

# Optimale Bahn- und Trajektorienplanung für Automobile

Zur Erlangung des akademischen Grades eines

**Doktors der Ingenieurwissenschaften**

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

**Dissertation**

von

DIPL.-INFORM. JULIUS ZIEGLER

aus Bonn-Bad Godesberg

Hauptreferent:	Prof. Dr.-Ing. C. Stiller
Korreferent:	Prof. Dr.-Ing. U. Hanebeck
Tag der mündlichen Prüfung:	10. September 2015



# Inhaltsverzeichnis

<b>Symbole und Notation</b>	<b>V</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Gliederung . . . . .	2
1.2 Einordnung . . . . .	2
1.2.1 Einbettung als Komponente eines automatischen Fahrsystems . . . . .	4
1.2.2 Einbettung in den Forschungsbereich . . . . .	6
1.3 Zielsetzung der Arbeit . . . . .	14
<b>2 Grundlagen</b>	<b>15</b>
2.1 Graphentheorie . . . . .	15
2.1.1 Graphen . . . . .	15
2.1.2 Geometrische Graphen . . . . .	17
2.1.3 Kürzeste Pfade . . . . .	17
2.2 Fahrzeugmodellierung . . . . .	19
2.3 Finite Differenzen . . . . .	21
<b>3 Kontinuierliche Distanztransformationen</b>	<b>23</b>
<b>4 Optimale Trajektorienplanung: Grundlagen</b>	<b>34</b>
4.1 Variationale Formulierung . . . . .	34
4.2 Exakte Lösungen . . . . .	36
4.2.1 Lösen der EULER-POISSON-Gleichung . . . . .	36
4.2.2 Lösen der RICCATI-Gleichung . . . . .	40

<b>5</b>	<b>Lokale, kontinuierliche Lösungsverfahren</b>	<b>43</b>
5.1	Diskretisierung des Energiefunktional	43
5.2	Optimalitätskriterien und Kostenfunktional	46
5.3	Nebenbedingungen	48
5.3.1	Äußere Nebenbedingungen	49
5.3.2	Innere Nebenbedingungen	52
5.4	Randbedingungen	53
5.4.1	Symmetrische Randbedingungen, Transversalität	54
5.4.2	Fortlaufende Neuplanung	54
5.5	Nichtlineare Optimierung	56
5.5.1	LAGRANGE-Multiplikatoren	56
5.5.2	Quadratisches Programmieren	58
5.5.3	Sequenzielles quadratisches Programmieren (SQP)	67
<b>6</b>	<b>Globale, diskrete Lösungsverfahren</b>	<b>69</b>
6.1	Gitter aus Grundgeometrien	69
6.2	Infinitesimales Gitter	71
6.2.1	Problembeschreibung	71
6.2.2	Ableitung eines Graphen	72
6.3	Schnelle parallele Graphsuche	77
6.4	Weitere Betrachtungen	82
6.4.1	Asymptotische Komplexität	82
6.4.2	Funktionale, die höhere Ableitungen enthalten	83
6.4.3	Grenzen	84
<b>7</b>	<b>Experimente</b>	<b>87</b>
7.1	Bertha-Benz-Fahrt 2013	87
7.1.1	Karten- und sensorbasiertes Lagebild	88
7.1.2	Äußere Nebenbedingungen	89
7.1.3	Ergebnisse	96
7.2	Navigation in beengter Umgebung	97

7.2.1	Gitter . . . . .	97
7.2.2	Kostenfunktional . . . . .	102
7.2.3	Nebenbedingungen . . . . .	102
7.2.4	Randbedingungen . . . . .	104
7.2.5	Lokale Verfeinerung . . . . .	104
7.2.6	Ergebnisse . . . . .	106
<b>8</b>	<b>Zusammenfassung und Ausblick</b>	<b>110</b>



# Symbole und Notation

Mathematische Symbole werden, sofern sie nicht allgemein gebräuchlich sind, in dieser Arbeit in ihrem jeweiligen Kontext definiert und eingeführt. Die Bildung der Symbole folgt dabei diesen allgemeinen Vorschriften:

Skalare	Kleinbuchstaben, normale Stärke	$a, b, \alpha, \beta$
Vektoren	Kleinbuchstaben, fett	$\mathbf{a}, \mathbf{b}, \alpha, \beta$
Matrizen	Großbuchstaben, fett	$\mathbf{A}, \mathbf{B}, \Sigma$
Mengen	Großbuchstaben, kursiv	$A, B$

In Einzelfällen wird von dieser Regelung abgewichen, zum Beispiel, um Mehrdeutigkeiten zu vermeiden, oder um Konventionen aus dem Bereich der Physik gerecht zu werden (der Grossbuchstabe  $E$  deutet zum Beispiel auf eine Energie hin). Solche Abweichungen werden im Kontext deutlich gemacht.





# 1 Einleitung

Automatisierte Fahrfunktionen in Automobilen sind nützlich, da sie das Potential bergen, Unfälle zu vermeiden, die durch menschliche Fehler verursacht werden. Ein wichtiges Teilproblem - neben der maschinellen Wahrnehmung - ist hierbei die Bahn- und Trajektorienplanung. Hiermit ist der Teil des automatischen Fahrsystems gemeint, der ein wahrnehmungs- und kartengestütztes Lagebild als Eingabe erhält, und daraus eine Solltrajektorie für das eigene Fahrzeug berechnet.

Das autonome Fahren kann je nach Situation und zu planendem Manöver sehr unterschiedliche Anforderungen an die Trajektorienplanung stellen. Die meisten Anwendungsszenarien lassen sich einer der beiden im Folgenden beschriebenen Klassen zuordnen:

- *Fahren in statischer, unstrukturierter Umgebung.* Die Bewegung des eigenen Fahrzeuges findet langsam statt, und der resultierende Anhalteweg des eigenen Fahrzeuges ist nahe null. Hindernisse in der Umgebung bewegen sich nicht. Die Umgebung ist nicht durch Verkehrswege strukturiert. Ein typisches Anwendungsszenario dieser Klasse ist die Navigation auf einer Parkfläche. Hier kommen besonders *kombinatorische* Aspekte zum Tragen, zum Beispiel durch die Notwendigkeit, vorwärts und rückwärts zu navigieren, um den verfügbaren Freiraum optimal auszunutzen. Das Planungsziel ist meistens, eine definierte Zielkonfiguration zu erreichen. Zur Lösung werden in dieser Arbeit kombinatorische Methoden (Kapitel 6) vorgeschlagen.
- *Fahren in dynamischer, strukturierter Umgebung.* Hierzu gehört das Fahren in fließendem Verkehr. Die Anforderungen sind zum Teil entgegengesetzt zu den im letzten Absatz genannten. Die Planung muss in hoher Rate wiederholt werden, um ausreichend kurze Reaktionszeiten zu gewährleisten. Bewegte Objekte müssen explizit berücksichtigt werden, indem ihre Trajektorien antizipiert werden. Die Umgebung ist hoch strukturiert und kann zum Beispiel durch die Fahrstreifen-geometrie und eine Objektliste repräsentiert werden. Die hohe Strukturierung schränkt den Fahrweg so stark ein, dass die meisten kombinatorischen Aspekte vernachlässigt werden können. Das Planungsziel

ist beispielsweise, eine Sollgeschwindigkeit einzuhalten und innerhalb des gewählten Fahrstreifens zu bleiben. Diese Problemklasse umspannt einen weiten Geschwindigkeitsbereich, weshalb eine genauere Modellierung der Fahrdynamik erforderlich ist. In dieser Arbeit wird gezeigt, dass dies die Dimensionalität des Lösungsraumes erhöht, weshalb solche Probleme mit kombinatorischen Ansätze nur schlecht behandelt werden können. Es werden alternativ lokale, kontinuierliche Verfahren vorgeschlagen (Kapitel 5).

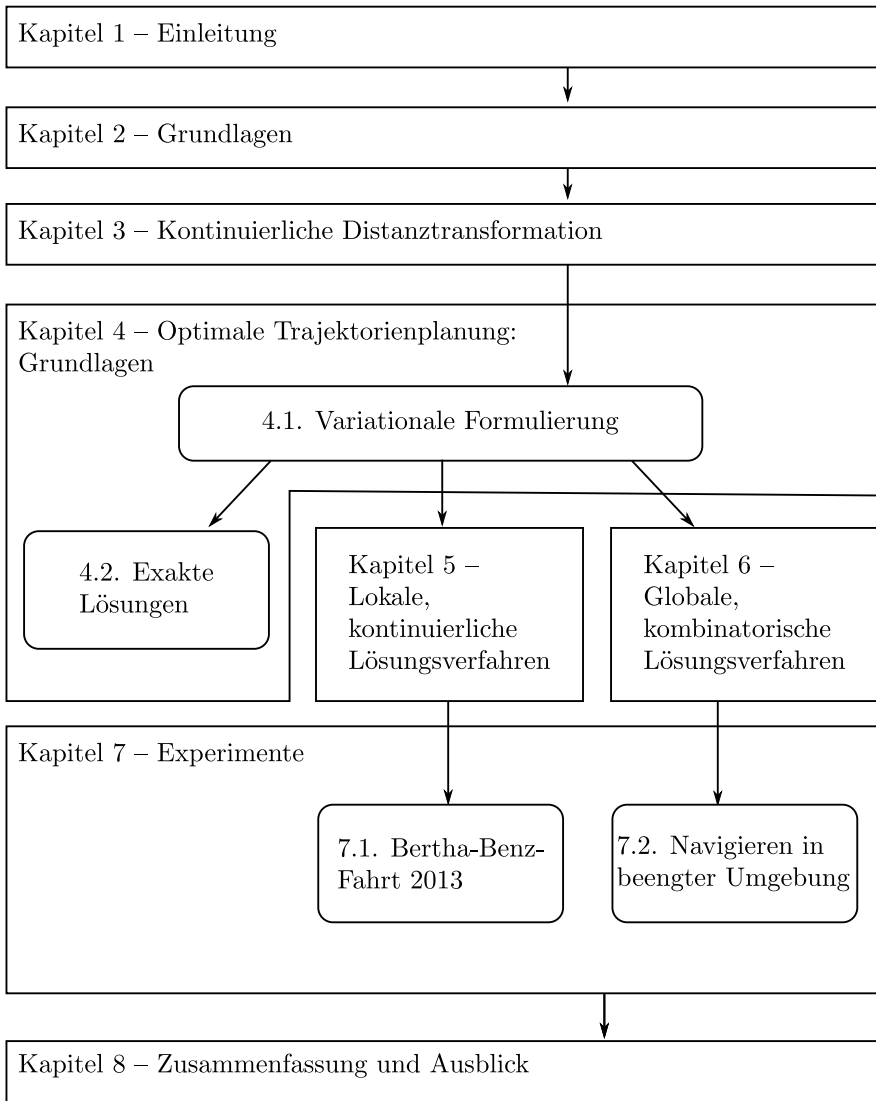
Es folgt eine Gliederung der Arbeit. Anschließend wird sie im Forschungsfeld der Bewegungsplanung eingeordnet. Danach werden die Ziele dieser Arbeit formuliert.

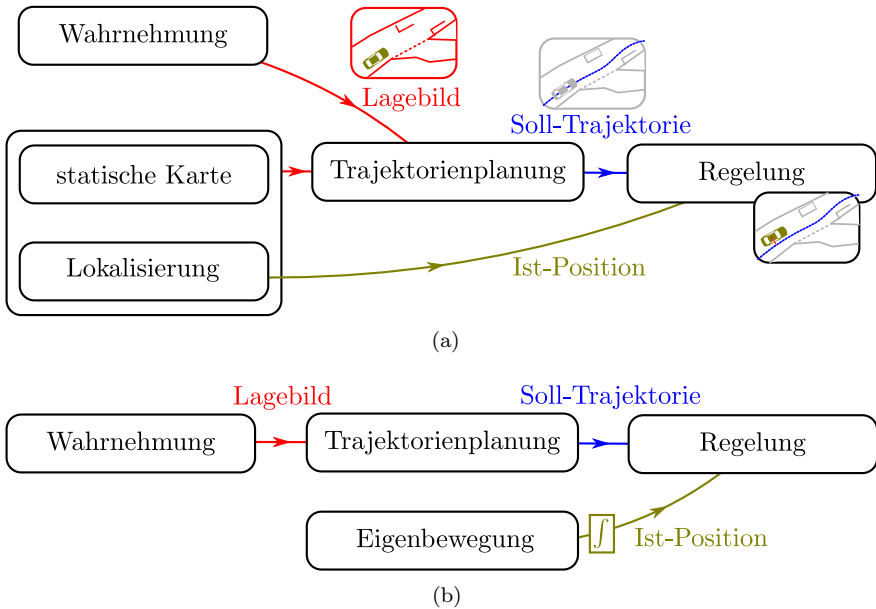
## 1.1 Gliederung

Die vorliegende Arbeit gliedert sich wie in Abbildung 1.1 dargestellt. Nach den einleitenden Kapiteln 1 und 2 wird in Kapitel 3 eine Methode der schnellen Distanzberechnung erläutert, welche ein wichtiges Hilfsmittel für die danach beschriebenen Verfahren der Trajektorienplanung darstellt. Danach beginnt der Hauptteil der Arbeit mit Kapitel 4. In diesem wird die Trajektorienplanung zunächst als ein Problem der Variationsrechnung formuliert (Abschnitt 4.1). Für das so formalisierte Problem werden dann drei verschiedene Lösungsmethoden vorgeschlagen. Die exakten Lösungen (Abschnitt 4.2) sind hierbei vor allem von theoretischem Interesse. Sie können nur auf einige einfache Problemklassen angewendet werden, einige dabei ableitbare Aussagen haben aber auch praktische Relevanz. In den Kapiteln 5 und 6 werden zwei unterschiedliche approximative Lösungsverfahren für Variationsprobleme hergeleitet. Beide Verfahren wurden im Rahmen dieser Arbeit zu funktionierenden Demonstratoren ausentwickelt, diese werden in Kapitel 7 vorgestellt. Gemeinsam decken die Verfahren typische Anwendungsfälle für die beiden eingangs genannten Grundscenarien – das Fahren in dynamischer, strukturierter Umgebung sowie das Fahren in (quasi-)statischer, unstrukturierter Umgebung – ab.

## 1.2 Einordnung

In diesem Abschnitt erfolgt eine Einordnung der vorliegenden Arbeit in zweierlei Hinsicht. Zunächst wird eine komponentenweise Einbettung

**Abbildung 1.1:** Gliederung der Arbeit



**Abbildung 1.2:** Einbettung der Trajektorienplanung in ein automatisches Fahrsystem. (a) kartenbasiert. (b) rein sensorbasiert.

vorgenommen, indem die Trajektorienplanung als Modul eines automatischen Fahrsystems betrachtet wird, wobei die Schnittstellen zu anderen Systemkomponenten erläutert werden. Anschließend wird sie im Forschungsfeld der Bewegungsplanung eingeordnet, und Ähnlichkeiten und Unterschiede zu einigen verwandten Arbeiten werden erläutert.

### 1.2.1 Einbettung als Komponente eines automatischen Fahrsystems

In diesem Abschnitt werden die Teilsysteme eines automatischen Fahrsystems erläutert, die mit der Trajektorienplanung in direkter Verbindung stehen.

Für Fahrsysteme mit hohem Automatisierungsgrad haben sich seit der Urban Challenge 2007 [BLS08] hauptsächlich kartenbasierte Ansätze etabliert. Bei diesen werden statische Eigenschaften des Verkehrsraumes in einer Karte abgelegt. Dies betrifft im Wesentlichen die im Verkehrsraum vorhandene

statische Infrastruktur, die sowohl aus geometrischen Attributen besteht - beispielsweise dem Verlauf von Fahrstreifen - als auch Relationen *zwischen* solchen Geometrien enthält - also beispielsweise die Vorfahrtsrelation zwischen zwei sich kreuzenden Fahrstreifen. Diese Teile des Verkehrsraumes müssen dann nicht mehr zur Fahrtzeit vom Fahrzeug sensorisch erfasst werden - das Fahrzeug „weiß“ stattdessen, dass sie gegeben sind. Die Voraussetzung für ein solches kartenbasiertes Vorgehen ist eine Methode zur Lokalisierung, mit der die Fahrzeugposition und -orientierung relativ zur Karte hinreichend genau bestimmt werden kann. Das kartenbasierte Lagebild des Fahrzeugumfeldes muss durch Wahrnehmungssysteme vervollständigt werden. Diese müssen zumindest alle veränderlichen Eigenschaften des Verkehrsraumes erfassen, also insbesondere andere Verkehrsteilnehmer. Der Trajektorienplanung nachgelagert ist ein Trajektorienfolgeregler. Dieser führt die Ist-Position des Fahrzeuges zurück und regelt so das Fahrzeug entlang der Solltrajektorie. Die Einbettung der Trajektorienplanung in ein kartenbasiertes Fahrsystem zeigt Abbildung 1.2a.

Es kann auch versucht werden, auf Kartendaten zu verzichten. Prinzipiell ist das erstrebenswert, da ja - wenn man nur ausreichend lange Zeiträume betrachtet - eigentlich kein Element des Verkehrsraumes als wirklich statisch betrachtet werden kann. Allerdings bürdet diese Vorgehensweise dem Wahrnehmungsmodul erhebliches auf. Das Fahrzeugumfeld muss in diesem Fall nämlich vollständig sensorisch erfasst werden („Fahren auf Sicht“). Nach dem heutigen Stand der Technik gilt dies nur in einigen spezifischen Domänen - beispielsweise auf der Autobahn - als durchführbar. Wenn man auf Kartendaten vollständig verzichten kann, wird auch kein System zur globalen Lokalisierung mehr benötigt. Es kann durch eine - im allgemeinen einfachere - Eigenbewegungsschätzung ersetzt werden. Die Eigenbewegung wird aufintegriert und schafft so ein lokales Koordinatensystem, das das globale Bezugssystem ersetzt. Das lokale Koordinatensystem ist einer Drift unterworfen. Dieser Driftfehler wird aber erst auf der Ebene des Trajektorienfolgereglers sichtbar und wird von diesem ausgeglichen. Die Einbettung der Trajektorienplanung in ein kartenloses, rein sensorbasiertes Fahrsystem zeigt Abbildung 1.2b.

Zusammenfassend kann festgestellt werden, dass die Schnittstellen der Trajektorienplanung von der gewählten Strategie - kartenbasiert oder kartenlos - unabhängig sind. Der Einfachheit halber wird deshalb im Folgenden davon ausgegangen, dass eine globale Lokalisierung vorhanden ist, so dass auch die Trajektorie in einem weltfesten Bezugssystem referenziert ist. In Kapitel 7 wird über zwei praktische Umsetzungen eines automatischen Fahrsystems berichtet - eines davon ist stark kartenbasiert, das andere kartenlos.

## 1.2.2 Einbettung in den Forschungsbereich

Die Bahn- und Trajektorienplanung - übergreifend spricht man auch von Bewegungsplanung - ist ein etabliertes Teilgebiet der Robotik, und eine wichtige Voraussetzung für die Realisierung aller Arten von mobilen, selbstfahrenden Plattformen. Automatisch fahrende PKW stellen hierbei einen interessanten, aber auch besonders anspruchsvollen Spezialfall dar.

Dieses recht umfangreiche Forschungsfeld wird im Folgenden zunächst strukturiert, indem wichtige Kriterien aufgeführt werden, anhand derer Problemstellungen im Bereich der Bewegungsplanung unterschieden werden können. Hierdurch soll auch ein Klassifikationsschema und eine einheitliche Terminologie für Bewegungsplanungsprobleme aufgebaut werden. Anschließend wird dann auf eine Auswahl von Arbeiten eingegangen, die von ihrer Methodik oder von ihrem Anwendungsfeld her der vorliegenden Dissertation nahe verwandt sind.

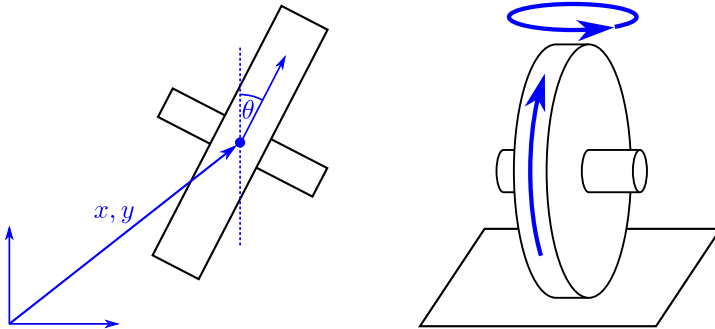
Die wichtigsten Kriterien, anhand derer Probleme in der Bewegungsplanung unterschieden werden können, sind folgende:

### **Dimensionalität**

Eine besonders fundamentale Kenngröße bei der Trajektorienplanung ist die Anzahl der Dimensionen eines bestimmten Planungsproblemles. Man beachte, dass hier nicht die euklidische Dimensionalität des Arbeitsraumes gemeint ist, sondern die Dimensionalität des Raumes aller möglichen Zustände, die das System einnehmen kann (Phasenraum). Einige der im Folgenden genannten Kriterien lassen sich prinzipiell als eine Erhöhung der Dimensionalität des Phasenraumes ausdrücken, wie an entsprechender Stelle gezeigt werden wird. Bei kombinatorischen Verfahren steigt die Anzahl zu betrachtender diskreter Systemzustände – und damit die Laufzeit – exponentiell mit der Dimensionalität des Planungsproblemles an.

### **Holonomie**

Grundsätzlich sind bei mobilen Plattformen holonome und nichtholonome Systeme zu unterscheiden. Bei nichtholonomen Systemen gibt es beim infinitesimalen Übergang von einem Systemzustand zu einem benachbarten Einschränkungen, die nur als Differentialgleichungen ausgedrückt werden können, in denen Ableitungen von Systemzuständen vorkommen. Beispielsweise weist ein Einrad wie in Abbildung 1.3 drei globale Freiheitsgrade



**Abbildung 1.3:** Globale und lokale Freiheitsgrade des nichtholonomen Einradmodells.

auf, nämlich die Position  $x, y$  und seine Ausrichtung  $\theta$ . Es stellen auch tatsächlich alle möglichen Kombinationen dieser drei Freiheitsgrade *gültige* Zustände dar. Eine infinitesimale Translation des Einrades kann aber nur in der Laufrichtung des Rades erfolgen. Es muss also die Differentialgleichung  $\dot{x} \cos \theta + \dot{y} \sin \theta = 0$  gelten. Zusammen mit der stets möglichen Drehung um die Hochachse ergeben sich so lokal nur zwei Freiheitsgrade. In [LSL98; Lat91a; BL93; ZP08; ZWS08] werden Bahnplanungsstrategien vorgestellt, die sich für nichtholonome Systeme eignen.

## Optimalität

Die vorliegende Arbeit behandelt ausschließlich optimale Verfahren. Sie liefern als Ergebnis Trajektorien, die ein – noch genauer zu definierendes – Kostenfunktional, oder eine sinnvolle Approximation eines solchen, minimieren. Mit Optimaltrajektorien befasst sich die Theorie der optimalen Steuerung. Beispielsweise ist die sich durch einen linear-quadratischen Regler (LQ-Regler) [Pon+64] ergebende Zustandstrajektorie in einem wohldefinierten Sinne optimal. Auch der moderneren Methode der modellprädiktiven Regelung (model predictive control, MPC) [GPM89] liegt ein wohldefiniertes Gütefunktional zu Grunde. Anders als die in dieser Arbeit vorgeschlagenen Verfahren berücksichtigen diese klassischen Methoden der Regelungstechnik im Allgemeinen keine Hindernisse im Arbeitsraum.

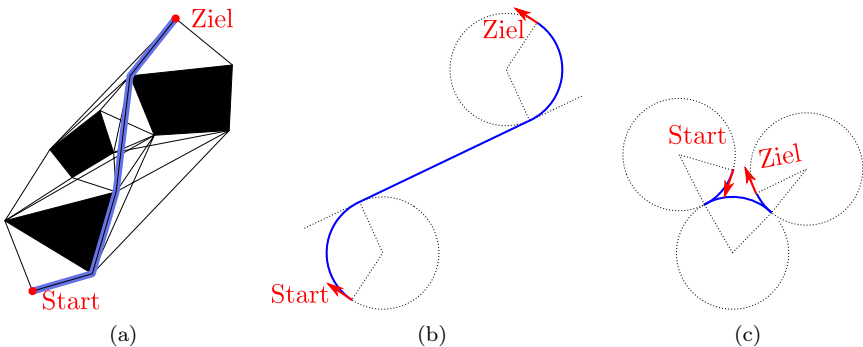
## Vollständigkeit

Man bezeichnet einen Planungsalgorithmus als *vollständig*, wenn er nach endlicher Zeit terminiert und entweder eine durchführbare Trajektorie berechnet oder richtig feststellt, dass keine solche existiert. Zu den Methoden, die in erster Linie Vollständigkeit anstreben, dafür aber meistens die Optimalität vernachlässigen, gehört das Verfahren der *rapidly exploring random trees* (RRT) [LaV98; LK01], das einen zufälligen Suchbaum im Arbeitsraum expandiert, und solche Methoden, die den Arbeitsraum topologieerhaltend auf einen geometrischen Graphen reduzieren. Ein solcher Graph wird anschaulich auch als *roadmap* bezeichnet. Zu letzteren gehören Methoden, die Bahnen entlang des Voronoi-Diagramms planen [ZWS08; OY85; Lat91b], oder die den Arbeitsraum in Zellen zerlegen [Lat91b]. Die Methode RRT ist streng genommen nicht vollständig, da sie im Fall, dass keine gültige Trajektorie existiert, nicht terminiert.

## Kostenfunktionale höherer Ordnung

Das traditionelle Problem der Bahnplanung befasst sich mit der Planung *kürzester* Pfade – also von Pfaden kürzester Bogenlänge. Solche Pfade weisen bei holonomen Systemen im Allgemeinen Knicke auf (vergl. Abbildung 1.4a). Durch Berücksichtigung nichtholonomer Einschränkungen können Knicke zwar vermieden werden, dennoch führen kürzeste Pfade dann im Allgemeinen zu extremalen Eingangsgrößen für das System. Die Abbildungen 1.4b und 1.4c zeigen Beispiele für solche Pfade. Das zugrundeliegende System ist das eines einfachen kinematischen Fahrzeuges mit Vorderradlenkung und beschränktem Lenkwinkel (gemäß dem später in Abschnitt 2.2 beschriebene Einspurmodell), das vorwärts und rückwärts fahren kann. Die kürzesten Pfade bestehen hier aus Kreisbögen mit minimalem Kurvenradius und Geradenstücken [RS91]. Für ein schnelles Durchfahren dieser Bahn ist dies unvorteilhaft, da sprungartig gelenkt werden müsste. Je weiter man sich von rein kinematischen Anforderungen entfernt, desto weniger wünschenswert ist das. Stattdessen werden Trajektorien angestrebt, die hinreichend glatt sind, und man ist deshalb bestrebt, nicht die Bogenlänge der Trajektorie zu minimieren, sondern ein Gütemaß, das zum Beispiel Zeitableitungen der Trajektorie enthält. Wird ein Trajektorienplanungsproblem in dieser Weise gestellt, so spricht man von einer Trajektorienplanung mit *Gütefunktionalen höherer Ordnung*. Die Behandlung solcher Gütefunktionale geht mit einer Erhöhung der Dimensionalität des der Planung zugrunde liegenden Phasenraumes einher.

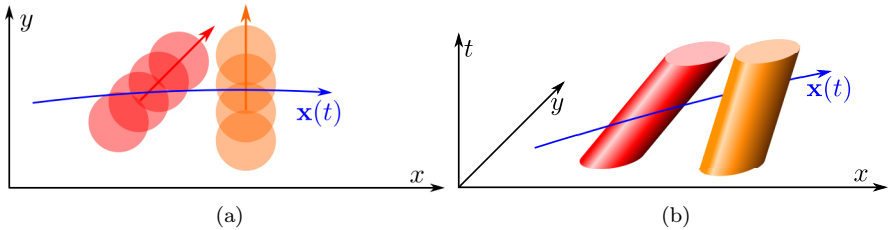




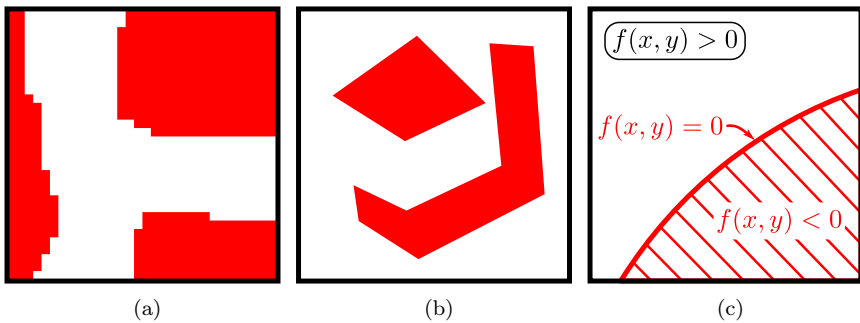
**Abbildung 1.4:** (a): Das zweidimensionale, holonome Problem kürzester Pfade kann mit dem Sichtbarkeitsgraphen gelöst werden. (b) und (c): Für ein kinematisches Fahrzeug mit Vorderradlenkung und begrenztem Lenkwinkel (Einspurmodell) bestehen die kürzesten Pfade aus Geradenstücken und Kreissegmenten minimalen Krümmungsradius’.

## Umgebung

Weitere Unterscheidungskriterien für Trajektorienplanungsprobleme ergeben sich aus angenommenen Eigenschaften der Umgebung. Im einfachsten Fall ist die Umgebung „leer“, enthält also keine Hindernisse. Methoden, die solche Probleme lösen, werden gelegentlich als *lokale Planer* bezeichnet. Sie werden oft als Komponente eines Trajektorienplaners eingesetzt, der eine globale Lösung stückweise aus solchen lokal optimalen Teillösungen zusammensetzt. Für nichtholonome Systeme stellt selbst die Trajektorienplanung in einer leeren Umgebung keine einfache Aufgabe dar. Es ist kein Verfahren bekannt, das für ein beliebiges nichtholonomes System eine Trajektorie berechnet, die einen Start- und Endzustand exakt miteinander verbindet. Für spezielle Systeme lassen sich aber Lösungen angeben. Beispielsweise wurde in [RS91] die *exakte* Lösung für das Problem *kürzester* Pfade für ein *kinematisches* Fahrzeug mit Vorderradlenkung und beschränktem Lenkwinkel (Einspurmodell) hergeleitet. Weitere optimale, lokale Planer lassen sich aus der Theorie der optimalen Steuerungen und mit Hilfe der Variationsrechnung ableiten. Zum Beispiel können durch Lösen der EULER-LAGRANGE-Gleichung [Els61] für bestimmte Klassen von Gütefunktionalen höherer Ordnung Optimaltrajektorien gefunden werden. Dieser Ansatz wird in Abschnitt 4.2 aufgegriffen.



**Abbildung 1.5:** (a) Trajektorienplanung in der Umgebung zweier dynamischer Hindernisse (rot und orange). (b) Der Lösungsraum wird hierzu um eine Zeitachse erweitert.



**Abbildung 1.6:** Umgebungsrepräsentationen. Hindernisse sind rot oder rot schraffiert. (a) Belegungsgrid. (b) polygonal. (c) funktional.

Im Allgemeinen werden sich in der Umgebung Hindernisse befinden. Diese können statisch oder bewegt sein. Der Unterschied zwischen einer statischen und einer bewegten Umgebung ist weniger fundamental als es auf den ersten Blick erscheint: Man kann den Phasenraum einfach um die Zeitachse  $t$  erweitern. Vorausgesetzt, dass vollständige Information über die zukünftige Bewegung der Hindernisse vorliegt, erhält man dann ein Problem mit um eins höherer Dimensionalität, das aber ansonsten fast wie ein statisches Problem zu behandeln ist. Dieses Vorgehen ist in Abbildung 1.5 illustriert. Anders als bei einer 3-dimensionalen, statischen Planung muss aber beachtet werden, dass  $x$  und  $y$  Funktionen von  $t$  sind – zu jeder Zeit muss es genau eine Position  $(x(t), y(t))$  geben, in Abbildung 1.5b verläuft die Trajektorie also monoton „nach oben“.

Je nach verwendetem Verfahren für die Trajektorienplanung gibt es Anforderungen an die *Repräsentation* der Umgebung. Sehr allgemeingültig

sind Belegungsgitter, wie beispielhaft in Abbildung 1.6a dargestellt. Bei Belegungsgittern kann ein Kollisionstest für einen Punkt besonders effizient durchgeführt werden. Ebenfalls verbreitet ist eine Repräsentation der Hindernisse als Menge von Polygonen (Abbildung 1.6b). Sie ist im Allgemeinen kompakter und kommt prinzipiell ohne räumliche Diskretisierungsfehler aus. Für die kontinuierlichen Verfahren in Kapitel 5 müssen Hindernisse in Form von Ungleichungen repräsentiert werden. Dies ist schematisch in Abbildung 1.6c veranschaulicht: Die Kontur eines Hindernisses in der  $x, y$ -Ebene ist implizit über die Gleichung  $f(x, y) = 0$  definiert. Für Punkte innerhalb des Hindernisses gilt  $f(x, y) < 0$ , für solche außerhalb  $f(x, y) > 0$ . Die Funktion  $f$  muss im Allgemeinen stetig und mehrfach differenzierbar sein. Eine Voraussetzung für die Umsetzung der kontinuierlichen Verfahren in Kapitel 5 ist eine Methode, die im Rahmen dieser Arbeit entwickelt wurde (Kapitel 3), und die es erlaubt, eine polygonale in eine geeignete funktionale Umfeldrepräsentation umzusetzen.

In der Umgebung von Hindernissen kann nur das *zweidimensionale, holonome* Problem *kürzester* Pfade als exakt und effizient gelöst betrachtet werden. Der kürzeste Pfad kann beispielsweise durch Suche im Sichtbarkeitsgraphen, vergleiche Abbildung 1.4a, identifiziert werden [LW79].

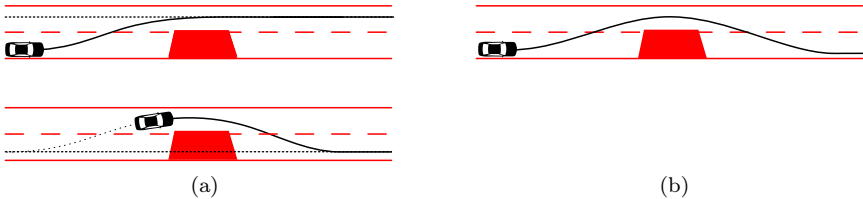
## Verwandte Arbeiten

Ein beträchtlicher Teil der Forschung im Bereich der Bewegungsplanung für Automobile befasst sich mit der Lösung *kombinatorischer* Probleme. Hierzu gehört der Anwendungsfall des Einparkens oder allgemein des langsamen Manövrierens in enger, unstrukturierter Umgebung. Der Lösungsraum der Trajektorien ist hierbei extrem vielfältig und muss prinzipiell alle möglichen Aneinanderreihungen von Lenkmanövern und Richtungswechseln enthalten. Es überrascht nicht, dass dieses Feld von kombinatorischen Methoden dominiert wird. Bei diesen wird stets ein Suchbaum expandiert, der den Arbeitsraum zufällig oder deterministisch exploriert. Betrachtet man einschlägige Publikationen, so scheinen randomisierte Methoden [LaV98; LK01; SMK12; Suc+08] dieses Feld zu beherrschen. Sie weisen aber schlechte theoretische Eigenschaften auf: Sie sind nicht vollständig, da sie nicht terminieren, wenn für ein gegebenes Problem keine Lösung existiert. Auch Optimalität kann prinzipiell nur nach unendlich langer Laufzeit garantiert werden, wobei keine Untersuchungen bekannt sind, die sich mit dem Konvergenzverhalten dieser Verfahren befassen, also untersuchen, wie schnell sich das Verfahren der Optimallösung annähert. Zu den deterministischen Verfahren dieser Familie gehört die Verwendung von Gittern aus Grundgeometrien (*state lattices*)

nach [PK05; PKK09]. Auf diesen Ansatz wird in Abschnitt 6.1 noch genauer eingegangen, da er Ausgangspunkt der Überlegungen ist, der auf den Ansatz der infinitesimalen Gitter führt (Abschnitt 6.2). Im Vergleich zu den state lattices erübrigt sich beim infinitesimalen Gitter die Lösung nichtholonomer Teilprobleme für die Berechnung der Grundgeometrien. Das Gitter wird stattdessen direkt algorithmisch aus einer variationalen Formulierung des Problems abgeleitet, wobei Kostenfunktionale, die Ableitungen bis zur zweiten Ordnung enthalten, verwendet werden können. Auch nichtholonome Zwangsbedingungen können realisiert werden, sofern die zugehörigen Differenzialgleichungen sich durch Ableitungen ebenfalls bis zur zweiten Ordnung ausdrücken lassen.

Für den Anwendungsfall des Fahrens in fließendem Verkehr, innerhalb einer Umgebung, die durch Fahrstreifen strukturiert ist, werden kombinatorische Verfahren nur selten eingesetzt. Hierfür gibt es zwei Gründe. Erstens sind kombinatorische Aspekte beim Fahren in fließendem Verkehr erfahrungsgemäß weniger wichtig. Zweitens werden in diesem Anwendungsfall andere Aspekte relevant, beispielsweise Glattheit der Trajektorie und die Möglichkeit, bewegte Objekte zu behandeln. Gerade diese Aspekte gehen aber mit einer Erhöhung der Dimensionalität des Phasenraumes einher, der um die Zeitachse und um innere Systemzustände oder Ableitungen der Systemzustände erweitert werden muss. Der Aufwand der kombinatorischen Verfahren steigt aber im Allgemeinen exponentiell mit der Dimensionalität des Phasenraumes. Das Verfahren kann so durch seine Laufzeit impraktikabel werden. Wird, um den Laufzeitnachteil zu kompensieren, der Zustandsraum weniger dicht abgetastet, so leiden die Genauigkeit und Vollständigkeit des Verfahrens. Trotz dieser Einschränkungen wurden kombinatorische Verfahren für diesen Anwendungsfall untersucht. Über eigene Erfahrungen mit einem geometrisch an den Straßenverlauf angepassten state lattice berichtete der Verfasser in [ZS09]. Dieses Verfahren wurde später in [McN+11] nochmals aufgegriffen.

