Erstellung eines sensorbasierten Straßenmodells für das automatisierte Fahren

Dissertation

zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)

am

Fachbereich Mathematik und Informatik
Freie Universität Berlin

vorgelegt von

Julian Thomas

Berlin 2020

Betreuer und Erstgutachter:

Prof. Dr. Raúl Rojas
Institut für Informatik
Fachbereich Mathematik und Informatik
Freie Universität Berlin
Arnimallee 7, 14195 Berlin
raul.rojas@fu-berlin.de

Zweitgutachter:

Prof. Dr. Daniel Watzenig
Institut für Regelungs- und Automatisierungstechnik
Fakultät für Elektrotechnik und Informationstechnik
Technische Universität Graz
Inffeldgasse 21, 8010 Graz
daniel.watzenig@tugraz.at

Datum der Disputation: 26. März 2021

Zusammenfassung

Das Wissen über die unmittelbare Umgebung ist von größter Wichtigkeit für sämtliche Anwendungen in der Robotik und des autonomen Fahrens. Für das autonome Fahren werden diese Informationen stets aus hochgenauen Karten extrahiert, die den Einsatz einer hochpräzisen Lokalisierung voraussetzen. Damit geht die Beschränkung einher, dass das System nur in denjenigen Arealen verwendet werden kann, die zuvor kartiert worden sind. Darüber hinaus sorgen nachträgliche Änderungen der Umgebung dafür, dass die hochgenaue Karte anschließend nicht mehr mit der Realität übereinstimmt und damit ihre Gültigkeit verliert.

Daher widmet sich die vorliegende Arbeit der Problemstellung, wie die unmittelbare Umgebung eines autonom fahrenden Fahrzeugs ohne Zuhilfenahme einer hochgenauen Karte, sondern ausschließlich mit Hilfe von im Fahrzeug verbauter Sensorik erfasst und repräsentiert werden kann. Ausgehend von der „Dempster-Shafer theory of evidence" wird eine neue Methode vorgestellt, um Sensor-Messdaten wie bspw. erkannte Fahrstreifenmarkierungslinien, Punktewolken von einer bildbasierten semantischen Segmentierung, Belegungskarten und andere Verkehrsteilnehmer in einem gridbasierten Umfeldmodell miteinander zu fusionieren und dabei die *semantische* Bedeutung jeder Gridzelle zu schätzen. Anschließend werden mit Hilfe einer Pfadplanung aus dem gridbasierten Modell potentielle Fahrstreifen extrahiert und in ein geeignetes geometrisches Straßenmodell überführt. Alle entwickelten Methoden wurden mit Sensor-Messdaten aus realen Verkehrssituationen evaluiert und die Leistungsfähigkeit der vorgestellten Lösungen exemplarisch dargestellt.

Im Rahmen der Arbeit entstanden unter anderem 4 Publikationen (Erstautorenschaft), in denen die Forschungsergebnisse veröffentlicht wurden [1, 2, 3, 4].

Danksagung

Mein besonderer Dank gilt Herrn Professor Dr. Raúl Rojas von der Freien Universität Berlin für die Betreuung der Arbeit, die fachlichen Diskussionen und inhaltlichen Anregungen, die zu dieser Arbeit geführt haben.

Bedanken möchte ich mich auch bei Herrn Professor Dr. Daniel Watzenig für die Erstellung des Zweitgutachtens.

Weiterer Dank gilt Herrn Professor Dr. Werner Huber für die Ermöglichung dieser Arbeit im Forschungsprojekt zum automatisierten Fahren bei der BMW AG und die große Unterstützung während und nach meiner Doktorandenzeit.

Darüber hinaus möchte ich mich besonders bei Julian Tatsch, Christian Panhuber und Daniel Althoff für die vielen fachlichen und nicht-fachlichen Diskussionen sowie die gemeinsamen Aktivitäten bedanken.

Weiterer Dank gilt Bernhard Seidl, Alexander Schulz und Thomas Kühbeck für ihre Unterstützung und die gemeinsame Zeit bei der BMW AG.

Mein größter Dank gilt meiner Familie, die es mir immer ermöglicht hat, meine Chancen zu nutzen, und mich zu jeder Zeit unterstützt hat.

Inhaltsverzeichnis

Erklärung

Name: Thomas
Vorname: Julian

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

München, 22. September 2020

(Julian Thomas)

Kapitel 1

Einleitung

Im Zuge der Weiterentwicklung heutiger Kraftfahrzeuge entsteht ein immer stärker werdendes Bedürfnis nach Sicherheit, Assistenz und Automatisierung.

Der Schwerpunkt des Fortschritts und der Serieneinführung im 20. Jahrhundert lag hierbei auf den Systemen und Komponenten zur *passiven Sicherheit*. Diese schützen die Fahrzeuginsassen im Fall eines Unfalls und mindern die Unfallfolgen. Bekannteste Beispiele für passive Sicherheitssysteme sind Dreipunkt-Anschnallgurt, Airbags, oder auch die Konstruktion der Fahrzeugkabine (Festigkeit der Kabine zum Schutz der Insassen, Seitenaufprallschutz).

Bereits ab Mitte des 20. Jahrhunderts wurden parallel zu den passiven Sicherheitssystemen Funktionen entwickelt, die durch ein aktives Eingreifen in das Fahrgeschehen Unfallfolgen abmildern oder im besten Fall auch vollständig verhindern können.

Das bekannteste System dieser sogenannten *aktiven Sicherheitssysteme* ist das Anti-Blockier-System ABS. Es verhindert beim Bremsen ein mögliches Blockieren der Räder durch die Reduzierung des Bremsdrucks. Hierdurch wird gewährleistet, dass das Fahrzeug während des Bremsvorgangs lenkbar und in der Spur bleibt.

Eine logische Weiterentwicklung der Funktionalität des ABS führte 1995 zur Einführung des elektronischen Stabilitätsprogramms ESP (auch Dynamische-Stabilitätskontrolle DSC). Es verhindert in kritischen Fahrsituationen durch ein Eingreifen in das Fahrgeschehen (gezieltes Abbremsen einzelner Räder) das Über- oder Untersteuern des Fahrzeugs.

Diverse aktive Sicherheitssysteme, wie das Anti-Blockier-System ABS oder das elektronische Stabilitätsprogramm ESP gehören heutzutage bei fast allen Fahrzeugen zur Serienausstattung oder sind bereits durch gesetzliche Vorschriften Pflichtbestandteil jedes Fahrzeugs.

Durch die aktiven Sicherheitssysteme verschwimmt zunehmend die Grenze zwischen Sicherheitssystemen, Assistenzsystemen und Automatisierungsfunktionen. Bereits das ESP stellt in gewisser Weise eine Art Automatisierung dar, weil der Computer die Kontrolle über Teile des Fahrzeugs übernimmt und die Aktionen des Fahrers bewusst ignoriert.

Im weiteren Verlauf der Entwicklung wurden neben den Sicherheitssystemen weitere Funktionen eingeführt, die auch als Assistenz- und Komfortfunktion gedacht sind. Diese greifen nicht nur in gefährlichen Situationen ein, sondern unterstützen den Fahrer auch während der normalen Fahrt. Bekanntestes Beispiel für ein solches System ist das *Adaptive Cruise Control* ACC, welches einen drohenden Aufprall auf ein vorausfahrendes Fahrzeug durch Abbremsen verhindert, aber gleichzeitig automatisch durch die Regulierung von Geschwindigkeit und Bremse einen konstanten Abstand zum Vorderfahrzeug hält.

Die Entwicklung der Sicherheitssysteme sowie der Assistenzsysteme bewegt sich stetig weiter in Richtung einer vollständigen Automatisierung. Der folgende Abschnitt gibt einen Überblick über die Thematik.

1.1 Fahrerassistenz und Automatisierung

Kompaß schreibt in [5], dass sich die Fahraufgabe in die drei Ebenen Stabilisierung, Führung und Planung aufteilen lässt. Dies kann nun als Grundlage für die Gruppierung der Fahrerassistenzsysteme benutzt werden, indem die Systeme in der Ebene eingeordnet werden, in der sie hauptsächlich aktiv sind.

Die eingangs erwähnten Systeme ABS und ESP, die in fahrdynamischen Grenzsituationen eingreifen, zählen zur Stabilisierungsebene. Da der Zeitrahmen, in dem mögliche Aktionen ablaufen, in der Stabilisierungsebene im Bereich von Millisekunden liegt, übertreffen diese Systeme meist die Handlungsfähigkeiten des Fahrers. Ebenfalls zur Stabilisierungsebene zählen weitere Fahrdynamik-Regelsysteme wie beispielsweise die dynamische Antriebsmomentenverteilung bei Allrad-Fahrzeugen oder die Anti-Schlupf-Regelung sowie deren Abwandlungen.

Zur Führungsebene gehören die unzähligen bereits auf dem Markt erhältlichen Assistenzsysteme. Aufgrund der Fülle der Systeme ist eine Unterteilung in aktive und passive Systeme hier sinnvoll. Die aktiven Systeme greifen - gleichsam wie bei eingangs beschriebenen Sicherheitssystemen - in das Fahrgeschehen ein, während die passiven dem Fahrer nur Warnungen oder Informationen anbieten. Beispiele für passive Systeme sind die Park-Distance-Control (PDC), die Spurverlassenswarnung und die Tempolimit- bzw. Verkehrszeichenanzeige. Zu den aktiven Systemen zählt das Adaptive-Cruise-Control, die Gefahrenbremsung

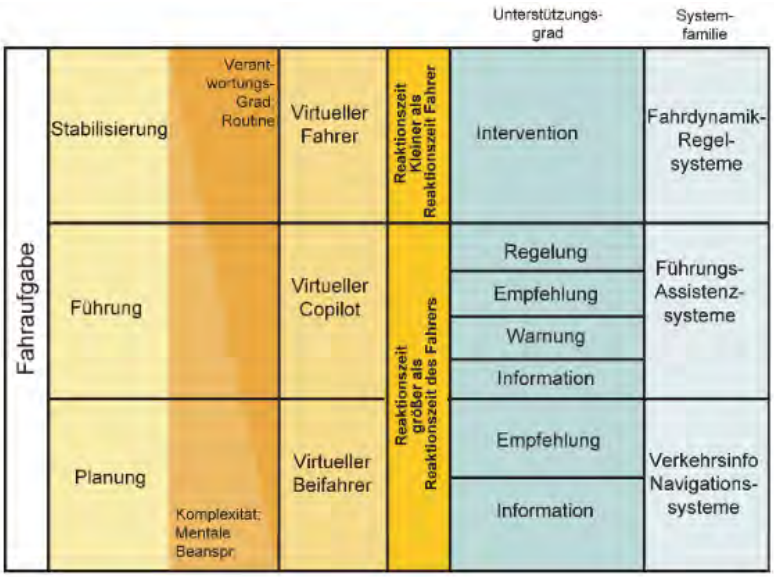


Abbildung 1.1: 3-Ebenen-Modell nach Kompaß [5]. Die Fahraufgabe wird unterteilt in Stabilisierung, Führung und Planung.

und sämtliche Systeme zum Halten der Spur sowie die Einparkfunktionalität und Spurwechselassistenten. Zur Planungsebene gehören zum Beispiel Navigationssysteme. Sie stellen dem Fahrer Informationen bereit, und unterbreiten ggf. Empfehlungen. Eine gute Übersicht über Assistenzsysteme ist in [6] zu finden.

Mit zunehmendem Fortschritt in der Sensor-Technologie und sinkenden Preisen von Rechenkapazitäten in eingebetteten Systemen, entwickelt sich aus der Fahrerassistenz mehr und mehr eine Fahrautomatisierung, bei der der Fahrer die Fahraufgabe vollständig an das System abgibt.

Neben den technischen Hürden stehen einer flächendeckenden Einführung allerdings auch rechtliche Hindernisse entgegen, da in vielen Ländern der Fahrer in jeder Situation verpflichtet ist, sein Fahrzeug vollständig zu kontrollieren. Eine entsprechende Bedingung ist im Wiener Übereinkommen über den Straßenverkehr [7] zu finden, welches in vielen Ländern als Gesetzesgrundlage für den Straßenverkehr dient.

In einer Überarbeitung des Wiener Übereinkommens im Jahre 2014 [8] ist dem Umstand der zunehmenden Automatisierung Rechnung getragen worden. Demnach sind nun auch Systeme zulässig, die die Fahraufgabe vollständig übernehmen, sofern der Fahrer jederzeit die Möglichkeit hat, die Systeme zu überstimmen oder zu deaktivieren.

Durch die Änderung des Abkommens wird Rechtssicherheit bezüglich der heutigen Assistenzsysteme geschaffen. Allerdings trägt der Umstand, dass nach wie vor der Fahrer die alleinige Verantwortung trägt, nicht dazu bei, die rechtlichen Bedingungen für vollständig automatisierte Fahrzeuge abschließend zu klären. Dadurch, dass der Fahrer die Verantwortung trägt, ist er in der Pflicht, das System permanent zu überwachen. Dies steht im Gegensatz zu wesentlichen Zielen des automatisierten Fahrens.

In Deutschland hat die *Bundesanstalt für Straßenwesen* (BASt) eine Projektgruppe ins Leben gerufen, um die rechtlichen Rahmenbedingungen und Konsequenzen der zunehmenden Automatisierung bei Fahrzeugen zu untersuchen [9]. Für die rechtliche Bewertung wurde in der Projektgruppe eine Nomenklatur und Definition von drei Automatisierungsgraden entwickelt: Teil-, Hoch- und Vollautomatisierung (siehe Abb. 1.2).

Benennung und Klassifizierung automatisierter Fahrfunktionen
(nicht abschließend)
Stand: 06.09.2010

Nomenklatur	Beschreibung Automatisierungsgrad und Erwartung des Fahrers	beispielhafte Systemausprägung
Driver Only	Fahrer führt dauerhaft (während der gesamten Fahrt) die Längsführung (Beschleunigen / Verzögern) und die Querführung (Lenken) aus.	Kein in die Längs- oder Querführung eingreifendes (Fahrerassistenz-)System aktiv
Assistiert	Fahrer führt dauerhaft entweder die Quer- oder die Längsführung aus. Die jeweils andere Fahraufgabe wird in gewissen Grenzen vom System ausgeführt. •Der Fahrer muss das System dauerhaft überwachen •Der Fahrer muss jederzeit zur vollständigen Übernahme der Fahrzeugführung bereit sein	Adaptive Cruise Control: -Längsführung mit adaptiver Abstands- und Geschwindigkeitsregelung Parkassistent: - Querführung durch Parkassistent (Automatisches Lenken in Parklücken. Der Fahrer steuert die Längsführung.)
Teilautomatisiert	Das System übernimmt Quer- und Längsführung (für einen gewissen Zeitraum und/ oder in spezifischen Situationen). • Der Fahrer muss das System dauerhaft überwachen • Der Fahrer muss jederzeit zur vollständigen Übernahme der Fahrzeugführung bereit sein	Autobahnassistent: -Automatische Längs- und Querführung - Auf Autobahnen bis zu einer oberen Geschwindigkeitsgrenze -Fahrer muss dauerhaft überwachen und bei Übernahmeaufforderung sofort reagieren
Hochautomatisiert	Das System übernimmt Quer- und Längsführung für einen gewissen Zeitraum in spezifischen Situationen. •Der Fahrer muss das System dabei nicht dauerhaft überwachen •Bei Bedarf wird der Fahrer zur Übernahme der Fahraufgabe mit ausreichender Zeitreserve aufgefordert •Systemgrenzen werden alle vom System erkannt. Das System ist nicht in der Lage, aus jeder Ausgangssituation den risikominimalen Zustand herbeizuführen.	Autobahn-Chauffeur: -Automatische Längs- und Querführung -Auf Autobahnen bis zu einer oberen Geschwindigkeitsgrenze -Fahrer muss nicht dauerhaft überwachen und nach Übernahmeaufforderung mit gewisser Zeitreserve reagieren
Vollautomatisiert	Das System übernimmt Quer- und Längsführung vollständig in einem definierten Anwendungsfall. •Der Fahrer muss das System dabei nicht überwachen •Vor dem Verlassen des Anwendungsfalles fordert das System den Fahrer mit ausreichender Zeitreserve zur Übernahme der Fahraufgabe auf •Erfolgt dies nicht, wird in den risikominimalen Systemzustand zurückgeführt •Systemgrenzen werden alle vom System erkannt, das System ist in allen Situationen in der Lage, in den risikominimalen Systemzustand zurückzuführen	Autobahnpilot: - Automatische Längs- und Querführung - Auf Autobahnen bis zu einer oberen Geschwindigkeitsgrenze - Fahrer muss nicht überwachen - Reagiert der Fahrer nicht auf eine Übernahmeaufforderung, so bremst das Fahrzeug in den Stillstand herunter.

Abbildung 1.2: Automatisierungsstufen nach BASt [9].

In den Vereinigten Staaten haben die ersten Bundesstaaten die rechtlichen Rahmenbedingungen für das Testen und die Zulassung von automatisierten Fahrzeugen geschaffen - darunter Nevada [10], Kalifornien [11], Florida [12], Michigan [13] und Pennsylvania [14]. Im Jahr 2014 (überarbeitete Version 2016), hat die *Society of Automotive Engineers (SAE)* bzw. die Projektgruppe *On-Road Automated Driving (ORAD) committee* den Standard SAE J 3016 [15] herausgegeben,

der die verschiedenen Automatisierungslevel definiert. Im Vergleich zur Definition der BASt ist die Definition der SAE detaillierter in Bezug auf die Automatisierungsstufen, unterscheidet sich aber nicht wesentlich von denen der BASt. Der SAE J 3016 Standard (siehe 1.3) beschreibt darüber hinaus auch, wer (Fahrer oder Fahrzeug) in welchem Automatisierungslevel verantwortlich ist für die Fahraufgabe.

Level	Name	Narrative definition	DDT		DDT fallback	ODD
			Sustained lateral and longitudinal vehicle motion control	OEDR		
Driver performs part or all of the *DDT*						
0	No Driving Automation	The performance by the *driver* of the entire *DDT*, even when enhanced by *active safety systems*.	*Driver*	*Driver*	*Driver*	n/a
1	Driver Assistance	The *sustained* and *ODD*-specific execution by a *driving automation system* of either the *lateral* or the *longitudinal vehicle motion control* subtask of the DDT (but not both simultaneously) with the expectation that the *driver* performs the remainder of the *DDT*.	*Driver* and *System*	*Driver*	*Driver*	Limited
2	Partial Driving Automation	The *sustained* and *ODD*-specific execution by a *driving automation system* of both the *lateral* and *longitudinal vehicle motion control* subtasks of the *DDT* with the expectation that the *driver* completes the *OEDR* subtask and *supervises* the *driving automation system*.	**System**	*Driver*	*Driver*	Limited
ADS ("*System*") performs the entire *DDT* (while engaged)			*System*	**System**	*Fallback-ready user (becomes the driver during fallback)*	Limited
3	Conditional Driving Automation	The *sustained* and *ODD*-specific performance by an *ADS* of the entire DDT with the expectation that the *DDT fallback-ready user* is *receptive* to *ADS*-issued *requests to intervene*, as well as to *DDT performance-relevant system failures* in other *vehicle* systems, and will respond appropriately.				
4	High Driving Automation	The *sustained* and *ODD*-specific performance by an *ADS* of the entire *DDT* and *DDT fallback* without any expectation that a *user* will respond to a *request to intervene*.	*System*	*System*	**System**	Limited
5	Full Driving Automation	The *sustained* and unconditional (i.e., not *ODD*-specific) performance by an *ADS* of the entire *DDT* and *DDT fallback* without any expectation that a *user* will respond to a *request to intervene*.	*System*	*System*	*System*	**Unlimited**

Abbildung 1.3: Automatisierungsstufen in SAE J 3016 [15]. (ADS: Automated Driving System, ODD: operational design domain (Einsatzgebiet, für das das System konstruiert wurde), DDT: Dynamic driving task, OEDR: Object and event detection and response.

Durch die größere internationale Akzeptanz des SAE J 3016 Standards gegenüber der BASt-Definition, wird in der vorliegenden Arbeit im Folgenden nur noch der SAE Standard verwendet.

Im folgenden Abschnitt werden die wichtigsten Projekte rund um das automatisierte Fahren als Übersicht über den Stand der Technik des automatisierten Fahrens vorgestellt.

Automatisiertes Fahren

Das Forschungsfahrzeug VaMoRs (Versuchsfahrzeug für autonome Mobilität und Rechnersehen) war sicherlich einer der ersten Meilensteine bei der Entwicklung des automatisierten Fahrens. Bereits 1987 baute ein Forschungsteam der Universität der Bundeswehr in München um Prof. Ernst Dickmanns diesen Mercedes-Benz-Transporter so um, dass das Fahrzeug automatisiert auf einer abgesperrten Autobahn fahren konnte (siehe Abb. 1.4).

(a) **(b)**

Abbildung 1.4: Versuchsfahrzeug VaMoRs aus [16].

Abbildung 1.5: VaMP, Prometheus Projekt.

Unmittelbar darauf startete die Europäische Forschungsförderungsorganisation EUREKA das Forschungsprojekt Prometheus (**PRO**gra**M**me for a **E**uropean **T**raffic of **H**ighest **E**fficiency and **U**nprecedented **S**afety) mit dem Ziel, die Machbarkeit des automatisierten Fahrens unter realen Verkehrsbedingungen zu erforschen. Ein wesentlicher Projekterfolg entstand abermals durch die Arbeiten in

Dickmanns' Team, als Mitte der 90er Jahre zwei umgebaute Mercedes-Benz-S-Klasse-Fahrzeuge (VaMoRs Passenger Car (kurz VaMP [17])) international präsentiert wurden (siehe Abb. 1.5). Die beiden Fahrzeuge absolvierten mehr als 1000 Kilometer in normalem Verkehr auf mehrspurigen Autobahnen. Bei Geschwindigkeiten bis zu 130 km/h wurde das Fahren auf freien Fahrstreifen, die Folgefahrt zu einem Vorderfahrzeugs bei konstantem Abstand, sowie Fahrstreifenwechsel nach links und rechts demonstriert.

Im Jahre 2002 kündigte die Defense Advanced Research Projects Agency (DARPA) einen Wettbewerb für autonome Fahrzeuge an [18]. Es galt, mit einem Fahrzeug ohne Fahrer an Bord einen festgelegten Kurs ohne menschliches Eingreifen zu absolvieren. Der Kurs war insgesamt rund 240 km lang und führte durch die Mojave-Wüste in den Vereinigten Staaten. Der Weg war durch GPS-Wegpunkte vorgegeben und führte sowohl über asphaltierte Wege, als auch über steinige Passagen mit Hindernissen, die umfahren werden mussten. Zum Start 2004 traten über 20 Teams an, jedoch konnte keines der Teams das Ziel erreichen. Selbst dem besten Team gelang es nicht, mehr als fünf Prozent der Strecke zurückzulegen.

Unmittelbar nach der Grand Challenge wurde die zweite Grand Challenge angekündigt, die 2005 stattfand [19]. Dieses Mal ging es nicht nur darum, überhaupt den Kurs von 212 km zu bewältigen, sondern dies auch in möglichst kurzer Zeit. Von den 23 Finalisten schafften es immerhin 5 Teams den Kurs vollständig zu bewältigen. Dabei siegte das Team der Stanford Universität mit ihrem Fahrzeug „Stanley" (siehe Abb. 1.6a), welches in Zusammenarbeit mit Volkswagen of America, Inc. und Intel Research entstand.

(a) **(b)**

Abbildung 1.6: (a) „Stanley": Sieger der 2. Darpa Grand Challenge [19], (b) „Boss": Sieger der Darpa Urban Challenge [20].

In [21] findet sich eine detaillierte Beschreibung der Systeme und Verfahren, die bei „Stanley" zum Einsatz kamen. Viele der dort beschriebenen Problemstellungen und Lösungen bilden noch heute die Grundlage für die Entwicklung im Bereich der Fahrerassistenz und des automatisierten Fahrens.

Die bisher letzte Darpa Challenge war die Urban Challenge. Sie fand im Jahr 2007 auf dem Gelände eines ehemaligen US Air Force Stützpunktes in Kalifornien statt [20].

Im Unterschied zu den beiden Grand Challenges fand die Urban Challenge nicht in der Wüste, sondern auf einem stadtähnlichen Testareal statt. Ziel war es, als Erstes den rund 96 km langen Parcour zu absolvieren. Zudem galt es, eigenständig im Parcour zu navigieren, sämtliche Verkehrsregeln einzuhalten, die anderen Verkehrsteilnehmern zu beachten und statische Hindernisse zu umfahren. Sieger war das Tartan Racing Team [22] der Carnegie Mellon University mit ihrem Fahrzeug „Boss" (Abb. 1.6b) gefolgt vom Team der Stanford University. Den erfolgreichsten deutschen Teams, Team Annieway [23] und Team CarOLO [24], gelang zwar der Einzug ins Finale, sie schieden dort jedoch aus.

Im Anschluss an die DARPA Challenges weiteten sowohl die Industrie als auch die Universitäten die Forschung und Entwicklung von automatisierten Fahrzeugen massiv aus. Und so präsentierte bereits im Jahr 2010 die Technische Universität Braunschweig mit ihrem Stadtpilot-Projekt das erste selbstfahrende Stadtauto - einen umgebauten VW Passat (siehe Abb. 1.7). Ziel des Projektes war das automatisierte Fahren auf dem Stadtring in Braunschweig. Auf dieser rund 11 km langen Strecke wurde unter anderem das Fahren im Stadtverkehr mit Geschwindigkeiten bis zu 60 km/h, der Umgang mit Kreuzungen und Ampeln, Fahrstreifenwechsel- und Abbiegemanövern sowie das Einfädeln in den fließenden Verkehr demonstriert [25]. Das Team konnte dabei auf die Erfahrungen in der DARPA Urban Challenge zurückgreifen, die es dort mit dem Fahrzeug CarOLO gesammelt hatte. Im Vergleich zum CarOLO-Aufbau musste allerdings für das automatisierte Fahren in der Braunschweiger Innenstadt sowohl der Sensor-Aufbau als auch der gesamte Software-Stack neu entwickelt werden [26].

Trotz des Ziels die Strecke vollständig automatisiert zu fahren, traten Situationen auf, in denen der Fahrer eingreifen musste, da das System die vorliegende Situation nicht beherrschte [27].

Auch das 2006 gegründete „Autonomos Labs" um Prof. Dr. Raúl Rojas an der Freien Universität Berlin beschäftigt sich mit dem automatisierten Fahren. Nach der Teilnahme an der Darpa Urban Challenge wurden 2 weitere Fahrzeuge entwickelt, die automatisiert fahren können. Seit 2011 fährt das Fahrzeug „MadeIn-Berlin" automatisiert durch die Straßen von Berlin (siehe Abb. 1.8). Bereits ein Jahr später folgte eine Fahrt in Mexico City [28] und 2015 eine 2400 km lange Fahrt von den USA nach Mexico [29]. Daneben ist das Fahrzeug das erste weltweit, das auf öffentlichen Straßen in der Schweiz fahren darf. Erste erfolgreiche Fahrten fanden im Jahr 2015 in der Stadt Zürich statt [30].

(a) (b)

Abbildung 1.7: Fahrzeug der Technischen Universität Braunschweig für das Stadtpilot-Projekt (Quelle :[25]).

(a) (b)

Abbildung 1.8: Automatsiertes Fahren der FU Berlin (Quelle: [31]).

Darüber hinaus existieren weitere universitäre Projekte, wie beispielsweise das Autonome Fahren an der Universität Ulm [32], die sich mit den verschiedenen anderen Aspekten des automatisierten Fahrens beschäftigen.

Neben den Universitäten gibt es auch einige öffentlich bekannte Projekte der Industrie zum Thema automatisiertes Fahren. Fast jeder Fahrzeughersteller sowie die großen Zulieferer-Firmen arbeiten an selbstfahrenden Fahrzeugen oder an Teilkomponenten davon. Der Fokus der Industrie liegt dabei meist beim automatisierten Fahren auf Autobahnen mit einer preiswerten und damit massenmarkt-tauglichen Sensorik für eine mögliche Serieneinführung in den kommenden Jahren. Dazu werden momentan Systeme entwickelt, die die Fahraufgabe auf der Autobahn vollständig übernehmen können. Dies umfasst neben dem Fahren auf dem aktuellen Fahrstreifen auch das automatisierte Überholen sowie die Bewältigung von Stausituationen. Im Falle einer kritischen Situation muss jedoch der Fahrer die Kontrolle über das Fahrzeug in angemessener Zeit wieder übernehmen.

Beispiele solcher Systeme finden sich in [33, 34, 35, 36]. Die Systeme nutzen Sensoren, die sich bereits in einem seriennahen Entwicklungsstadium befinden. Dachaufbauten mit großen rotierenden Laserscannern finden sich in diesen Projekten bereits nicht mehr. Die Integration der Sensoren in das Fahrzeug bei gleichzeitiger Beachtung des Designs spielt für Fahrzeughersteller naturgemäß eine große Rolle, da die Fahrzeuge an Endkunden verkauft werden sollen. Zum Einsatz kommen neben handelsüblichen Radarsensoren, die man bereits aus der Fahrerassistenz kennt (ACC), auch Stereo-Kameras und Laserscanner, um Fahrstreifenmarkierungen, andere Verkehrsteilnehmer und Hindernisse zuverlässig erkennen zu können.

Das erste dieser Systeme wurde 2011 von BMW auf einer Fahrt von München nach Ingolstadt präsentiert [37] (siehe Abb. 1.9). Dabei wurden 30 vollständig automatisierte Fahrstreifenwechsel durchgeführt, bei denen das System neben der Durchführung des Spurwechsels auch die Entscheidung, einen solchen durchzuführen, eigenständig getroffen hat. Die Architektur des entwickelten Systems besteht aus einer Umfelderfassung, der Verwendung einer Lokalisierung mit einer hochgenauen Karte, sowie einer Manöver- und Trajektorienplanung [38, 39].

(a) **(b)**

Abbildung 1.9: Hochautomatisiertes Fahren von BMW auf der Autobahn (Quelle: [40]).

Anfang 2015 legte dann „Jack", ein von Audi umgebauter A7 unter dem Projektnamen „Audi piloted driving concept", eine 900 Kilometer lange Strecke in den USA vom Silicon Valley nach Las Vegas zurück (siehe Abb. 1.10). Bereits im Vorfeld testete Audi auf deutschen Autobahnen das automatisierte Fahren mit Geschwindigkeiten bis zu 130 km/h.

Neben dem automatisierten Fahren auf der Autobahn entwickelte Daimler einen Prototypen mit dem Ziel, die historische Route der Bertha Benz aus dem Jahr 1888 nachzufahren - dieses Mal automatisiert [41]. Die rund 100 km lange Route besteht aus innerstädtischen Situationen und Landstraßen. 2013 präsentierte

(a) (b)

Abbildung 1.10: Autonom fahrender Audi A7 mit Erklärungen zur verbauten Sensorik (Quelle: [33]).

Daimler die umgebaute Mercedes S-Klasse und damit das automatisierte Fahren in der Stadt (siehe Abb. 1.11). Zum Einsatz kamen einige Sensoren, die sich bereits in einem seriennahen Entwicklungsstadium befinden wie bspw. das Computer-Vision-System auf Basis einer Stereo-Kamera [42]. Eine abgewandelte Form dieses Systems befindet sich bereits in der aktuellen auf dem Markt erhältlichen Mercedes S-Klasse.

(a) (b)

Abbildung 1.11: Hochautomatisiertes Fahren in der Stadt von Daimer (Quelle: [43]). Das System ist in der Lage, auch schwierige Situation wie bspw. das Beachten eines Zebrastreifens zu meistern.

Auch andere Unternehmen, die nicht aus dem Automobilsektor stammen, sind auf dem Gebiet des automatisierten Fahrens aktiv. So startete das weltweite Technologie-Unternehmen Google/Alphabet im Jahr 2009 die ersten Arbeiten

am „Google's self-driving car project". Mittlerweile firmiert das Projekt unter dem Namen Waymo. Es ist das bisher weltweit größte Projekt für das automatisierte bzw. autonome Fahren (siehe Abb. 1.12).

Laut dem Waymo Safety Report [44] haben autonom fahrende Fahrzeuge bis heute einige Millionen Kilometer zurückgelegt (Stand Mai 2017). Dabei konzentriert sich Waymo komplett auf das autonome Fahren, bei dem der Fahrer zum Passagier wird und zu keinem Zeitpunkt in das Fahrgeschehen eingreifen muss. Dementsprechend entfallen bei den Waymo-Fahrzeugen teilweise Lenkrad und Pedale.

(a) **(b)**

Abbildung 1.12: Autonome Fahrzeuge von Waymo (Quelle: [45]).

Automatisierung im Grenzbereich

Die Automatisierung im Grenzbereich kann als rein akademisches Problem angesehen werden. Die dahinterliegende Frage ist, wo die Grenze der realisierbaren Automatisierung verläuft. Zusätzlich kann man der Automatisierung im Grenzbereich auch einen gewissen Unterhaltungswert zuschreiben, da beispielsweise das Fahren auf einer Rennstrecke für viele Fahrer Spaß bedeutet. Allerdings existieren auch wichtige Gründe, Fahrzeuge im Grenzbereich automatisiert fahren zu lassen, um das Grenz-Verhalten zu erforschen bzw. zu beherrschen. Ein autonomes Fahrzeug fährt im normalem Verkehr zwar niemals im Grenzbereich und befindet sich stets unter der Kontrolle des Computers, doch auch in Gefahrensituationen, wie beispielsweise einem Notausweichmanöver oder einer Vollbremsung, sollte ein automatisiertes Fahrzeug jederzeit durch den Computer kontrollierbar sein. Da es bei der Forschung im Grenzbereich meist direkt und ausschließlich um Problemstellungen aus der Regelungstechnik und Fahrzeugregelung geht, werden die anderen Probleme wie Umfelderkennung oder Manöverplanung meist nicht mitbetrachtet. Dementsprechend finden sich auch in den Fahrzeugen dieser Projekte meist nur wenige Sensoren.

Eines der Vorhaben, die auch einen gewissen Unterhaltungswert besitzen, ist das automatisierte Fahren auf der Rennstrecke. Eines der ersten Projekte dieser Art war der BMW TrackTrainer [40]. Es ging einerseits darum, automatisiert Runden auf der Rennstrecke zu fahren, und andererseits darum, dem Fahrer gleichzeitig das Fahren der Ideallinie auf ebendieser Strecke beizubringen. Das Prinzip der technischen Umsetzung basiert darauf, vorab die Ideallinie der Strecke manuell zu fahren, während permanent die aktuelle Fahrzeugposition aufgezeichnet wird. Daraus kann im Nachgang eine Trajektorie generiert werden, die später automatisiert abgefahren werden kann. Dies ist ein gängiges Verfahren und kommt bei diversen Projekten dieser Art zum Einsatz. Um dem Fahrer das Fahren der Ideallinie beizubringen, wird während der Fahrt die aktuelle Abweichung der Fahrzeugposition zur Ideallinie angezeigt. So kann der Fahrer stets sehen, ob und wie er von der Ideallinie abweicht und - sofern er selbst fährt - sein Fahrverhalten korrigieren.

(a) **(b)**

Abbildung 1.13: BMW TrackTrainer: Automatisiertes Fahren auf der Rennstrecke (Quelle: [40]).

Für den BMW TrackTrainer wird zur permanenten Fahrzeugposition hochgenaues GPS verwendet. Da hochgenaues GPS nicht flächendeckend verfügbar ist und die Qualität wetterabhängig ist, wird die GPS-Position des Fahrzeugs mit einem kamerabasierten Positionierungsalgorithmus kombiniert. Unter Verwendung der detektierten Markierungslinien und einer hochgenauen Karte der Strecke gelingt eine zuverlässige und stabile Berechnung der Fahrzeugposition [46]. Das System wurde 2009 demonstriert, als ein Fahrzeug autonom eine vollständige Runde auf der Nürburgring-Nordschleife fuhr (siehe Abb. 1.13).

Später fuhr im Rahmen des DTM-Finale auf dem Hockenheimring ein Audi RS7 vollständig fahrerlos eine Runde im Renntempo über die Rennstrecke (siehe Abb. 1.14a). Das gleiche Experiment wurde mittlerweile auch auf anderen Rennstrecken u.a. in Sonoma (USA) und in Barcelona (Spanien) wiederholt [33].

Die Erfahrungen auf der Rennstrecke zeigen, dass das autonome Fahren auch bei hohen Geschwindigkeiten möglich ist.

Eine Gefahrensituation, die ein autonomes Fahrzeug auch beherrschen muss, ist das Übersteuern oder „Ausbrechen" des Fahrzeugs - eine Bewegung des Fahrzeug, die nicht entlang seiner Längsachse verläuft. Auf der Consumer Electronics Show 2014 präsentierte BMW daher das automatisierte Driften [47] (siehe Abb. 1.14b).

(a) **(b)**

Abbildung 1.14: (a) Autonom fahrender Audi RS7 auf der Rennstrecke (Quelle: [48]), (b) Automatisiertes Fahren im Grenzbereich. BMW zeigte das automatisierte Driften bei der Consumer Electronics Show 2014 in Las Vegas (Quelle: [47]).

Eine Forschungsgruppe an der Stanford University hat in Zusammenarbeit mit Volkswagen und Audi ein weiteres Projekt zum automatisierten Fahren im Grenzbereich durchgeführt. Es wurde ein Audi TTS entwickelt, der die Rennstrecke des Pike's Peak International Hill Climb [49] autonom fahren kann (siehe Abb. 1.15). Das Rennen wird auch „Race To The Clouds" genannt, da die rund 20 km lange Strecke mit 156 Kurven vom Start zum Ziel einen Höhenunterschied von ca. 1440 Metern aufweist. Der Start liegt bereits auf 2862 Metern Höhe. Genauso wie beim BMW TrackTrainer wurde hier ein hochgenaues GPS verwendet, um das Fahrzeug auf einem vorher aufgezeichneten Pfad fahren zu lassen. Der Audi fuhr die Strecke im Jahre 2010 fahrerlos in einer Zeit von 27 Minuten, und war somit nur 10 Minuten langsamer als normale Rennteilnehmer.

Die Entwicklung von Fahrdynamikregelungsalgorithmen für den Grenzbereich sind ein nicht zu unterschätzender Faktor auf dem Weg zur Markteinführung autonomer Fahrzeuge, da diese auch in gefährlichen Situationen stets zuverlässig und korrekt fahren können müssen. Projekte, in denen solche Algorithmen entwickelt werden, können daher - neben dem Unterhaltungsaspekt - die Sicherheit autonomer Fahrzeuge zukünftig enorm steigern.

Abbildung 1.15: „Shelly": Stanford University und Volkswagen entwickelten dieses Auto, um die Rennstrecke des „Pikes Peak race of Colorado Springs" autonom zu fahren (Quelle: [50]).

Automatisiertes Parken

Eine etwas andere, aber alltägliche Art des Fahrens ist das Ein- und Ausparken. Auch in diesem Bereich gibt es diverse Anstrengungen, den Park-Vorgang vollständig zu automatisieren und schlussendlich das vollständig fahrerlose Einparken zu ermöglichen. Das Parken unterscheidet sich in mancher Hinsicht grundlegend vom automatisierten Fahren. Die Wahrnehmung des Umfeldes ist vergleichbar mit dem Fahren in der Stadt. Es müssen ebenso wie beim Fahren in der Stadt alle Verkehrsteilnehmer zuverlässig erkannt werden. Hinzu kommt die Erkennung von Bordsteinen und Parkplätzen. Die Manöver- und Trajektorienplanung beim Parken unterscheidet sich allerdings grundlegend, da Rangieren normalerweise kein Bestandteil des Fahrens in der Stadt ist. Hinzu kommen neue Aspekte wie eine Anfahrtsfreigabe sowie das automatische Starten und Stoppen des Motors.

Für das Valet Parking wird eine hochgenaue Karte des Parkhauses benötigt, sowie eine darauf abgestimmte Lokalisierung des Fahrzeugs relativ zur Karte. Zusätzlich erkennen die im Fahrzeug verbauten Sensoren Objekte und können damit Kollisionen verhindern. Bewegungsplanungsalgorithmen planen für das Fahrzeug die Manöver und Trajektorien, die benötigt werden, um das Fahrzeug in einen freien Parkplatz einzuparken [51, 52, 53, 54, 55]. BMW demonstrierte 2015 das autonome Valet Parken mit einem elektrischen i3, der mit diverser Sensorik zur Umfelderfassung ausgestattet ist (siehe Abb. 1.16) [40].

Daimler und Bosch starteten ein ähnliches Projekt zum fahrerlosen Parken. Offiziell vorgestellt und demonstriert wurde das Valet Parking im Juli 2017 im Parkhaus des Mercedes-Benz-Museums in Stuttgart (siehe Abb. 1.17) [56]. Das System stellt eine App für das Smartphone bereit, mit dem ein Fahrzeug gebucht werden kann. Das Fahrzeug fährt anschließend autonom von seinem Parkplatz

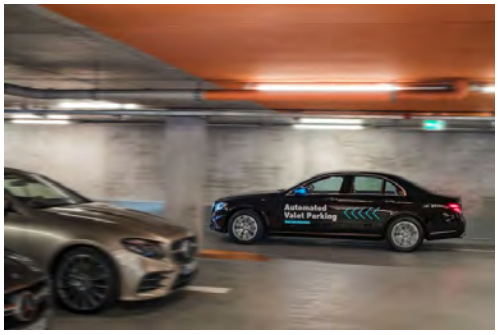
(a)

(b)

Abbildung 1.16: BMW's Remote Valet Parking Projekt, demonstriert auf der Consumer Electronics Show 2015 in Las Vegas. Ein fahrerloser i3 parkt autonom ein und aus (Quelle: [40]).

aus in eine vordefinierte „Pick-up-Area", wo der Kunde das Auto zur Weiterfahrt in Empfang nehmen kann. Bei der Fahrzeugrückgabe stellt der Kunde das Fahrzeug ebenfalls in einem speziellen Bereich des Parkhauses - der „Drop-off-Area" - ab und gibt es per App zurück an das Parkhaus-System. Dieses sorgt dafür, dass das Fahrzeug selbständig aus der „Drop-off-Area" zu einem Parkplatz fährt und einparkt. Dabei wird es vom Parkhaus-System geleitet. Im Gegensatz zu den Ansätzen aus bspw. [51] wurde die „Intelligenz" bei diesem System größtenteils nicht im Auto, sondern in der Parkhaus-Infrastruktur angesiedelt.

(a)



(b)

Abbildung 1.17: Automatisierter Parkservice per Smartphone. Daimler und Bosch testen das fahrerlose Parken in Stuttgart (Quelle: [56]).

Automatisiertes Fahren in anderen Bereichen

Neben dem autonomen Fahren auf der Straße in normalem Verkehr, gibt es weitere Projekte, die sich mit dem autonomen Fahren in anderen Einsatzgebieten beschäftigen. Den bereits vorgestellten Projekten ist das Projekt der deutschen Bahn am nächsten. Sie testete in Bad Birnbach einen autonom fahrenden Kleinbus (siehe Abb. 1.18). Die Sensoren des elektrisch angetriebenen Busses gleichen denen der autonomen PKWs. Es sind sowohl Kameras als auch Laserscanner verbaut.

Abbildung 1.18: Autonom fahrender Bus der Deutschen Bahn (Quelle:[57]).

Auch im militärischen Bereich gibt es Forschungsaktivitäten rund um das autonome Fahren, wie beispielsweise die Universität der Bundeswehr in München mit ihrem Forschungsfahrzeug MuCAR-3 [58, 59] gezeigt hat. Analog zu den Fahrzeugen der Universitäten besitzt auch dieses Fahrzeug einen 360 Grad Laserscanner auf dem Dach, sowie Kameras zur Umfelderfassung. Das Forschungsfeld umfasst die Navigation [60], Vehicle-Convoy-Anwendungen [61, 62, 63], Umfeldwahrnehmung in unstrukturierten Umgebungen [64] und das off-road Fahren [65, 66].

Den meisten hier vorgestellten Projekten ist gemein, dass stets ein Fahrer im Fahrzeug präsent sein muss, der das System überwacht, um in kritischen Situationen eingreifen zu können. Dies bedeutet aber auch, dass trotz der enormen Aktivitäten der letzten Jahre nach wie vor an den Themen und Problemstellungen geforscht werden muss, damit das vollständig autonome Fahren ohne Fahrer Wirklichkeit werden kann.

Darüber hinaus benötigen die heutigen Systeme viele Sensoren, die sich teilweise noch in einem prototypischen Entwicklungsstadium befinden oder aus Kostengründen nicht wirtschaftlich sind. Auch das Beherrschen von komplexen oder unvorhersehbaren Situationen, sowie Sondersituationen (Bildung von Rettungsgassen, Vorrang für Einsatzfahrzeuge von Polizei/Feuerwehr) stellen nach wie vor eine Herausforderung für autonome Fahrzeuge dar.

Einer der zentralen Aspekte ist die Erfassung und Repräsentation der aktuellen Fahrzeugumgebung. Da dies den Schwerpunkt der vorliegenden Arbeit darstellt, wird dieser Aspekt ausführlich im nächsten Abschnitt behandelt.

1.2 Umfeldmodellierung für das automatisierte Fahren

Durch die Entwicklung und Integration neuartiger Automatisierungs- und Assistenzfunktionen stellt sich zunehmend die Frage einer präzisen und robusten Abbildung der aktuellen Fahrzeugumgebung in ein geeignetes Umfeldmodell.

Viele der heutzutage bereits erhältlichen Assistenzfunktionen benötigen für Ihre Funktionen Kenntnisse über die Fahrzeugumgebung. Zur Fahrzeugumgebung gehören neben Hindernissen und anderen Verkehrsteilnehmern auch Informationen über die befahrbaren Flächen, die eigene Fahrzeugposition sowie den Verlauf von Fahrbahn und Fahrstreifen. Aufgrund der vielen unterschiedlichen Situationen, die im realen Straßenverkehr auftreten, ist die Umgebungswahrnehmung und -modellierung durch ein Fahrzeug auch heutzutage eine fortwährende Herausforderung.

Für die Umfelderkennung haben inzwischen verschiedene Sensoren in die Fahrzeuge Einzug gehalten. Als Beispiel seien hier Radarsensoren [67] für die automatische Abstandsregelung (Adaptive Cruise Control: ACC), Videokameras [68] (Anwendung: Spurhalteassistent, Spurverlassenswarner), sowie Ultraschall-Sensoren [69] für Park-Distance-Control (PDC) bzw. für einen Einparkassistenten erwähnt. Voraussichtlich werden zukünftig weitere Sensoren in den Fahrzeugen verbaut werden wie beispielsweise Laserscanner [70], um die Umfelderkennung zu verbessern. Die Verwendung weiterer Sensoren führt zwangsläufig dazu, dass die Daten aller Sensoren in einem geeigneten Modell zusammengeführt werden müssen. Dieses Modell, das einer Abbildung der realen Welt entspricht, wird auch als „Umfeldmodell" bezeichnet. Durch die Nutzung aller potentiell verfügbaren Sensordaten, können Assistenzsysteme deutlich an Qualität gewinnen bei gleichzeitiger Reduzierung der Kosten des Gesamtsystems [71, 72]. Komplexere Anwendungen wie beispielsweise das automatisierte Fahren werden durch diese Architektur überhaupt erst ermöglicht.

Das aus den Sensordaten erzeugte Umfeldmodell ist eine Abbildung der realen Welt in ein entsprechendes Modell und dient als Grundlage für die weiteren Aufgaben des automatisierten Fahrens, wie z.B. die Manöverplanung und Bewegungsplanung. Eine mögliche Systemarchitektur für das automatisierte Fahren ist in Abb. 1.19 abgebildet. Da diese Themen nicht Gegenstand der vorliegenden Arbeit sind, wird im weiteren Verlauf nicht weiter auf sie eingegangen. Weiterführende Information dazu finden sich in [36].

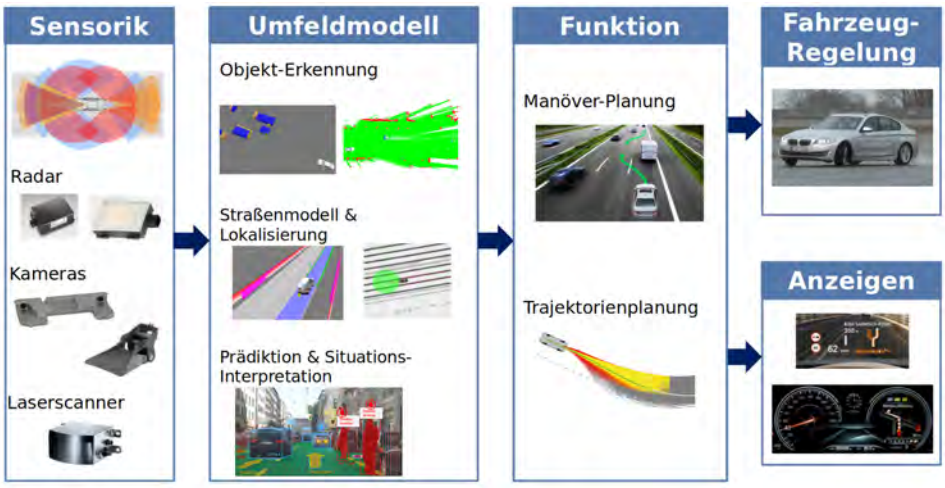


Abbildung 1.19: Exemplarische Architektur für das automatisierte Fahren. Aufteilung in Sensorik, Umfeldmodell, Funktion, Fahrzeugregelung und Anzeigen.

Das Konzept eines Umfeldmodells ist nicht auf das automatisierte Fahren beschränkt, sondern ein vielfach eingesetztes Konzept in der Robotik. Die konkrete Ausgestaltung des Modells ist dabei stets abhängig vom Einsatzszenario. So sieht ein Umfeldmodell für einen Rasenmähroboter anders aus, als für einen Roboterarm, der Sachen in ein Regal ein- bzw. aus einem Regal ausräumt. Für das automatisierte Fahren sollte das Umfeldmodell die nähere Fahrzeugumgebung abbilden und als Grundlage für eine darauf aufbauende Manöver- und Trajektorienplanung dienen. Daher beinhaltet das Umfeldmodell unter anderem:

- die Geometrie der Fahrbahn inkl. der Fahrstreifen und deren Verbindungen (Kreuzungen),
- Gehwege, Fahrradwege, Zebrastreifen,
- dynamische Objekte wie andere Fahrzeuge, Fahrradfahrer, Fussgänger,
- Verkehrsschilder, Ampeln inkl. deren Zuordnung zu Fahrstreifen und zu Anhaltelinien.

Nicht Bestandteil des Umfeldmodells sind Verkehrsregeln, wie Vorfahrtsregeln oder das Rechtsfahrgebot auf Autobahnen, da dies keine wahrnehmbaren Elemente sind, sondern Regeln, die Teil des domänen-spezifischen Wissens des Systems sein müssen.

Prinzipiell gibt es zwei unterschiedliche Datenquellen, aus denen ein Umfeldmodell erzeugt werden kann. Die Informationen können entweder in einer abgespeicherten Karte liegen oder aus Messdaten der im Fahrzeug verbauten Sensoren extrahiert werden. Während die meisten heutzutage erhältlichen Fahrassistenzsysteme vorwiegend auf sensor-basierte Umfeldmodelle setzen, benötigen die Projekte des automatisierten oder autonomen Fahrens eine hochgenaue, globale Karte der Umgebung, auf der das Umfeldmodell aufbaut. Die Verfügbarkeit solcher hochgenauen Karten wird oft als entscheidender Faktor für das autonome Fahren gesehen [73, 74, 75]. Daher beschäftigt sich das nächste Kapitel ausführlicher mit der Generierung eines Umfeldmodells mit Hilfe dieser Karten.

1.2.1 Kartenbasierte Umfeldmodellierung

Ein naheliegender Ansatz für die Generierung eines Umfeldmodells ist die Verwendung einer hochgenauen Karte. Es dient der Vereinfachung des ohnehin komplexen Problems des autonomen Fahrens. Dementsprechend muss die Karte alle für die Fahraufgabe relevanten Informationen mit einer hohen Genauigkeit und Aktualität beinhalten. Diese Anforderungen an die Karte übersteigen bei Weitem das, was an Karten in heutigen Navigationssystemen zu finden ist [76].

Darüber hinaus existieren Elemente im Straßenverkehr, die dynamisch sind - sich also mit der Zeit verändern. Dazu zählen unter anderem andere Verkehrsteilnehmer und bspw. der Status einer Ampel. Die dynamischen Elemente müssen daher trotz Verwendung eines kartenbasierten Umfeldmodells über geeignete Sensorik erfasst und mit der Karte zusammengeführt werden. So gesehen ist ein kartenbasiertes Umfeldmodell nichts anderes als eine Abspaltung und Auslagerung der statischen Teile des Umfeldmodells in eine Karte.

