Lösungstrajektorien festgehalten, so können die beiden nicht kontinuierlich ineinander überführt werden, ohne den ausgeschlossenen Rhombus zu schneiden. Die Initialisierung und Lösung hingegen *können* ineinander überführt werden, da sie Teil einer Homotopie sind. Ein lokales Lösungsverfahren kann angewendet werden, um die günstigste Lösung innerhalb der jeweiligen Homotopie zu finden. Beide Lösungskandidaten können anschließend verglichen werden, um die beste Variante zu identifizieren.

4.1 Homotopie

Die Homotopie ist ein Konzept aus der Topologie. Hatcher [Hat02] führt sie folgendermaßen ein:

Ein Pfad durch den Raum $\mathbb{X}$ sei definiert durch eine kontinuierliche Abbildung $f : [0,1] \to \mathbb{X}$. Eine Homotopie $F : [0,1] \times [0,1] \to \mathbb{X}$ ist eine Klasse von Abbildungen $f_t(s) = F(s,t)$, für die gilt:

1. Die Endpunkte $f_t(0) = \mathbf{x}_0$ und $f_t(1) = \mathbf{x}_1$ sind unabhängig von t.

2. F ist kontinuierlich.

Die erste Bedingung bedeutet, dass die Endpunkte der Pfade festgehalten werden – alle Pfade aus f_t haben gleiche Anfangs- ($f_t(0)$) und Endpunkte ($f_t(1)$). Die zweite Bedingung sagt aus, dass für jedes $t \in [0,1]$ ein gültiger Pfad existieren muss. Sind die Bedingungen erfüllt, können zwei Pfade f_t *homotop* genannt werden. Abb. 4.2 illustriert dies: $f(s)$ und $g(s)$ sind homotop ($f \simeq g$), $g(s)$ und $h(s)$ sind es nicht. Zwar lässt sich g in h überführen, jedoch nicht kontinuierlich, weil die Funktionen im Beispiel nicht für die komplette Ebene definiert sind, sondern es einen ausgeschlossenen Bereich gibt.

Da in der vorliegenden Arbeit nicht die Pfade, sondern die Trajektorien von besonderem Interesse sind, wird der Zielzustand nicht in jeder Dimension festgehalten, sondern nur in der zeitlichen Dimension. Was in der Pfadplanung ein fester Ziel*punkt* ist, ist in der Trajektorienplanung eine feste Ziel*ebene*. Der Ort kann variieren, die zeitliche Komponente des Endes der Trajektorie ist festgelegt.

Ein Planungsproblem, welches die bisherigen Annahmen und Erkenntnisse berücksichtigt, ließe sich folgendermaßen formulieren:

1. Identifikation der möglichen Homotopien,

2. Initialisierung einer Startlösung innerhalb der jeweiligen Klasse,

3. Optimierung unter der Bedingung, dass eine Homotopie zwischen der Initialisierung und der gefundenen Lösung existiert.

Das Kapitel folgt dieser Vorgehensweise, entsprechende Unterkapitel erklären die drei Schritte. Da sich zeigen wird, dass das Vorgehen in der Praxis einige Nachteile mit sich bringt, werden in einem weiteren Kapitel Vereinfachungen aufgezeigt, die sich aus dem Kontext *Fahren auf der Straße* ergeben. Den Abschluss

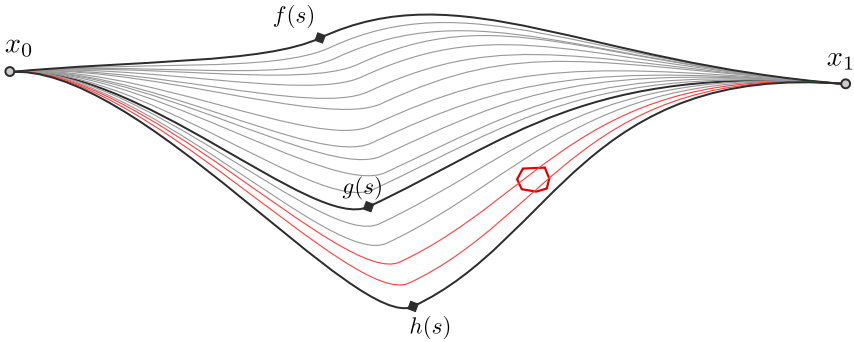


Abbildung 4.2: Homotopie für Pfade in der Ebene bezüglich x_0, x_1. Die Funktionen f, g, h sind nicht für die ganze Ebene definiert, es gibt einen rot markierten, ausgeschlossen Bereich.

bildet ein Abschnitt, welcher, basierend auf den genannten Vereinfachungen, experimentelle Ergebnisse diskutiert.

Die Begriffe *kombinatorische Aspekte* oder *Kombinatorik* werden in diesem Kapitel häufig fallen und sind nicht zu verwechseln mit kombinatorischen Planungsmethoden wie Steven LaValle sie vorstellt [LaV06], oder kombinatorischen Problemen wie dem *multiple depot travelling salesman problem* [MRDJ06].

Die Homotopie als Schlüssel zur Lösung der kombinatorischen Mehrdeutigkeiten wurden bereits durch Bhattacharya *et al.* behandelt [BKL10, BLK11]. Darin wurden Nebenbedingungen beschrieben, um verschiedene Homotopien zu erzwingen. Es ist dort jedoch weder gelungen, ein kontinuierliches Verfahren einzusetzen, noch, eine Trajektorie im Sinne eines *zeit*variablen Zustands zu planen. Die Ergebnisse der Arbeiten sind *Pfade* in zwei oder drei Dimensionen.

In den folgenden beiden Abschnitten werden Homotopien im Bereich der Trajektorienplanung untersucht. Nach einer Lösung mit Hilfe geometrischer Operationen wird eine Vereinfachung für Planungsprobleme auf der Straße aufgezeigt.

4.2 Dynamische Objekte in der statischen Welt

Die bisherige Betrachtung der Homotopie fand in der Ebene statt. Für sich bewegende Verkehrsteilnehmer und das Problem der Trajektorienplanung kommt eine zeitliche Dimension hinzu. In diesem Abschnitt wird eine Methode vorgestellt, um die sich hieraus ergebenden Homotopien zu identifizieren und zur Unterscheidung verschiedener Manöver zu nutzen. Ziel ist – wie im Abschnitt zuvor – die Identifikation aller möglichen Homotopien und die Bestimmung einer Optimallösung je Homotopie.

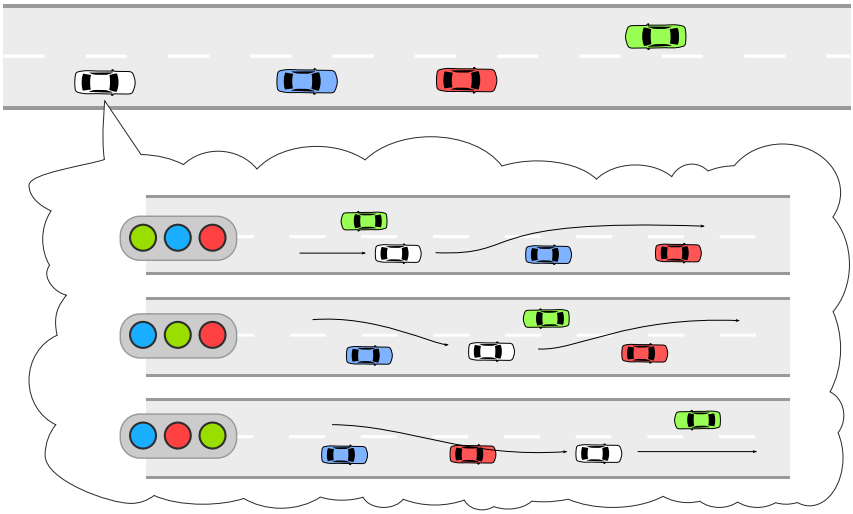
4.2.1 Problemstellung

Gegeben sei eine Situation, in welcher das Ego-Fahrzeug über ausreichend genaue Kenntnis der Trajektorien anderer Verkehrsteilnehmer innerhalb des Planungshorizontes verfügt. Eine solche Situation ist in Abb. 4.3a dargestellt. Die in der Abbildung verwendete Farbkodierung wird auch im weiteren Verlauf der Arbeit verwendet, um die Handlungsoptionen zu unterscheiden.

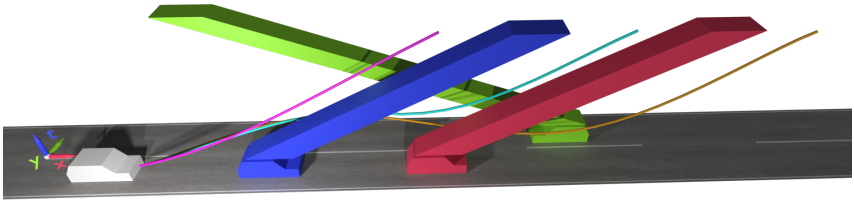
Gesucht ist eine optimale Trajektorie für das weiße Fahrzeug. Dabei ist zu beachten, dass die drei eingezeichneten Trajektorien Teil verschiedener Homotopien sind. Dies wird in der Darstellung 4.3b des gleichen Problems deutlich. Jeder Punkt in den gelb eingezeichneten Ego-Trajektorien ist nur in der (x, y)-Ebene verschiebbar. Unter dieser Bedingung ist es *nicht* möglich, eine der drei Trajektorien in eine der anderen beiden zu überführen.

4.2.2 Lösung

Der Planungshorizont wird *zeitlich* diskretisiert und äquidistant abgetastet. Zu jedem dieser Zeitschritte wird eine Belegtheitskarte bestimmt, welche *örtlich* diskretisiert und eine binäre Information *belegt*/*nicht belegt* bereitstellt. Gesucht sind lokale Optimallösungen für Trajektorien der verschiedenen sich ergebenden Homotopien.

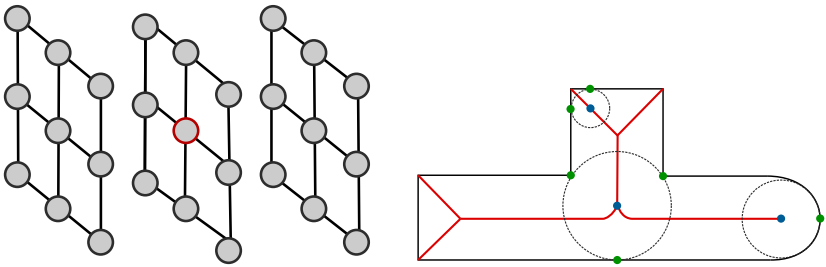


(a) Schematische Sicht der drei möglichen Optionen. Die Farbkodierung spiegelt dabei die Reihenfolge wider, in welcher das Ego-Fahrzeug den anderen Fahrzeugen begegnet.



(b) Das weiße Ego-Fahrzeug prädiziert Trajektorien für die anderen drei Fahrzeuge. Das Bild zeigt die Situation bei t_0, über den Fahrzeugen sind die Prädiktionen eingezeichnet. Für das weiße Fahrzeug sind drei mögliche Trajektorien qualitativ angegeben. Die Trajektorien entsprechen dabei jeweils einer der Möglichkeiten aus der oberen Abbildung.

Abbildung 4.3: Kombinatorisches Problem beim Überholen im Gegenverkehr. Aus Sicht des weißen Fahrzeugs (Ego-Fahrzeug) ergeben sich drei Optionen.