Ein verbreiteter Ansatz beim Fahren in fließendem Verkehr besteht darin, die Fortsetzung der aktuellen Trajektorien durch eine fächerartige Menge von s-förmigen Ausweichtrajektorien anzunähern. Aus dieser Menge wird dann nach einem Gütekriterium ausgewählt. Eine derartige Strategie wurde vom Siegerfahrzeug der Urban Challenge eingesetzt [How+08]. Werling u. a. [Wer+10] schlagen ein ähnliches Verfahren vor. Zur Berechnung der Grundtrajektorien wenden sie Methoden der optimalen Steuerung an, wobei das Gütemaß auf die nachgeschaltete Auswahlstrategie abgestimmt ist. Erst durch dieses Vorgehen lassen sich einige wünschenswerte Eigenschaften – Stabilität und zeitliche Konsistenz – nachweisen. Man kann diese Metho-



**Abbildung 1.7:** Suboptimalität beim Ansatz nach [Wer+10]. Erläuterungen im Fließtext.

den als „beschränkte“ kombinatorische Verfahren interpretieren, bei denen die Expansion des Suchbaumes bereits nach einem Schritt abgebrochen wird. Dies limitiert die Allgemeinheit der Manöver, die vorausschauend geplant werden können, allerdings wird die kombinatorische Komplexität drastisch gesenkt. Schon für simple Manöver ist dadurch die Optimalität des Verfahrens nicht mehr gewährleistet. Abbildung 1.7 illustriert die Problematik schematisch für das Passieren eines statischen Hindernisses. Es wird angenommen, dass das Gütefunktional in geeigneter Weise Glattheit der Trajektorie und das Beibehalten des rechten Fahrstreifens motivieren soll. Das Fahrzeug befindet sich initial im Zustand wie in Abbildung 1.7a oben. Die bei [How+08; Wer+10] zur Auswahl stehenden Trajektorien beschreiben stets ein Manöver, das – in optimaler Weise – eine seitliche Parallelversetzung der Fahrzeugposition bewirkt. Die geplante Trajektorie wird qualitativ eine Form wie in Abbildung 1.7a oben aufweisen. Optimal wäre aber eine Trajektorie wie Abbildung 1.7b. Schlussendlich wird auch das Fächerverfahren hinter dem Hindernis einscheren (Abbildung 1.7a unten), dies konnte aber zum initialen Zeitpunkt nicht antizipiert werden. Stellt man sich vor, dass in einem ähnlichen Szenario andere Fahrzeuge auf der Gegenspur entgegenkommen, so wird man einsehen, dass die zusätzliche „Weitsicht“ eines Optimalverfahrens erstrebenswert ist.

In dieser Arbeit wurde, aufgrund der speziellen Anforderungen beim Fahren in fließendem Verkehr, der kombinatorische Ansatz für diesen Anwendungsfall verworfen. Stattdessen wird ein kontinuierliches Optimierungsschema implementiert, bei dem der Arbeitsraum nicht diskretisiert werden muss. Der Lösungsraum der darstellbaren Trajektorien ist unendlich, die Genauigkeit wird nicht durch Quantisierungsfehler bei der Diskretisierung von vornherein eingeschränkt. Die Methode ist auch nicht vom exponentiellen Anstieg der Laufzeit in Bezug auf die Dimensionalität des Phasenraumes betroffen, es können deshalb Gütefunktionale hoher Ordnung (die

zum Beispiel den Ruck enthalten) verwendet werden.

### 1.3 Zielsetzung der Arbeit

In der vorliegenden Arbeit sollen Verfahren zur Bahn- und Trajektorienplanung entwickelt werden. Die Verfahren sollen für die Führung eines vorderadgelenkten Automobils eingesetzt werden. Folgende Anforderungen werden an die Verfahren gestellt:

- Die Verfahren müssen nichtholonome Einschränkungen, wie sie sich aus der Kinematik des Fahrzeugmodelles ergeben, berücksichtigen.
- Die Verfahren sollen in einem wohldefinierten Sinne optimal sein. Das Optimalitätskriterium soll dabei aus fahrdynamischen Anforderungen abgeleitet werden. Es wird davon ausgegangen, dass hierzu Kostenfunktionale höherer Ordnung berücksichtigt werden müssen.
- Die Verfahren sollen vollständig sein.
- Es soll eine Art der Umgebungsrepräsentation gewählt und implementiert werden, die geeignet ist, Schnittstellen mit der Fahrzeugsensorik und mit übergeordneten Entscheidungsprozessen (Verhaltensebene) herzustellen.
- Die entwickelten Verfahren sollen in ihrer Summe die eingangs genannten Anwendungsszenarien „*Fahren in statischer, unstrukturierter Umgebung*“ und „*Fahren in dynamischer, strukturierter Umgebung*“ abdecken.
- Die entwickelten Verfahren sollen anhand realer Fahrversuche validiert werden.

## 2 Grundlagen

### 2.1 Graphentheorie

Insbesondere im Kapitel 6 über kombinatorische Verfahren zur Bahn- und Trajektorienplanung werden Methoden der Graphentheorie zum Einsatz kommen. Dieser Abschnitt dient dazu, die wichtigsten Grundbegriffe und -algorithmen, insbesondere aber eine für diese Arbeit verbindliche Notation, einzuführen. Die Notation ist im wesentlichen an Volkmann [Vol96] angelehnt.

#### 2.1.1 Graphen

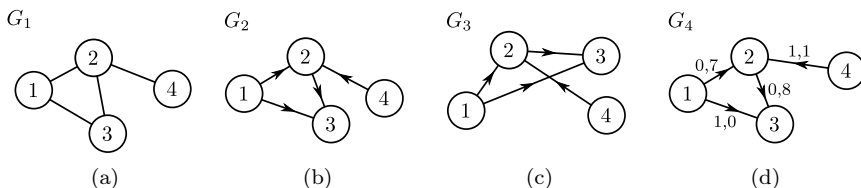
Man unterscheidet zwischen *ungerichteten* und *gerichteten* Graphen (Abbildung 2.1a-c). Für diese Arbeit sind nur gerichtete Graphen (*Digraphen*, *directed graphs*) relevant, deshalb wird diese Einführung auf diese Klasse beschränkt. Wird im folgenden nur von einem „Graphen“ gesprochen, so ist implizit ein gerichteter Graph gemeint.

Ein gerichteter Graph  $G$  ist ein Tupel

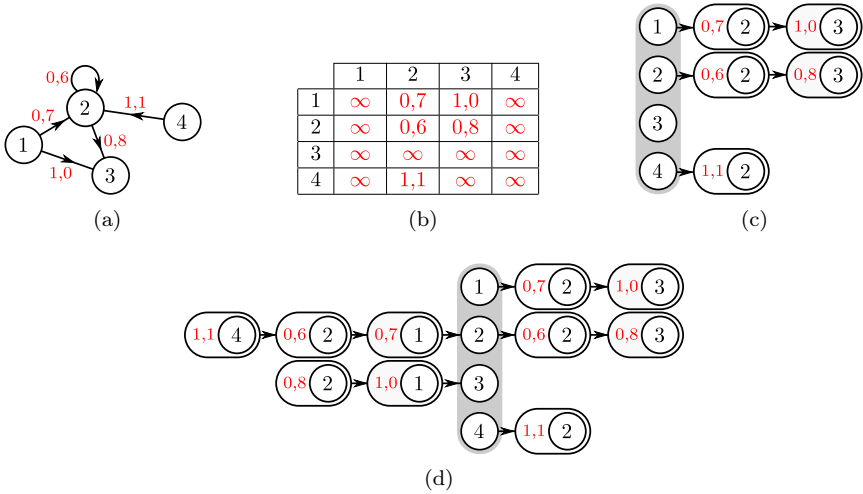
$$G = (V_G, E_G), \quad (2.1)$$

mit

$$E_G \subseteq V_G \times V_G. \quad (2.2)$$



**Abbildung 2.1:** Graphen. (a) ungerichtet. (b),(c) gerichtet. (d) gerichtet, gewichtet.



**Abbildung 2.2:** Graphrepräsentationen. (a) Diagramm. (b) Adjazenzmatrix. (c) Adjazenzlisten, ausgehend. (d) Adjazenzlisten, ein- und ausgehend.

Die Menge  $V_G$  enthält die *Knoten* (vertices) des Graphen, und die Menge  $E_G$  seine *Kanten* (edges). Eine Kante  $e$  ist ein geordnetes Paar  $(v, w)$  aus Knoten  $v, w \in V_G$ . Wenn  $(v, w) \in E_G$ , so sind die Knoten  $v$  und  $w$  mit einer Kante (Pfeil) verbunden, wobei  $v$  am Anfang und  $w$  am Ende der Kante steht. Der Graph  $G_2$  in Abbildung 2.1b kann so beschrieben werden:

$$\begin{aligned}
 G_2 &= (V_{G_2}, E_{G_2}) \\
 V_{G_2} &= \{1, 2, 3, 4\} \\
 E_{G_2} &= \{(1, 2), (1, 3), (2, 3), (4, 2)\}.
 \end{aligned} \tag{2.3}$$

Kanten eines Graphen können über eine Funktion  $w_G : E_G \mapsto \mathbb{R}$  gewichtet sein (Abbildung 2.1d).

Es gibt verschiedene Möglichkeiten, einen Graphen im Rechner zu repräsentieren. Abbildung 2.2a zeigt das Diagramm eines Graphen. Abbildung 2.2b zeigt diesen als Adjazenzmatrix repräsentiert, und Abbildung 2.2c-d die Darstellung mit Hilfe von Adjazenzlisten. Die Repräsentation als Adjazenzlisten hat den Vorteil, dass, gegeben einen Knoten  $v$ , die ein- bzw. ausgehenden Kanten schnell bestimmt werden können. Hierzu werden die Funktionen  $\text{in}_G, \text{out}_G : V_G \rightarrow \mathcal{P}(\mathbb{R} \times V_G)$  eingeführt, die jedem Knoten die durch



ein- beziehungsweise ausgehende Kanten erreichbaren Knoten, zusammen mit den dafür erforderlichen Kosten, zuordnet. Unter der Annahme, dass der Graph durch Adjazenzlisten für ein- und ausgehende Knoten repräsentiert wird, können  $\text{in}_G$  und  $\text{out}_G$  in konstanter Zeit berechnet werden.

## 2.1.2 Geometrische Graphen

Ein Graph stellt zunächst eine abstrakte, rein topologische Struktur dar. Wenn man die formale Beschreibung 2.3 mit dem Diagramm in Abbildung 2.1b vergleicht, so stellt man fest, dass keine Beschreibung der Position der Knoten enthalten ist, alle Diagramme, die die gleiche Kanten- und Eckenmenge aufweisen, beschreiben also äquivalente Graphen, und bei genauerem Hinsehen stellt man fest, dass für die Graphen aus Abbildung 2.1b,c gilt  $G_2 \equiv G_3$ .

Soll, wie in der Bahnplanung, ein Graph  $G$  eine *geometrische* Struktur beschreiben, so versieht man ihn mit einer *Einbettung*  $\mathbf{x}_G : V_G \mapsto \mathbb{R}^2$ , die jedem Knoten einen Ort zuordnet (hier ohne Beschränkung der Allgemeinheit im zweidimensionalen). Ein *geometrischer Graph*  $G$  ist dann das Tupel

$$G = (V_G, E_G, \mathbf{x}_G). \quad (2.4)$$

## 2.1.3 Kürzeste Pfade

Ein (gerichteter) *Weg* der Länge  $n$  im Graphen  $G$  ist eine Folge von Knoten  $W = (v_1, v_2, \dots, v_n), v_i \in V_G$ , so dass aufeinanderfolgende Knoten durch eine Kante verbunden sind:  $(v_i, v_{i+1}) \in E_G, \forall i \in \{1, \dots, n-1\}$ .

Ein *Pfad* ist ein Weg, bei dem alle Knoten voneinander verschieden sind. Ein  $v$ - $w$ -Weg bzw.  $v$ - $w$ -Pfad, ist ein Weg bzw. Pfad, dessen erster Knoten  $v$  und dessen letzter Knoten  $w$  ist. Diese Definition kann auch auf *Mengen* von Anfangs- und Endknoten ausgeweitet werden, ein  $A$ - $B$ -Weg ist dann ein Weg  $(v_1, v_2, \dots, v_n)$  mit  $v_1 \in A$  und  $v_n \in B$ .

Der *kürzeste  $v$ - $w$ -Pfad* in einem kantengewichteten Graphen ist der  $v$ - $w$ -Pfad, für den die Summe der Kantengewichte minimal wird.

Der bekannteste Algorithmus zur Berechnung kürzester Pfade ist der nach Dijkstra. Der Algorithmus berechnet für einen ausgezeichneten Startknoten  $v_s$  aus dem Graphen  $G$  den kürzesten  $v_s$ - $w$ -Pfad für *alle* Knoten  $w \in V_G$  (das *single source shortest path problem*, SSSP). Der Algorithmus verwendet das Prinzip des *dynamischen Programmierens*. Er bedient sich als Datenstruktur

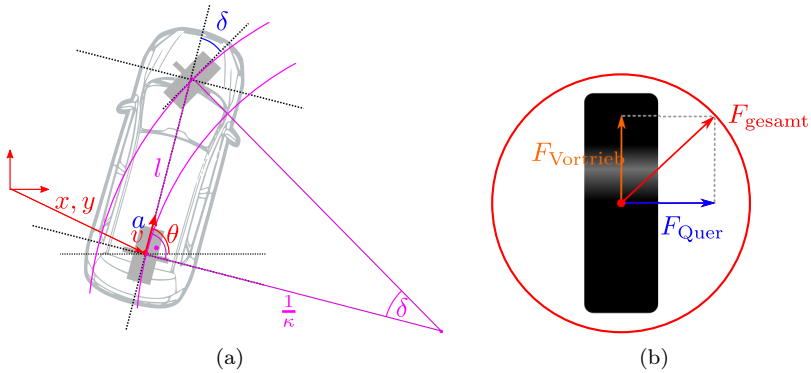
einer Vorrangwarteschlange (priority queue). Bei dieser werden die Elemente in einer Halbordnung vorgehalten, was bedeutet, dass das *kleinste* Element in der Warteschlange stets schnell (in konstanter Zeit) gefunden werden kann. Die Implementierung dieser Datenstruktur ist für die Effizienz des Algorithmus entscheidend, in praktischen Implementierungen bedient man sich meistens *binärer Heaps*.

Der Ablauf des Algorithmus ist wie folgt:

1. Für den Algorithmus wird jeder Knoten mit der zusätzlichen Eigenschaft „Distanz“ versehen.
2. Die Distanz des Startknotens wird mit 0 initialisiert, alle anderen mit Unendlich.
3. Der Startknoten wird in die Warteschlange eingefügt.
4. Solange die Warteschlange nicht leer ist
  - (a) wähle den Knoten mit der geringsten Distanz aus der Warteschlange.
  - (b) entferne den Knoten aus der Warteschlange.
  - (c) für alle Ausgangskanten des gewählten Knotens
    - i. Berechne die Summe aus Kantengewicht und Distanz des gewählten Knotens
    - ii. Ist diese Summe kleiner als die Distanz des Endknotens der gewählten Kante, aktualisiere die Distanz des Ausgangsknotens mit der Summe und füge den Ausgangsknoten in die Warteschlange ein.

Das Ergebnis dieses Algorithmus ist zunächst ausschließlich eine Funktion, die jedem Knoten die Länge des kürzesten Pfades, vom Startknoten aus, zuordnet. Diese Funktion wird im Folgenden mit  $d^* : V_G \mapsto \mathbb{R}$  bezeichnet. Aus  $d^*$  kann der kürzeste Pfad leicht rekonstruiert werden, wobei man rückwärts, beginnend beim Zielknoten, vorgeht:

1. initialisiere eine Liste  $l$  mit dem Zielknoten  $v_{\text{ziel}}$ .
2. Solange der Startknoten nicht erreicht ist,
  - (a) wähle den letzten Knoten  $v$  in  $l$
  - (b) für alle Eingangskanten des gewählten Knotens



**Abbildung 2.3:** (a) Kinematik eines Fahrzeuges mit Vorderradlenkung. (b) Kammscher Reibkreis.

- i. sei  $e$  die gewählte Kante
- ii. sei  $v_{\text{test}}$  der Startknoten von  $e$ .
- iii. wenn  $d^*(v_{\text{test}}) = d^*(v) + w_G(e)$ , hänge  $v_{\text{test}}$  an  $l$  an und mache weiter bei 2.

3.  $l$  enthält einen kürzesten Pfad von  $v_{\text{start}}$  nach  $v_{\text{ziel}}$ , in umgekehrter Reihenfolge.

Die Laufzeit von Dijkstras Algorithmus ist in  $\mathcal{O}(|E_G| + |V_G| \log(|V_G|))$ . Für allgemeine Graphen ist dies auch die beste erzielbare Laufzeit [AMO93]. Für spezielle Formen von Graphen lassen sich effizientere Algorithmen finden. In Abschnitt 6.3 wird ein Algorithmus vorgestellt, der für eine in der Trajektorienplanung wichtige Klasse von Graphen eine Laufzeit in  $\mathcal{O}(|E_G| + |V_G|)$  aufweist.

## 2.2 Fahrzeugmodellierung

Vorderradgelenkte Fahrzeuge sind nichtholonome Systeme, die außerdem dynamischen Beschränkungen unterworfen sind. Sowohl für die Trajektorienplanung als auch für den Entwurf eines stabilisierenden Reglergesetzes ist es deshalb erforderlich, sich mit ihren kinematischen und dynamischen Beschränkungen zu befassen.

Es wird zunächst vom *kinematischen Einspurmodell* ausgegangen, das eine gute Näherung eines vorderradgelenkten Fahrzeuges darstellt, das sich im langsamen Geschwindigkeitsbereich ( $< 15\text{km/h}$ ) bewegt. Die Lenkkineematik dieses Modells ist in Abbildung 2.3a illustriert. Die Zustandsgrößen des Modells sind in der Abbildung rot eingefärbt, es handelt sich um die Position  $\mathbf{x} = (x, y)^\top$ , die Ausrichtung  $\theta$  und die Längsgeschwindigkeit  $v$  des Fahrzeuges. Die Eingänge sind blau hervorgehoben, es handelt sich um den Vorderradwinkel  $\delta$  und die Längsbeschleunigung  $a$ .

Bei diesem Modell ist die Eigenschaft der *differenziellen Flachheit* [FSR05] wesentlich, da sie in einem gewissen Sinne erlaubt, vom System und seinen inneren Zuständen zu abstrahieren. Flache Systeme besitzen einen Ausgang (*flacher Ausgang*), so dass sich sämtliche Zustands- und Eingangsgrößen aus diesem Ausgang und endlichen vielen seiner Zeitableitungen herleiten lassen.

Für den Fall des kinematischen Einspurmodells ist der *Hinterachspunkt* - also der Punkt in der Mitte der Hinterachse - ein flacher Ausgang. Wenn  $\mathbf{x}(t) = (x(t) \ y(t))^\top$  eine Trajektorie für diesen Punkt beschreibt, so können alle weiteren Systemzustände und Eingänge hergeleitet werden. Für Längsgeschwindigkeit, -beschleunigung und die Ausrichtung des Fahrzeuges gilt einfach<sup>1</sup>

$$\begin{aligned} v &= \|\dot{\mathbf{x}}\|, \\ a &= \|\ddot{\mathbf{x}}\|, \\ \theta &= \arctan \frac{\dot{y}}{\dot{x}}. \end{aligned} \tag{2.5}$$

Der benötigte Vorderradwinkel lässt sich aus der Dreiecksbeziehung in Abbildung 2.3a herleiten:

$$\delta = \arctan l\kappa,$$

wobei  $l$  der Achsabstand und  $\kappa$  die Krümmung

$$\kappa = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{(\dot{x} + \dot{y})^2} \tag{2.6}$$

---

<sup>1</sup>Gleichung 2.5 gilt - ohne Beschränkung der Allgemeinheit - wegen der Periodizität des Arkustangens nur unter der Annahme  $\frac{\dot{y}}{\dot{x}} > 0$ . In vielen Programmiersprachen gibt es die Funktion `atan2(a, b)`, um den Ausdruck  $\arctan \frac{b}{a}$  zu ersetzen. Sie führt die notwendige Fallunterscheidung nach den Vorzeichen ihrer Argumente durch. Für den Randfall  $\dot{x} = \dot{y} = 0$  lässt sich keine Ausrichtung des Fahrzeuges bestimmen.

der Hinterachspunktstrajektorie ist.

Diese Betrachtung gilt nur für langsame Geschwindigkeiten, bei denen am Hinterachspunkt kein Schwimmwinkel auftritt. Im Folgenden wird aber davon ausgegangen, dass, auch bei höheren Geschwindigkeiten, der Schwimmwinkel des Fahrzeuges beim Folgen einer geplanten Trajektorie klein ist und deshalb aus Sicht der Trajektorienplanung vernachlässigt werden kann (nicht jedoch auf der Ebene des nachgeschalteten Trajektorienfolgereglers, [Wer12]). In [FSR05] wurde gezeigt, dass auch ein dynamisches Einspurmodell mit Schwimmwinkel einen flachen Ausgang besitzt, von dem dann die ersten drei Ableitungen betrachtet werden müssen, um alle Fahrzeugzustände und -eingänge rekonstruieren zu können.

## 2.3 Finite Differenzen

In diesem Kapitel wird eine Notation für finite Differenzen eingeführt. Finite Differenzen werden in dieser Arbeit häufig verwendet, um Ableitungen kontinuierlicher Funktionen zu approximieren.

Im Folgenden wird die Funktion  $x(t) : \mathbb{R} \rightarrow \mathbb{R}$  betrachtet.

Die Vorwärtsdifferenz mit Schrittweite  $h$  ist

$$\Delta_h[x](t) = x(t+h) - x(t), \quad (2.7)$$

die Rückwärtsdifferenz ist

$$\nabla_h[x](t) = x(t) - x(t-h) \quad (2.8)$$

und die Zentraldifferenz ist

$$\delta_h[x](t) = x\left(t + \frac{1}{2}h\right) - x\left(t - \frac{1}{2}h\right). \quad (2.9)$$

Für  $h \rightarrow 0$  ist (unter einfachen Stetigkeitsbedingungen) der Grenzwert der drei Terme

$$\frac{\Delta_h[x](t)}{h}, \frac{\nabla_h[x](t)}{h}, \frac{\delta_h[x](t)}{h} \quad (2.10)$$

die Ableitung  $\dot{x}$ . Ableitungen höherer Ordnung kann man approximieren, indem man die Gleichungen oben mehrmals anwendet. Auch hierfür wird eine Notation eingeführt. Wird zum Beispiel der Zentraldifferenzoperator aus (2.9) zweimal hintereinander ausgeführt, so verwendet man hierfür die

Schreibweise  $\delta_h^2[x](t)$ . Für die Approximation der zweiten Ableitung gilt dann beispielsweise

$$\begin{aligned} \ddot{x}(t) &\approx \frac{\delta_h^2[x](t)}{h^2} \\ &= \frac{\delta_h\left[\frac{\delta_h[x]}{h}\right](t)}{h} \\ &= \frac{\frac{\delta_h[x](t+\frac{1}{2}h)}{h} - \frac{\delta_h[x](t-\frac{1}{2}h)}{h}}{h} \\ &= \frac{x(t+h) - 2x(t) + x(t-h)}{h^2}. \end{aligned}$$

In den meisten Fällen wird  $x$  an äquidistanten Stellen  $t_i = hi, i \in \mathbb{N}$  abgetastet,  $h$  ist also konstant. Man bedient sich dann der vereinfachten Schreibweise

$$\delta x_i = \delta_h[x](ih). \quad (2.11)$$

## 3 Kontinuierliche Distanztransformationen

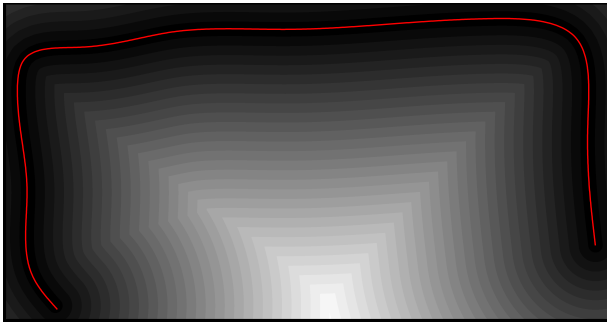
Für einige Verfahren im nächsten Kapitel wird es nötig sein, für jeden beliebigen Punkt  $\mathbf{x} = (x, y)^\top$  des Arbeitsraumes den euklidischen Abstand  $d(\mathbf{x}, G)$  zum Rand einer Geometrie  $G$  zu bestimmen, beispielsweise zu Hindernissen, Sollbahnen oder zu anderen Verkehrsteilnehmern. Eine Geometrie  $G \subset \mathbb{R}^2$  bezeichnet im Folgenden eine beliebige abgeschlossene Teilmenge der euklidischen Ebene. Im folgenden wird  $d(\mathbf{x}, G)$  als die *Distanztransformation* von  $G$  bezeichnet. Insbesondere für die lokalen, NEWTON-artigen Verfahren ist es wichtig, dass  $d$  bezüglich  $\mathbf{x}$  stückweise differenzierbar ist. Nur für einfache Geometrien  $G$  lässt sich  $d$  einfach bestimmen, sei  $G$  zum Beispiel eine Scheibe mit Radius  $r$ , deren Zentrum sich im Ursprung befindet, dann gilt

$$d(\mathbf{x}, G) = \left| \sqrt{x^2 + y^2} - r \right|. \quad (3.1)$$

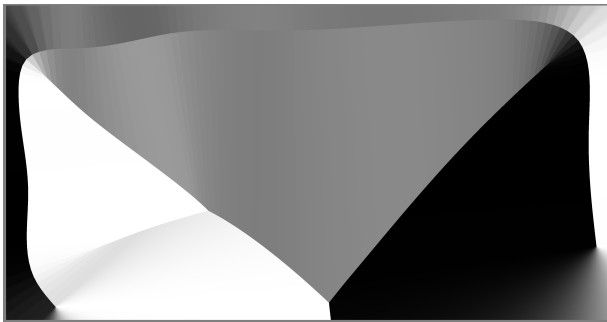
In diesem Kapitel werden Distanztransformationen für wichtige Geometrieklassen vorgestellt, nämlich für diskrete, gitterförmige Karten (*occupancy grids*) und für Mengen von Liniensegmenten. Diskrete Karten werden verwendet, um statische Hindernisse zu repräsentieren, und Liniensegmente eignen sich, um Strassennetzwerke oder Fahrwege topologisch und geometrisch zu repräsentieren. Diese Distanztransformationen haben folgende wichtige Eigenschaften:

- Sie sind kontinuierlich auf dem gesamten Arbeitsraum definiert und dadurch auch (stückweise) differenzierbar.
- Sie beruhen auf einer Vorberechnung, deren Aufwand linear mit der Komplexität der Geometrie wächst.
- Sie können nach der Vorberechnung in konstanter Zeit ausgewertet werden.

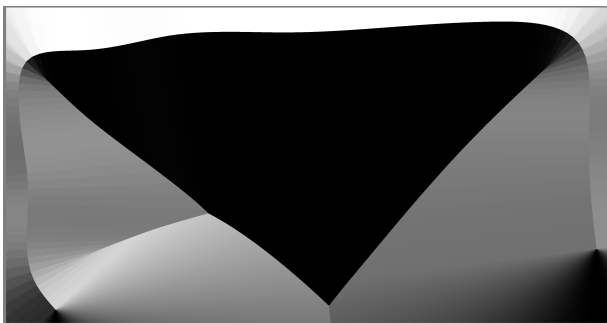
Abbildung 3.1 nimmt ein Ergebnis vorweg, um den Begriff der Distanztransformation zu erläutern. In Abbildung 3.1a ist als Beispielgeometrie  $G_{\text{Beispiel}}$  ein linienförmiges Straßensegment in rot eingezeichnet. Im gleichen Bild ist



(a)



(b)



(c)

**Abbildung 3.1:** (a): Distanztransformation eines Straßensegments (rot). (b): Ableitung entlang der horizontalen Richtung von (a). (c): Ableitung entlang der vertikalen Richtung von (a).



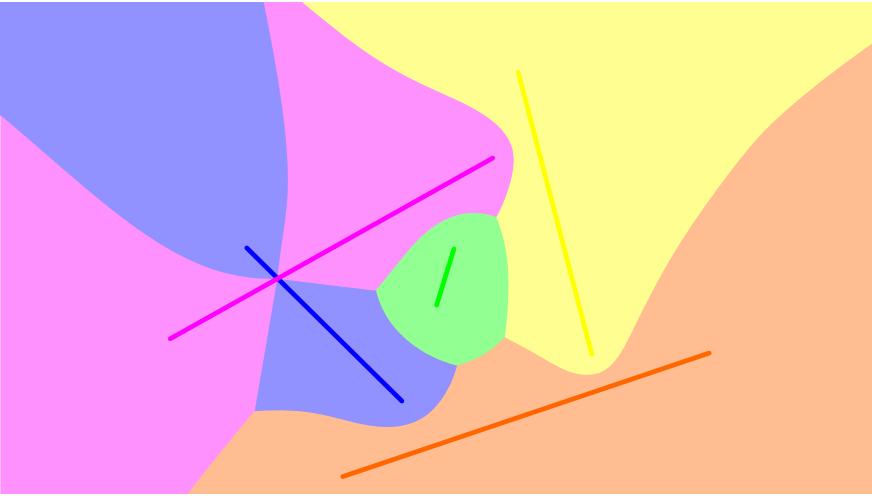
als Grauton - niedrige Werte sind dunkel, hohe hell - der Abstand zu diesem Straßensegment dargestellt, dies entspricht der Funktion  $d(\mathbf{x}, G_{\text{Beispiel}})$ . Um einige Konturlinien hervorzuheben, wurde die Anzahl der Grauwerte stark reduziert. Die Abbildungen 3.1b und 3.1c zeigen die partiellen Ableitungen von  $d(\mathbf{x}, G_{\text{Beispiel}})$  nach  $x$  und  $y$ .

Die meisten bisher veröffentlichten Verfahren Fabbri u. a. [Fab+08] zur euklidischen Distanztransformation sind diskreter Natur, und zwar in dem Sinne, dass sie sowohl die Hindernisse als auch die Orte  $\mathbf{x}$ , an denen die Distanz bestimmt wird, auf ein Gitter in  $\mathbb{R}^2$  beschränken: Sie erhalten ein digitales Bild als Eingabe, in dem einige Pixel als Hindernisse markiert sind, und bestimmen dann für jedes Pixel des Bildes den Abstand zum nächstgelegenen Hindernisspixel.

Das hier vorgestellte Verfahren ist eine Erweiterung eines solchen diskreten Verfahrens. Es erlaubt die exakte Auswertung der Distanztransformation auch „zwischen“ den ganzzahligen Pixelkoordinaten. Ausgangspunkt ist Danielssons 8SED (*Sequential Euclidean Distance Map*)-Verfahren Danielsson [Dan80]. Dieses macht sich eine Lokalitätseigenschaft der Distanztransformation zu nutze: Das einem Punkt nächstgelegene Liniensegment kann aus den Liniensegmenten bestimmt werden, die Punkten in der Nachbarschaft am nächsten gelegen sind.

Sei im folgenden zunächst die Geometrie  $G$  eine Menge von Liniensegmenten,  $G = \{l_1, l_2, \dots\}$ . Der kürzeste Abstand zwischen einem Punkt  $\mathbf{x}$  und einem Liniensegment  $l$  sei  $d(\mathbf{x}, l)$ . Weiterhin ordne die Funktion  $l_{\min}(\mathbf{x}) : \mathbb{R}^2 \rightarrow G$  jedem Punkt  $\mathbf{x}$  das ihm nächstgelegene Liniensegment aus  $G$  zu. Zunächst ist festzustellen, dass  $d(\mathbf{x}, l)$  für ein gegebenes  $l$  schnell berechnet werden kann, und dass dies auch für die Ableitungen  $\frac{d}{dx}d$  und  $\frac{d}{dy}d$  gilt. Daraus folgt, dass eine schnelle Berechnung von  $l_{\min}(\mathbf{x})$  ausreicht, um eine Distanztransformation mit den geforderten Eigenschaften zu realisieren. Die Distanztransformation  $d(\mathbf{x}, G)$  ist dann genau  $d(\mathbf{x}, l_{\min}(\mathbf{x}))$ .

$l_{\min}$  zerlegt  $\mathbb{R}^2$  in die sogenannten VORONOJ-Regionen. Die VORONOJ-Region der Linie  $l$  enthält genau die Punkte aus  $\mathbb{R}^2$ , die zu  $l$  einen kleineren Abstand haben als zu allen anderen Linien in  $G$ . Abbildung 3.2 zeigt ein Beispiel für diese Zerlegung. Die Distanztransformation ist innerhalb dieser Regionen stetig differenzierbar. Es wird nun angenommen, dass  $l_{\min}(\mathbf{p})$  für alle Punkte  $\mathbf{p}$  eines ortsfesten, äquidistanten Gitters schon bekannt ist. Später wird gezeigt werden, dass sich dies schnell – in linearer Zeit – erreichen lässt. Abbildung 3.3a zeigt, für einen Ausschnitt aus 3.2, das vorberechnete Gitter, wobei  $l_{\min}$  durch farbliche Kodierung der Punkte wiedergegeben ist. Um  $l_{\min}(\mathbf{x})$  schnell zu berechnen, wählt man nun eine



**Abbildung 3.2:** Voronoi-Regionen. Die Schattierung jeder Voronoi-Region entspricht der Farbe der zugehörigen Linie.

endlich grosse Nachbarschaft  $N = \{\mathbf{p}_0, \mathbf{p}_1, \dots\}$  aus diesem Gitter. Abbildung 3.3a zeigt eine 16-er Nachbarschaft  $N$ , mit der  $l_{\min}(\mathbf{x})$  für alle  $\mathbf{x}$  im eingerahmten Bereich berechnet werden kann. Jedem  $\mathbf{x}$  in diesem Bereich wird die nächstgelegene Linie zugeordnet, wobei nur die Linien betrachtet werden, die mindestens einem der Punkte aus  $N$  am nächsten liegen, im Beispiel also die Blaue, die Grüne und die Rote. Allgemein sei

$$L_N = \{l \mid l = l_{\min}(\mathbf{p}) \text{ für einen Punkt } \mathbf{p} \in N\} \quad (3.2)$$

diese eingeschränkte Menge an Linienkandidaten. Es wird die Näherungslösung

$$\check{l}_{\min}(\mathbf{x}) = \arg \min_{l \in L_N} d(\mathbf{x}, l). \quad (3.3)$$

für  $l_{\min}$  definiert, die an allen Punkten verwendet wird, die nicht exakt auf dem Gitters liegen. Abbildung 3.3b zeigt einen Fall, für den die 16er-Nachbarschaft bei der gewählten Gitterauflösung nicht ausreicht, um alle Punkte im eingerahmten Bereich richtig zuzuordnen. Die grüne Region wird von der Nachbarschaftsmenge nicht erfasst, weshalb die Punkte innerhalb des gelb markierten Keils falsch zugeordnet werden. Trotzdem erreicht das Verfahren mit der 16er-Nachbarschaft eine sehr gute Annäherung an die Distanztransformation, die nur für einen kleine Anteil der Ebene nicht exakt ist.

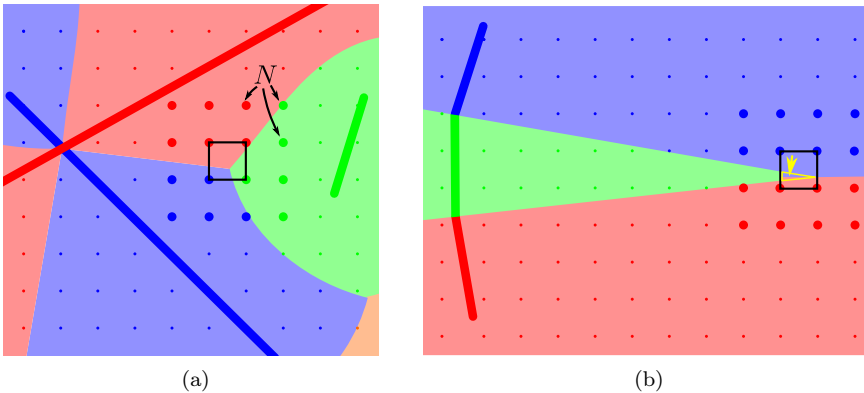


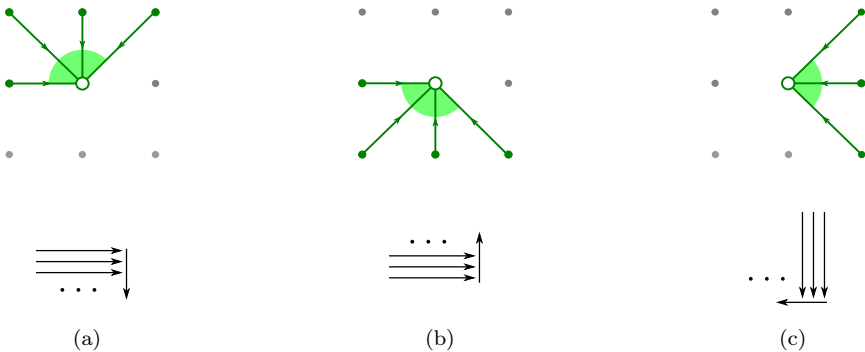
Abbildung 3.3: Lokalitätseigenschaft der Distanztransformation.

### Vorbereitung des Gitters

Die bisherigen Überlegungen gingen davon aus, dass  $l_{\min}$  an den Gitterpunkten bereits bekannt ist. Es wird nun gezeichnet, wie  $l_{\min}$  an diesen Punkten schnell berechnet werden kann. Das Grundprinzip heißt *Vektorpropagation* und wurde von DANIELSSON [Dan80] für diskrete Distanztransformationen entwickelt.

Zur Illustration werden zunächst nur solche Geometrien betrachtet, die aus punktförmigen Basisgeometrien zusammengesetzt sind. Das Verfahren lässt sich aber direkt verallgemeinern auf solche Mengen, die auch andere Basisgeometrien  $g$  enthalten, solange sich für diese  $d(\cdot, g) = d(\mathbf{x}, g)$  schnell berechnen lässt.

Bezeichne im folgenden  $\mathbf{x}_{ij}$  den Ortsvektor des Gitterpunktes in der  $i$ -ten Zeile und der  $j$ -ten Spalte des Gitters. Für das Gitter wird das zweidimensionale Feld  $L$  angelegt, das für jeden Gitterpunkt eine temporäre Zuordnung zu einer Basisgeometrie enthält. Nach Ablauf des Algorithmus enthält  $L$  die gesuchte Zuordnung, also  $L_{ij} = l_{\min}(\mathbf{x}_{ij})$ .  $L$  wird insgesamt dreimal durchlaufen, wobei die temporären Labels  $L_{ij}$  jeweils basierend auf Teilnachbarschaften (*Masken*) wie in den Abbildungen 3.4a-c aktualisiert werden. Wichtig ist hierbei, in welcher Reihenfolge die Gitterpunkte besucht werden, die Reihenfolge ist in den Abbildungen 3.4a-c jeweils unter der Maske abgebildet. Mit Maske „maske\_a“ beispielsweise wird  $L$  also zeilenweise, von oben nach unten, besucht. Die einzelnen Zeilen werden dabei von links nach rechts bearbeitet. Wie man sieht, stellt die Bearbeitungsrei-



**Abbildung 3.4:** Masken für 8SED mit Durchlaufrichtung (a): `maske_a`. (b): `maske_b`. (c): `maske_c`.

---

```

maske_a = [( 1, 0), ( 1, 1), ( 0, 1), (-1, 1)]
maske_b = [( 1, 0), ( 1,-1), ( 0,-1), (-1,-1)]
maske_c = [(-1, 1), (-1, 0), (-1,-1)]

```

```

def besuche( i, j, maske ):
    for (di, dj) in maske:
        d_alt = distanz( x[i][j], L[i][j] )
        d_neu = distanz( x[i][j], L[i-di][j-dj] )
        if d_neu < d_alt:
            L[i][j] = L[i-di][j-dj]

```

```

def besuche_alle():
    for j in range( 0, zeilen ):
        for i in range( 0, spalten ):
            besuche( i, j, maske_a )

    for j in reversed( range( 0, zeilen ) ):
        for i in range( 0, spalten ):
            besuche( i, j, maske_b )

    for i in reversed( range( 0, spalten ) ):
        for j in range( 0, zeilen ):
            besuche( i, j, maske_c )

```

henfolge, zusammen mit der Gestalt der Masken, sicher, dass nur solche Gitterpunkte zur Bestimmung einer Zuordnung herangezogen werden, die im gleichen Durchlauf zuvor schon bearbeitet wurden. Die drei Durchläufe propagieren auf diese Weise Information nur in bestimmte Richtungen. Diese Richtungen decken jeweils einen der Sektoren ab, die in Abbildung 3.4 jeweils grün hinterlegt sind. Wie man sieht, ergänzen sich die Sektoren zu einem vollen Kreis.