Daraus ergeben sich unter anderem die folgenden Vorteile:

- Das Fahrzeugumfeld muss nicht mehr vollständig wahrgenommen werden, weil ein Großteil der benötigten Informationen aus der Karte stammt.
- Messdaten der Sensoren können plausibilisiert und validiert werden gegen die Karte. Beispielsweise können „Geisterobjekte" als solche identifiziert werden, sofern sie sich nicht auf der Straße befinden (oder bei Fussgängern auch auf dem Gehweg).
- Die Vorausschau und Einsicht in Kreuzungsbereiche bzgl. der Fahrbahn und Fahrstreifen ist gegeben.
- Fahrbahn- und Fahrstreifen sind vollständig im Umfeldmodell enthalten.

Der Einsatz einer Karte erzwingt jedoch die Lösung des sogenannten *Lokalisierungsproblems*. Dabei muss permanent die Position des eigenen Fahrzeugs relativ zur Karte möglichst präzise bestimmt werden. Nur wenn hinreichend genau bekannt ist, wo sich das Ego-Fahrzeug relativ zur Karte gerade befindet, können die Informationen aus der Karte nutzbringend verwendet werden. Dabei muss die Lokalisierungs-Genauigkeit mit der Genauigkeit und dem Detaillierungsgrad der Karteninhalte zusammenpassen. Je detaillierter die Karte ist, desto genauer muss die Lokalisierung sein. In einer heute erhältlichen Karte eines Navigationssystem sind üblicherweise nur die Straßengeometrien hinterlegt. Weitere Substrukturen, wie beispielsweise die einzelnen Fahrstreifen sind nicht Bestandteil der Karte. Daher muss eine Lokalisierung auch nur so exakt sein, dass die Ego-Position auf der korrekten Straße liegt. Eine fahrstreifengenaue Lokalisierung ist nicht notwendig, da die Karte diesen Detaillierungsgrad nicht beinhaltet. Üblicherweise wird daher ein handelsübliches GPS-Signal zusammen mit den Odometrie-Daten des Fahrzeugs verwendet, um die Position des Ego-Fahrzeuges zu berechnen.

Für hochgenaue Karten, die für das automatisierte Fahren benötigt werden, ist diese Art der Lokalisierung nicht präzise genug. Um Fahrentscheidungen treffen zu können, ist es notwendig, nicht nur den weiteren Straßenverlauf zu kennen, sondern auch die Geometrie der einzelnen Fahrstreifen exakt zu kennen. Darüber hinaus, kann ein Positionsfehler von einigen Zentimetern bereits gravierende Auswirkungen haben.

Üblicherweise wird daher für die Lokalisierung auf das hochgenaue *Differential GPS (DGPS)* zurückgegriffen. Dabei wird die Fahrzeugposition relativ zur Welt (globale Position) bestimmt. Da die Karte üblicherweise ebenfalls in Weltkoordinaten vorliegt, ist damit die Positionsbestimmung relativ zur Karte direkt gegeben. Auch die in Abschnitt 1.1 vorgestellten Projekte nutzen bis auf wenige Ausnahmen diese Technik zur Lokalisierung.

Allerdings ist die Lokalisierung auf Basis von DGPS nicht robust genug für den tagtäglichen Einsatz. Besonders in Straßenzügen mit dichter Bebauung, wie sie in Großstädten mit Hochhäusern häufig vorkommen, treten sogenannte *Multipath-Effekte* auf, die zu gravierenden Verfälschungen der Positionsbestimmung führen können. Darüber hinaus ist kein flächendeckender Empfang von DGPS-Signalen gewährleistet, da bspw. in Tunneln kein Empfang möglich ist.

Aus diesen Gründen sind bereits andere Ansätze entwickelt worden, die das Lokalisierungsproblem behandeln. Entgegen einer (D)GPS-basierten Lokalisierung setzen diese Ansätze auf die Wiederkennung von Landmarken zur Positionsbestimmung. Sie basieren darauf, dass bestimmte, von der Fahrzeug-Sensorik gemessene, Landmarken den in der Karte gespeicherten Landmarken zugeordnet werden können. Daraus lässt sich anschließend eine Positionshypothese für

das Fahrzeug ableiten. Der Begriff *Landmarke* ist hierbei als Abstraktion zu sehen, da prinzipiell Messdaten in jeder Form als Landmarke dienen können, unter der Voraussetzung, dass einerseits die Karte diese Landmarke beinhaltet und andererseits eine eindeutige Zuordnung der gemessenen Landmarke zu der in der Karte gespeicherten gegeben ist. Ein einfaches Beispiel für Landmarken sind Verkehrsschilder. Sie sind einerseits in der Karte hinterlegt, und können andererseits durch im Fahrzeug verbaute Sensorik erkannt und klassifiziert (Feststellung des Schildertyps) werden. Das erkannte Verkehrsschild kann nun mit dem Schild in der Karte assoziiert werden und aus der Entfernungsmessung die exakte Position des Fahrzeugs relativ zur Karte berechnet werden. Für dieses Verfahren wird keine globale Fahrzeugposition bestimmt, wie dies beim (D)GPS der Fall ist, sondern ausschließlich die lokale Position relativ zur Karte - unabhängig von der tatsächlichen Welt-Position. Daher wird dieses Verfahren auch als *kartenrelative Lokalisierung* bezeichnet [41].

Die Wahl der Landmarken ist vielfältig und kann sowohl reale Objekte, als auch abstrakte Messgrößen beinhalten. Die Bandbreite reicht von Fahrstreifenmarkierungslinien [41] über abstrakte Feature-Vektoren aus Kamerabildern (ebenfalls [41]) bis zu Belegungskarten [36] oder kompletten Messungen von Laserscannern [77].

Die kartenrelative Lokalisierung setzt immer voraus, dass die Karte nicht nur die für die Fahraufgabe relevanten Informationen beinhaltet, sondern zusätzlich auch die für die Lokalisierung benötigten Inhalte bereitstellt. Diese können, wie im Fall der Feature-Vektoren oder der Laserscanner-Scans, komplett unabhängig, ja sogar nutzlos, für die eigentliche Fahraufgabe sein.

Das Lokalisierungsproblem ist bis heute nicht abschließend gelöst. Trotz der teils vielversprechenden Ansätze bleibt die robuste und präzise Positionsbestimmung relativ zur Karte eine Herausforderung. Dies ist jedoch eine Grundvoraussetzung für die Nutzung eines kartenbasierten Umfeldmodells.

Neben der Lokalisierung existieren weitere Fragestellungen, die bei der Verwendung einer hochgenauen Karte zu klären sind. Die wichtigesten Punkte sind die **Kartenverfügbarkeit** und die **Aktualisierung und Validierung der Karte**.

Die Verfügbarkeit einer Karte der Umgebung ist bei einem kartenbasierten Umfeldmodell eine zwingende Voraussetzung für die Fahrfunktion. Somit schränkt es die Verfügbarkeit des Systems auf diejenigen Strecken ein, die im Vorfeld kartiert worden sind. Da derzeit kein technischer Standard für eine solche Karte existiert, greifen die verschiedenen Hersteller und Universitäten auf unterschiedliche Lösungen zurück, die untereinander weitgehend inkompatibel sind. Darüber hinaus hängen die benötigten Karteninhalte auch davon ab, wie das Lokalisierungsproblem jeweils gelöst wird. Daher ergeben sich gegenwärtig nur wenig Synergien für die verschiedenen Projekte zum automatisierten Fahren.

Zusammen mit der Tatsache, dass jede zu kartierende Strecke teils mehrfach gefahren werden muss und der Kartenerstellungsprozess nach wie vor nicht vollständig automatisiert ist, ist die flächendeckende Bereitstellung solcher Karten eine große und teure Herausforderung.

(a) **(b)**

Abbildung 1.20: Zwei Fahrzeuge, die die Daten zur Erstellung einer hochgenauen Karte aufzeichnen können. Der Sensoraufbau unterscheidet sich in Umfang und Alltagstauglichkeit deutlich von Serienfahrzeugen und den autonomen Prototypen der Fahrzeughersteller (Quellen: [78], [79]).

Für automatisierte Fahrmanöver muss die verwendete Karte aktuell und korrekt sein. Hat sich die Umgebung jedoch nach der Kartierung der Strecke verändert, kann es passieren, dass die Karte nicht mehr korrekt ist, d.h. sie bildet nicht mehr die Wirklichkeit ab. Bekanntestes Beispiel aus dem Alltag sind Baustellen, in denen der Fahrstreifenverlauf oftmals temporär verändert wird. Aber nicht nur Baustellen, sondern auch permanente Veränderungen der Umgebung können dazu führen, dass die Karte ungültig wird. Beispiele hierfür sind das Ersetzen von Kreuzungen durch Kreisverkehre, oder die Sanierung von Straßen, bei denen der Fahrstreifenverlauf leicht geändert wird. Allein die Häufigkeit von Baustellen (siehe Abb. 1.21) zwingt zu einer Lösung des Problems der Kartenaktualisierung, da es sonst nur wenige Strecken geben würde, auf denen autonom gefahren werden könnte.

Neben dem in der Karte befindlichen Straßenmodell (Fahrstreifen) können auch die für eine Lokalisierung benötigten Inhalte ungültig sein, weil sich die Umgebung verändert hat. Die Bandbreite der Veränderungen reicht von kleinen Veränderungen wie der der Lichtverhältnisse, über baubedingte Veränderungen von Gebäudefassaden oder Bürgersteigen bis zu naturbedingten Veränderungen von Grünstreifen und der Vegetation abseits der Straße. Hinzu kommen dynamische Veränderungen wie bspw. parkende Fahrzeuge, die unter Umständen wichtige Landmarken wie Bürgersteige oder Randbebauungen verdecken. All diese Veränderungen können dazu führen, dass eine robuste Lokalisierung

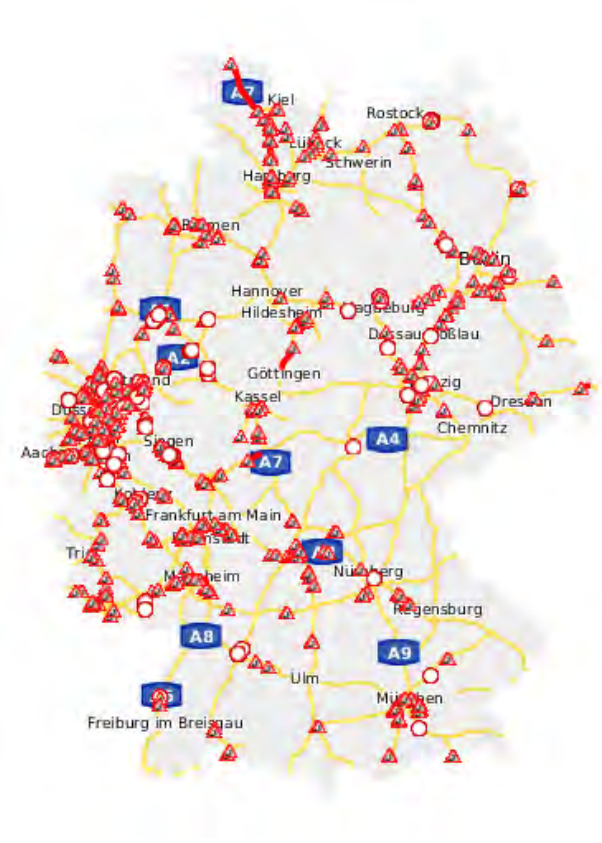


Abbildung 1.21: Deutschlandkarte mit allen tagesaktuellen Autobahnbaustellen, veröffentlicht von der Bundesanstalt für Straßenwesen. Jedes rote Dreieck entspricht einer Autobahnbaustelle, jedes Einfahrt-Verboten-Zeichen einer Autobahn-Sperrung (Quelle [80]).

nicht mehr möglich ist, da sich die Landmarken verändert haben oder zu wenige davon mit den Sensoren erfasst werden können.

Um zu verhindern, dass die Karte veraltet und damit ungültig wird, wäre es wünschenswert, die Strecken permanent neu zu kartieren. Dies ist jedoch aus praktischer Sicht unmöglich, da spezielle Fahrzeuge für die Messdatenaufzeichnung benötigt werden und diese in nur sehr begrenzter Zahl verfügbar sind. In Abb. 1.20 sind zwei dieser Kartierungsfahrzeuge abgebildet. Eine permanente Neu-Kartierung eines nationalen Straßennetzes ist somit nicht möglich. Seit einiger Zeit existiert die Idee, die Aktualisierung der Karte durch einen crowdsourcing-Ansatz zu lösen. Dabei sollen die automatisierten Fahrzeuge mit Hilfe ihrer Sensorik die Umgebung permanent erfassen und diese Daten laufend an ein Backend senden. Dort findet dann ein Abgleich mit der Karte statt und es

werden ggf. Änderungen an der Karte vorgenommen, die wiederum allen Fahrzeugen zur Verfügung gestellt werden. Die ersten Arbeiten hierzu sind bereits in Vorbereitung [81, 82]. Dem ersten Fahrzeug, dass eine ungültige Stelle der Karte passiert, fehlt jedoch stets das Wissen über die Änderung der Umgebung.

Neben der Frage, wie das System mit einer ungültigen Karte umgehen sollte, stellt sich natürlich auch die Frage, wie festgestellt werden kann, ob die Karte gültig oder ungültig ist. In der medizinischen Bildgebung und bei der Analyse von Satellitenbildern ist die Änderungserkennung schon weiter verbreitet als beim automatisierten Fahren. Einen guten Überblick über die verwendeten Methoden und deren Anwendungsgebiete findet sich bspw. in [83]. Auch in der Robotik finden sich erste Ansätze zur Änderungserkennung von Karten. So wird beispielsweise in [84] ein Verfahren vorgestellt, dass auf Basis von Punktewolken von Laserscannern Änderungen der realen Welt im Vergleich zur Karte erkennt. Die Karte besteht in diesem Fall nicht aus einem Straßenmodell, sondern aus einer dreidimensionalen Belegungskarte. Eine der wenigen Arbeiten, in denen die Änderungserkennung für hochgenaue Karten des automatisierten Fahrens behandelt wird, ist [85]. Dort wird eine Ausreißererkennung in der Lokalisierung dazu verwendet, Änderungen in der Karte zu erkennen. Dennoch bleibt die Änderungserkennung von hochgenauen Karten ein bis dato ungelöstes Problem. Insbesondere die Frage, was genau sich geändert hat, ist nach wie vor eine Herausforderung.

Aufgrund der oben diskutierten Probleme und Herausforderungen bei der Verwendung eines kartenbasierten Umfeldmodells wird im nächsten Abschnitt mit dem Konzept eines sensorbasierten Umfeldmodells eine mögliche Alternative vorgestellt.

1.2.2 Sensorbasierte Umfeldmodellierung

Im Gegensatz zum kartenbasierten Umfeldmodell, bei dem alle relevanten Informtionen über die Umgebung aus der Karte stammen und mit bestimmten dynamischen Inhalten aus Sensor-Messdaten „angereichert" werden, wird ein sensorbasiertes Umfeldmodell ausschließlich aus Sensor-Messdaten generiert.

Bei der Generierung eines sensorbasierten Umfeldmodells geht es darum, aus den Sensordaten, wie beispielsweise Entfernungsmessungen von Laserscannern oder Radarsensoren ein Abbild der Umgebung inklusive aller Objekte und deren Abhängigkeiten bzw. Beziehungen untereinander zu erzeugen.

Die Formulierung des Problems der Erstellung eines vollständigen Umfeldmodells als ein einziges Problem ist aufgrund der enormen Komplexität nicht zielführend. Daher hat es sich bewährt, die Erstellung des Umfeldmodells in Teilprobleme aufzuteilen, die unabhängig voneinander gelöst werden können. Beispiele hierfür sind: Die Erkennung und Unterscheidung von dynamischen und statischen Objekten; die Klassifizierung von Objekten (Fahrzeug, Fussgänger, LKW, Motorrad, Fahrradfahrer, Pylone/Bake, etc.); die Erkennung von Fahrstreifenmarkierungen; die Erkennung von Schildern und Ampeln oder die Erkennung der Randbebauung. Für viele dieser Teilprobleme existieren Lösungsansätze - jedoch ist derzeit keines der Probleme soweit gelöst, dass das autonome Fahren in komplexen Umgebungen rein auf einem sensorbasierten Umfeldmodell möglich ist. Dabei ist einerseits die Qualität der Sensoren noch nicht gut genug, und auf der anderen Seite die Robustheit der Algorithmen noch nicht auf einem Level, auf dem autonomes Fahren ohne Karte möglich ist.

Ein grundsätzliches Problem bei der Erstellung des Umfeldmodells aus Sensordaten ist der Umgang mit Unsicherheiten, Fehlmessungen und Messrauschen - im Gegensatz zu Karteninformationen, die man heutzutage nahezu fehlerfrei erstellen kann.

Üblicherweise werden beim automatisierten Fahren mehrere verschiedene Sensoren verwendet, um die Umgebung wahrzunehmen. Durch die unterschiedlichen physikalischen Eigenschaften, die den Messprinzipien der Sensoren zugrunde liegen, erhöht sich die Erkennungsrate und Qualität der Umfelderkennung. Manche Probleme können jedoch ausschließlich mit Hilfe eines bestimmten Sensors gelöst werden, wie bspw. die Klassifizierung von Verkehrsschildern, für die zwingend eine Kamera notwendig ist. Ein gängiges Sensor-Setup für das automatisierte Fahren besteht aus Laserscannern (LIDAR), Kameras, Radarsensoren und Ultraschallsensoren. Abgesehen von Laserscannern finden sich diese Sensoren bereits heute in Serienfahrzeugen, die auf dem Mark verfügbar sind (Kamera für Spurverlassenswarnung und Verkehrszeichenerkennung, Radar für ACC, Ultraschall für Parksensoren).

Um die Stärken und Schwächen der verschiedenen Sensoren optimal ausnutzen zu können, werden die Messdaten der Sensoren oftmals miteinander fusioniert. Eine Fusion erhöht unter Umständen zwar die Robustheit und die Zuverlässigkeit eines Ansatzes, jedoch treten bei der Fusion neue Probleme auf, wie beispielsweise die zeitliche Synchronisierung der Sensordaten (zwei Sensoren messen jeweils zu unterschiedlichen Zeiten mit unterschiedlichen Wiederholungsfrequenzen). Auch die Frage, auf welcher Ebene eine Fusion der Daten stattfindet, hat entscheidenden Einfluss auf die Robustheit und Qualität des Umfeldmodells [86], [87], [88].

Je nach Teilproblem des Umfeldmodells haben sich verschiedene Daten-Repräsentationen des Umfeldmodells bewährt. So leuchtet unmittelbar ein, dass sich dynamische Objekte wie beispielsweise andere Fahrzeuge gut als Boxen modellieren lassen. Für statische Objekte hingegen ist dies problematisch, da sich langgezogene gekrümmte Objekte wie Leitplanken in Kurven nur sehr schlecht als Boxen approximieren lassen. Auch bei der Randbebauung oder den Fahrstreifen führen geometrische Modelle schnell zu Problemen, da bspw. komplizierte Krümmungsverläufe von Randbebauungen oder Fahrstreifen nicht präzise genug in Klothoidenmodellen darstellbar sind [89]. Im Gegensatz zu dieser „objektbasierten" Sicht, haben sich weitere Repräsentationen etabliert, wie z.B. Stixel [90, 91, 92]. Ein Stixel ist ein rechteckiges Element, das im Bereich Computer Vision und Semantische Segmentierung von Bildern als Superpixel dient [93, 94].

Die mit Abstand älteste und am weitesten verbreitete Repräsentation ist das gridbasierte Modell [95, 96, 97]. Ein Grid unterteilt die Umgebung des Fahrzeugs in quadratische Zellen, sodass ein planares, zweidimensionales Gitter entsteht. Diese Darstellung eignet sich sehr gut für die Fusion von Sensordaten, da dem Datenmodell keine a-priori-Annahme über die Geometrie der Objekte zugrunde liegt. In diesem Sinne ist es eine modellfreie Repräsentation der Daten. Darüber hinaus ist es eine sensor-unabhängige Darstellung und erleichtert die Weiterverarbeitung wie beispielsweise eine Extraktion von Strukturen oder Objekten aus den Daten. Da die gridbasierte Umfeldmodellierung in der vorliegenden Arbeit eine wichtige Rolle spielt, wird sie in Kapitel 2 ausführlich behandelt.

Vergleicht man die Eigenschaften von einem sensorbasierten Umfeldmodell mit einem kartenbasierten, kann man feststellen, dass sie sich diametral gegenüberstehen. Beispielsweise ist die Karte robust (keine Unsicherheiten, keine Ausreißer, alle relevanten statischen Informationen enthalten), aber nicht unbedingt aktuell und korrekt. Das sensorbasierte Umfeldmodell ist nicht unbedingt robust, aber aktuell. In Abb. 1.22 sind die beiden Umfeldmodelle anhand einiger Kriterien gegenübergestellt.

Es ist anzunehmen, dass sich mit der weiteren Entwicklung der Sensorik der Schwerpunkt zukünftig weiter in Richtung des sensorbasierten Umfeldmodells verschieben wird.

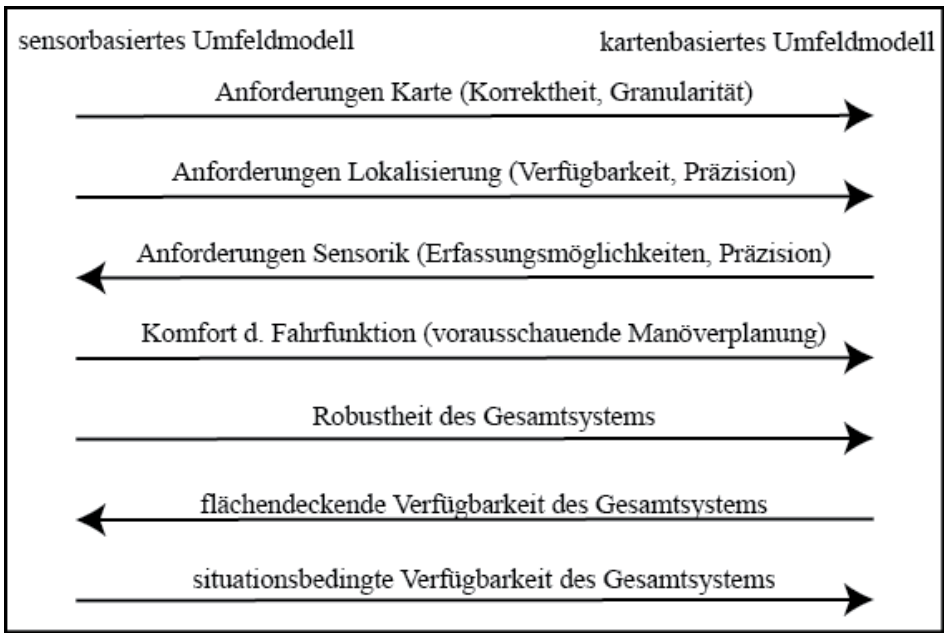


Abbildung 1.22: Die Eigenschaften von einem sensorbasierten Umfeldmodell im Vergleich mit einem kartenbasierten. Die Anforderungen an Karte und Lokalisierung nehmen zu, je mehr Anteile des Modells aus der Karte stammen; im Gegenzug erhöhen sich bei einem sensorbasierten Modell die Anforderungen an die Sensorik. Die Verwendung einer Karte erhöht den Fahrkomfort und die Robustheit des Systems. Der Einsatz beschränkt sich jedoch auf diejenigen Gebiete, die zuvor kartiert worden sind.

1.3 Wissenschaftlicher Beitrag und Gliederung der vorliegenden Arbeit

Wie eingangs bereits erwähnt, ist das automatisierte Fahren zum derzeitigen Zeitpunkt nur mit einem Umfeldmodell aus einer hochgenauen Karte möglich. Die Verwendung eines kartenbasierten Umfeldmodells funktioniert allerdings nur so lange, wie die Karte korrekt ist. Stimmt die Karte nicht mehr mit der Realität überein, versagt das System und es kann zu einem Unfall kommen.

Weitere Gründe für die Notwendigkeit der Verwendung eines sensorbasierten Umfeldmodells sind:

- Fehlende, falsche oder unpräzise Lokalisierung,
- Karte der aktuellen Strecke nicht verfügbar,
- Überbrückung der Fahrerübernahmezeit bei einem kritischen Ereignis,
- Möglichkeit der Validierung der Karte.

Insbesondere die letzten beiden Punkte haben in der Forschung und Entwicklung zum automatisierten Fahren bisher wenig Beachtung gefunden. Im Falle einer Automatisierung im SAE Level 3 (siehe Kap. 1.1) stellt der Fahrer die Rückfallebene dar, falls das System versagt. Bei einer kritischen Situation, wie beispielsweise einer ungültigen Karte aufgrund einer Baustelle, wird der Fahrer aufgefordert, die Fahraufgabe wieder zu übernehmen. Dabei wird ihm ein entsprechender Zeitrahmen eingeräumt, um die aktuelle Situation zu erfassen und die Fahraufgabe wieder aktiv zu übernehmen. Hier stellt sich zunächst die Frage, wie das Fahrzeug sich direkt nach der Fahrerübernahmeaufforderung verhalten soll, solange der Fahrer die Fahraufgabe noch nicht übernommen hat. Ein weiterer Punkt ist das Verhalten des Fahrzeugs, falls der Fahrer nicht innerhalb der vorgegebenen Zeit die Fahraufgabe übernommen hat. In [98] werden diese beiden Fragestellungen diskutiert und ein Ansatz vorgestellt, der garantiert, dass das Fahrzeug jederzeit innerhalb der vorgegebenen Zeitspanne der Fahrerübernahme zum vollständigen Stillstand gelangt. In dieser Zeitspanne muss das Fahrzeug jedoch - trotz in diesem Beispiel ungültiger Karte - sicher automatisiert weiterfahren können bis es zum Stillstand kommt. Abgesehen von dieser Schwierigkeit ist ein Stillstand im laufenden Verkehr (bspw. auf der Autobahn) auf einem gültigen Fahrstreifen ein riskantes Manöver und in der Praxis daher nicht anwendbar. Für einen deutlich praktikableren Stillstand auf dem Standstreifen oder Straßenrand muss das System allerdings das Umfeld kennen, obwohl - wie im Beispiel - die Karte aufgrund einer Baustelle ungültig ist. Daher ist es sinnvoll, sich mit dem in Abschn. 1.2.2 vorgestellten sensorbasierten Umfeldmodell näher zu befassen, da es sowohl zur Validierung der Karte als auch zur Überbrückung der Fahrerübernahmezeit verwendet werden kann.

Es existieren bereits einige Lösungen für Teilprobleme des sensorbasierten Umfeldmodells. Da das Thema der vorliegenden Arbeit die Generierung eines Straßenmodells behandelt, werden im Folgenden nur die für dieses Problem relevanten Ansätze vorgestellt. Der naheliegendste Ansatz, wenn es um die Erkennung von Fahrstreifen geht, ist die Erkennung und Verarbeitung von Fahrstreifenmarkierungslinien. Bis dato existieren dafür verschiedene Ansätze. Ein umfassender Überblick ist in [99] zu finden, beschränkt sich allerdings auf die Erkennung der weißen Linien in Kamerabildern. Darüber hinaus können nicht nur Kamerabilder, sondern auch Messdaten von Laserscannern zur Erkennung der Markierungslinien verwendet werden [100]. Auf der Basis von erkannten Fahrstreifenmarkierungen kann prinzipiell ein Straßenmodell erstellt werden [101, 102, 103].

Die Erzeugung eines Straßenmodells aus den erkannten Fahrstreifenmarkierungslinien hat jedoch diverse Nachteile:

- Eine Kamera alleine ist nicht robust genug (Gegenlicht durch die Sonne,

Spiegelungen auf nasser Fahrbahn oder Teerfugen führen zu Fehldetektionen).

- Prinzipiell kann eine LiDAR-basierte Fahrstreifenerkennung zur Erhöhung der Robustheit verwendet werden [100]. Dies setzt allerdings speziell reflektierende Markierungslinien voraus, die nicht flächendeckend eingesetzt sind.
- Komplizierte und teilweise verkehrte Linienführungen in Baustellen, sowie die parallele Existenz von weißen und gelben Markierungen stellen eine enorme Herausforderung an die Algorithmen zur Erkennung der „richtigen" Linien dar.
- Insbesondere im städtischen Bereich existieren viele Straßen, auf denen es keine Markierungen gibt.

Neben der Erkennung von Fahrstreifen existieren etliche Ansätze zur Schätzung der Randbebauung. Üblicherweise wird hierzu eine Belegungskarte der Fahrzeugumgebung als Ausgangsbasis verwendet, um anschließend aus den belegten Zellen des Gitters die Randbebauung zu extrahieren [104, 105, 106].

Bei den Ansätzen zur Erkennung der Randbebauung besteht jedoch grundsätzlich das Problem, dass das Ergebnis besagt, wo das autonome Fahrzeug nicht fahren kann. Jedoch gibt es in den meisten Fällen keine weitere Information darüber, wo tatsächlich gefahren werden sollte. Dies ist jedoch die wesentlich wichtigere Information. Darüber hinaus ist der Rand eines Fahrstreifens oder des befahrbaren Bereichs nicht zwangsläufig durch eine Randbebauung gekennzeichnet. Oftmals bilden Bürgersteige oder Grasnarben den Rand der Fahrbahn. Zusätzlich gibt die Randbebauung keinerlei Aufschluss darüber, wie die einzelnen Fahrstreifen auf der Fahrbahn verlaufen. Für autobahnähnliche Straßen kann die Randbebauung dennoch von Nutzen sein, da die Fahrbahn meist durch eine Randbebauung begrenzt ist und der Fahrstreifenverlauf sich meist an der Randbebauung orientiert.

Fahrstreifen können auch anhand von erkannten dynamischen Objekten erkannt werden. Allerdings bleibt die Frage, wie daraus ein konsistentes Straßenmodell erzeugt werden kann, bei den existierenden Ansätzen unbeantwortet [107].

Generell ist von der Idee abzuraten, ein Straßenmodell ausschließlich auf Basis anderer Verkehrsteilnehmer zu erzeugen, denn dabei ergeben sich unter anderem die folgenden Probleme:

- Es ist nicht immer ein Fahrzeug in Sichtweite der Sensoren
- Es können immer nur diejenigen Fahrtrichtungen und Fahrstreifen erkannt werden, auf denen sich ein Objekt bewegt.

- Biegt ein Fahrzeug ab, kann es dazu führen, dass die Fahrtrichtung „geradeaus" nicht erkannt wird und die Kreuzung fälschlicherweise als normale Kurve angesehen wird.
- Bei einem vorausfahrenden Fahrzeug kann nicht unterschieden werden, ob es auf demselben Fahrstreifen fährt, wie das Ego-Fahrzeug, oder auf einem benachbarten. Daher ist ein eindeutiger Schluss auf den Ego-Fahrstreifen nicht möglich.

Daher widmet sich diese Arbeit der Fragestellung, wie für ein sensorbasiertes Umfeldmodell ein Straßenmodell erzeugt und bereitgestellt werden kann. Dabei wird ein Konzept vorgestellt, wie die Messdaten der verschiedenen Sensoren fusioniert werden können und wie daraus eine robuste Schätzung des Straßenmodells inkl. aller Fahrstreifen generiert werden kann. Die Arbeit fand im Rahmen des Projektes zum automatisierten Fahren bei der BMW Group statt [36]. Im Gegensatz zu den meisten anfangs vorgestellten Forschungsarbeiten wird in diesem Projekt auf große externe Sensoren, die sich oftmals auf dem Fahrzeugdach befinden, verzichtet, um einerseits dem Design-Anspruch und andererseits der Alltagstauglichkeit gerecht zu werden.

Um eine nahtlose Integration des entwickelten Ansatzes in das Software-System des automatisierten Fahrens und der Fahrerassistenz zu gewährleisten, sollen die folgenden Anforderungen erfüllt werden:

- Erstellung eines Straßenmodells mit Fahrstreifen, deren Verlauf und deren Begrenzungen.
- Die Architektur des Ansatzes muss flexibel und modular sein, damit ein Einsatz auch mit unterschiedlichen Sensoren und Sensorkonfigurationen auf verschiedenen Fahrzeugplattformen möglich ist.
- Eine explizite Beschränkung auf das automatisierte Fahren soll nicht gegeben sein, da auch heutige Fahrerassistenzsysteme von einem sensorbasierten Straßenmodell enorm profitieren würden.
- Die Anforderungen an die Sensorik zur Umfeldwahrnehmung sollen möglichst allgemeingültig sein, um den Einsatz verschiedener und redundanter Sensoren und deren Messprinzipien zu ermöglichen.
- Die entwickelten Ansätze können sowohl im Realbetrieb in einem entsprechenden Versuchsfahrzeug als auch mit aufgezeichneten Realdaten getestet und bewertet werden.
- Die verwendete Sensorik orientiert sich an der Serientauglichkeit und Alltagstauglichkeit. Sofern möglich, sollte die Sensorik dem entsprechen, was

in absehbarer Zeit Stand-der-Technik im Bereich Fahrerassistenz/automatisiertes Fahren sein wird und Einzug in den Massenmarkt haben wird. Dies schließt teure und unpraktikable Dachaufbauten aus.

- Der Ansatz ermöglicht eine Straßenschätzung in der gesamten näheren Fahrzeugumgebung. Dazu zählt insbesondere auch der Bereich hinter dem Ego-Fahrzeug, da dieser für Fahrstreifenwechsel relevant ist. Darüber hinaus müssen Kreuzungen (oder Teile davon) - sofern sie im Sichtbereich der Sensorik liegen - im Straßenmodell enthalten sein.
- Die Algorithmen müssen in „Echtzeit" ablaufen, wobei der Begriff hier als Beschreibung dafür dient, dass die Laufzeit kleiner als die Wiederholungsrate der Sensoren sein muss.

Neue wissenschaftliche Beiträge der Arbeit

In der vorliegenden Arbeit wird ein neuer Ansatz vorgestellt, um aus den im Fahrzeug verfügbaren Sensor-Messdaten ein Straßenmodell zu erzeugen. Dabei werden zunächst die Messdaten der verschiedenen im Fahrzeug verbauten Sensoren in einem gridbasierten Umfeldmodell (DSTMap) fusioniert und zeitlich akkumuliert. Im Gegensatz zu existierenden Ansätzen (bspw. [108, 96, 109]) ist das hiesige gridbasierte Modell so aufgebaut, dass es die statische Straßeninfrastruktur abbildet. In der DSTMap werden anschließend iterativ Pfade geplant, um potentielle Fahrstreifen zu finden. Die gefundenen Pfade werden zusammen mit der DSTMap verwendet, um für jeden potentiellen Fahrstreifen die relevanten Fahrstreifengrenzen zu identifizieren. Anschließend wird daraus ein konsistentes Straßenmodell erstellt. Abb. 1.23 zeigt eine schematische Übersicht des Ansatzes.

Nachfolgend sind die wichtigsten wissenschaftlichen Neuerungen des Ansatzes in einer Liste kurz zusammengefasst.

- Extraktion eines vollständigen Straßenmodells aus Sensordaten inkl. Nachbar-Fahrstreifen ohne a-priori-Annahmen bzgl. Fahrstreifenparallelität, Geometrie oder Verlauf.
- Neues gridbasiertes Umfeldmodell mit allen relevanten Elementen der Straßeninfrastruktur.
- Fusion von allen verfügbaren Sensordaten in eine einheitliche gridbasierte Darstellung.
- Verwendung einer iterativen Pfadplanung, die statt zur Bewegungsplanung, als Mittel zur Suche nach Fahrstreifen verwendet wird.

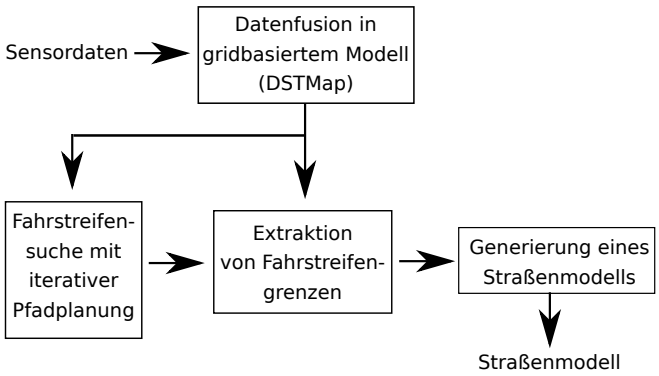


Abbildung 1.23: Die Sensordaten werden in einem gridbasierten Umfeldmodell (DSTMap) fusioniert. Anschließend werden mit Hilfe einer Pfadplanung potentielle Fahrstreifen gesucht. Die DSTMap wird zusammen mit den gefundenen Pfaden zur Extraktion von Fahrstreifengrenzen verwendet. Anschließend wird daraus ein konsistentes Straßenmodell erstellt.

Gliederung der Arbeit

Nachdem in Kapitel 1.2 bereits das karten- und sensorbasierte Umfeldmodell erläutert worden ist, folgt in Kapitel 2 die Vorstellung der neuartigen gridbasierten Umfeldmodellierung.

Kapitel 3 widmet sich der Frage, wie in der DSTMap mit Hilfe einer Pfadplanung potentielle Fahrstreifen gefunden werden können. Anschließend wird ebenfalls in diesem Kapitel die Extraktion von Fahrstreifengrenzen erläutert.

Daran anknüpfend wird in Kapitel 4 die Erstellung des Straßenmodells dargestellt.

In Kapitel 5 folgt eine Evaluierung und Bewertung des hier vorgestellten Ansatzes. Dabei werden die erzielten Ergebnisse qualitativ und quantitativ bewertet.

Zuletzt gibt Kapitel 6 eine Zusammenfassung und einen Ausblick auf mögliche zukünftige Erweiterungen und Abwandlungen des hier vorgestellten Ansatzes.

Die Kapitel 2, 3 und 4 sind strukturell ähnlich aufgebaut. Sie beinhalten jeweils einen Abschnitt mit dem aktuellen Stand der Technik, der für das jeweilige Thema relevant ist. Darauf folgt eine kurze Beschreibung der neu entwickelten Methodik, sowie die Erläuterung des wissenschaftlichen Beitrags. Am Ende dieser Kapitel findet sich jeweils ein Abschnitt mit Ergebnissen, sowie eine Zusammenfassung.

Kapitel 2

Gridbasierte Umfeldmodellierung

Das folgende Kapitel stellt das gridbasierte Umfeldmodell vor. Dabei werden die von den Sensoren gemessenen Daten in ein gridbasiertes Modell überführt, das die statische Straßeninfrastruktur widerspiegelt.

Das Kapitel beginnt zunächst mit der Beschreibung des Konzepts einer Belegungskarte. Diese war ursprünglich die erste Form der gridbasierten Umfeldmodellierung. Daran anschließend folgt eine Übersicht über den Stand der Technik zur gridbasierten Umfeldmodellierung. Anschließend wird der in dieser Arbeit neu-entwickelte Ansatz auf Basis der Dempster-Shafer-Theorie vorgestellt.

2.1 Einleitung

In vielen Situationen kann eine Umfelderkennung auf Basis geometrischer Modelle/Objekte verwendet werden, um das Fahrzeugumfeld abzubilden. Dabei werden die Objekte - ganz gleich welcher Art - als einfache geometrische Objekte mit variablen Parametern dargestellt. Als geometrische Objekte für Fahrzeuge können beispielsweise Quader mit Größe, Position und Geschwindigkeit verwendet werden. Für Fahrstreifenverläufe bieten sich Polynome, Klothoiden oder Splines an. In komplexeren Situationen, insbesondere dann, wenn die Art der Objekte zunächst nicht zuverlässig feststellbar ist, sind diese Modelle für die Umfelderkennung nicht immer ausreichend. Auch bei einer plötzlichen Änderung des Objekttyps (bspw. Wechsel von statischem zu dynamischem Objekt), muss in so einem Fall auch das zugrundeliegende mathematische Modell plötzlich geändert werden. Auch komplexe Alltagssituationen wie belebte Innenstadtkreuzungen stellen für Modelle, die auf geometrischen Objekten basieren, eine große Schwierigkeit dar.

Um diese Situationen optimal modellieren zu können, bietet es sich an, die Rohdaten aus der im Fahrzeug verbauten Sensorik zunächst in eine sensor-unabhängige Zwischenrepräsentation zu überführen, bevor höhere Algorithmen/ Verfahren wie eine Objektrekonstruktion oder eine Fahrstreifenerkennung angewandt werden. Eine mögliche Zwischenrepräsentation der Sensor-Rohdaten sind sogenannte Belegungskarten, die im Folgenden vorgestellt werden.

Belegungskarten stellen einen universellen Lösungsansatz für die Modellierung der Fahrzeugumgebung dar. Dabei wird die Fahrzeugumgebung durch eine endliche Anzahl an Gitterzellen diskretisiert und es entsteht ein zweidimensionales, planares Gitter. Aus den Messdaten, die von den im Fahrzeug verbauten Sensoren stammen, wird für jede Zelle des Gitters die Information abgeleitet, ob diese Zelle frei oder belegt ist. Jede Zelle in diesem Gitter enthält Informationen darüber, inwieweit die Zelle zu einem Hindernis oder zu Freiraum gehört. Da Messdaten jeder Form immer mit einer gewissen Unsicherheit behaftet sind, behandelt man die gemessenen Informationen grundsätzlich als hypothetisch und drückt dies über eine Wahrscheinlichkeit aus.

Die Belegung einer Zelle in einer Belegungskarte kann verschiedene Ursachen haben. Zellen können einerseits durch statische Objekte, wie Häuser, Straßenbebauung (Leitplanken, Bürgersteige, etc.), parkende Autos und andererseits auch durch dynamische Objekte (Fahrzeuge, Fußgänger, Fahrradfahrer, Tiere) belegt sein. Belegungskarten in ihrer ursprünglichen Form reduzieren die Umfeldwahrnehmung daher zunächst auf die Fragestellung, ob die Umgebung frei oder belegt ist - unabhängig davon, wodurch die Zelle ggfs. belegt ist.

Vermutlich ist das Konzept der Belegungskarten erstmals in [95] unter der Bezeichnung „Probabilistic Local Map" im Zusammenhang mit ultraschallbasierten Sensordaten von autonomen mobilen Robotern erschienen. Eine spätere Publikation des gleichen Autors [96] verwendet die Bezeichnung „Occupancy Grids" einschließlich der Definition eines Occupancy Grids „The occupancy grid is a multidimensional random field that maintains stochastic estimates of the occupancy state of the cells in a spatial lattice".

Abbildung 2.1 zeigt eine solche Belegungskarte mit der dazugehörigen Verkehrssituation. Für die rekursive Schätzung des Belegungszustandes wurde ein Bayes-Filter verwendet (siehe auch [97]).

Belegungskarten haben sich mittlerweile zum Quasi-Standard entwickelt, wenn es darum geht, das Fahrzeugumfeld hinsichtlich belegter und freier Bereiche zu analysieren. Über die Zeit haben sich diverse Abwandlungen entwickelt, die weitere Aspekte wie bspw. die Unterscheidung zwischen statischer und dynamischer Belegung, oder die Optimierung hinsichtlich Rechenzeit und Speicherverbrauch betrachten. Diese werden im Folgenden kurz vorgestellt.

(a) (b)

Abbildung 2.1: Video-Bild aus dem Fahrzeug und Belegungskarte, erzeugt mit einem Bayes-Filter. Die Wahrscheinlichkeit für die Belegung einer Zelle ist durch die Graustufen dargestellt. Dabei steht weiß für eine Wahrscheinlichkeit von 0 (Zelle nicht belegt) und schwarz für eine Wahrscheinlichkeit von 1 (Zelle ist belegt).

2.1.1 Stand der Technik

Seit der Veröffentlichung in [96] haben sich diverse Abwandlungen der Belegungskarten entwickelt. So finden sich in [110] Optimierungen hinsichtlich des Speicherverbrauchs und der Rechenzeit durch geeignete Kompressionsverfahren, oder in [111] eine andere Art der Datenrepräsentation durch sog. kd-Bäume, um den Speicherbedarf zu senken.

In der ursprünglichen Form der Belegungskarte mit Bayes-Filter [96] findet sich keine Möglichkeit, zwischen Bereichen zu unterscheiden, die noch nicht von einem Sensor erfasst worden sind, und denjenigen Bereichen, die aufgrund von widersprüchlichen Messungen einen unklaren Zustand besitzen. In beiden Fällen besitzt die Zelle den numerischen Wahrscheinlichkeitswert $p = 0.5$. Daher findet sich in [112] eine explizite Modellierung von *unbekannten* Bereichen in der Belegungskarte. Unbekannte Bereiche sind Bereiche, die von keinem Sensor erfasst werden - einerseits wegen Verdeckungen, andererseits wegen einer begrenzten Reichweite der Sensoren. Ebenfalls problematisch beim Bayes-Filter ist die Annahme einer statischen Welt (Markov „static world assumption"). Das Fahrzeugumfeld beim automatisierten Fahren ist jedoch keineswegs statisch. Durch die Annahme einer statischen Welt verursachen dynamische Objekte Artefakte in den Belegungskarten. Diese entstehen, weil die Zellen, auf denen sich das dynamische Objekt zu einem Zeitpunkt befindet, als belegt und damit als statisch belegt angesehen werden. Im weiteren Verlauf bewegt sich das Objekt jedoch weiter, die ursprünglich belegten Zellen werden frei und andere Zellen

werden belegt. Dies führt insbesondere bei Situationen mit direkt vorausfahrenden Fahrzeugen zu statischen Hindernissen unmittelbar vor dem Ego-Fahrzeug. Eine Manöverplanung würde nun entweder ein Ausweichmanöver oder eine Notbremsung einleiten.

Zu Lösung dieses Problems sind verschiedene Ansätze entwickelt worden, um dynamische Objekte in Belegungskarten korrekt zu behandeln. Eine beliebte Methode ist die Analyse von sich widersprechenden Messungen über die Zeit. Dabei wird der Wechsel einer Zelle von „frei" auf „belegt" und vice versa beobachtet und daraus abgeleitet, dass dieser Wechsel durch ein dynamisches Objekt verursacht worden sein muss [113, 114]. Die „Dempster-Shafer Theory of Evidence" bietet hierfür sogar eine explizite Möglichkeit, mit dieser Art von Konflikten umzugehen und dies entsprechend zur Erkennung von dynamischen Objekten zu nutzen [115, 116]. Andere Ansätze zur Schätzung der dynamischen Anteile einer Belegungskarte lassen sich auch in [117, 118] finden. Dort wird der Bayesian Occupancy Filter (BOF) vorgestellt - ein gridbasierter 4-dimensionaler Ansatz zur Schätzung der Belegtheit (x, y) und der Geschwindigkeit (v_x, v_y). Abwandlungen von dieser Methode sind unter anderem in [119] beschrieben. Auch partikel-basierte Methoden zur Dynamik-Schätzung in Belegungskarten finden sich in der Literatur [120, 121, 122].

Darüber hinaus existieren etliche Ansätze, um aus Belegungskarten einzelne Elemente des Umfeldmodells wie bspw. Freiraum oder die Randbebauung zu extrahieren [104, 105, 106, 123]. Sie bauen meist auf den „klassischen" Belegungskarten auf und sind in Bezug auf das Straßenmodell nur von begrenztem Nutzen.

Bezogen auf die Erstellung eines Straßenmodells aus Sensordaten lassen sich bisher nur wenige Ansätze finden, die mit einer grid-basierten Repräsentation arbeiten.

Eine der wenigen Arbeit ist [124], wo neben der Unterscheidung von statischen und dynamischen Objekten auch fahrbare und nicht-fahrbare Bereich in einer gridbasierten Repräsentation geschätzt werden. Darüber hinaus gehen die Daten einer Offline-Karte in das Endergebnis ein. Der Ansatz kombiniert damit die zwei grundverschiedenen Probleme der Dynamik-Statik-Schätzung und der Straßenmodellschätzung, ist jedoch schon im Ansatz inkonsistent und verletzt die Annahme der sich ausschließenden Hypothesen (Exklusivität) bei der Dempster-Shafer-Theorie. Die Wahl der Hypothesenmenge

$$\Omega := \{D, N, I, U, S, M\} \tag{2.1}$$

mit

$\{D\}$: Drivable space

$\{N\}$: free Non-drivable space
$\{I\}$: mapped Infrastructure (buildings)
$\{U\}$: Unmapped infrastructure
$\{S\}$: temporarily Stopped objects
$\{M\}$:mobile Moving objects

führt zu dazu, dass sowohl die Hypothese „Drivable space" als auch „mobile Moving objects" korrekt sein kann - man denke an ein vorausfahrendes Fahrzeug auf demselben Fahrstreifen. Der Bereich im Grid, der vom Fahrzeug überdeckt ist, zählt zum Fahrstreifen und ist grundsätzlich befahrbar. Für eine Manöverplanung ist dieses Wissen eine zwingende Voraussetzung - andernfalls wäre der Planungshorizont begrenzt durch das Vorderfahrzeug und würde zu unerwünschten Reaktionen wie bspw. einer Vollbremsung führen.

Zwar existieren Ansätze, die sich damit auseinandersetzen, wie bspw. [125], die Verletzung der Exklusivität wird in [124] jedoch weder erwähnt noch thematisiert. Auch die Kombination des Wissens aus der Offline-Karte mit den Sensordaten ist zweifelhaft. So stellt sich beispielsweise die Frage, inwiefern die Überführung des 'belief' von „intermediate space (e.g. sidewalks)" aus der Offline-Karte in die Hypothesen „drivable space, free non-drivable space, temporarily stopped object, mobile moving object, unmapped infrastructure" sinnvoll und korrekt ist. Darüber hinaus gibt es in der Publikation bei der Verarbeitung der Daten keine Unterscheidung zwischen „Drivable space" und „free Non-drivable space". Dies ist jedoch elementar wichtig und eine der anspruchsvollsten Aufgaben des Umfeldmodells. Klammert man diese Frage implizit durch Nicht-Unterscheiden bei den Messdaten aus, gibt es keinen validen Grund für eine Trennung in zwei Hypothesen.

Der in diesem Kapitel vorgestellte gridbasierte Ansatz ist durch die Arbeit in [126] inspiriert, wo ein „centerline evidence image" konstruiert wird. Allerdings unterscheidet sich der in dieser Arbeit entwickelte Ansatzs sowohl von Art und Methodik, als auch von der Sensordatenfusion her grundsätzlich von dem in [126].

Da im weiteren Verlauf dieser Arbeit zur Modellierung von Unsicherheiten die Dempster-Shafer Theory of Evidence verwendet wird, gibt der folgende Abschnitt eine kurze Zusammenfassung der Theorie und ihrer wichtigsten Eigenschaften.

2.1.2 Dempster-Shafer Theory of Evidence

Mitte der 60er Jahre entwickelte Arthur P. Dempster eine mathematische Theorie, um Unsicherheiten zu modellieren. Im Gegensatz zur Wahrscheinlichkeitstheorie, die für jede Hypothese stets nur eine Wahrscheinlichkeit berücksichtigt, enthielt seine Theorie „obere" und „untere" Wahrscheinlichkeiten für jede Hypothese ([127], [128]). Später erweiterte G. Shafer in [129] die Theorie von Dempster und entwickelte daraus eine eigenständige mathematische Theorie - die „Dempster-Shafer Theory of Evidence". Zur Vermeidung von Missverständnissen ist zu beachten, dass die Übernahme des englischen Titels „theory of evidence" in das Deutsche als „Evidenztheorie" eine Gleichsetzung der Begriffe „evidence" und „Evidenz" beinhaltet, die sachlich und sprachlich nicht gerechtfertigt ist. Das englische Wort „evidence" ist nicht gleichbedeutend mit dem deutschen Begriff der Evidenz, der das Offensichtliche oder das Unbezweifelbare bezeichnet. Die korrekte Übersetzung des Begriffes „evidence" lautet „Hinweis/Nachweis/Anhaltspunkt/Indiz/Beweis".

Die Dempster-Shafer-Theorie kann als Verallgemeinerung der Bayes'schen Wahrscheinlichkeitstheorie angesehen werden. In [129] wird gezeigt, dass die Bayes'sche Wahrscheinlichkeitstheorie ein Spezialfall der Dempster-Shafer-Theorie ist.

Im Unterschied zur klassischen Wahrscheinlichkeitstheorie, bei der aus einer Wahrscheinlichkeit $p(x)$ für eine Hypothese direkt abgeleitet werden kann, dass die Wahrscheinlichkeit für das Gegenteil $p(\neg x) = 1 - p(x)$ ist, gilt dies bei der Dempster-Shafer-Theorie nicht.

Im Folgenden werden kurz die für diese Arbeit relevanten Inhalte der Dempster-Shafer-Theorie vorgestellt. Eine umfangreiche und didaktisch aufbereitete Abhandlung findet sich in [129].

Frame of Discernment

Das 'frame of discernment' (Wahrnehmungsrahmen) ist eine Menge Ω, deren Elemente alle möglichen Hypothesen θ_i des zu betrachtenden Systems darstellen:

$$\Omega = \{\theta_1, \theta_2, ..., \theta_N\}, \tag{2.2}$$

wobei genau eine Hypothese der Wahrheit entsprechen muss. Die Wahl der Hypothesen muss einerseits berücksichtigen, welche Zustände des Systems möglich sind, andererseits muss sie auch einbeziehen, welche Informationen und Hinweise zum Wahrheitsgehalt einer Hypothese aus den Eingabedaten/Messdaten ableitbar sind. Die Wahl der Hypothesen hängt damit einerseits davon ab,

was man über das zu betrachtende System erfahren möchte, und andererseits, welche Informationen man aus den Eingabedaten überhaupt ableiten kann.