(a) Ein Gitterpunkt im Raum (rot) wird abhängig von seiner 26er-Nachbarschaft als zu entfernend oder nicht zu entfernend klassifiziert.

(b) Ein Beispiel für Mittelachsen in der Ebene. Eingezeichnet sind exemplarisch drei maximale Kreise mit ihren Mittel- und Berührpunkten.

Abbildung 4.4: Mittelachsentransformation im Raum und auf der Ebene.

Identifikation der Homotopien

Zur Identifikation der Manövervarianten wird die Topologie dieses Freiraums herangezogen. Gesucht ist eine Operation, welche das Volumen auf Linien reduziert, diese Linien dabei aber die topologischen Eigenschaften des ursprünglichen Volumens erhalten. Eine solche Operation ist die Mittelachsentransformation. Im zweidimensionalen, kartesischen Raum ist die Lage der Mittelachsen oder der *Medialen Achsen* M_G eines Gebiets $G \in \mathbb{R}^2$ definiert durch die Lage der Mittelpunkte aller maximalen Kreise in G. Ein Kreis ist dann maximal, wenn er vollständig in G liegt und es keinen größeren Kreis gibt, der G enthält. Allgemein reduziert diese Operation die Dimension immer um eins. Wird also ein Volumen (im euklidischer Raum) untersucht, lässt sich die obige Definition analog auf Kugeln anwenden und die Lage der maximalen Kugeln bilden diese Orte keine Linien, sondern Flächen aus. Entsprechend werden diese dann *Mediale Flächen* genannt.

Im Fall der Manövervarianten kann auf die *diskrete* Mittelachsentransformation zurückgegriffen werden um die Lage der Mittelachsen approximativ zu bestimmen. Hierbei wird innerhalb der 26er-Nachbarschaft einer Zelle nach bestimmten Mustern gesucht und eine Zelle so als *entfernbar* oder *nicht entfernbar*

klassifiziert (Abb. 4.4a). Dieses Vorgehen wird so lange wiederholt, bis keine entfernbaren Zellen mehr gefunden werden [LKC94, Pud98]. Für diese Art der Mittelachsentransformation wurde gezeigt, dass sie die Topologie des erodierten Volumens und grundlegende geometrische Eigenschaften erhält[1]. Übrig bleiben zusammenhängende Liniensegmente, entlang derer das Planungsvolumen mit jeweils maximaler Distanz zu einer Grenzfläche durchschritten werden kann. Diese Segmente werden auch *Skelett* genannt.

Ein solches Skelett zeigt Abb. 4.7b. Die magentafarbenen Volumenelemente sind das Ergebnis der Transformation. Sie werden zu einem Graphen vernetzt: zwei Knoten werden dann mit einer ungerichteten Kante verbunden, wenn ihr Abstand $d \leq \sqrt{3}$ ist.

Führen wir nun noch einen Start- und einen Zielknoten ein und verbinden diese mit den jeweils nächstgelegenen Knoten des Skeletts, so stellt jeder mögliche einfache Pfad durch diesen Graphen eine Manövervariante dar. Ein Pfad ist genau dann *einfach*, wenn kein Knoten doppelt vorkommt.

An Stellen, an denen sich der so gebildete Graph verzweigt, können *Cliquen* entstehen. Eine Menge von Knoten bildet genau dann eine Clique, wenn sie einen vollständig vernetzten Teilgraphen darstellt. Der Grad der Clique ist bestimmt durch die Anzahl der Elemente in der Menge. Eine Clique vom Grad 3 bewirkt, dass die Menge einfacher Pfade, welche durch die Clique führt, sich verdoppelt, ohne allerdings eine neue Homotopie zu markieren. Daher werden die Cliquen mit Hilfe des Algorithmus von Bron und Kerbosch [BK73] gefunden und entsprechend Abbildung 4.5 entfernt.

In Abhängigkeit der Auflösung können die Pfade sehr lang sein. Es ist also ein effizienter Weg nötig, um die Knotenanzahl zu reduzieren. Dabei kann der Abstand der Knoten als Kriterium genutzt werden. Ein Knoten c_i soll dann entfernt werden, wenn der Abstand zwischen c_{i-1} und c_{i+1} nicht größer ist als der gewählte Schwellwert.

Ein effizientes Verfahren wird in Abbildung 4.6 gezeigt. Der erste Knoten (0) wird in den neuen Pfad übernommen, gleichzeitig wird er als *aktueller Knoten* gesetzt. Beginnend mit dem nächsten Knoten wird ein *Folgeknoten* gesucht. Dabei werden die folgenden Distanzen so lange aufsummiert, bis das Ende

[1] Linienförmige Strukturen werden nicht erodiert, auch wenn – topologisch betrachtet – eine kurze und eine lange Linie identisch sind und die Erosion somit *topologieerhaltend* wäre.

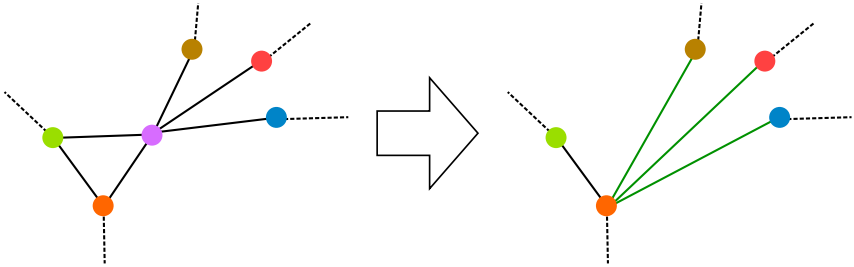


Abbildung 4.5: Cliquen der Größe 3, welche durch den Bron-Kerbosch-Algorithmus detektiert wurden, werden aufgelöst. Einer der Knoten wird entfernt, alle seine Nachbarknoten (die nicht Teil der Clique sind) werden mit einem der beiden verbleibenden Knoten verbunden.

erreicht ist oder der Schwellwert beim nächsten Knoten überschritten würde. Ist ein Folgeknoten gefunden, wird er in den Pfad übernommen, als *aktueller Knoten* gesetzt und es wird neu iteriert.

Abb. 4.7a zeigt dieses Vorgehen anhand einer einfachen Längsplanung, da in diesem Fall eine Dimension verschwindet und das Verfahren besser verständlich wird. In der Szene werden zwei Hindernisse betrachtet, die nur zu bestimmten Zeitpunkten vorhanden sind, aber jeweils am gleichen Ort. Dies dient nur der Illustration: in der Ebene existieren einfachere Verfahren, um die Mittelachsen (Abb. 4.4b) zu bestimmen.

Optimierung der Zeitkomponente

Für jede Manövervariante liegt eine Initialisierung vor, also ein Pfad, dessen Punkte in gewünschter Weise um die Hindernisse führen. Diese Initialisierung genügt keinen weiteren Bedingungen, weder fahrdynamischen noch solchen, welche die strenge Monotonie in der Zeitrichtung fordern.

Um dies zu erreichen, wird in zwei Stufen optimiert. Im ersten Schritt wird dafür gesorgt, dass die Initialisierung in ihrer zeitlichen Koordinate monoton ansteigt und die Schritte möglichst äquidistant sind. Hierfür wird die Summe

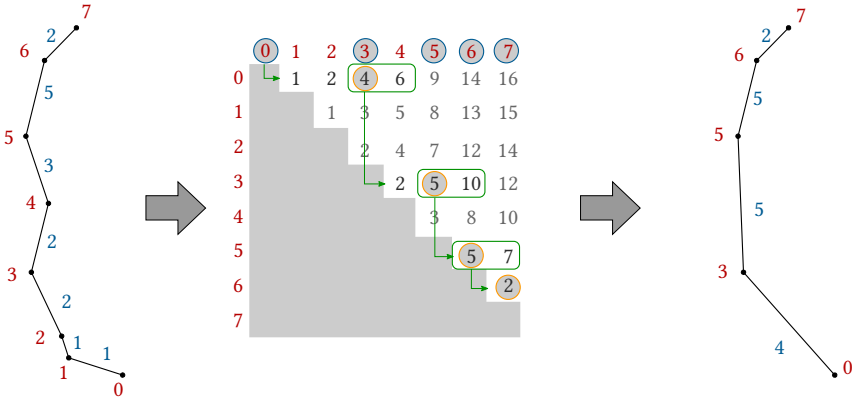
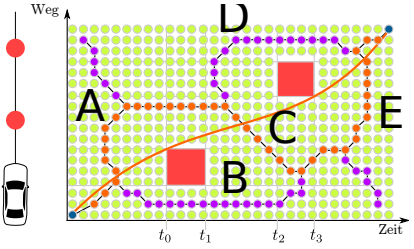
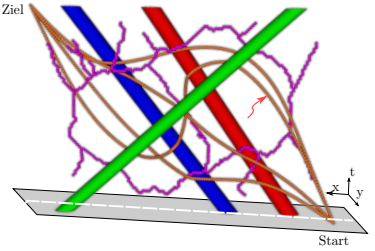


Abbildung 4.6: Entfernen von Knoten aus einem Pfad ohne einen bestimmten Höchstabstand zu überschreiten. Links ist der ursprüngliche Pfad mit Knotennummern und Entfernungen abgebildet, in der Mitte die konstruierte Tabelle, und rechts der resultierende, reduzierte Pfad. Die hellgrauen Werte in der Tabelle werden nicht benötigt und müssen auch nicht berechnet werden.



(a) Zuerst wird der Planungsraum diskretisiert, die Hindernisse werden abgezogen, die verbleibenden Punkte werden der Mittelachsentransformation unterzogen, welche die grünen Punkte entfernt. Die übrigen Punkte werden vernetzt, einschließlich der Start- und Endpunkte. Alle einfachen Pfade werden gesucht, exemplarisch ist einer der vier Pfade eingezeichnet (orange). Dieser dient als Initialisierung für die kontinuierliche Optimierung, das Ergebnis ist als durchgezogene Linie dargestellt.



(b) Beispiel in 3-D. Zu sehen sind drei Hindernisse (gescherte Zylinder), das Ego-Fahrzeug plant von links unten nach rechts oben (die Zeit steht senkrecht zur Straßenebene). Eingezeichnet sind vier Lösungen, von denen drei den Varianten aus Abb. 4.3a entsprechen. Die mit einem Pfeil markierte Trajektorie ist ungültig, da es bei dieser nicht möglich ist, die Forderung nach strenger Monotonie der Zeitkomponente zu erfüllen. Die Mittelachsentransformation ist magentafarben dargestellt.

Abbildung 4.7: Skelett des Freiraums in 2-D und 3-D.

der euklidischen Distanzen zwischen den einzelnen Punkten minimiert. Die Schwierigkeit liegt an dieser Stelle in der Formulierung einer geeigneten Nebenbedingung, die sicherstellt, dass die Punkte das Planungsvolumen und damit ihre Homotopie nicht verlassen. Die Aufgabe lautet *minimiere eine Gütefunktion unter der Nebenbedingung, dass eine Homotopie zwischen der Initialisierung und der Lösung existiert*, was in der Praxis einige Probleme mit sich bringt. Dies wird durch eine geeignete Einstellung des Optimierers gelöst: Die Schrittweite wird auf die minimale Ausdehnung eines Hindernisses begrenzt, sodass diese nicht übersprungen werden können. Folgt die Initialisierung der gewünschten Variante und wird kein Hindernis im Laufe der Optimierung geschnitten, so besteht eine Homotopie zwischen beiden Trajektorien. Das Ergebnis dieses ersten Schrittes erinnert an ein Gummiband, welches zwischen Start und Ziel gespannt ist und an bestimmten Punkten die Hindernisse berührt.