Programmtext 3.1 stellt den Algorithmus dar. Die Ortsvektoren  $\mathbf{x}_{ij}$  und die temporären Markierungen  $L_{ij}$  sind im Programmtext durch die Variablen `x[i][j]` und `L[i][j]` repräsentiert. Die Funktion `distanz(x, g)` gibt den Abstand zwischen dem Ortsvektor  $\mathbf{x}$  und der Basisgeometrie  $\mathbf{g}$  zurück. Alle Einträge der Matrix `L` sind zu Beginn mit dem Wert „ungültig“ initialisiert, und `distanz(x, „ungültig“)` gibt immer Unendlich zurück.

Abbildungen 3.5a-d illustrieren an einem Beispiel den Ablauf des Algorithmus. Die Geometrie besteht hier aus drei Punkten. Abbildung 3.5a zeigt das Feld `L` nach der Initialisierung. Als Verständnishilfe sind die Bisektoren von jeweils zwei Punkten eingetragen, die durchgezogenen Abschnitte dieser Linien bilden die Voronoi-Linien. Abbildung 3.5b zeigt das Ergebnis nach dem ersten Durchlauf mit der Maske `maske_a`. Das Ergebnis kann so interpretiert werden: Jeder Punkt  $L_{ij}$  ist der Geometrie zugeordnet, die ihm am nächsten liegt, wobei allerdings nur Geometrien betrachtet werden, die bezüglich  $L_{ij}$  in dem in Abbildung 3.4a grün markierten Sektor, also  $45^\circ$ - $180^\circ$ , liegen. Abbildung 3.5c-d zeigen die Anwendung der beiden verbleibenden Masken. Die Masken werden dabei jeweils auf das Zwischenergebnis des vorherigen Durchlaufes angewendet.

Aus Gründen der Anschaulichkeit wurden im Beispiel Punkte als Basisgeometrien verwendet. Sollen andere Geometrien verwendet werden, so muss lediglich die Initialisierungsphase angepasst werden. Abbildung 3.6 zeigt, wie eine Linie initialisiert werden muss. Die gezeigte Treppenlinie kann mit einer Variante des bekannten Algorithmus nach BRESENHAM zum Zeichnen gerasterter Linien erzeugt werden.

## Vorzeichenbehaftung

Meistens ist es zweckmäßig, die Distanz mit einem Vorzeichen zu versehen, und zwar derart, dass sie negativ ist für alle Punkte die sich innerhalb der Geometrie befinden, und positiv für alle Punkte außerhalb. Man spricht dann von einer *vorzeichenbehafteten Distanztransformation*.

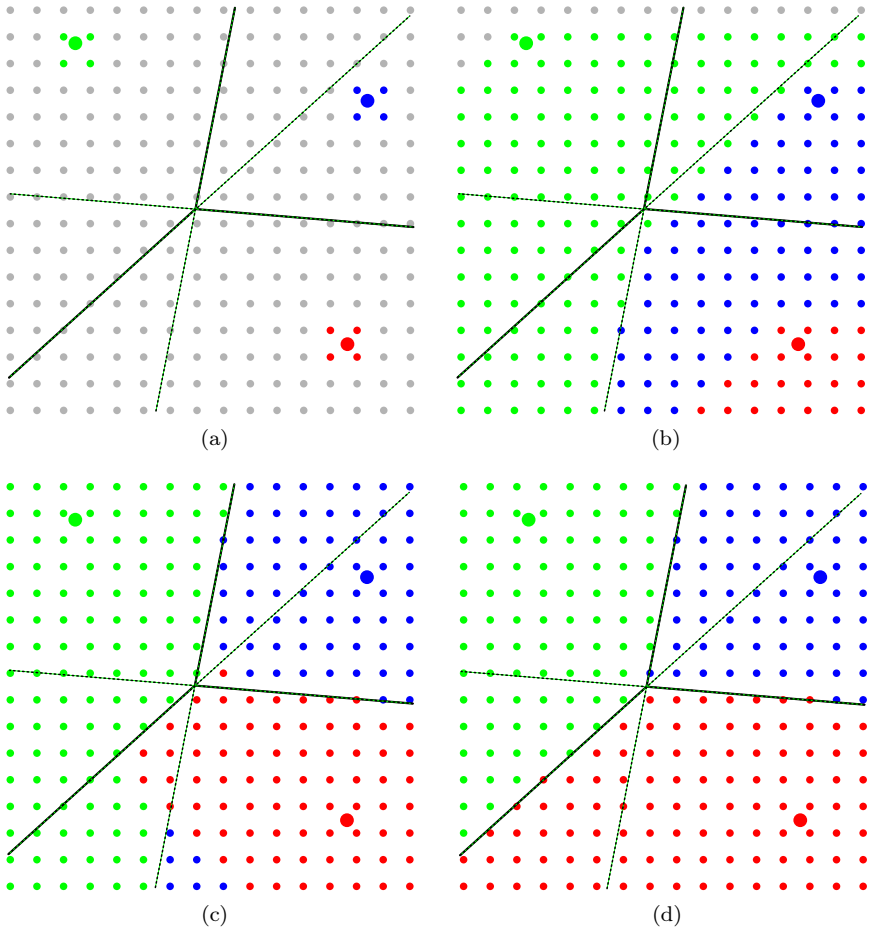


Abbildung 3.5: Durchläufe von 8SED, Erläuterungen im Text

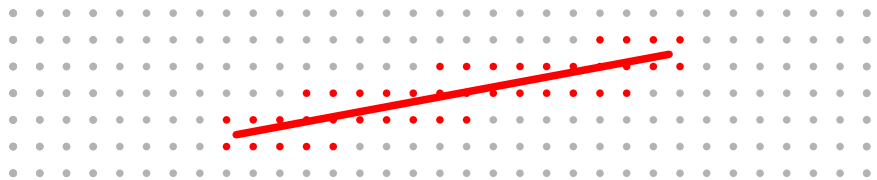


Abbildung 3.6: 8SED: Initialisierung einer Linie.

## Stetig differenzierbare Pseudodistanz

Für Optimierungsverfahren des NEWTON-Typs muss die Abstandsfunktion im Arbeitsbereich stetig differenzierbar sein. Für einen Polygonzug ist das im Allgemeinen nicht der Fall. Da die meisten in dieser Arbeit verwendeten Geometrien über Polygonzüge beschrieben werden, wird in diesem Abschnitt eine Pseudodistanz definiert, die die erforderlichen Stetigkeitsbedingungen erfüllt. Es wird davon ausgegangen, dass mit dem Polygonzug selber versucht wird, eine (im Allgemeinen unbekannte) glatte Kurve zu approximieren, die die Eckpunkte des Polygonzuges interpoliert. Es wird davon ausgegangen, dass nicht nur Punkte bekannt sind, die von der glatten Ausgangskurve interpoliert werden, sondern auch, unter welchem Tangentenwinkel sie interpoliert werden. Ziel des im folgenden beschriebenen Verfahrens ist nicht, diese glatte Ausgangskurve approximativ zu rekonstruieren, sondern lediglich ihre Abstandsfunktion.

Ein Segment des Polygonzuges wird über das Tupel  $G = (\overline{\mathbf{p}_1\mathbf{p}_2}, \mathbf{t}_1, \mathbf{t}_2)$  definiert, hierbei sind  $\mathbf{p}_1$  und  $\mathbf{p}_2$  die Randpunkte des Segments und  $\mathbf{t}_1$  und  $\mathbf{t}_2$  die Tangentenvektoren an den Randpunkten. Die Tangentenvektoren geben die Richtung der glatten Ausgangskurve vor. Je nach Anwendungsfall sind die Tangentenvektoren exakt bekannt - zum Beispiel wenn der Polygonzug durch Abtasten einer bekannten, parametrisch dargestellten Kurve  $(x(t), y(t))^\top$  entstanden ist. Der Tangentenvektor ist dann  $(\dot{x}(t), \dot{y}(t))^\top$ . Ist die Ausgangskurve nicht bekannt, kann ein Tangentenvektor durch finite Differenzen (vergleiche Abschnitt 2.3) bestimmt werden, zum Beispiel

$$\mathbf{t}_i = \mathbf{p}_{i+1} - \mathbf{p}_{i-1}. \quad (3.4)$$

Für alle Punkte, die sich nicht an den Rändern des Segmentes befinden, wird nun ein Pseudo-Tangentenvektor erzeugt, indem die Tangentenvektoren der Randpunkte über der Länge  $l$  des Liniensegments linear interpoliert werden. Es sei

$$\mathbf{t}_\lambda = \lambda \mathbf{t}_2 + (1 - \lambda) \mathbf{t}_1 \quad (3.5)$$

der Pseudo-Tangentenvektor im Punkte

$$\mathbf{p}_\lambda = \lambda \mathbf{p}_2 + (1 - \lambda) \mathbf{p}_1, \quad (3.6)$$

für  $\lambda \in [0, 1]$ . Der Pseudoabstand eines Punktes  $\mathbf{x} = (x, y)$  zum Liniensegment ist nun definiert als die Länge des Pseudo-Normalenvektors

$$\mathbf{n}_\lambda = \mathbf{x} - \mathbf{p}_\lambda, \quad (3.7)$$

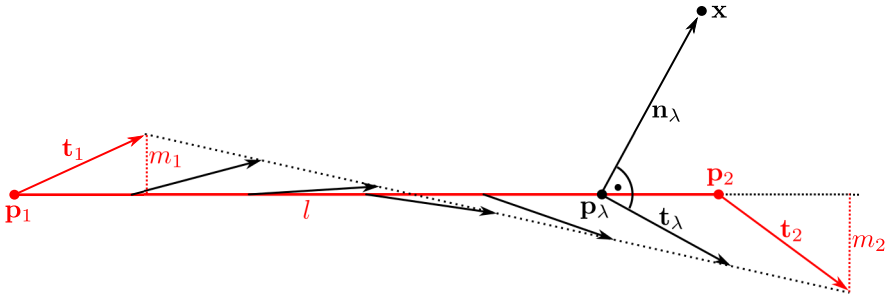


Abbildung 3.7: Normaleninterpolation

der  $\mathbf{x}$  mit dem Fußpunkt  $\mathbf{p}_\lambda$  seiner Projektion auf das Liniensegment verbindet. Die Projektion ist definiert über die Bedingung

$$\mathbf{n}_\lambda^\top \mathbf{t}_\lambda = 0, \quad (3.8)$$

Normalen- und Tangentenvektor sollen also senkrecht zueinander sein. Die geometrischen Beziehungen sind in Abbildung 3.7 illustriert. Es wird ohne Beschränkung der Allgemeinheit angenommen, dass sich  $\mathbf{p}_1$  im Ursprung befindet und  $\mathbf{p}_2$  auf der  $x$ -Achse bei  $x = l$ , also

$$\mathbf{p}_1 = (0, 0)^\top, \mathbf{p}_2 = (l, 0)^\top. \quad (3.9)$$

In diesem Fall kann man die Richtungen der Tangentenvektoren auch über die Steigungen  $m_1$  und  $m_2$  ausdrücken,

$$\mathbf{t}_1 = (1, m_1)^\top, \mathbf{t}_2 = (1, m_2)^\top. \quad (3.10)$$

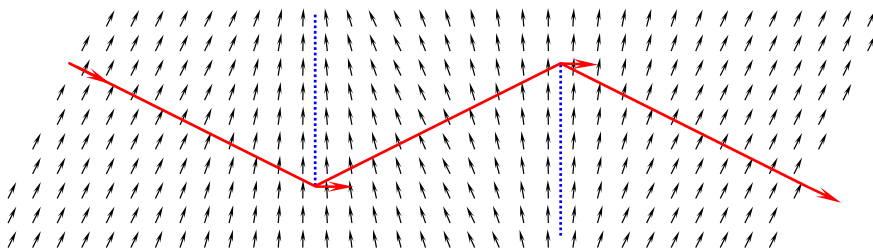
Man kann nun 3.8 leicht nach  $\lambda$  auflösen:

$$\lambda = \frac{m_1 y + x}{(m_1 - m_2)y + l}. \quad (3.11)$$

Der Betrag  $|\mathbf{n}_\lambda|$  ist nun der Pseudoabstand  $d(\mathbf{x}, G)$ , der Einheitsvektor  $\frac{\mathbf{n}_\lambda}{|\mathbf{n}_\lambda|}$  sein Pseudogradient. Höhere Ableitungen werden wieder exakt durch Ableiten des Pseudogradienten bestimmt.

Der Ansatz, den Normalenvektoren entlang des Liniensegmentes zu interpolieren, ist der Computergrafik entlehnt. Die Normaleninterpolation bezeichnet man in diesem Kontext nach BUI TUONG PHONG auch als PHONG-Shading Phong [Pho75], und sie bewirkt, dass auch Geometrien, die





**Abbildung 3.8:** Vorzeichenbehaftete Distanzfunktion mit Pseudogradienten.

durch Polygone angenähert werden, ein glattes Aussehen erhalten. Besonders unter einem spekularen Beleuchtungsmodell würde die Unstetigkeit der Oberflächennormale sonst sofort sichtbar.

Ein ähnliches Verfahren wurde auch in [Wer+10] angewendet, wobei hier allerdings der Tangenten*winkel* interpoliert wurde. Dies erfordert zur Abstandsbestimmung den Einsatz numerischer Lösungsverfahren, was erstens eine erhöhte Laufzeit mit sich bringt und zweitens die Bestimmung exakter Ableitungen erschwert.

Abbildung 3.8 zeigt beispielhaft das Pseudo-Gradientenfeld (schwarze Pfeile) für einen Polygonzug (rot). Die Voronoi-Linien sind blau gestrichelt eingezeichnet. Hier würde die wahre Abstandsfunktion einen unstetigen Gradienten aufweisen.

# 4 Optimale Trajektorienplanung: Grundlagen

## 4.1 Variationale Formulierung

Trajektorienplanung lässt sich als typisches Problem der Variationsrechnung auffassen. Die Variationsrechnung [Els61; MV98] befasst sich mit *Funktionalen* - das sind reelle Funktionen von Funktionen - und zwar insbesondere mit ihren Extremalen beziehungsweise ihren stationären Funktionen. Meistens handelt es sich beim Funktional  $E$  um das Integral einer Funktion  $L$ , die die zu maximierende oder minimierende, unbekannte Funktion  $x(t)$  mit ihren Ableitungen in Beziehung setzt:

$$E[x(t)] = \int_{t_0}^{t_1} L(t, x, \dot{x}) dt. \quad (4.1)$$

Wie im nächsten Abschnitt gezeigt wird, genügt die extremale Funktion  $x(t)$  dieses Funktionals der EULER-LAGRANGESchen Differentialgleichung. In dieser Arbeit ist die zu variierende Funktion meistens vektorwertig und stellt eine zweidimensionale Trajektorie  $\mathbf{x}(t) = (x(t), y(t))^T$  dar. Zu minimieren ist dann

$$E[x(t), y(t)] = \int_{t_0}^{t_1} L(t, x, y, \dot{x}, \dot{y}) dt. \quad (4.2)$$

Man hält dann abwechselnd die Funktionen  $x$  und  $y$  fest und variiert die andere, wodurch 4.2 jeweils die Form von 4.1 erhält, 4.2 lässt sich also analog zu 4.1 behandeln, wobei dann aber ein System aus zwei Differentialgleichungen zu lösen ist.

Der Integrand des Kostenfunktionals kann auch höhere Ableitungen enthalten, für eine eindimensionale Trajektorie schreibt man dann

$$E[x(t)] = \int_{t_0}^{t_1} L(t, x, \dot{x}, \dots, x^{(n)}) dt, \quad (4.3)$$

oder im zweidimensionalen

$$E[x(t), y(t)] = \int_{t_0}^{t_1} L(t, x, y, \dot{x}, \dot{y}, \dots, x^{(n)}, y^{(n)}) dt. \quad (4.4)$$

Es ist jeweils  $n \in \mathbb{N}$ , und  $x^{(n)}$  ist die  $n$ -te Zeitableitung von  $x$ . Gelegentlich schreibt man (4.4) auch in vektorieller Form

$$E[\mathbf{x}(t)] = \int_{t_0}^{t_1} L(t, \mathbf{x}, \dot{\mathbf{x}}, \dots, \mathbf{x}^{(n)}) dt. \quad (4.5)$$

mit  $\mathbf{x}(t) = (x(t), y(t))^T$ .

Um zu einer eindeutigen Optimaltrajektorie zu gelangen, muss diese im Allgemeinen noch *Randbedingungen* erfüllen, das heißt  $\mathbf{x}(t)$ ,  $\dot{\mathbf{x}}(t)$ ,  $\dots$  müssen am linken und rechten Rand des Integrationsbereiches,  $t_0$  und  $t_1$ , definierte Werte annehmen. Für das Fahren in fließendem Verkehr wird diese Aufgabe so formuliert, dass die Trajektorie am rechten Rand „frei“ ist,  $\mathbf{x}(t_1)$ ,  $\dot{\mathbf{x}}(t_1)$ ,  $\dots$  sind also beliebig, außerdem lässt man  $t_1$  gegen  $\infty$  gehen.

In einigen Fällen kann das Problem der Trajektorienplanung auch so gestellt sein, dass die Zeit  $t_1$  „frei“ ist und „mit optimiert werden“ muss. Die zu veranschlagende Reisezeit kann ja in der Regel nicht vorher festgelegt werden. Diese Eigenschaft des Problems führt auf eine zusätzliche notwendige Bedingung für eine Optimaltrajektorie, eine sogenannte *Transversalitätsbedingung*. Auf die Behandlung verschiedener Varianten von Rand- und Transversalitätsbedingungen wird an den entsprechenden Stellen noch eingegangen werden.

Bei der Minimierung von  $E$  müssen in der Praxis außer den Randbedingungen auch noch allgemeinere *Nebenbedingungen* berücksichtigt werden, mit denen Kollisionsfreiheit sowie Stellbegrenzungen und Grenzen der Fahrzeugdynamik ausgedrückt werden. Nebenbedingungen werden bei kontinuierlicher Betrachtung des Zustandsraumes (Abschnitt 4.2 und Kapitel 5) meistens als Ungleichungen formuliert. Diese können auch Zeitableitungen der Trajektorie in Relation setzen (zum Beispiel wenn die Geschwindigkeit des Fahrzeuges limitiert werden soll). Bei den kombinatorischen Verfahren (Kapitel 6) wird der Zustandsraum diskretisiert, er wird also durch eine endliche Untermenge angenähert. Zustände, die Nebenbedingungen verletzen, werden dann einfach im Vorhinein aus dieser Menge entfernt.

## 4.2 Exakte Lösungen

In diesem Abschnitt werden Ansätze vorgestellt, um das variationale Problem der Trajektorienplanung analytisch zu lösen. Dies ist nur in einfachen Fällen möglich, insbesondere wenn keinerlei expliziten Nebenbedingungen für Hindernisse, Stellbeschränkungen und die Fahrzeugdynamik gestellt werden. Für praktische Problemstellungen, in denen Nebenbedingungen behandelt werden sollen, wird auf numerische Verfahren ausgewichen, die die exakte Lösung nur annähern (Kapitel 5 und 6). Trotzdem ist es nützlich, sich zunächst mit exakten Lösung zu befassen, da sich einige Aussagen, beispielsweise über Stabilität und Einschwingverhalten, ableiten lassen, die sich auch auf das durch Nebenbedingungen eingeschränkte Problem übertragen lassen. Der Übersichtlichkeit halber wird in diesem Abschnitt gelegentlich nur eine eindimensionale Trajektorie  $x(t)$  behandelt. Dies ist näherungsweise ausreichend, wenn man ein Fahrzeug betrachtet, das sich entlang einer geraden Strecke mit gleichbleibender Längsgeschwindigkeit bewegt.

### 4.2.1 Lösen der EULER-POISSON-Gleichung

Das bekannteste Hilfsmittel der Variationsrechnung ist die EULER-LAGRANGESche *Differenzialgleichung* (EULER-LAGRANGE-Gleichung), die eine notwendige Bedingung für ein Minimum von Funktionalen der Form (4.1) darstellt (im Folgenden wird die Kurzschreibweise  $L_x$  für die Variation  $\frac{\delta}{\delta x}L(t, x, \dot{x})$  verwendet):

$$L_x - \frac{d}{dt}L_{\dot{x}} = 0. \quad (4.6)$$

Enthält der Integrand  $L$  höhere Ableitungen bis zur Ordnung  $n$ , wie in (4.3), so muss diese zur EULER-POISSONschen *Differenzialgleichung* verallgemeinert werden:

$$L_x - \frac{d}{dt}L_{\dot{x}} + \frac{d^2}{dt^2}L_{\ddot{x}} + \dots + (-1)^n \frac{d^n}{dt^n}L_{x^{(n)}} = 0. \quad (4.7)$$

Ist der Integrand  $L$  von zwei unabhängigen Veränderlichen  $x$  und  $y$  abhängig, wie in (4.4), so erhält man ein System von zwei Differentialgleichungen. Im Folgenden wird der Übersichtlichkeit halber angenommen  $n = 2$ ,

dann erhält man das Differentialgleichungssystem

$$\begin{aligned} L_x - \frac{d}{dt}L_{\dot{x}} + \frac{d^2}{dt^2}L_{\ddot{x}} &= 0 \\ L_y - \frac{d}{dt}L_{\dot{y}} + \frac{d^2}{dt^2}L_{\ddot{y}} &= 0. \end{aligned} \quad (4.8)$$

Für eine Herleitung der Gleichungen (4.6) bis (4.7) siehe zum Beispiel Elsgolc [Els61]. Eine Herleitung des Spezialfalles (4.8) ist in Abschnitt 5.1 enthalten.

**Beispiel 4.1.** Es sollen Trajektorien in der  $x, y$ -Ebene mit minimaler quadratischer Gesamtbeschleunigung gefunden werden. Man wählt

$$L^{\text{acc}}(t, x(t), \dot{x}(t), \ddot{x}(t), y(t), \dot{y}(t), \ddot{y}(t)) = \ddot{x}(t)^2 + \ddot{y}(t)^2, \quad (4.9)$$

die Gesamtkosten im Zeitraum  $[t_0, t_1]$  sind also

$$E^{\text{acc}}[x, y] = \int_{t_0}^{t_1} L^{\text{acc}}(t, x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}) dt. \quad (4.10)$$

Nach (4.8) erhält man folgendes Gleichungssystem als notwendige Bedingung für eine Extremalkurve:

$$\begin{aligned} x^{(4)}(t) &= 0 \\ y^{(4)}(t) &= 0. \end{aligned}$$

Dieses Gleichungssystem wird nur für polynomielle Kurven dritten Grades erfüllt:

$$\begin{aligned} x(t) &= a_1 + a_2t + a_3t^2 + a_4t^3 \\ y(t) &= b_1 + b_2t + b_3t^2 + b_4t^3. \end{aligned} \quad (4.11)$$

Eine konkrete Lösung erhält man nun, indem man Nebenbedingungen, beispielsweise Randwerte  $x(t_0) = x_0$ ,  $x(t_1) = x_1$ ,  $y(t_0) = y_0$  und  $y(t_1) = y_1$  für Start- und Endposition, sowie für Start- und Endgeschwindigkeiten  $\dot{x}(t_0) = \dot{x}_0$ ,  $\dot{x}(t_1) = \dot{x}_1$ ,  $\dot{y}(t_0) = \dot{y}_0$  und  $\dot{y}(t_1) = \dot{y}_1$  vorgibt, und dann die 8 freien Konstanten  $a_1, \dots, a_4$  und  $b_1, \dots, b_4$  bestimmt.

Man kann leicht zeigen, dass sich das Ergebnis aus Beispiel 4.1 folgendermaßen verallgemeinern lässt: Ein Kostenfunktional wie in (4.4) mit einem Integranden der Form  $x^{(n)}(t)^2 + y^{(n)}(t)^2$  wird durch polynomielle Kurven  $(2n - 1)$ -ten Grades minimiert. Hieraus folgt beispielsweise, dass quintische Polynome ruckoptimale Trajektorien darstellen [Tak+89].

**Beispiel 4.2.** Man wählt

$$L(t, x, \dot{x}, \ddot{x}) = w_{\text{beschl}} \ddot{x}^2 + w_{\text{vquer}} \dot{x}^2 + w_{\text{ablage}} x^2. \quad (4.12)$$

mit Gewichtungsfaktoren  $w_{\text{beschl}} > 0$ ,  $w_{\text{vquer}} > 0$  und  $w_{\text{ablage}} = 1$ . Die EULER-POISSON-Gleichung lautet

$$w_{\text{beschl}} x^{(4)} - w_{\text{vquer}} \ddot{x} + w_{\text{ablage}} x = 0. \quad (4.13)$$

Für die allgemeine Lösung dieser linearen, homogenen, gewöhnlichen Differentialgleichung vierter Ordnung betrachtet man die Nullstellen  $\lambda_1, \dots, \lambda_4$  der polynomiellen, charakteristischen Gleichung

$$w_{\text{beschl}} \lambda^4 + w_{\text{vquer}} \lambda^2 + w_{\text{ablage}} = 0 \quad (4.14)$$

von (4.13). Man erhält

$$\begin{aligned} \lambda_{1,2} &= a\sqrt{b \pm c} \\ \lambda_{3,4} &= -a\sqrt{b \pm c} \end{aligned}$$

mit

$$\begin{aligned} a &= \frac{\sqrt{2}}{2\sqrt{w_{\text{beschl}}}} \\ b &= w_{\text{vquer}} \\ c &= \sqrt{w_{\text{vquer}}^2 - 4w_{\text{vquer}}w_{\text{beschl}}}. \end{aligned}$$

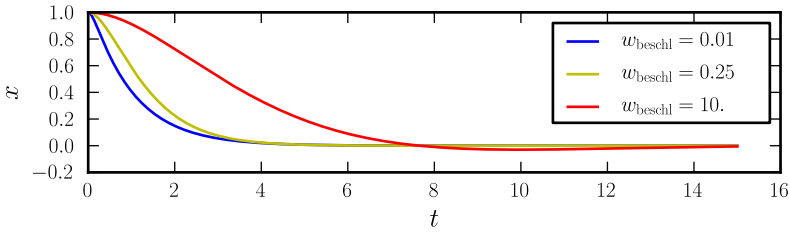
Die Nullstellen  $\lambda_1, \dots, \lambda_4$  sind im Allgemeinen paarweise verschieden, deshalb gilt für die Lösungstrajektorie der Ansatz [MV98]

$$x(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} + c_3 e^{\lambda_3 t} + c_4 e^{\lambda_4 t}, \quad (4.15)$$

mit wählbaren Parametern  $c_1, \dots, c_4$ . Es sollen am linken Rand  $t_0 = 0$  Startbedingungen

$$\begin{aligned} x(0) &= x_0 \\ \dot{x}(0) &= v_0 \end{aligned} \quad (4.16)$$

vorgeben werden, also Initialposition und -geschwindigkeit,  $x_0$  und  $v_0$ . Am rechten Rand soll die Trajektorie diesmal frei bleiben, deshalb müssen hier



**Abbildung 4.1:** Optimaltrajektorien für verschiedene Werte von  $w_{\text{beschl}}$ .

die sogenannten *natürlichen Randbedingungen* gelten [DS13],

$$\begin{aligned}
 \left. (L_{\dot{x}}(t, x, \dot{x}, \ddot{x}) - \frac{d}{dt}(L_{\ddot{x}}(t, x, \dot{x}, \ddot{x}))) \right|_{t=t_1} &= 0, \\
 \left. L_{\ddot{x}}(t, x, \dot{x}, \ddot{x}) \right|_{t=t_1} &= 0 \\
 &\iff \\
 w_{\text{vquer}}\dot{x}(t_1) - \ddot{x}(t_1) &= 0, \\
 \ddot{x}(t_1) &= 0.
 \end{aligned} \tag{4.17}$$

Aus (4.17) folgt, dass, wenn man  $t_1$  gegen  $\infty$  gehen lässt, in (4.15) die Terme mit positiven Exponenten  $\lambda_1$  und  $\lambda_2$  verschwinden müssen, es muss also gelten  $c_1 = c_2 = 0$ . Dies war zu erwarten, da dies genau die Glieder von (4.15) verschwinden lässt, die ansonsten die Beträge von  $x(t)$  und  $\dot{x}(t)$ , und damit auch den Wert des Kostenfunktional, unbeschränkt wachsen lassen würden. Die verbleibenden Parameter  $c_3$  und  $c_4$  werden über die Startbedingungen (4.16) bestimmt.

Man sieht, das für

$$w_{\text{beschl}} > \frac{1}{4}w_{\text{vquer}} \tag{4.18}$$

die Exponenten der Lösungstrajektorie komplex werden, sie schwingt dann gedämpft um die Zielposition. Abbildung 4.1 zeigt drei Trajektorien für  $w_{\text{vquer}} = w_{\text{ablage}} = 1$ ,  $x_0 = 1$ ,  $v_0 = 0$  und verschiedene Werte von  $w_{\text{beschl}}$ . Für  $w_{\text{beschl}} = 0.25$  (gelbe Kurve) tritt nach (4.18) der aperiodische Grenzfall auf, das System schwingt also gerade nicht mehr. Für  $w_{\text{beschl}} > 0.25$  (rote Kurve) schwingt das System über.

Im Randfall  $w_{\text{vquer}} = w_{\text{ablage}} = 0$  geht das Problem in das aus Beispiel 4.1 über. Man erhält dann eine vierfache Nullstelle des charakteristischen Polynoms (4.14),  $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0$ , und wählt deshalb statt (4.15)

den Ansatz

$$x(t) = c_1 e^{\lambda_1 t} + c_2 t e^{\lambda_2 t} + c_3 t^2 e^{\lambda_3 t} + c_4 t^3 e^{\lambda_4 t}, \quad (4.19)$$

der erwartungsgemäß auf ein kubisches Polynom führt.

## 4.2.2 Lösen der RICCATI-Gleichung

Kostenfunktionale mit quadratischem Integranden wie (4.9) und (4.12) werden auch bei der Auslegung von LQ-Reglern verwendet. Mit dieser Methodik kann eine lineare Rückführung für einen Zustandsregler gefunden werden, der das Kostenfunktional innerhalb eines beschränkten oder unbeschränkten Zeithorizonts minimiert. Durch Integration der Differentialgleichung des geschlossenen Regelkreises erhält man dann die Zustandstrajektorie.

Ein System wird über seine Systemmatrix  $\mathbf{A}$  und die Eingangsmatrix  $\mathbf{B}$  als System gewöhnlicher linearer Differentialgleichungen definiert,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}. \quad (4.20)$$

$\mathbf{x}(t)$  ist die Zustandstrajektorie des Systems,  $\mathbf{u}(t)$  der Systemeingang. Das Gleichungssystem des durch lineare Rückführung geregelten Systems ist

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}\mathbf{x}, \quad (4.21)$$

mit der Rückführungsmatrix  $\mathbf{K}$ . Das quadratische Kostenfunktional  $E$  wird über die Wichtungsmatrizen  $\mathbf{Q}$  und  $\mathbf{R}$  definiert,

$$E[\mathbf{u}, \mathbf{x}] = \int_0^T \mathbf{x}^\top(t) \mathbf{Q} \mathbf{x}(t) + \mathbf{u}^\top(t) \mathbf{R} \mathbf{u}(t) dt. \quad (4.22)$$

Die Trajektorie  $\mathbf{x}(t)$  des geschlossenen Regelkreises minimiert das Kostenfunktional (4.22) für  $T \rightarrow \infty$  genau dann, wenn die Rückführungsmatrix

$$\mathbf{K} = \mathbf{R}^{-1} \mathbf{B}^\top \mathbf{P} \quad (4.23)$$

lautet, wobei  $\mathbf{P}$  die algebraische RICCATI-Gleichung

$$\mathbf{P}\mathbf{A} + \mathbf{A}^\top \mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1} \mathbf{B}^\top \mathbf{P} + \mathbf{Q} = 0 \quad (4.24)$$

löst. Bei (4.24) handelt es sich um ein System von  $n^2$  quadratischen Gleichungen, wobei  $n$  die Dimensionalität des Zustandsraumes ist.



**Beispiel 4.3.** (4.12) wird um einen gewichteten Ruckterm

$$L(t, x, \dot{x}, \ddot{x}) = w_{\text{ablage}}x^2 + w_{\text{vquer}}\dot{x}^2 + w_{\text{beschl}}\ddot{x}^2 + w_{\text{ruck}}\ddot{x}^2 \quad (4.25)$$

erweitert. Das Kostenfunktional

$$E[x(t)] = \int_0^\infty L(t, x, \dot{x}, \ddot{x})dt \quad (4.26)$$

erhält die Form (4.22), wenn als System eine Integratorchette gewählt wird, mit dem Systemzustand  $\mathbf{x} = (x, \dot{x}, \ddot{x})^\top$ , dem Eingang  $\mathbf{u} = (\ddot{x})$  und

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{B} = (0, 0, 1)^\top$$

sowie den Wichtungsmatrizen

$$\mathbf{Q} = \begin{pmatrix} w_{\text{ablage}} & 0 & 0 \\ 0 & w_{\text{vquer}} & 0 \\ 0 & 0 & w_{\text{beschl}} \end{pmatrix},$$

$$\mathbf{R} = (w_{\text{ruck}}).$$

Das Gleichungssystem (4.24) muss im Allgemeinen numerisch gelöst werden. Im Beispiel erhält man eine Rückführungsmatrix  $\mathbf{K} = (k_1, k_2, k_3)$ . Die Trajektorie erhält man nun, indem man das Differentialgleichungssystem (4.21) löst, das in diesem Beispiel die Form

$$\begin{pmatrix} \dot{x} \\ \dot{\dot{x}} \\ \dot{\ddot{x}} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -k_1 & -k_2 & -k_3 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} \quad (4.27)$$

hat. Dieses lineare System gewöhnlicher Differentialgleichungen lässt sich über den Ansatz

$$x(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} + c_3 e^{\lambda_3 t} \quad (4.28)$$

lösen, wobei sich  $\lambda_1, \lambda_2, \lambda_3$  als Nullstellen der charakteristischen Gleichung

$$x^3 + k_3 x^2 + k_2 x + k_1 = 0 \quad (4.29)$$

ergeben, und  $c_1, c_2, c_3$  zum Beispiel über Startbedingungen  $x(0) = d_0$ ,  $\dot{x}(0) = v_0$  und  $\ddot{x}(0) = a_0$  eliminiert werden. Man erhält also wieder eine gewichtete Summe (komplexer) Exponentialfunktionen. Man bezeichnet  $\lambda_1, \lambda_2, \lambda_3$  auch als die Eigenwerte des geschlossenen Regelkreises.

Wie man sieht, müssen beim RICCATI-Ansatz zwar einige Parameter der Optimaltrajektorie numerisch bestimmt werden, dennoch erhält man die Trajektorie dann in Form einer Differentialgleichung, anhand derer Eigenschaften der Trajektorie analytisch untersucht werden können, beispielsweise ihr Schwingverhalten.

## 5 Lokale, kontinuierliche Lösungsverfahren

Gleichung (4.7) oder Gleichungssysteme der Form (4.8) lassen sich nur in Ausnahmefällen exakt lösen, man ist deshalb auf näherungsweise Lösungsverfahren angewiesen. Die grundlegende Idee besteht dabei darin, die Variationsaufgabe zunächst als Extremwertaufgabe gewöhnlicher Funktionen zu betrachten, und anschließend die Anzahl der Variablen gegen unendlich gehen zu lassen. Eine spezielle Variante dieses Vorgehens ist das EULERSche Differenzenverfahren, dass auch als die Methode der finiten Differenzen bezeichnet wird. Die hier vorgestellte Herangehensweise kann man zu den sogenannten *Kollokationsverfahren* zählen, da der Lösungsraum durch Stützpunkte parametrisiert wird, die von der Optimaltrajektorie interpoliert werden.

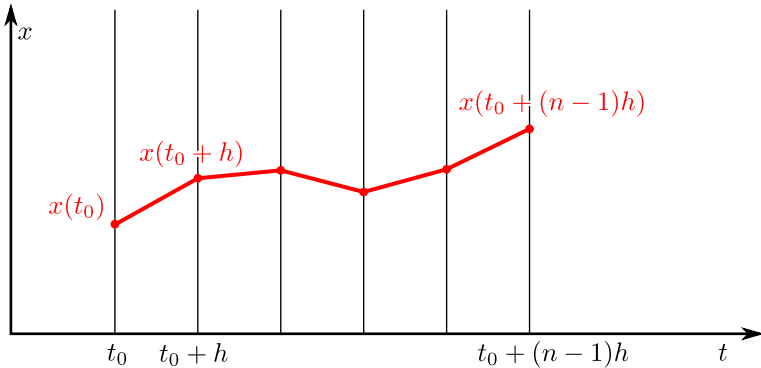
### 5.1 Diskretisierung des Energiefunktional

Beim EULERSchen Differenzenverfahren wird die Domäne des Funktionals  $E$  zunächst auf Polygone beschränkt, die  $n$  Ecken an vorgegebenen, äquidistanten Stellen  $\{t_i = t_0 + ih \mid 0 \leq i < n\}$  aufweisen, vergleiche Abbildung 5.1. Der Übersichtlichkeit halber werden im Folgenden wieder nur Integranden, die Ableitungen bis zu zweiter Ordnung enthalten, betrachtet. Außerdem wird nur eine eindimensionale Trajektorie  $x(t)$  betrachtet. Man ersetzt das Integral

$$E[x] = \int_{t_0}^{t_{n-1}} L(t, x, \dot{x}, \ddot{x}) dt \quad (5.1)$$

durch die endliche Summe

$$E_d(x_0, x_1, \dots, x_{n-1}) = \sum_{i=1}^{n-2} L(t_i, x_i, \frac{\delta x_i}{2h}, \frac{\delta^2 x_i}{h^2}) h, \quad (5.2)$$



**Abbildung 5.1:** Die Menge der Vergleichskurven wird auf Polygonzüge beschränkt.

wobei  $\delta x_i$  und  $\delta^2 x_i$  die symmetrischen finiten Differenzen erster und zweiter Ordnung wie in Abschnitt 2.3 sind. Ableitungen der Vergleichsfunktionen werden also durch Differenzen angenähert.  $E_d$  ist eine Funktion von  $n$  Veränderlichen und kann mit bekannten numerischen oder analytischen Verfahren minimiert werden.