Zur Betrachtung einzelner und kombinierter Hypothesen betrachtet man die Potenzmenge von Ω. Die Potenzmenge 2^Ω ist die Menge aller Teilmengen von Ω:

$$2^\Omega := \{U \mid U \subseteq \Omega\} \tag{2.3}$$

Zu beachten ist hier, dass die Potenzmenge sowohl die leere Menge $\varnothing$ beinhaltet, als auch die Menge Ω selbst. Die leere Menge entspricht der Behauptung, dass keine der in Ω enthaltenen Hypothesen wahr ist. Da dies nach Voraussetzung bei der Wahl von Ω nicht der Fall sein kann, ist diese Hypothese grundsätzlich falsch. Gegenteilig dazu verhält sich die Hypothese Ω. Da die wahre Hypothese in Ω nach Voraussetzung enthalten sein muss, ist die Hypothese Ω, die den Fall widerspiegelt, dass irgendeine der in Ω befindlichen Hypothesen wahr ist, stets wahr.

Mit der Verwendung der Potenzmenge gewinnt man eine Flexibilität, die insbesondere bei der Ableitung von Informationen aus den Eingabedaten (Messdaten) von großer Bedeutung sein kann. Beispielsweise ist ein Hinweis für eine Hypothese θ_1 ein Hinweis auf den Wahrheitsgehalt dieser Einzelhypothese, während hingegen ein Hinweis für $\{\theta_1, \theta_2\}$ ein Hinweis dafür ist, dass eine der beiden Hypothesen wahr ist, ohne dass dabei die Einzelhinweise von θ_1 und θ_2 berücksichtigt werden. Es ist somit auf elegante Weise möglich, Teilwissen in der Theorie abzubilden.

Belief Function

Wie auch die Bayes'sche Wahrscheinlichkeitstheorie verwendet die Dempster-Shafer-Theorie für die Modellierung von Hinweisen auf die Richtigkeit von Hypothesen 'belief functions'. Die Dempster-Shafer-Theorie behandelt die Möglichkeit, 'belief functions' für die Repräsentation von „partial belief" zu verwenden. Dabei wird der Ansatz der Bayes'schen Theorie zur Modellierung von 'belief' mit dem Ansatz der Repräsentierung von Teilwissen durch Verwendung der Potenzmenge der Hypothesenmenge kombiniert.

Eine 'belief function' auf einer Menge Ω, die durch die folgende Form gegeben ist

$$Bel : 2^\Omega \to [0, 1] \tag{2.4}$$

besitzt die folgenden Eigenschaften:

$$Bel(\varnothing) = 0 \tag{2.5}$$
$$Bel(\Omega) = 1 \tag{2.6}$$

Dabei werden die Elemente der Menge Ω als Hypothesen angesehen, von denen genau eine der Wahrheit entspricht. Für jede Teilmenge A von Ω kann der 'belief' $Bel(A)$ als Grad dafür angesehen werden, wie wahrscheinlich es ist, dass A der Wahrheit entspricht. 0 bedeutet dabei, dass es kein Vertrauen dafür gibt, dass die Hypothese wahr ist, 1 bedeutet, dass mit Sicherheit feststeht, dass die Hypothese wahr ist. Wichtig hierbei ist die Tatsache, dass es sich bei einem 'belief' **nicht** um eine Wahrscheinlichkeit handelt.

Die beiden Eigenschaften der 'belief function' spiegeln die Tatsache wider, dass eine der in der Hypothesenmenge befindlichen Hypothesen korrekt ist.

Basic Assignment Function

Die Idee der 'basic assignment function' (grundlegende Zuordnungsfunktion) basiert auf der Annahme, dass der gesamte 'belief' auf mehrere Hypothesen aufteilbar ist. Somit kann jeder Hypothese ein gewisser Grad des gesamten 'belief' zugeordnet werden, je nachdem wie sicher es ist, dass die Hypothese wahr ist.

Im Gegensatz zum Konzept des 'belief' bei der Bayes'schen Theorie bedeutet in der Dempster-Shafer-Theorie die Zuordnung eines Anteils eines 'belief' zu einer Hypothese nicht automatisch eine Zuordnung des restlichen 'belief' zur Gegenhypothese. Dies legt die Definition der folgenden Funktion nahe:

Definition: Die 'basic assignment function' m ist eine Zuordnung

$$m : 2^{\Omega} \rightarrow [0, 1], \tag{2.7}$$

für die gilt:

$$m(\varnothing) = 0 \tag{2.8}$$
$$\sum_{A \subseteq \Omega} m(A) = 1 \tag{2.9}$$

Dabei wird $m(A)$ als 'basic probability number' oder auch als 'belief mass' (Masse) bezeichnet. Diese kann als Anteil des 'belief' angesehen werden, der exakt der Hypothese A zugeordnet ist und gibt damit an, wie sicher es ist, dass exakt die Hypothese A wahr ist. Dabei kann A auch eine beliebige Teilmenge von 2^Ω sein. Wie bei den Wahrscheinlichkeiten in der Bayes'schen Theorie muss die Summe aller 'belief masses' 1 ergeben, da dies den gesamten 'belief' darstellt.

'Belief' und 'Plausibility'

Im Gegensatz zur 'belief mass' vom A steht der 'belief', $Bel(A)$, für den 'belief' dass A oder irgendeine Teilmenge von A wahr ist.

Um den gesamten 'belief', der einer Hypothese A zugeordnet ist, zu berechnen, müssen die 'belief masses' aller Teilmengen B von A summiert werden:

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{2.10}$$

Betrachtet man beispielsweise ein Frame of discernment $\Omega = \{A, B, C\}$, und die dazugehörige Potenzmenge $2^\Omega = \{A, B, C, \{A, B\}, \{A, C\}, \{B, C\}, \varnothing, \Omega\}$, ergibt sich exemplarisch für den belief

$$Bel(A) = m(A) \tag{2.11}$$
$$Bel(\{A, B\}) = m(A) + m(B) + m(\{A, B\}) \tag{2.12}$$

wobei $m(\{A, B\})$ die 'belief mass' für die Hypothesenmenge $\{A, B\}$ ist.

Darüber hinaus existiert in der Dempster-Shafer-Theorie eine weitere Größe, die als eine Art obere Schranke für die Wahrscheinlichkeit einer Hypothese angesehen werden kann. Sie wird als 'plausibility' bezeichnet und ist definiert als

$$Pl(A) = \sum_{B \cap A \neq \varnothing} m(B) \tag{2.13}$$

Die 'plausibility' gibt an, in welchem Maße der 'belief' der gegensätzlichen Hypothese $\neg A$ Spielraum lässt für den 'belief' von A. Entsprechend lässt sich für die 'plausibility' auch schreiben

$$Pl(A) = \sum_{B \cap A \neq \varnothing} m(B) \tag{2.14}$$

$$= 1 - Bel(\neg A) \tag{2.15}$$

Kombination von 'belief functions'

Verschiedene 'belief functions' mit unterschiedlichen 'basic probability assignments' können unter gewissen Voraussetzungen miteinander kombiniert werden. Das Ergebnis der Kombination ist dabei eine neue 'belief function', die das „Wissen" der beiden 'belief functions' beinhaltet.

Die erste vollständige Formulierung der Kombination von 'belief functions' stammt von Arthur Dempster aus dem Jahr 1967 ([127]) und wird als „Dempster's Rule of Combination" bezeichnet.

Definition: Gegeben sind zwei 'belief functions' Bel_1 und Bel_2 über dem gleichen 'frame of discernment' Ω mit den dazugehörigen 'belief masses' m_1 und m_2 sowie den Hypothesen $A_1, \ldots, A_n$ und $B_1, \ldots, B_n$. Dann ist die Funktion

$$m(A) = \frac{1}{1-k} \sum_{A_i \cap B_j = A} m_1(A_i) m_2(B_j),\ A \neq \varnothing \tag{2.16}$$

$$m(\varnothing) = 0 \tag{2.17}$$

mit

$$k = \sum_{A_i \cap B_j = \varnothing} m_1(A_i) m_2(B_j) \tag{2.18}$$

eine neue 'belief function', die eine Kombination der beiden 'belief functions' Bel_1 und Bel_2 darstellt. Sie wird auch als „Dempster's Rule of Combination" bezeichnet.

Gl. 2.16 kombiniert alle 'belief masses', die gemeinsame Teilmengen besitzen und damit gleiche Hypothesen unterstützen. Anschließend wird die Summe mit einem Vorfaktor normiert. Dabei ist k ein Maß dafür, wie stark die zu kombinierenden 'belief functions' im Konflikt zueinander stehen und damit widersprüchliche Hypothesen unterstützen. k ist die Summe aller Produkte zweier 'belief masses' der beiden 'belief functions', die keine gemeinsame Teilmenge besitzen. Jeder Summand ungleich 0 steht dabei für einen Konflikt. Im Prinzip

werden somit alle 'belief masses', die gleiche Hypothesen unterstützen, verrechnet und anschließend die Konflikte gleichmäßig auf alle übrigen 'belief masses' aufgeteilt.

In [129] wird explizit darauf hingewiesen, dass diese Kombinationsregel nur dann gilt, wenn $k < 1$ ist. Für den Fall, dass die beiden 'belief functions' sich stark widersprechen, kann die Kombinationsregel von Dempster nicht verwendet werden.

In [130] findet sich neben einer ausführlichen Darstellung dieser Problematik auch eine alternative Regel zur Kombination von 'belief functions', die dieses Problem löst, indem die Konflikte nicht auf alle übrigen 'belief masses' aufgeteilt werden, sondern vollständig der Hypothese Ω zugeordnet werden. Dies folgt der Annahme, dass ein Konflikt gleichbedeutend mit Nichtwissen ist.

Jedoch wird erst in [131], [132] bewiesen, dass Dempster's Regel nicht korrekt ist bei „cumulative or averaging belief fusion", sondern eine Methode für die serielle Kombination ist (serial combination of stochastic constraints). Dies ist insbesondere bei der zeitlichen Akkumulation von aufeinanderfolgenden Messungen entscheidend und wird in einem späteren Abschnitt des Kapitels von Relevanz sein.

2.1.3 Eigener Ansatz und wissenschaftlicher Beitrag

In diesem Kapitel wird ein neuer gridbasierter Ansatz in Form der DSTMap vorgestellt, der alle statischen, für eine darauffolgende Manöverplanung relevanten Elemente der Straßeninfrastruktur modelliert. Ein Schwerpunkt liegt dabei auf der Repräsentation von Fahrstreifen, da diese in einem späteren Kapitel aus der DSTMap extrahiert werden.

Gridbasierte Modellierungen stellen eine universelle modellfreie Repräsentation der Messdaten dar, in der die verschiedenen Informationen in geeigneter Weise fusioniert werden können. Im vorliegenden Fall werden die Messdaten verschiedener im Fahrzeug verbauter Sensoren verwendet, um sie zu fusionieren und damit die DSTMap zu generieren. Dabei werden keine a-priori-Annahmen oder Einschränkungen bzgl. des Verlaufs oder der Form von Fahrbahn oder Fahrstreifen getroffen.

Die wichtigsten Punkte des neuen Ansatzes sind in nachfolgender Liste zusammengestellt:

- Erstmalige Verwendung eines gridbasierten Ansatzes zur Abbildung der statischen Straßeninfrastruktur und zur Extraktion von Fahrstreifen. Bisherige gridbasierte Ansätze beschränkten sich auf die Modellierung statischer Hindernisse oder dynamischer Objekte.

- Ein neues 'frame of discernment' der Dempster-Shafer-Theorie wird definiert, um die verschiedenen Arten von statischen Straßeninfrastrukturelementen zu modellieren, die für eine spätere Pfad- oder Manöverplanung relevant sind.
- Interpretation der Sensordaten hinsichtlich ihrer *semantischen* Bedeutung in Bezug auf das Straßenmodell.
- Der Ansatz ist flexibel und sehr einfach durch zusätzliche Straßeninfrastrukturelemente erweiterbar, falls eine weitere Differenzierung der Modellierung notwendig sein sollte.
- Erstmalige Fusion von allen zur Verfügung stehenden Messdaten für die Erstellung einer konsistenten gridbasierten Darstellung der Fahrzeugumgebung in Bezug auf die Straßeninfrastruktur.
- Erstmals die Verwendung von Daten der semantischen Segmentierung von Kamerabildern in einer gridbasierten Repräsentation unter Berücksichtigung der Messunsicherheiten.
- Keine a-priori-Annahmen über die Geometrie von Fahrstreifen. Es wird ebenfalls keine Fahrstreifenparallelität vorausgesetzt. Somit eignet sich der Ansatz nicht nur für autobahnähnliche Gebiete, sondern auch für innerstädtische Verkehrssituationen.
- Durch die Verwendung von Grids wird eine einheitliche Datenrepräsentation der verschiedenartigen Sensordaten erreicht, die eine konsistente Fusion von modellfreien und modell-behafteten Sensordaten ermöglicht.
- Keine Annahme über Größe, Form oder andere spezielle Charakteristika von gemessenen Objekten.
- Der Ansatz ist unabhängig von speziellen Features oder einem speziellen Sensor.
- Die gridbasierte Darstellung bietet flexible Möglichkeiten, Daten über die Zeit zu akkumulieren. Sie bilden eine gute Grundlage für darauf aufbauende Extraktoren.
- Durch die Verwendung der Dempster-Shafer-Theorie kann Teilwissen aus den Sensordaten korrekt in der DSTMap abgebildet und verwendet werden. Dies ist insbesondere im Hinblick auf die Fusion der Daten verschiedener Sensoren von großem Vorteil, da nicht jeder Sensor alle Hypothesen unterscheiden kann, sondern u.U. nur Gruppen/Mengen von Hypothesen.

Nachfolgend wird zunächst die gridbasierte Umfeldmodellierung mit Hilfe der Dempster-Shafer-Theorie vorgestellt (Abschn. 2.2). Dazu werden zunächst die Sensor-Messdaten in sog. *ScanGrids* transformiert (Abschn. 2.2.1) und anschließend fusioniert und über die Zeit akkumuliert. In Abschnitt 2.3 werden die Ergebnisse präsentiert und in Abschnitt 2.4 wird eine Zusammenfassung des Kapitels gegeben.

2.2 Umfeldmodellierung mit der DSTMap

Das eingangs erklärte Konzept der Belegungskarten ist sehr flexibel und erlaubt eine gute Erweiterung. Darüber hinaus kann die Karte als universelle Repräsentation der Fahrzeugumgebung benutzt werden, die auch von verschiedenen Sensorsystemen fortlaufend aktualisiert wird und somit als Grundlage für eine Sensordaten-Fusion dienen kann.

Bereits in der Ursprungspublikation [96] wird die Möglichkeit diskutiert, statt einer binären Zufallsvariablen einen Zufallsvektor für jede Zelle zu speichern, der weitere Informationen über die Zelle beinhaltet. Diese Art der Karten wird auch als „inference grid" bezeichnet. Darüber hinaus existieren die bereits in Abschnitt 2.1.1 erwähnten Erweiterungen der gridbasierten Umfeldmodellierung auf Basis der Dempster-Shafer Theorie. Ausgehend von einem sensorbasierten Umfeldmodell, wie in Kapitel 1 beschrieben, ist es zwar möglich, die Erstellung eines solchen als ein einziges Gesamtproblem zu formulieren. Aufgrund der Komplexität und Dimensionalität des Zustandsraumes ist die Lösung dieses Problems bis dato aber nicht in angemessener (Rechen-)zeit durchführbar. Daher beschränkt sich die DSTMap auf die Modellierung der statischen Straßeninfrastruktur ohne Berücksichtigung der dynamischen Anteile, wie bspw. andere Verkehrsteilnehmer oder Ampelphasen.

Betrachtet man die Topologie der Straße, ergibt sich zunächst ein hierarchischer Zusammenhang (siehe Abb. 2.2). Die Fahrbahn ist der zusammenhängende geteerte Verkehrsraum (befahrbarer Bereich) der Straße. Sie kann grundsätzlich unterteilt werden in Fahrstreifen, deren Begrenzungen und Seitenstreifen, sowie Randstreifen. Im Gegensatz zur amtlichen Definition der Fahrbahn [133, 134, 135], bei der der Seitenstreifen nicht zur Fahrbahn gehört, der Randstreifen jedoch dazugehört, wird in dieser Arbeit der gesamte geteerte Bereich einer Straße als Fahrbahn bezeichnet, einschließlich Rand- und Seitenstreifen, da dies der potentiell (physikalisch mögliche) befahrbare Bereich ist. Diese Definition trägt dem Umstand Rechnung, dass das vorrangige Ziel der hier vorgestellten Modellierung das Auffinden von Fahrstreifen ist. Jeder Fahrstreifen bildet für sich eine Fläche, die einem Fahrzeug als befahrbarer Bereich zur Verfügung steht und ist daher zwangsläufig frei von statischen Hindernissen. Aus dem Bereich

eines Fahrstreifens lässt sich stets eine Fahrstreifenmitte definieren. Allerdings ist diese nicht unbedingt von großem Nutzen, da eine direkte Verwendung bei der Positionsregelung des Fahrzeugs in vielen Fällen nicht praktikabel ist. So ist beispielsweise ein Ausweichen innerhalb des eigenen Fahrstreifens nur dann möglich, wenn die Grenzen des Fahrstreifens bekannt sind - und nicht nur der Verlauf der Fahrstreifenmitte. Auch Regelungs- und Systemfehler, die zu einer von der Fahrstreifenmitte abweichenden Fahrzeugposition führen, lassen sich hinsichtlich ihrer Kritikalität nur dann sinnvoll bewerten, wenn die Fahrstreifengrenzen bekannt sind.

Die Fahrstreifengrenzen ergeben sich meist direkt aus den baulichen Gegebenheiten. So bilden Fahrstreifenmarkierungslinien und Leitplanken per Definition auf allen autobahnähnlichen Straßen die Fahrstreifengrenzen. Alle deutschen Autobahnen sind - zumindest zum Mittelstreifen hin - mit einer Leitplanke/Betonwand versehen, um zu verhindern, dass ein Fahrzeug in den Gegenverkehr gelangt (siehe bspw. Abb. 2.3a). Somit gehören sie - genauso wie die Fahrstreifenmarkierungslinien - zu den wichtigsten und häufigsten strukturgebenden Elementen der Straßeninfrastruktur.

Im städtischen Bereich sind insbesondere kleinere Straßen durch Bürgersteige begrenzt, aber auch Gebäude, parkende Fahrzeuge oder Pylonen und Baken können einen Fahrstreifen begrenzen. Insbesondere nicht-kontinuierliche Strukturen, wie ein Kollektiv von Baken, sind für die Fahrstreifenschätzung eine Herausforderung (siehe Abb. 2.4).

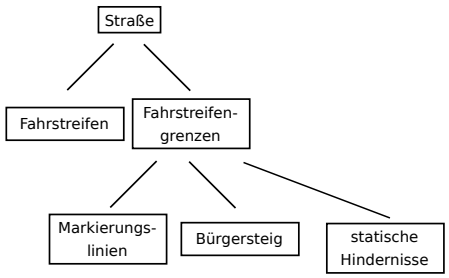


Abbildung 2.2: Topologie der Straße: Es ergibt sich ein hierarchischer Zusammenhang zwischen der Straße, den Fahrstreifen und deren Begrenzungen.

Fahrbahngrenzen sind dadurch definiert, dass sie den befahrbaren Bereich vom nicht-befahrbaren Bereich trennen. Darunter fallen unter anderem die statischen Infrastrukturelemente wie Leitplanken oder Bürgersteige. Im Hinblick auf die Begrenzung von Fahrstreifen ist die Definition nicht so einfach, da die meisten

(a) **(b)**

Abbildung 2.3: (a) Eine typische Autobahn in Deutschland (b) Fahrstreifenbegrenzung durch Randsteine aufgrund von Straßenbahnschienen.

(a) **(b)**

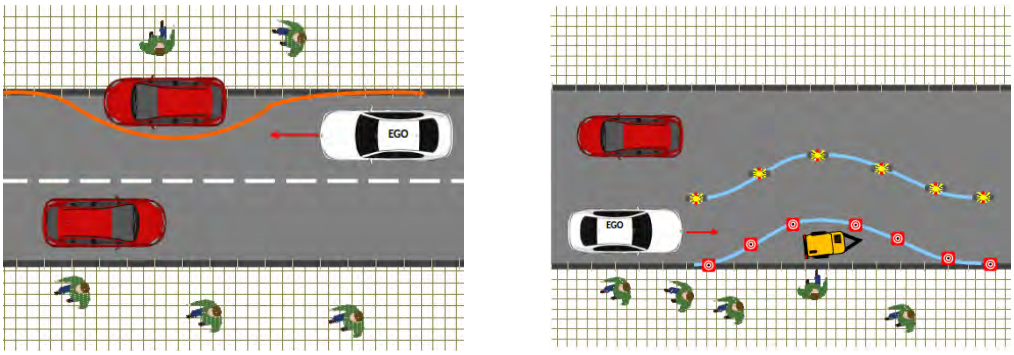
Abbildung 2.4: (a) Situation in einem Wohngebiet: Parkende Autos an beiden Fahrbahnseiten. (b) Baustelle auf der Autobahn mit geänderter Verkehrsführung durch Baken.

Fahrstreifenbegrenzungen befahrbar und überfahrbar sind (bspw. Fahrstreifenmarkierungslinien, Sperrflächen, etc.). Für den weiteren Verlauf spielt die Unterscheidung zwischen Fahrbahn- und Fahrstreifengrenze jedoch nur eine untergeordnete Rolle und der Begriff Fahrstreifengrenze wird für beides verwendet.

Fahrstreifengrenzen lassen sich wie in Abb. 2.2 dargestellt in verschiedene Unterelemente aufteilen. Dies ist insbesondere im Hinblick auf die Planung von Fahrmanövern zum Fahrstreifenwechsel notwendig, da die Art der Fahrstreifenbegrenzung darüber entscheidet, ob ein Wechsel physikalisch möglich und auch rechtlich erlaubt ist. Auch für andere höhere Funktionen, wie bspw. eine Fußgänger-Prädiktion ist es durchaus sinnvoll, den Bürgersteig als Unterelemente von Fahrstreifengrenzen separat zu modellieren, um zu einem späteren Zeitpunkt u.U. den Bürgersteig zu extrahieren und für eine Fußgänger-Prädiktion nutzen zu können.

Ein weiterer Aspekt, der bei der Repräsentation der Fahrbahn- und Fahrstreifengrenzen eine wichtige Rolle spielt, ist die Unterscheidung zwischen physikalischer und *semantischer* Begrenzung.

Die gebauten Fahrbahn- und Fahrstreifenbegrenzungen bilden nicht immer die für die Fahraufgabe inhaltlich relevanten Begrenzungen. Die relevanten Begrenzungen werden im Weiteren auch als *semantische* Begrenzungen bezeichnet, da sie nicht die physikalischen Gegebenheiten widerspiegeln, sondern die inhaltliche Bedeutung der Elemente hinsichtlich der Begrenzungen. Abbildung 2.5 zeigt zwei Beispiele von *semantischen* Begrenzungen.



(a) (b)

Abbildung 2.5: Relevanz der *semantischen* Begrenzung. (a) Situation mit einem parkenden Auto und einem Bürgersteig. Der Bürgersteig stellt die bauliche Fahrstreifengrenze dar, das parkende Auto die semantische. (b) Baustelle mit durch Pylonen vorgegebenem, geänderten Fahrstreifenverlauf. Die blauen Linien geben die semantische Fahrstreifengrenze an.

Die bloße Erkennung und Klassifizierung der Elemente ist somit nicht ausreichend, um die Fahrstreifen zu extrahieren, da die Semantik der Elemente hinsichtlich des möglichen Fahrstreifens von entscheidender Bedeutung ist.

Aufgrund der Hierarchie des Modells ist es nicht möglich, als mathematische Grundlage die „Bayes"-Formulierung (bekannt von Belegungskarten) zu verwenden, da diese keine hierarchischen Hypothesen abbilden kann. Daher bietet es sich an, hierfür auf die Dempster-Shafer-Theorie zurückzugreifen, die bereits in Kap. 2.1.2 erläutert worden ist. Darüber hinaus kann mit der Dempster-Shafer-Theorie sehr elegant Teilwissen im Modell abgebildet werden. Dies ist für die Verarbeitung und Fusion von Sensordaten von entscheidendem Vorteil.

Für die vorliegende Arbeit wird nun ein neuartiges 'Frame of Discernment' definiert:

$$\Omega := \{L, M, S, O\} \tag{2.19}$$

mit

$\{L\}$: Hypothese für Fahrstreifen (engl.: Lane)

$\{M\}$: Hypothese für Fahrstreifenmarkierungslinie (engl.: Marking)
$\{S\}$: Hypothese für Bürgersteig (engl.: Sidewalk)
$\{O\}$: Hypothese für eine Belegung durch ein statisches Hindernis (engl.: Obstacle) (Leitplanke, parkendes Auto, Verkehrsleitkegel/Baken, etc.)

Ferner wird ein Fahrstreifen als eine befahrbare Fläche zwischen einer linken und einer rechten Fahrstreifenbegrenzung definiert. Diese müssen nicht zwangsläufig mit den Fahrbahnbegrenzungen übereinstimmen. Fahrstreifenmarkierungslinien sind alle weißen (und gelben) Markierungslinien, die Fahrstreifenbegrenzungen darstellen. Die Hypothese $\{S\}$ beinhaltet sowohl die Begrenzung (Randstein) zwischen dem Fahrstreifen und dem Bürgersteig, als auch die Fläche des Bürgersteigs. $\{O\}$ repräsentiert alle anderweitigen statischen Objekte, die nicht ohne Kollision überfahren werden können.

Von der Potenzmenge

$$\begin{aligned} 2^{\Omega} = \{&L, M, S, O, \{L, M\}, \{L, S\}, \{L, O\}, \{M, S\}, \\ &\{M, O\}, \{S, O\}, \{L, M, S\}, \{L, S, O\}, \{L, M, O\}, \\ &\{M, S, O\}, \varnothing, \Omega\} \end{aligned} \tag{2.20}$$

wird nur eine Teilmenge 2_r^{Ω} benötigt, da die 'belief masses' für die übrigen Hypothesen mit dem für diese Arbeit verwendeten Sensor-Setup nicht direkt messbar ist.

$$2_r^{\Omega} = \{L, M, S, O, \{S, O\}, \{M, L\}, \{M, S, O\}, \{L, O\}, \varnothing, \Omega\} \tag{2.21}$$

Falls zukünftig mit einem geänderten Sensor-Setup weitere 'belief masses' direkt messbar sind, kann die Menge 2_r^{Ω} jederzeit um weitere Elemente aus 2^{Ω} erweitert werden. Eine konzeptionelle Änderung des Ansatzes ist dazu nicht erforderlich.

Grundsätzlich lässt sich die Gruppe der statischen Objekte auch in weitere Elemente unterteilen wie bspw. parkende Fahrzeuge oder Gebäude. Durch den hierarchischen Aufbau der hier gewählten Hypothesen lässt sich dies sehr einfach im Modell abbilden, indem O ersetzt wird durch die zwei neuen Hypothesen „parkende Fahrzeuge" und „Gebäude". Die Wahl des 'frame of discernment' muss sich dabei stets an der gewünschten Anwendung als auch an der Verfügbarkeit der Daten, aus denen die 'belief masses' für die Hypothesen generiert werden sollen, orientieren. Darüber hinaus muss sichergestellt sein, dass mindestens ein 'Sensor' die gewünschte Information bereitstellen kann. Andernfalls ist eine weitere Aufteilung einer Hypothese nicht sinnvoll.

Mit dem in Gl. 2.21 definierten 'frame of discernment' lassen sich nun Sensordaten von verschiedenen Sensoren fusionieren und über die Zeit akkumulieren. Dazu wird eine gridbasierte Struktur (Grid) verwendet, die nachfolgend als DSTMap bezeichnet wird. Anschließend kann daraus pro Zelle der 'belief' für beliebige Hypothesengruppen berechnet werden. Dieser wird später in einem in Kap. 3 vorgestellten Verfahren zur Extraktion von potentiellen Fahrstreifen genutzt.

Das Grid ist ein kartesisches planares Gitter mit quadratischen Zellen, in denen eine stochastische Information m gespeichert ist, die aus Sensordaten abgeleitet worden ist [96]. Analog zu dem Konzept der Belegungskarte aus Abschnitt 2.1 wird ein Grid als Menge voneinander unabhängiger Zellen definiert. Dies erlaubt eine effiziente und parallele Berechnung der Zellinhalte.

Analog zur Belegungskarte muss nun die Wahrscheinlichkeitsverteilung bzw. die belief-Funktion über alle kombinatorisch möglichen Grids berechnet werden, um dasjenige Grid zu finden, das - bei gegebenen Sensordaten - am ehesten mit der Realität übereinstimmt.

Mit der gegebenen Hypothesen- oder Zustandsmenge 2_r^Ω, und der Festlegung, dass sich jede Gridzelle in genau einem der Zustände aus 2_r^Ω befinden muss, ergeben sich $n = \left\|2_r^\Omega\right\|^N$ kombinatorische Möglichkeiten für ein Grid mit N Zellen bzw. $n = \left\|2_r^\Omega\right\|^N$ verschiedene Grids.

Sei nun $\mathcal{M}$ die Menge aller dieser Grids:

$$\mathcal{M} = \{\vec{m}_1, \ldots, \vec{m}_n\} \tag{2.22}$$

so lässt sich damit eine 'belief function' definieren, die für jedes dieser Grids $\vec{m}_i$ angibt, wie wahrscheinlich es ist, dass genau dieses Grid die Realität widerspiegelt. Da die Hypothesen aus 2_r^Ω keine kontinuierliche, sondern diskrete Zustände sind, lässt sich diese 'belief function' auch schreiben als

$$Bel(\mathcal{M}) = \left[Bel(\vec{m}_1), \ldots, Bel(\vec{m}_n)\right]^T \tag{2.23}$$

Ferner wird ein 'belief' pro Zelle definiert, der die 'belief masses' aller Hypothesen aus 2_r^Ω für diese Zelle beinhaltet. Also pro Zelle für jede Hypothese aus 2_r^Ω eine Zahl, die angibt, wie wahrscheinlich es ist, dass der Zustand der Zelle genau dieser Hypothese entspricht:

$$Bel(m^i) = \left[Bel(m^i = \omega_1), \ldots, Bel(m^i = \omega_n)\right]^T \tag{2.24}$$

für alle $\omega_n \in 2_r^\Omega$.

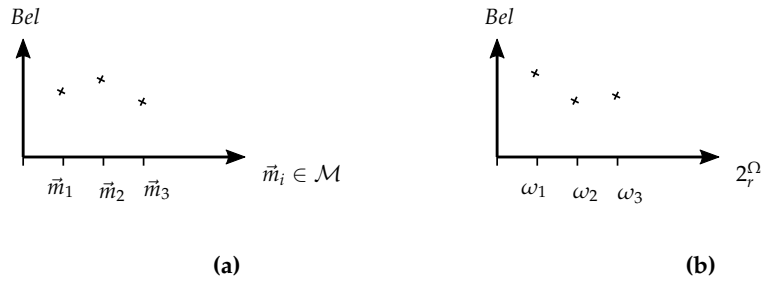


Abbildung 2.6: (a) Eine exemplarische Belief-Verteilung $Bel(\mathcal{M})$ aller kombinatorisch möglichen Grids $\vec{m}_i$. (b) Eine exemplarische Belief-Verteilung $Bel(m^i)$ für eine einzelne Grid-Zelle mit einem Belief pro Hypothese.

Unter der Annahme unabhängiger Zellen ergibt sich außerdem

$$Bel(\vec{m}_j) = \prod_i Bel(m^i = \vec{m}_j[i]) \tag{2.25}$$

wobei m^i der belief der Zelle i ist.

Um nun dasjenige Grid zu finden, das am ehesten der Realität entspricht, muss das Maximum der Verteilung $Bel(\mathcal{M})$ gefunden werden. Genauer formuliert, muss dasjenige $\vec{m}$ identifiziert werden, für das der belief maximal ist:

$$\arg\max_{\vec{m} \in \mathcal{M}} \left(Bel(\vec{m}_i) \right) \tag{2.26}$$

Das Problem dabei ist, dass die Verteilung nicht bekannt ist. Eine Berechnung der Verteilung würde die Berechnung aller kombinatorisch möglichen Grids und anschließend deren 'beliefs' bedeuten, um dasjenige mit dem maximalen 'belief' zu identifizieren. Dies ist - gegeben die extreme Anzahl der kombinatorisch möglichen Grids - praktisch nicht durchführbar.

Eine alternative Möglichkeit ist, ein fest vorgegebenes $\vec{m}$ iterativ zu verändern, sodass sich in jedem Iterationsschritt $Bel(\vec{m})$ erhöht. Dieser Vorgang wird auch als iterative/rekursive Zustandsschätzung bezeichnet. Mit der Annahme der Zellunabhängigkeit kann man das Problem weiter reduzieren, iterativ für jede Zelle den Zustand mit dem höchsten 'belief' zu finden. Dies ist sehr effizient durchführbar und in vertretbar kurzer Rechenzeit berechenbar.

Zur besseren Lesbarkeit wird nun $\mathbf{m}$ als ein Grid mit $n \times n$ Zellen definiert, dass für jede Zelle 'belief masses' für alle Elemente aus 2_r^Ω beinhaltet. Aus den 'belief masses' lassen sich dann jederzeit für alle Hypothesen die entsprechenden 'beliefs' *Bel* berechnen.

$$\mathbf{m} = \left[m(m^1), m(m^2), \ldots, m(m^{n \cdot n})\right]^T \tag{2.27}$$

mit dem Ziel, für jede Zelle unabhängig voneinander mit Hilfe von Sensordaten die 'belief masses' $m(m^i)$ für alle Hypothesen zu schätzen. Dabei bedeutet eine 'belief mass' von 0 für eine spezifische Hypothese A, dass es keinerlei Hinweise gibt, dass der Zustand der Zelle A ist. Eine 'belief mass' von 1.0 hingegen gibt an, dass der Zustand der Zelle mit absoluter Sicherheit genau A ist.

Die rekursive Zustands- oder Belief-Schätzung pro Zelle kann nun formuliert werden als

$$m(m^i) = m(m^i \mid \mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = m(m^i \mid \mathbf{z}_t, \mathbf{x}_t, \mathbf{m}_{1:t-1}) \tag{2.28}$$

mit den Sensor-Messdaten $\mathbf{z}$ von Zeitpunkt 1 bis zum aktuellen Zeitschritt t, und dem Ego-Fahrzeugzustand $\mathbf{x}$ (Position & Ausrichtung). Die Eigenbewegung des Fahrzeugs wird durch kontinuierliches Verschieben des Grids um $\Delta\mathbf{x}(i+1, i) = \mathbf{x}_{t_{i+1}} - \mathbf{x}_{t_i}$ kompensiert. Dabei wird vorausgesetzt, dass in der Schätzung $\mathbf{m}_{t-1}$ bereits alle Messdaten $\mathbf{z}$ und Fahrzeugzustände $\mathbf{x}$ bis zum Zeitpunkt $t-1$ enthalten sind (Markov „complete state assumption"). Somit hängt die aktuelle Schätzung (Zeitpunkt t) nur noch vom vorherigen Zustand sowie den Messdaten aus Zeitschritt t ab.

Für die rekursive Schätzung der 'belief masses' pro Zelle werden nun aus verschiedenen Sensordaten sog. ScanGrids erzeugt, die nachfolgend beschrieben werden. Anschließend werden die verschiedenen ScanGrids fusioniert und in der DSTMap über die Zeit akkumuliert.

2.2.1 Generierung der ScanGrids aus Sensordaten

Für den weiteren Verlauf wird nun $\mathbf{m}_t^S(\cdot)$ als ein Grid definiert, das die Messdaten $\mathbf{z}_t$ eines Sensors S zu einem speziellen Zeitpunkt t repräsentiert und pro Zelle die 'belief masses' m für eine Menge an Hypothesen beinhaltet. In Analogie zu der Bezeichnung LiDAR- oder Radar-Scan, der eine Momentaufnahme von Messdaten beschreibt, werden die Grids $\mathbf{m}_t^S(\cdot)$ im weiteren Verlauf auch als ScanGrids bezeichnet.

Für die DSTMap werden Sensordaten von erkannten Fahrstreifenmarkierungslinien, Informationen über dynamische Objekte (andere Verkehrsteilnehmer),

Belegungskarten, und eine Punktwolke von einem Stereo-Vision-System in Kombination mit einer bild-basierten semantischen Segmentierung verwendet, um 'belief masses' für die Hypothesen in 2_r^{Ω} abzuleiten. Für jeden Typ von Eingangsdaten wird ein ScanGrid erzeugt, das die Daten von einem einzigen Zeitschritt und von dieser einen Eingangsquelle widerspiegelt.

Nachfolgend wird die Erzeugung der ScanGrids für die verschiedenen Eingangsdaten erläutert.

Fahrstreifenmarkierungslinien

Der Verlauf von Fahrstreifen wird oftmals durch Markierungslinien vorgegeben, die auf der Fahrbahn aufgezeichnet sind. Sie teilen freie, befahrbare Areale je nach Linientyp in verschiedene Bereiche auf und liefern somit direkt semantische Informationen über die Umgebung.

Generell existieren in Deutschland *Fahrstreifenbegrenzungslinien*, um die einzelnen Fahrstreifen voneinander zu trennen, und *Fahrbahnbegrenzungslinien*, um den äußersten rechten Fahrstreifen vom Seitenstreifen oder dem Grünstreifen abzugrenzen. In Bezug auf die Wahrscheinlichkeit einer Befahrbarkeit können folgende Annahmen getroffen werden:

- *Fahrstreifenbegrenzungslinien* separieren die Fahrstreifen. Sowohl auf der rechten als auch auf der linken Seite der Linie befindet sich ein Fahrstreifen, der befahrbar ist. In Deutschland sind *Fahrstreifenbegrenzungslinien* gestrichelte Linien oder doppelt-durchgezogene Linien.
- *Fahrbahnbegrenzungslinien* werden verwendet, um die Fahrbahn zu begrenzen. Mögliche befahrbare Fahrstreifen existieren auf der inneren Seite (zur Fahrbahn hin). *Fahrbahnbegrenzungslinien* sind einfach durchgezogene Linien.
- Unabhängig vom Linientyp kann an der Position der Linie kein Fahrstreifen existieren. Dies ist offensichtlich, da die Linien die Fahrstreifen per Definition begrenzen.

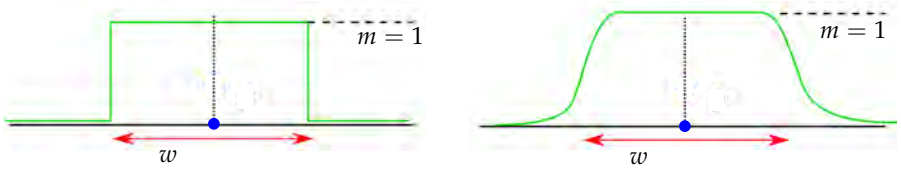
Heutige kamera-basierte Fahrstreifenerkennungssysteme sind in der Lage, sowohl den geometrischen Verlauf von Markierungslinien zu erkennen, als auch detaillierte Informationen über die Linie selbst. Beispiele hierfür sind der Linientyp (durchgezogene oder gestrichelte Linie), die Farbe oder die Linienbreite. Insbesondere in Baustellen ist die Farbe der Linie entscheidend für ihre semantische Bedeutung, da bei Existenz von weißen und gelben Linien die gelben Vorrang haben. Im vorliegenden Fall wird ein Kamerasystem verwendet, dass kontinuierlich eine Liste von erkannten Fahrstreifenmarkierungen $\mathcal{M}$ liefert:

$$\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_n\} \tag{2.29}$$

$$\mathcal{M}_i = \left[\{\mathbf{x}_1, \dots, \mathbf{x}_n\}, k, w, c, \sigma, p\right]^T \tag{2.30}$$

wobei $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ eine Menge von Punkten (x, y) ist und die Geometrie der Linie in 2D beschreibt. k ist der Linientyp: Durchgezogene oder gestrichelte Linie (einfach oder doppelt), Bürgersteig/Randstein, Straßenkante, sowie 'unbekannt'. w steht für die Linienbreite, c für die Farbe, σ für die Messunsicherheit, und p für die Existenzwahrscheinlichkeit.

Um nun 'belief masses' für die Hypothesen in 2_r^Ω abzuleiten, wird für jede Zelle eine BBA verwendet, die als Faltung von einer Gauß-Verteilung mit einer Rechteck-Funktion definiert ist (siehe Abb. 2.7). Die BBA fungiert hierbei als das Dempster-Shafer-Äquivalent zu einem inversen Sensormodell $p(m|\mathbf{z}_t)$ vom Bayes'schen Filter für Belegungskarten (bspw. [136]). Die BBA ist normalisiert zu $[0; 1]$, um Gl. (2.9) zu erfüllen.



(a) **(b)**

Abbildung 2.7: (a) Rechteck-Funktion für die Fläche, die die Linie aufgrund ihrer Breite einnimmt. (b) Faltung der Rechteck-Funktion mit einer Gauß-Verteilung.

In Abhängigkeit von der Hypothese und dem Linientyp k kann die BBA-Funktion in verschiedener Weise verwendet werden, um 'belief masses' abzuleiten und in die Gridzellen einzutragen. Bei durchgezogenen oder gestrichelten Linien kann die 'belief mass' für die Hypothese *Marking*, M, abgeleitet werden. Dazu wird die Gaußverteilung mit einer Rechteck-Funktion mit Breite w (von $\mathcal{M}_i$) gefaltet und anschließend auf der erkannten Linie auf jedem Punkt $\mathbf{x}_i$ orthogonal zur Richtung der Linie positioniert. Die Standardabweichung σ der Gauß-Verteilung wird direkt von $\mathcal{M}_i$ übernommen. Die Rechteck-Funktion repräsentiert dabei die Fläche bzw. Breite der Markierungslinie. Das gleiche Vorgehen wird angewandt, wenn der Linientyp „Bürgersteig" oder „Straßenkante" ist. Die Masse wird dann entsprechend S bzw. O zugeordnet. Wenn der Linientyp unbekannt ist, wird die Masse $\{M, S, O\}$ zugeordnet.

Eine weitere Schlussfolgerung ergibt sich aus der Tatsache, dass das 'frame of discernment' per Definition vollständig ist, d.h. es muss so definiert sein, dass

genau eine der Hypothesen stets zutrifft. Somit kann für alle Regionen, in denen keine Markierungslinien und kein Bürgersteig erkannt worden ist, die 'belief mass' $\neg(M \vee S) = \{L, O\}$ abgeleitet werden, sofern die Region im Sichtfeld der Kamera liegt.

Die zusätzliche Berücksichtigung der *semantischen* Bedeutung des Linientyps, lässt weitere Schlussfolgerungen zu. Da die Fahrstreifenmarkierungslinien naturgemäß den befahrbaren Bereich in einzelne Fahrstreifen unterteilen, beinhalten sie implizit die Existenz von Fahrstreifen. Darüber hinaus markiert eine gestrichelte Linie zwar die Grenze eines Fahrstreifens, die semantische Bedeutung jedoch ist, dass es erlaubt ist, die Linie zum Zweck eines Fahrstreifenwechsels zu überqueren. Daher muss auf beiden Seiten der Linie ein Fahrstreifen existieren. Im Gegensatz dazu kann bei durchgezogenen Linien die Existenz eines Fahrstreifens nur für die dem Ego-Fahrzeug zugewandten Seite angenommen werden. Um nun die 'belief mass' für Fahrstreifen, $m(L)$, zu berechnen, wird für jede Markierungslinie $\mathcal{M}_i$ die Mitte der gefalteten Gauß-Verteilung um die Hälfte einer typischen Fahrstreifenbreite nach links/rechts (in Abhängigkeit vom Linientyp) verschoben und die 'belief mass' in die darunterliegenden Zellen geschrieben (siehe Abb. 2.8). Die Breite der Rechteck-Funktion repräsentiert eine typische Fahrstreifenbreite.

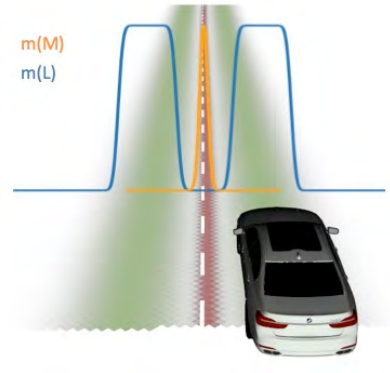


Abbildung 2.8: Situation mit einer erkannten gestrichelten Fahrstreifenmarkierungslinie. Oben: BBAs für *Marking*, $m(M)$ (orange) und *Lane*, $m(L)$ (blau). Unten: Das berechnete ScanGrid. Die Farben geben die Hypothesen an. $\{M\}$: rot; $\{L\}$: grün. Die Transparenz (Alpha-Wert) stellt die 'belief mass' dar.

In Abhängigkeit von der Existenzwahrscheinlichkeit p von $\mathcal{M}_i$, wird der eher konservative Ansatz (Zuordnung der 'belief mass' zu $\{L, O\}$) oder der optimistischere (Zuordnung zu L) gewählt. Damit wird die Entscheidung zwischen konservativer und optimistischer Schätzung automatisch direkt in Abhängigkeit von der Zuverlässigkeit der Messung gemacht. Dies führt bei zuverlässig erkannten Markierungen zu einer besseren Zustandsschätzung für $m(L)$ ohne höheres Risiko im Falle unzuverlässig detektierter Markierungslinien.

Um die Rechenzeit zur Erstellung des ScanGrids zu reduzieren, wird das Verfahren parallelisiert, damit es auf entsprechenden Graphikprozessoren (GPU) ausgeführt werden kann. Im Gegensatz zum naheliegenden Ansatz, die Berechnung für alle Zellen zu parallelisieren, wird hier auf der Ebene von $\mathcal{M}$ parallelisiert und somit jedes Element aus $\mathcal{M}$ parallel abgearbeitet. Dies führt zu einem enormen Geschwindigkeitsvorteil, da nur für diejenigen Zellen Berechnungen stattfinden, die in der näheren Umgebung der Elemente von $\mathcal{M}$ liegen.

Im weiteren Verlauf wird das so erzeugte ScanGrid als $\mathbf{m}_t^{\mathrm{LM}}(\cdot)$ bezeichnet, wobei der obere Index die Quelle der Sensordaten (Lane Markings) angibt.

Dynamische Objekte

Die Pfade von anderen Verkehrsteilnehmern sind eine andere wichtige Informationsquelle für die Umfeldmodellierung. Unter der Annahme, dass Fahrzeuge meist auf gültigen Fahrstreifen fahren, lassen sich mit geeigneten BBAs 'belief masses' für Fahrstreifen und Fahrstreifengrenzen ableiten.

Im vorliegenden Fall werden dynamische Objekte verwendet, die aus einer dynamischen Belegungskarte extrahiert werden und deren Zustand über die Zeit geschätzt wird. Der Ansatz ist in [137] beschrieben und liefert eine Liste von dynamischen Objekten $\mathcal{O}$:

$$\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_n\} \tag{2.31}$$

$$\mathcal{O}_i = \left[x, y, \phi, l, w, k\right]^T \tag{2.32}$$

wobei jedes Objekt als rechteckige Box mit Länge l und Breite w modelliert ist. (x, y) ist die 2D-Position der Box und ϕ deren Orientierung. k ist der Typ (PKW, LKW, Motorrad, Fussgänger, etc.).

Ähnlich wie im Fall der Fahrstreifenmarkierungen wird eine Faltung einer Gauß-Verteilung mit einer Rechteck-Funktion verwendet, um die 'belief masses' für die einzelnen Gridzellen abzuleiten. Die Breite w der Objekt-Box entspricht der Breite der Rechteck-Funktion. Für die Hypothese L (Lane) wird die Gauß-Funktion für jedes Objekt $\mathcal{O}_i$ mittig auf der Objektposition (x, y), orthogonal zur Orientierung ϕ platziert und die resultierenden 'belief masses' in die darunterliegenden Zellen geschrieben.

Setzt man voraus, dass Fahrzeuge in der Regel mittig in ihrem Fahrstreifen fahren, lassen sich auch 'belief masses' für die linke und rechte Fahrstreifengrenze ableiten. Links und rechts neben jedem dynamischen Objekt muss sich damit

in einem entsprechendem Abstand eine Fahrstreifengrenze befinden. Im Unterschied zum ScanGrid aus Fahrstreifenmarkierung ist bei den dynamischen Objekten allerdings nicht bekannt, um welche Art von Fahrstreifenbegrenzung es sich handelt. Daher wird die 'belief mass' nicht der Hypothese M (Marking), sondern $\{M, S, O\}$ zugeordnet.

Das aus den Objekten resultierende ScanGrid wird als $\mathbf{m}_t^{\text{Obj}}(\cdot)$ bezeichnet.

Semantische Segmentierung von Kamerabildern

Mit der Verfügbarkeit von pixel-genau beschrifteten (gelabelten) Kamerabildern wie beispielsweise KITTI [138, 139, 140, 141] oder CityScapes [142, 143] haben die Ansätze zur semantischen Segmentierung von Bildern in den letzten Jahren einen enormen Auftrieb erhalten. Ziel der semantischen Segmentierung ist die Klassifikation von Pixeln eines Kamerabildes, um somit Bildbereiche voneinander trennen zu können (segmentieren). Dabei wird jedem Pixel eine Klasse oder Label zugewiesen, wie bspw. Straße, Bürgersteig, Schienen, Gebäude, Zaun, Wand, Pfosten, Boden oder Vegetation.

Semantische Segmentierungen werden heute häufig für die Umfeldwahrnehmung in der Robotik oder auch im Bereich des autonomen Fahrens eingesetzt [144, 145, 146, 147, 148, 149, 150]. Für den hier vorgestellten Ansatz wird eine Re-Implementierung des PSPNet [147] verwendet. Abb. 2.9a zeigt die Ergebnisse der semantischen Segmentierung für eine städtische Verkehrssituation.

Damit die semantische Information aus dem Kamerabild nutzbringend verwendet werden kann, ist zusätzlich eine räumliche Information (Position des Pixels in der realen Welt) erforderlich. Es wird daher hier ein Stereo-Vision-System mit einer semantischen Segmentierung kombiniert. Die räumliche Information wird dabei mit Hilfe der Disparität (z.B. via Semi-Global Matching [151]) berechnet. Im Ergebnis liefert das System kontinuierlich eine Punktwolke $\mathcal{P}$, die für jeden Punkt $\mathcal{P}_i$ sowohl die semantische, als auch die räumliche Information beinhaltet:

$$\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\} \tag{2.33}$$

$$\mathcal{P}_i = \left[x, y, z, k, \sigma_k, \sigma_z\right]^T \tag{2.34}$$

wobei (x, y, z) die 3D-Position des Punktes, k die Klasse, σ_k die Konfidenz der Klassifizierung, und σ_z die geschätzte Standardabweichung des Fehlers der Tiefeninformation (Entfernung des Punktes zur Kamera) ist. Die Berechnung der Standardabweichung geschieht mit Hilfe des Stereo-Vision-Fehlermodells aus [152]. Abb. 2.9b zeigt eine solche semantische Punktwolke zusammen mit einer hochgenauen Offline Karte („ground-truth"-Daten).

(a) (b)

Abbildung 2.9: (a) Ein Kamerabild mit einer Überlagerung der semantischen Segmentierung in der rechten Bildhälfte (Re-Implementierung des PSPNet). Die Klasse ist durch die Farbe des Pixels gegeben. (b) Überlagerung einer hochgenauen Offline Karte mit der semantischen Punktwolke. Die Farbkodierung der hochgenauen Karte entspricht der der semantischen Segmentierung.

Für das ScanGrid wird nun die Punktwolke auf die Bodenebene projiziert und es werden 'belief masses' abgeleitet. Dabei werden die Klassen der semantischen Segmentierung den Hypothesen aus 2_r^Ω wie folgt zugeordnet:

$$\begin{aligned} \text{Straße} &\rightarrow \{M, L\} \\ \text{Bürgersteig} &\rightarrow \{S\} \\ \text{Wand, Zaun, Gebäude, Leitplanke, Vegetation, ...} &\rightarrow \{O\} \end{aligned}$$

Da die Klasse *Straße* der semantischen Segmentierung den gesamten geteerten Bereich inkl. Markierungslinien umfasst, kann bei der Zuordnung zu den Hypothesen nicht zwischen L (Lane) und M (Marking) unterschieden werden. Klassen, wie bspw. 'Wand' oder 'Zaun' werden zusammengefasst und als nicht näher klassifiziertes statisches Objekt behandelt, da sie für den vorliegenden Anwendungsfall eine identische semantische Bedeutung haben (nicht ohne Kollision überfahrbar). Auch Grünflächen (Vegetation) werden als statische Objekte behandelt, da eine Überfahrung unerwünscht ist. In Abhängigkeit der Klasse jedes Punktes, tragen die Punkte zur Erhöhung der 'belief mass' für die Hypothesen $\{M, L\}$, $\{S\}$ und $\{O\}$ der Gridzellen bei. Dabei ist die Art der Berechnung stets für alle Hypothesen gleich. Die Berechnung wird nun im Folgenden erläutert.