Anhand dieses Ergebnisses kann nun für jede Variante entschieden werden, ob sie durchführbar ist. Hierzu muss lediglich überprüft werden, ob die optimierte Zeitkomponente streng monoton steigend ist.

Optimierung der Ortskomponenten

In diesem Schritt wird die nun topologisch und temporär gültige Initialisierung im Sinne einer Trajektorie für ein Fahrzeug optimiert. Die folgende Herleitung des Optimierungsproblems basiert auf der Arbeit von Ziegler *et al.* , wo auch eine Diskussion der einzelnen Terme stattfindet [ZBDS14]. Im Anschluss daran wird das Problem so umformuliert, dass ein Löser für nicht-lineare Ausgleichsprobleme eingesetzt werden kann.

Eine fahrbare Trajektorie soll das Integral

$$J[\mathbf{x}(t)] = \int_{t_0}^{t_0+T} L(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, \dddot{\mathbf{x}})\, \mathrm{d}t \tag{4.1}$$

minimieren, wobei $\mathbf{x}(t)$ die Position in kartesischen Koordinaten angibt. Das Funktional L ist gegeben durch

$$L = j_{\text{offs}} + j_{\text{vel}} + j_{\text{acc}} + j_{\text{jerk}}. \tag{4.2}$$

Die einzelnen Summanden verursachen Kosten für Außermittigkeit relativ zum Fahrstreifen, Abweichung von der Wunschgeschwindigkeit und für dynamische Aspekte wie Beschleunigung, Gierrate und Ruck. Der Einfluss der Summanden wird durch die Gewichtungsfaktoren $w_{\text{offs}}, w_{\text{vel}}, \ldots$ gesteuert.

Die Kosten für die Abweichung von der Mittellinie ergeben sich zu

$$j_{\text{offs}}(\mathbf{x}(t)) = w_{\text{offs}} \left\| \frac{1}{2}(d_{\text{left}}(\mathbf{x}(t)) + d_{\text{right}}(\mathbf{x}(t))) \right\|^2, \tag{4.3}$$

wobei d_{left} und d_{right} vorzeichenbehaftete Abstandsfunktionen von den jeweiligen Fahrstreifenrändern links und rechts darstellen. Die Kosten für die Geschwindigkeitsabweichungen, Beschleunigung und Ruck berechnen sich wie folgt:

$$j_{\text{vel}}(\mathbf{x}(t)) = w_{\text{vel}} \left\| \mathbf{v}_{\text{des}}(\mathbf{x}(t)) - \dot{\mathbf{x}}(t) \right\|^2 \tag{4.4}$$

$$j_{\text{acc}}(\mathbf{x}(t)) = w_{\text{acc}} \left\| \ddot{\mathbf{x}}(t) \right\|^2 \tag{4.5}$$

$$j_{\text{jerk}}(\mathbf{x}(t)) = w_{\text{jerk}} \left\| \dddot{\mathbf{x}}(t) \right\|^2 \tag{4.6}$$

Nicht berücksichtigt wurde bisher, dass die Trajektorie durch einzelne Punkte angenähert wird. Hierdurch lässt sich das Integral durch eine Summe approximieren, die Ableitungen durch finite Differenzen:

$$\Delta = t_{i+1} - t_i \tag{4.7a}$$

$$\dot{\mathbf{x}}(t_i) \approx \mathbf{v}_i = \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\Delta} \tag{4.7b}$$

$$\ddot{\mathbf{x}}(t_i) \approx \mathbf{a}_i = \frac{\mathbf{v}_{i+1} - \mathbf{v}_i}{\Delta} \tag{4.7c}$$

$$\dddot{\mathbf{x}}(t_i) \approx \mathbf{j}_i = \frac{\mathbf{a}_{i+1} - \mathbf{a}_i}{\Delta} \tag{4.7d}$$

$$\tag{4.7e}$$

Δ bezeichnet hier den zeitlichen Abstand zweier diskreter Trajektorienpunkte.

Da alle Beiträge zur Summe im Quadrat stehen, kann Gl. 4.1 zu

$$[\mathbf{x}_4, \dots, \mathbf{x}_N] = \underset{\mathbf{x}_4,\dots,\mathbf{x}_N}{\arg\min} (\mathbf{W}\mathbf{r})^\top \mathbf{r} \tag{4.8}$$

umgeformt werden. Hierbei ist $\mathbf{r}$ eine Menge von Residuen und $\mathbf{W}$ eine Diagonalmatrix mit den entsprechenden Gewichten.

$\mathbf{x}_1, \mathbf{x}_2$ und $\mathbf{x}_3$ sind keine freien Parameter: sie werden festgehalten, um eine Initialdynamik vorzugeben.

Soll die Trajektorie durch N Punkte angenähert werden, so ergeben sich die Residuen aus:

$$\begin{aligned} \mathbf{r} = (&r_{\text{offs},1}, \dots, r_{\text{offs},N}, \\ &r_{\text{vel},1}, \dots, r_{\text{vel},N-1}, \\ &r_{\text{acc},1}, \dots, r_{\text{acc},N-2}, \\ &r_{\text{jerk},1}, \dots, r_{\text{jerk},N-3})^\top \end{aligned} \tag{4.9}$$

Weiterhin darf die Trajektorie keines der Hindernisse schneiden. Wird ein Hindernis durch einen Kreis angenähert, gilt weiterhin

$$\left\|\mathbf{h}_i^n - \mathbf{x}_i\right\| - R^n \geq 0, \tag{4.10}$$

für jedes Trajektoriensample $\mathbf{x}_i, i \in 4, \ldots, N$ und die Position $\mathbf{h}_i^n$ des n-ten Hindernisses zum Zeitpunkt i. R^n ist dabei der Radius der Kreisannäherung. Wird ein Hindernis durch mehrere Kreise angenähert, ist das Vorgehen gleich, es müssen nur entsprechend mehrere Kreise gewählt werden. Wird ein Hindernis durch ein Polygon angenähert, muss eine Abstandsfunktion gewählt werden, welche über das Vorzeichen signalisiert, ob sich ein Punkt innerhalb (negativ) oder außerhalb (positiv) des Polygons befindet. Werden Belegtheitskarten fortgeschrieben, müssen entsprechend die belegten Gitterzellen über Kreise ausgeschlossen werden (Abb. 4.8)

Das so gestellte Minimierungsproblem kann z. B. mit Ceres[2] gelöst werden. Ceres ist ein effizienter Löser für derartige Probleme, erlaubt allerdings nur Nebenbedingungen auf den Parametern selbst. Daher müssen die erforderlichen Nebenbedingungen durch weitere Kostenterme angenähert werden. Hierbei wird für jede Ungleichheitsnebenbedingung ein Residuum gebildet, dessen Wert 0 ist, falls die Bedingung erfüllt ist, oder ein langsam ansteigender Wert, welcher den Grad der Verletzung der Bedingung angibt. Wird dieses Residuum geeignet (stark) gewichtet, kann Ceres sehr gut mit dieser Approximation umgehen. Dieses Vorgehens wird *Barrier-Methode* genannt [GMW81].

4.2.3 Experimente und Ergebnisse

Dieser Beitrag ergänzt die kontinuierliche Trajektorienplanung um einen diskreten Aspekt, der mit Hilfe der Freiraumdiskretisierung und der Mittelachsentransformation gelöst wurde. Der kontinuierliche Teil wird als Least-Squares-Problem ohne harte Nebenbedingungen formuliert.

Als Szenario wurde der Überholvorgang im Gegenverkehr gewählt, der sich insbesondere dadurch auszeichnet, dass er auch im Alltag oft Gegenstand riskanter Manöver ist und es hier sehr stark vom Fahrer abhängt, wie der Vorgang gestaltet wird, siehe Abb. 4.3a auf Seite 54. In Tabelle 4.1 auf Seite 66 und den Abb. 4.9 und 4.10 werden die wichtigsten Ergebnisse zusammengefasst.

[2] Ceres, ein nichtlinearer Least-Squares-Solver, `http://ceres-solver.org`

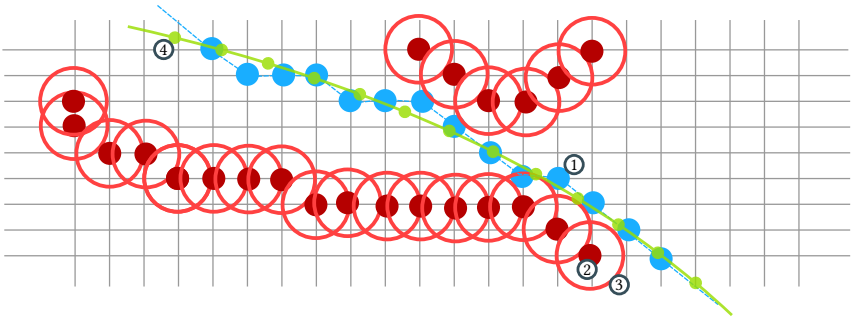


Abbildung 4.8: Werden Hindernisse nicht explizit modelliert, sondern nur implizit über das Forschreiben einer Belegtheitskarte, dann können die belegten Gitterzellen durch Kreise angenähert werden.

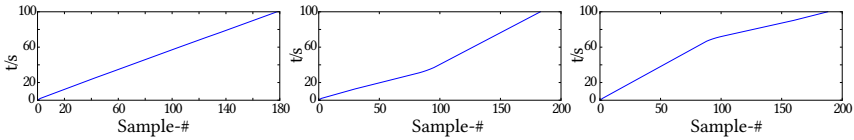
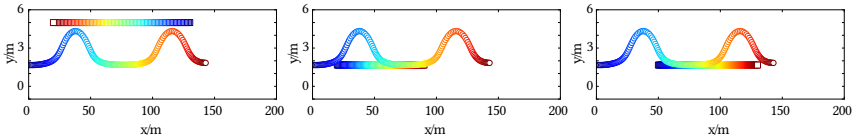


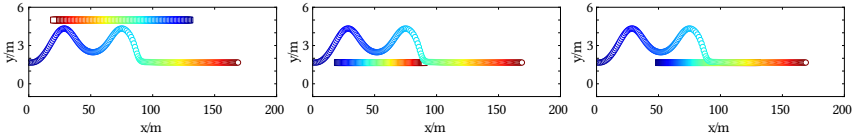
Abbildung 4.9: Zeitkomponente der Trajektorien nach der ersten Optimierungsstufe. Auf der Abszisse aufgetragen ist die Nummer des Samples, auf der Ordinate die zugehörige Zeit.

Die Ausweichtrajektorien werden in Abb. 4.10 gezeigt. Dabei stellt jede Zeile eine eigene Variante (Homotopie) dar. Die Ego-Trajektorie ist in den drei Diagrammen pro Zeile jeweils gleich, es werden aber pro Spalte die Hindernisse variiert, um Übersichtlichkeit zu wahren. Die Farben kodieren absolute Zeiten, sodass der Abstand zweier Punkte gleicher Farbe dem Abstand entspricht, den die Trajektorien voneinander zu diesem bestimmten Zeitpunkt haben.