Es werden nun einige Umformungen durchgeführt und so nachträglich bewiesen, dass die Minimierung von (5.1) auf die EULER-POISSONSche Gleichung (4.7) führt. Notwendige Bedingung für eine Extremstelle ist, dass die partiellen Ableitungen  $\frac{dE_d}{dx_i}$  verschwinden. Für jedes  $i$  hängen nur drei Glieder der Summe (6.3) von  $x_i$  ab, nämlich das  $(i-1)$ -te bis  $(i+1)$ -te. Es gilt also für die  $i$ -te partielle Ableitung (finite Differenzen hier ausgeschrieben):

$$\begin{aligned} \frac{dE_d}{dx_i} = & \frac{d}{dx_i} \left[ L(t_{i-1}, x_{i-1}, \frac{x_i - x_{i-2}}{2h}, \frac{x_i - 2x_{i-1} + x_{i-2}}{h^2})h \right. \\ & + L(t_i, x_i, \frac{x_{i+1} - x_{i-1}}{2h}, \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2})h \\ & \left. + L(t_{i+1}, x_{i+1}, \frac{x_{i+2} - x_i}{2h}, \frac{x_{i+2} - 2x_{i+1} + x_i}{h^2})h \right] = 0 \end{aligned}$$

Nach Anwendung der Kettenregel nimmt die Gleichung die Form

$$\begin{aligned} \frac{dE_d}{dx_i} = & L_{\dot{x}}(t_{i-1}, x_{i-1}, \frac{x_i - x_{i-2}}{2h}, \frac{x_i - 2x_{i-1} + x_{i-2}}{h^2})h \frac{1}{2h} \\ & + L_{\ddot{x}}(t_{i-1}, x_{i-1}, \frac{x_i - x_{i-2}}{2h}, \frac{x_i - 2x_{i-1} + x_{i-2}}{h^2})h \frac{1}{h^2} \end{aligned}$$

$$\begin{aligned}
& +L_x(t_i, x_i, \frac{x_{i+1} - x_{i-1}}{2h}, \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2})h \\
& +L_{\ddot{x}}(t_i, x_i, \frac{x_{i+1} - x_{i-1}}{2h}, \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2})h \frac{-2}{h^2} \\
& +L_{\dot{x}}(t_{i+1}, x_{i+1}, \frac{x_{i+2} - x_i}{2h}, \frac{x_{i+2} - 2x_{i+1} + x_i}{h^2})h \frac{-1}{2h} \\
& +L_{\ddot{x}}(t_{i+1}, x_{i+1}, \frac{x_{i+2} - x_i}{2h}, \frac{x_{i+2} - 2x_{i+1} + x_i}{h^2})h \frac{1}{h^2} = 0
\end{aligned}$$

an, die sich zu

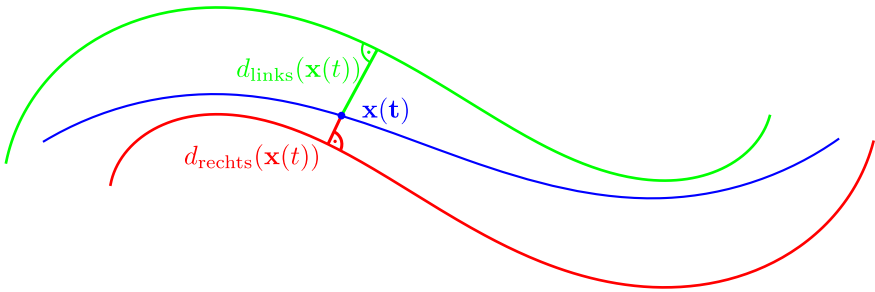
$$\begin{aligned}
& \frac{1}{2} \left( L_{\dot{x}}(t_{i-1}, x_{i-1}, \frac{\Delta x_{i-1}}{2h}, \frac{\Delta^2 x_{i-1}}{h^2}) \right. \\
& \left. - L_{\dot{x}}(t_{i+1}, x_{i+1}, \frac{\Delta x_{i+1}}{2h}, \frac{\Delta^2 x_{i+1}}{h^2}) \right) \\
& + \frac{1}{h} \left( L_{\ddot{x}}(t_{i-1}, x_{i-1}, \frac{\Delta x_{i-1}}{2h}, \frac{\Delta^2 x_{i-1}}{h^2}) \right. \\
& \left. - 2L_{\ddot{x}}(t_i, x_i, \frac{\Delta x_i}{2h}, \frac{\Delta^2 x_i}{h^2}) \right. \\
& \left. + L_{\ddot{x}}(t_{i+1}, x_{i+1}, \frac{\Delta x_{i+1}}{2h}, \frac{\Delta^2 x_{i+1}}{h^2}) \right) \\
& + hL_x(t_i, x_i, \frac{\Delta x_i}{2h}, \frac{\Delta^2 x_i}{h^2}) = 0
\end{aligned}$$

oder

$$\begin{aligned}
& L_x(t_i, x_i, \frac{\Delta x_i}{2h}, \frac{\Delta^2 x_i}{h^2}) \\
& - \frac{1}{2h} \Delta L_{\dot{x}}(t_i, x_i, \frac{\Delta x_i}{2h}, \frac{\Delta^2 x_i}{h^2}) \\
& + \frac{1}{h^2} \Delta^2 L_{\ddot{x}}(t_i, x_i, \frac{\Delta x_i}{2h}, \frac{\Delta^2 x_i}{h^2}) = 0.
\end{aligned} \tag{5.3}$$

vereinfachen lässt. Lässt man nun die Anzahl  $n$  der Variablen  $x_i$  gegen Unendlich gehen, wobei  $h$  gegen 0 geht, sieht man, dass der Grenzwert die EULER-POISSONSche Gleichung (für Integranden, die Ableitungen bis zur zweiten Ordnung enthalten) darstellt:

$$L_x - \frac{d}{dt}L_{\dot{x}} + \frac{d^2}{dt^2}L_{\ddot{x}} = 0. \tag{5.4}$$



**Abbildung 5.2:** Ablageterm des Kostenfunktionalen.

Im nächsten Abschnitt wird eine Methode vorgestellt, die es erlaubt, Extremwertaufgaben für Funktionen der Form (6.3) auch unter nichtlinearen Zwangsbedingungen zu lösen.

## 5.2 Optimalitätskriterien und Kostenfunktional

In diesem Abschnitt soll formalisiert werden, was im Rahmen der Trajektorienplanung als „optimal“ gilt. Hierzu soll das Kostenfunktional  $E$  aus (4.5) exemplarisch und konkret ausgeprägt werden. Dies geschieht durch Definition des Integranden  $L$ . Die Aufstellung der Kostenfunktion soll als exemplarisch betrachtet werden und ist auf den Anwendungsfall des automatischen Fahrens in fließendem Verkehr zugeschnitten. Im Allgemeinen muss eine problemangepasste Kostenfunktion erstellt werden.

$L$  ist die Summe aus mehreren Kostentermen,

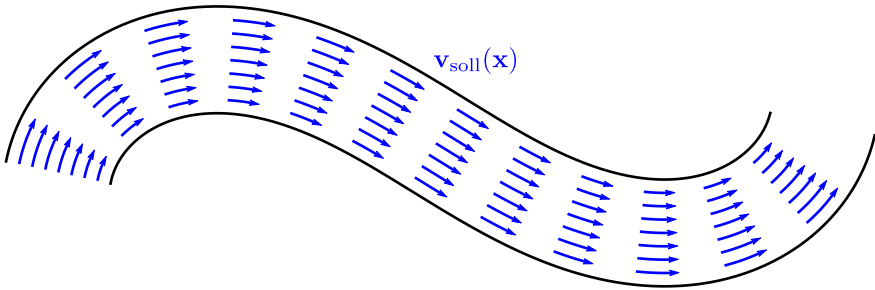
$$L(\mathbf{x}(t)) = e_{\text{ablage}} + e_{\text{geschw}} + e_{\text{beschl}} + e_{\text{ruck}} + e_{\text{gierrate}} + e_{\text{unendl}}. \quad (5.5)$$

Alle Summanden sind mit einer Gewichtung ( $w_{\text{ablage}}$ ,  $w_{\text{geschw}}$  usw.) versehen. Nun werden die einzelnen Summanden erläutert, über die integriert wird.

Der Term

$$e_{\text{ablage}}(\mathbf{x}(t)) = w_{\text{ablage}} \left| \frac{1}{2} (d_{\text{links}}(\mathbf{x}(t)) + d_{\text{rechts}}(\mathbf{x}(t))) \right|^2 \quad (5.6)$$

bewirkt, dass die Trajektorie möglichst in der Mitte zwischen den beiden Korridorändern verläuft. Die Funktionen  $d_{\text{links}}$  und  $d_{\text{rechts}}$  sind die vorzei-



**Abbildung 5.3:** Geschwindigkeitsterm des Kostenfunktionals. Die Korridorwände sind schwarz.

chenbehafteten Abstandsfunktionen (vergleiche Kapitel 3) zu den Korridorwänden, der Abstand ist also positiv für alle Punkte links des Randes, und negativ für alle Punkte rechts davon. Abbildung 5.2 veranschaulicht den Ablageterm.

Der Term

$$e_{\text{geschw}}(\mathbf{x}(t)) = w_{\text{geschw}} |\mathbf{v}_{\text{soll}}(\mathbf{x}(t)) - \dot{\mathbf{x}}(t)|^2 \quad (5.7)$$

enthält den quadratischen Fehler der vektoriellen Geschwindigkeit der Trajektorie im Vergleich zu einem Referenzgeschwindigkeitsvektor  $\mathbf{v}_{\text{soll}}$ . Der Vektor  $\mathbf{v}_{\text{soll}}$  wird in seinem Betrag  $v_{\text{soll}}$  von der Verhaltensentscheidung bestimmt, der einer der Karte entnommenen Höchstgeschwindigkeit entspricht. Die Richtung des Referenzgeschwindigkeitsvektors ist orthogonal zu den Gradienten der Korridor-Abstandsfunktionen, die Sollfahrtrichtung ist also parallel zu den Korridorwänden, Das Vektorfeld  $\mathbf{v}_{\text{soll}}(\mathbf{x})$  ist beispielhaft in Abbildung 5.3 illustriert.

Die beiden bis jetzt beschriebenen Terme,  $e_{\text{ablage}}$  und  $e_{\text{geschw}}$ , geben das Wunschverhalten der Trajektorie vor: Sie soll in der Mitte des Korridors verlaufen, Fortschritt entlang des Korridors erzielen und eine vorgegebene Wunschgeschwindigkeit aufweisen. Sie wirken damit den folgenden Termen entgegen, die fahrdynamisch motiviert sind und eine Glättung der Trajektorie bewirken. Der Term

$$e_{\text{beschl}}(\mathbf{x}(t)) = w_{\text{beschl}} |\ddot{\mathbf{x}}(t)|^2 \quad (5.8)$$

unterdrückt starke Beschleunigungen in Quer- und Längsrichtung, und damit die Kräfte, die auf die Passagiere einwirken. Der Ruckterm

$$e_{\text{ruck}}(\mathbf{x}(t)) = w_{\text{ruck}} |\ddot{\ddot{\mathbf{x}}}(t)|^2 \quad (5.9)$$

erzwingt zusätzliche Glattheit der Trajektorie, indem schnelle Änderungen der Beschleunigung unterdrückt werden. Erst dieser Ruckterm sichert die wichtige Beschleunigungs- und damit Krümmungstetigkeit der Trajektorie zu. In Abschnitt 2.2 wurde ja gezeigt, dass die Krümmung der Trajektorie in direktem Zusammenhang mit dem Lenkwinkel steht, der am Fahrzeug beim Folgen der Trajektorie eingestellt werden muss. Durch Krümmungstetigkeit der Trajektorie wird also sprungartiges Lenken vermieden.

Die Unterdrückung von Beschleunigung und Ruck kann noch nicht verhindern, das beim Fahren entlang der Trajektorie schnelle Gierwinkeländerungen erfolgen. Hierzu wird ein Term in den Integranden eingeführt, der hohe Drehraten dämpft,

$$e_{\text{gierrate}}(\mathbf{x}(t)) = w_{\text{gierrate}} \left| \dot{\theta}(t) \right|^2. \quad (5.10)$$

Das hier eingeführte Kostenfunktional weist große Ähnlichkeit zu denen aus den Beispielen 4.2 und 4.3 auf. Bis auf den Gierratenterm entspricht es dem aus Beispiel 4.3 unter folgender vereinfachenden Annahme: Die Strecke verlaufe geradeaus, entlang der  $x$ -Achse. Der Wunschgeschwindigkeitsvektor ist damit  $\mathbf{v}_{\text{soll}} = (v_{\text{soll}}, 0)^\top$ . Längs- und Querrichtung zur Strecke werden separiert betrachtet. Man kann leicht zeigen, dass die separierte Betrachtung beider Koordinatenachsen möglich ist, wenn man auf den Gierratenterm verzichtet. Erst dieser stellt eine Kopplung zwischen  $x$  und  $y$  her. Man kann daher bis zu einem gewissen Grade Ergebnisse der analytischen Betrachtung aus Abschnitt 4.2 nutzen, um das Verhalten des Trajektorienplaners mit diesem Kostenfunktional vorherzusagen, indem, basierend auf den gewählten Wichtungen  $w_{\text{ablage}}$ ,  $w_{\text{geschw}}$ ,  $w_{\text{beschl}}$  und  $w_{\text{ruck}}$ , die RICCATI-Gleichung wie in Beispiel 4.3 gelöst, und die Trajektorie des geschlossenen Regelkreises, zum Beispiel anhand seiner Eigenwerte, untersucht wird.

### 5.3 Nebenbedingungen

Neben dem beschriebenen Optimalitätskriterium müssen bei der Berechnung der Trajektorie noch Nebenbedingungen eingehalten werden. Dies sind einerseits solche, die Kollisionsfreiheit und das Verbleiben im Fahrkorridor zusichern. Diese werden im Folgenden als *äußere* Nebenbedingungen bezeichnet. Darüber hinaus gibt es Nebenbedingungen, die sich aus der Lenkinematik des Fahrzeuges und aus dynamischen Beschränkungen ergeben. Sie werden als *innere* Nebenbedingungen bezeichnet. Bei den



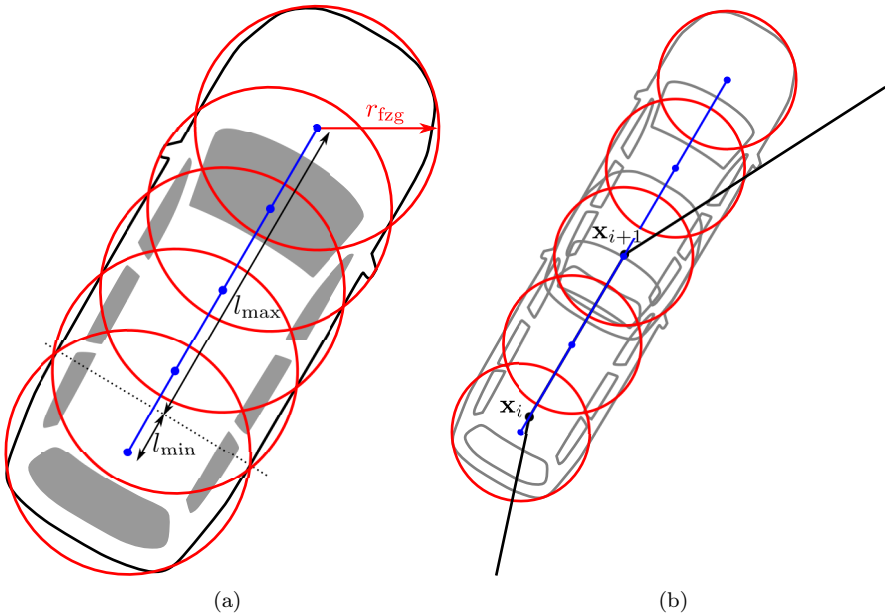
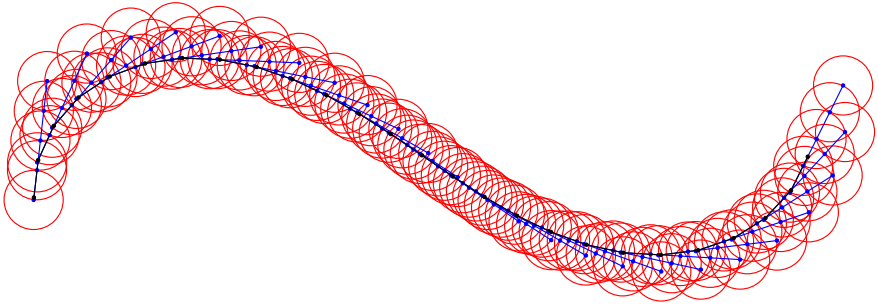


Abbildung 5.4: Kreisabdeckung Fahrzeug

lokalen kontinuierlichen Verfahren müssen Nebenbedingungen als stetig differenzierbare Ungleichungen ausgedrückt werden. Die inneren Nebenbedingungen werden in der Regel als Ungleichung zwischen Ableitungen der Trajektorie ausgedrückt.

### 5.3.1 Äußere Nebenbedingungen

Äußere Nebenbedingungen betreffen die Umgebung des Fahrzeuges, also Hindernisse, die sensorisch erfasst wurden oder Einschränkungen die sich aus einer Karte ergeben, zum Beispiel die Ränder eines Fahrstreifens. Es geht dabei immer um die Kollisionsfreiheit mit der Umgebung, deshalb ist es wichtig, sich mit der Ausdehnung und Form des Fahrzeuges zu befassen.



**Abbildung 5.5:** Annäherung der Manövrierfläche durch Kreise

### Fahrzeugform und Manövrierfläche

Die Fahrzeugform wird durch eine Menge von Kreisen mit Radius  $r_{\text{fzg}}$  angenähert, die den Aufriss des Fahrzeuges überdecken. Alle Kreise liegen auf der Fahrzeuglängsachse und die Mittelpunkte sind äquidistant verteilt. Abbildung 5.4a zeigt eine mögliche Überdeckung eines Volkswagen Passats durch fünf gleichgroße Kreise (rot). Die Mittelpunkte der Kreise sind blau dargestellt. Man ist nun an der *Manövrierfläche* einer Trajektorie interessiert, das ist die Fläche, die das Fahrzeug überstreicht, wenn es entlang dieser geführt wird.

Die Trajektorie sei wie in Abschnitt 5.1 als Menge von  $n$  Abtastpunkten  $\mathbf{x}_i = (x_i, y_i)^\top, i \in \{0, \dots, n-1\}$  repräsentiert. Man nimmt an, dass sich das Fahrzeug auf jedem linearen Teilstück  $\overline{\mathbf{x}_i \mathbf{x}_{i+1}}$  der Trajektorie gradlinig bewegt. Die Manövrierfläche auf dem Teilstück kann deshalb wie in Abbildung 5.4b angenähert werden, in dem die Mittelpunkte der Kreise um die Länge des Teilstückes gespreizt werden. Man berechnet nun die Lage der  $m$  Kreismittelpunkte  $\mathbf{c}_{i,j}, 0 < j < m$  für das Stück  $\overline{\mathbf{x}_i \mathbf{x}_{i+1}}$ . Sei  $l_{\min}$  die Lage des Mittelpunktes des achterlichsten Kreises entlang der Längsachse des Fahrzeuges, relativ zur Hinterachse,  $l_{\max}$  die des vordersten und  $L = l_{\max} + l_{\min}$  der Abstand dieser beiden äußersten Kreise. Die Länge des Teilstückes berechnet sich zu

$$l_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \quad (5.11)$$

und ein normalisierter Tangentenvektor  $\mathbf{t}_i$  ist

$$\mathbf{t}_i = \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{l_i}. \quad (5.12)$$

Die Lage des  $j$ ten Kreismittelpunktes ist nun

$$\mathbf{c}_{i,j} = \mathbf{x}_i + (\lambda_j(l_i + L) - l_{\min})\mathbf{t}_i \quad (5.13)$$

mit

$$\lambda_j = \frac{i}{m-1}. \quad (5.14)$$

Abbildung 5.5 zeigt die auf diese Weise angenäherte Manövrierfläche für eine klothoidenförmige Beispieltrajektorie.

Der beschriebene Ansatz enthält – über die Formapproximation hinaus – zwei Vereinfachungen: Erstens werden Schwimmwinkel vernachlässigt, es wird also angenommen, dass das Fahrzeug stets entlang der Trajektorie ausgerichtet ist. Zweitens wird angenommen, dass das Fahrzeug sich um den Hinterachspunkt dreht. Beide Vereinfachungen gelten in guter Näherung bei langsamer Fahrt (<15 km/h). Andererseits sind bei schnellerer Fahrt nur geringe Krümmungen der Trajektorie physikalisch möglich, so dass die Manövrierfläche durch das geringe Ausschwenken des Fahrzeuges kaum über die Fahrzeugbreite hinaus vergrößert wird. Das Verfahren berechnet deshalb – insbesondere, wenn die Kreisradien konservativ überabgeschätzt werden – auch bei höheren Geschwindigkeiten eine ausreichend gute Approximation der Manövrierfläche.

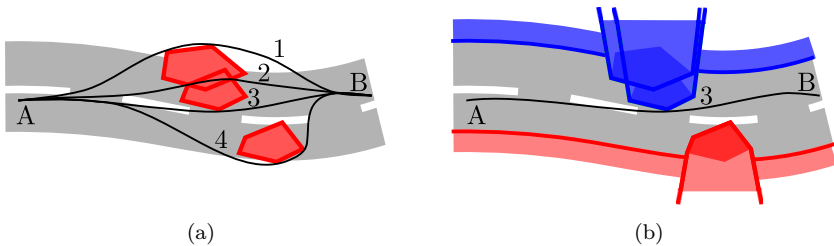
### Zeitabhängige Polygone

Sämtliche äußere Nebenbedingungen werden durch Polygonzüge ausgedrückt. Sei  $G = \{g_i : 0 \leq i < k-1, i \in \mathbb{N}\}$  die Menge der  $k$  Polygonzüge, die die Umgebung beschreibt. Für sie gibt es eine Distanztransformation  $d(\mathbf{x}, g), g \in G$  wie in Kapitel 3. Für den  $i$ ten Punkt  $\mathbf{x}_i$  der Trajektorie, jede Geometrie  $g \in G$  und jeden Kreismittelpunkt der Formapproximation ergibt sich nun eine Nebenbedingung

$$d(\mathbf{c}_{i,j}, g) - r_{\text{fzg}} > 0, j \in \{0, \dots, m-1\}, g \in G, \quad (5.15)$$

insgesamt für die Trajektorie also  $km(n-1)$  zusätzliche Ungleichungen. Die Lage der Polygonzüge kann hierbei auch zeitabhängig sein, um bewegte Objekte beschreiben zu können.

Die Auslegung der Nebenbedingungspolygone muss sorgfältig erfolgen. Es soll ja ein lokales, kontinuierliches Verfahren für die Optimierung der Trajektorie verwendet werden. In Abbildung 5.6a ist schematisch dargestellt, dass es durch die Lage der Hindernisse zu mehreren stationären Punkten im Optimierungsprozess kommen kann. Es soll eine Trajektorie von A nach



**Abbildung 5.6:** Stationäre Punkte bei der Optimierung der Trajektorie (a) ohne (b) mit vorgeschaltetem Entscheidungsprozess.

B geplant werden. Alle eingezeichneten Trajektorien sind stationäre Punkte des Optimierungsprozesses, sie können also nicht mehr lokal verbessert werden. Es kann nicht vorhergesehen werden, in welchem dieser Punkte die Optimierung konvergieren wird. Besonders problematisch ist die mit der Ziffer 2 gekennzeichnete Trajektorie, da sie offensichtlich zwei äußere Nebenbedingungen verletzt. Sie ist dennoch stationär, da – lokal betrachtet – keine der beiden verletzten Nebenbedingungen verbessert werden kann, ohne die andere zu verschlechtern. Deshalb muss die Form der Polygone modifiziert werden. In Abbildung 5.6b ist eine „gute“ Auslegung der Polygone dargestellt. Basierend auf dem Fahrbahnverlauf wurden Hindernisse dem linken (blau) oder rechten (rot) Fahrbahnrand zugeordnet. Auf diese Weise wurde entschieden, welches Hindernis links und welches rechts zu passieren ist. Entsprechend wurden die Polygone nach außen erweitert. Sie erzeugen jetzt keine stationären Punkte mehr, außer der tatsächlichen Optimaltrajektorie.

Im Allgemeinen muss die polygonale Repräsentation der Hindernisse erst aus einer niedrigeren Repräsentationsstufe abgeleitet werden. Es sind keine Fahrzeugsensoren bekannt, die die Umgebung direkt in Form von Polygonzügen erfassen. Später in Abschnitt 7.1 wird gezeigt, wie die Daten des für die BERTHA-BENZ-Fahrt verwendeten Stereosystems in eine geeignete polygonale Repräsentation aufbereitet wurden.

### 5.3.2 Innere Nebenbedingungen

Über die in den vorangehenden Abschnitten beschriebenen externen Nebenbedingungen hinaus gibt es interne Nebenbedingungen, die sich aus der Fahrzeugkinematik und -dynamik ergeben. Bei niedriger

Geschwindigkeit wird der Verlauf der Trajektorie zuerst durch die Lenkgeometrie des Fahrzeuges limitiert. Über den in Abschnitt 2.2 beschriebenen Zusammenhang bedeutet dies eine maximal mögliche Krümmung der Trajektorie,

$$|\kappa(t)| < \kappa_{\max}. \quad (5.16)$$

Bei höheren Geschwindigkeiten wird die Grenze der Querdynamik in der Regel mit dem Auftreten der maximalen Reibkraft der Reifen erreicht. Diese Grenze kann als der KAMM'sche Kreis [Kam37; Pac06] veranschaulicht werden und es muss gelten

$$\|\ddot{\mathbf{x}}(t)\| < a_{\max}. \quad (5.17)$$

(5.16) und (5.17) werden wie das Energiefunktional über finite Differenzen näherungsweise ausgedrückt. Mit der Notation aus Abschnitt 2.3 gilt

$$\dot{\mathbf{x}}_i \approx \widehat{\mathbf{x}}_i = \frac{\delta \mathbf{x}_i}{h}, \quad (5.18)$$

$$\ddot{\mathbf{x}}_i \approx \widehat{\ddot{\mathbf{x}}}_i = \frac{\delta^2 \mathbf{x}_i}{h^2} \quad (5.19)$$

und

$$\kappa_i \approx \widehat{\kappa}_i = \frac{\delta x_i \delta^2 y_i - \delta y_i \delta^2 x_i}{\sqrt{\delta x_i^2 + \delta y_i^2}^3}. \quad (5.20)$$

Es ergeben sich anstelle von (5.16) die  $2(N-1)$  Ungleichungen

$$\begin{aligned} \kappa_{\max} - \widehat{\kappa}_i &> 0 \\ -\kappa_{\max} + \widehat{\kappa}_i &> 0 \end{aligned}$$

sowie anstelle von (5.17) die  $N-1$  Ungleichungen

$$a_{\max}^2 - \|\widehat{\ddot{\mathbf{x}}}_i\|^2 > 0. \quad (5.21)$$

## 5.4 Randbedingungen

Zur Bestimmung einer eindeutigen Optimaltrajektorie müssen im Allgemeinen Randbedingungen, also Start- und Endzustände an den Rändern des zeitlichen Planungsintervalls  $[t_{\text{start}}, t_{\text{ende}}]$ , vorgegeben werden. Für die Startbedingungen bei  $t_{\text{start}}$  ergeben diese sich aus dem aktuellen (oder dem in die Zukunft extrapolierten) Fahrzeugzustand. Die Trajektorie wird ja

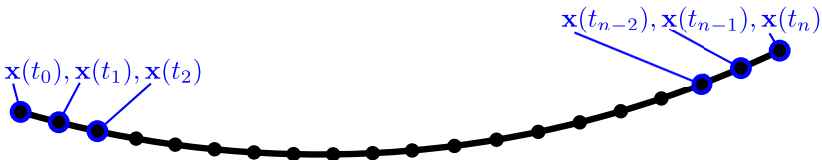


Abbildung 5.7: Symmetrische Randbedingungen

fortlaufend neu geplant und dient dabei als Führungsgröße eines Fahrzeugreglers. Sprünge in dessen Stellgrößen zum Zeitpunkt  $t_{\text{start}}$  sollen vermieden werden. Im Absatz 5.4.2 wird der Ablauf der fortlaufenden Neuplanung und die damit verbundene Behandlung der Startwerte genauer beschrieben.

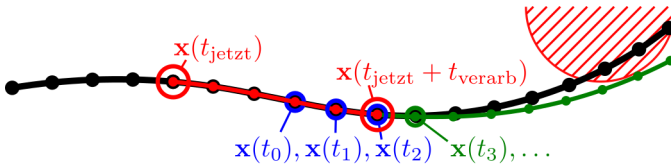
### 5.4.1 Symmetrische Randbedingungen, Transversalität

Wenn sowohl Start- als auch Endwerte der Trajektorie festgehalten werden, so werden diese im Folgenden als *symmetrischen Randbedingungen* bezeichnet. Ist  $t_{\text{ende}}$  von vornherein festgelegt, so ergeben sich hierdurch weitere Gleichungen als notwendige Bedingungen für das Optimierungsproblem. Im Falle der durch eine feste Anzahl Stützstellen repräsentierten Trajektorie können diese Randbedingungen realisiert werden, indem man einfach eine bestimmte Anzahl der Randpunkte auf konstanten Werten festhält. Abbildung 5.7 zeigt dies für den Fall, dass die Trajektorie und ihre ersten zwei Ableitungen an den Rändern festgelegt werden sollen. Jeweils drei Stützpunkte müssen dann am Anfang und Ende der Trajektorie festgehalten werden.

Im Allgemeinen kann nicht vorher bestimmt werden, welche Reisezeit für das Erreichen des Endzustandes vorzusehen ist. Der rechte Rand  $t_{\text{ende}}$  muss dann als freier Parameter bei der Minimierung des Kostenfunktional (4.1) behandelt werden und muss „mit optimiert werden“. Ein derartiges Variationsproblem wird als *transversal* bezeichnet.

### 5.4.2 Fortlaufende Neuplanung

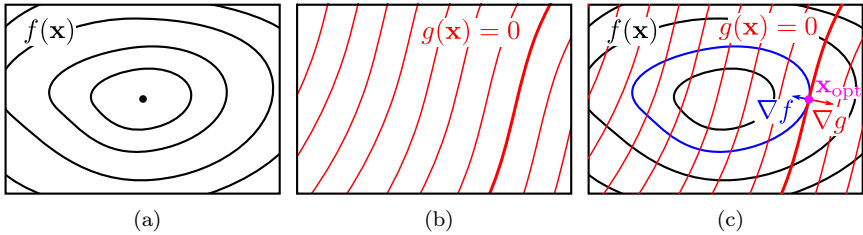
Symmetrische und transversale Probleme wie im letzten Abschnitt beschrieben werden erst später mit globalen, diskreten Lösungsverfahren (Kapitel 6) behandelt. Für die in diesem Kapitel entwickelten kontinuierlichen Lösungsverfahren, die ja für das Fahren in fließendem Verkehr einge-



**Abbildung 5.8:** Randbedingungen bei fortlaufender Neuplanung

setzt werden, werden Sie nicht betrachtet. Es soll auf die künstliche Festlegung eines Endwertes  $\mathbf{x}_{\text{ende}}$  zu einem vorher bestimmten Zeitpunkt  $t_{\text{ende}}$  verzichtet werden. In fließendem Verkehr ist das Ziel der Trajektorienplanung ja in der Regel über einem unendlichen Zeitbereich definiert. Informell wird dies durch Beschreibungen ausgedrückt wie beispielsweise „das Fahrzeug soll mit der Sollgeschwindigkeit der Straße folgen und dabei in der Mitte des Fahrstreifens fahren“. Im exemplarischen Kostenfunktional (5.5) wird diese Anforderung durch den Ablageterm (5.6) und den Geschwindigkeitsterm (5.7) formalisiert. In den (analytisch lösbaren) Beispielen 4.2 und 4.3 aus Abschnitt 4.2 ließ man die rechte Integrationschranke  $t_{\text{ende}}$  gegen  $\infty$  gehen. Wenn man sich mit einer Diskretisierung des Energiefunktionales wie in Abschnitt 5.1 beschrieben behilft, ist das nicht möglich. Man wählt dann die rechte Integrationschranke so groß wie möglich, um einen Minimierer des unbeschränkten Integrals möglichst gut anzunähern.

Während das Fahrzeug entlang einer geplanten Trajektorie geregelt wird und auf diese Weise Fortschritt entlang der Straße macht, gelangen immer neue Teile des Umgebung in den Erfassungsbereich seiner Sensoren. Die Trajektorienplanung muss deshalb ständig wiederholt werden um sich an neu erfasste Hindernisse anzupassen. Damit die Krümmungstetigkeit der Trajektorie nicht verletzt wird, wird dabei aus der jeweils zuletzt geplanten Trajektorie eine Startbedingung zweiter Ordnung erzeugt (vergleiche Abschnitt 5.4.1), es werden also die ersten drei Punkte der neuen Trajektorie auf Werten der letzten Trajektorie festgehalten. Abbildung 5.8 zeigt, wie genau diese drei Punkte gewählt werden. In schwarz ist die Trajektorie mit ihren Stützpunkten dargestellt, die zum aktuellen Zeitpunkt  $t_{\text{jetzt}}$  gerade verfügbar wurde. Zum gleichen Zeitpunkt wurde das rot schraffierte eingezeichnete Objekt erfasst - im letzten Planungszyklus wurde es noch nicht berücksichtigt. Zur Vereinfachung wird in diesem Abschnitt angenommen, das  $t_{\text{jetzt}}$  und andere Zeitpunkte jeweils genau mit einer der diskreten Abtastzeiten  $t_i$  der Stützpunkte zusammenfallen. Im Allgemeinen müssen die Zeiten hierzu erst gerundet werden. Nun soll eine neue Trajektorie berechnet



**Abbildung 5.9:** Gleichheits-Nebenbedingung. (a) Kostenfunktion  $f(\mathbf{x})$  anhand von Isolinien dargestellt. (b) Nebenbedingung  $g(\mathbf{x}) = 0$  und Isolinien von  $g$ . (c) Beim Minimum von  $f(\mathbf{x})$  (unter der Nebenbedingung  $g(\mathbf{x}) = 0$ ) berührt die Gerade  $g(\mathbf{x}) = 0$  eine Isolinie (blau) von  $f(\mathbf{x})$ .

werden, und es wird angenommen, dass die Berechnung dieser Trajektorie eine Zeit von höchstens  $t_{\text{verarb}}$  in Anspruch nehmen wird. Dies bedeutet, dass davon ausgegangen werden muss, dass das Fahrzeug bis zum Zeitpunkt  $t_{\text{jetzt}} + t_{\text{verarb}}$  entlang der alten Trajektorie geregelt wird - dieser Abschnitt ist in Abbildung 5.8 rot dargestellt. Dieser Bereich darf nicht mehr verändert werden, damit keine Sprünge in den Eingangsgrößen des Fahrzeuges erzeugt werden. Die Startbedingung ergibt sich nun aus den letzten drei Punkten dieses Bereiches, die in der Abbildung blau dargestellt sind. Die neu geplante Trajektorie ist grün.

## 5.5 Nichtlineare Optimierung

In den vorangehenden Kapiteln wurde beschrieben, wie das Problem der Trajektorienplanung von der Form eines Variationsproblems in ein gewöhnliches Extremwertproblem mit Nebenbedingungen überführt werden kann. Zielfunktion und Nebenbedingungen sind dabei im Allgemeinen nichtlinear. Dieser Abschnitt bietet einen kurzen Überblick über bei der nichtlinearen Optimierung verwendete Techniken.

### 5.5.1 LAGRANGE-Multiplikatoren

Um das Minimum einer *Zielfunktion*  $f(\mathbf{x})$  unter einer *Gleichheits-Nebenbedingung*

$$g(\mathbf{x}) = 0 \tag{5.22}$$



zu lösen, bedient man sich der Methode der LAGRANGE-Multiplikatoren. Die Herleitung hier soll das Prinzip veranschaulichen und erfolgt unbekümmert in Bezug auf Stetigkeitsbedingungen der beteiligten Funktionen. Auch wird angenommen, dass stationäre Punkte einer Funktion stets auch Extrema dieser Funktion darstellen (man kann leicht zeigen, dass dies für ein *konvexes* Problem stets der Fall ist). Für eine rigidiere Herleitung des Formalismus siehe Fletcher [Fle87].

Abbildung 5.9a zeigt exemplarisch eine in der  $(x, y)$ -Ebene definierte, skalarwertige *Zielfunktion*  $f(\mathbf{x})$ ,  $\mathbf{x} = (x, y)^\top$ , anhand einiger ihrer Isolinien (Linien gleichen Funktionswertes). Es soll nun das Minimum dieser Funktion ermittelt werden, das aber, als zusätzliche Nebenbedingung, auf einer vorgegebenen Kurve liegen soll. Diese Kurve sei implizit über die Gleichung (5.22) mit der *Nebenbedingungsfunktion*  $g(\mathbf{x})$  definiert. Die Nebenbedingungsfunktion kann zum Beispiel die vorzeichenbehaftete Abstandsfunktion (Kapitel 3) zur gewünschten Kurve darstellen. In Abbildung 5.9b ist  $g(\mathbf{x})$  wieder anhand einiger Isolinien dargestellt, die Isolinie  $g(\mathbf{x}) = 0$  (also der Ort aller Punkte, die die Nebenbedingung (5.22) erfüllen) ist hierbei hervorgehoben. In Abbildung 5.9c sind beide Funktionen einander überlagert. Die Stelle  $\mathbf{x}_{\text{opt}}$  des durch (5.22) restringierten Minimums ist eingezeichnet. Man sieht, dass an dieser Stelle die Nebenbedingung gerade von einer Isolinie der Zielfunktion *berührt* - also nicht geschnitten - wird. Die entsprechende Isolinie ist blau hervorgehoben. Offensichtlich führt unter dieser Bedingung eine Verschiebung von  $\mathbf{x}_{\text{opt}}$  entlang der Nebenbedingung stets zu einer Vergrößerung des Wertes der Zielfunktion. Diese Bedingung ist äquivalent dazu, dass die Gradienten  $\nabla f$  und  $\nabla g$  parallel sind - die Gradienten sind in Abbildung 5.9c als Pfeile eingezeichnet. Formal sucht man also Punkte  $\mathbf{x}$ , die Gleichung (5.22) erfüllen und für die zusätzlich gilt

$$\begin{aligned} \nabla_{x,y} f(\mathbf{x}) &= -\lambda \nabla_{x,y} g(\mathbf{x}), \\ &\Leftrightarrow \\ \nabla_{x,y} f(\mathbf{x}) + \lambda \nabla_{x,y} g(\mathbf{x}) &= \mathbf{0}, \end{aligned} \tag{5.23}$$

für irgendein  $\lambda \in R$ . Die Hilfsvariable  $\lambda$  nennt man *Lagrange-Multiplikator* (das negative Vorzeichen ist Konvention). Sie kompensiert die unterschiedliche Länge und gegebenenfalls ein unterschiedliches Vorzeichen der Gradientenvektoren, die ja nur parallel, aber nicht etwa gleich lang sein müssen. Joseph-Louis Lagrange erkannte, dass die linken Seiten der Gleichungen (5.22) und (5.23) die partiellen Ableitungen einer Hilfsfunktion

$$\Lambda(x, y, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}). \tag{5.24}$$

sind, die nach ihm *Lagrange-Funktion* benannt wurde. Man sieht nun, dass restringierte Extrema von  $f$  stationäre (bzw. extremale) Punkte von  $\Lambda$  sind, da sich (5.22) und (5.23) über die Gleichung

$$\nabla_{x,y,\lambda}\Lambda(x,y,\lambda) = \mathbf{0}, \quad (5.25)$$

ausdrücken lassen. Das beschriebene Vorgehen lässt sich auf mehrere Gleichheitsnebenbedingungen erweitern, bei  $m$  Nebenbedingungen ergibt sich dann als Bedingung für ein restringiertes Extremum

$$\Lambda(x,y,\lambda) = f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g(\mathbf{x}) \quad (5.26)$$

mit den  $m$  einzelnen LAGRANGE-Multiplikatoren  $\lambda_i$ . Vereinfacht ausgedrückt reduziert die Methode der LAGRANGE-Multiplikatoren das Problem, ein durch Gleichheits-Nebenbedingungen restringiertes Extremum zu finden, auf die Suche eines unrestringierten Extremums einer Hilfsfunktion, die pro Nebenbedingung einen zusätzlichen Freiheitsgrad aufweist. In einfachen Fällen – hierzu gehört der im nächsten Abschnitt diskutierte wichtige Fall einer quadratischen Zielfunktion mit einer linearen Nebenbedingung – kann man hierzu Gleichung 5.25 analytisch lösen. Andernfalls behilft man sich mit numerischen Methoden, zum Beispiel dem NEWTON-Verfahren [Fle87].