Jeder Punkt $\mathcal{P}_i$ der Punktwolke entspricht einem Pixel P im segmentierten Kamerabild. Sind die Position des Pixels im Bild und die intrinsischen/extrinsischen Werte der Kamera bekannt, so kann der Pixel auf eine Fläche auf der Bodenebene projiziert werden. Diese Pixelprojektionsfläche wird im Folgenden als A_{px} bezeichnet. Eine vereinfachte Darstellung dieser Projektion ist in Abb. 2.10 dargestellt.

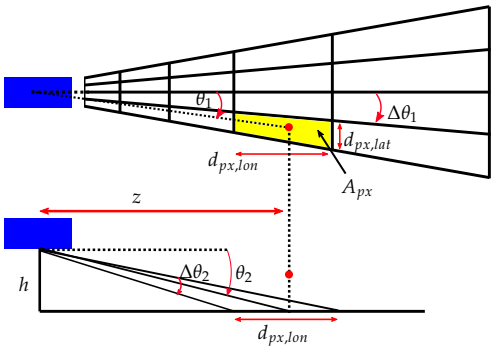


Abbildung 2.10: Projektion eines Bildpixels P (roter Punkt) auf die Bodenebene. Dargestellt ist die Kamera (blaues Rechteck) und die Pixelstrahlen (schwarze Linien). Oben: Draufsicht, unten: Seitenansicht.

mit

- θ_1: lateraler Projektionswinkel; direkte Berechnung aus den Pixelkoordinaten (im Bild) und der Brennweite des Objektivs. Da durch die Verwendung einer Stereo-Kamera die Pixelposition in 3D bereits bekannt ist, kann θ_1 mit $atan(z/x)$ direkt berechnet werden.
- θ_2: Vertikaler Projektionswinkel.
- z: Horizontaler Abstand von P zur Kamera.
- h: Höhe der Kamera über Bodenebene.
- $\Delta\theta_{1,2}$: Winkelabstand in horizontaler und vertikaler Richtung zwischen 2 Pixelstrahlen.

Die Pixelprojektionsfläche auf der Bodenebene kann nun approximiert werden durch:

$$A_{px} \approx d_{px,lon} \cdot d_{px,lat} \tag{2.35}$$

$$d_{px,lon} = \frac{z^2}{h \cdot f_y} \tag{2.36}$$

$$d_{px,lat} = \frac{z}{f_x} \tag{2.37}$$

wobei f_x, f_y die Brennweiten in horizontaler und vertikaler Richtung sind.

Berechnung der Positionswahrscheinlichkeit:
Die Positionswahrscheinlichkeit eines Pixels P in der Bodenebene ergibt sich aus der Unsicherheit der Projektion und ist hauptsächlich auf die Unsicherheit

σ_z der Entfernungsmessung (Disparitätsberechnung) zurückzuführen. Die Positionswahrscheinlichkeit wird hier als bivariate Gauß-Verteilung um den projizierten Punkt auf der Bodenebene modelliert. Die Unsicherheiten in der Entfernung und im Winkel definieren die Ausdehnung der Verteilung in Richtung des Pixelstrahls bzw. orthogonal dazu (s. Abb. 2.11).

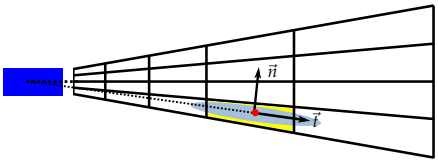


Abbildung 2.11: Positionswahrscheinlichkeit modelliert als bivariate Gauß-Verteilung (blaue Ellipse) mit der Hauptachse ($\vec{t}$) entlang des Pixelstrahls für einen projizierten Pixel P (roter Punkt). $\vec{n}$ zeigt entlang der Nebenachse. Die elliptische Form ergibt sich dadurch, dass die Entfernungsunsicherheit bei Stereo-Kameras stets wesentlich größer ist als die Winkelunsicherheit.

Die Positionswahrscheinlichkeit eines projizierten Pixels p auf der Bodenebene kann somit - für alle Punkte der Bodenebene - berechnet werden mit

$$p_P(u,v) = \frac{1}{2\pi \cdot \sigma_u(P)\sigma_v(P)} \cdot exp\left(-\frac{1}{2}\left(\frac{u^2}{\sigma_u^2(P)} + \frac{v^2}{\sigma_v^2(P)} \right) \right) \tag{2.38}$$

wobei u, v jeweils die Entfernung des Punktes (der Bodenebene) zum projizierten Punkt P entlang der Achsen ist: u entlang $\vec{t}$ und v entlang $\vec{n}$. $\sigma_u(P)$ und $\sigma_v(P)$ sind die entsprechenden Standardabweichungen, die die Unsicherheit in Entfernung und Winkel modellieren.

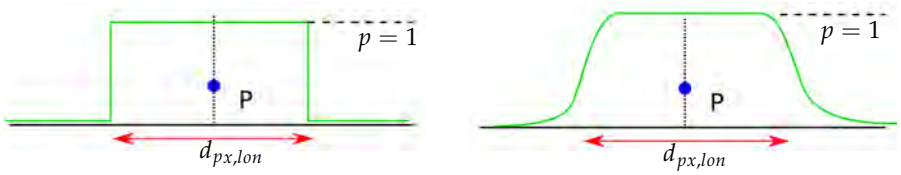
Jedoch gilt die Wahrscheinlichkeits-Dichtefunktion aus Gl. (2.38) nur dann, wenn der projizierte Punkt einem einzelnen **Punkt** der Bodenebene entspricht. Ein projizierter Bildpixel P entspricht jedoch einer Pixelprojektionsfläche A_{px} und nicht einem einzelnen Punkt der Bodenebene. Gäbe es keine Positionsunsicherheiten, wäre die Wahrscheinlichkeits-Dichtefunktion für die Projektion eines Pixels eine uniforme Verteilung mit dem konstanten Wert 1 für die Pixelprojektionsfläche (Abb. 2.12a). Unter Berücksichtigung der Unsicherheiten ist die Wahrscheinlichkeits-Dichtefunktion somit eine Faltung von beiden Verteilungen (Abb. 2.12b).

Die Faltung wird hier approximiert mit

$$p_{pixel,P}(u,v) = \frac{1}{2\pi \cdot \sigma_u\sigma_v} \cdot exp\left(-\frac{1}{2}\left(\frac{u^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2} \right) \right) \tag{2.39}$$

$$\sigma_u = \sigma_u(P) + d_{px,lon} \tag{2.40}$$

$$\sigma_v = \sigma_v(P) + d_{px,lat} \tag{2.41}$$



(a) **(b)**

Abbildung 2.12: Wahrscheinlichkeits-Dichteverteilung der Pixelprojektion als uniforme Verteilung für den Fall ohne Unsicherheiten (a) und in (b) für den Fall mit Unsicherheiten. Letztere Verteilung ist eine Faltung der Verteilung aus (a) mit einer Gauß-Verteilung.

Berechnung der Konfidenz der Klassifizierung pro Gridzelle:
Jeder Punkt $\mathcal{P}_i$ der Punktwolke besitzt eine Konfidenz σ_k der Klassifizierung. Die Klassifizierungskonfidenz einer Gridzelle ist somit eine Fusion von Klassenkonfidenzen von all denjenigen Pixeln, die bei der Projektion in eine Zelle bzw. deren unmittelbare Umgebung fallen. Daher wird die Klassenkonfidenz einer Gridzelle als gewichteter Durchschnitt berechnet, wobei die Positionswahrscheinlichkeit als Gewicht verwendet wird:

$$p_{class}(q)_i = \frac{\sum_{P \in P^q} \sigma_k \cdot p_{pixel,P}(i)}{\sum_{P \in P^q} p_{pixel,P}(i)} \tag{2.42}$$

mit q als Hypothesengruppe, i der Index der Gridzelle, σ_k die Konfidenz von Punkt $\mathcal{P}_i$ und P^q als Menge aller unmittelbar in der Nähe befindlichen projizierten Pixel, für die der dazugehörige Punkt $\mathcal{P}_i$ die Klassifizierung q besitzt.

Berechnung der Positionswahrscheinlichkeit pro Gridzelle:
Die Fusion von Positionswahrscheinlichkeiten von Punkten, die in dieselbe Gridzelle fallen, ist ein Produkt von inversen Wahrscheinlichkeiten (es ist die inverse Wahrscheinlichkeit, dass die Gridzelle **nicht** zur Pixelprojektionsfläche gehört)

$$p_{pixel}(q)_i = 1 - \Pi_{P \in P^q}(1 - p_{pixel,P}(i)) \tag{2.43}$$

Berechnung der 'belief mass' pro Gridzelle:
Die 'belief mass' einer Hypothese q für eine Gridzelle i ergibst sich als Produkt der berechneten Klassenkonfidenz der Zelle und der Positionswahrscheinlichkeit:

$$m(q)_i = p_{class}(q)_i \cdot p_{pixel}(q)_i \tag{2.44}$$

Das so entstandene ScanGrid wird im weiteren Verlauf als $\mathbf{m}_t^{SS}(\cdot)$ bezeichnet. Abb. 2.13 zeigt ein solches ScanGrid von einer innerstädtischen Verkehrssituation.

(a) (b)

Abbildung 2.13: ScanGrid, erstellt aus einer semantischen Punktwolke. (a) Kamerabild der Situation. (b) ScanGrid $m_t^{SS}(\cdot)$ mit den 'belief masses'. Jedes Pixel repräsentiert eine Gridzelle. Die Farbe kodiert die Hypothese (Lila: $\{M, L\}$; Magenta: $\{S\}$), die 'belief mass' ist als Transparenz des Pixels (Alpha-Kanal) dargestellt.

Belegungskarten

Auch eine Belegungskarte enthält wichtige Informationen über die aktuelle Fahrzeugumgebung. Heutige Ansätze bieten im Gegensatz zur Ursprungspublikation [96] nicht nur eine einzige Belegungswahrscheinlichkeit pro Zelle, sondern weitere Informationen wie bspw. die separate Modellierung von unbekannten Arealen oder die explizite Schätzung der dynamischen Anteile (andere Verkehrsteilnehmer).

Die Belegungskarte ist jedoch insofern ein Spezialfall, weil sie meist die Messdaten aus **mehreren** Zeitschritten enthält und damit bereits zeitlich akkumuliert ist. Der Grund hierfür ist die Modellierung und Schätzung der Dynamik (zeitliche Veränderung), damit auch dynamische Objekte korrekt repräsentiert werden können (siehe u.a. [122]). Dies setzt eine Schätzung des Zellzustandes über mehrere Zeitschritte hinweg und damit die zeitliche Akkumulation der Messdaten voraus.

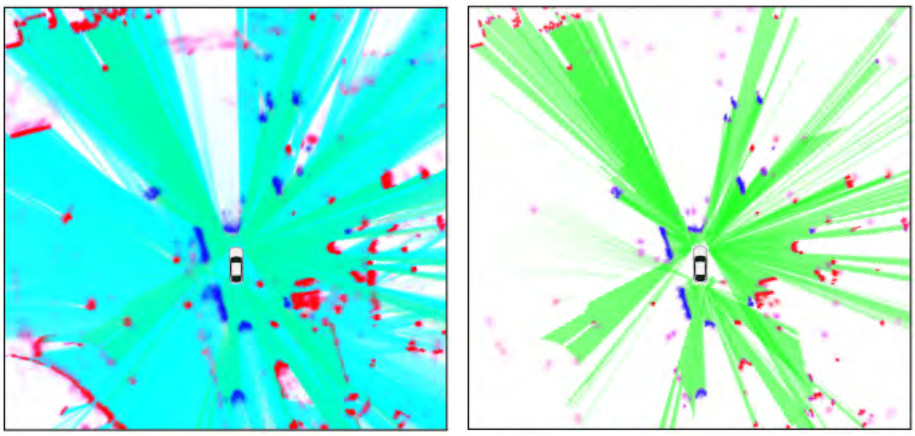
Im vorliegenden Fall findet der Ansatz aus [108] Verwendung, bei dem eine dynamische Belegungskarte mit Hilfe der Dempster-Shafer-Theorie erzeugt wird. Das verwendete 'frame of discernment' ist $\Theta = \{F, S, D\}$. Dabei steht F für die Hypothese 'frei', S für 'belegt durch statisches Objekt' und D für 'belegt durch dynamisches Objekt'.

Die Belegungskarte entsteht durch zeitliche Akkumulation von Belegungskarten-ScanGrids und enthält eine zeitlich akkumulierte Hypothesenschätzung pro Zelle. Die zeitlich akkumulierte Schätzung der Art der Belegung (Hypothese D oder S) wird zusätzlich im Belegungskarten-ScanGrid genutzt, um zwischen statischer und dynamischer Belegung zu unterscheiden (siehe Abb. 2.14). Der belief einer Belegung $Bel(\{S, D\})$ bleibt dabei unverändert, nur die Aufteilung

der 'belief masses' zwischen $\{S\}$, $\{D\}$, und $\{S, D\}$ wird mit Hilfe der zeitlich akkumulierten Schätzung auf folgende Art pro Zelle verändert (siehe [108]):

$$\begin{aligned} m'_t(S) &= m_{1:t}(S) \cdot m_t(SD) \\ m'_t(D) &= m_{1:t}(D) \cdot m_t(SD) \\ m'_t(SD) &= m_t(SD) - m'_t(S) - m'_t(D) \end{aligned} \tag{2.45}$$

Dabei ist $m_{1:t}(\cdot)$ die 'belief mass' aus der akkumulierten Belegungskarte, $m_t(\cdot)$ die ursprüngliche 'belief mass' des Belegungskarten-ScanGrids und $m'_t(\cdot)$ die resultierenden 'belief masses' des ScanGrids.



(a) **(b)**

Abbildung 2.14: Dynamische Belegungskarte (a) zeitlich akkumulierte Belegungskarte. (b) modifiziertes ScanGrid. Die Farben spiegeln die Hypothesen wider. S: rot; D: blau, F: grün; $\{F, D\}$: türkis (Quelle: [108]).

Die 'belief masses' aus dem resultierenden Belegungskarten-ScanGrid müssen nun in 'belief masses' für die Hypothesen aus 2^{Ω}_r überführt werden. Dies kann durch eine Hypothesen-Zuordnung (mit optionaler Gewichtung) erfolgen:

$$\begin{aligned} m'_t(S) &\rightarrow m_t^{OG}(O) \text{ oder } m_t^{OG}(\{S, O\}) \\ m'_t(F) &\rightarrow m_t^{OG}(\{M, L\}) \text{ oder } m_t^{OG}(\{M, L, S\}) \end{aligned}$$

In Abhängigkeit von der Sensorik kann die Belegung einer Zelle zu den Hypothesen $\{O\}$ oder $\{S, O\}$ beitragen - je nachdem, ob Bürgersteige in der Belegungskarte grundsätzlich als Hindernis oder Freiraum erkannt werden (dies ist sensor-abhängig!).

Für die Verwendung der freien Zellen der Belegungskarte gibt es - ähnlich wie im Fall der Fahrstreifenmarkierungen - die Möglichkeit einer konservativen oder optimistischen Ableitung von 'belief masses'. So deuten freie Zellen in der Belegungskarte zwar auf freie Flächen hin, jedoch sind nicht alle freien Flächen (gut) befahrbar. Grünstreifen, Wiesen und Gräben beispielsweise sind (in einer Belegungskarte ohne Höheninformationen) frei, aber nicht befahrbar. Geht man von der optimistischen Annahme einer Befahrbarkeit aller freien Flächen aus, kann eine Zuordnung von „frei" (F) zu $\{M, L\}$ erfolgen. Die konservative Möglichkeit besteht darin, nur die belegten Zellen der Belegungskarte zu berücksichtigen und die 'belief mass' für F nicht zu verwenden. Andere Zuordnung von „frei" zu bspw. $\{M, L, S\}$ sind ebenfalls möglich, und hängen von den Charakteristika der verwendeten Sensorik ab, mit der die Belegungskarte erstellt wird.

Da das Belegungskarten-ScanGrid bereits als Grid vorliegt, ist keine komplexe Projektion der Daten in das zu erzeugende ScanGrid $\mathbf{m}_t^{OG}$ notwendig. Die Grids liegen bereits zellenweise „übereinander".

2.2.2 Fusion und zeitliche Filterung

Die ScanGrids aus dem vorherigen Abschnitt müssen nun fusioniert werden. Damit entsteht pro Gridzelle und Hypothese eine 'belief mass', die die Sensordaten aus allen Quellen (Fahrstreifenmarkierungen, dynamische Objekte, semantische Segmentierung, Belegungskarten) berücksichtigt.

Darüber hinaus müssen die Grids über die Zeit akkumuliert und gefiltert werden, um alle Sensordaten von der Vergangenheit bis zum aktuellen Zeitpunkt t kombinieren zu können.

Hierbei zeigt sich ein weiterer Vorteil einer gridbasierten Repräsentation, da die Grids bereits implizit eine zeitliche und örtliche Korrelation der Sensordaten darstellen. Dadurch kann eine Fusion sehr einfach und flexibel umgesetzt werden.

Im folgenden Abschnitt wird zunächst die Fusion der ScanGrids beschrieben, die mit Hilfe der verschiedenen Sensordaten erzeugt wurden. Anschließend wird in Abschnitt 2.2.2 die Erzeugung der DSTMap erklärt, die das Ergebnis einer zeitlichen Filterung/Akkumulation der fusionierten ScanGrids darstellt.

ScanGrid Fusion

Formal kann die Fusion von ScanGrids beschrieben werden als

$$\mathbf{m}_t^{\text{all}}(\cdot) = \mathbf{m}_t^{\text{LM}}(\cdot) \oplus \mathbf{m}_t^{\text{Obj}}(\cdot) \oplus \mathbf{m}_t^{\text{SS}}(\cdot) \oplus \mathbf{m}_t^{\text{OG}}(\cdot) \tag{2.46}$$

Damit wird ein fusioniertes ScanGrid $\mathbf{m}_t^{\text{all}}(\cdot)$ berechnet, das die ScanGrids (und damit auch die Sensordaten) von allen Datenquellen für einen einzigen Zeitpunkt t beinhaltet.

Im Gegensatz zu beispielsweise der ScanGrid-Fusion aus [122], beinhalten die ScanGrids der vorliegenden Arbeit ausschließlich Informationen über die statische Umgebung, aber keine dynamischen, d.h. zeitlich veränderlichen Anteile. Somit ist es für eine korrekte Fusion hinreichend, lediglich den exakten Aufnahmezeitpunkt der Sensordaten zu kennen - eine Synchronisation der Aufnahmezeitpunkte der einzelnen Sensoren ist **nicht** notwendig.

Die einfachste Form einer Fusion ist stets die Annahme des schlechtest-möglichen Ergebnisses. Im vorliegenden Fall würde dies für die ScanGrid-Fusion bedeuten, jeweils das Minimum der 'belief masses' einer Zelle aus allen ScanGrids zu wählen. Dies ist zugleich auch die defensivste und risikoärmste Variante. Bei einer Iteration über die ScanGrids aller Sensoren (Index k) ist demnach für jede Hypothese A dasjenige ScanGrid $\mathbf{m}_t^j(A)$ gesucht, bei dem die 'belief mass' der Zelle i für die Hypothese A verglichen mit allen Zellen i der anderen ScanGrids am niedrigsten ist:

$$\begin{aligned} j &= \arg\min_k (m_t^k(A)_i) \\ m_t^{\text{all}}(A)_i &= m_t^j(A)_i \end{aligned} \tag{2.47}$$

Beinhalten die ScanGrids jedoch nicht nur 'belief masses' für einzelne Hypothesen, sondern auch für Gruppen (Elemente aus 2_r^Ω und nicht nur aus Ω), müssen für die Fusion die 'belief masses' aller Untergruppen berücksichtigt werden. Dafür wird für jede Hypothese $A \subseteq \Omega$ derjenige Sensor j gesucht, bei dem die 'belief mass' $m_t^{j(A)}(A)_i$ für die Zelle i am niedrigsten ist. Da die so gefundenen 'belief masses' der Zelle i in der Summe nicht notwendigerweise 1 ergeben, müssen die entstandenen 'belief masses' normalisiert werden.

$$\begin{aligned} \underset{A\subseteq\Omega}{j(A)} &= \arg\min_k \underset{A\subseteq\Omega}{m_t^k(A)_i} \\ m_t^{\text{all}}(A)_i &= \frac{m_t^{j(A)}(A)_i}{\sum_{A\subseteq\Omega} m_t^{j(A)}(A)_i} \end{aligned} \tag{2.48}$$

Eine intelligentere Fusion als die Suche der kleinsten 'belief mass' stellt die Fusion mit der „Dempster rule of combination" (Gl. 2.16) dar. Dabei werden die 'belief masses', die einen Konflikt der Messungen hervorrufen, gleichmäßig auf

alle Hypothesen verteilt. Neben der „Dempster Rule of combination" existieren viele weitere Regeln zur Kombination von Massen. So wird in [131] für die Kombination von 'belief masses' zweier Sensoren die „averaging fusion"-Regel vorgeschlagen:

$$\begin{cases} m^1(A) \oplus^{AR} m^2(A) = \frac{m^1(A)m^2(\Omega)+m^1(\Omega)m^2(A)}{m^1(\Omega)m^2(\Omega)} \\ \\ m^1(\Omega) \oplus^{AR} m^2(\Omega) = \frac{2m^1(\Omega)m^2(\Omega)}{m^1(\Omega)m^2(\Omega)} \end{cases} \tag{2.49}$$

Diese ist jedoch nicht assoziativ, es gilt:

$$m^1(A) \oplus^{AR} \left(m^2(A) \oplus^{AR} m^3(A) \right) \neq \left(m^1(A) \oplus^{AR} m^2(A) \right) \oplus^{AR} m^3(A) \tag{2.50}$$

Damit führt die Kombination von mehr als 2 Sensoren mit dieser Regel zu einem unerwünschten Effekt, da die Reihenfolge der Sensoren bei der Fusion Einfluss auf das Ergebnis hat. Diese Regel ist also im vorliegenden Fall nicht geeignet für die ScanGrid-Fusion.

Auch die „cumulative rule" von Josang [131], hier mit $\oplus^J$ bezeichnet ist nicht geeignet für die ScanGrid-Fusion, da nur die 'belief masses' gleicher Hypothesenmengen kombiniert und die Untermengen ignoriert werden, wie das Beispiel in Tabelle 2.1 zeigt.

	S_1	S_2	$max(S_1, S_2)$	$m_1 \oplus^{max} m_2$	$m_1 \oplus^D m_2$	$m_1 \oplus^J m_2$
$m(M)$	0	0	0	0	0	0
$m(S)$	0.8	0	0.8	0.444	0.8	0.706
$m(O)$	0	0	0	0	0	0
$m(\{S, O\})$	0	0.4	0.4	0.222	0.08	0.118
$m(\Omega)$	0.2	0.6	0.6	0.333	0.12	0.176

Tabelle 2.1: Exemplarische Messung mit zwei Quellen S_1 und S_2. Vergleich der Ergebnisse bei unterschiedlichen Fusionsregeln.

Bei $max(S_1, S_2)$ wird die 'belief mass' verwendet, die im Vergleich der beiden Sensoren größer ist. Da die Summe der 'belief masses' damit nicht notwendigerweise 1 ergibt, ist eine Normierung erforderlich (Ergebnisse in Spalte $m_1 \oplus^{max} m_2$). Da $m(S)$ keinen Konflikt mit $m(\{S, O\})$ hat, sollte $m(S)$ nach der Kombination nicht kleiner als 0.8 sein. Sowohl bei der Verwendung des Maximums, als

auch bei der Josang-Regel ($m_1 \oplus^J m_2$) ist dies jedoch nicht der Fall, da die Obermengen bei der Fusion ignoriert werden. Nur die „Dempster rule of combination" liefert für das gezeigte Beispiel plausible Ergebnisse. Daher wurde für die vorligende Arbeit die „Dempster rule of combination" für die ScanGrid-Fusion verwendet.

Zeitliche Filterung

Ziel der zeitlichen Filterung ist die Erstellung der DSTMap aus den ScanGrids. Hierzu werden die fusionierten ScanGrids aus dem vorherigem Abschnitt über die Zeit akkumuliert. Man spricht in diesem Zusammenhang auch von einer zeitlichen Filterung. So entsteht aus dem fusionierten ScanGrids $\mathbf{m}_t^{\text{all}}(\cdot)$ über die Zeit die DSTMap $\mathbf{m}_{1:t}^{\text{all}}(\cdot)$, die die Informationen (Sensordaten) aus der Vergangenheit bis zum aktuellen Zeitpunkt t berücksichtigt. Dabei wird jeweils die DSTMap aus dem vorangegangenen Zeitschritt mit dem fusionierten ScanGrid des aktuellen Zeitschritts fusioniert. Das Ergebnis ist die DSTMap des aktuellen Zeitschritts:

$$\mathbf{m}_{1:t}^{\text{all}}(\cdot) = \mathbf{m}_{1:t-1}^{\text{all}}(\cdot) \oplus \mathbf{m}_t^{\text{all}}(\cdot) \tag{2.51}$$

Analog zur ScanGrid Fusion stellt sich für die zeitliche Filterung die Frage nach der richtigen Kombinationsregel. „Dempster's rule of combination" wird in der Literatur häufig dafür kritisiert, dass die Ergebnisse in gewissen Situationen nicht intuitiv seien [153, 131, 132, 130]. Dies trifft im Wesentlichen auf den Fall zu, bei dem zwischen den Messungen ein hoher Konflikt auftritt, und gleichzeitig die 'belief mass' für $m(\Omega)$ (*unbekannt*) gleich 0 ist. Illustrieren lässt sich dies an Zadeh's Beispiel [153]: Seien $m_1(A) = 0.1, m_1(B) = 0.9, m_1(C) = 0$ Massen eines Sensors für die Hypothesen A, B und C, und $m_2(A) = 0.1, m_2(B) = 0, m_2(C) = 0.9$ die Messungen von Sensor 2, so ergibt sich nach Dempster's Regel $m_1(A) \oplus^D m_2(A) = 1$ (die übrigen Kombinationen sind 0). Die Hypothese, die beide Sensoren als sehr unwahrscheinlich eingestuft haben, ist **mit Sicherheit** die korrekte. Darüber hinaus konvergiert die 'belief mass' für $m(\Omega)$ gegen 0 bei der Verwendung der Dempster rule. Dies ist im vorliegenden Fall jedoch nicht erwünscht, da stets eine gewisse Restunsicherheit bestehen bleiben sollte und das resultierende $m(\Omega) > 0$ sein sollte.

Daher wird in dieser Arbeit für die zeitliche Filterung eine Abwandlung der „conjunctive rule" [131] verwendet. Die „conjunctive rule" stellt die nicht-normalisierte Variante der Dempster rule dar:

$$m_{1:t}^{\text{all}}(A) = m_{1:t-1}^{\text{all}}(A) \oplus^{CR} m_t^{\text{all}}(A),\ A \neq \varnothing \tag{2.52}$$

$$= \sum_{A_i \cap B_j = A} m_{t-1}(A_i) \cdot m_t(B_j) \tag{2.53}$$

Die hier verwendete Kombinationsregel

$$m_{1:t}^{\text{all}}(A) = m_{1:t-1}^{\text{all}}(A) \oplus^{C} m_t^{\text{all}}(A) \tag{2.54}$$

$$= N \cdot \left(m_{1:t-1}^{\text{all}}(A) \oplus^{CR} m_t^{\text{all}}(A) \right), \text{für} A \neq \Omega \tag{2.55}$$

und

$$m_{1:t}^{\text{all}}(\Omega) = \begin{cases} \theta & \text{, wenn} \frac{1}{1-k} m_1(\Omega) m_2(\Omega) < \theta \\ N \cdot m_{1:t-1}^{\text{all}}(\Omega) m_t^{\text{all}}(\Omega) & \text{, sonst} \end{cases} \tag{2.56}$$

mit

$$N = \begin{cases} \frac{1-\theta}{1-k-m_1(\Omega)m_2(\Omega)} & \text{, wenn} \frac{1}{1-k} m_1(\Omega) \cdot m_2(\Omega) < \theta \\ \frac{1}{1-k} & \text{, sonst} \end{cases} \tag{2.57}$$

und

$$k = \sum_{A_i \cap B_j = \varnothing} m_1(A_i) m_2(B_j) \tag{2.58}$$

stellt eine normalisierte Variante dar, die im Unterschied zur Dempster-Regel stets garantiert, dass $m_{1:t}^{\text{all}}(\Omega) = m_{1:t-1}(\Omega) \oplus^{C} m_t(\Omega) > \theta$ ist. Durch den modifizierten Normalisierungsfaktor ist die aus der Kombination resultierende Unsicherheit in Form von $m(\Omega)$ stets größer als eine vorab definierte minimale Unsicherheit θ. Tabelle 2.2 verdeutlicht dies anhand eines Beispiels:

	S_1	S_2	$m_1 \oplus^D m_2$	$m_1 \oplus^J m_2$	$m_1 \oplus^C m_2$
$m(M)$	0.1	0.6	0.471	0.412	0.447
$m(S)$	0.6	0.005	0.322	0.356	0.306
$m(O)$	0	0.095	0.05	0.056	0.047
$m(\Omega)$	0.3	0.3	0.157	0.176	0.2

Tabelle 2.2: Sowohl bei der Dempster-Regel als auch bei der Josang-Regel konvergiert $m(\Omega)$ über die Zeit zu 0. In diesem Beispiel wurde für die Minimal-Unsicherheit $\theta = 0.2$ angenommen.

2.3 Ergebnisse

Der folgende Abschnitt zeigt qualitative Ergebnisse der gridbasierten Umfeldmodellierung mit der DSTMap von verschiedenen Verkehrssituationen.

Für die Darstellung der Grids wird die folgende Farbkodierung verwendet:

Abbildung 2.15: Farbkodierung der Hypothesen.

Der Transparenz-Wert jedes Pixels (Alpha-Kanal) entspricht der 'belief mass' der entsprechenden Hypothese bzw. Hypothesenmenge.

Zum Vergleich ist bei jeder Situation die mit [108] erzeugte Belegungskarte mit abgebildet. Dabei wurde die folgende Farbgebung verwendet:

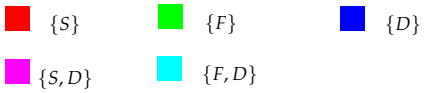
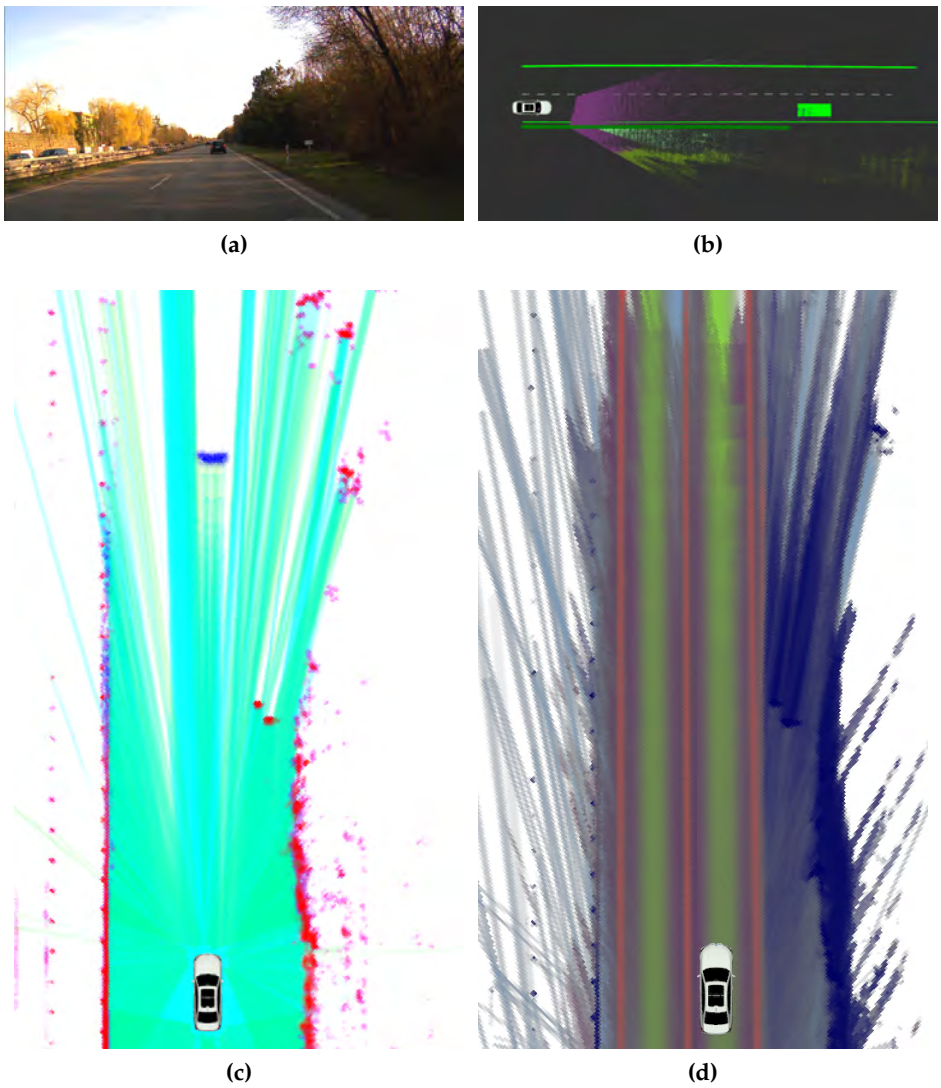


Abbildung 2.16: Farbkodierung der Hypothesen der Belegungskarte, wobei hier F für 'frei', S für 'statisch belegt', und D für 'dynamisch belegt' steht.

Verkehrssituation 1: Bundesstraße

Die erste Situation (Abb. 2.17), zeigt ein typisches Verkehrsszenario auf einer Bundesstraße mit zwei parallel verlaufenden Fahrstreifen und einem vorausfahrenden Fahrzeug. Abb. 2.17a zeigt die Sensordaten aus dem aktuellen Zeitschritt t: Fahrstreifenmarkierungen, Punktwolke der semantischen Segmentierung und ein vorausfahrendes dynamisches Objekt. In Abb. 2.17b ist die erzeugte DSTMap dargestellt. Zum Vergleich ist in Abb. 2.17c eine Belegungskarte der

Situation abgebildet. In der DSTMap sind die zwei parallel verlaufenden Fahrstreifen inkl. ihrer Fahrstreifengrenzen (rot), sowie die Fläche der Fahrstreifen (grün) klar erkennbar, wohingegen die Belegungskarte beide Fahrstreifen als zusammenhängende freie Fläche darstellt. Darüber hinaus ist die Grünfläche rechts neben dem Ego-Fahrstreifen in der Belegungskarte fälschlicherweise als freie Fläche erkannt worden.



(a) **(b)** **(c)** **(d)**

Abbildung 2.17: Verkehrssituation auf einer Bundesstraße mit zwei parallel verlaufenden Fahrstreifen und einem vorausfahrendem Fahrzeug. Das Ego-Fahrzeug befindet sich auf dem rechten Fahrstreifen. (a) Kamerabild der Situation, (b) Sensordaten, (c) Belegungskarte, (d) DSTMap.

Verkehrssituation 2: Baustelle

Abb. 2.18 zeigt eine Verkehrssituation in einer Baustelle. Baustellen stellen stets eine besondere Herausforderung dar, weil der Straßenverlauf oft nur schwer zu erkennen ist. In der vorliegenden Situation ist der Fahrstreifenverlauf des Ego-Fahrstreifens hauptsächlich durch Baken und teilweise durch gelbe Fahrstreifenmarkierungslinien vorgegeben.

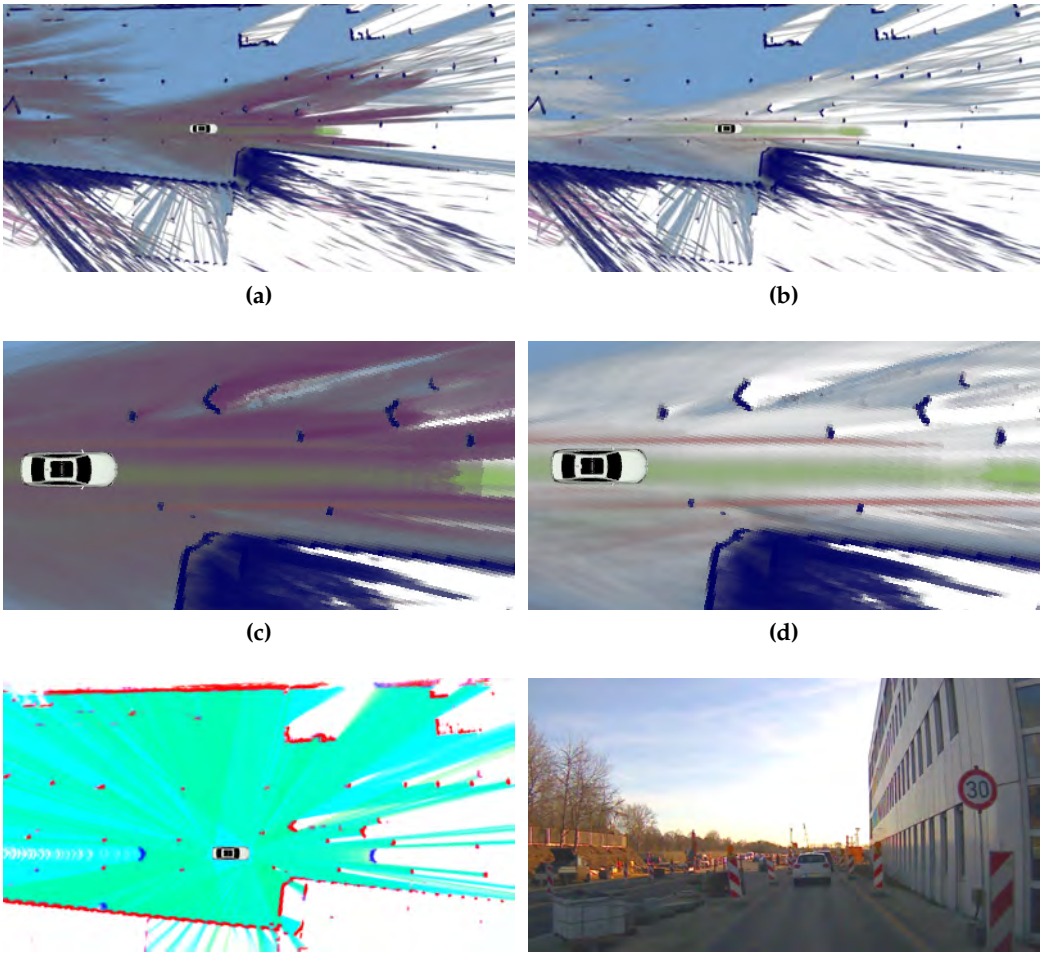


Abbildung 2.18: Verkehrssituation in einer Baustelle. Jeweils vor und hinter dem Ego-Fahrzeug befindet sich ein anderes fahrendes Fahrzeug. Der Ego-Fahrstreifen ist hauptsächlich durch Baken und nur teilweise durch Fahrstreifenmarkierungslinien vorgegeben. (a) DSTMap mit allen Hypothesen, (b) DSTMap ohne die Hypothese $\{M, L\}$ zur besseren Sichtbarkeit der 'belief mass' für $\{L\}$ (Lane, grün) und $\{M\}$ (Markierungslinie, rot) , (c) Aussschnittsvergrößerung von (a), (d) Aussschnittsvergrößerung von (b), (e) Belegungskarte, (f) Kamerabild der Situation.

In der DSTMap in Abb. 2.18a und 2.18b ist der eigene Fahrstreifen gut zu erkennen. In der Belegungskarte kann im Vergleich zur DSTMap nicht zwischen Ego-Fahrstreifen und sonstigen Freiflächen unterschieden werden. Einzig die Baken und das Gebäude sind gut als statische Hindernisse (rot) zu erkennen.

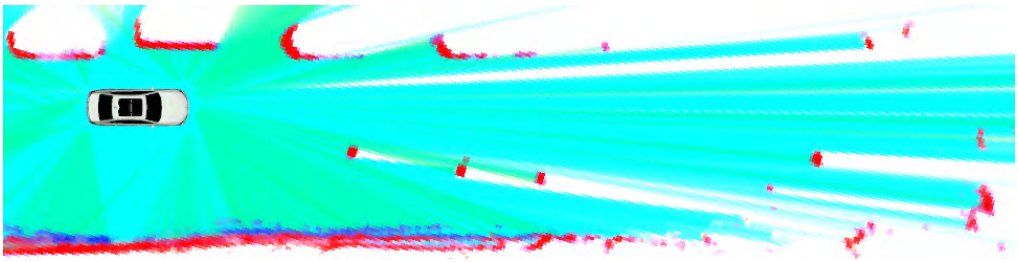
Verkehrssituation 3: Geänderter Fahrstreifenverlauf in einer Baustelle

Die Situation in Abb. 2.18 zeigt einen geänderten Fahrstreifenverlauf in einer Baustelle, in der der Ego-Fahrstreifen über den Bürgersteig geführt wird. Die bildbasierte semantische Segmentierung liefert auch für diesen schwierigen Fall eine gute Klassifzierung der Bildpixel: Der Bürgersteig ist klar zu erkennen. Abb. **??** zeigt die Sensordaten des aktuellen Zeitschritts. Trotz einer fehlerhaft erkannten Fahrstreifenmarkierung links vom Ego-Fahrzeug, zeigt die DSTMap deutlich den eigenen Fahrstreifen. Die staubedingt stehenden Fahrzeug auf dem Nachbarfahrstreifen werden in der Belegungskarte und damit auch in der für die DSTMap verwendeten Objektliste fälschlicherweise als statisch klassifiziert. In der Belegungskarte wird der Bürgersteig fälschlicherweise als frei erkannt. Im Gegensatz dazu ist er in der DSTMap korrekt klassifiziert.

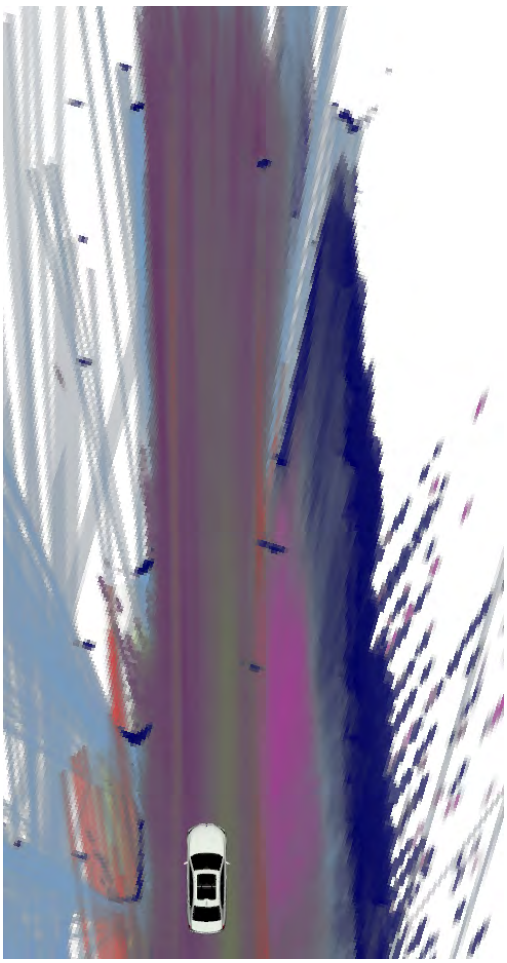
Die Situation verdeutlicht sowohl den Vorteil der DSTMap gegenüber der Belegungskarte, als auch die Robustheit der gridbasierten Sensordatenfusion. Trotz einiger Sensorfehler (falsch erkannte Fahrstreifenmarkierungslinie, dynamische Objekte als statisch in der Belegungskarte) bildet die DSTMap auch in dieser sehr komplexen Situation die Umgebung und im Besonderen den eigenen Fahrstreifen gut ab.

(a)

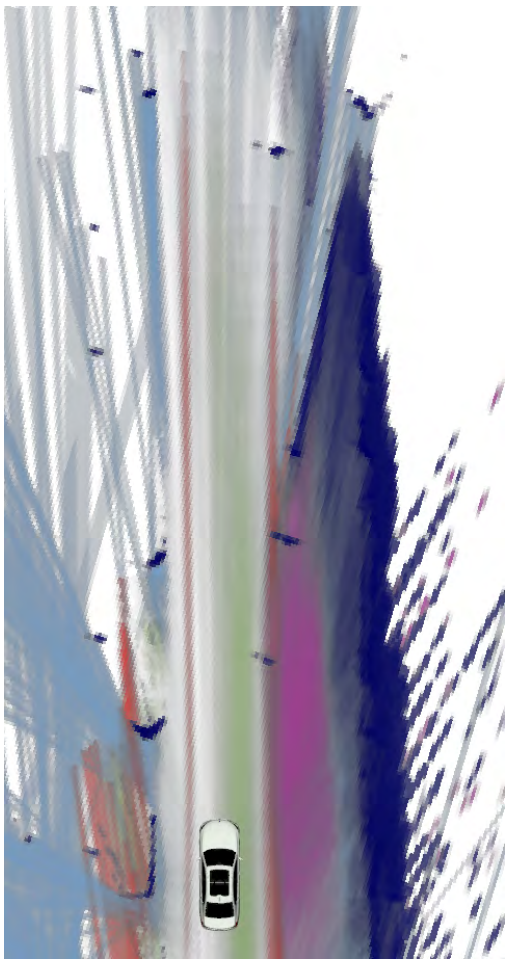
(b)



(c)



(d)



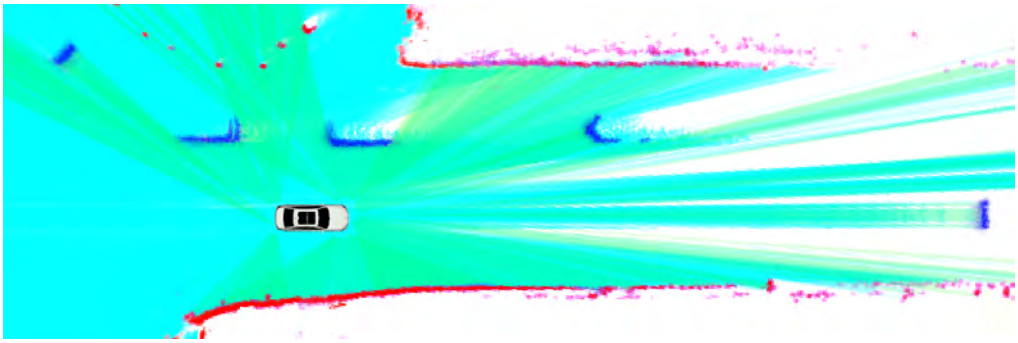
(e)

Abbildung 2.18: (a) Kamerabild der Situation, (b) Ergebnis der semantischen Segmentierung, (c) Belegungskarte (d) DSTMap mit allen Hypothesen, (e) DSTMap ohne die Hypothese $\{M, L\}$ zur besseren Sichtbarkeit der 'belief mass' für $\{L\}$ (Lane).

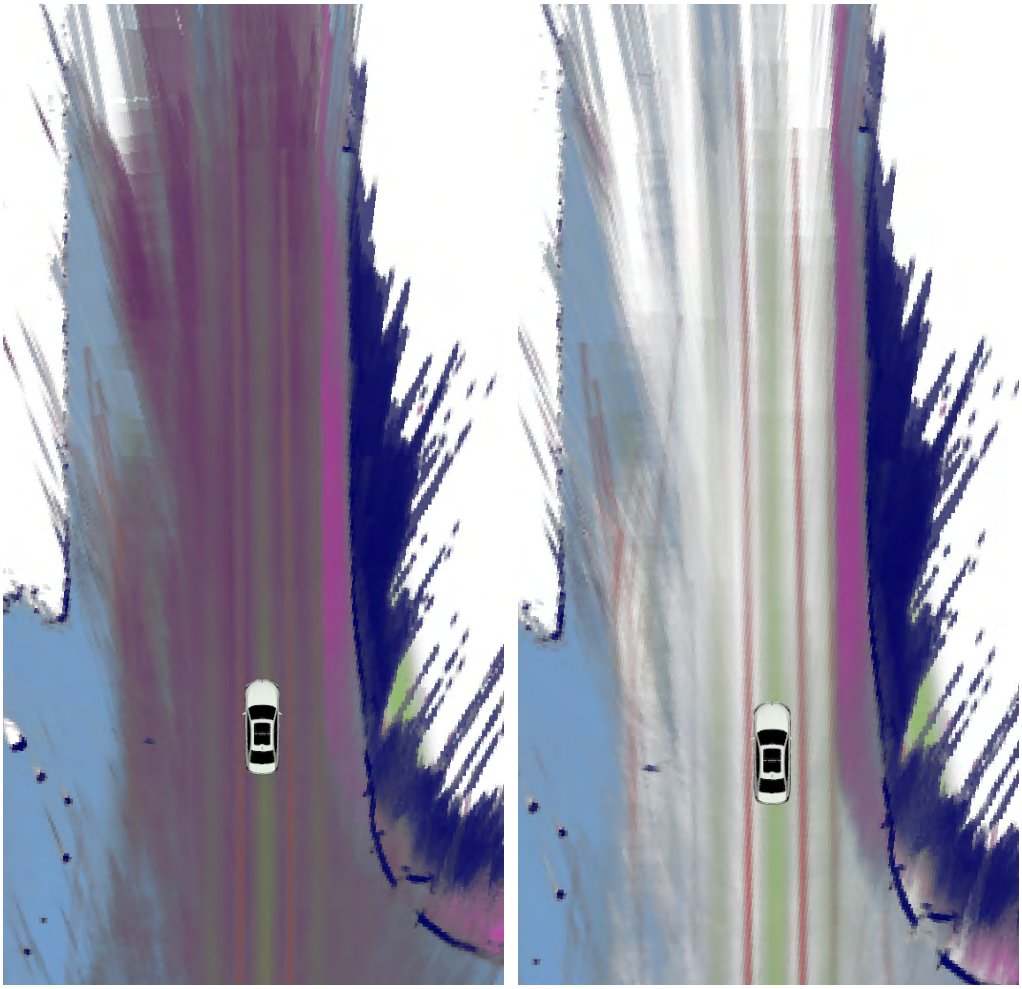
Verkehrssituation 4: Innerstädtische Situation mit einem Fahrradweg

Die nächste Situation in Abb. 2.18 zeigt eine innerstädtische Situation mit einem Fahrradweg auf der Fahrbahn. In der Belegungskarte sind zwar die dynamischen Objekte gut erkennbar, aber sowohl Fahrradweg als auch Bürgersteig sind als freie Flächen klassifiziert. Auch die Fahrstreifengrenze zum Fahrstreifen des Gegenverkehrs existiert nicht in der Belegungskarte. Hier zeigt sich ein weiterer Vorteil der DSTMap, da die Fläche des Fahrradwegs zwar eine gewisse 'belief mass' für $\{M, L\}$ (Markierungslinie: M oder Fahrstreifen L: Lane) besitzt (siehe 2.18c), jedoch kaum 'belief mass' für L: Lane (siehe 2.18d). Darüber hinaus ist der Ego-Fahrstreifen in der DSTMap sowohl zum Fahrradweg als auch zum Gegenfahrstreifen klar abgegrenzt. In Abb. 2.18d ist zusätzlich auch die Grenze zwischen Fahrradweg und Bürgersteig, sowie die Grenze zwischen Bürgersteig (magenta) und statischer Begrenzung (blau) gut erkennbar. Die statische Begrenzung entspricht dem Gebüsch auf der rechten Seite des Ego-Fahrzeugs.

(a)



(b)



(c) (d)

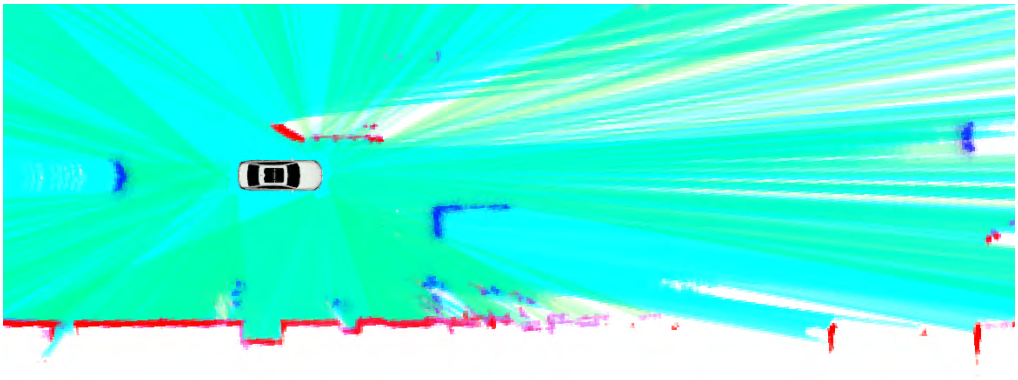
Abbildung 2.18: Innerstädtische Situation mit einem Fahrradweg auf der Fahrbahn. Bilder analog zu den vorherigen Situationen.