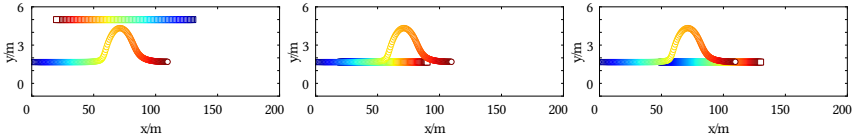
Zusammenfassend lässt sich sagen: Die topologische Exploration mit Hilfe morphologischer Verfahren ermöglicht die Identifikation von Homotopien für komplexe Szenarien. Außer der Darstellung als Belegtheitskarte und damit verbundener Prädiktion dieser Karte für die Zukunft ist keine weitere Abstraktion



(a) Überhol- und Ausweichtrajektorie für die Reihenfolge (●, ●, ●).



(b) Überhol- und Ausweichtrajektorie für die Reihenfolge (●, ●, ●).



(c) Überhol- und Ausweichtrajektorie für die Reihenfolge (●, ●, ●).

Abbildung 4.10: Ausweichtrajektorien für die drei identifizierten Manövervarianten. Die Zeilen zeigen jeweils die Ego-Trajektorien der Manövervarianten, die Spalten zeigen eine Hindernistrajektorie. Die erste Spalte zeigt die Ego-Trajektorie zusammen mit Trajektorie des grünen Fahrzeugs (●, ○), die nächsten Spalten entsprechend (●, ○) und (●, ○) Zur besseren Übersichtlichkeit sind die Ausdehnungen der Objekte jeweils verkleinert. Die Achsen zeigen Orte an, die Farben kodieren die absoluten Zeiten.

Variante	Anz. Samples	Anz. Parameter	Kosten	Strecke/m
(●, ●, ●)	179	352	$1,66 \cdot 10^3$	149
(●, ●, ●)	184	362	$3,02 \cdot 10^3$	168
(●, ●, ●)	189	372	$1,41 \cdot 10^3$	110

Tabelle 4.1: Ergebnis des kontinuierlichen Planers. Nach dem gewählten Kriterium ist (●, ●, ●) die beste Variante. Die Strecke gibt hier den Fortschritt entlang der Straße an, dieser Wert wird jedoch implizit über die Kosten berücksichtigt und sollte bei der Auswahl keine Rolle mehr spielen. Die Angabe dient hier lediglich der Orientierung. Die Farbkodierung zur Unterscheidung der Varianten findet sich in Abb. 4.3a wieder.

notwendig. Die Ergebnisse zeigen, dass die so identifizierten Varianten mit den in Abb. 4.3a unterstellten Varianten übereinstimmen.

Da Objekte nicht explizit adressiert werden, sondern im morphologisch modifizierten Planungsvolumen ihre Identität verlieren, ist es schwer, Hystereseeffekte zu berücksichtigen. Diese können jedoch für zeitlich schlüssiges Verhalten eine wichtige Rolle spielen. Kann dies durch weitere Vereinfachungen behoben werden? Hierzu bietet der folgende Abschnitt eine Antwort, die Diskussion der Konsistenz findet in Kapitel 4.3.4 statt.

4.3 Dynamische Objekte auf der Straße

Das Fahren auf der Straße hat die Eigenschaft, dass die Pfade der anderen Verkehrsteilnehmer stark vorhersehbar sind: Fahrzeuge fahren üblicherweise innerhalb ihres Fahrstreifens, Abweichungen davon unterliegen gewissen Regeln und sind meist angekündigt und erwartbar. Das führt dazu, dass auch für das eigene Fahrzeug im Wesentlichen ein Längsprofil geplant werden muss. Das Querprofil ist durch den Fahrstreifen vorgegeben. In einer Kurve passt ein Fahrzeug sehr viel stärker seine Geschwindigkeit an als dass es seinen Pfad ändert,

da dieser – etwa im Falle einfacher Ortsverbindungsstraßen – nur eine laterale Abweichung von 40 Zentimetern um die Fahrstreifenmitte herum zulässt.

Diese Idee der festen Pfade und variablen Geschwindigkeitsprofile wurde mit der Path-Velocity-Decomposition (PVD) [KZ86, FL93] aufgegriffen. Abb. 4.11 skizziert ein entsprechendes Planungsproblem. Die graue Box in der rechten Hälfte kennzeichnet einen räumlichen Konflikt, welcher nur zu bestimmten Zeiten besteht. Obwohl sich beide Pfade (in die $(x \times y)$-Ebene projizierte Trajektorien) schneiden, sind die Trajektorien selbst kollisionsfrei. Das Beispiel zeigt auch den kombinatorischen Aspekt auf: Die rote Trajektorie kann auch so geplant werden, dass sie *nach* der schwarzen Trajektorie durch den Konfliktbereich führt. Im $(s \times t)$-Diagramm bedeutet das, dass die Kurve unter der grauen Box verläuft. Diese Probleme sind effizient zu lösen [JH12, JH13], haben aber zwei Nachteile. Zum einen sind die Trajektorien nicht notwendigerweise *optimal* bezüglich eines Gütekriteriums, welches etwa Querbeschleunigung und Ruck berücksichtigt. Wenn der Pfad für eher große Geschwindigkeit geplant ist (und entsprechend kleine Krümmungen aufweist), dann ist eine *langsamere* Trajektorie konservativer bezüglich fahrdynamischer Limitierungen als nötig. Schnellere Trajektorien können das Fahrzeug hingegen in Bereiche bringen, welche fahrdynamisch ausgeschlossen sind. Zum anderen – und das ist der im Rahmen dieser Arbeit relevante Unterschied – *existiert* für einige Manöver kein derartiger, vorab bestimmbarer Pfad [NGGP08]. Diese Untrennbarkeit von Ort und Zeit ist in Abb. 4.12 skizziert.

Anstelle einer Entkopplung von *Pfad* und *Geschwindigkeit* wird in diesem Abschnitt eine Entkopplung von *Manöver* und *Geschwindigkeit* vorgestellt. Durch eine kombinatorische Analyse einer Szene und deren Zerlegung in diskrete Manöver werden Nebenbedingungen hergeleitet, welche ein bestimmtes Manöver erzwingen und die Nutzung eines kontinuierlichen Lösungsverfahrens erlauben.

4.3.1 Betrachtungen im Weg-Zeit-Diagramm

Das oben angesprochene *Längsprofil* stellt die hauptsächlich zu optimierende Größe dar. Eine Möglichkeit der Visualisierung sind *Weg-Zeit-Diagramme*. Hierbei wird die Längskoordinate über der Zeit aufgetragen. Die Konversion zwischen den Koordinatensystemen ist z. B. mit der in [ZBDS14] vorgestellten kontinuierlichen Abstandsfunktion möglich.

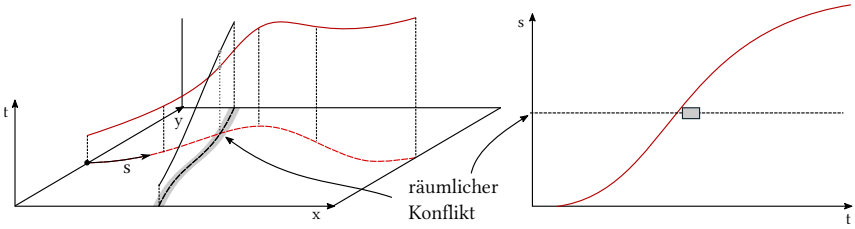


Abbildung 4.11: Die PVD. Für vorab bestimmte Pfade (links, rot und schwarz gestrichelt) werden Geschwindigkeitsprofile bestimmt. Dadurch werden örtliche Konflikte aufgelöst. Rechts zu sehen ist die Sicht der roten Trajektorie, deren Pfadkomponente sich mit der schwarzen Trajektorie schneiden wird, was aber nur zu bestimmten Zeiten (graue Box) tatsächlich zu einem Problem führt. Dies wird hier gelöst, indem die rote Trajektorie den kritischen Ort vor der schwarzen Trajektorie passieren wird (zu sehen am Abstand in t-Richtung im linken Bild).

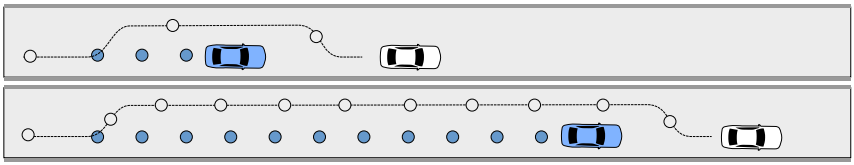


Abbildung 4.12: Überholmanöver, schnell (oben) und langsam (unten). Nicht nur die Geschwindigkeitsprofile (angedeutet durch die Kreise, welche äquidistante Zeitpunkte repräsentieren), sondern auch die Pfade selbst unterscheiden sich. Der Pfad hängt vom Geschwindigkeitsprofil ab.

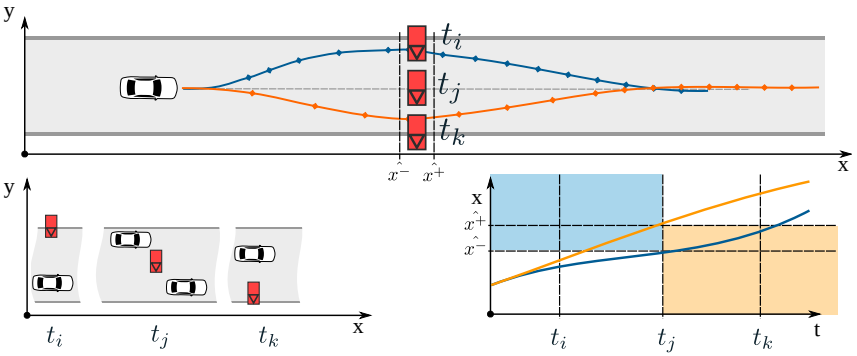


Abbildung 4.13: Beispiel für querenden Verkehr. Das Hindernis (rot, Position für drei diskrete Zeiten $t_{i,j,k}$ eingezeichnet) blockiert keinen Abschnitt vollständig, trotzdem sind zwei Entscheidungen möglich. Die Entscheidung für die blaue Trajektorie schließt den blauen Bereich aus, die Entscheidung für die orangene Trajektorie den orangenen Bereich, eingezeichnet in das Weg-Zeit-Diagramm unten rechts.

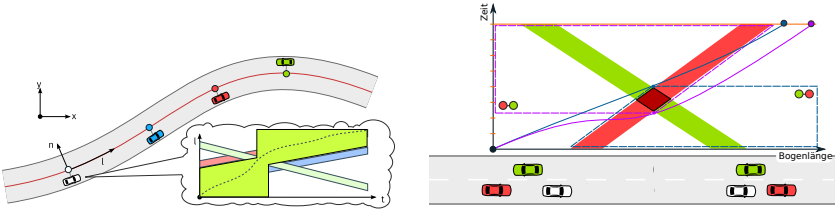
Das Weg-Zeit-Diagramm visualisiert den Längsverlauf blockierter Bereiche. Im Längsverkehr entstehen blockierte Abschnitte dann, wenn sich zwei Objekte begegnen. Das kann auch der Fall sein, wenn ein Fahrzeug eine Engstelle passiert. Bei Querverkehr ist ein blockierter Abschnitt immer dann vorhanden, wenn sich ein Hindernis über die komplette Breite der Straße erstreckt. Die so blockierten Abschnitte haben die Form von Rechtecken. Ist das Hindernis *kleiner* (und kein Abschnitt der Straße komplett blockiert), so findet dieser Fall *keine* Repräsentation im Weg-Zeit-Diagramm. Trotzdem sind diskrete Entscheidungen nötig wie in Abb. 4.13 auf der vorherigen Seite gezeigt wird.