## 5.5.2 Quadratisches Programmieren

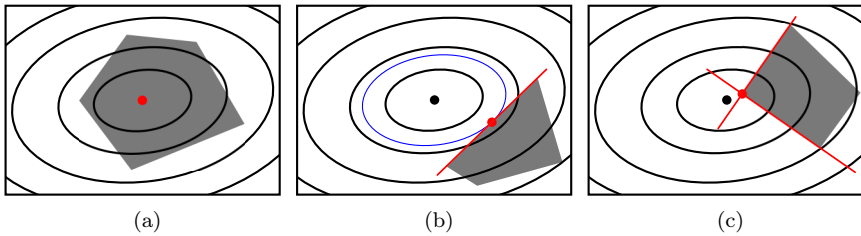
Ein quadratisches Programm (QP) ist ein konvexes Optimierungsproblem, bei dem die Zielfunktion eine quadratische Form darstellt, zu minimieren ist also

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x}, \quad (5.27)$$

mit  $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$  und  $\mathbf{A} \in \mathbb{R}^{n \times n}$ . Die Matrix  $\mathbf{A}$  muss positiv definit sein – nur dann ist  $f$  konvex und weist ein wohldefiniertes Minimum auf. Die Minimierung soll unter  $m$  linearen Nebenbedingungen erfolgen. Dies können sowohl Gleichungen als auch Ungleichungen sein. Im Rahmen der Trajektorienplanung erscheinen im Rahmen dieser Arbeit nur Ungleichungen als Nebenbedingungen, weshalb im Folgenden nur diese betrachtet werden. Die  $m$  linearen Nebenbedingungen kann man kompakt als Vektorungleichung

$$\mathbf{C} \mathbf{x} - \mathbf{d} \geq \mathbf{0} \quad (5.28)$$

schreiben, mit  $\mathbf{C} \in \mathbb{R}^{m \times n}$  und  $\mathbf{d} \in \mathbb{R}^m$ . Die Ungleichungen (5.28) bedeuten, dass der zulässige Bereich die Schnittmenge von  $m$  Halbräumen darstellt.



**Abbildung 5.10:** Lösungsklassen für ein zweidimensionales quadratisches Programm. Die quadratische Funktion ist durch einige ihrer Isolinien dargestellt. Der zulässige Bereich ist schattiert. Aktive Nebenbedingungen sind als rote Geraden dargestellt. Das Optimum ist beim roten Punkt. (a) Optimum innerhalb des zulässigen Bereichs. (b) Optimum an einer Kante des zulässigen Bereichs, eine aktive Nebenbedingung. (c) Optimum in einer Ecke des zulässigen Bereichs, zwei aktive Nebenbedingungen.

Hierbei handelt es sich stets um ein konvexes Polytop (das aber auch leer sein kann). Für den anschaulichen Fall  $n = 2$  repräsentiert jede der  $m$  Ungleichungen (5.28) eine Halbebene, und deren Schnittmenge stellt ein konvexes Polygon dar.

Aus Gründen der Anschaulichkeit wird für die nächsten Überlegungen das zweidimensionale Problem betrachtet, es gelte also im Folgenden  $n = 2$  und  $\mathbf{x} = (x, y)^\top$ . Beim zweidimensionalen Problem können drei Lösungsklassen unterschieden werden, die in Abbildung 5.10 veranschaulicht sind. In Abbildung 5.10a befindet sich das Optimum im Inneren des zulässigen Bereichs. Dies bedeutet, dass die Lösung des durch Nebenbedingungen restringierten Problems identisch ist mit der Lösung des unrestringierten Problems. Wegen der speziellen Form von (5.27) lässt sich diese Lösung geschlossen angeben. Die Ableitung  $\nabla_{\mathbf{x}} f$  wird hierfür zu  $\mathbf{0}$  gesetzt,

$$\nabla_{\mathbf{x}} f = (\mathbf{A}\mathbf{x} + \mathbf{b})^\top = \mathbf{0}. \quad (5.29)$$

Das lineare Gleichungssystem (5.29) kann mit Standardverfahren gelöst werden. Im in Abbildung 5.10b dargestellten Fall dagegen befindet sich das Optimum auf dem Rand des zulässigen Bereichs, und zwar auf einer seiner Kanten. Genau eine Nebenbedingung ist *aktiv* - das bedeutet, dass nach Einsetzen des Optimums für  $\mathbf{x}$  die linke Seite von (5.28) in genau einer Zeile gleich 0 ist, während sie in allen anderen Zeilen echt größer 0 ist. Falls bekannt ist, welche der Nebenbedingung aktiv ist, so kann sie als einzelne

Gleichheits-Nebenbedingung behandelt werden. Sei in diesem Fall

$$g(\mathbf{x}) = \mathbf{c}\mathbf{x} - d = 0 \quad (5.30)$$

die aktive Nebenbedingung,  $\mathbf{c}$  ist hierbei die entsprechende Zeile von  $\mathbf{C}$ , und  $d$  der entsprechende Eintrag im Vektor  $\mathbf{d}$ . Für die Nebenbedingung wird ein LAGRANGE-Multiplikator  $\lambda$  eingeführt. Nach Abschnitt 5.5.1 wird nun die LAGRANGE-Funktion

$$\Lambda(x, y, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$$

aufgestellt. Es muss gelten

$$\nabla_{x,y,\lambda}\Lambda(x, y, \lambda) = \mathbf{0}, \quad (5.31)$$

also

$$\begin{aligned} \nabla_{x,y}\Lambda(x, y, \lambda) &= (\mathbf{A}\mathbf{x} + \mathbf{b})^\top + \lambda\mathbf{c} \\ &= \mathbf{0}, \end{aligned} \quad (5.32)$$

$$\begin{aligned} \nabla_\lambda\Lambda(x, y, \lambda) &= \mathbf{c}\mathbf{x} - d, \\ &= 0. \end{aligned} \quad (5.33)$$

Die beiden Gleichungen (5.34) und (5.32) lassen sich kompakt als

$$\begin{pmatrix} \mathbf{A} & \mathbf{c}^T \\ \mathbf{c} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} -\mathbf{b} \\ d \end{pmatrix} \quad (5.34)$$

schreiben - es handelt sich hierbei wieder um ein lineares Gleichungssystem, das sich leicht lösen lässt.

Auch im in Abbildung 5.10c dargestellten Fall befindet sich das Optimum am Rand des zulässigen Bereiches, es liegt allerdings in einer seiner Ecken. In diesem Fall sind zwei Nebenbedingungen aktiv, die die Stelle des Optimums bereits festlegen. Es muss nur noch der Schnittpunkt der beiden Geraden bestimmt werden.

Wie man sieht, lässt sich, falls bekannt ist, welche Nebenbedingungen für die Optimallösung aktiv sind, das QP leicht lösen. Die eigentliche Herausforderung besteht also darin, die Menge aktiver Nebenbedingungen zu bestimmen. Prinzipiell könnten hierzu alle gültigen Kombinationen aus Nebenbedingungen ausprobiert werden. Man entscheidet sich dann für die Kombination, deren Lösung den kleinsten Wert von  $f$  hervorbringt. Es

gibt in der Praxis effizientere Verfahren [GI83; NW06], um die Menge aktiver Nebenbedingungen zu finden. Ausgehend von einem Startwert wandern diese auf dem Rand des zulässigen Bereiches entlang. Das Verfahren kann abgebrochen werden, sobald ein Punkt gefunden ist, der die sogenannten Karush-Kuhn-Tucker-Bedingungen (KKT-Bedingungen) erfüllt. Bei den KKT-Bedingungen handelt es sich um eine Verallgemeinerung der LAGRANGE-Bedingung (5.25), für Optimierungsprobleme mit Ungleichungen [Fle87].

**Beispiel** Quadratische Programme sind dem Anschein nach eine recht beschränkte Problemklasse. Dennoch eignet sich das quadratische Programmieren durchaus, um praxisrelevante Trajektorienplanungsprobleme zu lösen. Als Beispiel wird ein Kostenfunktional betrachtet, das Beispiel 4.2 ähnlich ist. Zu minimieren sei

$$E[x(t)] = \int_{t_0}^{t_1} L(x, \dot{x}, \ddot{x}) dt, \quad (5.35)$$

mit

$$L(x, \dot{x}, \ddot{x}) = w_{\text{beschl}} \ddot{x}^2 + w_{\text{vquer}} (\dot{x} - v_{\text{ref}})^2 + w_{\text{ablage}} (x - x_{\text{ref}})^2, \quad (5.36)$$

und Gewichtungsfaktoren  $w_{\text{beschl}} = 1$ ,  $w_{\text{vquer}} > 0$  und  $w_{\text{ablage}} > 0$ . Nach der Methode finiter Differenzen – vergleiche Abschnitt 5.1 – wird das Variationsproblem (5.35) in ein Extremwertproblem umgewandelt. Der Einfachheit halber wird für die Abtastung eine zeitliche Schrittweite von  $h = 1$  gewählt, und man erhält

$$E_{\text{d}}(\mathbf{x}) = \sum_{i=1}^{n-2} L(x_i, x_{i+1} - x_i, x_{i+1} - 2x_i + x_{i-1}), \quad (5.37)$$

mit dem Vektor  $\mathbf{x}$  aus  $n$  Stützstellen,  $\mathbf{x} = (x_0, x_1, \dots, x_{n-1})^T$ . Der  $i$ -te Summand von (5.37) ist eine quadratische Form

$$E_{\text{d},i}(\mathbf{x}_i) = \mathbf{x}_i^T \mathbf{A}_i \mathbf{x}_i + \mathbf{b}_i \mathbf{x}_i \quad (5.38)$$

mit

$$\mathbf{x}_i = (x_{i-1}, x_i, x_{i+1})^T \quad (5.39)$$

und

$$\mathbf{A}_i = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 + w_{\text{geschw}} + w_{\text{ablage}} & -w_{\text{geschw}} - 2 \\ 1 & -w_{\text{geschw}} - 2 & w_{\text{geschw}} + 1 \end{pmatrix} \quad (5.40)$$

$$A = \begin{pmatrix} \boxed{A_0} & & & \mathbf{0} \\ & \boxed{A_i} & & \\ & & \ddots & \\ & & & \boxed{A_{n-1}} \\ \mathbf{0} & & & & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \boxed{A_0} & & & \mathbf{0} \\ & \ddots & & \\ & & \ddots & \\ \mathbf{0} & & & \mathbf{0} \end{pmatrix} + \dots + \begin{pmatrix} \ddots & & & \mathbf{0} \\ & \boxed{A_i} & & \\ & & \ddots & \\ \mathbf{0} & & & \mathbf{0} \end{pmatrix} \dots$$

**Abbildung 5.11:** Die Summierung der Einzeltermen der Kostenfunktion führt zu einer banddiagonalen HESSE-Matrix.

sowie

$$\mathbf{b}_i = \left( 0 \quad 2(w_{\text{geschw}}v_{\text{ref}} - w_{\text{ablage}}x_{\text{ref}}) \quad -2w_{\text{geschw}}v_{\text{ref}} \right) \quad (5.41)$$

Auch die Summe der Einzeltermen  $E(\mathbf{x}_i)$  ist eine quadratische Form. Es ist

$$\begin{aligned} E_d(\mathbf{x}) &= \sum_{i=1}^{n-2} E_{d,i}(\mathbf{x}_i) \\ &= \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}, \end{aligned}$$

wobei  $A$  eine banddiagonale Matrix ist. Abbildung 5.11 zeigt schematisch auf, wie die Bandsstruktur von  $A$  zustande kommt. Eine derartige Bandstruktur ist typisch für Probleme in der Trajektorienplanung, die über Kollokation diskretisiert werden. Sie ist äußerst erstrebenswert, weil so die Teilprobleme (5.29) und (5.34) sehr effizient lösbar werden. Ein banddiagonales Gleichungssystem mit  $k$  Unbekannten kann mit einer Anzahl Operationen in  $\mathcal{O}(k)$  gelöst werden. Ist das gleiche System dicht besetzt, so wird hierfür eine Anzahl Operationen in  $\mathcal{O}(k^3)$  benötigt.

Das Ausgangsproblem wird nun mit Nebenbedingungen versehen: Position  $x$  und Geschwindigkeit  $\dot{x}$  der Trajektorie sollen beschränkt sein,

$$x_{\min} < x < x_{\max}, \quad (5.42)$$

$$v_{\min} < \dot{x} < v_{\max}. \quad (5.43)$$

Auf (5.42) und (5.43) wird wieder die gleiche Zeitdiskretisierung wie oben angewendet, und man erhält für alle  $i$  vier lineare Ungleichungen

$$x_{\min} < x_i < x_{\max}, \quad (5.44)$$

$$v_{\min} < x_{i+1} - x_i < v_{\max}, \quad (5.45)$$

$$C = \begin{pmatrix} \boxed{C_0} & & & \mathbf{0} \\ & \boxed{C_1} & & \\ & & \boxed{C_2} & \\ \mathbf{0} & & & \vdots \end{pmatrix}$$

**Abbildung 5.12:** Struktur der Nebenbedingungsmatrix

Für jedes  $i$  lassen sich diese in Matrixschreibweise angeben,

$$C_i \begin{pmatrix} x_i \\ x_{i+1} \end{pmatrix} - \mathbf{d}_i \geq \mathbf{0} \quad (5.46)$$

mit

$$C_i = \begin{pmatrix} 1 & 0 \\ -1 & 1 \\ -1 & 0 \\ 1 & -1 \end{pmatrix} \quad (5.47)$$

und

$$\mathbf{d}_i = (x_{\min} \ v_{\min} \ -x_{\max} \ -v_{\max})^\top. \quad (5.48)$$

Man kann nun die  $C_i$  blockweise zur Matrix  $C$  zusammenfassen wie in Abbildung 5.12 und erhält ein System aus Ungleichungen

$$C\mathbf{x} - \mathbf{d} \geq \mathbf{0}, \quad (5.49)$$

wie (5.53), mit

$$\mathbf{d} = (\mathbf{d}_0 \ \mathbf{d}_1 \ \dots \ \mathbf{d}_{n-1}). \quad (5.50)$$

Man wird im Allgemeinen noch Randbedingungen (vergleiche Abschnitt 5.4) vorgeben wollen. In diesem Beispiel können Randbedingungen für Position und Geschwindigkeit vorgegeben werden. Eine Randbedingung für die Beschleunigung - oder eine mit noch höherer Ordnung - wäre wirkungslos, da ja das Kostenfunktional eine *Änderung* der Beschleunigung gar nicht mit Kosten belegt. Die Beschleunigung kann sich also am Rande - wie überall auf der Trajektorie - instantan ändern. Die Randbedingungen werden wie in Abschnitt 5.4 beschrieben ausgedrückt, indem Stützpunkte am Rand der

Trajektorie festgelegt und nicht mit optimiert werden. Im Beispiel müssen an den Rändern je zwei Stützpunkte festgelegt werden. Man kann leicht zeigen, dass das derart durch Randbedingungen restringierte Problem immer noch ein quadratisches Programm darstellt. Abbildung 5.13a zeigt die konkrete Ausprägung des Optimierungsproblems aus diesem Beispiel. Die Trajektorie wurde auf 100 Stützstellen (blau) diskretisiert. An den Rändern wurden symmetrische Randbedingungen für Position und Geschwindigkeit (magenta) festgelegt. In diesem Beispiel ist  $v_{\text{ref}} = w_{\text{ablage}} = 0$ .

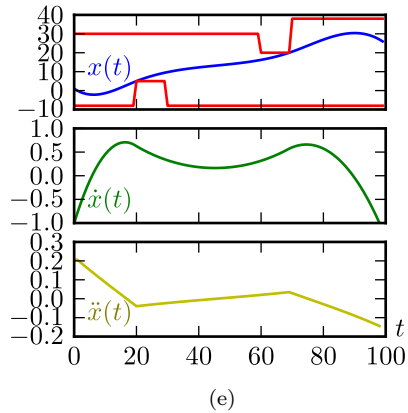
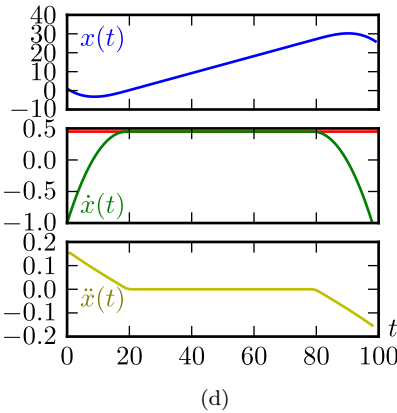
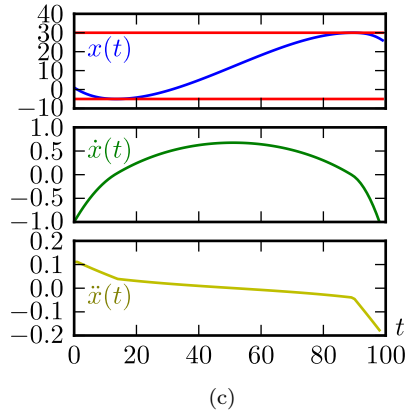
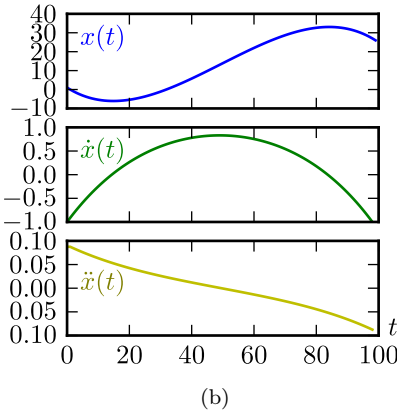
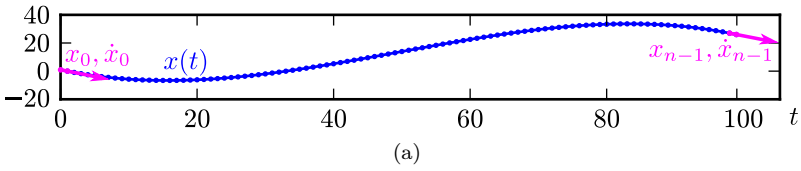
Die Abbildungen 5.13b-5.13e zeigen Ergebnisse für das Beispielproblem, wobei unterschiedliche Nebenbedingungen gewählt wurden. Abbildung 5.13b zeigt die Lösung des unrestringierten Problems. Für Abbildung 5.13c wurde die Position der Trajektorie oben und unten beschränkt. Für Abbildung 5.13d wurde eine Maximalgeschwindigkeit festgelegt. Für Abbildung 5.13e schließlich wurde die Nebenbedingung (5.42) verallgemeinert, und zwar derart, dass die Positionsbeschränkung für jeden zeitlichen Abtastpunkt unterschiedlich sein kann, also

$$x_{\min,i} < x_i < x_{\max,i}. \quad (5.51)$$

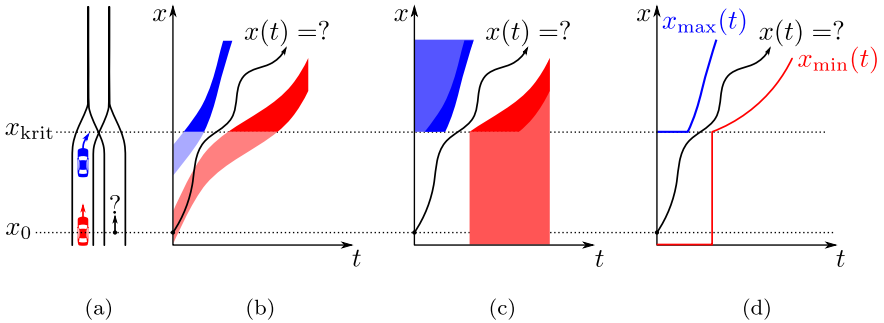
Es ist bemerkenswert, dass sich die Lösungen zu all diesen Problemen durch das quadratische Programmieren in einer endlichen Anzahl Schritten *exakt* (bis auf die zeitliche Diskretisierung) bestimmen lassen, ohne, dass eine iterative Näherung notwendig wäre: Im *worst case* müssen – analog zu Abbildung 5.10 – alle Ecken und Kanten (bzw. höherdimensionale Entsprechungen) des Nebenbedingungs-Polytops aufgezählt werden. In jedem Schritt muss ein lineares System analog zu 5.34 gelöst werden. Die Anzahl der Schritte ist linear in der Anzahl der Nebenbedingungen.

Die Variante mit zeitlich varianter Positionsbeschränkung wie in Abbildung 5.13e lässt sich in der Praxis zum Beispiel nutzen, um Einfädelvorgänge optimal zu planen oder sogar zu regeln. Abbildung 5.14 illustriert, wie man zu einer entsprechenden Problembeschreibung gelangt. Der Arbeitsraum wird als eindimensional betrachtet, es wird nur der Fortschritt der beteiligten Fahrzeuge in Längsrichtung betrachtet. In den Abbildungen 5.14a-d zeigt diese Raumachse  $x$  jeweils nach oben. In Abbildung 5.14a sind die geometrischen Verhältnisse dargestellt. Das eigene Fahrzeug befindet sich zur Zeit  $t_0 = 0$  am Ort  $x_0$ , im rechten Arm einer Y-förmigen Kreuzung. Ohne Beschränkung der Allgemeinheit wird es als punktförmig angenommen. Zwei weitere Fahrzeuge (rot und blau) befinden sich im linken Arm. Für alle Orte unterhalb des Kreuzungspunktes  $x_{\text{krit}}$  kann in den parallelen Kreuzungsarmen nebeneinander her gefahren werden. Erst über  $x_{\text{krit}}$  ist eine Kollisi-





**Abbildung 5.13:** Beispiele für QP-Trajektorien. (a): Zeitliche Diskretisierung und Randbedingungen des Problems. (b)-(e): Untereinander befindet sich jeweils der Verlauf von Position (blau), Geschwindigkeit (grün) und Beschleunigung (gelb) über der Zeit. (b): Unrestringiertes Problem. (c): Position limitiert (rot). (d): Geschwindigkeit limitiert (rot). (e): Position mit zeitvarianter Limitierung (rot).



**Abbildung 5.14:** Optimale Längstrajektorien für einen Einfädelvorgang lassen sich mit quadratischem Programmieren bestimmen. Erläuterungen im Fließtext.

sion möglich. Es wird angenommen, dass vollständige Information über die zukünftigen Trajektorien des roten und blauen Fahrzeuges vorhanden ist. In Abbildung 5.14b sind entsprechend in der Raum-Zeit Ebene die Bereiche gekennzeichnet, die von den beiden Fahrzeugen überstrichen werden. Der Bereich unterhalb von  $x_{\text{krit}}$  ist schattiert, da er ja für die Kollisionsvermeidung irrelevant ist. In Abbildung 5.14c wurden die Bereiche erweitert, und zwar basierend auf der Annahme, dass ein Einfädeln *zwischen* den beiden Fahrzeugen stattfinden soll. Abbildung 5.14d zeigt schließlich die obere und untere Grenze für  $x$ , die dann nach Diskretisierung wie in 5.51 als Nebenbedingung für das Trajektorienplanungsproblem verwendet werden können. Im Kostenfunktional setzt man in diesem Anwendungsfall  $w_{\text{ablage}}$  wieder zu 0, und für  $v_{\text{ref}}$  wählt man die zu fahrende Wunschgeschwindigkeit. Zweckmäßigerweise wählt man eine Randbedingung zweiter Ordnung nur am Anfang der Trajektorie und lässt sie am Ende frei (vergleiche Abschnitt 5.4.2).

Auch unter einigen Verallgemeinerungen bleibt die Eigenschaft eines quadratischen Programmes erhalten, beispielsweise

- beim Übergang in einen zweidimensionalen Arbeitsraum. Es ist aber zu beachten, dass Positionsbeschränkungen wie oben durch die Angabe von Intervallen (*box constraints*), wie in 5.51, ausgedrückt werden, allerdings für beide Komponenten separat. Das bedeutet anschaulich, dass für jeden Stützpunkt der Trajektorie ein achsenparalleles Rechteck angegeben werden kann, *innerhalb* dessen dieser bei der Optimierung bleiben muss. Eine sinnvolle Beschreibung einer Umgebung mit Hindernissen lässt sich im Allgemeinen so nicht erzielen. Man

kann auf diese Weise aber eine grob vorgeplante Trajektorie verfeinern. Quadratisches Programmieren wurde auf diese Weise verwendet, um nachträglich die Diskretisierungseffekte abzuschwächen, die sich bei den globalen, kombinatorischen Verfahren zwangsweise ergeben. Hierüber wird später in Abschnitt 7.2.5 berichtet.

- bei Erhöhung der Ordnung der im Kostenfunktional enthaltenen Ableitungen, zum Beispiel bis zum Ruck. Dann können als Randbedingungen auch Beschleunigungen vorgegeben werden. Das ermöglicht es, krümmungsstetige Trajektorien zu planen.
- können die Vorgaben für  $v_{\text{ref}}$  und  $x_{\text{ref}}$  – ähnlich wie  $x_{\text{min}}$ ,  $x_{\text{max}}$  in (5.51) – zeitvariant sein, man kann also zum Beispiel eine Referenztrajektorie vorgeben, an die sich die Optimaltrajektorie dann anschmiegt.

Im nächsten Abschnitt wird gezeigt, wie man die Methodik des quadratischen Programmierens anpassen kann, um noch allgemeinere Optimierungsprobleme zu lösen.

### 5.5.3 Sequenzielles quadratisches Programmieren (SQP)

Sequenzielles quadratisches Programmieren ist ein Verfahren, um nichtlineare Optimierungsprobleme unter nichtlinearen Nebenbedingungen zu lösen. Bei SQP [Bar72; NW06] wird das Problem in jedem Iterationsschritt durch ein quadratisches Programm angenähert, das dann, wie im letzten Abschnitt beschrieben, gelöst wird. Man findet so eine Abstiegsrichtung, und damit einen verbesserten Initialwert für die nächste Iteration.

Zielfunktionen und Nebenbedingungen werden jetzt über allgemeine Funktionen  $f$  und  $g$  definiert, zu minimieren sei

$$f : \mathbb{R}^n \rightarrow \mathbb{R}, \mathbf{x} \mapsto f(\mathbf{x}) \quad (5.52)$$

über  $\mathbf{x} \in \mathbb{R}^n$ , so dass die  $m$  Nebenbedingungen

$$g(\mathbf{x}) \geq \mathbf{0}, \quad (5.53)$$

erfüllt sind, wobei

$$g : \mathbb{R}^n \rightarrow \mathbb{R}^m, \mathbf{x} \mapsto g(\mathbf{x}). \quad (5.54)$$

Wie im letzten Abschnitt werden auch hier wieder nur Ungleichungsnebenbedingungen betrachtet. Die Funktionen  $f$  und  $g$  müssen

wieder einige Stetigkeitsbedingungen erfüllen, die im Folgenden vorausgesetzt werden.

Sei  $\mathbf{x}_0$  ein Initialwert für die Suche nach dem Optimum, und  $\mathbf{x}_k$  die beste gefundene Näherung nach  $k$  Iterationen. Der Vektor  $\lambda_k$  ist die Schätzung der Lagrange-Multiplikatoren nach  $k$  Iterationen. Für die  $k + 1$ -te Iteration wird das nichtlineare Programm am Punkte  $\mathbf{x}_k$  durch ein quadratisches Programm angenähert, man minimiert also über der optimalen Suchrichtung  $\mathbf{d}_k$

$$f(\mathbf{x}_k) + \nabla_{\mathbf{x}} f(\mathbf{x}_k)^\top \mathbf{d}_k + \frac{1}{2} \mathbf{d}_k^\top \nabla_{\mathbf{x}, \mathbf{x}}^2 \Lambda(\mathbf{x}_k, \lambda_k) \quad (5.55)$$

so dass

$$g(\mathbf{x}_k) + \nabla_{\mathbf{x}} g(\mathbf{x}_k)^\top \mathbf{d}_k \geq \mathbf{0}. \quad (5.56)$$

$\mathbf{x}_{k+1}$  kann man dann theoretisch über einen vollen NEWTON-Schritt gewinnen, also

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}, \quad (5.57)$$

aus Robustheitsgründen führt man aber meistens nur einen beschränkten NEWTON-Schritt

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{d} \quad (5.58)$$

aus, wobei die Schrittweite  $\alpha < 1$  durch eine Liniensuche entlang der Abstiegsrichtung bestimmt wird [Arm66; NW06].

Man beachte, dass in der quadratischen Näherung (5.55) die HESSE-Matrix  $\nabla_{\mathbf{x}, \mathbf{x}}^2 \Lambda(\mathbf{x}_k, \lambda_k)$  der LAGRANGE-Funktion verwendet wird, und nicht die der Zielfunktion. Die HESSE-Matrix der LAGRANGE-Funktion enthält nämlich die Krümmung der Nebenbedingungen, die nach der letzten Iteration aktiv waren [Bar72; NW06].

Dass Verfahren SQP wurde für den Trajektorienplaner verwendet, der für die automatische Befahrung der BERTHA-BENZ-Route 2013 verwendet wurde. In Abschnitt wird 7.1 über dieses Experiment berichtet. Dabei wird auch detailliert auf das dort verwendeten Kostenfunktional und die Bildung der Nebenbedingungsfunktionen eingegangen.

## 6 Globale, diskrete Lösungsverfahren

Mit einem lokalen, kontinuierlichen Verfahren lassen sich zufriedenstellende Ergebnisse insbesondere im Anwendungsfall des Fahrens in fließendem Verkehr erreichen. Der Lösungsraum für die Trajektorienplanung ist hier insbesondere durch die Nebenbedingungen des Fahrkorridors von vornherein so stark eingeschränkt, dass in der Regel die lokal optimale Trajektorie auch dem globalen Optimum entspricht. Kombinatorische Aspekte können ad hoc und vor der eigentlichen lokalen Optimierung behandelt werden.

Beim Navigieren auf einer Parkfläche – oder beim ganz allgemein formulierten Trajektorienplanungsproblem schlechthin – fallen die oben genannten Zusatzbedingungen weg. Deshalb werden in diesem Abschnitt einige globale Verfahren zur Trajektorienplanung vorgestellt, die prinzipiell auch allgemeinere Probleme lösen können. Aus praktischen Gründen sind aber auch diese Verfahren auf bestimmte Anwendungsszenarien limitiert. Auf die Grenzen ihrer Verallgemeinerbarkeit wird in Abschnitt 6.4 eingegangen.

Bei den hier vorgestellten globalen Verfahren wird das Problem auf das Standardproblem kürzester Pfade in einem kantengewichteten Graphen reduziert. Das Kernproblem ist dabei, das Kontinuum des Phasenraumes durch eine endliche Menge an Graphknoten zu approximieren.

### 6.1 Gitter aus Grundgeometrien

Bei einem Gitter aus Grundgeometrien wird die Grundmenge möglicher Trajektorien auf eine endliche Untermenge beschränkt, und zwar derart, dass alle Trajektorien eine diskrete, gitterartige Untermenge des Phasenraumes interpolieren. Als Dimensionen des Phasenraumes können zum Beispiel die Position  $\mathbf{x} = (x, y)^\top$ , die Orientierung  $\theta$  und die Krümmung  $\kappa$  der Trajektorie gewählt werden. Es wurde ja in Abschnitt 2.2 gezeigt, dass diese differentialgeometrischen Größen direkt mit Fahrzeugzuständen und -eingängen zusammenhängen.

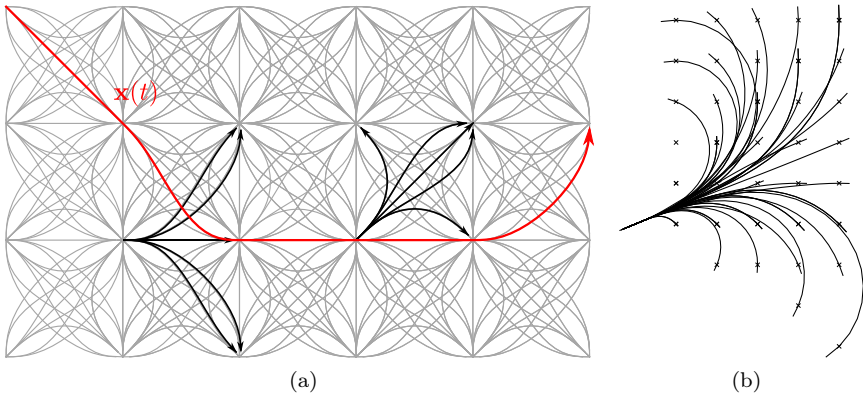


Abbildung 6.1

Ein Gitter entsteht nun, indem die Komponenten des Phasenraumes äquidistant abgetastet werden. *Zwischen* zwei Gitterpunkten besteht die Trajektorie aus einem vorberechneten, optimalen Unterabschnitt. Durch die regelmäßige Gitterstruktur werden nur wenige Grundgeometrien benötigt, um alle Gitterpunkte zu verbinden. In Abbildung 6.1a ist ein einfaches Beispielgitter dargestellt. Der Übersichtlichkeit halber enthält der Konfigurationsraum dieses Gitters keine Krümmungen, die Krümmungen an den Gitterpunkten sind also „frei“, die Trajektorien deshalb nicht krümmungsstetig. Der Winkel  $\theta$  ist auf 8 mögliche Richtungen abgetastet. Für zwei konkrete Konfigurationen sind die ausgehenden Grundgeometrien in schwarz hervorgehoben. Eine aus den Grundgeometrien zusammengesetzte Beispieltrajektorie  $x(t)$  ist rot dargestellt. Gitterpunkte sind nur mit einer kleinen Menge benachbarter Punkte verbunden. Abbildung 6.1b zeigt ein Beispiel für Grundgeometrien aus einem komplexeren Gitter, bei dem auch die Krümmung im Konfigurationsraum enthalten ist. Es sind die Nachfolger einer einzelnen Konfiguration dargestellt.

Zur Erstellung jeder einzelnen Grundgeometrie muss ein lokales Trajektorienplanungsproblem mit Randbedingungen am Anfang und am Ende gelöst werden. Dies kann zum Beispiel mit einer der in Abschnitt 4.2 oder in Kapitel 5 vorgestellten Methoden geschehen. Die Grundgeometrien müssen dabei nur die inneren Nebenbedingungen erfüllen. Äußere Nebenbedingungen werden erst bei der Kombination der Grundgeometrien zu einer Gesamtrajektorie berücksichtigt.

## 6.2 Infinitesimales Gitter

Die Erzeugung eines geeigneten Gitters für das im letzten Abschnitt beschriebene Verfahren ist schwierig. Viele Parameter müssen vorab gewählt werden, beispielsweise der Grad der Diskretisierung des Phasenraumes und der Grad der Vernetzung zwischen den Knoten. Wie diese Parameter einzustellen sind um das Kontinuum ideal zu approximieren ist nicht unmittelbar einsichtig. Außerdem muss, um die Grundgeometrien zu erstellen, ein lokaler Planer bereit gestellt werden, mit dem zumindest das nicht durch Hindernisse restringierte Problem optimal gelöst werden kann. Hierfür ist kein allgemeingültiges Verfahren bekannt.

In diesem Abschnitt wird ein weiteres Verfahren vorgeschlagen, das den Arbeitsraum auf eine gitterartige, diskrete Menge reduziert. Bei ihm ist die Struktur des diskreten Suchraumes durch die Wahl der zeitlichen und der räumlichen Diskretisierung bereits vollständig festgelegt. Die Methode ist - bis auf das Auflösungslimit der Diskretisierung - optimal und vollständig.

### 6.2.1 Problembeschreibung

Es werden wieder Trajektorien betrachtet, die ein Funktional der Form

$$E[\mathbf{x}(t)] = \int_0^T L(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) dt \quad (6.1)$$

minimieren. Die Zielmenge der Funktion im Integranden sei hierbei  $\mathbb{R}_{\geq 0} \cup \{\infty\}$ . Auf die Behandlung von Funktionen, die Ableitungen höherer als zweiter Ordnung enthalten, wird noch eingegangen werden. Die Lösung soll durch Zwangsbedingungen restringiert sein, die implizit durch die Angabe einer Menge  $I$  unzulässiger Elemente des Phasenraumes definiert werden. Man kann einfach erreichen, dass ein Minimierer des Funktionals (6.1) die Zwangsbedingungen erfüllt, indem man den Integranden  $L$  so definiert, dass gilt

$$(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) \in I \Rightarrow L(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) = \infty. \quad (6.2)$$

Der erste Diskretisierungsschritt verläuft genau wie in Abschnitt 5.1, das Funktional  $E$  wird also durch die Funktion

$$E_d(x_0, x_1, \dots, x_{n-1}) = \sum_{i=1}^{n-2} L(t_i, x_i, \frac{\Delta x_i}{2h}, \frac{\Delta^2 x_i}{h^2}) h, \quad (6.3)$$

angenähert, wobei die konstanten Stützstellen  $\{x_1, x_2, \dots, x_N\}$  durch zeitdiskrete Abtastung mit Schrittweite  $h$  generiert wurden.  $\Delta_c$  und  $\Delta_c^2$  sind die zentralen finiten Differenzen wie in (2.7). Der Übersichtlichkeit halber wird die Funktion

$$L_d(x_{i-1}, x_i, x_{i+1}) = L\left(t_i, x_i, \frac{\Delta x_i}{2h}, \frac{\Delta^2 x_i}{h^2}\right)h \quad (6.4)$$

eingeführt, dann gilt

$$E_d(x_0, x_1, \dots, x_{n-1}) = \sum_{i=1}^{n-2} L_d(x_{i-1}, x_i, x_{i+1}). \quad (6.5)$$

Sei jetzt  $G$  ein geometrischer Digraph, der in  $\mathbb{R}^2$  eingebettet ist. Sei  $V(G)$  die Menge seiner Knoten, und  $\mathcal{E}(G) \subseteq V(G) \times V(G)$  die seiner Kanten. Die Einbettung der Knoten in  $\mathbb{R}^2$  sei durch die Funktion

$$\mathbf{x} : V(G) \rightarrow \mathbb{R}^2 \quad (6.6)$$

definiert. Für einen Pfad  $P$  in  $G$ , der aus  $n$  benachbarten Knoten  $(v_1, v_2, \dots, v_n)$  besteht, bezeichne  $\mathbf{x}(P) = (\mathbf{x}(v_1), \mathbf{x}(v_2), \dots, \mathbf{x}(v_n))$  seine geometrische Repräsentation. Bei gegebenen Start- und Zielknoten  $v_{\text{start}}$  und  $v_{\text{end}}$  wird nun  $n^*$  und ein Pfad  $P^* = (v_1^*, v_2^*, \dots, v_{n^*}^*)$  mit  $v_1^* = v_{\text{start}}$  und  $v_{n^*}^* = v_{\text{end}}$  gesucht, sodass  $E_d(\mathbf{x}(P^*))$  minimal wird. Der Pfad  $P^*$  wird im Folgenden der *kürzeste Pfad zweiter Ordnung* (bezüglich der Kostenfunktion  $L$ ) genannt. Die Aufgabe, einen solchen Pfad zu berechnen wird im folgenden als 2SPP (*second order shortest path problem*) bezeichnet.