Verkehrssituation 5: Innerstädtische Situation mit Bürgersteigen

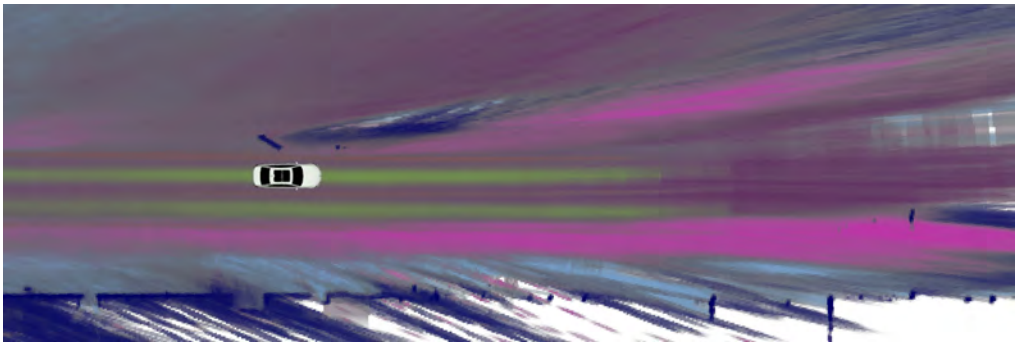
Die letzte Situation, Abb. 2.18, zeigt ein innerstädtisches Szenario mit einer Straßenbahnhaltestelle auf der linken Seite, die als eine Art Bürgersteig mit „Wartehäuschen" aufgefasst werden kann. Die zweispurige Fahrbahn ist auch auf der anderen Seite durch einen Bürgersteig begrenzt. In der Belegungskarte finden sich beide Bürgersteige fälschlicherweise als Freiflächen wieder. Nur das Wartehäuschen ist auf Höhe des Ego-Fahrzeugs in der Belegungskarte korrekt

als statisches Hindernis repräsentiert. Hingegen sind in der DSTMap die Bürgersteige deutlich als solche zu erkennen (magentafarbene Bereiche). Auch die Separierung der beiden Fahrstreifen ist in der DSTMap gut zu erkennen.

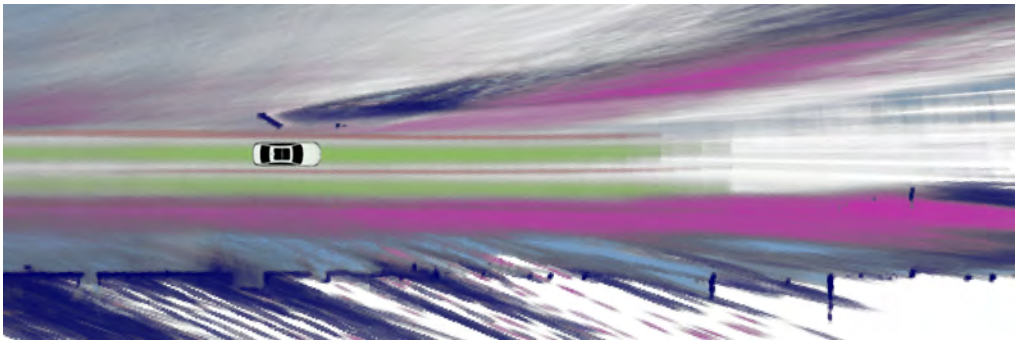
(a)



(b)



(c)



(d)

Abbildung 2.18: Innerstädtische Situation mit Bürgersteigen. Bilder analog zu vorherigen Situationen.

2.4 Zusammenfassung

Die Umfeldmodellierung der aktuellen Fahrzeugumgebung ist die Grundvoraussetzung für das autonome Fahren. Gridbasierte Ansätze haben sich dabei als besonders geeignet erwiesen, da sie einerseits eine flexible Modellierung der Umgebung erlauben und andererseits eine gute Abstraktionsebene für die Messdaten der Sensoren bieten.

In diesem Kapitel wurde ein neuartiger gridbasierter Ansatz in Form der DST-Map präsentiert, um ein konsistentes Abbild der statischen Straßeninfrastruktur zu erzeugen. Er besteht aus einer gridbasierten Sensordatenfusion, die ausschließlich Messdaten verwendet, die während der Fahrt zur Verfügung stehen. Damit ist der Ansatz unabhängig von offline HD-Karten und einer hochgenauen Lokalisierung.

Unter Verwendung der „Dempster-Shafer Theory of Evidence" wurde für eine korrekte Abbildung der Straßeninfrastruktur ein neuartiges 'frame of discernment' definiert, das Hypothesen enthält, die die Straßeninfrastruktur widerspiegeln.

Je nach Sensor-Charakteristik können aus den Messdaten 'belief masses' nicht nur für einzelne Hypothesen, sondern auch für verschiedenste Hypothesengruppen abgeleitet werden. Der vorgestellte Ansatz ist sehr flexibel und leicht um neue Sensordaten oder Hypothesen erweiterbar.

Die Erstellung der DSTMap wurde im vorliegenden Fall anhand von Fahrstreifenmarkierungslinien, von erkannten dynamischen Objekte, von Punktwolken einer bildbasierten semantischen Segmentierung und von Belegungskarten demonstriert. Darüber hinaus wurden die Ergebnisse der DSTMap mit denen der

Belegungskarte verglichen, und die Vorteile der DSTMap gegenüber der Belegungskarte anhand von verschiedenartigen Situationen präsentiert. Selbst in sehr komplexen Baustellen-Situationen lässt sich in der DSTMap zumindest der Ego-Fahrstreifen stets gut erkennen. Es zeigt sich eine deutliche Überlegenheit des vorgestellten Ansatzes.

Kapitel 3

Extraktion von Fahrstreifen

Das folgende Kapitel befasst sich mit der Frage, wie aus DSTMap, die im vorangegangenen Kapitel vorgestellt worden ist, potentielle Fahrstreifen extrahiert werden können.

Zunächst wird in der Einleitung ein Einblick in den Stand der Technik gegeben. Dabei werden auch Methoden zur Extraktion von anderweitigen Informationen aus Grids diskutiert, sofern sie für die vorliegende Arbeit von Relevanz sind.

Anschließend folgt die Vorstellung eines neuartigen Konzepts der iterativen Pfadplanung, das zur Extraktion von Fahrstreifen und deren Grenzen verwendet wird.

3.1 Einleitung

In den letzten Jahren hat der Einsatz von Belegungskarten im Bereich der Umfelderkennung stark zugenommen. Der Fokus liegt dabei stets auf der Erkennung von belegten Arealen, um einerseits Kollisionen mit unerwarteten Hindernissen (bspw. verlorene Ladung auf der Straße) vermeiden und andererseits auch durch unbekanntes Terrain navigieren zu können. Zur Extraktion von Informationen aus Belegungskarten sowie zur Interpretation der belegten Zellen existieren weit weniger Publikationen als zur eigentlichen Erstellung der Belegungskarte. Die wenigen Publikationen zur Extraktion von Strukturen aus Belegungskarten beschränken sich meist auf die Extraktion der Randbebauung und freier Areale. Allerdings ist deren Nutzen in Bezug auf das automatisierte Fahren begrenzt (siehe hierzu auch Kap. 1.3), da Randbebauungen wenig über den Verlauf der Fahrstreifen aussagen und freie Areale nicht zwangsläufig befahrbar sind.

Im Gegensatz zur Extraktion von Randbebauungen aus Belegungskarten, wird in diesem Kapitel eine neue Methode beschrieben, mit deren Hilfe Fahrstreifen

extrahiert werden können. Dennoch widmet sich der Abschnitt über den Stand der Technik zunächst der allgemeinen Frage, wie Informationen aus einer Belegungskarte extrahiert und interpretiert werden können, da manche Ideen in modifizierter Form auch für die Extraktion von Fahrstreifen aus der DSTMap verwendet werden können.

3.1.1 Stand der Technik

Neben den zahlreichen Veröffentlichungen zu Abwandlungen der Belegungskarte (siehe Kap. 2.1.1), existieren auch einige Verfahren zur Extraktion von Informationen aus Belegungskarten. So findet sich beispielsweise in [154] eine Methode zur Erkennung von dynamischen Objekten in Belegungskarten. Bewegte Objekte hinterlassen in Belegungskarten charakteristische Muster („occupancy trails"). Diese Muster lassen sich aus der Belegungskarte extrahieren und damit als ein dynamisches Objekt identifizieren.

Der weitaus größte Teil der Arbeiten zur Extraktion von Informationen aus Belegungskarten beschäftigt sich mit der Randbebauungs- und Freiraumextraktion. Meist wird dabei die Fahrzeugumgebung in einen linken und einen rechten Bereich unterteilt und in den beiden Bereichen - unabhängig voneinander - die jeweilige Randbebauung bzw. der Freiraum extrahiert [155, 156, 106, 154, 104]. Für die Separierung können unterschiedliche Verfahren angewendet werden. In [106, 155, 154] wird eine imaginäre Linie entlang der aktuellen Fahrzeugausrichtung erzeugt, die die Trennung in einen linken und rechten Bereich darstellt. In [156] hingegen wird die nach vorne prädizierte Fahrzeugtrajektorie als Trennlinie verwendet.

Ein anderer Ansatz ist in [136] zu finden. Dort wird für die Randbebauungsextraktion ein geometrisches Modell direkt in das Grid „gelegt" und die Modellparameter werden durch eine histogrammbasierte Optimierungsmethode bestimmt.

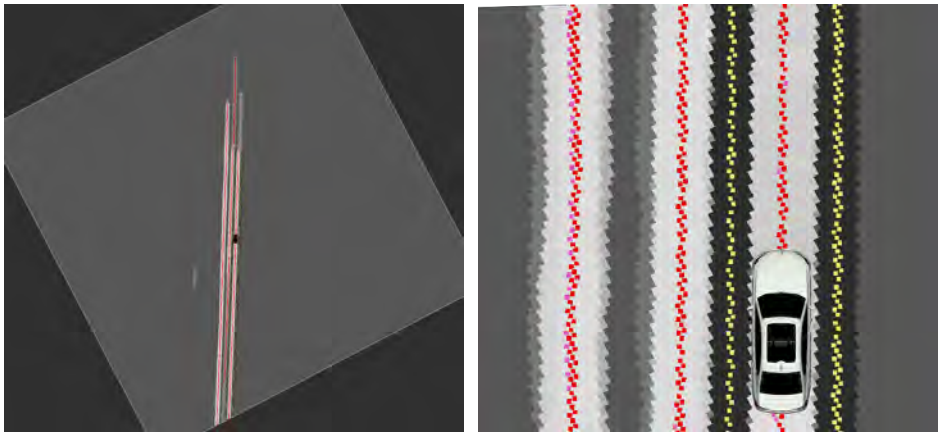
Neben den Verfahren zur Randbebauungsextraktion, finden sich auch Ansätze, die weitergehende Informationen aus Grids extrahieren. So wird in [157] eine Methode beschrieben, bei der ein partikelfilter-basiertes Framework zusammen mit einer Höhenkarte der Umgebung ein topologisches Straßenmodell extrahiert. Dabei werden der Kreuzungsmittelpunkt sowie ein Klothoidenmodell pro Abzweigung geschätzt. Die Anzahl der Abzweigungen muss jedoch im Vorfeld bereits bekannt sein und kann daher nicht als Basis für eine rein sensorbasierte Straßenmodellschätzung verwendet werden. Darüber hinaus sind Klothoidenmodelle nur bedingt geeignet, komplexe Fahrstreifenverläufe, wie sie oftmals in Baustellen zu finden sind, korrekt abzubilden.

Daneben gibt es weitere Ansätze, Strukturen mit Hilfe der digitalen Bildverarbeitung aus Grids zu extrahieren. Dabei wird das Grid als Bild aufgefasst, bei dem jede Zelle des Grids genau einem Bildpixel entspricht. Dies eröffnet die Möglichkeit, etliche aus der digitalen Bildverarbeitung bekannten Verfahren und Algorithmen anwenden zu können, um beispielsweise den Freiraum zu extrahieren. In [123] wird der Freiraum in parametrisierter Form aus einem Grid extrahiert, das mit Sensordaten von Radarsensoren und einer Stereo-Kamera erzeugt wurde. Anschließend wird eine Erosion mit einem Kreis als strukturierendem Element auf dem Grid durchgeführt. Die Dimension des Kreises orientiert sich dabei an den Dimensionen des Ego-Fahrzeugs. Die Erosion führt dazu, dass die freien Bereiche, in die das Fahrzeug nicht hineinpassen würde, entfernt werden. Desweiteren werden größere freie Areale mit nur einer sehr kleinen Verbindung voneinander separiert. Darauf folgt ein „connected component labeling", um die einzelnen Areale zu identifizieren sowie die Auswahl desjenigen Areals, in dem sich das Ego-Fahrzeug aktuell befindet.

Auch die aus der digitalen Bildverarbeitung bekannte Skelettberechnung von Bildern kann für die Extraktion von Informationen aus Grids verwendet werden. Im Rahmen der vorliegenden Arbeit ist ein Ansatz getestet worden, der die Ergebnisse einer Skelettberechnung verwendet, um aus dem Grid eine Punktwolke zu erzeugen. Jeder Punkt der Wolke repräsentiert dabei einen potentiellen Fahrstreifenmittelpunkt. Anschließend werden mit Hilfe des RANSAC-Verfahrens einzelne Fahrstreifen bestimmt. Ein Grid mit dem Ergebnis der Skelettberechnung ist in Abb. 3.1 dargestellt. Alternativ lassen sich auch andere Methoden wie bspw. die aus [158] verwenden, um geometrische Modelle in Punktwolken zu finden. Jedoch sind die meisten Verfahren nicht in angemessener Rechenzeit durchführbar und oftmals anfällig gegenüber Messfehlern.

Eine andere Methode wird in [1] beschrieben, die ebenfalls im Verlauf der vorliegenden Arbeit entwickelt und getestet wurde. Es werden dabei die lokalen Maxima zeilenweise aus einem Grid extrahiert, das pro Zelle eine Fahrstreifenmittenwahrscheinlichkeit beinhaltet. Durch ein einfaches Clustering werden die so gefundenen Maxima assoziiert und es entstehen die Mittellinien der einzelnen Fahrstreifen. In Abb. 3.2 ist das Verfahren schematisch dargestellt.

Im Gegensatz zur Optimierung eines geometrischen Modells zur Repräsentation bzw. Interpolation der Punkte, lässt sich ein Grid als auch Graph auffassen, bei dem die Zellen als Knoten und die Nachbarschaft der Zellen als Kanten betrachtet werden können. Dies ermöglicht den Einsatz einer ganzen Reihe von Algorithmen, da viele bekannte Probleme aus der Informatik als Graphenproblem formuliert und etliche Lösungen bereits bekannt sind. So berechnet beispielsweise der Pfadplanungsalgorithmus A* den kürzesten Pfad zwischen einem gegebenen Startknoten und einem bestimmen Zielknoten in einem Graph,



(a) (b)

Abbildung 3.1: Ein Grid, das pro Zelle eine Fahrstreifenmittenwahrscheinlichkeit beinhaltet, dargestellt als Graustufen-Bild. Je weißer der Pixel, desto höher die Wahrscheinlichkeit einer Fahrstreifenmitte. Je dunkler der Pixel, desto höher die Wahrscheinlichkeit für eine Fahrstreifengrenze. Die farbigen Punkte zeigen das Ergebnis einer Skelettberechnung für die Fahrstreifenmitte (rot) und die Fahrstreifengrenzen (gelb). (a) Gesamtes Grid, (b) Vergrößerung rund um das Ego-Fahrzeug.

dessen Kanten gewichtet sind. A* nutzt dabei eine Heuristik, um eine zielgerichtete und effiziente Suche vom Startknoten zum Zielknoten durchzuführen. Ein weiteres bekanntes Beispiel ist der Dijkstra-Algorithmus, der die Kosten von jedem Knoten des Graphen zu einem Zielknoten berechnet. Eine Übersicht über Pfadplanungsalgorithmen findet sich unter anderem in [159].

Es existieren zahlreiche Ansätze zu Suchaufgaben in Graphen, unter anderem für die Bewegungsplanung von Greifarmen, Robotern und Fahrzeugen. Im Zuge dessen wurde der bekannteste Suchalgorithmus A* vielfach abgewandelt und erweitert. Dynamic A* (D*) [160, 161] ist in der Lage, bei Veränderung der Umgebung entsprechend schnell neu zu planen. Dabei werden die bereits gefundenen Lösungen inkrementell repariert, wenn sich der zugrundeliegende Suchbaum ändert. Daher findet dieser Algorithmus häufig Anwendung bei der Bewegungsplanung von mobilen Robotern in dynamischen Umgebungen. Eine andere Abwandlung von A* ist der „Anytime Repairing A*" [162], bei dem sehr schnell ein potentiell suboptimaler Pfad gefunden wird und ein optimaler Pfad mit zunehmender Rechenzeit gesucht wird.

Neben dem weit verbreiteten A*-Algorithmus, lassen sich auch häufig „Rapidly Exploring Random Trees (RRT)" [163] für die Lösung von Pfadplanungsproblemen finden. Ähnlich wie bei A* wird ein Suchbaum aufgebaut, allerdings wird im Gegensatz zu A* nicht derjenige Knoten expandiert, der aktuell die

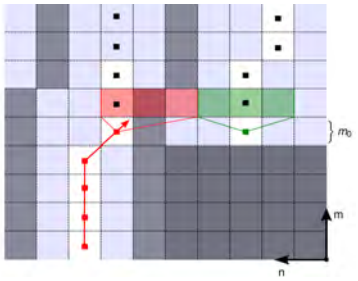


Abbildung 3.2: Extraktion der Maxima aus einem Grid mit anschließendem lokalen Clustering zur Bildung eines Fahrstreifens bzw. dessen Mittellinie [1].

geringsten Kosten aufweist. Stattdessen wird der Suchbaum in Richtung zufällig gewählter Stichproben des Suchraumes expandiert. Wird die Stichprobenziehung uniform ausgeführt, ist die Wahrscheinlichkeit der Expansion eines bekannten Knotens im Suchbaum proportional zur Größe seiner Voronoi-Region. Der Baum wird demnach stets in die Richtung von großen, noch nicht erreichten Regionen des Suchraumes expandiert. Da RRT in der ursprünglichen Variante keine Kosten berücksichtigt, entstehen oftmals Pfade, die nicht optimal sind. Im Laufe der Zeit wurden daher wie auch beim A* diverse Abwandlungen entwickelt, die Kosten berücksichtigen und zu besseren Ergebnissen führen [164, 165, 166].

Bei den klassischen Suchbaumverfahren wird stets davon ausgegangen, dass ein Zielzustand (Zielknoten im Suchbaum) existiert. Ohne einen solchen Zielzustand wird aus dem Problem der Pfadplanung in einem Suchbaum das Problem der Erreichbarkeitsanalyse. Dabei geht es darum, alle möglichen erreichbaren Zielknoten von einem gegebenen Startknoten aus zu finden [167]. Daneben existieren auch einige Ansätze zur Trajektorienplanung, die ohne Zielzustand planen. In [168] werden beispielsweise Scharen von Trajektorien entlang eines Navigationspfades oder Straßenmodells generiert und bewertet. Ein anderes Beispiel ist in [169] zu finden, wo ein Trajektorienplaner zur Kollisionsvermeidung vorgestellt wird.

Zusammenfassend ergibt sich jedoch, dass keiner der erwähnten Ansätze direkt für die Extraktion von potentiellen Fahrstreifen aus der DSTMap geeignet ist. Zwar existieren einige Ansätze zur Randbebauungsextraktion aus Belegungskarten, allerdings eignet sich keiner direkt für die Extraktion von Fahrstreifen. Insbesondere die Extraktion mehrerer (u.U. nicht parallel verlaufender) Fahrstreifen ist mit keinem existierenden Ansatz möglich. Anderweitige Verfahren, die ein gewisses Maß an a-priori-Wissen über die Umgebung voraussetzen, können hier nicht verwendet werden, da dies bei der Problemstellung für die vorliegende Arbeit ausgeschlossen wurde. Bei den Ansätzen mit starken Beschränkungen hinsichtlich der verwendeten geometrischen Modelle lassen sich keine

komplexen Fahrstreifenverläufe abbilden. Dies ist jedoch - gerade in städtischen Situationen oder in Baustellen - zwingend erforderlich.

Daher wurde im Rahmen der vorliegenden Arbeit ein neuer Ansatz entwickelt, um Fahrstreifen aus der DSTMap zu extrahieren. Im nachfolgenden Abschnitt wird dieser kurz charakterisiert und der wissenschaftliche Beitrag erläutert.

3.1.2 Eigener Ansatz und wissenschaftlicher Beitrag

In diesem Kapitel wird ein neuer Ansatz zur Extraktion von potentiellen Fahrstreifen aus einer gridbasierten Repräsentation der Umgebung vorgestellt. Als Ausgangsbasis dient die DSTMap, die in Kapitel 2 vorgestellt worden ist. Es sollen unabhängig voneinander potentielle Fahrstreifen in der DSTMap gefunden sowie deren Verlauf bzw. Geometrie bestimmt werden. Dabei wird im folgenden ausgeführte Zusammenhang zwischen der Befahrbarkeit von Arealen und der Existenz von Fahrstreifen ausgenutzt: Jeder Fahrstreifen ist per Definition befahrbar. Dies entspricht der natürlichen Definition von Fahrstreifen aus der Straßenverkehrsordnung (§7 Abs. 1 StVO), wo ein Fahrstreifen als derjenige Teil der Fahrbahn definiert ist, den ein Fahrzeug zum ungehinderten Fahren im Verlauf der Fahrbahn benötigt. Es ist damit zunächst einmal nur ein befahrbarer Bereich, der durch nicht-befahrbare Areale (Hindernisse) oder Markierungen begrenzt ist. Das hier vorgestellte Konzept nutzt diese Kausalität von Befahrbarkeit und Fahrstreifen, um mit Hilfe einer Bewegungsplanung potentielle Fahrstreifen in der DSTMap zu finden. Kann durch eine entsprechende Bewegungsplanung ein fahrbarer Pfad ermittelt werden, lässt sich daraus auf die Existenz eines Fahrstreifens schließen. Dafür wird nun zunächst aus der DSTMap eine Kostenkarte für eine iterative Pfadplanung abgeleitet. Mit der iterativen Pfadplanung werden in der DSTMap kollisionsfrei fahrbare Pfade geplant, die anschließend zu Clustern gruppiert werden. Jeder Cluster bildet damit einen potentiellen Fahrstreifen. Das Ergebnis der Pfadgruppierung wird zusammen mit der DSTMap nun dazu verwendet, für jeden dieser potentiellen Fahrstreifen die dazugehörigen Fahrstreifengrenzen zu finden. In Abb. 3.3 ist der vorgestellte Ansatz schematisch dargestellt.

Die wichtigsten Eigenschaften des Ansatzes sind im Folgenden aufgelistet:

- Der Ansatz baut auf einer iterativer Pfadplanung auf, ist jedoch hinsichtlich des Planungsalgorithmus agnostisch.
- Durch die Pfadplanung wird garantiert, dass alle geometrischen und kinematischen Randbedingungen des automatisiert fahrenden Ego-Fahrzeugs erfüllt werden.

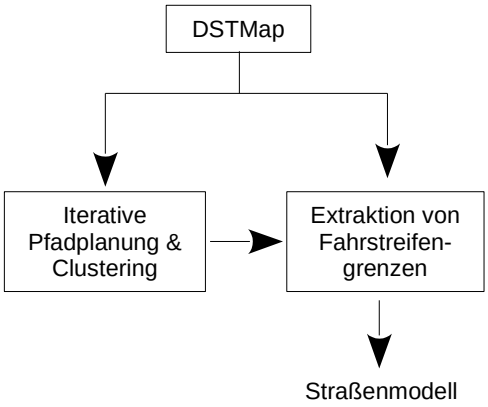


Abbildung 3.3: In der DSTMap werden mit Hilfe einer iterativen Pfadplanung potentielle Fahrstreifen gesucht. Die DSTMap wird zusammen mit den gefundenen Pfaden zur Extraktion von Fahrstreifengrenzen verwendet. Das Ergebnis dient später als Grundlage für die Erzeugung eines konsistenten Straßenmodells.

- Die Extraktion der einzelnen Fahrstreifen ist unabhängig von der zugrunde liegenden Straßen-Topologie. Damit können Fahrstreifen für alle erdenklichen Topologien extrahiert werden.
- Es gibt keine Beschränkungen hinsichtlich des geometrischen Verlaufs der Fahrstreifen.
- Es ist garantiert, dass die Fahrstreifen kollisionsfrei befahrbar sind.
- Es werden keine a-priori-Informationen über die Straßen-Topologie oder die Umgebung benötigt.
- Durch die einzigartige Modellierung der DSTMap können alle für die Pfadplanung relevanten Entitäten wie Hindernisse oder extrinsische Kosten direkt aus der DSTMap berechnet bzw. entnommen werden ohne das bereits in Kap. 2 vorgestellte Modell erweitern zu müssen.
- Soweit bekannt, werden in der vorliegenden Arbeit erstmals 'beliefs' von Hypothesengruppen und die 'plausibility' im Zusammenhang mit der Kollisions- und Kostenberechnung für einen Pfadplaner verwendet.
- Der Ansatz ist in der Lage nicht nur den Ego-Fahrstreifen, sondern sowohl Nachbahrfahrstreifen als auch aufgehende Fahrstreifen und Einmündungen zu erkennen.

Im folgenden Abschnitt wird die Methode der iterativen Pfadplanung im Detail erläutert. Darüber hinaus werden zum besseren Verständnis kurz die Grundlagen der Pfadplanung beschrieben.

3.2 Iterative Pfadplanung

Die Pfadplanung oder verallgemeinert die Bewegungsplanung ist eines der zentralen Probleme der Robotik. Dabei geht es in erster Linie darum, einen kollisionsfreien Pfad von einem definierten Startzustand zu einem oder mehreren Endzuständen zu finden [170]. Präziser formuliert: Es wird eine Sequenz von Konfigurationen von einer gegebenen Startkonfiguration zu einer gegebenen Endkonfiguration gesucht. Dabei ist die Konfiguration eine eindeutige Beschreibung des Zustands eines Systems/Roboters. Sie beinhaltet feste Werte für alle veränderlichen Variablen, die das System oder den Roboter charakterisieren bzw. dessen Zustand beschreiben. Der Raum aller möglichen Konfigurationen wird als Konfigurationsraum C bezeichnet. Die Dimension des Konfigurationsraums ist die Anzahl der n unabhängigen Freiheitsgrade des Systems.

Ein Pfad ist eine kontinuierliche Bahn im Konfigurationsraum C. Die Pfadplanung kann damit mathematisch als das Problem angesehen werden, eine Abbildung

$$\mathbf{p} : [0, 1] \rightarrow C \tag{3.1}$$

zu finden, bei der $\mathbf{p}(0)$ eine vorgegebene Start- und $\mathbf{p}(1)$ eine vorgegebene Zielkonfiguration darstellen. Meist werden bei der Pfadplanung auch zusätzliche Kriterien berücksichtigt, wie bspw. die Kollisionsfreiheit oder die Sicherheit (Abstand zu Hindernissen).

In Bezug auf das automatisierte Fahren wird die Pfadplanung meist in verschiedene Ebenen unterteilt und beginnt üblicherweise mit der Routenplanung. Diese ist mit der Funktionalität eines heutzutage verfügbaren Navigationssystems vergleichbar. Darunter findet sich meist eine sogenannte „high-level" Manöverplanung, die oft auch als Fahrstrategie bezeichnet wird [36, 171], sowie Pfad- und Trajektorienplaner [172].

Bei einer Routenplanung möchte der Fahrer von der aktuellen Fahrzeugposition zu einem beliebigem Ziel. Die Route besteht aus groben Wegpunkten bzw. den Straßen, die zum gewünschten Ziel führen. Im Gegensatz dazu wird bei einer Manöverplanung entschieden, welches Manöver als nächstes durchgeführt werden soll. Typische Manöver sind beispielsweise der Fahrstreifenwechsel auf einen benachbarten Fahrstreifen oder das Abbiegen auf Kreuzungen.

Bei der Pfad- und Trajektorienplanung wird - ausgehend von eine Startkonfiguration, die üblicherweise dem aktuellen Fahrzeugzustand (Position, Geschwindigkeit, etc.) entspricht - ein Pfad oder eine Trajektorie für das bereits ausgewählte Manöver geplant. Meist ist dies eine fix gewählte Zielkonfiguration mit fest vorgegebenem Planungshorizont in Abhängigkeit vom Manöver. Je nach Manöver kann dies beispielsweise eine weiter vorne gelegene Position auf dem aktuellen Fahrstreifen (dem eigenen Fahrstreifen folgend), oder eine festgelegte Position auf einem anderen Fahrstreifen sein (Fahrstreifenwechsel).

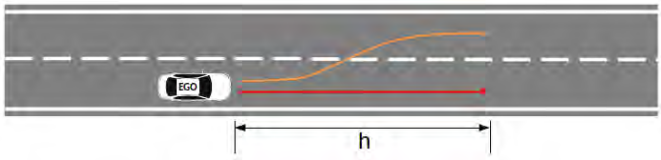


Abbildung 3.4: Pfadplanung für zwei verschiedene Manöver für das Ego-Fahrzeug. Ziel der Pfadplanung ist im ersten Fall ein Mittelpunkt des Ego-Fahrstreifens und im anderen Fall ein Mittelpunkt des Nachbar-Fahrstreifens jeweils unter Berücksichtigung eines festgelegten Planungshorizonts h.

Sowohl die Manöverplanung als auch die darunterliegende Pfadplanung benötigen ein entsprechendes Umfeldmodell, das die nähere Fahrzeugumgebung beschreibt (siehe Kap. 2). Der Verlauf der Fahrstreifen ist dabei von entscheidender Bedeutung. Ein Fahrstreifen ist per Definition befahrbar und insbesondere der Verlauf der Fahrstreifenmittenlinie ist stets ein kollisionsfrei kinematisch fahrbarer Pfad. Andernfalls wäre es auch einem menschlichen Fahrer nicht möglich, mit seinem Fahrzeug in der Mitte des Fahrstreifens zu fahren.

Das Problem lässt sich jedoch auch umkehren: Unter der Voraussetzung einer Umfeldmodellierung, die befahrbare Flächen und deren Begrenzungen korrekt abbildet lassen sich von fahrbaren Pfaden potentielle Fahrstreifen ableiten. Der wesentliche Unterschied zur klassischen Pfadplanung besteht darin, dass bei der Suche nach Fahrstreifen die Zielkonfiguration der Pfadplanung gänzlich unbekannt ist.

Der hier vorgestellte Ansatz zur Extraktion von potentiellen Fahrstreifen aus der DSTMap basiert daher auf einer iterativen Pfadplanung. Er ist in Abb. 3.5 schematisch dargestellt.

Aus der DSTMap aus Kap. 2 wird zunächst eine Kosten- und Hinderniskarte für die Pfadplanung generiert. Die iterative Planung besteht aus zwei Blöcken: Der Pfadplanung und dem Clustering. Die Pfadplanung generiert für eine gegebene Startkonfiguration fahrbare Pfade (Zielpfade), die eine vordefinierte Länge besitzen. Anschließend werden diese Pfade gruppiert und pro Gruppe/Cluster ein Repräsentant ausgewählt. Der Endpunkt jedes Cluster-Repräsentanten dient

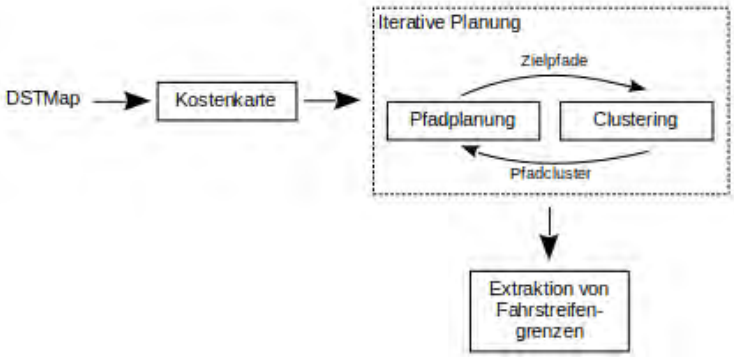


Abbildung 3.5: Schematischer Überblick über den hier vorgestellten Ansatz zur Extraktion potentieller Fahrstreifen aus der DSTMap. Zunächst wird aus der DSTMap eine Kostenkarte für die Kollisions- und Kostenprüfung für eine Pfadplanung erzeugt. In einem iterativen Verfahren werden anschließend Pfade geplant und jeweils zu Clustern zusammengefasst. Anschließend werden pro Cluster Punkte von möglichen Fahrstreifengrenzen aus der DSTMap extrahiert.

als neue Startkonfiguration für die Pfadplanung bei der nachfolgenden Iteration. Nach Abarbeitung einer vordefinierten Anzahl an Iterationen werden die so gefundenen Cluster-Repräsentanten für die Extraktion von Fahrstreifengrenzen verwendet.

Der folgende Abschnitt beschreibt die Kollisionsprüfungen und Kostenberechnung auf Basis der DSTMap für die Pfadplanung. Die eigentliche Pfadplanung und das Clustering wird anschließend in Kap. 3.2.2 erläutert.

3.2.1 Kollisionsprüfung und Kostenberechnung

Die Kollisionsprüfung und Kostenberechnung von Pfaden und Trajektorien ist eine der zentralen Komponenten der Bewegungsplanung in der Robotik. Daher existieren dafür bereits etliche Lösungsansätze (siehe bspw. [170]).

Grundsätzlich kann bei der Kollisionsprüfung zwischen der Kollision mit statischen und der mit dynamischen Objekten unterschieden werden. Bei der Kollisionsprüfung mit statischen Objekten ist das Ergebnis nicht von der Zeit abhängig. Ein Pfad ist entweder kollisionsfrei oder führt zu einer Kollision - unabhängig davon, zu welchem Zeitpunkt das Fahrzeug dem Pfad folgt. Bei dynamischen Objekten ist der Zeitpunkt, zu dem der Pfad abgefahren wird, entscheidend, da die aktuelle Situation nicht mit einer in der Zukunft liegenden übereinstimmt. Durch die Bewegung anderer Objekte können sich potentielle Kollisionsorte jederzeit ändern. Für die Kollisionsprüfung bei dynamischen Objekten ist die zukünftige Position daher viel entscheidender als die aktuelle. In

der Praxis wird hierfür eine Prädiktion der Objekte durchgeführt, bei der die zukünftige Position der Objekte aus den aktuellen Messdaten geschätzt wird. Da die Straßeninfrastruktur jedoch statisch ist, und auch die DSTMap nur die statischen Anteile des Umfeldmodells abbildet, ist eine Prädiktion im vorliegenden Fall nicht notwendig.

Die konkrete Berechnung von Kollisionen und Kosten hängt von der Repräsentation der Umgebung ab. Bei einem grid-basierten Umfeldmodell ist der Rechenaufwand vergleichsweise hoch im Vergleich zu objektbasierten Darstellungen. Beispielsweise ist die Überprüfung einer Kollision zwischen einem geraden Pfad (modelliert als Gerade) und einem Objekt (modelliert als Polygon) deutlich einfacher, als die Überprüfung einer Kollision zwischen einem Pfad, der aus einer Menge von Gridzellen besteht, und einem Objekt, das ebenfalls aus einer Menge an Gridzellen besteht.

Formal betrachtet ist ein Pfad $\mathbf{p} : [0, l] \rightarrow C$ kollisionsfrei, wenn alle seine Punkte $\mathbf{p}$ im Konfigurationsraum kollisionsfrei sind:

$$\forall s \in [0; l] : \mathbf{p}(s) \notin C_{Hindernis} \tag{3.2}$$

Dabei ist $C_{Hindernis}$ die Menge aller Punkte/Konfigurationen q in C, die eine Kollision mit einem Hindernis beinhalten:

$$C_{Hindernis} = \{q \in C | S_q \cap \mathcal{H} \neq \emptyset\} \tag{3.3}$$

Dabei ist S_q das Volumen (Fussabdruck) des Fahrzeugs im Ortsraum bei gegebener Konfiguration q, welches mit irgendeinem Hindernis aus der Menge aller Hindernisse $\mathcal{H}$ überlappt und daher eine Kollision hervorruft.

Für die vorliegende Arbeit wird eine gridbasierte Darstellung der Umgebung und damit auch eine gridbasierte Darstellung von Hindernissen und Kosten verwendet. Die Hindernisse werden als binäres Grid B repräsentiert, bei dem jede Gridzelle entweder ein Hindernis beinhaltet oder hindernisfrei ist. Für die Berechnung von B wird nun auf die DSTMap aus Kap. 2 zurückgegriffen. Dabei wird die 'plausibility', die in der Dempster-Shafer Theorie definiert ist (siehe hierzu Abschn. 2.1.2), als Maß für die Klassifizierung der Zellen verwendet.

Mit dem in Kap. 2 in Gl. 2.19 definierten 'Frame of Discernment' und der allgemeinen Definition der 'plausibility' ergibt sich für die 'plausibility' der Hypothese 'Lane' für eine einzelne Gridzelle

$$Pl(L) = \sum_{B \cap L \neq \emptyset} m(B) \tag{3.4}$$

$$= 1 - Bel(\neg L) \tag{3.5}$$

$$= 1 - Bel(\{M, S, O\}) \tag{3.6}$$

Unterschreitet nun diese 'plausibility' einen festgelegten Grenzwert, stellt diese Gridzelle ein Hindernis dar:

$$B_i = \begin{cases} 1 & \text{wenn } Pl(L) \leq \tau \\ 0 & \text{sonst} \end{cases} \tag{3.7}$$

Dabei steht B_i für die i-te Zelle im Hindernisgrid B. Ein Wert von 1 in einer Gridzelle entspricht einem Hindernis, ein Wert von 0 einer hindernisfreien Zelle.

Die Verwendung der 'plausibility' bringt gleich mehrere Vorteile mit sich:

- Unbekannte Areale werden nicht als Hindernis angesehen.
- Alle Hinweise, die **potentiell** auf die Korrektheit der Hypothese 'Lane' hindeuten, sind in der 'plausibility' berücksichtigt.
- Die 'plausibility' stellt eine obere Grenze für den 'belief' der entsprechenden Hypothese dar. Somit ist die Kollisionsberechnung hier absichtlich optimistisch ausgelegt. So würde bspw. eine 'belief mass' $m(\{M, L\}) = 1$ zu einer 'plausibility' von $Pl(L) = 1$ und damit zu einer als hindernisfrei klassifizierten Zelle führen, da es keinerlei Hinweis darauf gibt, welche der beiden Hypothesen korrekt ist. Es ist lediglich bekannt, dass eine der beiden korrekt ist. Da L in $\{M, L\}$ enthalten ist, trägt $m(\{M, L\})$ stets zur Plausibilität von L bei.
- Eine Berechnung von $Bel(\{M, S, O\})$ muss in jedem Fall erfolgen, da dies für die spätere Extraktion der Fahrstreifengrenzen benötigt wird. Es entsteht daher kaum zusätzlicher Rechenaufwand für die Berechnung der 'plausibility'.

Für die Kollisionsberechnung wird das Fahrzeuges als Box S mit x- und y-Koordinaten und der Ausrichtung θ modelliert. Damit ist ein Zustand $q \in C$ definiert als

$$q = (x, y, \theta)^T \tag{3.8}$$

Für jede beliebige Fahrzeugausrichtung θ kann $C_{Hindernis}$ mit Hilfe der Minkowski-Differenz $\ominus$ von B und S berechnet werden (siehe hierzu auch [170])

$$C^{\theta}_{Hindernis} = B \ominus S_{\theta} = \{b - s | b \in B, s \in S_{\theta}\} \tag{3.9}$$

oder über die Minkowski-Summe ausgedrückt

$$C^{\theta}_{Hindernis} = B \oplus -S_{\theta} = \{b + s | b \in B, s \in -S_{\theta}\} \tag{3.10}$$

Die Minkowski-Summe kann auch mit Hilfe einer binären morphologischen Dilatation berechnet werden, die aus der digitalen Bildverarbeitung bekannt ist. Dies ist in mehrere Hinsicht vorteilhaft, da einerseits die Beschreibung des Umfeldmodells bereits in gridbasierter Form vorliegt und somit als Bild aufgefasst werden kann. Andererseits existieren bereits optimierte Algorithmen für die effiziente Berechnung der morphologischen Verfahren der Bildverarbeitung (siehe bspw. [173]), sodass eine Berechnung der Dilatation in angemessener Rechenzeit gegeben ist. Das Hindernis-Grid wird nun erzeugt, indem das strukturierende Element - also die geometrische Form des Fahrzeugs (Fahrzeugmaske) - mit ihrem Ursprung auf jede Zelle des Grid B, die ein Hindernis enthält, gesetzt wird und die binäre Dilatation vom Grid B mit der Fahrzeugmaske berechnet wird. Bei einer fest vorgegebenen Ausrichtung des Fahrzeugs ergibt sich

$$C^{\theta}_{Hindernis} = \bigcup_{b \in B} S^{\theta}_{B} \tag{3.11}$$

Falls die Fahrzeugmaske rotationsinvariant bzgl. der xy-Ebene ist (Kreis), ist $C^{\theta}_{Hindernis} = C_{Hindernis}$ und damit unabhängig von der Fahrzeugausrichtung. Falls die Maske nicht rotationsinvariant ist, muss $C^{\theta}_{Hindernis}$ für jede mögliche Ausrichtung θ berechnet werden. Damit ergibt sich für die Menge aller Konfigurationen, die eine Kollision hervorrufen

$$C_{Hindernis} = \bigcup_{\theta} C^{\theta}_{Hindernis} \tag{3.12}$$

In der Praxis kann die Fahrzeugausrichtung diskretisiert und für jede mögliche Ausrichtung θ ein $C^{\theta}_{Hindernis}$ berechnet werden, indem die Fahrzeugmaske schräg im Winkel θ auf jede Zelle des Grids gesetzt wird.

Eine effiziente Berechnung der Dilatation kann beispielsweise durch FAMOD oder den van Herk-Gil-Werman-Algorithmus erfolgen (siehe u.a. [174]).

Abb. 3.6 zeigt exemplarisch die Berechnung von $C_{Hindernis}$ für eine rotationsinvariante Fahrzeugmaske (Kreis) auf Basis von einem binären Hindernisgrid mit Hilfe der morphologischen Dilatation.

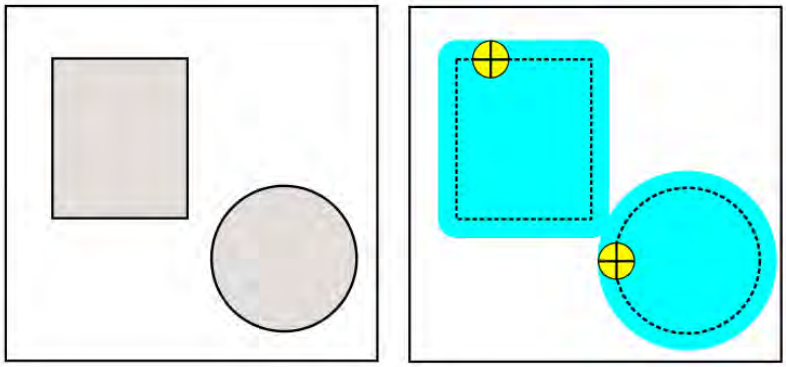


Abbildung 3.6: Beispiel einer morphologischen Dilatation anhand von einem Rechteck und einem Kreis. Das strukturgebende Element ist in diesem Fall ein Kreis (gelber Kreis mit schwarzem Kreuz).

Kostenberechnung

Meist reicht es jedoch nicht aus, nur die Kollisionsfreiheit eines Pfades zu garantieren. Neben der Kollisionsfreiheit sollte ein „guter" Pfad auch gewisse Qualitätsmerkmale besitzen. Diese werden invers in Kosten abgebildet. Je höher die Kosten sind, desto „schlechter" ist der Pfad.

Grundsätzlich kann zwischen *intrinsischen* und *extrinsischen* Kosten unterschieden werden. Die intrinsischen Kosten beziehen sich auf Form und Verlauf des Pfades und hängen von den gewählten Aktionen bei der Expansion des Graphen ab. Bspw. können starke Krümmungen oder Unstetigkeiten in der 1. Ableitung („Knicke" im Pfad) dazu führen, dass ein Pfad höhere Kosten besitzt, als ein „glatter" Pfad. Extrinsische Kosten sind solche Kosten, die sich nicht auf den Pfad, sondern auf die Umgebung beziehen. Dabei repräsentieren sie Kosten, die durch das Überqueren von bestimmten Arealen verursacht werden. Bekanntestes Beispiel hierfür sind Belegungswahrscheinlichkeiten von Belegungskarten. Das Überqueren von Zellen mit hoher Belegungswahrscheinlichkeit verursacht damit bei der Pfadplanung hohe Kosten. Dadurch ist gewährleistet, dass eine Pfadplanung die Überquerung von freien Arealen gegenüber *unbekannten* oder *belegten* bevorzugt. Belegungskarten stellen nur eine von vielen Möglichkeiten dar, wie sich Kosten, die sich aus dem aktuellen Umfeldmodell ergeben, in einer gridbasierten Darstellung als Kostenkarte für eine Pfadplanung nutzen lassen. Der Begriff 'Kostenkarte' ist hierbei als Karte des Konfigurationsraums zu verstehen, der die Kosten für eine bestimmte Konfiguration q des Fahrzeugs beinhaltet.

Andere Formen von Kostenkarten sind beispielsweise Grids mit inversen Distanzen zum nächsten Hindernis (Distanztransformation), bei deren Verwendung eine Pfadplanung stets den Weg mit dem größtmöglichen Abstand zu allen Hindernissen wählt.

Für die Erzeugung einer Kostenkarte aus der DSTMap fließen neben der 'belief mass' für die Hypothese 'Lane' als inverses Maß für die Kosten auch die 'belief mass' für 'unbekannt' und der 'belief' für eine Fahrstreifengrenze, $\{M, S, O\}$, ein. Für jede Zelle i der DSTMap werden die Kosten für die Zelle i der Kostenkarte wie folgt berechnet:

$$G_i = m(\Omega) + bel(\{M, S, O\}) - m(L) \tag{3.13}$$

bzw. normiert auf das Intervall $I = [0; 1]$

$$G_i = \frac{m(\Omega) + bel(\{M, S, O\}) - m(L) + 1}{2} \tag{3.14}$$

Je höher die 'belief mass' für 'Lane' - und damit das Vertrauen in die Korrektheit der Hypothese - ist, desto geringer sind die Kosten. Der 'belief' $bel(\{M, S, O\})$ geht, ebenso wie die 'belief mass' für 'unbekannt', als positive Kosten in die Gesamtkosten ein. Im Gegensatz zur Kollisionsberechnung ist die Berechnung der Kosten konservativ ausgelegt. Ausschließlich die 'belief mass' der Hypothese 'L' bewirkt eine Reduzierung der Kosten. In Tabelle 3.1 sind exemplarisch einige Fälle der Plausibilität und Kosten für verschiedenen Messungen von 'belief masses' dargestellt. Im Vergleich vom ersten zum zweiten Fall lässt sich der Optimismus der Hindernisberechnung erkennen, da der Anteil $m(\{M, L\})$ die Plausibilität erhöht, die Kosten jedoch nicht komplett verschwinden. Weiterhin ist festzustellen, dass durch die Verschiebung der 'belief mass' von $m(S)$ nach $m(\{M, L\})$ die Kosten erwartungsgemäß sinken, da die Masse $m(S) = 0.4$ (erster Fall) eindeutig für eine Fahrstreifengrenze spricht und sich positiv auf die Kosten auswirkt, wohingegen bei $m(\{M, L\})) = 0.4$ (zweiter Fall) nicht klar ist, ob es sich um 'Lane' oder 'Marking' - also um eine Fahrstreifengrenze - handelt.

Im dritten Fall ergibt sich aus der Messung, dass die Zelle entweder zu 'Marking' oder zu 'Lane' gehört, eine weitere Unterscheidung ist jedoch nicht möglich. Dies wird nicht als Hindernis angesehen, da es sich **potentiell** um 'Lane' handeln könnte. Allerdings sind die Kosten identisch zu Fall 1, da auch hier nicht näher zwischen Fahrstreifen und Fahrstreifengrenze unterschieden werden kann. Grundsätzlich werden damit mehrdeutige Zellen bei einer Pfadplanung berücksichtigt, da sie nicht a-priori als Hindernis angesehen werden. Die hohen Kosten dieser Zellen sorgen bei der Planung jedoch dafür, dass der Planer Areale bevorzugt, die eindeutig als 'Lane' gemessen worden sind.

$m(L)$	$m(S)$	$m(\{M,L\})$	$m(\Omega)$	$Pl(L)$	G_i
0.4	0.4	0	0.2	0.6	0.6
0.4	0	0.4	0.2	1	0.4
0	0	0.8	0.2	1	0.6

Tabelle 3.1: Exemplarische Fälle für Plausibilität $Pl(L)$ und Kosten G_i für unterschiedliche Messungen von 'belief masses'.

Mit Gl. 3.14 werden Lane-Areale gegenüber unbekannten Arealen bevorzugt. Zellen mit einer hohen 'belief mass' für 'Lane' weisen demnach geringere Kosten auf als Zellen mit der gleichen 'belief mass' für 'unbekannt'.

Ähnlich wie bei der Kollisionsprüfung müssen für die Berechnung der Kosten für eine gegebene Konfiguration q des Fahrzeugs alle Zellen der Kostenkarte, die das Fahrzeug abdeckt, ermittelt werden und ein einzelner Kostenwert für diese Konfiguration definiert werden. Es gibt keinen allgemeingültigen Weg, wie aus den Kosten der einzelnen relevanten Zellen, die für eine bestimmte Konfiguration q ermittelt wurden, die Gesamtkosten für q berechnet werden. Analog zu [174] wird im vorliegenden Fall der größte Kostenwert (das Maximum) der relevanten Zellen verwendet.

Das Verfahren zur Berechnung der relevanten Zellen ist dem der Kollisionsprüfung sehr ähnlich. Die Kostenkarte G wird dabei als Graustufen-Bild aufgefasst, da o.B.d.A. die Kosten als größer oder gleich 0 angesehen werden können. Weil die Kosten im Gegensatz zur Kollisionskarte nicht binär sind, wird für die Kostenberechnung wird statt der binären morphologischen Dilatation eine morphologische Graustufen-Dilatation verwendet.

Analog zu $C_{Hindernis}$ lässt sich damit für die Kosten auch schreiben

$$C^{\theta}_{Kosten} = G \oplus S_q \tag{3.15}$$

wobei S_q das strukturgebende Element - die Fahrzeugmaske mit Zustand q - ist. Diese enthält ausschließlich Nullen, da die Dilatation nur dazu dient, die Zellen zu ermitteln, die unter S liegen. G ist die Kostenkarte - aufgefasst als Graustufenbild.

Schlussendlich ergeben sich die vollständigen Pfadkosten durch Integration der Kosten aus der Kostenkarte entlang des Pfades:

$$\int_0^l C_{Kosten}(\tau(s))ds \tag{3.16}$$

3.2.2 Pfadplanung und Clustering

In der vorliegenden Arbeit wurde eine neue Pfadplanung entwickelt, die eine Weiterentwicklung des Pfadplanungsansatzes aus [175] darstellt. Dieser ist in der Lage, Pfade ohne vorgegebene Zielkonfiguration zu planen. Statt des Erreichens der vorher festgelegten Zielkonfiguration als Zielkriterium wird ein alternatives Kriterium zur Bewertung des Pfades - hier die Pfadlänge - definiert.

Formal beschreiben lässt sich diese Problem folgendermaßen:

Sei $q_S \in C$ im Konfigurationsraum C ein Punkt, der die Startkonfiguration des Fahrzeuges darstellt, und l_G die gewünschte Pfadlänge im Ortsraum, $\mathbf{p} : [0, l] \rightarrow C_{free}$ ein kollisionsfreier Pfad mit Ortsraum-Länge l.

Das Ziel ist es, eine Menge $\mathcal{P}$ von Pfaden $\mathbf{p}$ zu finden, sodass

$$\mathcal{P} = \{\mathbf{p} | \mathbf{p}(0) = q_S \wedge l = l_G\} \tag{3.17}$$

und

$$\mathbf{p}(l_G) = q_S + \int_0^{l_G} f(\mathbf{p}(l), u(l))dl \tag{3.18}$$

mit einem entsprechenden Bewegungsmodell f, das die Bewegung des Fahrzeugs modelliert, die aus der Wahl der Aktion u hervorgehen soll.

$q = (x, y, \theta)^T$ ist ein Punkt in C und beschreibt den Zustand des Fahrzeugs auf dem Pfad als 2D-Position mit einer Ausrichtung. $\mathbf{p}(l)$ repräsentiert einen Punkt eines Pfades und ist damit ein Punkt in C.

Wie schon eingangs beschrieben, kann eine gridbasierte Kostenkarte für einen graphenbasierten Pfadplaner verwendet werden. Hierbei werden die Zellen als Knoten und die Verbindungen zwischen Zellen als Kanten modelliert. Beginnend von Startpunkt aus wird nun ein Graph aufgebaut, dessen Knoten neben der Position auch die Kosten (Pfadkosten vom Startpunkt bis zum Knoten) enthalten. Die Expansion des Graphen ergibt sich aus den Aktionen u.