Im Längsverkehr ergeben sich rhombenförmige Muster für den Fall, dass die Teilnehmer mit konstanter Geschwindigkeit fahren. Die markanten Punkte des Rhombus ergeben sich aus Gleichsetzen der Position für die vier möglichen Kombinationen aus den Koordinaten für jeweils vorderes und hinteres Ende der beiden sich begegnenden Fahrzeuge.

4.3.2 Homotopie: Begegnungsreihenfolge

Die Ansatz ist, die *Begegnungsreihenfolge* als Manöver unterscheidendes Merkmal heranzuziehen. Hierfür wird eine Notation verwendet, welche Fahrzeuge farblich gemäß den Abbildungen kodiert und die Begegnungsreihenfolge aus Sicht des Ego-Fahrzeugs wiedergibt. Die Variante (●, ●, ●) in Abb. 4.14a adressiert beispielsweise das Manöver, in welchem das Ego-Fahrzeug zuerst das blaue Fahrzeug überholt, dann in der Lücke zwischen blau und rot einschert, um das grüne Fahrzeug passieren zu lassen, bevor es den Überholvorgang fortsetzt. Durch den Bezug der Begegnungsreihenfolge auf eine Referenzlinie wird das Planungsproblem Ungleichheitsnebenbedingungen entlang dieser Linie unterworfen. Die auf die Referenzlinie projizierte Position muss also zu bestimmten Zeitpunkten innerhalb bestimmter Intervalle liegen, wie es in der Abbildung dargestellt ist.

Jede mögliche Begegnungsreihenfolge stellt eine Manövervariante dar. In jeder Variante werden die anderen Fahrzeuge paarweise betrachtet. Hieraus ergeben sich die Nebenbedingungen, welche die jeweilige Begegnungsreihenfolge erzwingen. Aus der obigen Begegnungsreihenfolge resultieren die Paarungen (●-●, ●-●). Wir betrachten die Paarung ●-●, die auch in Abb. 4.14b dargestellt wird. Hier wird gefordert, dass das Ego-Fahrzeug zuerst das grüne Fahrzeug passiert,



(a) Reduktion des Planungsproblem auf eine Referenzlinie, um eine Reihenfolge sicherzustellen. Durch Kombination von Ausschlussbereichen verbleibt der grüne Bereich, in dem sich die Trajektorie bewegen darf.

(b) Beispiel mit zwei Fahrzeugen. Die gewählte Reihenfolge ●● schließt den blau umrandeten Bereich aus. Die Trajektorie (ebenfalls blau) muss sich also außerhalb dieses Bereichs bewegen.

Abbildung 4.14: Die Nebenbedingungen für das Planungsproblem werden aus dem Weg-Zeit-Diagramm abgeleitet.

also hinter dem roten Fahrzeug verbleiben muss, bis sich die beiden Fahrzeuge begegnet sind.

Diese Forderung schließt den blau umrandeten Bereich aus. Bei mehreren solcher zeitlicher Forderungen werden diese *und*-verknüpft.

4.3.3 Konstruktion der Nebenbedingungen

Doch wie werden die möglichen Varianten berechnet? Nicht jede Permutation stellt gleichzeitig eine mögliche Variante dar. Im Beispiel 4.14a wäre (●, ●, ●) *nicht* möglich.

Zuerst werden alle möglichen Paare gebildet. Im gewählten Beispiel sind das (●, ●), (●, ●) und (●, ●). Im nächsten Schritt werden die Paare wie folgt betrachtet:

Für ein Hindernispaar (a, b), dessen Position $a^l(t)$, $b^l(t)$ über die Zeit t als bekannt angenommen wird, können die Zeitpunkte bestimmt werden, in welchen das Paar alle Fahrstreifen blockiert und somit ein Überholen nicht möglich ist. Hierfür werden die Positionen wie in Abb. 4.15 gleichgesetzt und nach t aufgelöst. Im Fall einer wie in der Abbildung angenommenen konstanten Geschwindigkeit ergeben sich die Zeitpunkte zu

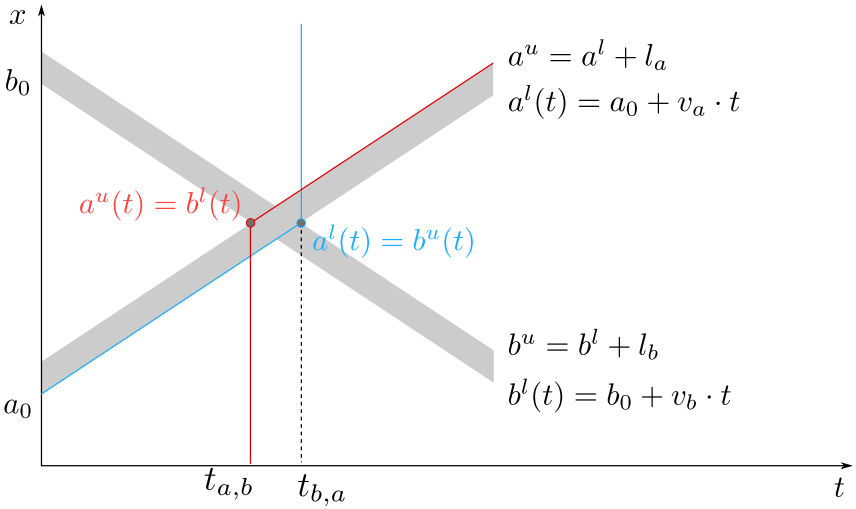


Abbildung 4.15: Konstruktion der Nebenbedingungen bezüglich der Längspositionen. Gegeben ist ein Hindernispaar $(\{a_0, v_a, l_a\}, \{b_0, v_b, l_b\})$, dessen Begegnung bei $t_{a,b}$ beginnt und bei $t_{b,a}$ endet. Die geometrischen Zusammenhänge sind der Zeichnung zu entnehmen. Eine gültige Trajektorie $x_{a,b}$ ist nach unten durch die rote Kurve beschränkt, eine Trajektorie $x_{b,a}$ ist nach oben durch die blaue Kurve beschränkt.

$$t_{a,b} = \frac{-l_a - a_0 + b_0}{v_a - v_b} \tag{4.11}$$

$$t_{b,a} = \frac{l_b - a_0 + b_0}{v_a - v_b}. \tag{4.12}$$

Für die Längskomponente $x_{a,b}$ einer Trajektorie, welche zuerst Hindernis a und dann Hindernis b passieren wird, gilt dann

$$x_{a,b}(t) \geq \begin{cases} 0, & \text{wenn } t < t_{a,b} \\ b^l(t) + l_b, & \text{sonst} \end{cases} \tag{4.13}$$

und für den Fall b, a entsprechend

$$x_{b,a}(t) \leq \begin{cases} a^l(t), & \text{wenn } t < t_{b,a} \\ \infty, & \text{sonst.} \end{cases} \tag{4.14}$$

Für das Paar (●, ●) entstehen also die beiden Möglichkeiten ●-● sowie ●-●, welche jeweils mit den entsprechenden Nebenbedingungen verknüpft sind (entsprechend der Indizes $\cdot_{a,b}$, $\cdot_{b,a}$). Für alle Paare ergeben sich die Möglichkeiten

$$\{●-●, ●-●\} \times \{●-●, ●-●\} \times \{●-●, ●-●\},$$

also acht Kombinationen (allgemein: 2^N bei N Fahrzeugen). Innerhalb einer Kombinationen werden die Nebenbedingungen betrachtet und miteinander verknüpft. Obere Grenzen werden mit dem *min*-Operator verknüpft, untere Grenzen entsprechend mit *max*.

Eine Kombinationen scheidet als Variante dann aus, wenn die obere Grenze nicht durchgängig größer ist als die untere Grenze – dies ist eine notwendige, aber nicht hinreichende Bedingung, eine Variante kann auch ausscheiden, weil innerhalb gültiger Grenzen keine fahrbaren Trajektorien gefunden werden können.

4.3.4 Zeitliche Konsistenz der Entscheidung für eine Homotopie

Wie im vorherigen Kapitel muss auch hier untersucht werden, wie die zeitliche Konsistenz der Entscheidung sichergestellt werden kann. Dies ist durch die explizite Adressierung der Hindernisse und der durch Hindernisse blockierten Abschnitte möglich.

Die Idee dabei ist, die verschiedenen Varianten mit ihrer jeweiligen Güte über der Zeit zu beobachten. Dabei fällt die Entscheidung initial auf die Variante mit den geringsten Kosten (der besten Güte) und wird im weiteren Verlauf nur geändert, wenn eine andere Variante um einen gewissen Mindestbetrag günstiger ist. Eine neu geplante Trajektorie darf sich also von der alten Trajektorie unterscheiden, muss aber homotop zu ihr sein.

Der Gedanke dahinter ist, dass es unter Umständen sicherer sein kann, eine dynamischere Trajektorie zu akzeptieren und dafür in den Augen anderer Verkehrsteilnehmer konsistenter zu agieren. Exemplarisch sei hier das Vorbeifahren an einem parkenden Fahrzeug genannt. Kurz nach Start des Vorgangs ändert sich die Prädiktion für ein entgegenkommendes Fahrzeug. Um trotzdem noch Vorbeifahren zu können, muss mit größerer Beschleunigung gearbeitet werden. Dies wird in einem gewissen Maß akzeptiert, um andere Verkehrsteilnehmer nicht mit einer sehr kurzfristigen Umentscheidung zu verwirren. Siehe dazu Abb. 4.16 auf der nächsten Seite.

4.3.5 Experimente und Ergebnisse

Für die Experimente in diesem Abschnitt wird ein einfacher SLSQP-Löser verwendet.

Szenario mit drei Fahrzeugen Betrachtet wird wieder ein Szenario *Überholen im Gegenverkehr*, diesmal in einer kleinen Abwandlung gegenüber Kap. 4.2.3. Dem Ego-Fahrzeug, welches sich hinter Fahrzeug ● befindet, kommen Fahrzeug ● und Fahrzeug ● entgegen.

Der Planungshorizont von 15 s wird mit 50 Samples diskretisiert. Optimiert wird in zwei Schritten: (1) Optimierung der Längskoordinate x und (2) Optimierung der Querkoordinate y.