### 6.2.2 Ableitung eines Graphen

Das Problem kürzester Pfade zweiter Ordnung (2SPP), das im letzten Abschnitt formalisiert wurde, ist mit dem klassischen Problem kürzester Pfade (*shortest path problem*, SPP) verwandt, kann aber nicht direkt mit den bekannten Methoden (zum Beispiel nach DIJKSTRA) gelöst werden. Dies liegt daran, dass ja drei benachbarte Knoten eines Pfades  $P$  benötigt werden, um ein Glied der Summe  $E_d(x(P))$  auswerten zu können. Beim SPP werden Kosten jedoch nur mit den Kanten assoziiert, also mit einem Paar aus genau zwei Knoten. In diesem Abschnitt wird die *Ableitung eines Graphen* eingeführt. Mit Hilfe dieser Transformation kann das 2SPP auf das SPP im transformierten Graphen zurückgeführt werden.



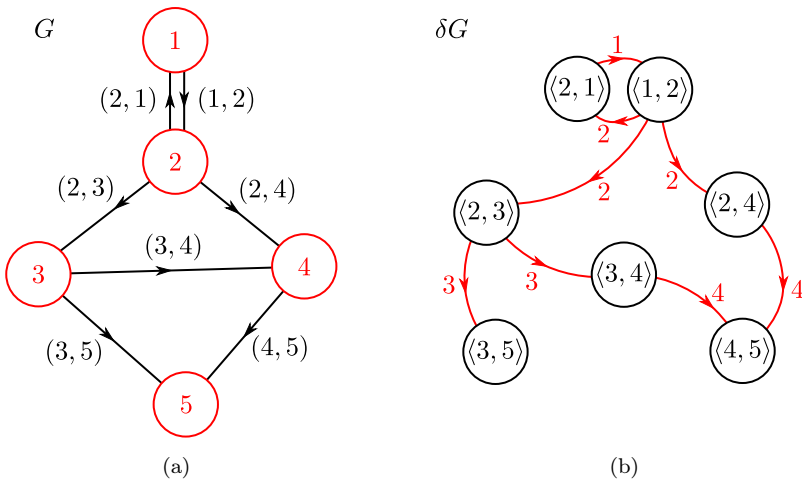


Abbildung 6.2: Ableitung eines Graphen.

Das Konzept einer Ableitung von Graphen wurde in verschiedenen Kontexten und unter verschiedenen Bezeichnungen mehrfach wiederentdeckt. Allgemein am verbreitetsten ist die Bezeichnung *Kantengraph* (engl. *Line-graph*, Aigner [Aig67]) für den abgeleiteten Graphen. Er wurde aber auch schon als *adjungierter Graph* Menon [Men65] oder *Kanten-Knoten-Dualität* [SR61] bezeichnet. Für diese Arbeit wurde die Bezeichnung „Ableitung eines Graphen“ nach [Bei70] übernommen, da die Transformation hier in genau dieser Weise interpretiert wird: Die Knoten eines geometrischen Graphen entsprechen Orten, die Knoten seiner Ableitung entsprechen Geschwindigkeitsvektoren.

Zur Definition der Ableitung eines Graphen  $G$  werden zunächst die beiden Funktionen  $\text{in}_G, \text{out}_G : V(G) \rightarrow \mathcal{P}(V(G))$  eingeführt, die jeden Knoten auf ihre ein- beziehungsweise ausgehenden Nachbarn abbilden. Die Knoten des abgeleiteten Graphen  $\delta G$  sind die Kanten von  $G$ ,  $V(\delta G) = \mathcal{E}(G)$ . Die *Kanten* des Graphen  $G$  und die *Knoten* des Graphen  $\delta G$  entstammen also der selben Grundmenge. Um Verwechslung zu vermeiden, wird deshalb im Folgenden die Notation  $(u, v)$  für Elemente aus  $\mathcal{E}(G)$ , und die Notation  $\langle u, v \rangle$  für Elemente aus  $V(\delta G)$  verwendet. Die Kanten von  $\delta G$  werden nun gemäß folgender Regel gebildet: Zwei Knoten  $x$  und  $y$  in  $\delta G$  sind genau dann mit einer Kante  $(x, y)$  verbunden, wenn gilt  $x = \langle r, s \rangle$  und  $y = \langle s, t \rangle$ , für beliebige Knoten  $r, s, t \in V(G)$ . Abbildung 6.2 zeigt ein Beispiel für die

Bildung des abgeleiteten Graphen. Für die Mächtigkeit der Knoten- und Kantenmengen folgt *per constructionem*:

$$|V(\delta G)| = |\mathcal{E}(G)| \quad (6.7)$$

und

$$|\mathcal{E}(\delta G)| = \sum_{v \in V(G)} \text{in}_G(v) \cdot \text{on}_G(v). \quad (6.8)$$

Wenn  $G$  ein geometrischer Graph ist, also eine Einbettung  $\mathbf{x}$  wie in (6.6) existiert, und die Pfade in  $G$ , wie in Abschnitt 6.2.1, einer Interpretation als finite Differenzen unterworfen werden, so können die Kanten im Ausgangsgraphen, und entsprechend die Knoten des abgeleiteten Graphen, als Geschwindigkeitsvektoren interpretiert werden.

Bezeichne im folgenden  $\Pi(G)$  die Menge aller Pfade im Graphen  $G$ . Aus der Konstruktion von  $\delta G$  folgt, dass jeder Pfad  $P'$  der Länge  $n$  aus  $\Pi(\delta G)$  die Form  $(\langle v_0, v_1 \rangle, \langle v_1, v_2 \rangle, \langle v_2, v_3 \rangle, \dots, \langle v_{n-1}, v_n \rangle)$  aufweist. Es folgt weiterhin, dass

$$\begin{aligned} (\langle v_0, v_1 \rangle, \langle v_1, v_2 \rangle, \langle v_2, v_3 \rangle, \dots, \langle v_{n-1}, v_n \rangle) \in \Pi(\delta G) \\ \iff \\ (v_0, v_1, v_2, v_3, \dots, v_{n-1}, v_n) \in \Pi(G). \end{aligned} \quad (6.9)$$

Es wird die Bijektion  $c : \Pi(\delta G) \leftrightarrow \Pi(G)$  eingeführt, die die Kontraktion durchführt, die von der Äquivalenzrelation (6.9) impliziert wird. Wenn jetzt jede Kante  $(\langle u, v \rangle, \langle v, w \rangle) \in \mathcal{E}(\delta G)$  mit den Kosten  $L_d(\mathbf{x}(u), \mathbf{x}(v), \mathbf{x}(w), h)$  aus Gleichung (6.4) annotiert wird, so ergeben sich die aufsummierten Gesamtkosten für einen Pfad  $P'$  zu  $\sum_k L_d(\mathbf{x}(v_{k-1}), \mathbf{x}(v_k), \mathbf{x}(v_{k+1}), h)$ , was genau den Kosten  $E_d(\mathbf{x}(c(P')))$  entspricht. Es gilt also: Ist  $P'$  ein kürzester Pfad in  $\delta G$ , dann ist  $c(P')$  ein kürzester Pfad zweiter Ordnung im Ausgangsgraphen  $G$ .

### Ableitung eines Gittergraphen

Um Illustration und Notation zu vereinfachen, wird im folgenden, ohne Beschränkung der Allgemeinheit, eine zeitliche Diskretisierungsstufe von  $\Delta t = 1$  angenommen. Als Beispiel wird die Kostenfunktion

$$L(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) = \begin{cases} p(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) & (\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) \notin I \\ \infty & \text{sonst.} \end{cases} \quad (6.10)$$

mit

$$p(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) = 1 + b \|\ddot{\mathbf{x}}\|^2 \quad (6.11)$$

gewählt. Die Funktion  $p$  bewirkt durch den konstanten Term, dass die Fahrzeit minimiert wird. Da dies aber im Allgemeinen zum Auftreten starker Beschleunigungen führt, wird der gewichtete Regularisierungsterm  $\|\ddot{\mathbf{x}}\|^2$  dazuaddiert. Die Menge  $I$  beschreibt alle Restriktionen des Systems. Man wählt

$$I = I_{\text{accel}} \cup I_{\text{vel}} \cup I_{\text{curv}} \cup I_{\text{obs}}, \quad (6.12)$$

mit

$$\begin{aligned} I_{\text{accel}} &= \{(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) : \|\ddot{\mathbf{x}}\| > a_{\text{max}}\}, \\ I_{\text{vel}} &= \{(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) : \|\dot{\mathbf{x}}\| > v_{\text{max}}\}, \\ I_{\text{curv}} &= \{(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) : \|\kappa\| > \kappa_{\text{max}}\}, \\ I_{\text{obs}} &= \{(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) : (x, y, \theta) \in O(t)\}. \end{aligned} \quad (6.13)$$

$I_{\text{accel}}$ ,  $I_{\text{vel}}$  und  $I_{\text{curv}}$  enthalten Zustände, die Grenzwerte bezüglich Beschleunigung, Geschwindigkeit und Krümmung überschreiten.  $I_{\text{obs}}$  sind Restriktionen, die sich durch Überschneidung mit Hindernissen ergeben. Die Menge  $I_{\text{obs}}$  ist gleich der Menge der Kollisionsraumhindernisse, die später in Abschnitt 7.2.3 beschrieben werden. Um später Rechenzeit und Speicherplatz zu sparen, wird beim Aufbau des abgeleiteten Graphen folgende unmittelbar einsichtige Tatsache ausgenutzt

**Lemma 6.1.** *Entfernt man aus einem Graphen alle Kanten mit unendlichen Kosten, so ändert dies an der Lösung eines SSP nichts.*

Als generierender Graph wird ein kartesisches Gitter mit Schrittweite  $\Delta s$  verwendet. Jeder Knoten kann durch ein Paar ganzer Zahlen beschrieben werden,  $V(G) \subset \mathbb{Z} \times \mathbb{Z}$ . Über diesen Index wird auch die geometrische Einbettung  $\mathbf{x}$  definiert, für einen Knoten  $(i, j) \in V(G)$  gilt  $\mathbf{x}((i, j)) = (i\Delta s \ j\Delta s)^\top$ . Jeder Knoten ist mit mehreren Knoten in seiner Nachbarschaft verbunden, wobei man die Größe der Nachbarschaft durch  $v_{\text{max}}$  einschränken kann, dies folgt aus Lemma 6.1. In Abbildung 6.3a wird die Nachbarschaftsbeziehung exemplarisch für einen Knoten gezeigt. Um die Bildung der Ableitung des Graphen zu illustrieren, betrachten wir, der Übersicht halber, nur einen Teilgraphen  $S \subset G$ , der in Abbildung 6.3b dargestellt ist. Die Knoten sind, aus Platzgründen, nur mit Einzelziffern indiziert.

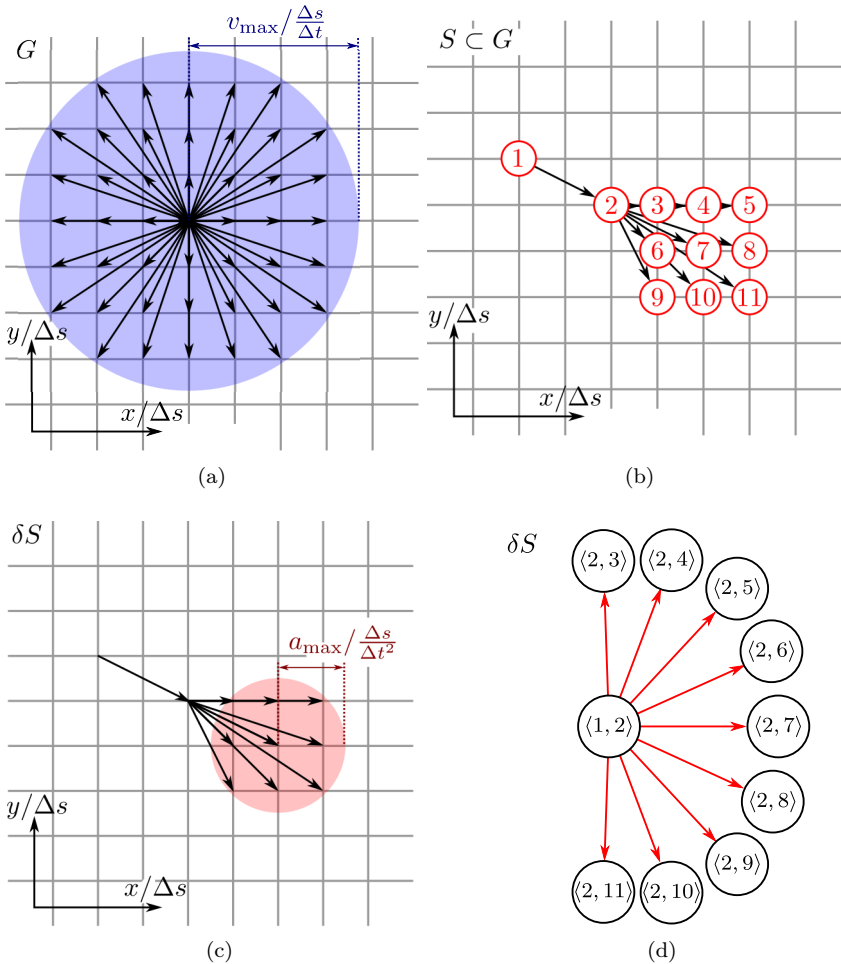


Abbildung 6.3: Gitter mit finiten Differenzen

Bei der Bildung des abgeleiteten Graphen kann man wieder einige Kanten weglassen, die zu unendlich hohen Kosten führen würden. Limitierend ist hierbei im wesentlichen die Maximalbeschleunigung  $a_{\max}$ , ein Knoten des abgeleiteten Graphen entspricht ja einem Geschwindigkeitsvektor, der Übergang zwischen zwei Knoten also einer Beschleunigung. Dieser Zusammenhang ist in Abbildung 6.3c veranschaulicht. Auch die Restriktion der Krümmung durch (6.13) kann schon bei der Bildung des Graphen berücksichtigt werden. Abbildung 6.3d zeigt die Topologie der Ableitung des Teilgraphen  $S$  aus Abbildung 6.3b.

Abbildung 6.4 illustriert den Zusammenhang eines Pfades im Gittergraphen mit der ursprünglichen variationalen Formulierung. Im rechten oberen Quadranten dieses Kreuzplots ist die Trajektorie  $\mathbf{x}(t)$  kartesisch in der  $x, y$ -Ebene als Pfad in einem Gitter dargestellt. Die vorgegebene Randwerte  $\mathbf{x}_{\text{start}}, \dot{\mathbf{x}}_{\text{start}}, \mathbf{x}_{\text{ende}}$  und  $\dot{\mathbf{x}}_{\text{ende}}$  sind blau dargestellt. Darunter und links davon sind die einzelnen Komponenten  $x(t)$  und  $y(t)$  über der Zeit abgetragen. Diese Plots der Einzelkomponenten können jetzt genauso interpretiert werden wie Abbildung 5.1. Man sieht aber, dass, außer den Zeiten, auch die Positionen der Vergleichskurven auf diskrete Werte beschränkt sind. Legt man einen endlichen Arbeitsraum zugrunde, so ist auch die Menge der Vergleichskurven endlich. Durch eine 2SSP-Suche wird die optimale Kurve, die die Randwerte miteinander verbindet, aus dieser Menge identifiziert.

Wie man sieht, wird, anders als bei der kontinuierlichen Formulierung, *kein* explizites Zeitintervall  $[t_{\text{start}}, t_{\text{ende}}]$  für die Trajektorie vorgegeben. In der kombinatorischen Formulierung können also auch transversale Probleme – also solche, bei denen die Reisezeit „mit optimiert“ wird – spezifiziert und gelöst werden.

Im folgenden Abschnitt wird eine besonders effiziente Methode beschrieben, um das vorliegende Problem zu lösen. Wie die meisten Pfadsuchalgorithmen löst es das Problem „1 zu  $n$ “, das heißt vom Startzustand wird die Optimaltrajektorie zu *allen* Zuständen im Arbeitsraum gefunden. Das Problem kann auch invers behandelt werden, es wird dann von *allen* Startzuständen die Optimaltrajektorie zu *einem* Zielzustand berechnet. Die so gefundene Funktion wird auch als *feedback-plan* bezeichnet LaValle [LaV06].

## 6.3 Schnelle parallele Graphsuche

Das Standardverfahren, um einen minimalen Spannbaum in einem kantengewichteten Graphen zu berechnen ist DIJKSTRAS Algorithmus. Man

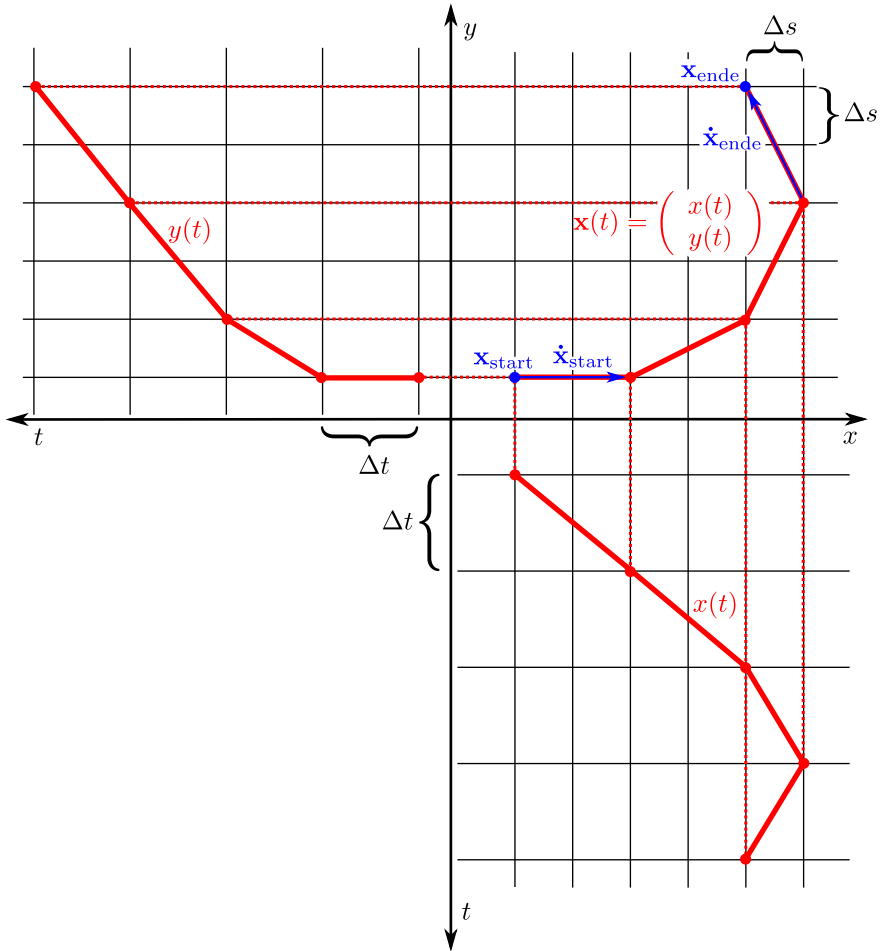


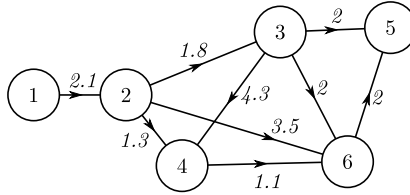
Abbildung 6.4: Diskretisierung mit Gittergraph.

kann zeigen, dass dieser für allgemeine Graphen optimal effizient ist [Vol96]. Die mit den in den letzten Abschnitten gezeigten Techniken konstruierten Gittergraphen haben aber spezielle Eigenschaften. Diese kann man sich zu Nutze machen, um einen noch effizienteren Algorithmus zu entwerfen, der sich darüber hinaus gut für den Einsatz auf paralleler Hardware anpassen lässt. In [Tsi95] wurde eine Variante von Dials [Dia69] SSSP-Algorithmus vorgeschlagen, die lineare Laufzeit in Bezug auf die Anzahl der Kanten aufweist. Sie ist anwendbar, wenn alle Kantengewichte größer oder gleich 1 sind. Dies kann – sofern alle Kantengewichte größer 0 sind – stets durch Skalieren der Gewichte erreicht werden.

Sowohl Dials Algorithmus als auch dessen Variante nach Tsitsiklis ersetzen die Prioritätswarteschlange aus Dijkstras Algorithmus' mit einem sequenziellen Puffer aus Behältern. Zusätzlich zu diesem Puffer („**buckets**“) wird ein Puffer („**costs**“) zum Speichern der geringsten Kosten, mit denen bisher ein Knoten erreicht werden konnte, vorgehalten. Der Einfachheit halber können wir uns diese Puffer zunächst als unendlich groß vorstellen. In der Praxis können sie als endliche Ringpuffer implementiert werden, für deren Größe es eine feste obere Schranke gibt. Eine Invariante, die während des Ablaufes des Algorithmus stets zugesichert wird, ist, dass der Behälter mit dem Index  $k$  alle Knoten enthält, für die der bisher gefundene kürzeste Pfad Kosten von  $c$  hat, mit  $k \leq c < k + 1$ . Für das Verständnis des Algorithmus beachte man, dass **costs** durch einen Knoten-Deskriptor indiziert wird. Dahingegen wird **buckets** durch Pfadkosten indiziert, die jeweils auf die nächste Ganzzahl abgerundet werden.

Der Algorithmus wird initialisiert, indem der Startknoten in Behälter 0 platziert und seine Kosten zu 0 gesetzt werden. Anschließend werden die Knoten in **buckets** von links nach rechts expandiert. Da alle Kantenkosten größer als 1 sind, landen neu besuchte Knoten immer in Behältern, die sich rechts vom gerade expandierten Behälter befinden. Der Algorithmus terminiert, wenn sich kein Knoten mehr in irgendeinem Behälter befindet. Abbildung 6.5 zeigt die ersten Iterationen des Algorithmus, der auf einen Beispielgraphen angewendet wird.

Programmtext 4.1 zeigt funktionierenden Python-Code für den kürzester-Pfad-Algorithmus nach Tsitsiklis. Man beachte, dass, aus Gründen der Übersichtlichkeit, diese spezifische Implementierung nicht optimal effizient ist. Der Grund dafür ist, dass die Puffer und Behälter als einfach verkettete Listen implementiert sind, so dass das Entfernen eines Elementes – wie in Zeile 22 – lineare Zeit bezüglich der Länge der Liste benötigt. In der Praxis erhält jeder Knoten einen Zeiger, der zurück auf das Listenelement des Be-



costs:

1	2	3	4	5	6
0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

buckets:

0	1	2	3	4	5	6	...
---	---	---	---	---	---	---	-----

↓  
①

1	2	3	4	5	6
0	2.1	$\infty$	$\infty$	$\infty$	$\infty$

0	1	2	3	4	5	6	...
---	---	---	---	---	---	---	-----

↓  
②

1	2	3	4	5	6
0	2.1	3.9	3.4	$\infty$	5.6

0	1	2	3	4	5	6	...
---	---	---	---	---	---	---	-----

↓  
③      ↓  
④      ⑥

1	2	3	4	5	6
0	2.1	3.9	3.4	5.9	5.6

0	1	2	3	4	5	6	...
---	---	---	---	---	---	---	-----

↓  
④      ↓  
⑤      ↓  
⑥

1	2	3	4	5	6
0	2.1	3.9	3.4	5.9	4.5

0	1	2	3	4	5	6	...
---	---	---	---	---	---	---	-----

↓  
⑥      ↓  
⑤

Abbildung 6.5: Tsitsiklis Algorithmus



---

```

1 from collections import defaultdict
2 from math import trunc
3 infnty = Float( "infinty" )
4
5 def tsitsiklis( G, start ):
6     buckets      = defaultdict( lambda : [] )
7     costs        = defaultdict( lambda : infnty )
8     buckets[0]   = [start]
9     costs[start] = 0
10    bucket_idx   = 0
11    while len( buckets ) > 0:
12        for current in buckets[bucket_idx]:
13            for edge_cost, succ in G[current]:
14                new_cost = costs[current] + edge_cost
15                old_cost = costs[succ]
16                if new_cost < old_cost:
17                    costs[succ] = new_cost
18                    new_idx = trunc( new_cost )
19                    old_idx = trunc( old_cost )
20                    if new_idx < old_idx:
21                        if old_idx != infnty:
22                            buckets[old_idx].remove( succ )
23                            buckets[new_idx].append( succ )
24
25        del buckets[bucket_idx]
26        bucket_idx += 1
27
28    print costs

```

---

Programmtext 6.1: Tsitsiklis' single source shortest path algorithm in Python.

hälters in `buckets` zeigt, in dem er zur Zeit abgelegt ist. Auf diese Weise kann jeder Knoten in konstanter Zeit aus seinem Behälter entfernt werden.

Dieser Algorithmus zeigt in der Praxis besonders gutes Laufzeitverhalten, wenn die Kosten der teuersten Kante im Graph klein sind im Vergleich zu seinem Durchmesser. Der Durchmesser ist die Länge des längsten aller kürzesten Pfade. Diese Eigenschaft trifft auf die Gittergraphen, die in den vorangehenden Abschnitten beschrieben wurden, zu. Die Konnektivität eines Knotens beschränkt sich ja immer nur auf eine kleine räumliche Nachbarschaft, die klein ist im Verhältnis zum Gesamtgraphen.

Die Expansion eines Knotens kann sich nicht auf Knoten im gleichen Behälter auswirken. Deshalb können die Knoten eines Behälters parallel expandiert werden. Dies bedeutet, dass die `for`-Schleife in Zeile 12 des Pro-

rammtexes parallel ausgeführt wird. Auf diese Weise lässt sich auf parallelen Prozessorarchitekturen eine Beschleunigung der Graphsuche erreichen.

## 6.4 Weitere Betrachtungen

In diesem Abschnitt wird zunächst die asymptotische Komplexität des Verfahrens hergeleitet. Dann wird auf zwei Themen eingegangen, die die Allgemeinheit der vorgeschlagenen Methode betreffen. Zuerst wird untersucht, inwiefern Kostenfunktionale behandelt werden können, die höhere (höher als die zweite) Ableitungen enthalten. Dann werden Grenzen betrachtet, die sich aus dem der Planung zugrunde liegenden dynamischen System ergeben.

### 6.4.1 Asymptotische Komplexität

Wie in Abschnitt 6.3 gezeigt wurde, ist die Laufzeit der erschöpfenden Graphsuche linear in der Anzahl Knoten  $n$  im Graphen. Diese Anzahl wird jetzt zu folgenden Kennzahlen in Beziehung gesetzt:

- $r$ : Radius des Arbeitsraumes.
- $h(= \Delta t)$ : Dauer eines diskreten Zeitschrittes.
- $d(= \Delta s)$ : räumliche Schrittweite des Gitters.
- $v_{\max}$ : Maximalgeschwindigkeit des Fahrzeuges.
- $a_{\max}$ : Maximalbeschleunigung des Fahrzeuges.

Die Anzahl der Knoten im (nicht abgeleiteten) Gitter,  $|V(G)|$ , ist in  $\mathcal{O}(r^2 \frac{1}{d^2})$ . Jeder Knoten hat – außer am Rand des Arbeitsraumes, dieser kann aber für die asymptotische Betrachtung vernachlässigt werden –  $k = v_{\max}^2 h^2 \frac{1}{d^2}$  ausgehende Kanten (siehe Abbildung 6.3a), insgesamt ist also  $|\mathcal{E}(G)| \in \mathcal{O}(r^2 v_{\max}^2 h^2 \frac{1}{d^4})$ . Wie im nächsten Abschnitt gezeigt wird, vervielfältigt sich die Anzahl Knoten des Graphen durch Ableiten um den Faktor  $k$ , es gilt also  $|V(G')| \in \mathcal{O}(r^2 v_{\max}^2 h^2 \frac{1}{d^4})$ . Auch die Anzahl der Kanten vervielfältigt sich nach erster Abschätzung um den Faktor  $k$ . Da aber viele der Kanten die Nebenbedingung für die Maximalbeschleunigung verletzen und deshalb nicht in den Graph mit aufgenommen werden (vergleiche Abbildung 6.3c), kann diese Schranke noch etwas geschärft werden, die Anzahl



**Abbildung 6.6:** Die Anzahl der Knoten und Kanten wächst bei einmaliger Ableitung eines Graphen multiplikativ mit dem Ausgangsgrad der Knoten.

Kanten vervielfältigt sich nur um den Faktor  $a_{\max}^2 h^4 \frac{1}{d^2}$  und es gilt

$$|\mathcal{E}(G')| \in \mathcal{O}(r^2 v_{\max}^2 a_{\max}^2 h^6 \frac{1}{d^6}). \quad (6.14)$$

Die innerste Schleife des Suchalgorithmus' aus Abschnitt 6.3 wird einmal pro Kante durchlaufen, die Laufzeit des Verfahrens ist also  $\in \mathcal{O}(|\mathcal{E}(G')|)$ . Gleichung (6.14) erweckt den Eindruck, dass durch (eigentlich ja erstrebenswertes) Verfeinern der zeitlichen Abtastung  $h$  der Aufwand des Verfahrens beliebig gesenkt werden könnte. In Abschnitt 6.4.3 wird gezeigt, wo hierbei die Grenzen liegen.

### 6.4.2 Funktionale, die höhere Ableitungen enthalten

In der vorgestellten Form kann das Verfahren für Kostenfunktionale verwendet werden, die Ableitungen bis zur zweiten Ordnung enthalten. Prinzipiell ließe es sich auf beliebig hohe Ableitungen erweitern, indem das Grundgitter mehrfach abgeleitet wird (Abschnitt 6.2.2). Allerdings steigt die Anzahl der Knoten im Graphen exponentiell mit jeder weiteren Ableitung. Die Anzahl der Knoten im Graphen vervielfältigt sich nämlich bei jeder weiteren Ableitung mit dem Ausgangsgrad der Knoten im Ausgangsgitter, wie jetzt gezeigt wird. In einem Gittergraphen wie in Abschnitt 6.2.2 gilt, dass jeder Knoten einen konstanten Ausgangsgrad  $k$  hat, der mit seinem Eingangsgrad identisch ist. Dies gilt nicht am Rand des Gitters, bei einem ausreichend großen Gitter können Effekte am Rand aber für die folgende Betrachtung vernachlässigt werden. In Abbildung 6.6a ist ein Ausschnitt eines solchen Graphen für  $k = 2$  dargestellt. Abbildung 6.6b zeigt den entsprechenden Ausschnitt des abgeleiteten Graphen. Aus dem einzelnen Knoten  $a$  sind  $k^2 = 4$  Kanten geworden, und aus seinen  $k = 2$  ausgehenden Kanten 3 und 4 wurde jeweils ein Knoten. Diese Beobachtung kann man auf alle Graphen mit konstantem Ausgangsgrad übertragen und stellt fest, dass sich sowohl

die Anzahl der Kanten als auch die der Knoten im Graphen bei einmaliger Ableitung ver- $k$ -facht. Man sieht weiterhin, dass der abgeleitete Graph den gleichen Ein- und Ausgangsgrad  $k$  wie der Ursprungsgraph aufweist. Die Gesamtanzahl Knoten in einem  $j$  mal abgeleiteten Graphen aus  $n$  Knoten mit einem Ausgangsgrad von  $k$  beträgt also  $nk^j$ .

Man kann aus dieser Betrachtung die allgemeine Einschätzung ableiten, dass kombinatorische Verfahren für hochdimensionale Probleme schlecht geeignet sind, insbesondere wenn sie auf einer Diskretisierung des Zustandsraumes beruhen. Die Anzahl der Abtastpunkte des Zustandsraumes wächst offensichtlich exponentiell mit der Anzahl seiner kontinuierlichen Dimensionen. Die zur Beschreibung des Kostenfunktionalen erforderlichen Ableitungen kann man als Systemzustände betrachten (man nehme als Systemmodell eine Integratorkette an).

### 6.4.3 Grenzen

Durch die Konstruktion aus dem Gitter folgt, dass Beschleunigungsvektor, Geschwindigkeitsvektor und Ortsvektor äquidistant diskretisiert werden. Dadurch eignet sich das Verfahren ideal, wenn dem betrachteten System eine Integratorkette (Doppelintegrator) zu Grunde liegt, also für das System

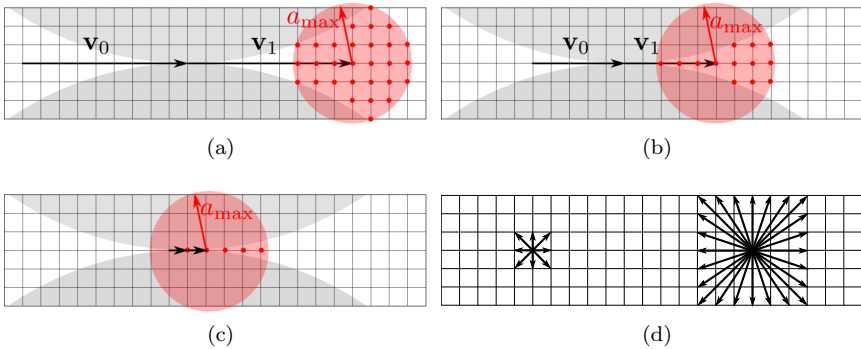
$$\dot{\mathbf{x}} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{b} \quad (6.15)$$

mit

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix}, \mathbf{b} = \begin{pmatrix} a_x \\ a_y \end{pmatrix}. \quad (6.16)$$

Hierbei beschreibt  $x, y$  die Position des Schwerpunktes und die Eingänge  $a_x$  und  $a_y$  seine Beschleunigung. Bei der Wahl der Gitterkonstanten  $\Delta s$  und  $\Delta t$  ist – außer ihrem direkten Einfluss auf die Laufzeit (Abschnitt 6.4.1) – zu beachten, dass die Eingänge und Zustände durch die Diskretisierung sinnvoll abgedeckt werden. Für die Schrittweiten von Geschwindigkeit und Beschleunigung,  $\Delta v$  und  $\Delta a$ , gilt

$$\Delta v = \frac{\Delta s}{\Delta t} \quad (6.17)$$



**Abbildung 6.7:** Innere Nebenbedingungen schränken den Eingangsraum sowie den Raum innerer Zustände ein. Erläuterungen im Fließtext.

und

$$\Delta a = \frac{\Delta s}{(\Delta t)^2}. \quad (6.18)$$

Die Wahl von  $\Delta s$  fällt intuitiv am leichtesten, da es ja einfach die „räumliche“ Diskretisierung des Arbeitsraumes darstellt. Soll „auf zehn Zentimeter genau“ eingeparkt werden können, so sollte  $\Delta s$  höchstens 0.1m betragen. Es empfiehlt sich jetzt,  $\Delta t$  so zu wählen, dass der Eingangsraum des Systems ausreichend viele diskrete Werte enthält – je kleiner  $\Delta t$ , desto *weniger* mächtig wird der Eingangsraum: Man erhält (für die Beschleunigung in *einer* Koordinatenachse)

$$n_a = \left\lfloor \frac{2a_{\max}(\Delta t)^2}{\Delta s} \right\rfloor \quad (6.19)$$

diskrete Beschleunigungen, für  $\Delta s = 0.1\text{m}$ ,  $a_{\max} = 2\frac{\text{m}}{\text{s}^2}$  und  $\Delta t = 0.5\text{s}$  erhält man zum Beispiel  $n_a = 10$ . Da die Beschleunigungen (beim Doppelintegrator) separat angewendet werden können, kann man diesen Wert in etwa quadrieren, um die Gesamtmächtigkeit des diskreten Eingangsraumes für den Doppelintegrator zu erhalten.

Dass eine Anwendung des Verfahrens für das System „Fahrzeug“ überhaupt möglich ist, folgt aus der Flachheit des Fahrzeugmodells. In Abschnitt 2.2 wurde ja gezeigt, dass sich aus der Trajektorie des Hinterachspunktes und ihren ersten beiden Ableitungen eine Trajektorie für den Fahrzeugzustand  $(x, y, \theta, v)$  und die Fahrzeugeingänge  $(\delta, a)$  rekonstruieren lässt. Allerdings folgen aus der Lenkkinematik des Fahrzeuges Beschränkungen, die bei der

Wahl einer geeigneten Diskretisierung des Arbeitsraumes untersucht werden sollten. Die zu betrachtende Beschränkung ergibt sich aus der Limitierung des Vorderradwinkels des Fahrzeuges, aus der eine Limitierung der Krümmung der Trajektorie resultiert. Abbildung 6.7 zeigt exemplarisch, was hierbei zu beachten ist. In Abbildung 6.7a ist ein infinitesimales Gitter dargestellt. Ein beispielhafter Geschwindigkeitsvektor  $\mathbf{v}_0$  ist hervorgehoben. In Rot sind die möglichen Endpunkte eines an  $\mathbf{v}_0$  anschließenden Geschwindigkeitsvektors dargestellt (vergleiche Abschnitt 6.2.2). Der Bereich der erlaubten Beschleunigungen ist rot schattiert. Grau schattiert sind Bereiche, die aufgrund des Krümmungslimits nicht erreicht werden können. Die folgenden Abbildungen 6.7b-c zeigen, dass der Eingangsraum immer schlechter repräsentiert wird, je geringer die Geschwindigkeit der Trajektorie wird. Dies geht so weit, dass bei sehr langsamer Geschwindigkeit wie in Abbildung 6.7c keine Querbeschleunigung mehr möglich ist. Auch die aus den Ableitungen der Hinterachspunktstrajektorie abgeleitete innere Zustandsgröße  $\theta$  – die Ausrichtung des Fahrzeuges – ist bei langsamen Geschwindigkeiten nicht ideal repräsentiert. Bei der langsamsten möglichen Geschwindigkeit können nur noch acht diskrete Richtungen gefahren werden, wie man links in Abbildung 6.7d sieht. In der Praxis machen sich die genannten Probleme nicht bemerkbar. Vom Kostenfunktional her – die Fahrtzeit und damit die Anzahl der diskreten Schritte wird ja mit optimiert – ist der Planer ohnehin bestrebt, wenige lange anstatt vieler kurze Schritte zu machen.