Für die Optimierung der Rechenzeit müssen für die Expansion des Graphen Randbedingungen formuliert werden. Andernfalls ist die Expansion nicht in Echtzeit durchführbar. Darüber hinaus unterliegen die physikalisch möglichen Bewegungen eines Fahrzeugs gewissen Grenzen, sodass nicht jede beliebige Aktion u möglich ist. Die Randbedingungen bei der Definition der Aktionen u beschränken sich hier auf die Expansionsrichtung (Form des Bewegungsprimitivs)

und die Länge des Primitivs. In der Praxis ist die maximal fahrbare (oder von einem Insassen tolerierte) Querbeschleunigung begrenzt. Bei einer vorgegebenen Geschwindigkeit ergibt dies eine Obergrenze der Krümmung der Bewegungsprimitive. Auch die Pfadlänge ist von einer vorgegebenen Geschwindigkeit abhängig und wird daher - wie auch bei einer Trajektorienplanung häufig zu finden - als Zeitintervall angegeben.

Die Bewegungsprimitive werden mit Hilfe eines Bewegungsmodells berechnet, dessen Aufgabe es ist, aus einer Konfiguration q alle realisierbaren nachfolgenden Konfigurationen q' zu generieren. Als Modell wird hier das einspurige Fahrradmodell verwendet, da es effizient ist, und dennoch die nicht-holonomischen Bedingungen erfüllt. Die möglichen Aktionen u leiten sich einfach aus den Lenkwinkeln α der Räder ab. In Abb. 3.7 ist das Fahrradmodell schematisch dargestellt. Die Herleitung ist im Folgenden beschrieben.

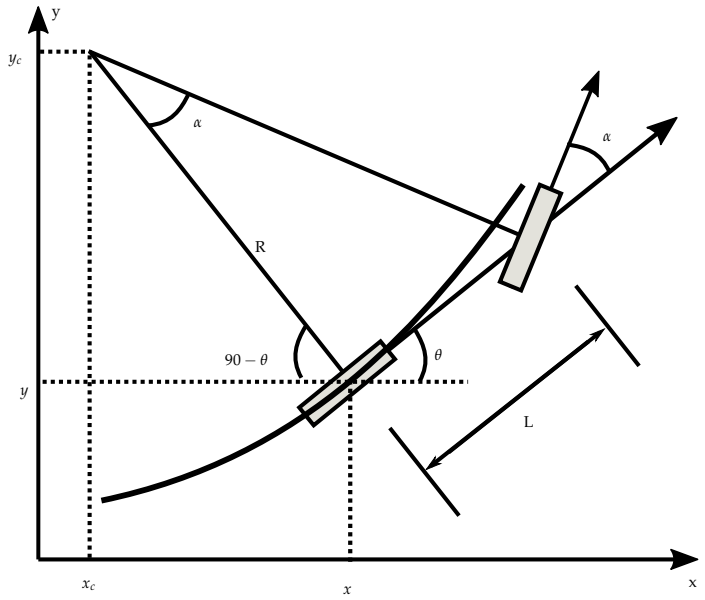


Abbildung 3.7: Schematische Zeichnung des einspurigen Fahrradmodells. Die grauen Rechtecke stellen Vorder- und Hinterrad dar.

Die Winkelgeschwindigkeit $\dot{\Theta}$ des Fahrzeugs ist

$$\dot{\Theta} = \frac{v}{R} \tag{3.19}$$

wobei R der Radius der Kreisbahn ist, auf der sich das Fahrzeug bewegt. Außerdem gilt

$$\tan \alpha = \frac{L}{R} \tag{3.20}$$

Die Winkeländerung bei einer Bewegung des Fahrzeugs ist daher

$$\beta = \int \mathrm{d}t \, \dot{\Theta} = \frac{d}{L} \tan \alpha = \frac{d}{R} \tag{3.21}$$

wobei die Integration über ein Zeitintervall erfolgt, bei der das Hinterrad die Distanz d zurücklegt.

Außerdem ist aus Abb. 3.7 ist ersichtlich, dass

$$\begin{aligned} x_c &= x - R \cdot \cos(90 - \Theta) = x - R \cdot \sin \Theta \\ y_c &= y + R \cdot \sin(90 - \Theta) = y + R \cdot \cos \Theta \end{aligned} \tag{3.22}$$

Da bei einer Bewegung des Fahrzeugs die neue Position ebenfalls auf der Kreisbahn liegt, ergibt sich

$$\begin{aligned} x' &= x_c + R \cdot \sin(\Theta + \beta) \\ y' &= y_c - R \cdot \cos(\Theta + \beta) \end{aligned} \tag{3.23}$$

Für eine neue Konfiguration q' des Fahrzeugs aus einer vorherigen Konfiguration q ergibt sich demnach

$$\begin{aligned} q' &= f(\mathbf{p}(l), u(l)) \\ &= f(q, \alpha) \\ &= \begin{cases} ((x + d\cos\Theta), (y + d\sin\Theta), \Theta)^T & \alpha = 0 \\ ((x - R\sin\Theta + R\sin(\Theta + \beta)), (y + R\cos\Theta - R\cos(\Theta + \beta)), \\ (\Theta + \beta))^T & \text{sonst} \end{cases} \end{aligned}$$

wobei α der Lenkwinkel der Räder, L der Achsabstand und d die zurückgelegte Distanz ist.

Für die konkrete Wahl der Aktionen u wird ein maximal möglicher Lenkwinkel α_{max} definiert und die Menge U von Aktionen definiert:

$$U = \left\{ -\alpha_{max}, -\frac{n-1}{n}\alpha_{max}, -\frac{n-2}{n}\alpha_{max}, \ldots, 0, \frac{1}{n}\alpha_{max}, \frac{2}{n}\alpha_{max}, \ldots, \alpha_{max} \right\} \quad (3.24)$$

wobei $n \in \mathbb{N}$ und damit die Anzahl der gewünschten Expansionsrichtungen pro Knoten auf $2n-1$ festgelegt ist.

Es wird nun ein Graph mit dem Ursprung q_s - der Initialposition des Pfades - erzeugt. Mit Hilfe des Bewegungsmodells f wird der Graph expandiert und für jede neue Kante werden die Kosten und die Kollisionsmöglichkeit ermittelt. In der Regel ist es jedoch aufgrund der hierzu benötigten Rechenleistung nicht möglich, den Graphen **vollständig** bis zur gewünschten Tiefe (Erreichen des Zielzustandes) zu expandieren. Daher haben sich im Laufe der Zeit verschiedene alternative Verfahren zur geschickten Expansion des Graphen etabliert.

In der vorliegenden Arbeit wird die Methode aus [175] für die Expansion des Graphen verwendet. Dabei werden die beiden bekanntesten Planungsalgorithmen A* [176] und RRT [163] kombiniert. Der A*-Algorithmus ist durch die Benutzung einer Kosten-Heuristik sehr effizient und bei der Expansion zielgerichtet.

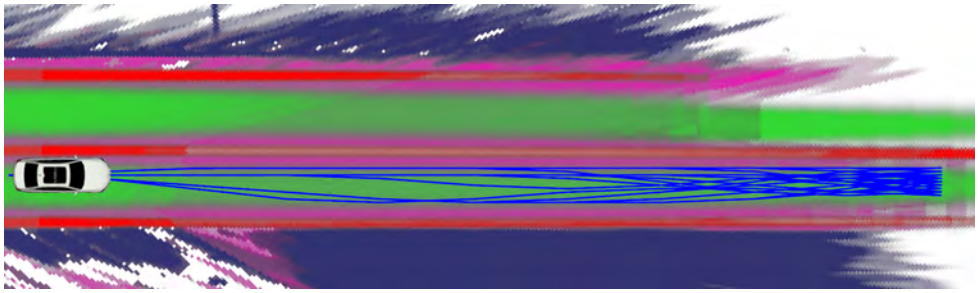
Darüber hinaus ist er

- vollständig: Sofern eine Lösung existiert, wird sie gefunden. Falls keine Lösung existiert, wird dies erkannt.
- optimal: Die beste Lösung (mit den geringsten Kosten) wird gefunden.
- optimal effizient: Es wird nur die minimal erforderliche Anzahl an Knoten expandiert.

Im Gegensatz dazu wird beim RRT-Algorithmus keine auf das Ziel gerichtete Expansion durchgeführt, sondern die Expansion stets in Richtung großer Voronoi-Regionen vorgenommen. Der Suchbaum wächst daher stets in Richtung der großen, unbekannten (noch nicht expandierten) Areale.

Für die kombinierte Planung mit A* und RRT wird zunächst A* verwendet, um den kostengünstigsten Pfad zu finden. Sofern dieser existiert, wird die Expansion des Suchbaumes mit Hilfe des RRT fortgesetzt. Dabei wird als Heuristik für A* nicht wie üblich die Entfernung des aktuellen Knotens zum Zielzustand benutzt, da dieser ja nicht bekannt ist. Stattdessen wird die Differenz von der a-priori definierten Ziel-Pfadlänge l_G und der aktuell erreichten Pfadlänge als Heuristik verwendet. Neben dem verwendeten Pfadplanungsansatz basierend

auf der Kombination von A* und RRT können natürlich auch andere Pfadplaner verwendet werden, sofern sie nicht die Festlegung eines bestimmten Zielzustands voraussetzen. Abbildung 3.8 zeigt das Ergebnis der Pfadplanung auf der DSTMap.



(a)

(b)

Abbildung 3.8: (a) Die DSTMap, auf deren Basis eine Pfadplanung durchgeführt wurde. Die gefundenen Zielpfade sind blau dargestellt. Für die Planung wurde eine vordefinierte Pfadlänge von 50m verwendet. (b) Kamerabild der Situation.

Clustering

Durch die Expansion des Suchbaumes entstehen viele Pfade, von denen sich einige ähnlich sind. Diese müssen zusammengefasst werden, damit die Gesamtanzahl der gefundenen Pfade reduziert wird. Andernfalls könnten die darauf aufbauenden Schritte wie die iterative Planung und die Extraktion von Fahrstreifengrenzen nicht in angemessener Rechenzeit durchgeführt werden.

Die einfachste Methode zur Reduzierung der Anzahl der gefundenen Pfade ist die Beschränkung auf den Pfad mit den geringsten Kosten. Diese sehr simple

Methode hat allerdings den gravierenden Nachteil, dass das Erkennen von Abzweigungen unmöglich ist. Es wird stets nur die Abzweigung weiterverarbeitet, in der der gewählte Pfad liegt.

Eine andere Möglichkeit ergibt sich bei der Betrachtung der geometrischen Ähnlichkeit von Pfaden. Dabei wird von jedem Knotenpunkt eines Pfades die Distanz zum anderen Pfad ermittelt. Liegt die Distanz jeweils unterhalb eines Schwellwertes, werden die Pfade als *ähnlich* angesehen. Anschließend können die Pfade mit Hilfe von entsprechenden Clustering-Algorithmen (bspw. agglomeratives hierarchisches Clustering) gruppiert werden (siehe z.B. [175]). Der Nachteil dieser Methode wird schnell deutlich, wenn zwei Pfade dicht beieinander liegen, aber dennoch von einem Areal mit hohen Kosten oder einem Hindernis getrennt sind. Ein Clustering basierend auf der geometrischen Ähnlichkeit von Pfaden würde diese Pfade fälschlicherweise zusammenfassen.

Daher wird in der vorliegenden Arbeit das Clustering aus [177] verwendet. Dieses Clustering untersucht die Homotopie der Pfade. Generell gelten zwei Pfade als homotop, wenn der eine in den anderen deformiert werden kann ohne dabei ein Hindernis zu schneiden. Allerdings setzt diese Definition bei den Pfaden identische Start- und Zielkonfigurationen voraus. Da die Zielkonfigurationen bei der kombinierten Pfadplanung mit A* und RRT nicht identisch sind, wird eine Abwandlung des Homotopie-Clusterings verwendet. Aus den beiden Pfaden wird zunächst ein Polygon gebildet, indem die beiden Pfadenden durch eine Gerade miteinander verbunden werden. Anschließend wird untersucht, ob sich in dem Polygon Bereiche mit hohen Kosten oder einem Hindernis befinden. Falls dies der Fall ist, gehören die Pfade zu verschiedenen Clustern. Ist kein Areal mit hohen Kosten und auch kein Hindernis im Polygon, gehören die Pfade zu demselben Cluster.

Am Ende besteht jeder Cluster aus einer endlichen Anzahl an gefundenen Pfaden. Zusätzlich wird pro Cluster derjenige Pfad als repräsentativer Pfad für das Cluster ausgewählt, der die niedrigsten Kosten besitzt.

3.2.3 Iterative und parallele Planung

Die im Vorangegangenen beschriebene Pfadplanung auf Basis von A* und RRT benutzt als Zielkriterium nicht das Erreichen des Zielzustandes, sondern die stetige Expansion des Suchbaumes bis der Pfad eine vorher festgelegte Länge erreicht hat. Damit ist es zwar möglich, die Pfadplanung als Explorationsverfahren zu nutzen, allerdings bringt die fixe Pfadlänge diverse Nachteile mit sich. Eine sehr hohe Pfadlänge führt unter Umständen dazu, dass keine Pfade gefunden werden, weil die Mindestlänge der Pfade bei der Expansion/Planung nicht erreicht werden konnte. Häufige Ursachen hierfür sind die beschränkte

Sichtweite der Sensoren sowie die mit der Entfernung zunehmenden Fehldetektionen/Unsicherheiten in den Sensor-Messdaten. Für beide Fehlerfälle ist es nicht möglich, einen fixen Planungshorizont zu definieren, in dem die Planung unter Garantie funktioniert, da Sichtweite, Unsicherheiten und Fehlmessungen stets von der jeweiligen Situation abhängen und im Vorfeld meist nicht bekannt sind. Darüber hinaus wird bei einer langen Pfadlänge viel Rechenzeit benötigt. Insbesondere der RRT-Algorithmus wird sehr langsam, da die maximale Anzahl der Expansions-Iteration sehr hoch sein muss, damit überhaupt Pfade mit entsprechender Länge gefunden werden können.

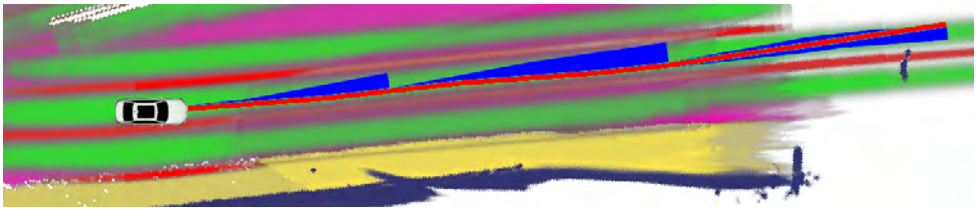
Auf der anderen Seite sorgt eine sehr kurze Pfadlänge dafür, dass sehr viele Pfade gefunden werden. Das Ergebnis ist ein stark verästelter Suchbaum mit sehr kurzen Pfaden. In diesem Fall ist zwar die Rechenzeit der Pfadplanung im Rahmen des Akzeptablen. Allerdings wird die benötigte Zeit bei der Verarbeitung der Pfade in allen nachfolgenden Schritte enorm ansteigen. Darüber hinaus entstehen bei kurzer Pfadlänge viele Zielpfade, die als solche nicht nutzbar sind.

Um diese Nachteile zu umgehen und bei möglichst kurzer Rechenzeit möglichst lange Pfade zu erhalten, wird hier der iterativer Planungsansatz verwendet, der im Rahmen dieser Arbeit entstanden ist [3] und aus den folgenden Schritten besteht:

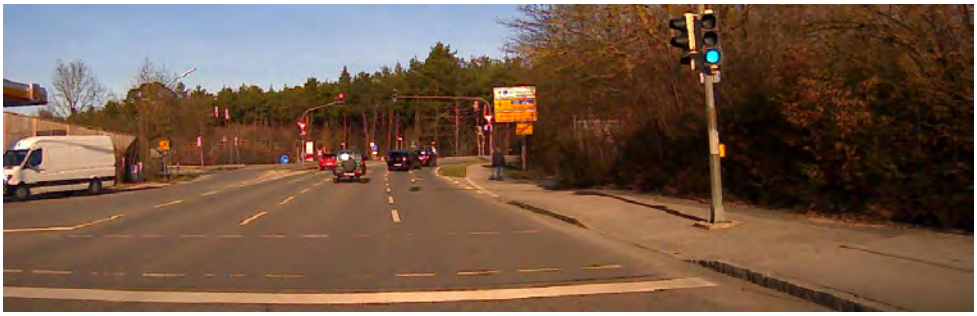
1. Wahl eines geeigneten Startzustands. Der einfachste Fall ist die Verwendung der aktuellen Position und Ausrichtung des Ego-Fahrzeugs als Startzustand.
2. Ausgehend vom Startzustand erfolgt eine Pfadplanung mit A* & RRT mit fixer Pfadlänge l_G.
3. Die gefundenen Zielpfade werden gruppiert mit dem bereits beschriebenen Cluster-Verfahren. Pro Cluster wird derjenige Pfad mit den geringsten Kosten als für diesen Cluster repräsentativer Pfad definiert.
4. (2) und (3) wird für eine feste Anzahl an Iterationen n wiederholt. Dabei werden die Endzustände der repräsentativen Pfade als neue Startzustände für die Pfadplanung verwendet.
5. Zusammensetzen der repräsentativen Pfade zu einer Baumstruktur.

Durch diese iterative Planung kann die vorgegebene Pfadlänge entsprechend kurz gewählt werden; dennoch ergeben sich durch die iterative Planung lange Pfade. Das Clustering wirkt dabei wie eine Art „Filter-Effekt":
Zunächst erfolgt eine Expansion mit relativ kurzer Pfadlänge. Somit entstehen viele Zielpfade. Diese werden anschließend zusammengefasst und durch einen einzigen (den repräsentativen) ersetzt. Ausgehend von diesem wird eine erneute Expansion gestartet.

Abb. 3.9 zeigt das Ergebnis einer solchen iterativen Planung.



(a)



(b)

Abbildung 3.9: (a) DSTMap, auf deren Basis eine Pfadplanung durchgeführt wurde. Die gefundenen Zielpfade sind blau dargestellt, die Clusterpfade rot. Für die Planung wurde eine vordefinierte Pfadlänge von 20m verwendet und 3 Iterationen geplant. (b) Kamerabild der Situation. Das Fahrzeug führt einen Fahrstreifenwechsel durch, weswegen die Ausrichtung nicht parallel zum Fahrstreifen ist.

Dieser neuartige iterative Pfadplanungsansatz ist in der Lage, Abzweigungen, aufgehende Fahrstreifen, sowie Einmündungen zu erkennen - im Rahmen dessen, was der Inhalt des Grids zulässt. Im Vergleich zu einer direkten Planung von Pfaden mit der gleichen Länge, ist der iterative Ansatz **deutlich** effizienter, da jeweils nur die Enden der repräsentativen Pfade neu expandiert werden.

Parallele Planung

Bisher wurde davon ausgegangen, dass die aktuelle Position und Ausrichtung des Ego-Fahrzeugs als Startzustand für den Pfadplaner verwendet wird. Grundsätzlich ist dies eine valide und einfache Methode, den Startzustand des Pfadplaners festzulegen. Es setzt lediglich voraus, dass sich das Fahrzeug auf einem gültigen Fahrstreifen befindet.

Damit kann der Pfadplaner allerdings nur Pfade finden, die auf dem Ego-Fahrstreifen liegen. Logischerweise kann damit letztendlich bei der Ableitung der Fahrstreifen aus den Pfaden nur der eigene Fahrstreifen gefunden werden. Es ist nicht möglich durch diese Planung potentielle Nachbarfahrstreifen zu finden.

Um dennoch Nachbarfahrstreifen extrahieren zu können, wird im Folgenden eine Erweiterung vorgestellt, die eine parallele Planung für die Extraktion von mehreren Fahrstreifen durchführt.

Die Idee dabei ist, dass für diejenigen Fahrstreifen, die gefunden werden sollen, ein eigener iterativer Pfadplaner aufgesetzt und verwendet wird. Die Schwierigkeit besteht darin, dass im Vorfeld weder bekannt ist, wie viele Fahrstreifen es tatsächlich gibt, noch wo diese liegen. Dementsprechend sind weder die Anzahl der benötigten Planer noch die Startzustände der Planer bekannt.

Die Startzustände der Planer werden nun auf Höhe der Ego-Position orthogonal zur aktuellen Fahrzeugausrichtung gesucht. Dabei werden auf dieser Orthogonalen im Grid zunächst alle lokalen Maxima des 'belief' einer Fahrstreifengrenze, $Bel(\{M, O, S\})$, gesucht. Befinden sich zwischen 2 Maxima hinreichend viele Zellen mit einer 'belief mass' für 'Lane' $m(L) > 0$, so wird die Mitte der beiden Maxima als Startposition für den Pfadplaner verwendet. Für die Ausrichtung des Startpunktes wird die Ausrichtung des Ego-Fahrzeugs übernommen. Auf diese Weise lassen sich beliebig viele Planer mit Startzuständen initialisieren. Da für die Fahraufgabe neben dem Ego-Fahrstreifen zunächst nur die beiden direkt benachbarten Fahrstreifen von relevanter Bedeutung sind, kann man sich auf diese beschränken und mit 3 Pfadplanungsinstanzen planen. In Abhängigkeit des Einsatzszenarios und der verfügbaren Rechenleistung können aber auch weitere Instanzen geplant werden.

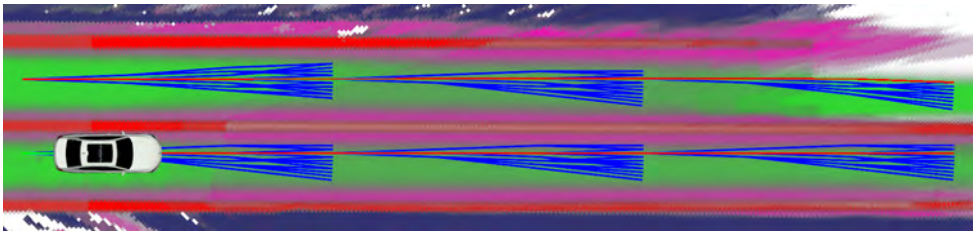
Abb. 3.10 zeigt das Ergebnis einer Pfadplanung mit zwei Planern, die parallel geplant haben und sowohl im Ego-Fahrstreifen als auch im linken Nachbarfahrstreifen Zielpfade gefunden haben.

Aufgrund des Zufälligkeitsprinzips in der RRT-Pfadplanung eignen sich die Clusterpfade nicht, um direkt eine akkurate Fahrstreifenmitte abzuleiten. Die Clusterpfade sind lediglich als Hypothesen für die Fahrstreifenmitte anzusehen.

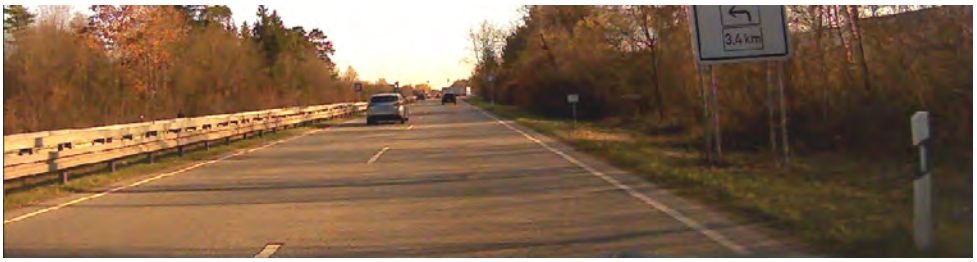
Im folgenden Abschnitt wird zunächst erläutert, wie mit Hilfe der Pfade die Fahrstreifengrenzen aus der DSTMap extrahiert werden. Im weiteren Verlauf können dann aus den Pfaden in Kombination mit den Fahrstreifengrenzen die vollständigen Fahrstreifen gebildet werden.

3.3 Extraktion von Fahrstreifengrenzen

Mit Hilfe der DSTMap aus Kap. 2 können alle Zellen bestimmt werden, die potentiell eine Fahrstreifengrenze darstellen. Mit dem in Gl. 2.19 definierten 'frame of discernment' werden nun alle Zellen als potentielle Fahrstreifengrenze angesehen, die in der DSTMap als *M (Marking)*, *S: Sidewalk* oder *O: Obstacle*



(a)



(b)

Abbildung 3.10: (a) Ergebnis der Pfadplanung mit der DSTMap auf einer autobahnähnlichen Straße. Die gefundenen Zielpfade sind blau dargestellt, die Clusterpfade rot. Für die Planung wurde eine vordefinierte Pfadlänge von 15m verwendet und 3 Iterationen geplant. (b) Kamerabild der Situation.

klassifiziert wurden. Dies kann in der Dempster-Shafer-Theorie über den 'belief', $Bel\{M, S, O\}$, ausgedrückt werden. Im nachfolgenden wird daher jede Zelle der DSTMap als Kandidat für einen Punkt einer Fahrstreifengrenze angesehen, wenn

$$Bel\{M, S, O\} > \tau_{bnd} \tag{3.25}$$

erfüllt, also der 'belief' größer als ein vordefinierter Schwellwert ist. Es werden nun entlang der Clusterpfade jeweils die linken und rechten Fahrstreifengrenzpunkte in der DSTMap gesucht. Dafür wird zunächst für alle Zellen, bei denen Gl. 3.25 erfüllt ist, die Lage (links/rechts) der Zelle relativ zum Clusterpfad ermittelt.

Bei einem gegebenem Pfad p mit den Knoten $\mathcal{K} = \{n_1, \ldots, n_n\}$ mit $n_i \in \mathcal{R}^2$ und Fahrstreifengrenz-Zellen $\mathcal{B} = \{b_1, \ldots, b_n\}, b_i \in \mathcal{R}^2$ kann mit Hilfe von

$$det(n_i - n_{i-1}, b - n_i) \; \forall \; b \in \mathcal{B} \tag{3.26}$$

die relative Lage der Fahrstreifengrenz-Zelle b bestimmt werden. Bei Verwendung des Ego-Koordinatensystems, dessen Ursprung stets in der aktuellen Fahrzeugposition liegt, die x-Achse nach vorne und die y-Achse nach links zeigt, ergibt sich folgende Fallunterscheidung:

$det(\ldots) < 0 \Rightarrow b$ liegt auf **rechten** Seite des Pfades

$det(\ldots) > 0 \Rightarrow b$ liegt auf **linken** Seite des Pfades

Durch die Verwendung eines 'connected component labeling' (CCL) lässt sich der Rechts-Links-Test beschleunigen, da der Test nur einmal pro 'Component'/ Label ausgeführt werden muss, und nicht für jede Zelle einzeln.

Nachdem nun die relative Lage der Zellen bekannt ist, kann die Suche der Fahrstreifengrenzen erfolgen, indem pro Pfadknoten jeweils für die linke und rechte Seite diejenige Fahrstreifengrenz-Zelle gesucht wird, die am nächsten am Pfadknoten liegt. Der entsprechende Algorithmus hierzu ist in Alg. 1 beschrieben.

Algorithm 1 Finde Fahrstreifengrenzpunkte für jeden Pfadknoten

```
1: procedure FindeFahrstreifengrenzpunkte(𝒦, C, ℬ)   ▷ Pfadknoten 𝒦, Cluster C aus CCL, ℬ Fahrstreifengrenz-Zellen
2:     for each n ∈ 𝒦 do
3:         min_distance_right ← ∞
4:         min_distance_left ← ∞
5:         for each c ∈ C do
6:             for each m ∈ c do                      ▷ m: Gridzelle in c
7:                 dist ← euclidean_distance(n, m)
8:                 if c.is_on_right_side then
9:                     if dist < min_distance_right then
10:                        min_distance_right ← dist
11:                        min_dist_right_cell ← idx(m)
12:                    end if
13:                else
14:                    if dist < min_distance_left then
15:                        min_distance_left ← dist
16:                        min_dist_left_cell ← idx(m)
17:                    end if
18:                end if
19:            end for
20:        end for
21:        n.right_boundary_point = min_dist_right_cell_idx
22:        n.left_boundary_point = min_dist_left_cell_idx
23:    end for
24: end procedure
```

Die Fahrstreifengrenzpunkte werden jedoch nur bis zum Ende des Pfades extrahiert, auch wenn die DSTMap über den Pfad hinaus Grenz-Zellen beinhaltet. Da der weitere Verlauf des Pfades unbekannt ist, kann für diese Zellen nicht festgestellt werden, ob sie links oder rechts vom Pfad liegen würden.

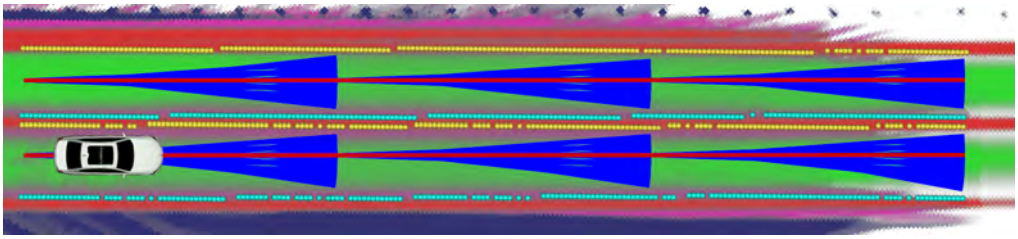
3.4 Ergebnisse

Der folgende Abschnitt zeigt einige qualitative Ergebnisse der Fahrstreifenextraktion mit Hilfe der Pfadplanung von verschiedenen Verkehrssituationen. Für die Darstellung der DSTMap in diesem Abschnitt sowie im anschließenden Kapitel die folgende Farbkodierung verwendet:

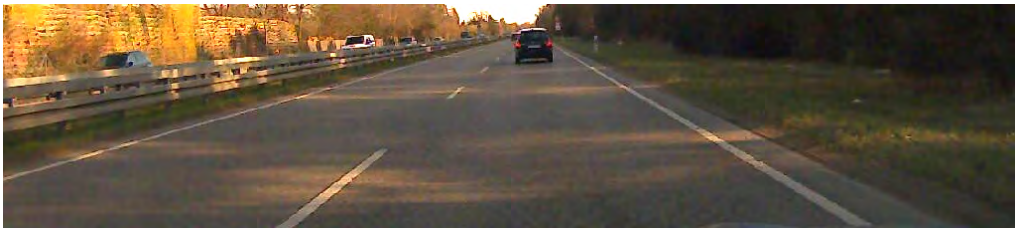
Abbildung 3.11: Farbkodierung der Hypothesen in der DSTMap.

Verkehrssituation 1: Bundesstraße mit parallel verlaufenden Fahrstreifen

Die erste Situation zeigt eine typische Verkehrssituation auf einer Bundesstraße mit 2 parallel verlaufenden Fahrstreifen. In Abb. 3.12a ist die DSTMap mit dem Ergebnis der in diesem Kapitel vorgestellten Verfahren abgebildet. Es wurden für zwei Pfadplaner jeweils 3 Iterationen geplant und die Zielpfade in Cluster gruppiert. Anschließend wurden für jeden Cluster-Pfad jeweils mögliche linke und rechte Fahrstreifengrenzen als Punkte extrahiert. Das Ergebnis zeigt eine fast durchgängige Extraktion der Punkte der Fahrstreifengrenzen - sowohl für den Ego-Fahrstreifen als auch für den Nachbarfahrstreifen.



(a)



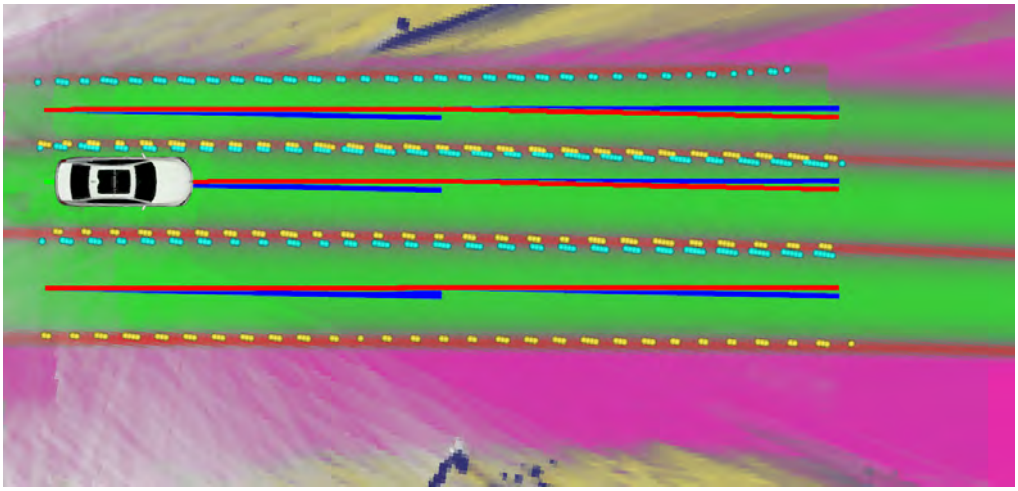
(b)

Abbildung 3.12: (a) DSTMap mit dem Ergebnis der Pfadplanung mit einer Pfadlänge von 15m und 3 Iterationen. Die gefundenen Zielpfade sind blau und die Cluster-Pfade rot dargestellt. Die extrahierten Fahrstreifengrenzpunkte sind jeweils als gelbe (linke Grenze) und türkise (rechte Grenze) Punkte dargestellt. (b) Kamerabild der Situation.

Verkehrssituation 2: Zufahrt auf eine große innerstädtische Kreuzung

Die zweite Situation zeigt die Zufahrt auf eine große innerstädtische Kreuzung mit mehreren parallel verlaufenden Fahrstreifen.

In Abb. 3.13a ist die DSTMap mit dem Ergebnis der in diesem Kapitel vorgestellten Verfahren abgebildet. Es wurden für drei Pfadplaner jeweils 2 Iterationen geplant. Die Extraktion aller Fahrstreifengrenzen ist gut erkennbar.



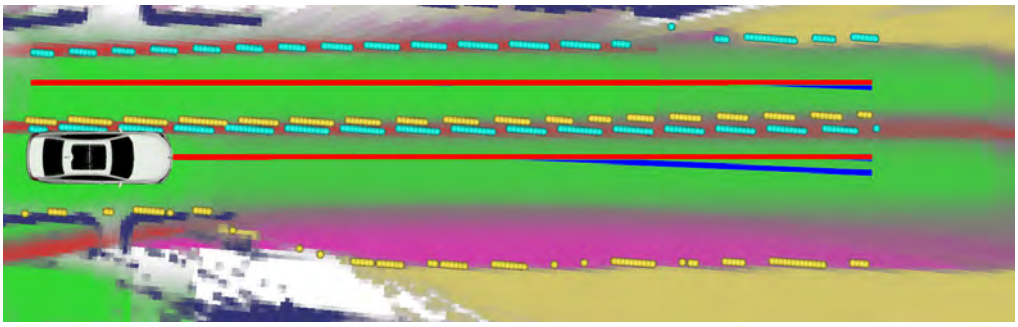
(a)

(b)

Abbildung 3.13: (a) DSTMap mit dem Ergebnis der Pfadplanung. Die gefundenen Zielpfade sind blau dargestellt, die Cluster-Pfade rot. Für die Planung wurde eine vordefinierte Pfadlänge von 15m verwendet und 2 Iterationen geplant. Die extrahierten Punkte der Fahrstreifengrenzen sind als türkise (jeweils die linke Grenze) und gelbe Punkte (jeweils die rechte Grenze) abgebildet. (b) Kamerabild der Situation.

Verkehrssituation 3: Innerstädtische Situation vor Fussgängerampel

In Abb. 3.14 ist eine innerstädtische Situation vor einer Fussgängerampel zu sehen. Auf der rechten Seite ist der Ego-Fahrstreifen nicht durch eine Markierung, sondern zunächst durch parkende Fahrzeuge (letztes Fahrzeug rechts unten im Bild auf Höhe des Ego-Fahrzeugs) und anschließend durch einen Bürgersteig begrenzt. In der DSTMap ist die Unterscheidung zwischen *Fahrbahn* ($\{M, L\}$: Magenta), *Fahrstreifen* ($\{L\}$: Grün) und *Bürgersteig* ($\{S\}$: Gelb) gut zu erkennen. Die Extraktion der Fahrstreifengrenzen erfolgt wie gewünscht: Zunächst die parkenden Fahrzeuge auf Höhe des Ego-Fahrzeugs und anschließend der Rand des Bürgersteigs.



(a)

(b)

Abbildung 3.14: (a) DSTMap mit dem Ergebnis der Pfadplanung. Die gefundenen Zielpfade sind blau dargestellt. Für die Planung wurde eine vordefinierte Pfadlänge von 15m verwendet und 2 Iterationen geplant. (b) Kamerabild der Situation.

3.5 Zusammenfassung

Für die Modellierung der aktuellen Fahrzeugumgebung wird häufig auf gridbasierte Ansätze zurückgegriffen, da diese eine direkte zeitliche und örtliche Korrelation der Messdaten abbilden bzw. herstellen.

In diesem Kapitel wurde ein neuartiger Ansatz präsentiert, um Strukturen aus einer gridbasierten Umgebungsdarstellung zu extrahieren. Aufbauend auf den Ergebnissen aus Kap. 2 wird aus der DSTMap eine Hindernis- und Kostenkarte berechnet, die in Kombination mit einer iterativen Pfadplanung zur Extraktion von potentiellen Fahrstreifen genutzt wird. Für die iterative Planung wird eine Kombination aus dem A*-Algorithmus und dem RRT für die Planung der Pfade verwendet.

Für die Hindernis- und Kostenkarte kann dabei direkt auf die Methoden der Dempster-Shafer Theorie zurückgegriffen werden, die bereits bei der Erstellung der DSTMap verwendet wurde. Das mathematische Framework muss nicht erweitert oder verlassen werden.

Im vorliegenden Fall wird der explorative Charakter der Pfadplanung verwendet, um kinematisch fahrbare Pfade mit fixer Länge zu finden. Anschließend werden die gefundenen Pfade gruppiert und ein Cluster-Repräsentant (Cluster-Pfad) pro Gruppe definiert. Der Endpunkt jedes Cluster-Pfads wird als Ausgangspunkt für eine neue Planung verwendet (iterative Planung). Im Vergleich zu einer direkten Planung von Pfaden mit der gleichen Länge, ist der iterative Ansatz **deutlich** effizienter, da jeweils nur die Enden der Cluster-Pfade neu expandiert werden.

Jeder Cluster-Pfad ist ein potentieller Fahrstreifenabschnitt. Daher werden für jeden dieser Pfade anschließend jeweils auf der linken und rechten Seite mögliche Fahrstreifengrenzen aus der DSTMap extrahiert. Im Ergebnis entsteht eine Baumstruktur von Cluster-Pfaden mitsamt linker und rechter Begrenzung.

Der vorgestellte Ansatz ist flexibel und agnostisch bzgl. des verwendeten Planungsalgorithmus. Ein Austausch des Pfadplaners ist somit jederzeit möglich ohne dass der Ansatz grundlegend verändert oder angepasst werden muss. Darüber hinaus lassen sich durch die parallele Planung mit mehreren Pfadplanern auch die jeweils linken und rechten Nachbarfahrstreifen aus der DSTMap extrahieren.

Im Vergleich zu diversen Algorithmen aus der digitalen Bildverarbeitung, mit denen ebenfalls Strukturen aus Grids extrahiert werden können (bspw. 'Region Growing'-Algorithmen oder 'Connected Component Labeling'), bietet eine pfadplanungsbasierte Methode den Vorteil, dass die extrahierten Fahrstreifen

kinematisch garantiert fahrbar sind und somit eine implizite Validierung der Ergebnisse stattfindet.

Im vorletzten Abschnitt des Kapitels wurden die Ergebnisse der Pfadplanung und der Extraktion von Fahrstreifengrenzen anhand von verschiedenartigen Situationen präsentiert. Anhand der Ergebnisse lässt sich gut erkennen, dass sich die potentiellen Fahrstreifen sowie deren linke und rechte Grenzen mit Hilfe der Pfadplanung sehr gut aus der DSTMap extrahieren lassen.

Das nun nachfolgende Kapitel widmet sich der Frage, wie aus den Ergebnissen dieses Kapitels ein vollständiges Straßenmodell erzeugt werden kann, das für die Manöverplanung und Fahrstrategie eines autonomen Fahrzeugs verwendet werden kann.

Kapitel 4

Generierung eines Straßenmodells

Mit der in Kapitel 2 bereits vorgestellten DSTMap zur Fusion von Sensordaten in einem gridbasierten Modell und der in Kapitel 3 beschriebenen Methode zur Extraktion von potentiellen Fahrstreifen auf Basis einer iterativen Pfadplanung ergibt sich für jeden Pfadplaner eine Baumstruktur von Cluster-Pfaden mit einer Vorgänger-Nachfolger-Beziehung. Darüber hinaus ist jedem Pfad jeweils eine Menge von rechten und linken Punkten zugeordnet, die eine potentielle Fahrstreifengrenze darstellen.

Das nun folgende Kapitel widmet sich der Frage, wie sich aus diesen Ergebnissen ein Straßenmodell erzeugen lässt, das für die Manöver- und Trajektorienplanung eines automatisiert fahrenden Fahrzeugs verwendet werden kann.

4.1 Einleitung

Die aus den vorangegangenen Kapiteln gewonnenen Ergebnisse können in der Regel nicht direkt dazu verwendet werden, ein Fahrzeug autonom navigieren zu lassen. Zwar ist es generell möglich, aus der Baumstruktur durch Aneinanderreihung von Pfaden über die Vorgänger-Nachfolger-Beziehung einen Gesamtpfad zu wählen, dem das Fahrzeug folgen soll. Allerdings bleiben hierbei einige wichtige Punkte unberücksichtigt. Das sind u.a. die im Folgenden aufgeführten:

- Die Pfadplanung aus Kapitel 3 berücksichtigt keinerlei dynamische Hindernisse wie andere Fahrzeuge. Dadurch droht in Anwesenheit von anderen Verkehrsteilnehmer eine Kollisionsgefahr.
- Die Cluster-Pfade befinden sich nicht zwangsläufig in der Mitte eines Fahrstreifens.

- Der Pfad ist zwar kinematisch fahrbar, allerdings findet keine Optimierung hinsichtlich des Fahrkomforts statt.
- Da es sich um Pfade und nicht um Trajektorien handelt, fehlen die Zeit- bzw. Geschwindigkeitsinformationen.

Daher ist es erforderlich, aus den Ergebnissen der Pfadplanung ein Straßenmodell zu erzeugen, das schlussendlich zur Manöver- und Trajektorienplanung verwendet werden kann. Der nachfolgende Abschnitt beschäftigt sich zunächst mit dem Stand der Technik, wie aus vorverarbeiteten Sensormessdaten ein Straßenmodell generiert werden kann. Anschließend wird in Abschnitt 4.2 hierzu ein neues Verfahren vorgestellt, das im Rahmen der vorliegenden Arbeit entwickelt worden ist.

4.1.1 Stand der Technik

Die Erzeugung eines Straßenmodells aus vorverarbeiteten Sensordaten ist bis dato in der Literatur nicht sehr häufig zu finden und nur wenige Arbeiten beschäftigen sich mit direkt vergleichbaren Themen.

Wie bereits in Kapitel 3.1.1 beschrieben existieren einerseits etliche Ansätze, um Strukturen aus Grids zu extrahieren. Dabei handelt es sich überwiegend um Randbebauungen oder dynamische Objekte. Andererseits existieren Ansätze, um Fahrstreifen direkt aus Sensordaten - ohne die Erstellung von Grids - zu schätzen. Diese Ansätze stützen sich jedoch meist auf die Erkennung von Fahrstreifenmarkierungslinien und beschränken sich daher auf Situationen, in denen diese existieren (Autobahnen, Landstraßen). Eine gute Übersicht dazu findet sich unter anderem in [99].

In [116] wird ein Verfahren beschrieben, das im Rahmen der vorliegenden Arbeit entwickelt wurde. Dort werden die lokalen Maxima von Fahrstreifenzellen über eine Nachbarschaftssuche assoziiert (lokales Clustering). Anschließend erfolgt eine Regression zur Bildung eines geometrischen Fahrstreifenmittenverlaufs.

Darüber hinaus existieren einige Ansätze, um Strukturen in Punktewolken zu finden, die auf dem RANSAC-Verfahren [178] basieren [179, 126, 180].

Einige wenige Ansätze überführen die Messdaten in eine graphenbasierte Form und sind somit grundsätzlich in der Lage, nicht nur Fahrstreifenmarkierungen, sondern auch andere Arten von Eingangsdaten, zu verarbeiten. So wird beispielsweise in [179] ein graphenbasiertes Modell mit einer Kreisbogen-Schätzung verwendet, um aus den aus Kamerabildern extrahierten Fahrstreifenmarkierungsmessungen ein Kreisbogenmodell zu generieren. Dabei wird ein „inverse

perspective mapping" durchgeführt, sowie eine Hough-Transformation zur Extraktion der Fahrstreifengrenzen. Anschließend wird das RANSAC-Verfahren genutzt, um ein Kreisbogenmodell zu schätzen.

Eine andere graphenbasierte Methode ist in [101] bzw. [181] zu finden. Dort wird ein Graph verwendet, um die Messdaten hinsichtlich ihrer Konfidenz/Unsicherheit zu optimieren. Aus gemessenen Fahrstreifenmarkierungen, den Odometrie-Positionen des Ego-Fahrzeugs und den Positionen anderer Fahrzeuge wird ein Graph aufgebaut, der mit Hilfe des GraphSLAM-Verfahrens optimiert und anschließend zur Erstellung eines Straßenmodells verwendet wird.

Ein anderes Verfahren ist in [158] beschrieben. Dort wird eine Punktewolke in ein Grid projiziert und anschließend eine Skelettberechnung durchgeführt, um die Punktemenge zu reduzieren und dicht beieinanderliegende Punkte zu entfernen. Aus dem Grid werden wiederum Punkte extrahiert und mit Hilfe einer PCA („principal component analysis") gruppiert. Allerdings sind die angegebenen Laufzeiten mit 3-10 Sekunden wesentlich zu hoch und der Ansatz ist für die vorliegende Arbeit somit nicht einsetzbar.

4.1.2 Eigener Ansatz und wissenschaftlicher Beitrag

In diesem Kapitel wird ein Ansatz vorgestellt, um aus den Ergebnissen der Pfadplanung aus Kap. 3 ein konsistentes Straßenmodell zu erzeugen. Schwerpunkt des Ansatzes ist die Interpretation der Pfade und deren Grenzen im Hinblick auf die lateralen Beziehungen der Fahrstreifen untereinander (Parallelität) sowie die korrekte Handhabung von Abzweigungen/Einmündungen, bei denen konsistente Fahrstreifengrenzen berechnet werden müssen.

Die wichtigsten Charakteristika des in diesem Kapitel vorgestellten Ansatzes sind in nachfolgender Liste kurz zusammengestellt:

- Die Repräsentation beliebiger Fahrstreifengeometrien (bspw. S-Kurven-Verläufe, nicht-parallel verlaufende Fahrstreifengrenzen) ist möglich. Es werden keine a-priori-Annahmen oder Einschränkungen bzgl. des Verlaufs oder der Form von Fahrbahn oder Fahrstreifen getroffen.
- Parallel verlaufende Fahrstreifen werden als solche erkannt und korrekt modelliert.
- Der Ansatz ist in der Lage, auch mit Abzweigungen und Einmündungen umzugehen.
- Es existieren keine Einschränkungen hinsichtlich der verschiedenen Typen von Fahrstreifenbegrenzungen (Markierungslinien, statische Hindernisse wie parkende Autos, Pylonen/Baken, Bürgersteige, etc.). Alle in der

DSTMap identifizierten Fahrstreifengrenzen können als solche verwendet werden.

- Die Methode ist unabhängig von der konkreten Implementierung der DST-Map (Kap. 2) als auch von der Pfadplanungsmethode (Kap. 3).

Im folgenden Abschnitt wird der Ansatz im Detail beschrieben.

4.2 Modellbildung

Im vorliegenden Fall soll nun ein Straßenmodell erzeugt werden, das keinerlei Einschränkungen oder Annahmen hinsichtlich des geometrischen Verlaufs eines Fahrstreifens beinhaltet. Deshalb werden alle geometrischen Verläufe als Polylinien abgebildet. Im Gegensatz zu funktionalen Modellen (Polynome, Klothoiden, Splines, etc.) garantiert dieses Modell eine fehlerfreie Abbildung der Geometrie ohne Einschränkungen.

Das Straßenmodell ($\mathcal{R}$: Road) wird als Menge von Fahrstreifen ($\mathcal{L}$: Lane) definiert:

$$\mathcal{R} := \{\mathcal{L}_1, \ldots, \mathcal{L}_n\} \tag{4.1}$$

wobei jeder Fahrstreifen aus einer linken und rechten Fahrstreifengrenze ($\mathcal{B}$: Boundary) besteht. Darüber hinaus enthält jeder Fahrstreifen Informationen über die Beziehung zu anderen Fahrstreifen:

$$\mathcal{L} := \left\{\mathcal{B}^{left}, \mathcal{B}^{right}, Pred(\mathcal{L}), Succ(\mathcal{L}), Left(\mathcal{L}), Right(\mathcal{L})\right\} \tag{4.2}$$

mit

$$\mathcal{B} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}, \mathbf{p} = (x, y)^T \tag{4.3}$$

$$Pred(\mathcal{L}_i) = \{\mathcal{L} | \mathcal{L} \in \mathcal{R} \wedge \mathcal{L} \prec \mathcal{L}_i\} \tag{4.4}$$

$$Succ(\mathcal{L}_i) = \{\mathcal{L} | \mathcal{L} \in \mathcal{R} \wedge \mathcal{L} \succ \mathcal{L}_i\} \tag{4.5}$$

Jede Fahrstreifengrenze besteht aus einer Liste von Punkten mit 2-dimensionalen Koordinaten (x, y). $Succ(\mathcal{L})$ bezeichnet all diejenigen Fahrstreifen, die dem Fahrstreifen $\mathcal{L}$ nachfolgen. Entsprechend gibt $Pred(\mathcal{L})$ den Vorgänger-Fahrstreifen

von $\mathcal{L}$ an. Per Definition kann in der vorliegenden Arbeit ein Fahrstreifen mehrere Nachfolger, jedoch nur einen Vorgänger haben. Ausserdem besitzt jeder Fahrstreifen einen potentiellen linken und rechten Nachbarn ($Left(\mathcal{L})$, $Right(\mathcal{L})$), um die Parallelität von Fahrstreifen abbilden zu können.

Es wird zunächst bewusst darauf verzichtet, für jeden Fahrstreifen eine Mittellinie anzugeben, da die Mittellinie in der Praxis ohnehin nicht zur direkten Fahrzeugregelung benutzt werden sollte. In vielen Situationen ist es nicht zweckmäßig, das Fahrzeug der Fahrstreifenmitte folgen zu lassen. Ein „Ausweichen" innerhalb des eigenen Fahrstreifens ist oftmals aus Komfort- und Sicherheitsgründen erforderlich, um einen größtmöglichen Abstand zu allen Hindernissen und Verkehrsteilnehmern zu erreichen.

Die Fahrstreifengrenzen müssen in jedem Fall ermittelt werden, da sie die Begrenzung des befahrbaren Raumes darstellen und eine Überfahrung ausgeschlossen werden muss. Die Nachbarschaftsbeziehungen der Fahrstreifen ist für eine Manöverplanung unerlässlich, da ohne diese Informationen keine Fahrstreifenwechsel geplant werden können. Weitere Informationen wie bspw. der Typ der Fahrstreifenbegrenzung (Pylone, statisches Hindernis, Bürgersteig, etc.) sind zunächst nicht Bestandteil des Modells, lassen sich später jedoch leicht durch eine Erweiterung der Definition von $\mathcal{B}$ or $\mathbf{p}$ hinzufügen. Aus den Fahrstreifengrenzen kann im Nachhinein stets auch die Fahrstreifenmitte oder die Fahrstreifenbreite berechnet werden, falls dies erforderlich sein sollte. Die Fahrstreifengrenzen müssen - da sie nicht zur direkten Fahrzeugregelung verwendet werden - keine besonderen Stetigkeitsanforderungen erfüllen.

Wie bereits in Kapitel 3 beschrieben, besteht das Ergebnis der Extraktion von potentiellen Fahrstreifen aus der DSTMap aus einer Baumstruktur von Cluster-Pfaden pro verwendetem Pfadplaner. Ein mögliches Ergebnis mit 3 Pfadplanern und einer gefundenen Abzweigung ist in Abb. 4.1 dargestellt.

Für die Modellbildung werden nun aus den Cluster-Pfaden entsprechende Fahrstreifenabschnitte generiert. Diese bilden die Grundlage, um ein Straßenmodell zu erzeugen. Die Fahrstreifenabschnitte dienen dabei als Zwischenrepräsentation, aus der in einem späteren Schritt die Fahrstreifen $\mathcal{L}$ für das Straßenmodell $\mathcal{R}$ generiert werden.

Die Erzeugung und Behandlung dieser Fahrstreifenabschnitte wird nachfolgend erläutert.