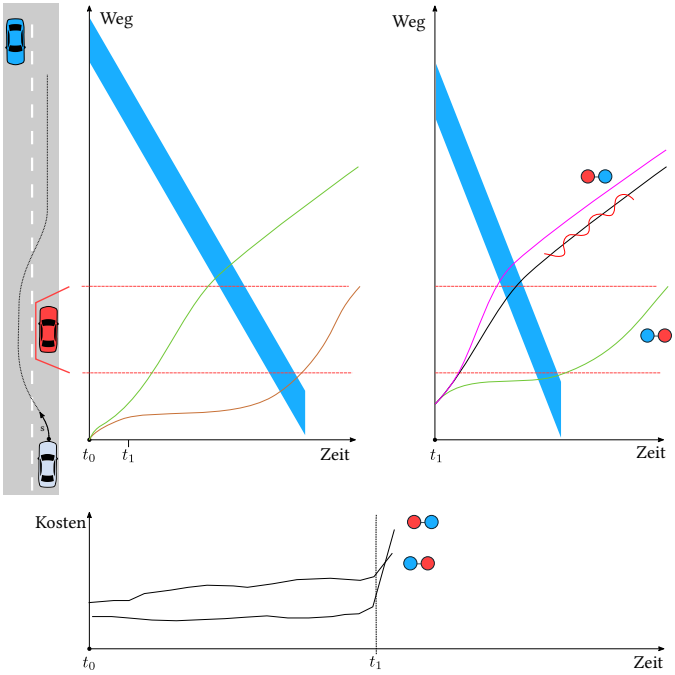


Abbildung 4.16: Zeitliche Konsistenz. In diesem Beispiel ändert sich zum Zeitpunkt t_1 die Prädiktion für das blaue Fahrzeug (Ego-Fahrzeug: weiß): Es bewegt sich schneller, die Analyse im Weg-Zeit-Diagramm zeigt einen zeitlich gesehen früheren Schnittpunkt mit der Wegkoordinate des rot markierten Hindernisses. Die Kosten steigen sprunghaft, und die initial günstigere Manövervariante wird teurer als die Alternative. Hier ist es zu überlegen, mit geeigneten Hysteresen einmal getroffene Entscheidungen beizubehalten, um das eigene Handeln für andere Verkehrsteilnehmer nachvollziehbar zu machen.

Variante	Anz. Samples	Anz. Parameter	Kosten	Strecke/m
(●, ●, ●)	50	94	$46 \cdot 10^3$	200
(●, ●, ●)	50	94	$37 \cdot 10^3$	180
(●, ●, ●)	50	94	$3 \cdot 10^3$	116

Tabelle 4.2: Ergebnis des kontinuierlichen Planers. Nach dem gewählten Kriterium wäre (●, ●, ●) die beste Variante. Die Strecke gibt hier den Fortschritt entlang der Straße an, wie im letzten Abschnitt aber wird dieser Wert implizit über die Kosten auf der Abweichung von der Wunschgeschwindigkeit berücksichtigt und dient nur der Orientierung.

Dabei werden im ersten Schritt die Nebenbedingungen gemäß Abb. 4.15 und Gl. 4.13, 4.14 auferlegt. Mit Hilfe des so ermittelten Längsprofils werden die Zeitpunkte ermittelt, zu denen sich das Ego-Fahrzeug auf dem benachbarten Fahrstreifen aufhalten muss, um ein Überholen zu ermöglichen. Zu diesen Zeitpunkten wird die Querkoordinate dann mit den entsprechen Nebenbedingungen belegt.

Das Ergebnis sind die Trajektorien in den Abbildungen 4.17, 4.18, 4.19. Es sind jeweils die Ergebnisse der beiden Schritte dargestellt.

Zeitliches Verhalten Um die Aussagen bezüglich der zeitlichen Konsistenz zu motivieren, wird nun die Entfernung der Hindernisse variiert. Dies ist äquivalent zu einer Bewegung des Ego-Fahrzeugs oder einer Änderung in der Prädiktion der anderen Verkehrsteilnehmer. Abb. 4.20 zeigt dabei den Verlauf der Kosten. Wie erwartet ändert sich mit der Variation die jeweils günstigste Trajektorie.

Szenario mit vier Fahrzeugen Erweitert wird dieser Aufbau durch ein weiteres Fahrzeug auf dem eigenen Fahrstreifen. Der zeitliche Horizont ist wieder 15 s, diesmal mit 80 Punkten diskretisiert. Erwartungsgemäß erhöht sich die Anzahl der Möglichkeiten. In diesem Beispiel bleiben fünf Varianten

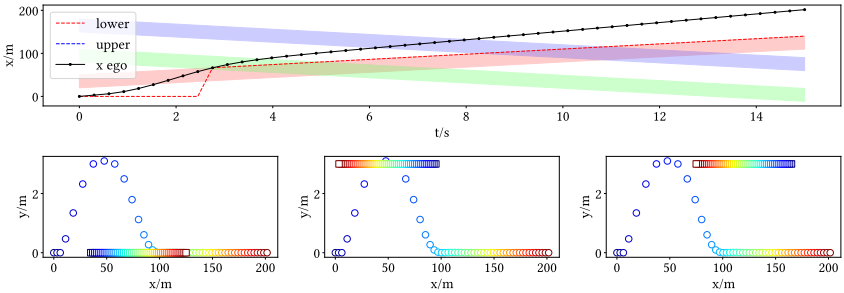


Abbildung 4.17: Variante (●, ●, ●). Die obere Zeile zeigt den Verlauf der Längskoordinate x über der Zeit. Die untere Zeile zeigt die Ego-Trajektorie (x, y) sowie pro Spalte jeweils eine Hindernistrajektorie. Die Farben kodieren dabei die unterschiedlichen Zeiten. Von links nach rechts sind die Hindernistrajektorien ●, ● und ● aufgetragen.

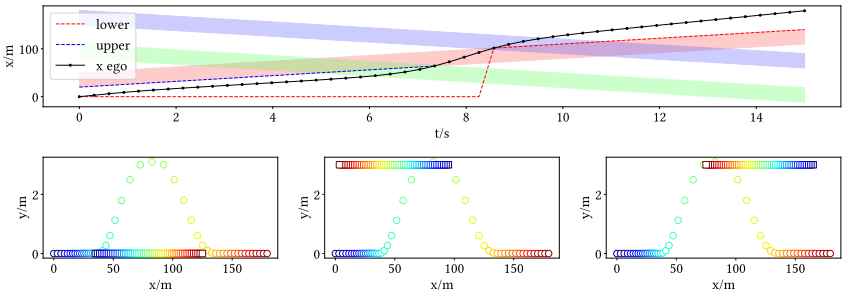


Abbildung 4.18: Variante (●, ●, ●).

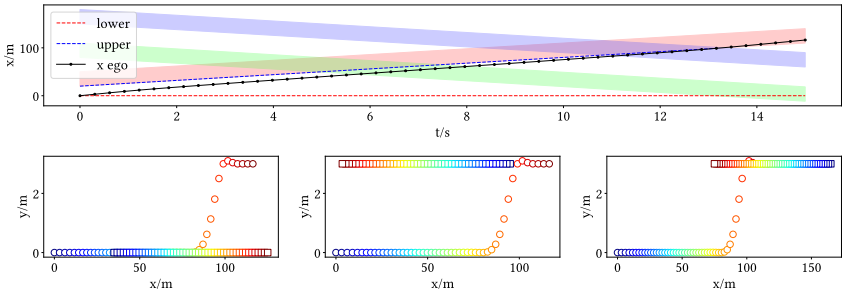


Abbildung 4.19: Variante (●, ●, ●). Dies ist die bezogen auf das gewählte Gütekriterium beste Variante.

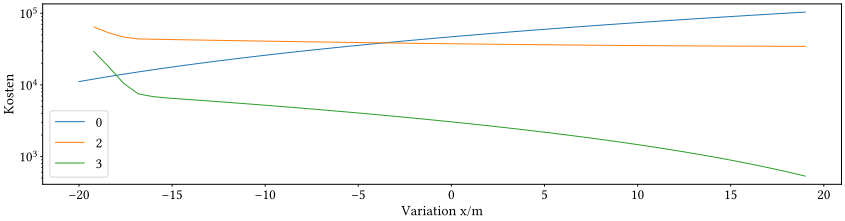


Abbildung 4.20: Variation der Hindernisentfernung. Die Varianten 0, 2 und 3 entsprechen dabei den Zeilen 1, 2 und 3 in Tabelle 4.2.

Variante	Anz. Samples	Anz. Parameter	Kosten	Strecke/m
(●, ●, ●, ●)	80	154	$212 \cdot 10^3$	241
(●, ●, ●, ●)	80	154	$64 \cdot 10^3$	224
(●, ●, ●, ●)	80	154	≈ 0	150
(●, ●, ●, ●)	80	154	$6 \cdot 10^3$	142
(●, ●, ●, ●)	80	154	$3 \cdot 10^3$	123

Tabelle 4.3: Ergebnis des kontinuierlichen Planers. Nach dem gewählten Kriterium wäre (●, ●, ●, ●) die beste Variante. Wieder dient der Fortschritt entlang des Fahrstreifens nur der Orientierung.

offen. Die Abbildungen 4.21, 4.22, 4.23, 4.24 und 4.25 zeigen die Varianten mit den zugehörigen Nebenbedingungen und Trajektorien, Tabelle 4.3 fasst die Ergebnisse zusammen.

Es wird deutlich, dass diese Art der Planung geeignet ist, um den Planungsraum zu unterteilen. Für die Trajektorie relevant sind dabei nur die Abschnitte, in denen sich Hindernisse im Weg-Zeit-Diagramm schneiden. Für die Bereiche, in denen die Trajektorie selbst ein Hindernis schneidet, ist kein besonderes Verhalten vorgesehen. Das entspricht nicht unbedingt dem menschlichen Verhalten. Ein Mensch würde möglicherweise höhere Beschleunigungen akzeptieren, um dafür Sicherheitsabstände vergrößern zu können. Dies kann bei dieser Art von Planung durch eine Beaufschlagung der Fahrzeuglängen gemindert werden, allerdings wird diese Art der Modellierung immer eine konstante Geschwindigkeit bevorzugen.

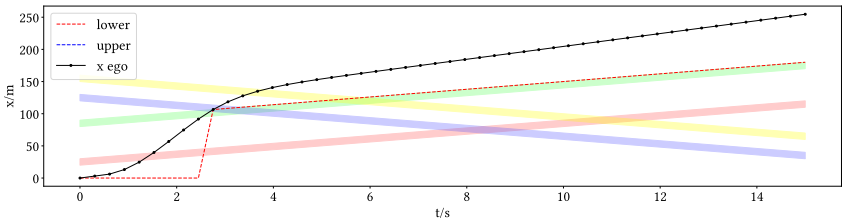


Abbildung 4.21: Variante (●, ●, ●, ●). Hier wird das Ego-Fahrzeug beide vorausfahrenden Fahrzeuge vor dem Gegenverkehr überholen. Das resultiert in hohen Kosten für Beschleunigung und die deutliche Überschreitung der Sollgeschwindigkeit.

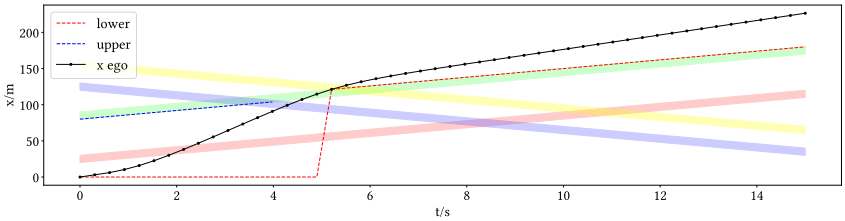


Abbildung 4.22: Variante (●, ●, ●, ●). Hier wird Fahrzeug ● überholt, der Gegenverkehr ● abgewartet und dann ● vor ● überholt.