Zusammenfassend kann festgehalten werden, dass sich das Verfahren des infinitesimalen Gitters prinzipiell auf alle Systeme verallgemeinern lässt, die einen zweidimensionalen, flachen Ausgang in der kartesischen Ebene besitzen. Es muss aber im Einzelfall geprüft werden, wie gut sich der diskretisierte flache Ausgang auf die Eingangsgrößen und die inneren Zustände des Systems abbilden lässt. Auch die Verträglichkeit der Diskretisierung mit inneren Nebenbedingungen des Systems muss untersucht werden.

## 7 Experimente

Im Rahmen dieser Arbeit wurden sowohl lokale, kontinuierlichen Verfahren (Kapitel 5) als auch globale, kombinatorische Verfahren (Kapitel 6) für die Trajektorienplanung implementiert. Ein lokales Verfahren wurde für die Bewältigung der BERTHA-BENZ-MEMORIAL-ROUTE [Zie+14c] angewendet. Ein globaler Planer ist das Herzstück eines Demonstrators zur Navigation in beengter Umgebung. Über diese beiden konkreten Implementierungen wird in diesem Kapitel berichtet.

### 7.1 Bertha-Benz-Fahrt 2013

Im August 2013 fuhr eine zum autonomen Fahren ausgestattete Mercedes-Benz S-Klasse entlang der 103 km langen BERTHA BENZ MEMORIAL ROUTE (Abbildung 7.1) autonom von Mannheim nach Pforzheim. Die dabei eingesetzte Trajektorienplanung war ein lokales, kontinuierliches Verfahren, bei dem die Trajektorie als Polygonzug mit einer endlichen Anzahl Stützstellen repräsentiert wurde, wie in Kapitel 5 beschrieben. Das angewendete Optimalitätskriterium ist das aus Gleichung (5.5), und das Problem wurde ohne Bedingungen am rechten Rand und ohne Transversalitätsbedingungen formuliert. Es wurde eine fortlaufende Neuplanung wie in Abschnitt 5.4.2 durchgeführt. Die Trajektorie wird durch  $N = 30$  Stützpunkte repräsentiert, die mit einer Schrittweite von  $\Delta t = 0.25$ s abgetastet sind.

Für die Befahrung der BERTHA-BENZ-Route wurde ein kartenbasierter Ansatz verfolgt (vergleiche Abschnitt 1.2.1), das bedeutet, dass die Trajektorienplanung in einem global (bzw. kartenrelativ) referenzierten Koordinatensystem erfolgt. Ein Trajektorienfolgeregler [Zie+14c] führt eine glob-

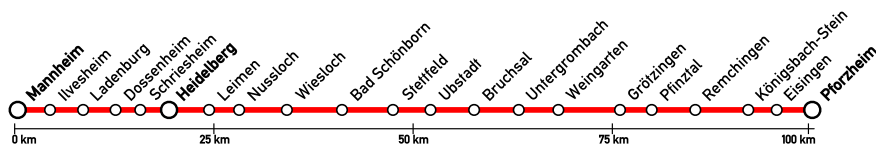
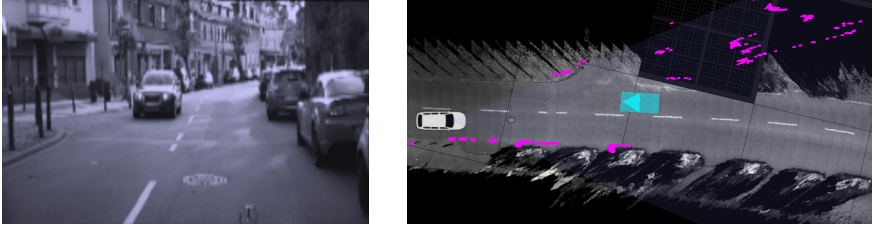


Abbildung 7.1: BERTHA-BENZ-MEMORIAL-ROUTE



**Abbildung 7.2:** Umfildrepräsentation: Statische Hindernisse und Objekte.

ale, videobasierte Posenschätzung zurück [SKF13; LS14; Zie+14d], um das Fahrzeug entlang der geplanten Trajektorie zu führen.

### 7.1.1 Karten- und sensorbasiertes Lagebild

In diesem Abschnitt werden kurz die Sensordaten beschrieben, die im BERTHA-BENZ-Projekt zur Verfügung standen. Der Schwerpunkt liegt dabei auf der Repräsentation, in der diese Sensordaten in die Trajektorienplanung eingehen. Es kamen ausschließlich seriennahe Kamera- und Radarsysteme zum Einsatz. Die Beschreibung von Sensorhardware ist nicht Teil dieser Arbeit. Dasselbe gilt für die Bild- und Signalverarbeitung, sowie die dieser nachgeschalteten Fusions- und Validierungsprozesse. Für eine ausführliche Darstellung der verwendeten Bildverarbeitungsalgorithmen wird auf [Fra+13] verwiesen. Die sensorische Umgebungserfassung wurde im Projekt durch detaillierte digitale Straßenkarten [BZS14b] ergänzt, in denen statische Eigenschaften der Umgebung im Vorhinein abgelegt wurden. Dies setzt eine kartenrelative Lokalisierung voraus, die durch videobasierte Verfahren realisiert wurde [SKF13; LS14; Zie+14d].

Das sensorische Lagebild setzt sich zusammen aus einer Abtastung statischer Hindernisse und einer Liste von Objekten. Unter Hindernissen werden im folgenden ausschließlich *statische* Objekte im Verkehrsraum verstanden. In diesem Projekt wurden sie durch Stereobildverarbeitung detektiert. Das Kamerabild wird hierbei in Spalten zerlegt, die einige Pixel breit sind. Pro Spalte wird dann die Entfernung zum jeweils nächstgelegenen Hindernis bestimmt. Die Hindernisse werden also an ungefähr äquidistanten Winkelstellen abgetastet. Ein einzelner Abtastpunkt wird im folgenden auch als *Stixel* bezeichnet. Die Stixel werden über einen kurzen Zeitraum in einem weltfesten Koordinatensystem akkumuliert. In Abbildung 7.2 sind Stixel rechts magentafarben dargestellt.



*Objekte* werden durch Fusion von Stereobild- und Radardatenverarbeitung erzeugt. Während die statische Hinderniskontur lediglich den Fahrkorridor des Fahrzeuges geometrisch einschränkt, spielen Objekte eine entscheidende Rolle auch auf der Ebene der Verhaltensentscheidung, beispielsweise für die Bewertung von Vorfahrtsituationen. In Abbildung 7.2 wird das entgegenkommende Fahrzeug als Objekt repräsentiert und ist hier als zyanfarbenes Viereck eingezeichnet. In Fusions- und Validierungsstufen wird sichergestellt, dass die beiden Repräsentationen – also die stixelbasierte Repräsentation der Freiraumkontur, und die objektlistenbasierte Repräsentation der Verkehrsteilnehmer – konsistent sind, und dass Redundanzen aufgelöst wurden. In der im Bild dargestellten Szene beispielsweise erzeugt der Stereo-Sensor zunächst Stixel, die sich an der Front des entgegenkommenden Fahrzeuges befinden. Dieses wird aber – redundant – auch von der radarbasierten Sensorik erkannt. Beide Messungen werden zu einer Objekthypothese fusioniert. Die dem Objekt zugeordneten Stixel werden aus der statischen Hindernisskontur entfernt.

Die im Bild zu sehende Bodentextur wurde bei einer Kartierungsfahrt mit einer nach hinten gerichteten Stereokamera erstellt. Die so gewonnenen Daten dienen sowohl als Grundlage zur Erzeugung einer digitalen Straßenkarte als auch zur Erzeugung einer Referenzkarte zur videobasierten Lokalisierung.

Mithilfe der detaillierten Straßenkarte wird das Lagebild um einen statischen Fahrkorridor ergänzt.

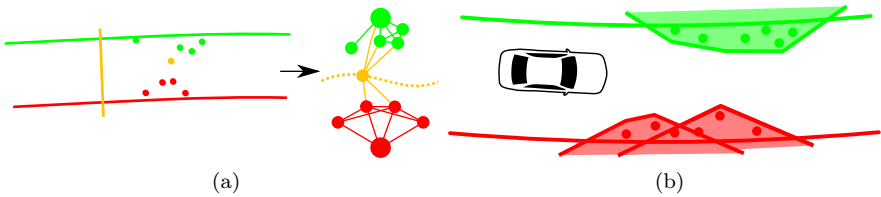
## 7.1.2 Äußere Nebenbedingungen

### Fahrkorridor und Statische Hindernisse

Als *Fahrkorridor* wird die linke und rechte seitliche Begrenzung des Fahrbereiches bezeichnet. Er wird einer Straßenkarte entnommen und ist aus den Fahrstreifensegmenten zusammengesetzt, die die Route zum Ziel bilden [BZS14b].

### Hindernisse

Statische Hindernisse werden wie in Abschnitt 7.2 beschrieben durch eine Menge von Stixeln repräsentiert. Für alle Stixel, die innerhalb des Fahrkorridors liegen, wird entschieden, ob das Fahrzeug sie links oder rechts passieren soll. Hierzu wird ein optimaler Graphschnitt durchgeführt. Der Aufbau



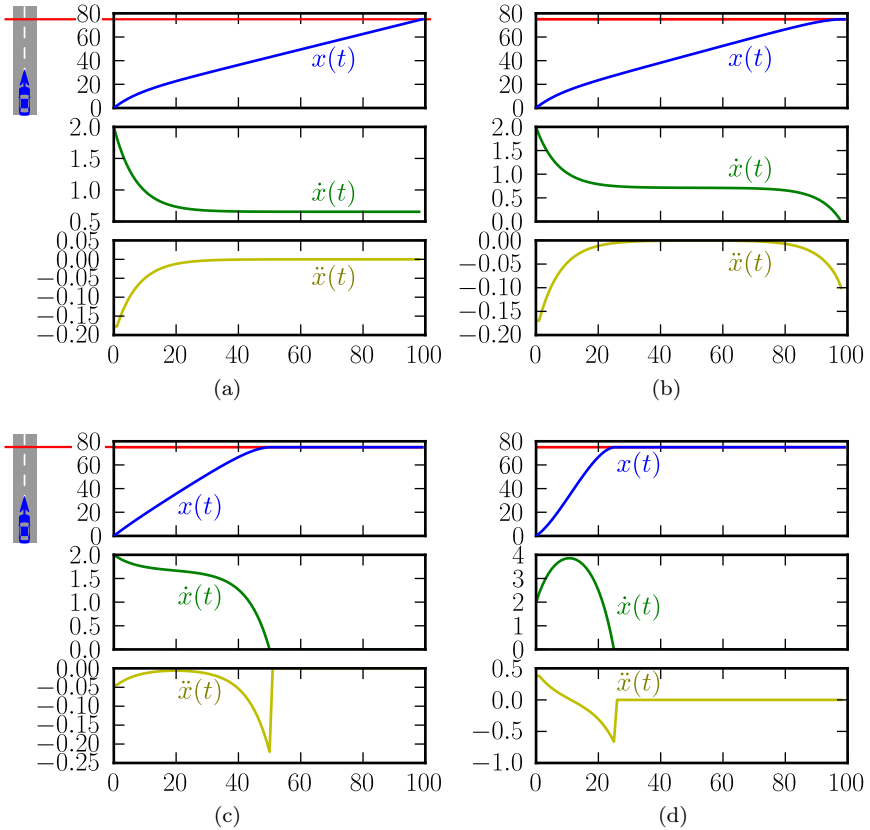
**Abbildung 7.3:** Vorverarbeitung der Hindernisdaten unter Berücksichtigung des Verlaufes des Fahrkorridors.

des Graphen ist in Abbildung 7.3a schematisch dargestellt: Jedem einzelnen Stixel entspricht ein Knoten im Graph. Die beiden größer dargestellten Knoten repräsentieren den linken und rechten Rand des Fahrkorridors. Zwei Knoten werden verbunden, wenn zwischen den entsprechenden Stixeln, bzw. zwischen entsprechendem Stixel und Korridorrand, eine Durchfahrt geometrisch nicht kollisionsfrei möglich ist. Es wird nun ein minimaler Schnitt (engl. *minimum vertex cut*) [EK72] durch den Graphen berechnet, das bedeutet, es wird die kleinst-mögliche Menge Knoten aus dem Graphen entfernt, sodass linker und rechter Fahrbahnrand nicht mehr zusammen hängen. In Abbildung 7.3a kann der Graph durch Entfernung von einem Knoten (orange) geschnitten werden. Wenn die Entfernung mindestens eines Knotens erforderlich ist, so bedeutet dies immer, dass an dieser Stelle ein Durchfahren nicht möglich ist. Dann wird an einer geeigneten Stelle eine Haltelinie erzeugt (im Bild orange). Haltelinien stellen eine besondere Form von Nebenbedingungen dar, sie werden im nächsten Unterabschnitt beschrieben.

Wenn alle Stixel auf diese Weise dem linken oder rechten Fahrstreifenrand zugeordnet wurden, werden sie wie in Abbildung 7.3b durch konvexe Polygone eingehüllt. Dies hat folgende praktische Gründe: Erstens kann auf diese Weise ein Sicherheitsabstand vorgesehen werden. Zweitens ist es zur Formulierung einer Nebenbedingung für die Trajektorienplanung erforderlich, eine differenzierbare Abstandsfunktion zu den Hindernissen anzugeben. In Kapitel 3 wurde demonstriert, wie eine solche aus einer polygonalen Repräsentation erzeugt werden kann.

## Haltelinien

Wie im vorangehenden Absatz erläutert wurde, wird, wenn eine Durchfahrt aufgrund von Hindernissen blockiert ist, an geeigneter Stelle eine Haltelinie platziert, um das Fahrzeug auf definierte Art und Weise zum Halten



**Abbildung 7.4:** Untersuchung des Kostenfunktional für das Anhalten an einer Haltelinie. Es sind jeweils Ort, Geschwindigkeit und Beschleunigung über der Zeit aufgetragen. Die rechte Randbedingung variiert: (a) rechter Rand frei. (b) Stopbedingung bei  $t = 100$ , (c) ...  $t = 50$ , (d) ...  $t = 25$ .

zu bringen. Auch andere Entscheidungen, die auf der Ebene der Trajektorienplanung übergeordneten Ebene der Verhaltensentscheidung [BZS14b] getroffen werden, führen zur Platzierung von Haltelinien, zum Beispiel rote Ampeln, Stoppschilder und die Behandlung von Kreuzungen im Allgemeinen.

Prinzipiell können Haltelinien, wie die Korridorländer, als eine gewöhnliche Ungleichungs-Nebenbedingung behandelt werden (die dann sogar linear ist). Aus folgenden Gründen wurden Haltelinien während der BERTHA-BENZ-Fahrt aber als Spezialfall behandelt.

Erstens stellt die Aufgabe des Anhaltens ein besonders gut untersuchtes Problem im Bereich der Fahrerassistenzforschung dar. Serienfahrzeuge sind – in Form von erweiterten, radarbasierten Abstandsregeltempomaten [WDS12] – mit Systemen zum definierten Anhalten hinter einem anderen Fahrzeug bereits ausgestattet. Im Fall des BERTHA-Fahrzeuges wurde, an Stelle einer Haltelinie, am Eingang des serienmäßigen Abstandsregeltempomaten einfach ein stehendes Objekt fingiert, wodurch sich das Fahrzeug an definierter Stelle in den Stand bringen ließ.

Zweitens ist die Art, wie Randbedingungen behandelt werden (Abschnitt 5.4.2) in Verbindung mit dem Kostenfunktional aus Abschnitt 5.2 schlecht für das Anhalten geeignet, wie jetzt veranschaulicht werden soll.

Abbildung 7.4 zeigt optimale (1D-)Trajektorien, die mit quadratischem Programmieren (Abschnitt 5.5.2) berechnet wurden. Eine Haltelinie (rot) liegt jeweils am Ort  $x_{\max} = 80.0$ . Der Integrand des Kostenfunktionals ist

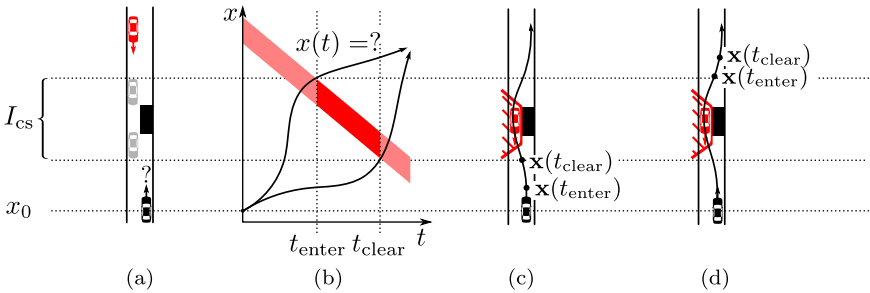
$$L(x, \dot{x}, \ddot{x}) = w_{\text{geschw}}(v_{\text{ref}} - \dot{x})^2 + w_{\text{besch}}\ddot{x}^2, \quad (7.1)$$

mit der Wunschgeschwindigkeit  $v_{\text{ref}} = 2.0$ . Am linken Rand wurde die Geschwindigkeit  $v_0 = v_{\text{ref}}$  festgehalten. Auch wenn dieses Kostenfunktional vom tatsächlich verwendeten etwas abweicht – Ablageterm, Drehratenterm und Ruckterm fehlen – beschreibt er die reine Längsdynamik für die folgenden Betrachtungen ausreichend gut. Der Ablageterm (vergleiche Abbildung 5.2) und der Drehratenterm sind ja bei reiner Betrachtung der Längsrichtung gar nicht wirksam. Die Regularisierung des Beschleunigungsverlaufes durch den Ruckterm fehlt zwar, dies ändert aber an den Schlussfolgerungen der folgenden Untersuchungen nichts. Abbildung 7.4a zeigt die Optimaltrajektorie für Randbedingungen wie in Abschnitt 5.4.2, die Trajektorie ist also am rechten Rand frei. Der Verlauf der Trajektorie erscheint nicht optimal. Erstens kommt das Fahrzeug über dem verwendeten Zeithorizont von 100 überhaupt nicht in den Stand. Stattdessen schmiegt sich die Trajektorie einem geradenförmigen Verlauf mit einer Geschwindigkeit unterhalb der Wunschgeschwindigkeit an. Die hierzu erforderliche Beschleunigung wird

praktisch sofort, am Anfang der Trajektorie, aufgebracht. Der Verlauf der Trajektorie lässt sich verbessern, indem der Anhaltewunsch in Form einer Randbedingung in die Problemformulierung eingebracht wird. In den Abbildungen 7.4ab-c wurde das Problem mit einer entsprechenden Randbedingung versehen, wobei der Zeitpunkt des Anhaltens jeweils unterschiedlich gewählt wurde (nach  $t = 100$ ,  $t = 50$  und  $t = 25$ ). Die ersten beiden Fälle führen jeweils dazu, dass ein Teil der Beschleunigung am Anfang aufgebracht wird, und ein anderer unmittelbar vor dem Anhalten. Im letzten Fall muss sogar erst (positiv) beschleunigt werden, um rechtzeitig an der Haltelinie anhalten zu können. Die drei Fälle sollen veranschaulichen, dass nach der optimalen Anhaltezeit erst „gesucht“ werden muss. Sie müsste mit optimiert werden, es ist also eine Transversalbedingung (Abschnitt 4.1) zu erfüllen. Transversale Probleme lassen sich in das verwendete Diskretisierungsschema nicht unmittelbar einbringen, weshalb auf ein „geplantes“ Anhalten an Haltelinien verzichtet wurde, und stattdessen wie oben beschrieben ein einfacher Anhalteregler verwendet wurde.

### Kritische Abschnitte

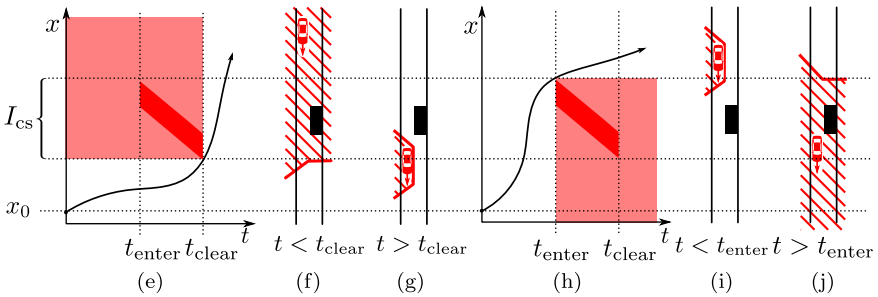
Ein kritischer Abschnitt (*critical section*) entsteht, wenn das eigene Fahrzeug einem entgegenkommenden in einem Bereich begegnet, der so schmal ist, dass er nicht von beiden Fahrzeugen gleichzeitig passiert werden kann. Eine derartige Situation ist in Abbildung 7.5a schematisch dargestellt. Das rote Fahrzeug ist das entgegenkommende, und für das schwarze Fahrzeug soll eine Trajektorie geplant werden. Das schwarze Rechteck stellt ein statisches Hindernis dar, durch das die Engstelle zustande kommt. Der Bereich  $I_{cs}$  stellt den Bereich dar, in dem sich nur genau ein Fahrzeug befinden darf, dieser Bereich ist um zweimal die Länge des entgegenkommenden Fahrzeuges länger als das Hindernis – dies wird durch die grau schattierten Fahrzeuge veranschaulicht. Das eigene Fahrzeug wird der Einfachheit halber im Folgenden als punktförmig angenommen. Zunächst wird diese Situation im Eindimensionalen betrachtet. Die Raumkoordinate  $x$  zeigt in Abbildung 7.5 immer nach oben. In Abbildung 7.5b ist sie über der Zeit abgetragen. Es wird angenommen, dass die Trajektorie des entgegenkommenden bekannt ist, und der Bereich, den er in der Raum-Zeit-Ebene überstreichen wird, ist in Abbildung 7.5b rot hervorgehoben, wobei die Bereiche außerhalb des kritischen Abschnittes aufgehellt sind. Der entgegenkommende tritt zum Zeitpunkt  $t_{\text{enter}}$  mit dem vorderen Ende in den kritischen Abschnitt ein, und hat diesen zum Zeitpunkt  $t_{\text{clear}}$  wieder vollständig verlassen. Die Topologie des Freiraumes erlaubt genau zwei unterschiedliche Manöverklassen – beide wer-



**Abbildung 7.5:** Bildung der Nebenbedingungen bei kritischen Abschnitten. Fortsetzung auf gegenüberliegender Seite. Erläuterungen im Fließtext.

den in Abbildung 7.5b durch je einen Repräsentanten (schwarz) visualisiert. Die Trajektorien der einen Klasse passieren den kritischen Abschnitt zeitlich nach dem Entgegenkommer, die der anderen vor ihm. In Abbildung 7.5c-d sind die beiden Beispieltrajektorien in der  $xy$ -Ebene dargestellt, die Orte  $\mathbf{x} = (x, y)^\top$  des eigenen Fahrzeuges zu den Zeitpunkten  $t_{enter}$  und  $t_{clear}$  sind hervorgehoben. Die Position des Entgegenkommers und des entsprechenden zeitabhängigen Nebenbedingungspolygons entspricht einem Zeitpunkt zwischen  $t_{enter}$  und  $t_{clear}$ . Die Darstellung soll veranschaulichen, dass ein lokales Optimierungsverfahren diese Situation nicht ohne Weiteres lösen können. In welcher der beiden Lösungsklassen das Optimierungsverfahren konvergieren wird, ist nicht vorauszusehen, und wird stark von der gewählten Initialisierung der Trajektorie abhängen. Der Optimierer müsste ja, um von der in Abbildung 7.5c dargestellten Lösungsklasse in die aus Abbildung 7.5d überzugehen, die beiden hervorgehobenen Punkte über einen unzulässigen Bereich „schieben“ – es müsste also eine Nebenbedingung zunächst verschlechtert werden. Ein lokales Optimierungsverfahren wird aber prinzipiell in jedem Schritt eine Verbesserung der Nebenbedingungen anstreben. Verallgemeinernd kann festgestellt werden, dass lokale Verfahren zur Trajektorienoptimierung nicht geeignet sind, um derartige diskrete Entscheidungen „selbst“ zu treffen.

Um in Situationen wie diesen ein vorhersehbares Fahrzeugverhalten zu erreichen, wurde der Entscheidungsprozess der Optimierung vorgelagert. Die getroffene Entscheidung wird dann in die Bildung der Nebenbedingungen eingebracht, um eine eindeutige Konvergenz des Optimierers zu erzwingen. Die Entscheidung selber wird *ad hoc*, basierend auf einfachen Regeln, getroffen. Prinzipiell wird entschieden, *vor* dem Entgegenkommer den kritischen Abschnitt zu passieren, wenn, basierend auf der eigenen Geschwindigkeit



und der des Entgegenkommers, davon auszugehen ist, dass das eigene Fahrzeug die Engstelle zuerst erreicht. Diese Entscheidung wird aber konservativ – also mit zeitlicher Reserve – getroffen, und außerdem mit einer Hysterese versehen, damit sie über der Zeit stabil bleibt. Der Entscheidungsprozess kann auch zum Platzieren einer Haltelinie führen, die dann so, wie im letzten Abschnitt beschrieben wurde, behandelt wird.

Die Abbildungen 7.5e-g illustrieren, wie das System der Nebenbedingungen aufgebaut wird, und zwar für den Fall, dass die Entscheidung getroffen wurde, die Engstelle *nach* dem Entgegenkommer zu passieren. Wir betrachten die Situation zunächst im Eindimensionalen (Abbildung 7.5e). Aufgrund der getroffenen Entscheidung kann der unzulässige Bereich in der Raum-Zeit-Ebene wegen zweier Monotoniebedingungen wie dargestellt erweitert werden: Das Fahrzeug kann sich in der Zeitdimension  $t$  nicht „rückwärts“ bewegen, und es wird angenommen, dass es sich auch in der Raumdimension  $x$  monoton vorwärts bewegt. In der  $xy$ -Ebene führt dies zu einem Nebenbedingungspolygon wie in Abbildung 7.5f, das sich für die Zeiten  $t < t_{clear}$  nicht verändert. Für Zeiten  $t > t_{clear}$  wird das entgegenkommende Fahrzeug durch ein zeitvariantes, trapezförmiges Polygon wie in Abbildung 7.5g eingehüllt. Die Begründung für die Einhüllung des Objektes durch ein solches nach links offenes Trapez folgt der gleichen Argumentationsweise wie die Bildung der Polygone für statische Hindernisse (siehe oben und Abbildung 7.3): Es soll für den Optimierer eindeutig festgelegt werden, an welcher Seite das Fahrzeug zu passieren ist. Dabei wird angenommen, dass entgegenkommende Fahrzeuge stets rechts zu passieren sind. Insgesamt führt die Auslegung der Nebenbedingung dazu, dass die Trajektorienoptimierung, unabhängig von ihrer Initialisierung, immer in der gewählten Manöverklasse konvergiert. Die Abbildungen 7.5h-j zeigen analog die Konstruktion der Nebenbedingungen, falls entschieden wurde, als Erster den kritischen Abschnitt zu passieren.

### 7.1.3 Ergebnisse

Mit dem vorgestellten System gelang es, die anspruchsvolle Route in sechs Teilabschnitten ohne Fahrereingriff zu durchfahren. Über das Gesamtexperiment wurde in [Zie+14c; Zie+14a] berichtet, Teilsysteme wurden jeweils in [Fra+13; Zie+14d; Zie+14b; BZS14a; Lat+13] präsentiert. Durch den sorgfältigen Entwurf des Gütefunktional gelang es, der Trajektorie viele wünschenswerte Eigenschaften implizit aufzuprägen. Das Gütefunktional bewirkt beispielsweise, dass das Fahrzeug in Kurven oder beim Abbiegen die Fahrt verlangsamt, ohne dass hierzu „von Außen“ explizit die Geschwindigkeitsvorgabe geändert werden muss. Kurven werden vom Fahrzeug angeschnitten. Das resultierende Fahrverhalten wurde von Passagieren als natürlich und angenehm beschrieben.

Es werden nun einige Ergebnisse der Trajektorie anhand von Abbildungen präsentiert. Es ist jeweils folgendes dargestellt:

- schwarz: das eigene Fahrzeug und die vorausgeplante eigene Trajektorie des Hinterachspunktes
- grau: durch Kreise angenäherte Manövrierfläche des eigenen Fahrzeuges (vergl. Abschnitt 5.3.1)
- rot/grün: linker/rechter Rand des Fahrkorridors, der digitalen Karte entnommen
- blau: Ränder anderer (nicht zum eigenen Fahrkorridor gehörender) Fahrstreifen
- violett: statische Hindernispunkte, die so weit außerhalb des Fahrkorridors liegen, das sie die Fahrentscheidung nicht beeinflussen können
- rot/grün: statische Hindernispunkte, die nach dem vorgeschalteten Entscheidungsprozess (Abschnitt 7.1.2) rechts bzw. links passiert werden sollen.
- orange: einhüllende Polygone für statische Hindernispunkte (Abschnitt 7.1.2).

Die Abbildungen 7.6a-c zeigen Beispieltrajektorien in simulierter Umgebung. Bei der Durchfahrung des Kreisverkehrs (Abbildung 7.6a) erkennt man gut den Einfluss der fahrdynamischen Komponenten des Gütefunktional: Der Kreisverkehr wird zunächst links angefahren, um dann mit



einem möglichst großen Kurvenradius nach rechts in ihn einbiegen zu können. Die Scheitelpunkte der drei gefahrenen Kurven werden jeweils berührt. Die Abbildungen 7.6a und 7.6b zeigen, dass auch in der Umgebung von Hindernissen eine subjektiv ideale Linie gefunden wird.

Die Abbildungen 7.7 und 7.8 zeigen in Einzelbildern Durchfahrten der Orte Ubstadt und Feudenheim. Es handelt sich um Bildschirmkopien der während der Fahrt in Echtzeit laufenden Planungsapplikation. Bei dem kurzen Streckenabschnitt in Ubstadt (Abbildung 7.7) handelt es sich um eine Einbahnstraße. Eine Reihe parkender Autos wird links passiert. Die Sequenz aus Feudenheim (Abbildung 7.8) zeigt eine ähnliche Sequenz, bei der aber auch entgegenkommender Verkehr zu behandeln ist. Die Radfahrerinn – man sieht sie im vierten und fünften Bild von oben – bewirkt zeitweise die Platzierung einer Haltelinie (rot, im vierten Bild von oben zu sehen). Wegen ihrer niedrigen Geschwindigkeit wird sie vom Wahrnehmungssystem als statisches Hindernis interpretiert. Die entgegenkommenden Kraftfahrzeuge werden richtig als dynamische Hindernisse interpretiert. Ihre prädierten Trajektorien sind als Ketten verschiedenfarbiger Dreiecke dargestellt.

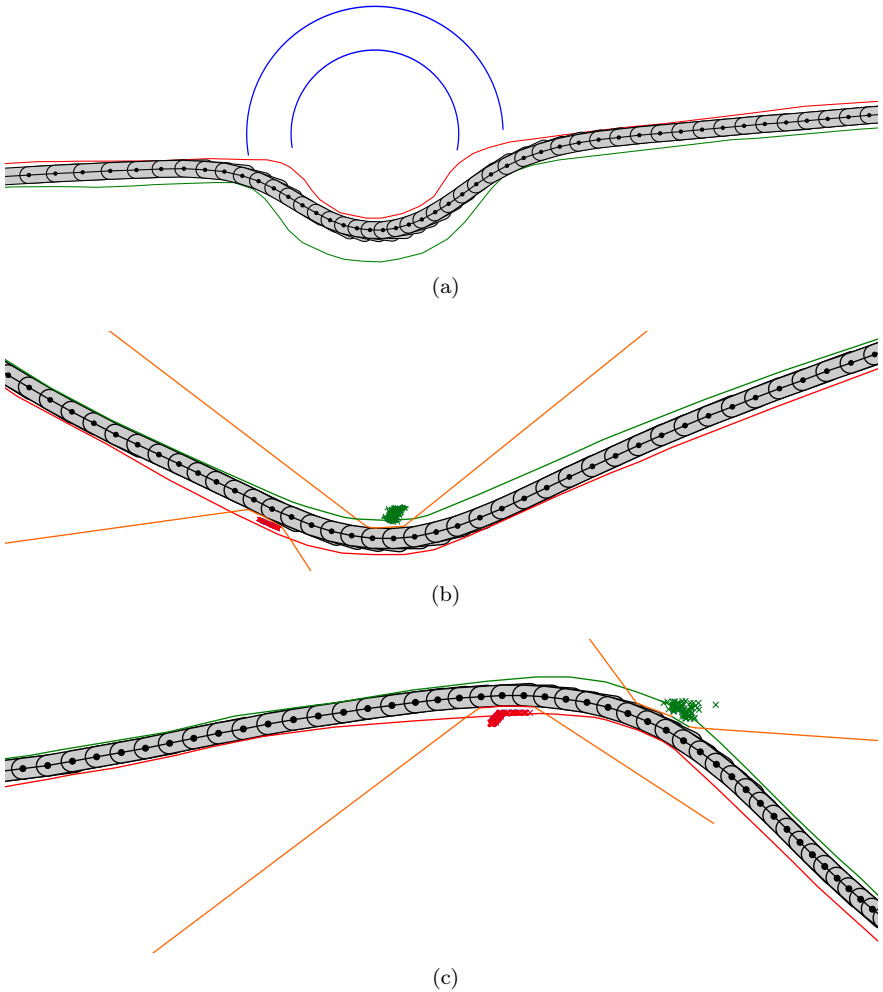
## 7.2 Navigation in beengter Umgebung

Für den Anwendungsfall der Navigation in beengter Umgebung wurde das in Abschnitt 6.2 beschriebene globale Verfahren mit einem infinitesimalen Gitter umgesetzt. Zur nachträglichen Verfeinerung der so geplanten Trajektorie wird ein kontinuierliches, lokales Verfahren nachgeschaltet. Gemäß dem Anwendungsszenario werden die Trajektorien für langsame Geschwindigkeiten ( $<15\text{km/h}$ ) geplant, die dann als Pfade interpretiert werden, es wird also keine Führung des Fahrzeuges in Längsrichtung angestrebt.

In diesem Abschnitt wird auf praktische Details der Umsetzung dieses hybriden Ansatzes eingegangen.

### 7.2.1 Gitter

Das Gitter hat eine räumliche Auflösung  $\delta s$  von 0.1 m, die zeitliche Auflösung beträgt  $\delta t = 1\text{s}$ . Als Maximalgeschwindigkeit wurde  $v_{\max} = 1.5\frac{\text{m}}{\text{s}}$  gewählt, bei einer maximalen Beschleunigung von  $a_{\max} = 0.5\frac{\text{m}}{\text{s}^2}$ . Der generierte Graph enthält pro diskretem Raumpunkt 1368 Knoten (diese entsprechen den Geschwindigkeiten, vergleiche Abbildung 6.3), und pro Knoten durchschnittlich 33 eingehende und ausgehende Kanten. Es wurde



**Abbildung 7.6:** Beispieltrajektorien mit simulierten Hindernissen. Fahrtrichtung von links nach rechts. Die Manöverfläche des Fahrzeuges ist grau, Nebenbedingungspolygone orange, die Fahrstreifenbegrenzungen sind rot (links) und grün (rechts). Einzelne Hindernisspunkte (Stixel) sind als Kreuze dargestellt und in der Farbe des entsprechenden Fahrstreifenrandes eingefärbt.