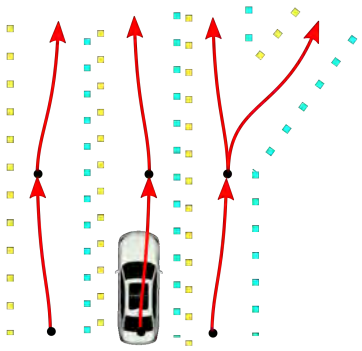


Abbildung 4.1: Schematische Darstellung eines Ergebnisses der Extraktion von potentiellen Fahrstreifen gemäß Kap. 3. Es wurden 3 Pfadplanungsinstanzen verwendet, um neben dem Ego-Fahrstreifen auch die Nachbarfahrstreifen extrahieren zu können. Abgebildet sind jeweils die Cluster-Pfade und die extrahierten Fahrstreifengrenzpunkte.

4.2.1 Bildung von Fahrstreifenabschnitten

Ein Fahrstreifenabschnitt repräsentiert einen Abschnitt eines Fahrstreifens in longitudinaler Richtung. Diese Art der Zwischenrepräsentation erleichtert die Verarbeitung der Cluster-Pfade. Je nach Situation können Fahrstreifenabschnitte erzeugt, verändert, miteinander verglichen, fusioniert oder auch weiter in kleinere Abschnitte unterteilt werden.

Die Fahrstreifenabschnitte werden aus den Cluster-Pfaden gebildet, indem jeder Cluster-Pfad auf seine Eigenschaften hin untersucht wird. Anschließend wird ein entsprechender Fahrstreifenabschnitt angelegt. Dabei müssen für jeden Cluster-Pfad folgende Fälle untersucht werden:

1. Der Cluster-Pfad hat keinen oder einen Nachfolger.
2. Der Cluster-Pfad hat mehr als einen Nachfolger.
3. Die linke oder rechte Fahrstreifengrenze des Cluster-Pfades weist größere Lücken auf.
4. Eine Fahrstreifengrenze des Cluster-Pfades liegt dicht an der Fahrstreifengrenze eines benachbarten Cluster-Pfades (eines anderen Pfadplaners).

Im ersten Fall kann davon ausgegangen werden, dass der Cluster-Pfad in der Realität genau einem Abschnitt eines Fahrstreifen entspricht. Somit wird für diesen Cluster-Pfad ein Fahrstreifenabschnitt erzeugt und die zum Pfad gehörigen Fahrstreifengrenzen werden dem neu erzeugten Fahrstreifenabschnitt zugeordnet.

Beim zweiten Fall handelt es sich um eine Kreuzung, Einmündung oder anderweitig erkannte Abzweigung, da ausgehend von einem gemeinsamen Punkt zwei Cluster-Pfade gefunden wurden, die nicht auf demselben Fahrstreifen liegen. Da in Kreuzungsbereichen oftmals keine klaren Fahrstreifengrenzen existieren, müssen in diesem Fall *virtuelle* Fahrstreifengrenzen generiert werden. Dies wird unten im Abschnitt „Modellierung der Fahrstreifengrenzen" näher erläutert.

Der dritte Fall behandelt unterbrochene Fahrstreifengrenzen. Es kann zum Beispiel vorkommen, dass durch Verdeckungen oder eine begrenzte Sensorreichweite keine durchgehende Fahrstreifengrenze erkannt werden kann. Lücken, zwischen zwei aufeinanderfolgenden Punkten in einer Fahrstreifengrenze, die größer als eine durchschnittliche Fahrzeugbreite sind, erfordern eine genauere Betrachtung. Einerseits könnte es sich um eine Abzweigung oder Einmündung einer anderen Straße handeln, in die (noch) kein Pfad geplant worden ist. In einem zukünftigen Zeitschritt könnte jedoch ein Pfad gefunden wird, der in diese Lücke plant und die Abzweigung/Einmündung „findet". Andererseits könnte es sich in der Realität aber auch nicht um eine Abzweigung/Einmündung, sondern eine tatsächliche Fahrstreifengrenze handeln, die fälschlicherweise nicht erkannt worden ist (Messfehler, Verdeckungen, zu geringe Sensorreichweite). Um beide Möglichkeiten von vorneherein unterscheiden und berücksichtigen zu können, müssen diese Lücken in den Fahrstreifengrenzen identifiziert und die entsprechenden Pfadabschnitte in separate Fahrstreifenabschnitte umgewandelt werden.

Der vierte Fall deutet daraufhin, dass die Fahrstreifen, zu denen die beiden Cluster-Pfade jeweils gehören, benachbart sind. In dieser Situation müssen die beiden Fahrstreifengrenzen zu einer einzigen gemeinsamen Grenze fusioniert werden.

In Abb. 4.2 sind für diese 4 Fälle die entstehenden Fahrstreifenabschnitte dargestellt.

Im folgenden Abschnitt wird nun die Bildung der Fahrstreifengrenzen für die oben genannten Fälle erläutert.

Modellierung der Fahrstreifengrenzen

Fahrstreifen sind im Wesentlichen definiert durch ihre Fahrstreifengrenzen. Deshalb wird im Folgenden kurz beschrieben, wie mit Hilfe der einzelnen Fahrstreifenabschnitte die Fahrstreifengrenzen ermittelt und modelliert werden können.

Im einfachsten Fall, ((a) in Abb. 4.2), wurde für jeden Cluster-Pfad genau ein Fahrstreifenabschnitt erzeugt, der für jede Grenze jeweils eine durchgehende

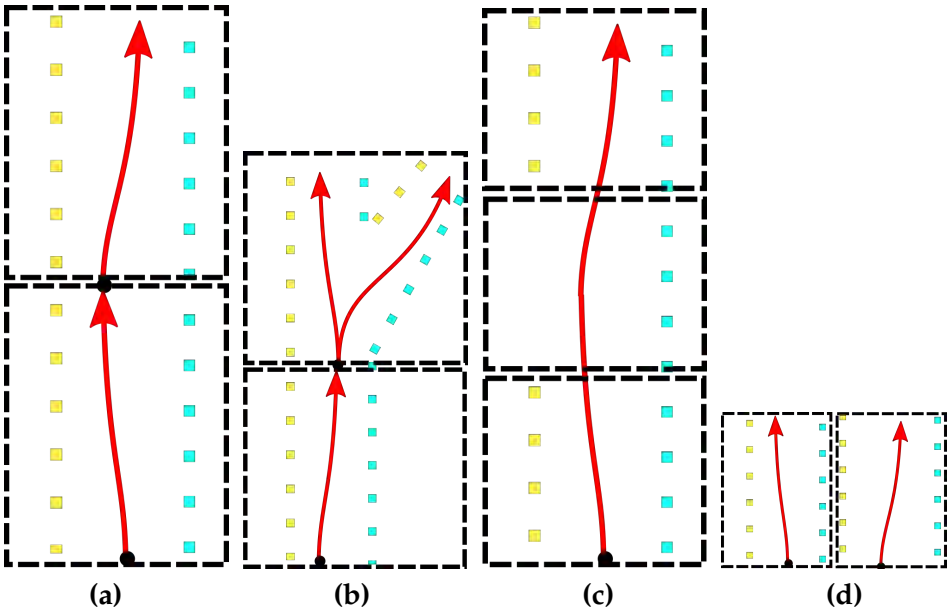


Abbildung 4.2: Bildung von Fahrstreifenabschnitten in Abhängigkeit von den Ergebnissen der Pfadplanung. (a) Cluster-Pfad mit einem Nachfolger, (b) Abzweigung, (c) Lücken in der Fahrstreifengrenze, (d) benachbarte Fahrstreifen.

Polylinie besitzt. Diese Polylinien können deshalb direkt für die spätere Bildung der Fahrstreifen verwendet werden.

Im Fall von parallel verlaufenden Fahrstreifen, wie in Abb. 4.2d dargestellt, muss diese Parallelität erkannt und eine **gemeinsame** Fahrstreifengrenze gebildet werden. Hierzu werden die Punkte mit Hilfe des 'nearest neighbour'-Verfahren assoziiert. Dabei wird für jeden Punkt der einen Fahrstreifengrenze der Abstand zum nächstgelegenen Punkt der anderen Fahrstreifengrenze berechnet. Ist der Abstand geringer als ein vordefinierter Schwellwert, werden die beiden Punkte zu einem Punkt zusammengefasst. Dieser repräsentiert damit die **gemeinsame** Grenze der Fahrstreifen an diesem Ort. Ein alternatives Verfahren zur Assoziation/Fusion der Fahrstreifengrenzen stellt die Untersuchung der Fläche zwischen den beiden Grenzen dar. Hierzu wird - ähnlich wie beim Clustering der Zielpfade in Kap. 3.2.2 - aus den Fahrstreifengrenzpunkten ein Polygon gebildet und auf die DSTMap projiziert. Anschließend wird untersucht, ob sich in diesem Polygon Zellen befinden, die nicht als *Marking* (Fahrstreifenmarkierungslinie) klassifiziert worden sind. Falls das Polynom ausschließlich aus Fahrstreifenmarkierungslinien-Zellen besteht, können die beiden Fahrstreifengrenzen assoziiert und fusioniert werden. Letztere Methode garantiert implizit, dass bei den als parallel erkannten Fahrstreifen ein Fahrstreifenwechsel physikalisch ohne Kollision möglich ist, da sichergestellt ist, dass sich zwischen

den beiden Fahrstreifen nur eine Fahrstreifenmarkierungslinie und keine anderen Hindernisse (gemeint sind alle andere Hypothesen der DSTMap) befinden. Sollte diese Parallelitätsanalyse ergeben, dass die Fahrstreifen nur in einem gewissen Teil der untersuchten Fahrstreifenabschnitte parallel verlaufen, werden die Abschnitte in jeweils einen Abschnitt aufgeteilt, bei dem die Fahrstreifen parallel verlaufen und einen weiteren, bei dem die Fahrstreifen nicht parallel verlaufen. Dieses Vorgehen erleichtert die spätere Modellbildung der einzelnen Fahrstreifen, weil die Nachbarschaftsbeziehung der Fahrstreifen damit auf der Ebene der Fahrstreifenabschnitte bereits festgestellt worden ist und sich innerhalb eines Abschnitts nicht ändern. In Abb. 4.3 ist eine solche Situation schematisch dargestellt.

(a) (b)

Abbildung 4.3: (a) Situation mit zunächst parallel verlaufenden Fahrstreifen. (b) Ergebnis der Fusion gemeinsamer Fahrstreifengrenzen. Es werden für den parallel verlaufenden Abschnitt und den nicht-parallel verlaufenden Abschnitt des Fahrstreifens separate Fahrstreifenabschnitte gebildet.

Die Situation in Abb. 4.2b zeigt ein typisches Ergebnis der Fahrstreifenextraktion in einer Situation mit einer Abzweigung oder Einmündung.

Wie bereits in Kapitel 2 erläutert, stellen die baulichen Fahrbahn- und Fahrstreifenbegrenzungen nicht immer die inhaltlich relevanten Begrenzungen dar. Für das Straßenmodell sind schlussendlich nur die *semantischen* Begrenzungen von Bedeutung, da sie den für die aktuelle Fahraufgabe zur Verfügung stehenden Bereich begrenzen. Im Fall von Abzweigungen, Einmündungen oder Kreuzungen existieren oftmals keine physischen Fahrstreifengrenzen im unmittelbaren Kreuzungs- oder Einmündungsbereich. Dennoch werden für die Manöver- und Trajektorienplanung die *semantischen* Grenzen benötigt. Deshalb muss im endgültigen Straßenmodell für jeden Fahrstreifen eine Grenze existieren. Die Aufgabe besteht darin, innerhalb der Kreuzung die semantischen Fahrstreifengrenzen für jeden Fahrstreifen zu berechnen.

Im vorliegenden Beispiel einer Abzweigung nach rechts müssen jeweils die rechte Fahrstreifengrenze für den geradeaus-führenden Fahrstreifen und die linke Fahrstreifengrenze für den rechts abbiegenden Fahrstreifen gefunden werden.

Dafür wird eine 'penalized regression' mit Splines verwendet [182], um die semantischen Grenzen im Kreuzungsbereich zu generieren. Dabei führt das durch die Regression gefundene geometrische Modell möglichst exakt durch die bereits existierenden Grenzpunkte „hindurch". Anschließend erfolgt eine Aufteilung in 5 Fahrstreifenabschnitte: Der erste Abschnitt beinhaltet den Bereich bis zum Beginn des Kreuzungsbereich. Den Kreuzungsbereichs bilden zwei verschiedene Abschnitte, die stark überlappen. Dabei beinhaltet einer den Teil des geradeaus führenden Fahrstreifens und der andere Abschnitt den rechts abbiegenden Teil. Die anderen beiden Abschnitte liegen jeweils hinter dem Kreuzungsbereich.

Das hier gezeigte Beispiel der Generierung von virtuellen Fahrstreifengrenzen anhand einer Abzweigung/Einmündung lässt sich auch auf andere Kreuzungsgeometrien entsprechend übertragen.

Nachdem nun sowohl die Erzeugung der Fahrstreifenabschnitte als auch die Berechnung aller Fahrstreifengrenzen pro Abschnitt abgeschlossen ist, erläutert der nun folgende Abschnitt, wie daraus das endgültige Straßenmodell erzeugt wird.

4.2.2 Bildung des Straßenmodells

Um schlussendlich ein endgültiges Straßenmodell $\mathcal{R}$ zu erzeugen, werden die einzelnen Fahrstreifenabschnitte zu Fahrstreifen zusammengesetzt. Beginnend mit jeweils dem ersten Fahrstreifenabschnitt jedes Pfadplaners wird ein Fahrstreifen $\mathcal{L}$ angelegt und alle Eigenschaften des Fahrstreifenabschnitts (Grenzen, Beziehungen) übernommen. Anschließend werden entsprechend der Vorgänger-Nachfolger-Beziehungen die nachfolgenden Fahrstreifenabschnitte in longitudinaler Richtung assoziiert und an den bereits vorhandenen Fahrstreifen angestückt. Besitzt ein nachfolgender Fahrstreifenabschnitt jedoch signifikant andere Eigenschaften als der Fahrstreifen, zu dem dieser assoziiert werden soll, wird ein neuer Fahrstreifen angelegt und über die Vorgänger-Nachfolger-Beziehung mit dem vorhandenen Fahrstreifen verknüpft. Dies erfolgt immer genau dann, wenn

- sich die Parallelitätsbeziehungen ändern oder
- es sich um eine Abzweigung, Kreuzung oder Einmündung handelt

Dadurch ist stets gewährleistet, dass sich die grundlegenden Eigenschaften innerhalb des Fahrstreifens nicht ändern und sämtliche Nachbarschaftsbeziehungen von Anfang bis Ende des Fahrstreifens gültig sind.

Zusätzlich zu einer punktweisen Darstellung von Fahrstreifengrenzen, ist oftmals eine kontinuierliche Repräsentation notwendig. Dies gilt insbesondere dann,

wenn nur wenige Grenzpunkte extrahiert werden konnten oder sich zwischen den einzelnen Punkten größere Lücken befinden. Um einen glatten und kontinuierlichen Verlauf der semantischen Fahrstreifengrenzen zu erzeugen, können die Grenzpunkte der Fahrstreifen durch ein funktionales Modell approximiert werden. Bei diesem Verfahren werden für ein vorgegebenes mathematisches Modell diejenigen Parameter ermittelt, die die vorgegebenen Grenzpunkte am besten annähern (Regression). Da die Grenzpunkte der einzelnen Fahrstreifen bereits sortiert sind, lassen sich eine Vielzahl von Regressionsmethoden verwenden. Im einfachsten Fall können die Grenzen durch einen simplen Polygonzug repräsentiert werden. Im vorliegenden Fall wird jedoch eine 'penalized regression' mit Splines verwendet [182], da sich bei diesem Verfahren die Flexibilität der Modellierung und die Glätte der entstehenden Kurven beeinflussen lässt.

Ein weiterer Vorteil bei der zusätzlichen Generierung eines geometrischen Modells ist die Möglichkeit der Berechnung von Tangenten, Krümmungen und Normalen. Diese können beispielsweise für eine spätere Berechnung der Fahrstreifenbreite verwendet werden.

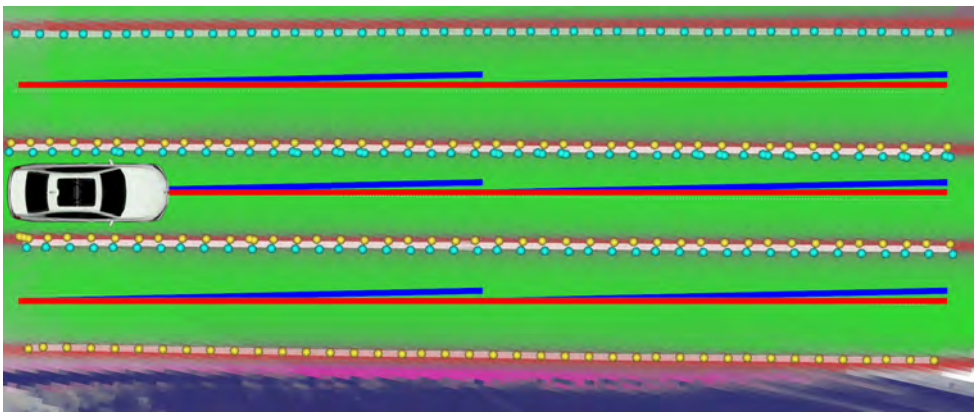
Das erstellte Straßenmodell ist so flexibel, dass es sich bei Bedarf sowohl auf der Ebene der Fahrstreifen $\mathcal{L}$ als auch auf der Ebene der Straße $\mathcal{R}$ durch weitere Attribute wie bspw. Fahrtrichtungen, Haltelinien, Geschwindigkeitsbegrenzungen oder Straßentypen einfach erweitern lässt.

4.3 Ergebnisse

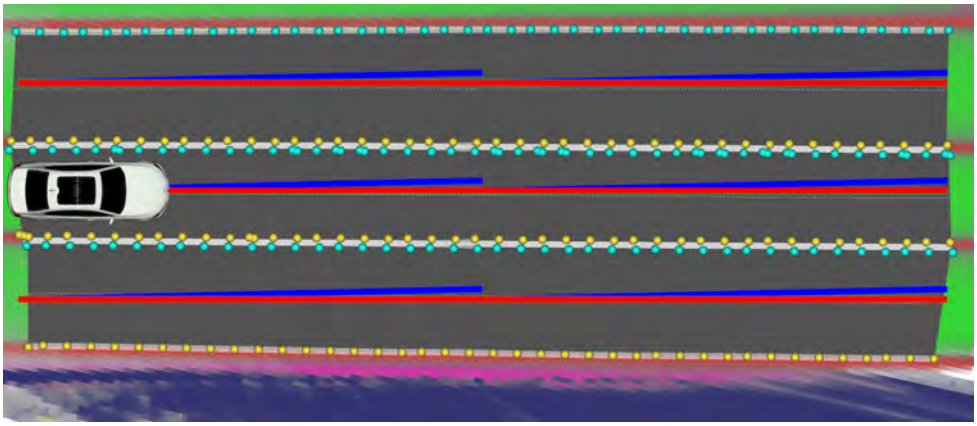
Der folgende Abschnitt zeigt einige qualitative Ergebnisse der Fahrstreifenextraktion mit Hilfe der Pfadplanung von verschiedenen Verkehrssituationen.

Verkehrssituation 1: Bundesstraße mit parallel verlaufenden Fahrstreifen

Die erste Situation zeigt eine typische Verkehrssituation auf einer Bundesstraße mit 4 parallel verlaufenden Fahrstreifen, die auf eine Kreuzung führen. In Abb. 4.4a ist die DSTMap mit dem Ergebnis der in diesem Kapitel vorgestellten Verfahren abgebildet. Es wurden für drei Pfadplaner jeweils 2 Iterationen geplant. Anschließend wurden für jeden Cluster-Pfad jeweils die linken und rechten Fahrstreifengrenzen als Punkte extrahiert. Das Ergebnis zeigt eine durchgängige Extraktion der Punkte der Fahrstreifengrenzen - sowohl für den Ego-Fahrstreifen als auch für beide Nachbarfahrstreifen. In Abb. 4.4b ist das Ergebnis der Straßenmodellbildung zu sehen. Bei allen drei Fahrstreifen wurde korrekt erkannt, dass sie parallel verlaufen und es wurden jeweils **gemeinsame** Fahrstreifengrenzen gebildet.



(a)



(b)

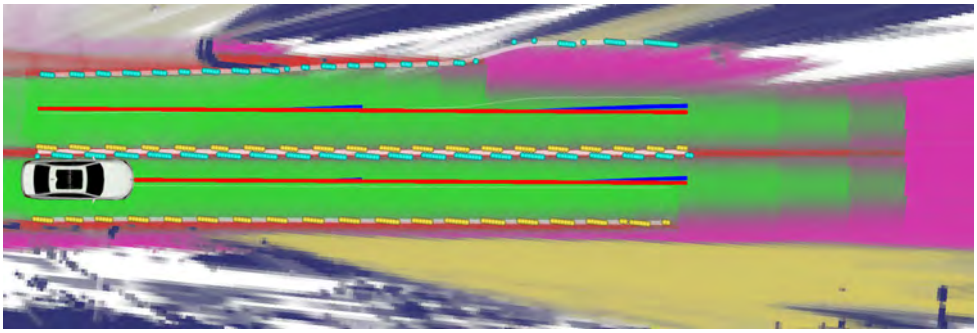
(c)

Abbildung 4.4: (a) DSTMap mit dem Ergebnis der Pfadplanung und der Modellbildung. Die gefundenen Zielpfade sind blau, die Cluster-Pfade rot dargestellt. Für die Planung wurde eine vordefinierte Pfadlänge von 15m verwendet und 2 Iterationen geplant. (b) Generiertes Straßenmodell mit drei parallel verlaufenden Fahrstreifen mit gemeinsamen Fahrstreifengrenzen (weiße Linien). (c) Kamerabild der Situation.

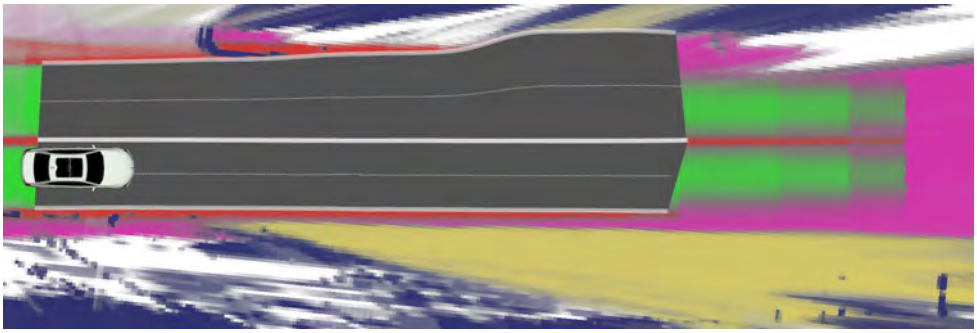
Verkehrssituation 2: Innerstädtische Situation mit parkenden Fahrzeugen

Die zweite Situation ist eine kleine innerstädtische Straße mit jeweils einem Fahrstreifen pro Fahrtrichtung, parkenden Fahrzeugen, Parkbuchten und einem Bürgersteig. Somit beinhaltet die Situation typische Elemente von städtischer Straßeninfrastruktur. Ein Bild der Situation sowie die generierten Ergebnisse sind in Abb. 4.5 abgebildet.

Die Parkbucht auf der rechten Seite wird in der DSTMap fälschlicherweise als Bürgersteig klassifiziert. Dies ist nicht überraschend, da bei der hier verwendeten semantischen Segmentierung Parkbuchten nicht separat klassifiziert werden. Insoweit ist die Klassifizierung als Bürgersteig am Naheliegendsten und hat auf das Ergebnis des Straßenmodells keine negative Auswirkung. Der tatsächliche Bürgersteig wird korrekt als Bürgersteig klassifiziert. Aufgrund von parkenden Fahrzeugen auf der linken Seite ist der Fahrstreifen der Gegenrichtung auf Höhe des Ego-Fahrzeugs zunächst schmaler als baulich vorgesehen. Die parkenden Fahrzeuge bilden hier die *semantische* Fahrstreifengrenze. Erst hinter dem letzten Fahrzeug entspricht die bauliche Fahrstreifengrenze (Bürgersteig) auch der semantischen. Sowohl die DSTMap als auch das generierte Straßenmodell bilden diese Tatsache korrekt ab. Die Breite des linken Fahrstreifen ändert sich und der Fahrstreifen wird breiter. Dies ist auch an der Änderung des Verlaufs der Fahrstreifenmittenlinie (weiße Punkte in Abb. 4.5b) gut zu erkennen.



(a)



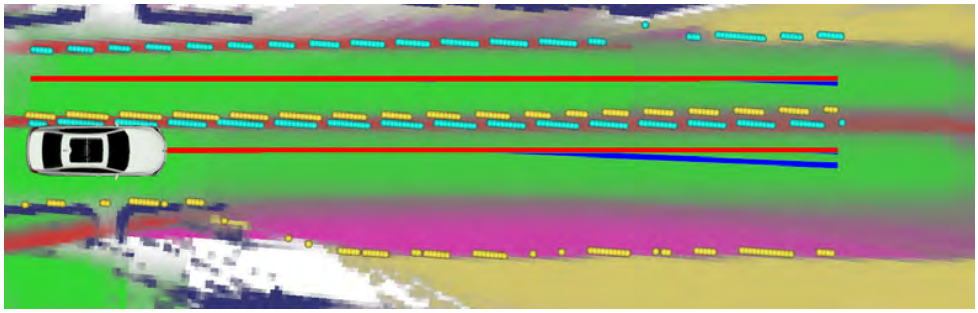
(b)

(c)

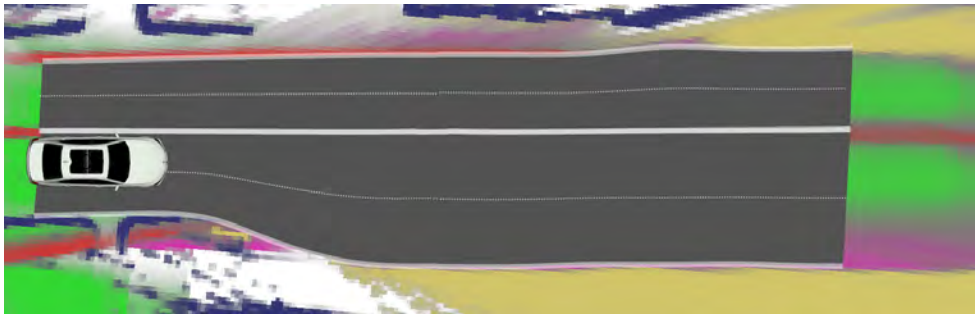
Abbildung 4.5: (a) DSTMap mit dem Ergebnis der Pfadplanung mit einer Pfadlänge von 15m und 2 Iterationen. (b) Generiertes Straßenmodell mit Fahrstreifengrenzen (weiße Linien) und Fahrstreifenmitte (weiße Punkte). Die Verbreiterung des Fahrstreifens des Gegenverkehrs ist deutlich zu erkennen. (c) Kamerabild der Situation.

Verkehrssituation 3: Innerstädtische Situation vor Fussgängerampel

Die dritte Situation ist identisch mit der Situation in Abb. 3.14 aus Kapitel 3. Der Ego-Fahrstreifen ist auf der rechten Seite auf Höhe des Ego-Fahrzeugs zunächst durch parkende Fahrzeuge und anschließend durch einen Bürgersteig begrenzt. Durch den Wegfall der parkenden Fahrzeuge verbreitert sich der Fahrstreifen. Das Straßenmodell wird auch in dieser Situation korrekt erzeugt und der Ego-Fahrstreifen verbreitert sich entsprechend, wie in Abb.4.6b gut zu erkennen ist.



(a)



(b)

(c)

Abbildung 4.6: (a) DSTMap wie in Situation 2. (b) Generiertes Straßenmodell mit Fahrstreifengrenzen (weiße Linien) und Fahrstreifenmitte (weiße Punkte). Die Verbreiterung des Ego-Fahrstreifens ist deutlich zu erkennen. (c) Kamerabild der Situation.

4.4 Zusammenfassung

Ausgehend von den Ergebnissen aus Kapitel 2 und 3 wurde in diesem Kapitel erläutert, wie das Straßenmodell generiert wird.

Für eine erfolgreiche Fahrstrategie und Manöverplanung reichen die Pfade, die aus den Ergebnissen von Kapitel 3 stammen, nicht aus. So sind die Pfade zwar kinematisch fahrbar, jedoch nicht zur direkten Steuerung des Fahrzeugs geeignet. Es fehlen u.a. Zeit- oder Geschwindigkeitsvorgaben, die Berücksichtigung dynamischer Objekte sowie eine Optimierung hinsichtlich des Fahrkomforts. Abgesehen davon lassen sich ohne ein entsprechendes Straßenmodell keine Manöver wie bspw. Fahrstreifenwechsel planen. Hier ist ein Straßenmodell unerlässlich. Auch für andere Einsatzzwecke wie bspw. eine sensorbasierte Validierung der HD-Karte wird ein Straßenmodell benötigt.

Daher wurde in diesem Kapitel ein Straßenmodell definiert, das alle erforderlichen Eigenschaften/Bedingungen erfüllt und gleichzeitig eine möglichst geringe Komplexität aufweist. Es ist in der Lage, beliebiger Fahrstreifengeometrien, parallel und nicht-parallel verlaufende Fahrstreifen, verschiedenen Typen von Fahrstreifengrenzen und Vorgänger-Nachfolger-Beziehungen zu repräsentieren.

Für die Generierung des Straßenmodells wurde zunächst die Baumstruktur von Cluster-Pfaden und deren Grenzen in sog. Fahrstreifenabschnitte überführt. Anschließend wurden für die einzelnen Abschnitte unter Berücksichtigung ihrer Beziehungen untereinander die Fahrstreifengrenzen gebildet. Schlussendlich wurden aus den einzelnen Abschnitten Fahrstreifen erzeugt und zu einem Straßenmodell zusammengesetzt.

Die Funktionsfähigkeit und Leistungsfähigkeit des Ansatzes wurde anhand von realen Sensormessdaten in verschiedenen innerstädtischen Verkehrssituationen demonstriert. In allen vorgestellten Situationen konnte aus den Ergebnissen der DSTMap und der Pfadplanung ein entsprechendes Straßenmodell generiert werden, das für eine Fahrstrategie und Manöverplanung verwendet werden kann.

Kapitel 5

Evaluierung und Bewertung

Im Folgenden werden die Verfahren evaluiert, die in den vorangegangenen Kapiteln vorgestellt worden sind. Jedes der vorangegangenen Kapitel - mit Ausnahme von Kapitel 1 - enthält einen Abschnitt, in dem die für das Kapitel jeweils relevanten Ergebnisse bereits qualitativ vorgestellt wurden. In diesem Kapitel werden nun Methoden beschrieben, mit deren Hilfe eine quantitative Evaluierung der DSTMap und des generierten Straßenmodells durchgeführt werden kann. Zudem wird jeweils exemplarisch eine Auswertung präsentiert. Eine umfangreiche qualitative Auswertung ist wegen der fehlenden großflächigen Verfügbarkeit von Referenzdaten leider bis dato nicht möglich.

Zunächst werden das verwendete Versuchsfahrzeug sowie die Mess- und Referenzdatengewinnung erläutert. Anschließend folgt in Abschnitt 5.2 die Evaluierung der DSTMap und in Abschnitt 5.3 die Bewertung des Straßenmodells.

5.1 Mess- und Referenzdatengewinnung

Für die vorliegende Arbeit wurde ein modifizierter BMW 7er (G12) verwendet. Das Versuchsfahrzeug ist in Abb. 5.1 abgebildet.

Es ist mit 5 Laserscannern (LiDAR-Sensoren), einem kamerabasierten System zur Erkennung von Fahrstreifenmarkierungslinien, einer Stereo-Kamera, sowie Radarsensoren für den Nah- und Fernbereich ausgerüstet. Die Radarsensoren wurden für die vorliegende Arbeit jedoch nicht verwendet. Der Sichtbereich der Sensoren ist in Abb. 5.2 dargestellt.

Jeder Laserscanner verfügt über 4 horizontale Scan-Ebenen mit einem horizontalen Sichtbereich von 110°. Die horizontale Winkelauflösung beträgt 0.25°. Die Scan-Ebenen sind in einem vertikalen Winkelabstand von jeweils 0.8° angeordnet. Die Abtastfrequenz für einen vollständigen Scan aller Ebenen liegt bei 25

Abbildung 5.1: Photo des verwendeten Versuchsfahrzeug. Es kam ein modifizierter BMW 7er (G12) zum Einsatz, der mit entsprechenden Sensoren zur Umfelderkennung ausgestattet ist. Das Fahrzeug besitzt ein elektronisch steuerbares Lenk-, Brems- und Antriebssystem. Es ist damit in der Lage, automatisiert zu fahren (Quelle [183]).

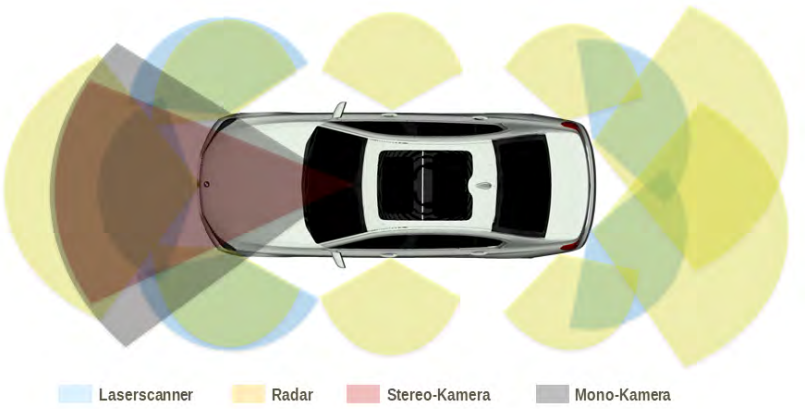


Abbildung 5.2: Sichtbereiche der verwendeten Sensorik. Die Sensoren sind so angeordnet, dass eine fast vollständige 360°-Umfelderfassung möglich ist.

Hz. Die mittlere Reichweite ist mit ca. 200m bei einer Genauigkeit der Entfernungsmessung von 0.1m angegeben.

Das kamerabasierte System zur Erkennung der Fahrstreifenmarkierungslinien stammt von einem Fremdhersteller und stellt erkannte Markierungslinien in Form von geometrischen Objekten zur Verfügung. Die dazugehörigen Kameras arbeiten mit Graustufenbildern, aus denen die Linien mit einer Frequenz von 36 Hz extrahiert werden.

Die Stereo-Kamera ist hinter der Windschutzscheibe des Fahrzeugs verbaut und hat eine Basis von 12 cm. Es werden RGB-Bilder mit einer Auflösung von 1024x 510 px mit einer Frequenz von 16 Hz und eine dazugehörige Disparitätskarte erzeugt.

Darüber hinaus ist im Fahrzeug ein Messtechnik-Computer verbaut, der die Messdaten aller Sensoren entgegennimmt, verarbeitet und ggf. auch aufzeichnet. Somit ist einerseits ein „Online"-Betrieb möglich, bei dem alle Software-Module während der Fahrt laufen und das Fahrzeug ggf. automatisiert fahren kann. Andererseits ist es aber auch möglich, das Fahrzeug manuell zu fahren und alle Sensordaten aufzuzeichnen. Diese können dann „offline" am Arbeitsplatz-Rechner abgespielt und alle Software-Funktionen nachprozessiert werden.

Für die in der vorliegenden Arbeit entwickelten Konzepte wurde eine prototypische Umsetzung in der Programmiersprache C++ durchgeführt. Zum Einsatz kam ein 64-bit Ubuntu Linux mit dem GNU C++-Compiler und dem Robot Operating System (ROS™ [184]). Für die DSTMap wurde eine Implementierung auf der GPU mit Hilfe des CUDA™ Frameworks von nVidia™ entwickelt.

Für eine quantitative Bewertung der DSTMap und des Straßenmodells ist die Verfügbarkeit von Referenzdaten („ground truth"-Daten) erforderlich. Durch einen Vergleich von DSTMap/Straßenmodell mit den Referenzdaten können die Güte und Qualität bestimmt werden. Als Referenzdaten dient in der vorliegenden Arbeit eine hochgenaue Straßenkarte der Umgebung, die alle statischen Straßeninfrastrukturelemente (Fahrstreifen, Bürgersteige, Ampeln, Verkehrsschilder, etc.) enthält. Um jedoch die Karteninhalte nutzbringend verwenden zu können, muss jederzeit die Position und Ausrichtung des eigenen Fahrzeugs relativ zur Karte hinreichend exakt bestimmt werden können. Hierfür wird ein satellitengestütztes Positionssystem (DGPS) mit gekopppelter Inertialsensorik verwendet. Das System ist in der Lage, die Position und Ausrichtung in globalen Koordinaten (WGS84) mit einer Genauigkeit von $0.01m$ (Position) bzw. 0.1° (Ausrichtung) zu bestimmen und arbeitet mit einer Frequenz von 100 Hz. Ein solches Systeme ist in Abb. 5.3 abgebildet.

Für die Erzeugung der hochgenauen Straßenkarte (auch HD-Karte) wurde ein Fahrzeug mit einem wesentlich umfangreicheren Sensor-Setup verwendet, als dasjenige, mit dem die Messdaten aufgezeichnet wurden. Die zu kartierende Strecke wurde mehrfach abgefahren und mit hochauflösenden Laserscannern und Kameras erfasst. Aus diesen Daten wurden halbautomatisch geometrische Fahrstreifensegmente extrahiert, topologisch miteinander verbunden und mit semantischen Attributen versehen.

Die HD-Karte beinhaltet eine geometrische Beschreibung aller Fahrstreifen und

Abbildung 5.3: Abbildung eines Positionierungssystems mit DGPS und Inertialsensorik. Hier das System „RT3000" der Firma „Oxford Technical Solutions". Quelle: Webseite des Herstellers.

Kreuzungen, sowie Bürgersteige. Jeder Fahrstreifen besitzt eine Mittellinie, sowie eine rechte und linke Grenze. Für die Modellierung wurden Polylinien verwendet. Die Straßentopologie wurde durch entsprechende Vorgänger-Nachfolger-Beziehungen der Fahrstreifen abgebildet. Die HD Karte ist *global* exakt referenziert, d.h. sie liegt in weltfesten Koordinaten vor und ist hinreichend „genau" (die Kartenungenauigkeit ist vernachlässigbar klein im Vergleich zur Lokalisierungsungenauigkeit). Abb. 5.4 zeigt eine typische innerstädtische Verkehrssituation mit dem Ego-Fahrzeug und der HD-Karte.

Abbildung 5.4: Ausschnitt der verwendeten HD-Karte für eine innerstädtische Verkehrssituation während der Fahrt. Dargestellt sind neben dem Ego-Fahrzeug (Auto-Modell) die einzelnen Fahrstreifen (dunkelgrau), die Fahrstreifengrenzen (graue Linien), Bürgersteige (gelb) sowie Parkbuchten (blau).

5.2 Bewertung der DSTMap

Um die DSTMap zu bewerten wird der entsprechende Ausschnitt der HD-Karte (in Kombination mit der hochgenauen Lokalisierung) in ein Grid mit identischer Position und Größe wie der DSTMap projiziert. Anschließend kann die DSTMap zellweise mit dem Grid der HD-Karte verglichen und ausgewertet werden. In Analogie zum Begriff „DSTMap" wird das Grid aus der HD-Karte im Folgenden als HDMap bezeichnet.

Im Gegensatz zur DSTMap, bei der pro Zelle jeweils eine 'belief mass' pro Hypothese gespeichert ist, enthalten die Zellen HDMap jeweils nur einen einzigen numerischen Wert, der angibt, welche der Hypothesen laut HD-Karte korrekt ist. Für die Erstellung dieser HDMap werden die einzelnen Punkte der Fahrstreifengrenze sowie die Flächen der Fahrstreifen und Bürgersteige aus der HD-Karte in das Grid projiziert und die betroffenen Zellen mit den entsprechenden numerischen Werten gefüllt.

Für den zellweisen Vergleich der DSTMap mit der HDMap werden folgende Festlegungen getroffen:

- Für die Auswertung muss die Summe aller 'belief masses' der auszuwertenden Zelle größer sein, als ein vordefinierter Schwellwert. Andernfalls wird die Zelle der DSTMap nicht mit der HDMap verglichen. Es werden damit nur diejenigen Zellen in der Auswertung berücksichtigt, bei denen die Sensormessdaten hinreichend Informationen ('belief mass') über die Zelle zur Verfügung stellen. *Unbekannte* Zellen werden somit nicht ausgewertet.
- Um die Zellen auswerten zu können, wird davon ausgegangen, dass pro Zelle diejenige Hypothese mit der höchsten 'belief mass' der Wahrheit entspricht.

So kann für jede Zelle der DSTMap durch den Vergleich mit der HDMap evaluiert werden, ob die Zelle der DSTMap korrekt klassifiziert worden ist. Anschließend werden die Ergebnisse des zellweisen Vergleichs in einer sog. „Wahrheitsmatrix" (confusion matrix) gespeichert. Sie beinhaltet die 4 Werte *Richtig-Positiv*, *Richtig-Negativ*, *Falsch-Positiv* und *Falsch-Negativ*. Mit Hilfe der Wahrheitsmatrix bzw. den 4 Bestandteilen können weitere statistische Gütekriterien wie bspw. Sensitivität, Exaktheitsquote oder ROC-Kurven berechnet werden. Eine Veranschaulichung der 4 Werte der Wahrheitsmatrix ist in Abb. 5.5 dargestellt.

Für das in Kap. 2 definierte 'Frame of Discernment' $\Omega := \{L, M, S, O\}$ ergeben sich daher 4 Wahrheitsmatrizen - für jede Hypothese jeweils eine. Leider sind statische Elemente, die nicht zur Straßeninfrastruktur zählen - wie bspw.

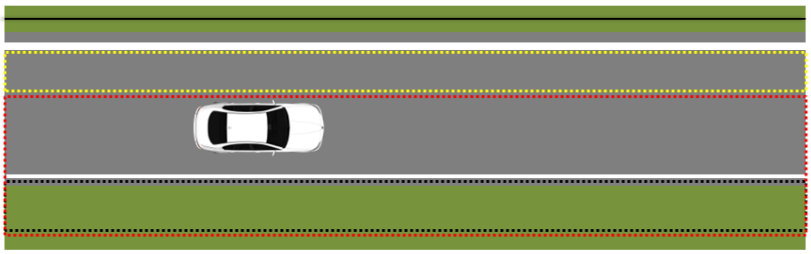


Abbildung 5.5: Graphische Veranschaulichung der Wahrheitsmatrix anhand eines Vergleichs der DSTMap mit der HDMap am Beispiel der Hypothese *Lane*. Der Bereich zwischen den beiden weißen Linien stellt den *Lane*-Bereich aus der HDMap dar und der Bereich innerhalb der rot gestrichelten Linien den *Lane*-Bereich der DSTMap. Somit ist der Bereich innerhalb der gelb gestrichelten Linien der Bereich mit *falsch-negativer* Klassifizierung und der Bereich innerhalb der schwarz-gestrichelten Linien ist *falsch-positiv* klassifiziert. *Richtig-positiv* klassifiziert ist der Bereich innerhalb der rot gestrichelten Linien abzüglich des Bereichs innerhalb der schwarz gestrichelten Linien. Alle übrigen Bereiche sind *richtig-negativ* klassifiziert.

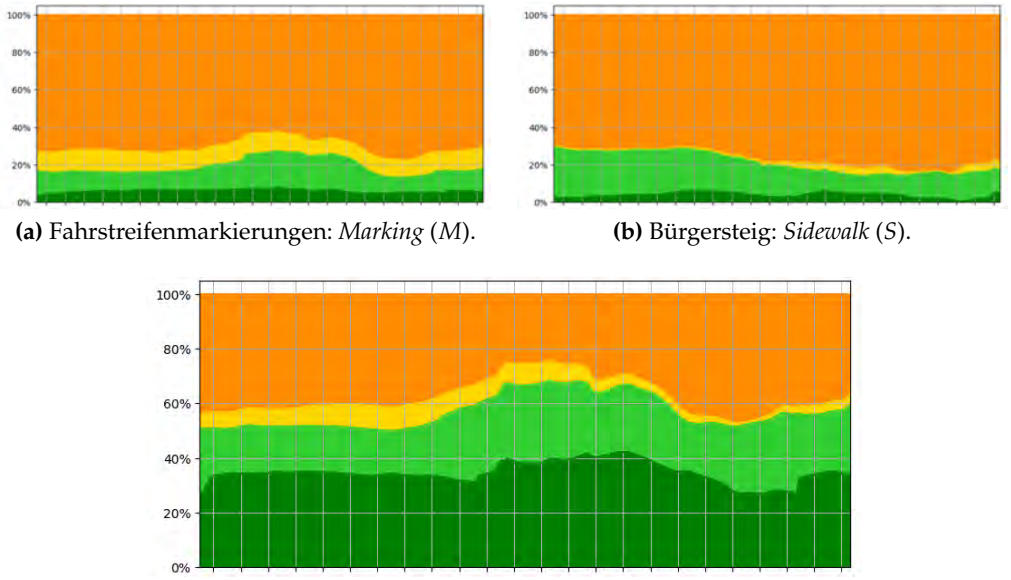
Gebäude - nicht in der verwendeten HD-Karte und damit auch nicht in der HD-Map enthalten. Daher kann keine sinnvolle Auswertung der Zellen vorgenommen werden, die als statische Hindernisse klassifiziert worden sind (Hypothese *O*: *Obstacle*). Für die übrigen 3 Hypothesen des 'Frame of Discernment' kann jeweils eine Wahrheitsmatrix berechnet werden.

Für eine exemplarische Auswertung wurde eine kurze Sequenz einer Messfahrt ausgewählt, die größtenteils innerstädtische Verkehrssituationen beinhaltet. Abb. 5.5 zeigt die Ergebnisse der Auswertung in Form von Wahrheitsmatrizen über einen kurzen zeitlichen Verlauf.

Die Qualität der Fahrstreifenmarkierungen in der DSTMap wird fast ausschließlich durch die erkannten Fahrstreifenmarkierungslinien beeinflusst. Im vorliegenden Fall stammen diese direkt von einem Kamerasystem eines Fremdherstellers und ließen sich nachträglich kaum verbessern. Darüber hinaus ist dies der einzige im Fahrzeug verbaute Sensor, der in der Lage ist, Fahrstreifenmarkierungslinien zu erkennen. Für eine Erhöhung der Qualität bei der Erkennung müsste daher zukünftig ein weiterer Sensor eingebaut werden oder die bildbasierte semantische Segmentierung (Kap. 2) insoweit erweitert werden, dass sie Fahrstreifenmarkierungslinien erkennen kann.

Bei der Klassifizierung von Bürgersteigen wird deutlich, dass die Erkennung dieser Bereiche noch nicht durchgehend zuverlässig funktioniert:

In vielen Situationen werden Bürgersteige zwar gut erkannt, wie die Ergebnisse der vorangegangenen Kapitel zeigen. Insbesondere in Situationen, in denen der Bürgersteig vollständig frei von Hindernissen und parkenden Fahrzeugen ist,



(a) Fahrstreifenmarkierungen: *Marking* (M).

(b) Bürgersteig: *Sidewalk* (S).

(c) Fahrstreifen: *Lane* (L).

Abbildung 5.6: Wahrheitsmatrizen für die drei Hypothesen *Marking*, *Sidewalk* und *Lane* im zeitlichen Verlauf. Dunkelgrün: *Richtig-positiv*, hellgrün: *Falsch-negativ*, gelb: *Falsch-positiv*, orange: *Richtig-negativ*.

werden die entsprechenden Zellen in der DSTMap richtig klassifiziert. Problematisch bei der Klassifizierung von Bürgersteigen ist aber einerseits die permanente Verdeckung großer Bereiche durch parkende Fahrzeuge und andererseits der teilweise große Abstand vom Bürgersteig zur Fahrbahn durch Grünstreifen oder Fahrradwege. Insbesondere die häufig vorkommenden Verdeckungen durch parkende Fahrzeuge sorgen für die hohe Zahl an falsch-negativ klassifizierten Zellen, da diese in der DSTMap als statisches Hindernis (Hypothese *Obstacle*) und nicht als Bürgersteig klassifiziert werden. Dies verschlechtert naturgemäß die Bewertung der DSTMap massiv, hat jedoch keine Auswirkungen auf das generierte Straßenmodell, da dies sowohl statische Hindernisse als auch Bürgersteige korrekterweise als Fahrstreifengrenze ansieht.

Die hohe Rate an falsch-negativ klassifizierten Zellen ist bedingt durch eine häufig mangelnde Vorausschau in innerstädtischen Situationen. Oftmals gelingt bei der bildbasierten semantischen Segmentierung zwar eine Klassifizierung als Fahrbahn, allerdings werden keinerlei Fahrstreifenmarkierungslinien erkannt. Dadurch werden weite Bereiche der Fahrstreifen zwar korrekt als Fahrbahn (Hypothese $\{M, L\}$) klassifiziert, die detailliertere Klassifizierung und Unterscheidung in *Lane* (Hypothese L) und *Marking* (Hypothese M) gelingt jedoch nicht.

5.3 Bewertung des Straßenmodells

Die Bewertung und Evaluierung des Straßenmodells erfolgt durch einen Vergleich der einzelnen Bestandteile des Modells mit denen in der HD-Karte. Dies sind neben der Fahrstreifenmittenlinie die jeweils linken und rechten Fahrstreifengrenzen.

Mit Hilfe der hochgenauen Lokalisierung, wie sie auch im letzten Abschnitt für die Bewertung der DSTMap verwendet worden ist, lässt sich die exakte Position des Ego-Fahrzeugs in den globalen Koordinaten ermitteln, in denen auch die HD-Karte referenziert ist. Somit lässt sich das erzeugte Straßenmodell in dasselbe Koordinatensystem transformieren, wie die HD-Karte. Anschließend wird der Ausschnitt der HD-Karte bestimmt, der zur Evaluierung benötigt wird. Eine einfache Methode ist die Wahl eines Kreises mit der Position des Ego-Fahrzeugs als Mittelpunkt und einem Radius, der der Länge des längsten Fahrstreifens des Straßenmodells entspricht.

Für einen Vergleich der einzelnen Elemente wird nun nur noch die Zuordnung der Elemente des Straßenmodells zu den Elementen in der HD-Karte erforderlich. Für den Fahrstreifen, auf dem sich zum aktuellen Zeitpunkt das Ego-Fahrzeug befindet, ist die Zuordnung offensichtlich, da über die Fahrzeugposition der eigene Fahrstreifen sowohl im erzeugten Straßenmodell als auch in der HD-Karte direkt ersichtlich ist. Für alle anderen Fahrstreifen muss die Zuordnung separat bestimmt werden, sofern sie sich nicht durch eine direkte Nachbarschaftsbeziehung (Fahrstreifenparallelität) mit dem Ego-Fahrstreifen ergibt. Die Zuordnung kann beispielsweise durch die Berechnung und den anschließenden Vergleich der geometrischen Mitten aller Fahrstreifen erfolgen.

Für die Bewertung der einzelnen Fahrstreifen werden jeweils deren Mittenlinie sowie die linke und rechte Fahrstreifengrenze mit den Daten aus der HD-Karte verglichen. Die zu vergleichenden Elemente sind in allen Fällen Polylinien, d.h. sie bilden eine Menge an zusammengehörigen Punkten, die jeweils eine Fahrstreifenmitte oder -grenze bilden.

Für alle Elemente (Fahrstreifenmitten und -grenzen) werden nun jeweils punktweise die Differenzen berechnet. Da die Anzahl der Punkte variiert und die Punktdichte ('sampling density') in der HD-Karte sehr hoch ist, kann relativ einfach jeder Punkt aus dem Straßenmodell dem nächstgelegenen Punkt in der HD-Karte zugeordnet werden und anschließend die Differenz in Ortskoordinaten berechnet werden. Die Menge aller Differenzen eines Elements kann als Fehlermaß des Elements angesehen werden. Zusätzlich lassen sich Mittelwert und Varianz bestimmen.

Ähnlich wie bei der Auswertung der DSTMap wurde eine kurze Sequenz einer Messfahrt ausgewählt und ausgewertet, um die Methoden und Verfahren der Auswertung zu demonstrieren.

Auf eine Bewertung der generellen Verfügbarkeit des Straßenmodells wird an dieser Stelle verzichtet, da die Verfügbarkeit maßgeblich durch die Qualität der Sensorik bzw. der Beobachtbarkeit der Umgebung durch die Sensorik bestimmt wird. Die für diese Arbeit verwendete Sensorik war bereits zu Beginn an festgelegt und konnte im weiteren Verlauf nicht geändert werden. Abgesehen davon ist die Beobachtbarkeit der Umgebung und die prinzipielle Erfassungsqualität der Sensorik ein separates, umfangreiches Thema und daher nicht Bestandteil der vorliegenden Arbeit.