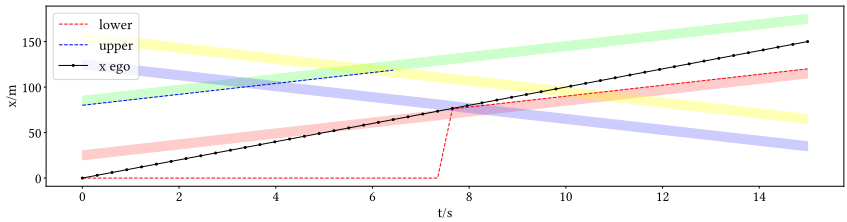


Abbildung 4.23: Variante (●, ●, ●, ●). Dies ist die nach dem gewählten Kriterium beste Variante. Die Nebenbedingung ●● passt zufällig genau zur Wunschgeschwindigkeit des Ego-Fahrzeugs, so lassen sich die geringen Kosten erklären.

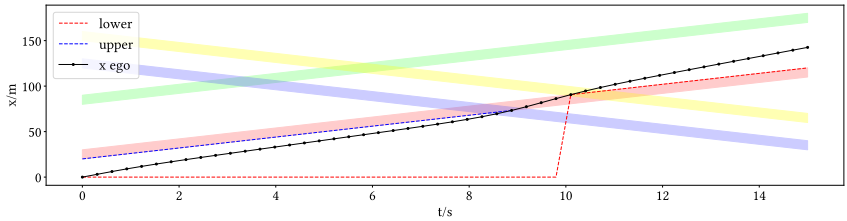


Abbildung 4.24: Variante (●, ●, ●, ●). Hier wird der Gegenverkehr ● abgewartet und dann überholt. Diese Variante ist langsamer, aber nicht günstiger.

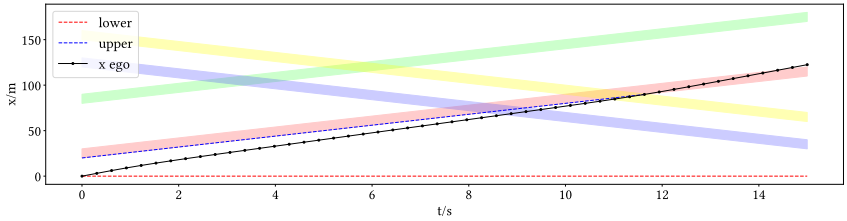


Abbildung 4.25: Variante (●, ●, ●, ●). Hier wird der Gegenverkehr komplett abgewartet, Kosten entstehen hier hauptsächlich durch das Abbremsen zu Beginn und die lange Fahrtzeit mit geringer Geschwindigkeit (bezogen auf die Sollgeschwindigkeit).

5 Zusammenfassung und Ausblick

Mit einer Kombination aus Karte, einer auf Zustandsautomaten basierten Verhaltensentscheidung und einer lokalen, schnellen, kontinuierlichen Optimierung lassen sich semantisch mächtige und im Detail präzise Fahrerassistenzsysteme (FAS) darstellen. Die in Kap. 3 vorgestellten Methoden wurden im Rahmen der Bertha-Benz-Fahrt (BBF) äußerst erfolgreich im realen Straßenverkehr erprobt.

Als besonders vorteilhaft hat sich die Einführung einer geometrischen und topologischen Karte erwiesen. Die in dieser Arbeit vorgestellte Karte und der Begriff der *Lanelets* hat in der FAS-Community Einzug erlangt, wie die Zitate des Originalpapers [BZS14], das Interesse am Quellcode[1] und darauf aufbauende Arbeiten zeigen [AUK18, PPJ+18].

Auf die BBF aufbauend wurde dargelegt, wie sich ein Bewegungsplanungsproblem in eine Menge diskreter Varianten überführen lässt und wie eine bestimmte Variante in Nebenbedingungen für ein kontinuierliches Optimierungsverfahren übersetzt werden kann.

Dabei wurde deutlich, welche praktischen Probleme eine zu allgemeine Formulierung (Kap. 4.2.1 auf Seite 53) mit sich bringt. Als Ausweg wurden verschiedene Vereinfachungen eingeführt, die schließlich in der Identifikation diskreter Fahrmanöver mündeten, die der Alltagserfahrung im Straßenverkehr entsprechen.

Die vorgestellten Methoden setzen präzises Wissen über das Umfeld voraus. Besonders die Annahmen über die Trajektorien der anderen Verkehrsteilnehmer werden als von der eigenen Bewegung unabhängig angenommen, was nur bis zu einem gewissen Grad gilt. Für die jeweils konservativsten Trajektorien wird es keinen Unterschied machen. Für ein Überholmanöver hingegen ist dies sehr

[1] `https://github.com/phbender/liblanelet`, abgerufen am 20. April 2021.

relevant, denn diese Manöver lassen sich nur sehr schlecht abbrechen und sollten nur durchgeführt werden, wenn ihre Anwendung nicht nur aktuell sicher ist, sondern auch davon ausgegangen wird, dass sich an dieser Bewertung nichts mehr ändert. Die in dieser Arbeit entwickelte Methode der Manövervariantenbetrachtung kann in dieser Richtung erweitert werden: Wie wahrscheinlich ist es, dass eine Manöver für die erforderliche Zeit *durchführbar bleibt*? In diesem Zusammenhang wird es interessant sein, eine probabilistische Vorhersage zu treffen und eventuell auch eine Reaktion der anderen Verkehrsteilnehmer auf die eigene Trajektorie zu modellieren.

Besonders das Ergebnis aus Abb. 4.23 zeigt, dass der Verzicht auf Zustand [2] auch zu eher unnatürlichem Verhalten führen kann. Ein Fahrer würde hier aus Sicherheitsgründen zügiger überholen, was aber nicht leicht im Gütekriterium unterzubringen ist, ohne die Konvergenz zu gefährden. Hier kann die vorgestellte Betrachtungsweise für eine grobe Vorplanung und Entscheidung verwendet werden. Die endgültige Trajektorie könnte dann abschnittweise geplant werden, wobei jeder Abschnitt einem bestimmten Zustand im Manöver entspricht und unterschiedlich mit Planungsparametern (Sollgeschwindigkeit, Anzahl Samples, Gewichte) konfiguriert werden kann.

Ebenfalls vielversprechend könnte es sein, die vorgestellte Manövervariantenbetrachtung mit Arbitrationsgraphen (2.2.3 auf Seite 17) zu kombinieren. Im Schema der Verhaltensbausteine ist es nicht möglich, eine Manövervariante direkt abzubilden, da eine Manövervariante im Sinne dieser Arbeit vielmehr einer Abfolge von Verhaltensbausteinen entspricht. Ein Arbitrator hingegen könnte die vorgestellte Manöveranalyse durchführen und dann den Verhaltensbaustein auswählen, welcher die günstigste Variante initiiert.

[2] Zustand im Sinne von: „Ich befinde mich mitten in einem Überholmanöver (und fahre entsprechend zügig)“

Literaturverzeichnis

[ASA20] ASAM e.V.: *Open Curved Regular Grid , Version 1.2.0*. Hohenkirchen: https://www.asam.net/standards/detail/opencrg/, 09 2020. – Autoren nicht namentlich genannt; Online; Zugriff 20. April 2021

[ASA21] ASAM e.V.: *Open Dynamic Road Information for Vehicle Environment , Version 1.6.1*. Hohenkirchen: https://www.asam.net/standards/detail/opendrive/, 03 2021. – Autoren nicht namentlich genannt; Online; Zugriff 20. April 2021

[AUK18] Althoff, Matthias; Urban, Stefan; Koschi, Markus: Automatic conversion of road networks from opendrive to lanelets. In: *Proceedings of the IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*. Singapore, 2018, S. 157–162

[BDTD+16] Bojarski, Mariusz; Del Testa, Davide; Dworakowski, Daniel; Firner, Bernhard; Flepp, Beat; Goyal, Prasoon; Jackel, Lawrence D.; Monfort, Mathew; Muller, Urs; Zhang, Jiakai u. a.: *End to end learning for self-driving cars*. arXiv:1604.07316, https://arxiv.org/abs/1604.07316, 04 2016. – Ausschließlich online; Zugriff 20. April 2021

[BGU13] Brubaker, Marcus A.; Geiger, Andreas; Urtasun, Raquel: Lost! leveraging the crowd for probabilistic visual self-localization. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Portland, OR, USA, 2013, S. 3057–3064

[BIS09] Buehler, Martin; Iagnemma, Karl; Singh, Sanjiv: *The DARPA urban challenge: autonomous vehicles in city traffic*. Bd. 56. Berlin: Springer, 2009

[BK73] Bron, Coen; Kerbosch, Joep: Algorithm 457: Finding all Cliques of an Undirected Graph. In: *Communications of the ACM* 16 (1973), Nr. 9, S. 575–577

[BKL10] Bhattacharya, Subhrajit; Kumar, Vijay; Likhachev, Maxim: Search-based path planning with homotopy class constraints. In: *Proceedings of the AAAI Conference on Artificial Intelligence* Bd. 24. Atlanta, GA, USA, 2010, S. 1230–1239

[BKSS90] Beckmann, Norbert; Kriegel, Hans-Peter; Schneider, Ralf; Seeger, Bernhard: The R*-tree: an efficient and robust access method for points and rectangles. In: *Proceedings of ACM SIGMOD International Conference on Management of Data* Bd. 19. Atlantic City, NJ, USA, 1990, S. 322–331

[BLK11] Bhattacharya, Subhrajit; Likhachev, Maxim; Kumar, Vijay: Identification and representation of homotopy classes of trajectories for search-based path planning in 3d. In: *Robotics – Science and Systems Online Proceedings* (2011), S. 9–16

[Bro86] Brooks, Rodney: A robust layered control system for a mobile robot. In: *Journal on Robotics and Automation* 2 (1986), Nr. 1, S. 14–23

[BS15] Bender, Philipp; Stiller, Christoph: Trajektorienplanung: Manöveridentifikation anhand der Topologie des Freiraums. In: *10. Workshop Fahrerassistenz-Systeme.* Walting, 2015, S. 51–60

[BTM10] Bétaille, David; Toledo-Moreo, Rafael: Creating enhanced maps for lane-level vehicle navigation. In: *IEEE Transactions on Intelligent Transportation Systems* 11 (2010), Nr. 4, S. 786–798

[BTS15] Bender, Philipp; Taş, Şahin; Stiller, Christoph: The Combinatorial Aspect of Motion Planning: Maneuver Variants in Structured Environments. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV).* Seoul, Südkorea, 2015, S. 1386–1392

[BZS14] Bender, Philipp; Ziegler, Julius; Stiller, Christoph: Lanelets: Efficient map representation for autonomous driving. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV).* Dearborn, MI, USA, 2014, S. 420–425

[DLB+15] Dang, Thao; Lauer, Martin; Bender, Philipp; Schreiber, Markus; Ziegler, Julius; Franke, Uwe; Fritz, Hans; Strauß, Tobias; Lategahn, Henning; Keller, Christoph G. u. a.: Autonomes Fahren auf der historischen Bertha-Benz-Route. In: *tm - Technisches Messen* 82 (2015), Nr. 5, S. 280–297

[DM92] Dickmanns, Ernst D.; Mysliwetz, Birger D.: Recursive 3-d road and relative ego-state recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (1992), Nr. 2, S. 199–213

[DPRW13] Damm, Werner; Peter, Hans-Jörg; Rakow, Jan; Westphal, Bernd: Can we build it: formal synthesis of control strategies for cooperative driver assistance systems. In: *Mathematical Structures in Computer Science* 23 (2013), Nr. 04, S. 676–725

[FL93] Fraichard, Thierry; Laugier, Christian: Path-velocity decomposition revisited and applied to dynamic trajectory planning. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).* Atlanta, GA, USA, 1993, S. 40–45

[FSR05] Fuchshumer, Stefan; Schlacher, Kurt; Rittenschober, Thomas: Nonlinear vehicle dynamics control-a flatness based approach. In: *Proceedings of the IEEE Conference on Decision and Control.* Nassau, Bahamas, 2005, S. 6492–6497

[FTO+08] Fletcher, Luke; Teller, Seth; Olson, Edwin; Moore, David; Kuwata, Yoshiaki; How, Jonathan; Leonard, John; Miller, Isaac; Campbell, Mark; Huttenlocher, Dan u. a.: The MIT–Cornell collision and why it happened. In: *Journal of Field Robotics* 25 (2008), Nr. 10, S. 775–807

[GLM+12] Geiger, Andreas; Lauer, Martin; Moosmann, Frank; Ranft, Benjamin; Rapp, Holger; Stiller, Christoph; Ziegler, Julius: Team

AnnieWAY's entry to the 2011 Grand Cooperative Driving challenge. In: *IEEE Transactions on Intelligent Transportation Systems* 13 (2012), Nr. 3, S. 1008–1017

[GMW81] Gill, Philip E.; Murray, Walter; Wright, Margaret H.: *Practical optimization.* London, Vereinigts Königreich: Academic Press Inc., 1981

[GPGW16] Gutjahr, Benjamin; Pek, Christian; Gröll, Lutz ; Werling, Moritz: Recheneffiziente Trajektorienoptimierung für Fahrzeuge mittels quadratischem Programm. In: *at - Automatisierungstechnik* 64 (2016), Nr. 10, S. 786–794

[Gut84] Guttman, Antonin: R-trees: A dynamic index structure for spatial searching. In: *Proceedings of ACM SIGMOD International Conference on Management of Data.* Boston, MA, USA, 1984, S. 47–57

[GW12] Gasser, Tom M.; Westhoff, Daniel: *BASt-study: Definitions of automation and legal issues in Germany.* Proceedings of the Road Vehicle Automation Workshop, online, http://onlinepubs.trb.org/onlinepubs/conferences/2012/Automation/presentations/Gasser.pdf, 2012. – Präsentationsfolien, Ausschließlich online verfügbar, Abrufdatum: 20. April 2021

[Har87] Harel, David: Statecharts: A visual formalism for complex systems. In: *Science of Computer Programming* 8 (1987), Nr. 3, S. 231–274

[Hat02] Hatcher, Allen: *Algebraic topology.* New York, NY, USA: Cambridge University Press, 2002. – ISBN 9780521795401

[HBD+17] Hubschneider, Christian; Bauer, Andre; Doll, Jens; Weber, Michael; Klemm, Sebastian; Kuhnt, Florian; Zöllner, J M.: Integrating end-to-end learned steering into probabilistic autonomous driving. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC).* Yokohama, Japan, 2017, S. 1–7

[HBWZ17] Hubschneider, Christian; Bauer, Andre; Weber, Michael; Zöllner, J M.: Adding navigation to the equation: Turning decisions for end-to-end vehicle control. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Yokohama, Japan, 2017, S. 308–315

[HHS12] Hasberg, Carsten; Hensel, Stefan; Stiller, Christoph: Simultaneous localization and mapping for path-constrained motion. In: *IEEE Transactions on Intelligent Transportation Systems* 13 (2012), Nr. 2, S. 541–552

[HLLR08] Hafner, Roland; Lange, Sascha; Lauer, Martin; Riedmiller, Martin: Brainstormers tribots team description. In: *RoboCup International Symposium*. Suzhou, China, 2008, S. n. v.

[HWB+13] Hornung, Armin; Wurm, Kai M.; Bennewitz, Maren; Stachniss, Cyrill; Burgard, Wolfram: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. In: *Autonomous Robots* 34 (2013), Nr. 3, S. 189–206

[JCB97] Johnson, Paul J.; Chapman, Kevin L.; Bay, John S.: Navigation of an autonomous ground vehicle using the subsumption architecture. In: *Mobile Robots XI and Automated Vehicle Control Systems* Bd. 2903. Boston, MA, USA, 1997, S. 54–62

[JH12] Johnson, Jeff; Hauser, Kris: Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Saint Paul, MN, USA, 2012, S. 2035–2041

[JH13] Johnson, Jeff; Hauser, Kris: Optimal longitudinal control planning with moving obstacles. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Gold Coast, Australien, 2013, S. 605–611

[KCK96] Kang, Dong J.; Choi, Jang W.; Kweon, In S.: Finding and tracking road lanes using "line-snakes". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Tokyo, Japan, 1996, S. 189–194

[Ken11] Kennedy, James: Particle swarm optimization. In: *Encyclopedia of machine learning*. New York, NY, USA: Springer Science + Business Media, 2011, S. 760–766

[KFT+08] Kuwata, Yoshiaki; Fiore, Gaston A.; Teo, Justin; Frazzoli, Emilio; How, Jonathan P.: Motion planning for urban driving using RRT. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nizza, Frankreich, 2008, S. 1681–1686

[Kra94] Kraft, Dieter: Algorithm 733: TOMP–Fortran modules for optimal control calculations. In: *ACM Transactions on Mathematical Software (TOMS)* 20 (1994), Nr. 3, S. 262–281

[KZ86] Kant, Kamal; Zucker, Steven W.: Toward efficient trajectory planning: The path-velocity decomposition. In: *The International Journal of Robotics Research* 5 (1986), Nr. 3, S. 72–89

[Lau11] Lauer, Martin: A case study on learning a steering controller from scratch with reinforcement learning. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Baden Baden, 2011, S. 260–265

[LaV06] LaValle, Steven M.: *Planning algorithms*. New York, NY, USA: Cambridge University Press, 2006

[LKC94] Lee, Ta-Chih; Kashyap, Rangasami L.; Chu, Chong-Nam: Building skeleton models via 3-D medial surface axis thinning algorithms. In: *CVGIP: Graphical Models and Image Processing* 56 (1994), Nr. 6, S. 462–478

[LKJ00] LaValle, Steven M.; Kuffner Jr, James J.: Rapidly-exploring random trees: Progress and prospects. In: *Algorithmic and Computational Robotics: New Directions*. Boca Raton, FL, USA: CRC Press, 2000, S. 293–308

[Mom01] Momjian, Bruce: *PostgreSQL: introduction and concepts*. Bd. 192. New York City, NY, USA: Addison-Wesley, 2001

[MPW+18] Mirchevska, Branka; Pek, Christian; Werling, Moritz; Althoff, Matthias; Boedecker, Joschka: High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC).* Maui, HI, USA, 2018, S. 2156–2162

[MRDJ06] Malik, Waqar; Rathinam, Sivakumar; Darbha, Swaroop ; Jeffcoat, David: Combinatorial motion planning of multiple vehicle systems. In: *Proceedings of the IEEE Conference on Decision and Control.* San Diego, CA, USA, 2006, S. 5299–5304

[NGGP08] Naranjo, José E.; Gonzalez, Carlos; Garcia, Ricardo; Pedro, Teresa de: Lane-change fuzzy control in autonomous vehicles for the overtaking maneuver. In: *IEEE Transactions on Intelligent Transportation Systems* 9 (2008), Nr. 3, S. 438–450

[NIM+13] Nasir, Jauwairia; Islam, Fahad; Malik, Usman; Ayaz, Yasar; Hasan, Osman; Khan, Mushtaq; Muhammad, Mannan S.: RRT*-SMART: A rapid convergence implementation of RRT. In: *International Journal of Advanced Robotic Systems* 10 (2013), Nr. 7, S. 299

[OBL20] Orzechowski, Piotr F.; Burger, Christoph; Lauer, Martin: Decision-making for automated vehicles using a hierarchical behavior-based arbitration scheme. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV).* Las Vegas, CA, USA, 2020, S. 767–774

[OBLS21] Orzechowski, Piotr F.; Burger, Christoph; Lauer, Martin; Stiller, Christoph: Verhaltensentscheidung für automatisierte Fahrzeuge mittels Arbitrationsgraphen. In: *at - Automatisierungstechnik* 69 (2021), Nr. 2, S. 171–181

[OH15] Obe, Regina O.; Hsu, Leo S.: *PostGIS in action.* Shelter Islands, NY, USA: Manning Publications Co., 2015

[PKB07] Poli, Riccardo; Kennedy, James; Blackwell, Tim: Particle swarm optimization. In: *Swarm intelligence* 1 (2007), Nr. 1, S. 33–57

[PPJ+18] Poggenhans, Fabian; Pauls, Jan-Hendrik; Janosovits, Johannes; Orf, Stefan; Naumann, Maximilian; Kuhnt, Florian; Mayr, Matthias: Lanelet2: A high-definition map framework for the future of automated driving. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC).* Maui, HI, USA, 2018, S. 1672–1679

[Pud98] Pudney, Chris: Distance-ordered homotopic thinning: a skeletonization algorithm for 3D digital images. In: *Computer Vision and Image Understanding* 72 (1998), Nr. 3, S. 404–413

[RA15] Rizaldi, Albert; Althoff, Matthias: Formalising traffic rules for accountability of autonomous vehicles. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC).* Las Palmas de Gran Canaria, Spanien, 2015, S. 1658–1665

[Raw00] Rawlings, James B.: Tutorial overview of model predictive control. In: *IEEE Control Systems Magazine* 20 (2000), Nr. 3, S. 38–52

[RT10] Ramm, Frederik; Topf, Jochen: *OpenStreetMap: Die freie Weltkarte nutzen und mitgestalten.* Köln: Lehmanns Media, 2010

[WDA12] Wang, Yifei; Dahnoun, Naim; Achim, Alin: A novel system for robust lane detection and tracking. In: *Signal Processing* 92 (2012), Nr. 2, S. 319–334

[ZBDS14] Ziegler, Julius; Bender, Philipp; Dang, Thao; Stiller, Christoph: Trajectory planning for Bertha — A local, continuous method. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV).* Dearborn, MI, USA, 2014, S. 450–457

[ZBL+14] Ziegler, Julius; Bender, Philipp; Lategahn, Henning; Schreiber, Markus; Strauss, Tobias; Stiller, Christoph: Kartengestütztes automatisiertes Fahren auf der Bertha-Benz-Route von Mannheim

nach Pforzheim. In: *9. Workshop Fahrerassistenz-Systeme* Bd. 496. Walting, 2014, S. 79–93

[ZBS+14] Ziegler, Julius; Bender, Philipp; Schreiber, Markus; Lategahn, Henning; Strauss, Tobias; Stiller, Christoph; Dang, Thao; Franke, Uwe; Appenrodt, Nils; Keller, Christoph G. u. a.: Making Bertha drive—An autonomous journey on a historic route. In: *IEEE Intelligent Transportation Systems Magazine* 6 (2014), Nr. 2, S. 8–20

[Zie15] Ziegler, Julius: *Optimale Bahn- und Trajektorienplanung für Automobile*. Karlsruhe: KIT Scientific Publishing, 2015. – Dissertation

[ZS09] Ziegler, Julius; Stiller, Christoph: Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, MO, USA, 2009, S. 1879–1884