Abb. 5.7 und 5.8 zeigen jeweils die Fehler in Längs- (x-) und Querrichtung (y-Koordinaten) der Punkte des Straßenmodells im Ego-Fahrzeug-Koordinatensystem über die Zeit. Die blauen Punkte stellen die Mittelwerte der Fehler (Differenzen) aller Punkte eines Elements dar, die roten Linien geben die Schwankungsbreite wieder.

Die Auswertung zeigt, dass die Gesamtqualität im Mittel relativ hoch ist. Allerdings gibt es immer wieder Augenblicke, in denen Teile des generierte Straßenmodell höhere Fehler aufweisen. Kleinere Längsfehler sind nicht so gravierend, da sie keinen negativen Einfluss auf die weitere Fahrstrategie und Manöverplanung haben. Im Gegensatz dazu sind die Fehler in Querrichtung entscheidend für die Gesamtqualität. Hier sind insbesondere die Standardabweichungen noch zu hoch für einen zuverlässigen Einsatz des Straßenmodells für das automatisierte Fahren.

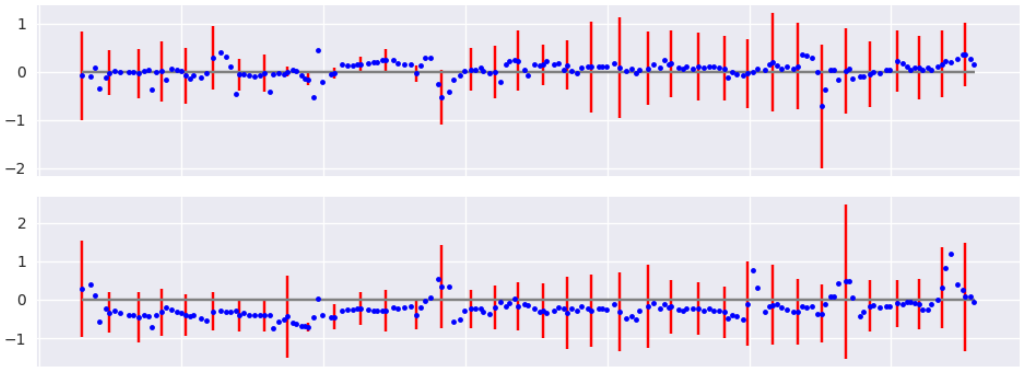


Abbildung 5.7: Durchschnittlicher Fehler des Ego-Fahrstreifens entlang der Fahrzeuglängsachse in [m] im zeitlichen Verlauf. Oben: Linke Fahrstreifengrenze; unten: Rechte Fahrstreifengrenze.

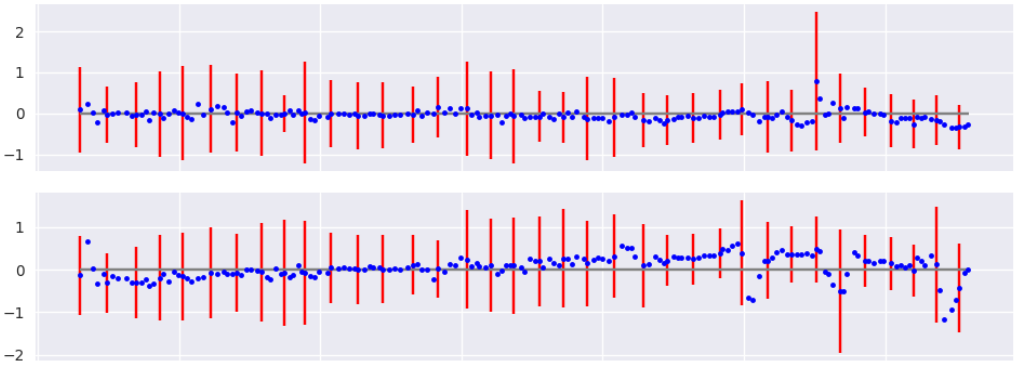


Abbildung 5.8: Durchschnittlicher Fehler des Ego-Fahrstreifens entlang der Fahrzeugquerachse in [m] im zeitlichen Verlauf. Oben: Linke Fahrstreifengrenze; unten: Rechte Fahrstreifengrenze.

5.4 Zusammenfassung

In diesem Kapitel wurden Methoden und Verfahren vorgestellt, anhand derer die DSTMap aus Kapitel 2 und das Straßenmodell aus Kapitel 4 evaluiert und quantitativ bewertet werden können.

Für die Messdatengewinnung wurde ein modifiziertes Fahrzeug der BMW 7er Serie verwendet, das mit umfangreicher Sensorik für die Umfelderkennung ausgestattet ist. Darüber hinaus verfügt es über ein elektronisch steuerbares Antriebs- und Lenksystem, um automatisierte Testfahrten durchführen zu können. Die Messdaten standen sowohl „live" im Fahrzeug während der Fahrt als auch in abgespeicherter Form für die nachträgliche Auswertung zur Verfügung.

Neben den Messdaten ist für eine quantitative Auswertung die Verfügbarkeit von Referenzdaten, sog. „ground truth"-Daten, erforderlich. Im vorliegenden Fall wurde eine hochgenaue Straßenkarte eines öffentlichen Straßenabschnitts verwendet, um sowohl die DSTMap als auch das generierte Straßenmodell bewerten zu können. Aufgrund der sehr eingeschränkten Verfügbarkeit von Referenzdaten liegt der Fokus dieses Kapitels auf der Vorstellung und Beschreibung der Methodik, mit der sowohl die DSTMap als auch das Straßenmodell quantitativ bewertet werden können, sobald entsprechende Referenzdaten zur Verfügung stehen. Dennoch wurden die DSTMap und das Straßenmodell anhand einer kurzen Messdatensequenz quantitativ bewertet, um die Ergebnisse der Auswertungsmethodik zu demonstrieren. Die Auswertung zeigt, dass die Qualität für ein zuverlässiges automatisiertes Fahren ohne Karte noch nicht hoch genug ist. Jedoch ist es durchaus schon möglich streckenweise nur aus Sensormessdaten ein Straßenmodell zu generieren, das zur Steuerung des Fahrzeugs verwendet werden kann.

Kapitel 6

Fazit

Aktuelle Systeme und Verfahren des automatisierten Fahrens, wie sie in Kap. 1 vorgestellt worden sind, stützen sich derzeit auf ein Umfeldmodell aus einer hochgenauen Karte in Kombination mit einer hochgenauen Lokalisierung. Ein solches kartenbasiertes Umfeldmodell birgt jedoch gewisse Nachteile, die einer zuverlässigen und jederzeit verfügbaren automatisierten Fahrfunktion entgegenstehen. Neben einer erforderlichen flächendeckenden Verfügbarkeit des Kartenmaterials und einer fehlenden oder unpräzisen Lokalisierung ist die Existenz von Kartenfehlern eines der zentralen Probleme. Bei Kartenfehlern stimmt der kartierte Streckenverlauf nicht mehr mit dem realen überein. Die häufigste Ursache für diese Art Fehler sind Baustellen und permanente Streckenveränderungen.

Daher wurden in der vorliegenden Arbeit Methoden und Verfahren für die Erstellung eines sensorbasierten Umfeldmodells vorgestellt. Dabei werden die Messdaten der im Fahrzeug verbauten Sensoren dazu verwendet, ein möglichst genaues Abbild der aktuellen Fahrzeugumgebung zu erzeugen.

Hierfür wurde ein neuartiger gridbasierter Ansatz - die DSTMap - entwickelt, der die statische Straßeninfrastruktur repräsentiert. Gridbasierte Ansätze erlauben einerseits eine flexible Modellierung der Umgebung und bieten andererseits eine gute Abstraktionsebene für die Schnittstellen der Sensorik. Für die DSTMap wurde die 'Dempster-Shafer Theory of Evidence' verwendet und ein neues 'frame of discernment' definiert, das Hypothesen für die Straßeninfrastruktur enthält. Anhand von den im Fahrzeug zur Verfügung stehenden Sensormessdaten wurde die Erstellung der DSTMap erläutert. Darüber hinaus wurde die DSTMap mit einer klassischen Belegungskarte verglichen. Anhand von verschiedenen realen Situationen wurden die Vorteile der DSTMap demonstriert und diskutiert.

Anschließend wurde ein neues Verfahren vorgestellt, um aus einem gridbasierten Umfeldmodell potentielle Fahrstreifen zu extrahieren. Das Verfahren erzeugt mit Hilfe der DSTMap eine Baumstruktur von kinematisch fahrbaren Pfaden. Jeder dieser Pfade repräsentiert dabei einen Abschnitt eines potentiellen Fahrstreifens und besitzt jeweils eine linke und rechte Fahrstreifengrenze. In Anlehnung an den Pfadplanungsansatz aus [175] wurde dazu eine iterative Pfadplanung mit explorativem Charakter entwickelt, um kinematisch fahrbare Pfade und damit einhergehend auch potentielle Fahrstreifen zu finden. Mit Hilfe des in dieser Arbeit definierten 'frame of discernment' konnte sowohl die Kostenberechnung als auch die Extraktion von Fahrstreifengrenzen direkt anhand der DSTMap erfolgen. Durch die Verwendung verschiedener 'beliefs' und der 'plausibility' bietet sie die nötige Flexibilität für die Berechnung aller Messgrößen, die für die Extraktion von Fahrstreifen erforderlich sind.

Da die Ergebnisse aus der Pfadplanung nicht dazu verwendet werden können, ein Fahrzeug automatisiert fahren zu lassen, wurde in Kapitel 4 eine Methode beschrieben, um aus der Baumstruktur ein geeignetes Straßenmodell für die Bewegungsplanung des Fahrzeugs zu erzeugen. Dabei können beliebige Fahrstreifengeometrien aus der Pfadplanung modelliert werden. Bei parallel verlaufenden Fahrstreifen wird die Nachbarschaftsbeziehung korrekt abgebildet und für beide Fahrstreifen eine **gemeinsame** Fahrstreifengrenze berechnet.

Schlussendlich wurden sowohl die DSTMap als auch das Straßenmodell bewertet und anhand von Referenzdaten quantitativ evaluiert.

Im Rahmen der Arbeit entstanden unter anderem 4 Publikationen (Erstautorenschaft), in denen die Forschungsergebnisse veröffentlicht wurden [1, 2, 3, 4].

Die Ergebnisse der vorliegenden Arbeit zeigen, dass

- es gelungen ist, Methoden und Verfahren zu entwickeln, die aus realen Sensormessdaten ein Straßenmodell für das automatisierte Fahren erzeugen.
- beliebige Fahrstreifengeometrien aus der DSTMap extrahiert und unter Berücksichtigung von Fahrstreifen-Parallelität und Abzweigungen modelliert werden können.
- die DSTMap eine hervorragende sensorunabhängige Methode darstellt, das aktuelle Fahrzeugumfeld auch hinsichtlich der Straßeninfrastruktur abzubilden, und darüber hinaus sehr flexibel und einfach erweiterbar ist.
- die DSTMap aufgrund ihrer Flexibilität sehr gut als Hindernis- und Kostenkarte verwendet werden kann. Die Verwendung der einzelnen Elemente wie bspw. der 'belief' und die 'plausibility' sind für die Berechnung

von Kosten vorteilhaft. Aufgrund der Flexibilität lässt sich die DSTMap aber auch für andere Zwecke einsetzen (bspw. zur Lokalisierung oder für eine sensorbasierte Validierung einer hochgenauen Karte).

- eine Pfadplanung nicht nur für die Bewegungsplanung, sondern auch für die Extraktion von Informationen aus einem gridbasierten Umfeldmodell verwendet werden kann. Die entwickelte Methode der iterativen Planung ist hierbei für die effiziente Durchführung der Pfadplanung von großem Vorteil.
- die Geometrien von Fahrstreifen inkl. Topologie, Abzweigungen und Fahrstreifenparallelität aus Sensormessdaten extrahiert und korrekt modelliert werden können.

Die vorliegende Arbeit stellt somit einen wichtigen Schritt in Richtung eines sensorbasierten Umfeldmodells dar. Allerdings erreichen die Ergebnisse bis dato noch nicht die erforderliche Qualität, die für ein zuverlässiges automatisiertes Fahren ohne den Einsatz einer hochgenauen Karte notwendig ist.

Im nächsten Abschnitt werden daher einige Ideen für eine Weiterführung und Weiterentwicklung der hier erarbeiteten Methoden präsentiert.

6.1 Ausblick

Im Zuge der technologischen Weiterentwicklung im Bereich der Sensorik und der semantischen Segmentierung von Kamerabildern ist zu erwarten, dass sich die Qualität der Ergebnisse in naher Zukunft deutlich steigern lässt. Insbesondere der Einsatz von Laserscannern mit einer Vielzahl an vertikalen Ebenen in Verbindung mit der semantischen Segmentierung von Kamerabildern (statt einer Stereokamera) wird dazu führen, dass die DSTMap das Fahrzeugumfeld sehr präzise wird abbilden können. Erste Versuche dieser Art sind im Rahmen der vorliegenden Arbeit erfolgt und sind vielversprechend verlaufen. Die entwickelte DSTMap bringt bereits alle Voraussetzungen mit, um auch in schwierigen Situationen zuverlässig und mit hoher Genauigkeit die statische Straßeninfrastruktur erfassen zu können.

Für die Extraktion von potentiellen Fahrstreifen bietet es sich an, neben der hier verwendeten Kombination aus A* und RRT weitere Planungsalgorithmen zu implementieren und gegeneinander zu vergleichen.

Darüber hinaus ist es erforderlich, durch die Durchführung und Auswertung von langen und umfangreichen Messfahrten statistisch signifikante Ergebnisse

zu erzeugen. Dies ist allerdings erst dann sinnvoll möglich, wenn auch in entsprechendem Umfang Referenzdaten in Form einer großflächigen hochgenauen Karte zur Verfügung stehen.

Die Entwicklung eines sensorbasierten Umfeldmodells wird zukünftig verstärkt in den Fokus des automatisierten Fahrens gelangen. Die ersten Einsatzmöglichkeiten werden vermutlich im Bereich einer sensorbasierten Kartenvalidierung liegen und erst später wird in Abhängigkeit von der Zuverlässigkeit der Straßenmodellschätzung auch das automatisierte Fahren mit einem sensorbasierten statt einem kartenbasierten Straßenmodell möglich sein.

Literatur

[1] J. Thomas u. a. „Grid-based online road model estimation for advanced driver assistance systems“. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2015, S. 71–76. DOI: 10.1109/IVS.2015.7225665.

[2] J. Thomas und R. Rojas. „Sensor-based road model estimation for autonomous driving“. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, S. 1764–1769.

[3] J. Thomas u. a. „Semantic Grid-Based Road Model Estimation for Autonomous Driving“. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2019, S. 2329–2336. DOI: 10.1109/IVS.2019.8813790.

[4] J. Thomas u. a. „Online Road Model Generation From Evidential Semantic Grids“. In: *2020 23th International IEEE Conference on Intelligent Transportation Systems*. 2020.

[5] Klaus Kompaß. „Fahrerassistenzsysteme der Zukunft – auf dem Weg zum autonomen Pkw?“ In: *Forschung für das Auto von Morgen: Aus Tradition entsteht Zukunft*. Hrsg. von Volker Schindler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 261–285. ISBN: 978-3-540-74151-0. DOI: 10.1007/978-3-540-74151-0_7. URL: https://doi.org/10.1007/978-3-540-74151-0_7.

[6] Robert Bosch. *Safety, Comfort and Convenience Systems*. Robert Bosch GmbH, 2006. ISBN: 9780837613918.

[7] *Convention on Road Traffic*. 1968.

[8] *Report of the sixty-eighth sessionof the Working Party on Road Traffic Safety*. 2014. URL: http://www.unece.org/fileadmin/DAM/trans/doc/2014/wp1/ECE-TRANS-WP1-145e.pdf.

[9] *Rechtsfolgen zunehmender Fahrzeugautomatisierung*. Berichte der Bundesanstalt für Straßenwesen, Heft F 83, 2012. 2012.

[10] *Nevada Revised Statutes Chapter 482A*. 2012.

[11] *California Senate Bill 1298*. 2012.

[12] *Florida Committee Substitute House Bill 1207*. 2012.

[13] *Michigan Senate Bill 0169*. 2013.

[14] *An Act amending Title 75 (Vehicles) of the Pennsylvania Consolidated Statutes, in operation of vehicles, providing for highly automated vehicles and platooning testing.* 2017.

[15] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Technical Report SAE-J 3016. 2016.

[16] A. Zapp. *Automatische Straßenfahrzeugführung durch Rechnersehen*. 1988.

[17] E. D. Dickmanns u. a. „The seeing passenger car 'VaMoRs-P'“. In: *Intelligent Vehicles '94 Symposium, Proceedings of the*. Okt. 1994, S. 68–73. DOI: 10.1109/IVS.1994.639472.

[18] *DARPA Grand Challenge 2004*. Accessed: 2017-11-05. URL: http://archive.darpa.mil/grandchallenge04/index.htm.

[19] *DARPA Grand Challenge 2005*. Accessed: 2017-11-05. URL: http://archive.darpa.mil/grandchallenge05/.

[20] *DARPA Urban Challenge*. Accessed: 2017-11-05. URL: http://archive.darpa.mil/grandchallenge.

[21] Sebastian Thrun u. a. „Stanley: The Robot That Won the DARPA Grand Challenge: Research Articles“. In: *J. Robot. Syst.* 23.9 (Sep. 2006), S. 661–692. ISSN: 0741-2223. DOI: 10.1002/rob.v23:9. URL: http://dx.doi.org/10.1002/rob.v23:9.

[22] Chris Urmson u. a. „Autonomous driving in urban environments: Boss and the Urban Challenge“. In: *Journal of Field Robotics* 25.8 (2008), S. 425–466. ISSN: 1556-4967. DOI: 10.1002/rob.20255. URL: http://dx.doi.org/10.1002/rob.20255.

[23] Soeren Kammel u. a. „Team AnnieWAYs autonomous system for the 2007 DARPA Urban Challenge“. In: *Journal of Field Robotics* 25.9 (2008), S. 615–639. ISSN: 1556-4967. DOI: 10.1002/rob.20252. URL: http://dx.doi.org/10.1002/rob.20252.

[24] Christian Berger und Bernhard Rumpe. „Autonomes Fahren: Erkenntnisse aus der DARPA Urban Challenge (Autonomous Driving: Insights from the DARPA Urban Challenge)“. In: 50 (Jan. 2008), S. 258–264.

[25] *Technische Universität Braunschweig: Projekt Stadtpilot*. Accessed: 2017-11-08. URL: https://www.tu-braunschweig.de/iff/forschung/projekte.

[26] J. M. Wille, F. Saust und M. Maurer. „Stadtpilot: Driving autonomously on Braunschweig's inner ring road“. In: *2010 IEEE Intelligent Vehicles Symposium*. Juni 2010, S. 506–511. DOI: 10.1109/IVS.2010.5548034.

[27] F. Saust u. a. „Autonomous Vehicle Guidance on Braunschweig's inner ring road within the Stadtpilot Project“. In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2011, S. 169–174. DOI: 10.1109/IVS.2011.5940568.

[28] *Freie Universität Berlin*. Accessed: 2017-11-09. URL: http://www.fu-berlin.de/presse/informationen/fup/2012/fup_12_286/.

[29] *Freie Universität Berlin*. Accessed: 2017-11-09. URL: http://www.fu-berlin.de/presse/informationen/fup/2015/fup_15_319-autonom-in-mexiko/index.html.

[30] *Freie Universität Berlin*. Accessed: 2017-11-09. URL: http://www.fu-berlin.de/presse/informationen/fup/2015/fup_15_131-autonom-in-zuerich-unterwegs/index.html.

[31] *Freie Universität Berlin: AutoNOMOS Labs*. Accessed: 2017-11-09. URL: http://autonomos-labs.com/.

[32] F. Kunz u. a. „Autonomous driving at Ulm University: A modular, robust, and sensor-independent fusion approach“. In: *2015 IEEE Intelligent*

Vehicles Symposium (IV). Juni 2015, S. 666–673. DOI: 10.1109/IVS.2015.7225761.

[33] *Audi Media Center*. Accessed: 2017-11-09. URL: https://www.audi-mediacenter.com/de/pilotiertes-fahren-3651.

[34] *Bosch Group: Cars that Drive Themselves - Highway Pilot Technically Viable in Five Years*. Accessed: 2017-11-09. URL: http://www.bosch-presse.de/.

[35] *BMW Group PressClub Global: Ready for Takeover!* Accessed: 2017-11-09. URL: http://www.press.bmwgroup.com.

[36] M. Aeberhard u. a. „Experience, Results and Lessons Learned from Automated Driving on Germany's Highways". In: *IEEE Intelligent Transportation Systems Magazine* 7.1 (Spring 2015), S. 42–57. ISSN: 1939-1390. DOI: 10.1109/MITS.2014.2360306.

[37] M. Ardelt, C. Coester und N. Kaempchen. „Highly Automated Driving on Freeways in Real Traffic Using a Probabilistic Framework". In: *IEEE Transactions on Intelligent Transportation Systems* 13.4 (Dez. 2012), S. 1576–1585. ISSN: 1524-9050. DOI: 10.1109/TITS.2012.2196273.

[38] Nico Kaempchen u. a. „Technologies for highly automated driving on highways". In: *ATZ worldwide* 114 (Juni 2012).

[39] S Rauch u. a. „Autonomes Fahren auf der Autobahn - eine Potentialstudie für zukünftige Fahrerassistenzsysteme." In: *5. Tagung Fahrerassistenz* (2012).

[40] *BMW Group PressClub Global*. Accessed: 2017-11-09. URL: http://www.press.bmwgroup.com/.

[41] J. Ziegler u. a. „Making Bertha Drive - An Autonomous Journey on a Historic Route". In: *IEEE Intelligent Transportation Systems Magazine* 6.2 (Summer 2014), S. 8–20. ISSN: 1939-1390. DOI: 10.1109/MITS.2014.2306552.

[42] U. Franke u. a. „Making Bertha See". In: *2013 IEEE International Conference on Computer Vision Workshops*. Dez. 2013, S. 214–221. DOI: 10.1109/ICCVW.2013.36.

[43] *Mercedes-Benz*. Accessed: 2017-11-09. URL: https://www.mercedes-benz.com/.

[44] *on the road to fully self-driving: Waymo Safety Report*. Accessed: 2017-11-09. URL: https://storage.googleapis.com/sdc-prod/v1/safety-report/waymo-safety-report-2017-10.pdf.

[45] *Waymo Press*. Accessed: 2017-11-09. URL: https://waymo.com/press/.

[46] Peter Waldmann und Daniel Niehues. „Der BMW Track Trainer - automatisiertes Fahren im Grenzbereich auf der Nürburgring Nordschleife". In: *4. Tagung Sicherheit durch Fahrerassistenz*. Apr. 2010.

[47] *BMW AG: BMW at the Consumer Electronics Show (CES) in Las Vegas 2014*. Accessed: 2017-11-09. URL: http://www.press.bmwgroup.com/.

[48] *AUDI AG: Audi RS 7 piloted driving concept (2014)*. Accessed: 2017-11-09. URL: https://www.audi-mediacenter.com/de/pressemitteilungen/audi-rs-7-piloted-driving-concept-faehrt-selbstaendig-rekordzeit-auf-spanischer-rennstrecke-5161.

[49] *Pike's Peak International Hill Climb*. Accessed: 2017-11-09. URL: http://www.ppihc.com/.

[50] *Stanford's robotic Audi to brave Pikes Peak without a driver*. Accessed: 2017-11-09. URL: https://news.stanford.edu/news/2010/february1/shelley-pikes-peak-020310.html.

[51] Martin Friedl, Adrian Hupka und Georg Tanzmeister. „Vollautomatisiertes Valet Parking: Funktions- und Planungsarchitektur". In: *10. Workshop Fahrerassistenzsysteme*. 2015, S. 1.

[52] R. Kümmerle u. a. „Autonomous driving in a multi-level parking structure". In: *2009 IEEE International Conference on Robotics and Automation*. Mai 2009, S. 3395–3400. DOI: 10.1109/ROBOT.2009.5152365.

[53] H. Banzhaf u. a. „The future of parking: A survey on automated valet parking with an outlook on high density parking". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2017, S. 1827–1834. DOI: 10.1109/IVS.2017.7995971.

[54] S. Klemm u. a. „Autonomous multi-story navigation for valet parking". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2016, S. 1126–1133. DOI: 10.1109/ITSC.2016.7795698.

[55] S. Klaudt, A. Zlocki und L. Eckstein. „A-priori map information and path planning for automated valet-parking". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2017, S. 1770–1775. DOI: 10.1109/IVS.2017.7995963.

[56] *Bosch und Daimler zeigen fahrerloses Parken im realen Verkehr: Weltpremiere im Parkhaus des Mercedes-Benz Museums*. Accessed: 2017-11-09. URL: http://media.daimler.com.

[57] *DB stellt autonomes Bus-Shuttle für Bad Birnbach vor.* Accessed: 2017-11-09. URL: http://www.deutschebahn.com/presse/muenchen/de/aktuell/presseinformationen/.

[58] Michael Himmelsbach u. a. „Autonomous Off-Road Navigation for MuCAR-3 - Improving the Tentacles Approach: Integral Structures for Sensing and Motion". In: 25 (Mai 2011), S. 145–149.

[59] B. C. Heinrich, T. Luettel und H. J. Wuensche. „A new control architecture for MuCAR". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2017, S. 1861–1866. DOI: 10.1109/IVS.2017.7995976.

[60] Hanno Jaspers, Dennis Fassbender und Hans-Joachim Wuensche. „Visual Navigation with Efficient ConvNet Features". In: *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*. Vancouver, BC, Canada, Sep. 2017.

[61] Dennis Fassbender u. a. „An Optimization Approach to Trajectory Generation for Autonomous Vehicle Following“. In: *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*. Vancouver, BC, Canada, Sep. 2017.

[62] A. Muller u. a. „A model-based object following system“. In: *2009 IEEE Intelligent Vehicles Symposium*. Juni 2009, S. 242–249. DOI: 10.1109/IVS.2009.5164285.

[63] C. Fries und H. J. Wuensche. „Real-time unsupervised feature model generation for a vehicle following system“. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2016, S. 2450–2455. DOI: 10.1109/ITSC.2016.7795950.

[64] M. Manz u. a. „Monocular model-based 3D vehicle tracking for autonomous vehicles in unstructured environment“. In: *2011 IEEE International Conference on Robotics and Automation*. Mai 2011, S. 2465–2471. DOI: 10.1109/ICRA.2011.5979581.

[65] M. Manz u. a. „Detection and tracking of road networks in rural terrain by fusing vision and LIDAR“. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sep. 2011, S. 4562–4568. DOI: 10.1109/IROS.2011.6094559.

[66] M. Manz, F. von Hundelshausen und H. J. Wuensche. „A hybrid estimation approach for autonomous dirt road following using multiple clothoid segments“. In: *2010 IEEE International Conference on Robotics and Automation*. Mai 2010, S. 2410–2415. DOI: 10.1109/ROBOT.2010.5509983.

[67] Hermann Winner. „Radarsensorik“. In: *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. Hrsg. von Hermann Winner, Stephan Hakuli und Gabriele Wolf. Wiesbaden: Vieweg+Teubner, 2009, S. 123–171. ISBN: 978-3-8348-9977-4. DOI: 10.1007/978-3-8348-9977-4_13. URL: https://doi.org/10.1007/978-3-8348-9977-4_13.

[68] Christoph Stiller, Alexander Bachmann und Christian Duchow. „Maschinelles Sehen“. In: *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. Hrsg. von Hermann Winner, Stephan Hakuli und Gabriele Wolf. Wiesbaden: Vieweg+Teubner Verlag, 2012, S. 198–222. ISBN: 978-3-8348-8619-4. DOI: 10.1007/978-3-8348-8619-4_16. URL: https://doi.org/10.1007/978-3-8348-8619-4_16.

[69] Martin Noll und Peter Rapps. „Ultraschallsensorik“. In: *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. Hrsg. von Hermann Winner u. a. Wiesbaden: Springer Fachmedien Wiesbaden, 2015, S. 243–258. ISBN: 978-3-658-05734-3. DOI: 10.1007/978-3-658-05734-3_16. URL: https://doi.org/10.1007/978-3-658-05734-3_16.

[70] Georg Geduld. „Lidarsensorik“. In: *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. Hrsg. von Hermann Winner, Stephan Hakuli und Gabriele Wolf. Wiesbaden: Vieweg+Teubner Verlag, 2012, S. 172–185. ISBN: 978-3-8348-8619-4. DOI: 10.1007/978-3-8348-8619-4_14. URL: https://doi.org/10.1007/978-3-8348-8619-4_14.

[71] Nico Kaempchen u. a. „Sensor Fusion for Multiple Automotive Active Safety and Comfort Applications“. In: *Advanced Microsystems for Automotive Applications 2004*. Hrsg. von Jürgen Valldorf und Wolfgang Gessner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, S. 137–163. ISBN: 978-3-540-76989-7. DOI: 10.1007/978-3-540-76989-7_11. URL: https://doi.org/10.1007/978-3-540-76989-7_11.

[72] M. Darms und H. Winner. „A modular system architecture for sensor data processing of ADAS applications“. In: *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* Juni 2005, S. 729–734. DOI: 10.1109/IVS.2005.1505190.

[73] Heiko G. Seif und Xiaolong Hu. „Autonomous Driving in the iCity - HD Maps as a Key Challenge of the Automotive Industry“. In: *Engineering* 2.2 (2016), S. 159–162. ISSN: 2095-8099. DOI: https://doi.org/10.1016/J.ENG.2016.02.010. URL: http://www.sciencedirect.com/science/article/pii/S2095809916309432.

[74] *Mapping the way to autonomous driving - High-resolution maps from HERE*. Accessed: 2017-11-10. URL: https://www.daimler.com/innovation/autonomous-driving/mapping-the-way-to-autonomous-driving.html/.

[75] The Boston Consulting Group. *Revolution in the Driver's Seat*. Accessed: 2017-11-10. URL: http://img-stg.bcg.com/BCG-Revolution-in-the-Drivers-Seat-Apr-2015_tcm9-64351.pdf.

[76] S. Rauch u. a. „Hochgenaue Fahrzeugeigenlokalisierung und kollektives Erlernen hochgenauer digitaler Karten“. In: *Proc. of the AAET Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel*. 2012.

[77] E. B. Olson. „Real-time correlative scan matching“. In: *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*. Mai 2009, S. 4387–4393. DOI: 10.1109/ROBOT.2009.5152375.

[78] HERE 360: The official HERE blog. *HERE 360: The official HERE blog*. Accessed: 2017-11-21.

[79] NVidia Youtube Channel. *NVIDIA and TomTom: Mapping the Road Ahead*. Accessed: 2017-11-21. URL: https://www.youtube.com/watch?v=71SCDTZCJnA.

[80] BaSt Bundesanstalt für Straßenwesen. *MDM - Mobilitäts Daten Marktplatz*. Accessed: 2017-11-21. URL: http://www.bast.de/DE/Fahrzeugtechnik/Baustelleninformation/baustelleninformation_hidden_node.html.

[81] HERE 360: The official HERE blog. *HERE and Mobileye: crowd-sourced HD mapping for autonomous cars*. Accessed: 2017-11-11. URL: https://360.here.com/2016/12/30/here-and-mobileye-crowd-sourced-hd-mapping-for-autonomous-cars/.

[82] Qualcomm Press Release. *Qualcomm Drive Data Platform Powers TomTom's Plans to Crowdsource High-Definition Mapping Data for Autonomous Driving*. Accessed: 2017-11-11. URL: https://www.qualcomm.com/news/releases/2017/02/26/qualcomm-drive-data-platform-powers-tomtoms-plans-crowdsource-high.

[83] R. J. Radke u. a. „Image change detection algorithms: a systematic survey". In: *IEEE Transactions on Image Processing* 14.3 (März 2005), S. 294–307. ISSN: 1057-7149. DOI: 10.1109/TIP.2004.838698.

[84] L. Wellhausen u. a. „Reliable real-time change detection and mapping for 3D LiDARs". In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. Okt. 2017, S. 81–87. DOI: 10.1109/SSRR.2017.8088144.

[85] Tim Rakowski. „Informationstheoretische Änderungserkennung von hochgenauen Straßenmodellen als Grundlage für automatisierte Fahrfunktionen". Magisterarb. Berlin: FU Berlin, Arbeitsgruppe Intelligente Systeme und Robotik, 2013.

[86] Nico Kaempchen. „Feature-level fusion of laser scanner and video data for advanced driver assistance systems". Diss. Fakultät für Ingenieurswissenschaften und Informatik, Universität Ulm, 2007.

[87] Michael Aeberhard u. a. „Track-to-Track Fusion with Asynchronous Sensors Using Information Matrix Fusion for Surround Environment Perception". In: *IEEE Transactions on Intelligent Transportation Systems* 13.4 (Dez. 2012), S. 1717–1726.

[88] T. Scharwächter und U. Franke. „Low-level fusion of color, texture and depth for robust road scene understanding". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2015, S. 599–604. DOI: 10.1109/IVS.2015.7225750.

[89] D.A. Schwartz. „Clothoid road geometry unsuitable for sensor fusion clothoid parameter sloshing". In: *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*. Juni 2003, S. 484–488. DOI: 10.1109/IVS.2003.1212959.

[90] U. Franke H. Badino und D. Pfeiffer. „The Stixel World - A Compact Medium Level Representation of the 3D-World". In: *Proceedings of the 31st DAGM Symposium on Pattern Recognition (DAGM)*. 2009. URL: http://www.lelaps.de/papers/badino_dagm09.pdf.

[91] D. Pfeiffer und U. Franke. „Efficient representation of traffic scenes by means of dynamic stixels". In: *Intelligent Vehicles Symposium (IV), 2010 IEEE*. 2010, S. 217–224. DOI: 10.1109/IVS.2010.5548114.

[92] David Pfeiffer und Uwe Franke. „Towards a Global Optimal Multi-Layer Stixel Representation of Dense 3D Data“. In: *British Machine Vision Conference (BMVC)*. Dundee, Scotland, Aug. 2011.

[93] Lukas Schneider u. a. „Semantic Stixels: Depth is Not Enough“. In: ().

[94] Marius Cordts u. a. „Tree-Structured Models for Efficient Multi-Cue Scene Labeling“. In: *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on* (2016).

[95] A. Elfes. „A sonar-based mapping and navigation system“. In: *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*. Bd. 3. 1986, S. 1151–1156. DOI: 10.1109/ROBOT.1986.1087534.

[96] A. Elfes. „Using occupancy grids for mobile robot perception and navigation“. In: *Computer* 22.6 (1989), S. 46–57. ISSN: 0018-9162. DOI: 10.1109/2.30720.

[97] Sebastian Thrun, Wolfram Burgard und Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. The MIT Press, 2005. ISBN: 0262201623.

[98] M. Bahram, M. Aeberhard und D. Wollherr. „Please take over! An analysis and strategy for a driver take over request during autonomous driving“. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2015, S. 913–919. DOI: 10.1109/IVS.2015.7225801.

[99] Heidi Loose. „Dreidimensionale Straßenmodelle für Fahrerassistenzsysteme auf Landstraßen“. Diss. Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie, 2013.

[100] F. Homm, N. Kaempchen und D. Burschka. „Fusion of laserscannner and video based lanemarking detection for robust lateral vehicle control and lane change maneuvers“. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. Juni 2011, S. 969–974. DOI: 10.1109/IVS.2011.5940424.

[101] A. Abramov u. a. „A flexible modeling approach for robust multi-lane road estimation“. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2017, S. 1386–1392. DOI: 10.1109/IVS.2017.7995904.

[102] J. Beck und C. Stiller. „Non-parametric lane estimation in urban environments“. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. Juni 2014, S. 43–48. DOI: 10.1109/IVS.2014.6856551.

[103] H. Loose und U. Franke. „B-spline-based road model for 3d lane recognition“. In: *13th International IEEE Conference on Intelligent Transportation Systems*. Sep. 2010, S. 91–98. DOI: 10.1109/ITSC.2010.5624968.

[104] R. Matthaei, B. Lichte und M. Maurer. „Robust grid-based road detection for ADAS and autonomous vehicles in urban environments“. In: *Proceedings of the 16th International Conference on Information Fusion*. Juli 2013, S. 938–944.

[105] G. Tanzmeister u. a. „Road course estimation in unknown, structured environments“. In: *Intelligent Vehicles Symposium (IV), 2013 IEEE*. Juni 2013, S. 630–635. DOI: 10.1109/IVS.2013.6629537.

[106] M. Konrad, M. Szczot und K. Dietmayer. „Road course estimation in occupancy grids“. In: *Intelligent Vehicles Symposium (IV), 2010 IEEE*. Juni 2010, S. 412–417. DOI: 10.1109/IVS.2010.5548041.

[107] T. Weiherer, E. Bouzouraa und U. Hofmann. „A generic map based environment representation for driver assistance systems applied to detect convoy tracks“. In: *2012 15th International IEEE Conference on Intelligent Transportation Systems*. Sep. 2012, S. 691–696. DOI: 10.1109/ITSC.2012.6338658.

[108] S. Steyer, G. Tanzmeister und D. Wollherr. „Grid-Based Environment Estimation Using Evidential Mapping and Particle Tracking“. In: *IEEE Transactions on Intelligent Vehicles* 3.3 (Sep. 2018), S. 384–396. ISSN: 2379-8904. DOI: 10.1109/TIV.2018.2843130.

[109] M. Kurdej u. a. „Map-aided Fusion Using Evidential Grids for Mobile Perception in Urban Environment“. In: *Proceedings of the 2nd International Conference on Belief Functions*. Hrsg. von Marie-Hélène Denoeux Thierry; Masson. Bd. 164. Advances in Intelligent and Soft Computing. Compiègne, France: Springer, Mai 2012, S. 343–350. URL: http://hal.archives-ouvertes.fr/hal-00714465.

[110] Ralph Grewe. „Optimierung der Repräsentation von Occupancy-Grids für Fahrerassistenzsysteme“. Diss. Fachgebiet Fahrzeugtechnik, Technische Universität Darmstadt, 2014.

[111] M. Schmid. „Umgebungserfassung für Fahrerassistenzsysteme mit hierarchischen Belegungskarten“. Diss. Institut für Technik Autonomer Systeme, Universität der Bundeswehr München, 2012.

[112] D. Pagac, E. M. Nebot und H. Durrant-Whyte. „An evidential approach to map-building for autonomous vehicles“. In: *IEEE Transactions on Robotics and Automation* 14.4 (Aug. 1998), S. 623–629. ISSN: 1042-296X. DOI: 10.1109/70.704234.

[113] M. Schreier, V. Willert und J. Adamy. „Grid mapping in dynamic road environments: Classification of dynamic cell hypothesis via tracking“. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Mai 2014, S. 3995–4002. DOI: 10.1109/ICRA.2014.6907439.

[114] Trung-Dung Vu, O. Aycard und N. Appenrodt. „Online Localization and Mapping with Moving Object Tracking in Dynamic Outdoor Environments“. In: *Intelligent Vehicles Symposium, 2007 IEEE*. Juni 2007, S. 190–195. DOI: 10.1109/IVS.2007.4290113.

[115] J. Moras, V. Cherfaoui und P. Bonnifait. „Credibilist occupancy grids for vehicle perception in dynamic environments“. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. Mai 2011, S. 84–89. DOI: 10.1109/ICRA.2011.5980298.

[116] Julian Thomas. „Dynamische Belegungskarten“. Magisterarb. Frankfurt: Fachbereich Physik, Universität Frankfurt, 2013.

[117] Christophe Coué u. a. „Bayesian Occupancy Filtering for Multitarget Tracking: An Automotive Application“. In: *The International Journal of Robotics Research* 25.1 (2006), S. 19–30. DOI: 10.1177/0278364906061158.

[118] Christopher Tay u. a. „The Bayesian Occupation Filter“. In: *Probabilistic Reasoning and Decision Making in Sensory-Motor Systems*. Hrsg. von P. Bessière, C. Laugier und R. Siegwart. Bd. 46. Springer Tracts in Advanced Robotics Series. Springer, 2008. URL: http://hal.inria.fr/inria-00295084.

[119] Manuel Yguel u. a. *Velocity Estimation on the Bayesian Occupancy Filter for Multi-Target Tracking*. Techn. Ber. RR-5836. INRIA, 2006, S. 17. URL: http://hal.inria.fr/inria-00070190.

[120] R. Danescu, F. Oniga und S. Nedevschi. „Modeling and Tracking the Driving Environment With a Particle-Based Occupancy Grid“. In: *IEEE Transactions on Intelligent Transportation Systems* 12.4 (2011), S. 1331–1342. ISSN: 1524-9050. DOI: 10.1109/TITS.2011.2158097.

[121] R. Danescu u. a. „Particle Grid Tracking System Stereovision Based Obstacle Perception in Driving Environments“. In: *Intelligent Transportation Systems Magazine, IEEE* 4.1 (spring 2012), S. 6–20. ISSN: 1939-1390. DOI: 10.1109/MITS.2011.2178492.

[122] G. Tanzmeister u. a. „Grid-based mapping and tracking in dynamic environments using a uniform evidential environment representation“. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Mai 2014, S. 6090–6095. DOI: 10.1109/ICRA.2014.6907756.

[123] M. Schreier, V. Willert und J. Adamy. „From grid maps to Parametric Free Space maps - A highly compact, generic environment representation for ADAS“. In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2013, S. 938–944. DOI: 10.1109/IVS.2013.6629587.

[124] M. Kurdej u. a. „Map-Aided Evidential Grids for Driving Scene Understanding“. In: *IEEE Intelligent Transportation Systems Magazine* 7.1 (Spring 2015), S. 30–41. ISSN: 1939-1390. DOI: 10.1109/MITS.2014.2352371.

[125] Laurence Cholvy. „Non-exclusive hypotheses in Dempster-Shafer Theory“. In: *International Journal of Approximate Reasoning* 53.4 (2012), S. 493–501. ISSN: 0888-613X. DOI: https://doi.org/10.1016/j.ijar.2011.12.003. URL: http://www.sciencedirect.com/science/article/pii/S0888613X11001691.

[126] Albert Y Huang u. a. „Multi-Sensor Lane Finding in Urban Road Networks“. In: *Robotics: Science and Systems*. 2008.

[127] AP Dempster. „Upper and Lower Probabilities Induced by a Multivalued Mapping“. In: *The Annals of Mathematical Statistics* 38.2 (1967), S. 325–339.

[128] A.P. Dempster. „A generalization of Bayesian inference“. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1968), S. 205–247.

[129] G. Shafer. *A mathematical theory of evidence*. Princeton University Press, 1976.

[130] Ronald R. Yager. „On the dempster-shafer framework and new combination rules“. In: *Information Sciences* 41.2 (1987), S. 93–137. ISSN: 0020-0255. DOI: 10.1016/0020-0255(87)90007-7. URL: http://www.sciencedirect.com/science/article/pii/0020025587900077.

[131] Audun Jøsang, Javier Diaz und Maria Rifqi. „Cumulative and averaging fusion of beliefs“. In: *Information Fusion* 11.2 (2010), S. 192–200.

[132] Audun Jøsang und Simon Pope. „Dempster's rule as seen by little colored balls“. In: *Computational Intelligence* 28.4 (2012), S. 453–474.

[133] Forschungsgesellschaft für Strassen- und Verkehrswesen. *Richtlinien für die Anlage von Autobahnen (RAA)*.

[134] Forschungsgesellschaft für Strassen- und Verkehrswesen. *Richtlinien für die Anlage von Landstraßen (RAL)*.

[135] Forschungsgesellschaft für Strassen- und Verkehrswesen. *Richtlinien für die Anlage von Stadtstraßen (RASt)*.

[136] F. Homm u. a. „Efficient occupancy grid computation on the GPU with lidar and radar for road boundary detection“. In: *Intelligent Vehicles Symposium (IV), 2010 IEEE*. Juni 2010, S. 1006–1013. DOI: 10.1109/IVS.2010.5548091.

[137] S. Steyer, G. Tanzmeister und D. Wollherr. „Object tracking based on evidential dynamic occupancy grids in urban environments“. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2017, S. 1064–1070. DOI: 10.1109/IVS.2017.7995855.

[138] Andreas Geiger, Philip Lenz und Raquel Urtasun. „Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite“. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[139] Andreas Geiger u. a. „Vision meets Robotics: The KITTI Dataset“. In: *International Journal of Robotics Research (IJRR)* (2013).

[140] Jannik Fritsch, Tobias Kuehnl und Andreas Geiger. „A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms“. In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2013.

[141] Moritz Menze und Andreas Geiger. „Object Scene Flow for Autonomous Vehicles“. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[142] Marius Cordts u. a. „The Cityscapes Dataset“. In: *CVPR Workshop on The Future of Datasets in Vision*. 2015.

[143] Marius Cordts u. a. „The Cityscapes Dataset for Semantic Urban Scene Understanding“. In: *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. 2016.

[144] Jonathan Long, Evan Shelhamer und Trevor Darrell. „Fully convolutional networks for semantic segmentation“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, S. 3431–3440.

[145] J. Uhrig u. a. „Pixel-level encoding and depth layering for instance-level semantic segmentation“. In: *German Conference on Pattern Recognition (GCPR)*. 2016. URL: http://lmb.informatik.uni-freiburg.de//Publications/2016/BU16.

[146] Adam Paszke u. a. „ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation“. In: *CoRR* abs/1606.02147 (2016). arXiv: 1606.02147. URL: http://arxiv.org/abs/1606.02147.

[147] Hengshuang Zhao u. a. „Pyramid scene parsing network“. In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, S. 2881–2890.

[148] Hengshuang Zhao u. a. „ICNet for Real-Time Semantic Segmentation on High-Resolution Images“. In: *CoRR* abs/1704.08545 (2017). arXiv: 1704.08545. URL: http://arxiv.org/abs/1704.08545.

[149] E. Romera u. a. „Efficient ConvNet for real-time semantic segmentation“. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2017, S. 1789–1794. DOI: 10.1109/IVS.2017.7995966.

[150] Liang-Chieh Chen u. a. „Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation“. In: *CoRR* abs/1802.02611 (2018). arXiv: 1802.02611. URL: http://arxiv.org/abs/1802.02611.

[151] Heiko Hirschmuller. „Accurate and efficient stereo processing by semi-global matching and mutual information“. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Bd. 2. IEEE. 2005, S. 807–814.

[152] Marius Huber u. a. „Cubic Range Error Model for Stereo Vision with Illuminators“. In: *CoRR* abs/1803.03932 (2018).

[153] Lotfi A. Zadeh. „Review of Shaferś A Mathematical Theory of Evidence“. In: *The AI Magazine: Association for the Advancement of Artificial Intelligence* 5 (1984), S. 81–83.

[154] T. Weiss, B. Schiele und K. Dietmayer. „Robust Driving Path Detection in Urban and Highway Scenarios Using a Laser Scanner and Online Occupancy Grids“. In: *Intelligent Vehicles Symposium, 2007 IEEE*. Juni 2007, S. 184–189. DOI: 10.1109/IVS.2007.4290112.

[155] Q. Baig und O. Aycard. „Improving moving objects tracking using road model for laser data“. In: *2012 IEEE Intelligent Vehicles Symposium*. Juni 2012, S. 790–795. DOI: 10.1109/IVS.2012.6232300.

[156] M. Darms, M. Komar und S. Lueke. „Map based road boundary estimation“. In: *2010 IEEE Intelligent Vehicles Symposium*. Juni 2010, S. 609–614. DOI: 10.1109/IVS.2010.5548011.

[157] Michael Manz. „Modellbasierte visuelle Wahrnehmung zur autonomen Fahrzeugführung“. Diss. Institut für Technik Autonomer Systeme, Universität der Bundeswehr München, 2013.

[158] D. Zhu u. a. „Fitting Multiple Curves to Point Clouds with Complicated Topological Structures“. In: *Computer-Aided Design and Computer Graphics (CAD/Graphics), 2013 International Conference on*. Nov. 2013, S. 60–67. DOI: 10.1109/CADGraphics.2013.15.

[159] D. González u. a. „A Review of Motion Planning Techniques for Automated Vehicles“. In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (Apr. 2016), S. 1135–1145. ISSN: 1524-9050. DOI: 10.1109/TITS.2015.2498841.

[160] A. Stentz. „Optimal and efficient path planning for partially-known environments“. In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. Mai 1994, 3310–3317 vol.4. DOI: 10.1109/ROBOT.1994.351061.

[161] Anthony (Tony) Stentz. „The Focussed D* Algorithm for Real-Time Replanning“. In: *Proceedings of the International Joint Conference on Artificial Intelligence*. Aug. 1995.

[162] Maxim Likhachev u. a. „Anytime Dynamic A*: An Anytime, Replanning Algorithm.“ In: *ICAPS*. 2005, S. 262–271.

[163] Steven M. Lavalle. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Techn. Ber. 1998.

[164] D. Ferguson und A. Stentz. „Anytime RRTs“. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Okt. 2006, S. 5369–5375. DOI: 10.1109/IROS.2006.282100.

[165] L. Jaillet, J. Cortés und T. Siméon. „Sampling-Based Path Planning on Configuration-Space Costmaps“. In: *IEEE Transactions on Robotics* 26.4 (Aug. 2010), S. 635–646. ISSN: 1552-3098. DOI: 10.1109/TRO.2010.2049527.

[166] C. Urmson und R. Simmons. „Approaches for heuristically biasing RRT growth“. In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. Bd. 2. Okt. 2003, 1178–1183 vol.2. DOI: 10.1109/IROS.2003.1248805.

[167] Matthias Althoff. „Reachability analysis and its application to the safety assessment of autonomous cars“. Diss. Lehrstuhl für Steuerungs- und Regelungstechnik, Technische Universität München, 2010.

[168] Felix von Hundelshausen u. a. „Driving with tentacles: Integral structures for sensing and motion“. In: *Journal of Field Robotics* 25.9 (2008), S. 640–673. ISSN: 1556-4967. DOI: 10.1002/rob.20256. URL: http://dx.doi.org/10.1002/rob.20256.

[169] B. Gutjahr und M. Werling. „Automatic collision avoidance during parking and maneuvering; An optimal control approach“. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. Juni 2014, S. 636–641. DOI: 10.1109/IVS.2014.6856575.

[170] Steven M. LaValle. *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006. ISBN: 0521862051.

[171] M. Bahram u. a. „A prediction-based reactive driving strategy for highly automated driving function on freeways“. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. Juni 2014, S. 400–406. DOI: 10.1109/IVS.2014.6856503.

[172] Moritz Werling. „Ein neues Konzept für die Trajektoriengenerierung und -stabilisierung in zeitkritischen Verkehrsszenariens“. Diss. Institut für Angewandte Informatik/Automatisierungstechnik (AIA), Karlsruher Institut für Technologie, 2010.

[173] M. J. Thurley und V. Danell. „Fast Morphological Image Processing Open-Source Extensions for GPU Processing With CUDA“. In: *IEEE Journal of Selected Topics in Signal Processing* 6.7 (Nov. 2012), S. 849–855. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2012.2204857.

[174] G. Tanzmeister u. a. „Efficient Evaluation of Collisions and Costs on Grid Maps for Autonomous Vehicle Motion Planning“. In: *IEEE Transactions on Intelligent Transportation Systems* 15.5 (Okt. 2014), S. 2249–2260. ISSN: 1524-9050. DOI: 10.1109/TITS.2014.2313562.

[175] G. Tanzmeister u. a. „Path planning on grid maps with unknown goal poses“. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. Okt. 2013, S. 430–435. DOI: 10.1109/ITSC.2013.6728269.

[176] P. E. Hart, N. J. Nilsson und B. Raphael. „A Formal Basis for the Heuristic Determination of Minimum Cost Paths“. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (Juli 1968), S. 100–107. ISSN: 0536-1567. DOI: 10.1109/TSSC.1968.300136.

[177] G. Tanzmeister, D. Wollherr und M. Buss. „Environment-based trajectory clustering to extract principal directions for autonomous vehicles“. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sep. 2014, S. 667–673. DOI: 10.1109/IROS.2014.6942630.

[178] Martin A. Fischler und Robert C. Bolles. „Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography“. In: *Communications of the ACM* 24.6 (1981), S. 381–395.

[179] M. Nieto, L. Salgado und F. Jaureguizar. „Robust road modeling based on a hierarchical bipartite graph“. In: *2008 IEEE Intelligent Vehicles Symposium*. Juni 2008, S. 61–66. DOI: 10.1109/IVS.2008.4621241.

[180] Roman Mansilla Martin. „Gridbasierte Online-Schätzung von Fahrstreifen für das automatisierte Fahren“. Magisterarb. München: Technische Universität München, Lehrstuhl für Mensch-Maschine Kommunikation, 2015.

[181] A. Abramov u. a. „Multi-lane perception using feature fusion based on GraphSLAM“. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Okt. 2016, S. 3108–3115. DOI: 10.1109/IROS.2016.7759481.

[182] D. Ruppert, M.P. Wand und R.J. Carroll. *Semiparametric Regression*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2003.

[183] *BMW Group PressClub Deutschland, Automatisiertes Fahren bei der BMW Group: #NEXTGen 19.* Accessed: 2020-01-21. URL: http://www.press.bmwgroup.com/.

[184] Morgan Quigley u. a. „ROS: an open-source Robot Operating System“. In: *ICRA Workshop on Open Source Software*. 2009.