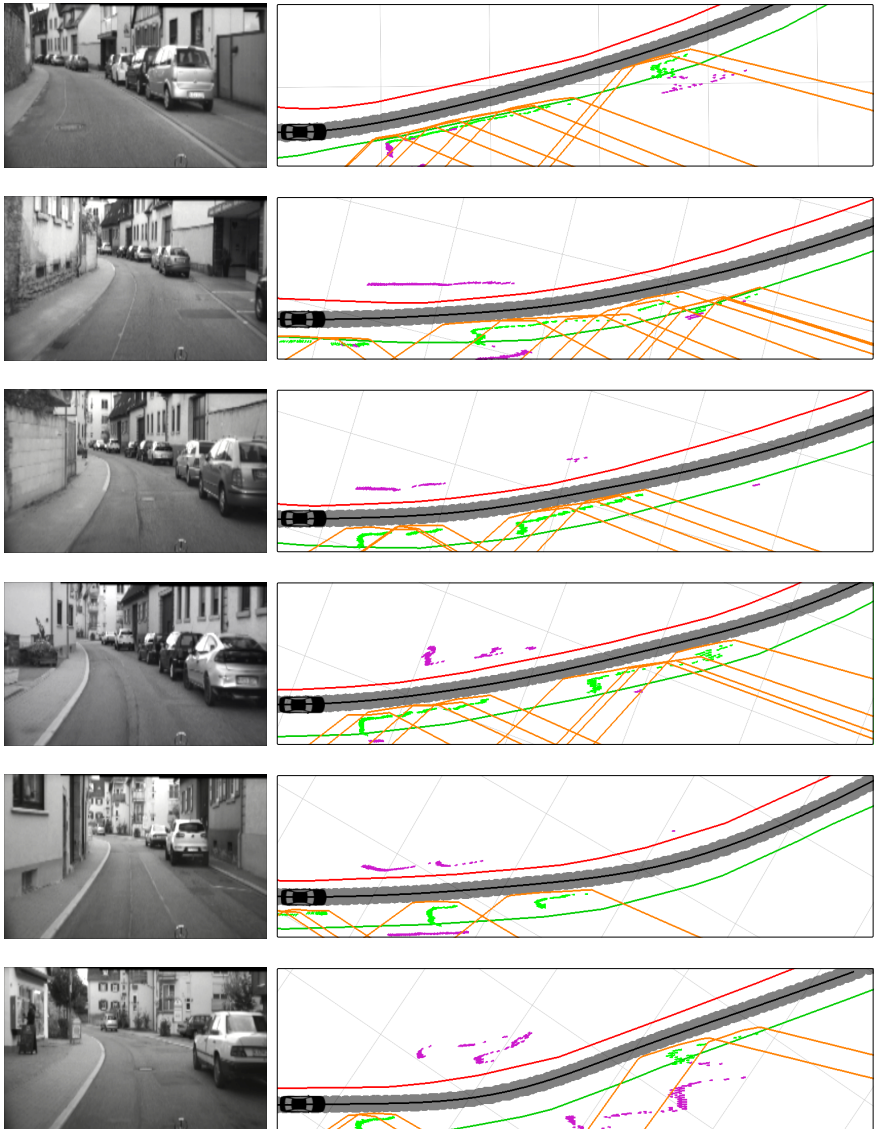


Abbildung 7.7: Ortsdurchfahrung Ubstadt

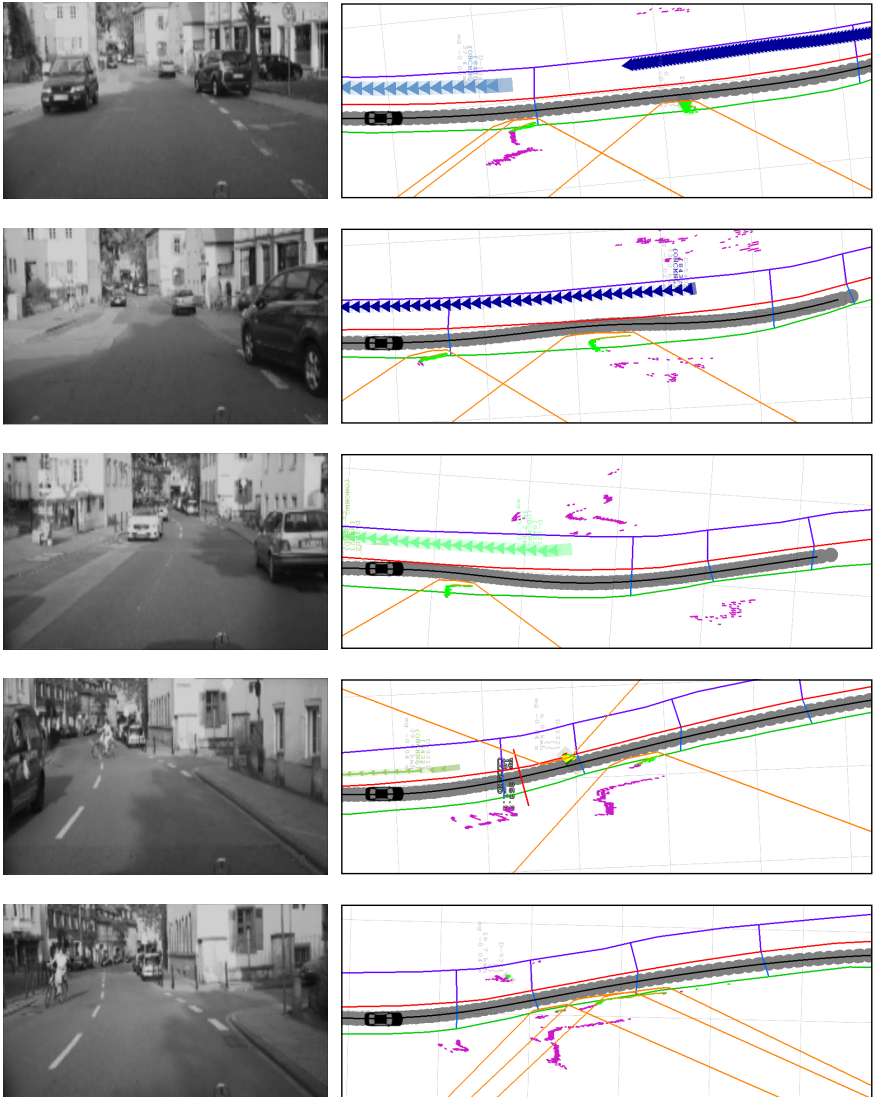
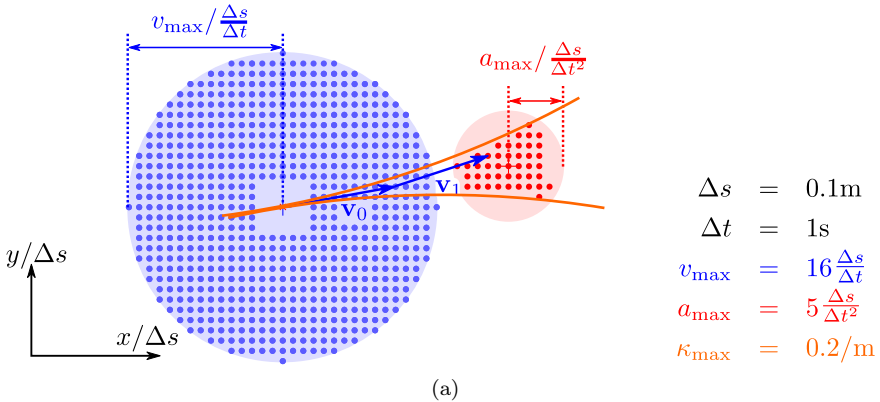


Abbildung 7.8: Ortsdurchfahrung Feudenheim



**Abbildung 7.9:** Gitter für die Navigation in beengter Umgebung.

in einem quadratischem Arbeitsraum bis zu einer Größe von 512 mal 512 diskreten Raumpunkten geplant, was einer Ausdehnung des Arbeitsraumes von 51.2 mal 51.2 Metern entspricht. Der Graph enthält insgesamt etwa  $3.6 \cdot 10^8$  Knoten und  $1.12 \cdot 10^{10}$  Kanten. Die Laufzeit für eine erschöpfende Suche in diesem Graphen - also für die Bestimmung des kürzesten Pfades vom Startknoten zu allen anderen Knoten im Graphen - beträgt auf zeitgemäßer Hardware (Intel i7, 3.3 GHz) ca. 10 Sekunden. Zu beachten ist, dass der Speicheraufwand für das Vorhalten der Knoten erheblich ist - pro Knoten werden mindestens 4 Bytes benötigt, hier insgesamt also etwa 1.4 Gigabyte. Abbildung 7.9 stellt einen Ausschnitt aus diesem Graphen dar. Die Darstellung ist an Abbildung 6.3 angelehnt. Jeder blaue Punkt repräsentiert den Endpunkt eines möglichen Geschwindigkeitsvektors, wobei sein Fußpunkt in der Kreismitte liegt. Beispielfhaft ist im Bild von diesen Geschwindigkeitsvektoren ein einzeln herausgehoben und mit  $\mathbf{v}_0$  bezeichnet. Die roten Punkte stellen nun die Endpunkte aller möglichen *nachfolgenden* Geschwindigkeitsvektoren da, wobei der Fußpunkt dieses Nachfolgevektors dem Endpunkt von  $\mathbf{v}_0$  entspricht. Ein beispielhafter Nachfolgevektor ist wieder hervorgehoben und als  $\mathbf{v}_1$  gekennzeichnet. Die Grenzwerte  $v_{\max}$  und  $a_{\max}$  sind als schattierte Kreise dargestellt. Im roten  $a_{\max}$ -Kreis fehlen einige rote Punkte - er ist nicht komplett damit ausgefüllt - da diese als Nachfolger von  $\mathbf{v}_0$  den maximal erlaubten Krümmungsbetrag  $\kappa_{\max}$  überschreiten würden.

## 7.2.2 Kostenfunktional

Das Kostenfunktional, das in diesem Beispiel verwendet wurde, hat die Form

$$E(\mathbf{x}) = \int_0^{t_{\text{ende}}} (1 + w_{\text{beschl}}\ddot{\mathbf{x}})dt + c_{\text{schalt}}n_{\text{schalt}}, \quad (7.2)$$

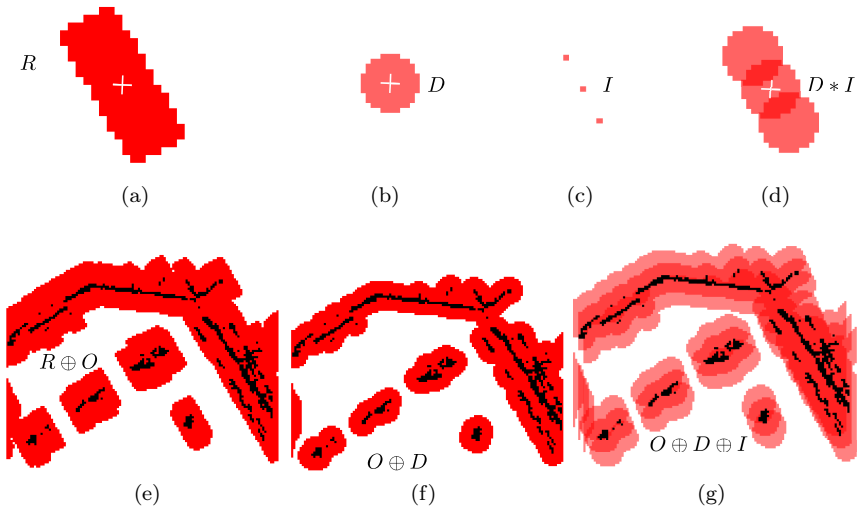
da der Integrand einen konstanten Anteil enthält, hängen die Gesamtkosten von der Reisezeit  $t_{\text{ende}}$  ab. Diese wird mit optimiert, es wird also ein transversales Problem gelöst (vergleiche Abschnitt 5.4). Die Reisezeit macht den Hauptteil des Kostenfunktionals aus, der enthaltenen Beschleunigungsterm dient im wesentlichen der Regularisierung der Ergebnistrajektorien, da er - auch ohne die Einführung von Nebenbedingungen - verhindert, dass die Richtung der Trajektorie sich sprungartig verhält. Um auch einen Richtungswechsel des Fahrzeuges zu ermöglichen - damit ist gemeint, dass das Fahrzeug von Vorwärts- auf Rückwärtsfahrt wechseln kann - werden noch zusätzliche Kanten in den Graphen mit aufgenommen, nämlich all jene, bei denen ein diskreter Geschwindigkeitsvektor auf seine Inverse übergeht, es gibt also eine Kante zwischen den Knoten  $\langle s, t \rangle$  und  $\langle t, s \rangle$ , für alle  $t, s \in V(G)$ , vergleiche hierzu auch Abbildung 6.3 und die Notation aus Abschnitt 6.2.2. Eine derartige „Schalt“-Kante wird mit den Kosten  $c_{\text{schalt}} = 15$  belegt, ein Schaltvorgang ist damit genau so „teuer“ wie 15 Sekunden gleichförmige Geradeausfahrt. In Gleichung (7.2) ist  $n_{\text{schalt}}$  die Anzahl solcher Schaltvorgänge, die in einer Kandidatentrajektorie enthalten sind.

## 7.2.3 Nebenbedingungen

Bei dem Verfahren mit infinitesimalem Gitter gehen die Nebenbedingungen direkt in die Bildung des Graphen ein.

Die *inneren* Nebenbedingungen für Geschwindigkeit, Beschleunigung und Krümmung definieren, wie oben gezeigt, die lokale Nachbarschaft von Knoten im Graphen.

*Äußere* Nebenbedingungen werden in Form einer diskreten Hinderniskarte eingebracht, die mit Hilfe eines Laserscanners erstellt wurde. Die Zellgröße dieser Karte ist an das Auflösungslimit  $\delta s$  des Suchgitters angepasst. Basierend auf dieser Hinderniskarte werden Knoten aus dem Graphen entfernt, die nicht kollisionsfrei befahren werden können. Zu beachten ist dabei, dass der Kollisionszustand nicht nur von der Position des Fahrzeuges abhängt, sondern auch von seiner Ausrichtung. In diesem Zusammenhang



**Abbildung 7.10:** Dekomposition des Kollisionstests in Faltungen. (a): Fahrzeugförmiger Faltungskern für eine bestimmte Orientierung (b): Kreisförmiger Faltungskern. (c): Faltungskern aus drei Impulsfunktionen. (d): Faltung der vorigen beiden Kerne. (e): Minkowski-Summe aus Hinderniskarte und Formkern. (f): Minkowski-Summe aus Hinderniskarte und Kreiskern. (g): Faltung aus Kreiskern, Hinderniskarte und Impulskern.

bezeichnet man die Kombination aus Position und Orientierung auch als *Konfiguration* des Fahrzeuges [Kav95]. Da diese Überprüfung im *worst case* für jeden der  $3.6 \cdot 10^8$  Knoten des Graphen durchgeführt werden muss, macht sie insgesamt einen signifikanten Anteil der Rechenzeit aus. Die Berechnung kann im wesentlichen auf Faltungsoperationen reduziert werden, die schnell berechnet werden können. Ein effizientes Verfahren für dieses Problem wurde in [ZS10] präsentiert. Die Grundidee ist in Abbildung 7.10 dargestellt. Ausgangspunkt ist die Überlegung, das man durch Faltung der diskreten Hinderniskarte mit einem fahrzeugförmigen Kern (Abbildungen 7.10a und 7.10e) alle Zellen bestimmen kann, in denen sich das Fahrzeug in Kollision befände, wenn man es dort platzieren würde. Die Summe dieser Zellen nennt man die Konfigurationsraumhindernisse. Abbildung 7.10e zeigt das Ergebnis einer solchen Faltung, wobei alle Zellen mit einem Wert  $> 0$  rot eingefärbt sind. Dieses Vorgehen würde erfordern, das man für jede mögliche Fahrzeugorientierung eine solche Faltungsoperation

durchführt. Stattdessen kann man die Fahrzeugform - ähnlich wie schon in Abschnitt 5.3.1 - durch Vereinigung mehrerer Kreisscheiben annähern. Die Kreisform ist offensichtlich rotationssymmetrisch, deshalb reicht für die Bestimmung der Kollisionraumhindernisse eines Kreises eine einzelne Faltung aus (Abbildungen 7.10b und 7.10f). Um ein Fahrzeug näherungsweise auf Kollision zu überprüfen, müssen dann nur noch die Kreiszentren überprüft werden. Die Gültigkeit dieses Vorgehens folgt aus der Assoziativität der Faltungsoperation. Man kann statt einer Faltung der Hinderniskarte  $O$  mit dem angenäherten Formkern  $D * I$  auch eine Faltung der mit dem Kreiskern gefalteten Hinderniskarte  $O * D$  mit dem Impulskern  $I$  durchführen, da stets gilt  $O * (D * I) = (O * D) * I$ .

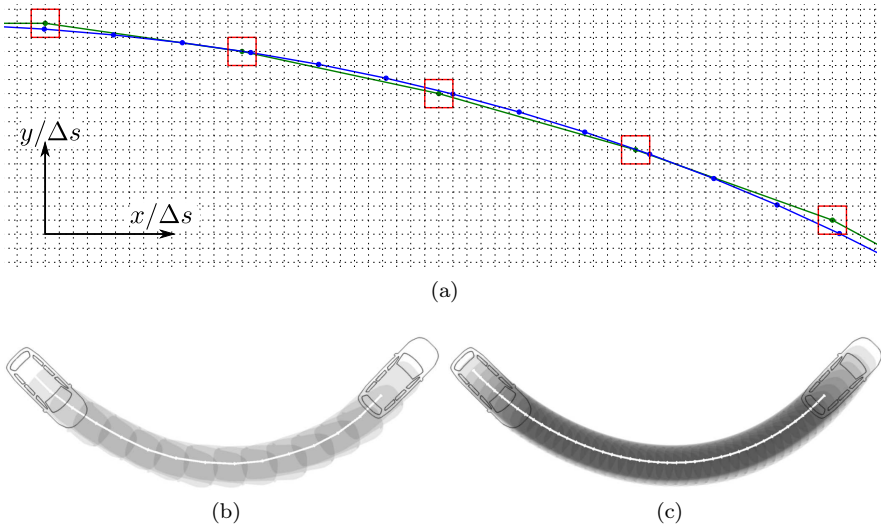
### 7.2.4 Randbedingungen

Wie in Abschnitt 6.2 beschrieben wurde, wird die global optimale Trajektorie durch das Gitter gefunden, indem eine erschöpfende kürzester-Pfad-Suche durchgeführt wird. Graphentheoretisch bedeutet dies, dass ein Spannbaum des Graphen von einem Startknoten aus aufgespannt wird, und zwar so, dass der Pfad von jedem Knoten zur Wurzel des Spannbaumes dem kürzesten Pfad zum Startknoten durch den Gesamtgraphen entspricht. Man erhält also den kürzesten Pfad von einem Startknoten zu *allen* anderen Knoten im Graphen. Da ein Knoten ja einem Geschwindigkeitsvektor entspricht, bedeutet dies geometrisch, dass man nur *eine* Randbedingung  $\mathbf{x}_{\text{start}}, \dot{\mathbf{x}}_{\text{start}}$  für Position und Richtung des Fahrzeuges festlegt, und nach Expansion der Suche die Optimaltrajektorie zu *jeder* Zielpose  $\mathbf{x}_{\text{ende}}, \dot{\mathbf{x}}_{\text{ende}}$  gefunden hat. Man kann die Suche auch *invers* betreiben, das heißt, man legt die Zielpose fest und bestimmt mit der Expansion des Suchbaumes die Optimaltrajektorie dorthin, von jeder beliebigen Startpose aus. Ein solcher inverser Suchbaum wird in der Literatur LaValle [LaV06] auch als *Feedback-Plan* bezeichnet.

### 7.2.5 Lokale Verfeinerung

Durch das Auflösungslimit  $\delta s$  bei der erforderlichen Diskretisierung auf ein räumliches Gitter ist die Trajektorie suboptimal. Sie wird ja durch einen Polygonzug approximiert, dessen Stützpunkte erstens zu diskreten Zeiten abgetastet sind, und die zweitens immer genau auf einen Gitterpunkt fallen, also einen zusätzlichen räumlichen Quantisierungsfehler aufweisen. Um die Trajektorie nachträglich zu glätten, wird ein einfaches lokales, kon-





**Abbildung 7.11:** Lokale Verfeinerung der Trajektorie. (a) Die Trajektorie wird optimiert, wobei die Stützstellen durch kastenförmige Nebenbedingungen (rot) eingeschränkt werden. Trajektorie (b) vor (c) nach lokaler Verfeinerung.

tinuierliches Verfahren angewendet, wobei Nebenbedingung für den Ort der Stützpunkte (*box constraints*) aus der mit dem globalen Verfahren gewonnenen Trajektorie erzeugt werden. Abbildung 7.11a illustriert das Vorgehen: Die Stützpunkte der Ausgangstrajektorie (grün) verbinden exakte Eckpunkte des diskreten Gitters. Mit einem kontinuierlichen Verfahren werden nun erstens die Stützpunkte verschoben, aber jeweils maximal um die Gitterauflösung. Dazu werden Nebenbedingungen erzeugt, die in der Abbildung als rote Kästen eingezeichnet sind. Jede Stützstelle darf verschoben werden, muss aber innerhalb des entsprechenden Kastens bleiben. Zweitens werden Stützstellen hinzugefügt, um die zeitliche Auflösung zu erhöhen, im Beispiel werden auf jedem linearen Segment der Ausgangstrajektorie zwei zusätzliche Stützstellen erzeugt. Die dem kontinuierlichen Verfahren zugrunde liegende Kostenfunktion entspricht Gleichung (7.2). Bei der lokalen Optimierung wird aber keine Entscheidung mehr über Richtungswechsel getroffen, deshalb ist der Term  $c_{\text{schalt}}n_{\text{schalt}}$  konstant und kann entfallen. Auch die Reisezeit  $t_{\text{ende}}$  wird nicht mehr mit optimiert. Wie in Abschnitt 5.5.2 gezeigt wurde, hat das so gestellte lokale Optimierungsproblem die Sonderform eines quadratischen Programms. Es kann deshalb in einem Berechnungsschritt ex-

akt gelöst werden. Die lokal optimierte Trajektorie ist in Abbildung 7.11a blau eingezeichnet. Die Abbildungen 7.11b und 7.11c zeigen ein weiteres Beispiel für das Ergebnis der lokalen Verfeinerung.

### 7.2.6 Ergebnisse

Die Abbildungen 7.12 und 7.13 zeigen Beispieltrajektorien, die mit dem Verfahren berechnet wurden. Abbildungen 7.12a bis 7.12d zeigen vier Varianten des Standard-Parkmanövers „quer einparken“. Abbildung 7.12e zeigt ein Beispiel für das Manöver „seitwärts einparken“. Die Abbildungen 7.13a und 7.13b zeigen zwei Varianten eines 180°-Wendemanövers in beengter Umgebung. In Abbildung 7.13a wird in einem Korridor der Breite 8 Meter gewendet. Der Korridor in Abbildung 7.13b ist nur 4 Meter breit, es steht aber ein Seitenkorridor als Wendehammer zur Verfügung.

Zur Erprobung des Systems im realen Fahrversuch wurde es in ein Erprobungsfahrzeug des Instituts für Mess- und Regelungstechnik integriert. Die Erfassung der Umgebung wurde mit Hilfe eines Laserscanners bewerkstelligt. Abbildung 7.14 zeigt eine Beispielsitzung mit dem System. Das Manöver ist ähnlich Abbildung 7.12b, es wird aber durch einen vor der Parklücke platzierten Reifenstapel erschwert. Abbildung 7.14a zeigt die Sicht des Fahrers. Er erhält auf einem Touchscreen eine Darstellung der mit dem Laserscanner erfassten Repräsentation der Umgebung. Über den Touchscreen und einen Drehsteller kann er eine beliebige Zielposition und -orientierung für das Fahrzeug vorgeben. Er erhält dann eine Darstellung der vom System geplanten Bahn, die dann auf Knopfdruck automatisch ausgeführt wird.

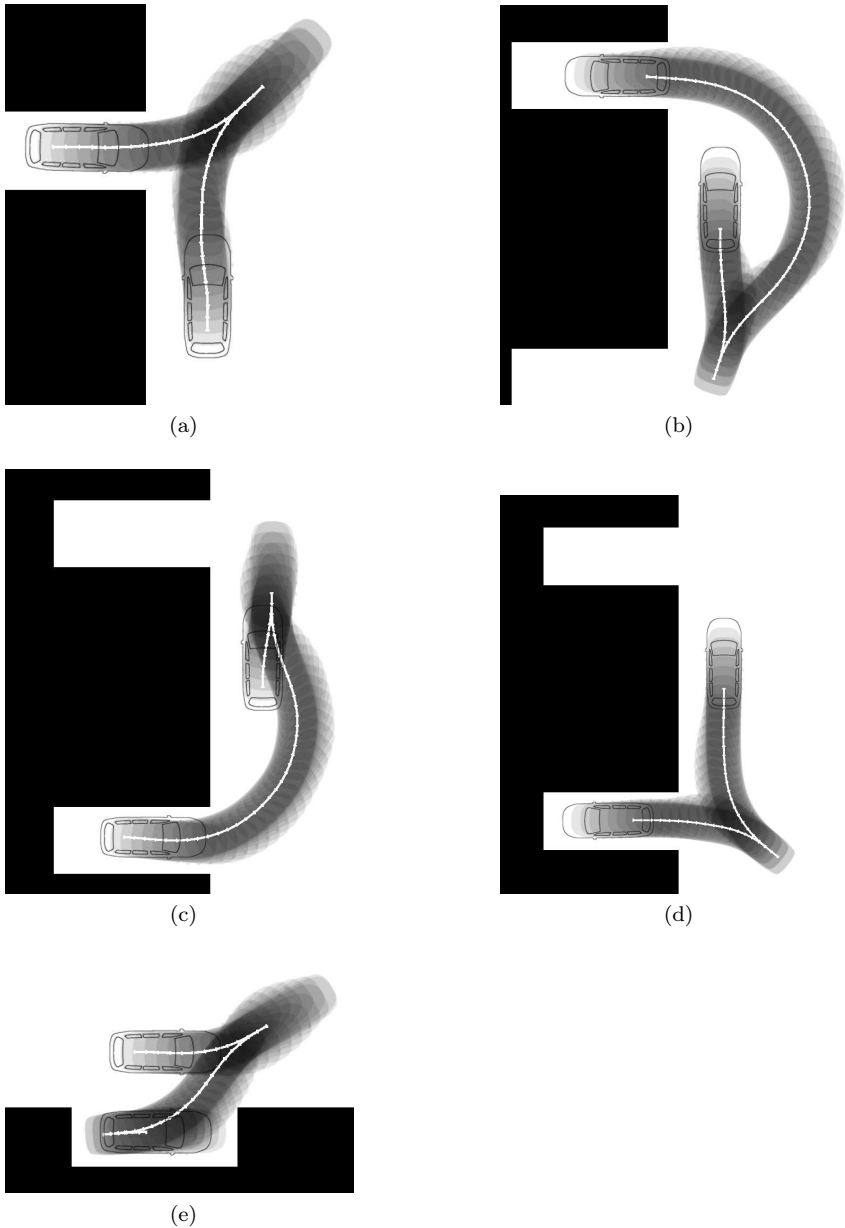


Abbildung 7.12: Standard-Parkmanöver

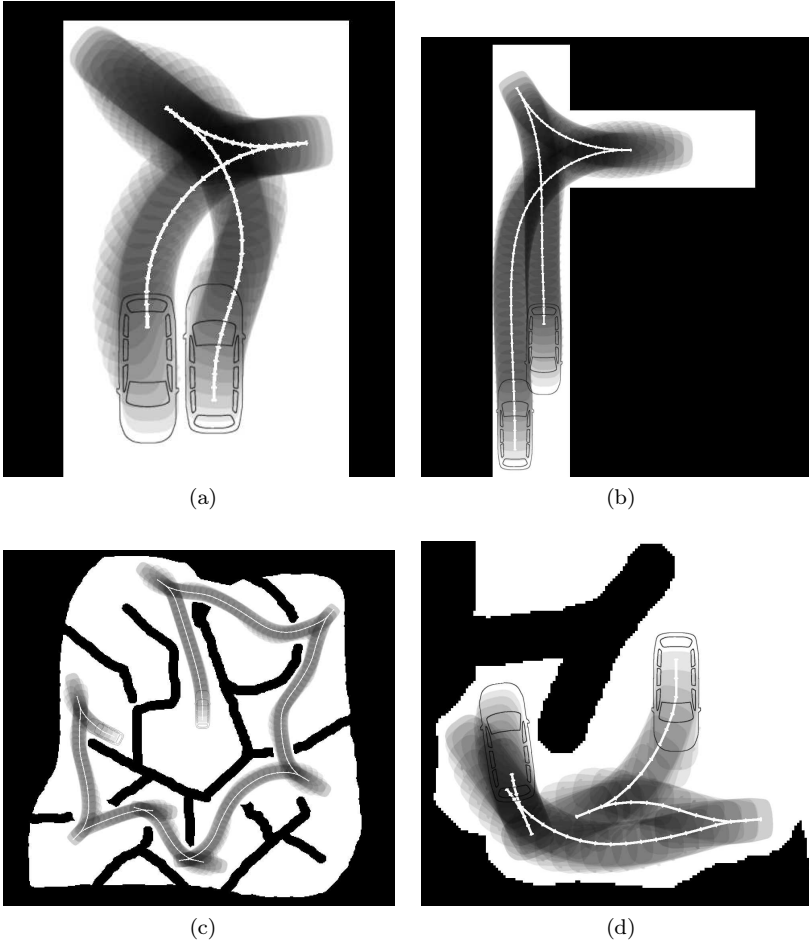


Abbildung 7.13: Wendemanöver



(a)



(b)



(c)

**Abbildung 7.14:** Einparken im realen Fahrversuch. (a) Benutzerschnittstelle mit Touchscreen und Drehsteller. (b) Beginn und (c) Abschluss des Manövers.

## 8 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurden Verfahren zur Bahn- und Trajektorienplanung entwickelt, und ihre Praxistauglichkeit durch reale Fahrversuche demonstriert. Von Beginn an wurden zwei unterschiedliche Klassen von Anwendungsfällen untersucht, die zusammen die meisten praktisch auftretenden Fahrsituationen abdecken: Das Fahren in unstrukturierter, statischer Umgebung, sowie das Fahren in einer dynamischen Umgebung, die durch Verkehrswege strukturiert ist. Die erste Klasse umfasst das Einparken und andere Anwendungsbereiche, die präzises Manövrieren in beengter Umgebung erfordern. Die zweite beinhaltet das Fahren im fließenden Straßenverkehr. Für beide Klassen konnte die Bewegungsplanung einheitlich als Variationsproblem formuliert werden. Zur Lösung des so formalisierten Problems wurden dann aber für die beiden Anwendungsklassen jeweils unterschiedliche Ansätze gewählt.

Für die Navigation in unstrukturierter, beengter Umgebung wurde das Prinzip der infinitesimalen Gitter entwickelt. Hierbei wird der Lösungsraum auf einen Graphen, und damit auf eine abzählbare Menge von Trajektorien, reduziert. In dieser Menge wurde die global optimale Trajektorie dann durch eine erschöpfende Graphsuche identifiziert. Das Verfahren ist – bis auf das sich durch die Diskretisierung ergebende Auflösungslimit – optimal und vollständig. Es eignet sich für den vorgesehenen Anwendungsfall ideal. Seine Grenzen liegen aber bei der Verallgemeinerbarkeit auf Kostenfunktionale und nichtholonome Nebenbedingungen, die höhere als die zweite Ableitung enthalten. Es wurde gezeigt, dass solche Kostenfunktionale und Nebenbedingungen höherer Ordnung notwendig sind, um die Planung dynamischer Fahrmanöver zu ermöglichen. Für das zweite Anwendungsszenario wurde deshalb der globale, diskrete Ansatz verworfen.

Für die Navigation in strukturierter, dynamischer Umgebung wurde stattdessen ein lokales Lösungsverfahren entworfen, das die Trajektorie, von einer Initiallösung ausgehend, schrittweise verbessert. Die Trajektorie wird hierzu durch eine endliche Menge an Stützpunkten repräsentiert. Die Koordinaten der Stützpunkte werden zu den Parametern eines nichtlinearen Op-

timierungsproblems, das, durch Hindernisse, kinematische und dynamische Restriktionen, Nebenbedingungen unterworfen ist. Es wird mit der Methode des sequentiellen quadratischen Programmierens gelöst. Bei einem solchen lokalen Verfahren besteht prinzipiell die Gefahr, dass es, ausgehend von der Initiallösung, in einem lokalen Optimum konvergiert. Durch sorgfältige Auslegung der Nebenbedingungen konnte dies vermieden werden. Die hierbei angewendeten Regeln setzen aber voraus, dass der mögliche Verlauf der Trajektorie von vornherein durch einen Fahrkorridor eingeschränkt ist. Diese Bedingung ist beim Fahren in einer strukturierten Umgebung gegeben. Der Ansatz erlaubt es, auch höhere Ableitungen der Trajektorie bei der Definition eines Gütekriteriums zu berücksichtigen. Für das Experiment „BERTHA-BENZ-Fahrt“ wurde die dritte Zeitableitung der Trajektorie – also der Ruck, dem das Fahrzeug und seine Passagiere ausgesetzt sind – mit in das Gütekriterium aufgenommen. So wird erreicht, dass die berechneten Trajektorien zweimal stetig differenzierbar sind. Durch diese Eigenschaft wird ruckartiges Lenken beim Fahren der Trajektorie vermieden. Die Trajektorien gewährleisten so einen hohen Fahrkomfort, sind aber insbesondere auch gut zu regeln.

Beide Verfahren wurden an Bord von unterschiedlich ausgestatteten Versuchsfahrzeugen zum praktischen Einsatz gebracht.

Für die Demonstration des Fahrens in statischer Umgebung wurde für die Umgebungserfassung auf einen Laserscanner zurückgegriffen, dessen Messdaten in ein Belegunggitter umgesetzt wurden. Als ein Flaschenhals bezüglich der Rechenzeit stellte sich hierbei die Kollisionsüberprüfung heraus, da diese – wegen des erschöpfenden Charakters der Graphsuche – für jeden möglichen Fahrzeugzustand durchgeführt werden muss. Für das Teilproblem der Kollisionsüberprüfung wurde deshalb ein Verfahren entwickelt, das sich optimierter Faltungsoperationen bedient.

Besonders anspruchsvoll war die Zielsetzung für die Demonstration des Fahrens in dynamischer, strukturierter Umgebung. Die historische BERTHA-BENZ-MEMORIAL-ROUTE von Mannheim nach Pforzheim sollte automatisch befahren werden. Dieses Ziel wurde im September 2013 erreicht. Das Projekt unterlag besonderen Randbedingungen bezüglich der sensorischen Ausstattung des Fahrzeuges. Es war nur mit video- und radarbasierter Sensorik ausgestattet. Bei der Umsetzung der Sensordaten in funktionale Nebenbedingungen für die Trajektorienoptimierung hat sich die in dieser Arbeit vorgeschlagene polygonale Repräsentation der Umgebung bewährt.

Ein mögliches Ziel für zukünftige Untersuchungen ist die gemeinsame Trajektorienplanung durch mehrere Agenten, um *kooperatives* Verhalten au-

tomatisierter Fahrzeuge zu ermöglichen. Eine weitere interessante Herausforderung für zukünftige Forschung ist die Entwicklung einer probabilistischen Umgebungsrepräsentation, die der Unsicherheit in den verwendeten Sensordaten Rechnung trägt.



# Literatur

- [Aig67] AIGNER, M. On the linegraph of a directed graph. *Mathematische Zeitschrift* Bd. 102 (1967), 56–61.
- [AMO93] AHUJA, R. K., MAGNANTI, T. L. und ORLIN, J. B. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.
- [Arm66] ARMIJO, L. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics* Bd. 16, Nr. 1 (1966), 1–3.
- [Bar72] BARTHOLOMEW-BIGGS, M. Constrained minimization using recursive quadratic programming. In: *Numerical Methods for Nonlinear Optimization*. Hrsg. von LOOTSMA, F. Academic Press, 1972, 411–428.
- [Bei70] BEINEKE, L. Characterization of derived graphs. *Journal of Combinatorial Theory* Bd. 9 (1970), 129–135.
- [BL93] BARRAQUAND, J. und LATOMBE, J.-C. Nonholonomic multi-body mobile robots: controllability and motion planning in the presence of obstacles. *Algorithmica* Bd. 10 (1993), 121–155.
- [BLS08] BUEHLER, M., LAGNEMMA, K. und SINGH, S., Hrsg. *Journal of Field Robotics* Bd. 25.Nr. (8-10) (2008): *Special Issue on DARPA Urban Challenge (Parts I-III)*.
- [BZS14a] BENDER, P., ZIEGLER, J. und STILLER, C. Lanelets: efficient map representation for autonomous driving. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. Juni 2014, 420–425.
- [BZS14b] BENDER, P., ZIEGLER, J. und STILLER, C. Lanelets: efficient map representation for autonomous driving. In: *Intelligent Vehicles Symposium*. IEEE. Dearborn, Michigan, USA, 2014.
- [Dan80] DANIELSSON, P. E. Euclidean distance mapping. *Comput. Graphics Image Processing* Bd. 14 (1980), 227–248.
- [Dia69] DIAL, R. Algorithm 360: shortest path forest with topological ordering. *Communications of the ACM* Bd. 12 (1969), 632–633.

- [DS13] DYM, C. und SHAMES, I. Solid mechanics: a variational approach, augmented edition. In: Springer New York, 2013. Kap. 2.
- [EK72] EDMONDS, J. und KARP, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM* Bd. 19, Nr. 2 (Apr. 1972), 248–264.
- [Els61] ELSGOLC, L. E. *Calculus of Variations*. Pergamon Press, 1961.
- [Fab+08] FABBRI, R., COSTA, L. F., TORELLI, J. C. und BRUNO, O. M. 2d euclidean distance transform algorithms: a comparative survey. *ACM Computing Surveys* Bd. 40, Nr. 1 (2008), 2:1–2:44.
- [Fle87] FLETCHER, R. *Practical methods of optimization*. A Wiley Interscience Publication. John Wiley & Sons, Chichester, New York, Brisbane, 1987.
- [Fra+13] FRANKE, U., PFEIFFER, D., RABE, C., KNÖPPEL, C., ENZWEILER, M., STEIN, F. und HERRTWICH, R. G. Making Bertha see. In: *IEEE ICCV Workshop Computer Vision for Autonomous Vehicles*. Sydney, Australia, Dez. 2013, 1–10.
- [FSR05] FUCHSHUMER, S., SCHLACHER, K. und RITTENSCHÖBER, T. Nonlinear vehicle dynamics control - a flatness based approach. In: *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*. Dez. 2005, 6492–6497.
- [GI83] GOLDFARB, D. und IDNANI, A. A numerically stable dual method for solving strictly convex quadratic programs. English. *Mathematical Programming* Bd. 27, Nr. 1 (1983), 1–33.
- [GPM89] GARCIA, C. E., PRETT, D. M. und MORARI, M. Model predictive control: theory and practice – a survey. *Automatica* Bd. 25, Nr. 3 (1989), 335–348.
- [How+08] HOWARD, T. M., GREEN, C. J., KELLY, A. und FERGUSON, D. State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. *Journal of Field Robotics* Bd. 25 (2008), 325–345.
- [Kam37] KAMM, W. Die entwicklung des kraftfahrzeugs. In: *Deutsches Museum. Abhandlungen und Berichte*. VDI-Verl., 1937.
- [Kav95] KAVRAKI, L. Computation of configuration-space obstacles using the fast fourier transform. *IEEE Transactions on Robotics and Automation* Bd. 11, Nr. 3 (1995), 408–413.

- [Lat+13] LATEGAHN, H., SCHREIBER, M., ZIEGLER, J. und STILLER, C. Urban localization with camera and inertial measurement unit. In: *Proc. IEEE Intelligent Vehicles Symposium (IV2013)*. Gold Coast, Australia, 2013, 719–724.
- [Lat91a] LATOMBE, J.-C. A fast path planner for a car-like indoor mobile robot. In: *AAAI*. Hrsg. von DEAN, T. L. und MCKEOWN, K. AAAI Press / The MIT Press, 1991, 659–665.
- [Lat91b] LATOMBE, J.-C. *Robot Motion Planning*. Kluwer, 1991.
- [LaV06] LAVALLE, S. M. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006.
- [LaV98] LAVALLE, S. M. *Rapidly-exploring random trees: A new tool for path planning*. Techn. Ber. Computer Science Dept., Iowa State University, 1998.
- [LK01] LAVALLE, S. M. und KUFFNER, J. Randomized kinodynamic planning. *The International Journal of Robotics Research* Bd. 20 (2001), 378–400.
- [LS14] LATEGAHN, H. und STILLER, C. Vision only localization (submitted). *IEEE Transactions on Intelligent Transportation Systems* Bd. 00, Nr. 00 (2014).
- [LSL98] LAUMOND, J.-P., SEKHAVAT, S. und LAMIRAUX, F. Guidelines in nonholonomic motion planning for mobile robots. In: *Robot Motion Planning and Control*. Hrsg. von LAUMOND, J.-P. Springer-Verlag, Berlin, 1998, 1–53.
- [LW79] LOZANO-PÉREZ, T. und WESLEY, M. A. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM* Bd. 22, Nr. 10 (Okt. 1979), 560–570.
- [McN+11] MCNAUGHTON, M., URMSON, C., DOLAN, J. M. und LEE, J.-W. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In: *ICRA*. IEEE, 2011, 4889–4895.
- [Men65] MENON, V. The isomorphism between graphs and their adjoint graphs. *Canadian Mathematical Bulletin* Bd. 8 (1965), 7–15.
- [MV98] MEYBERG, K. und VACHENAUER, P. Differentialgleichungen, Funktionentheorie, Fourier-Analysis, Variationsrechnung. In: *Höhere Mathematik*. Bd. 2. Springer-Lehrbuch. Springer, 1998.
- [NW06] NOCEDAL, J. und WRIGHT, S. J. *Numerical Optimization*. 2nd. Springer, New York, 2006.

- [OY85] O'DUNLAING, C. und YAP, C.-K. A "retraction" method for planning the motion of a disc. *J. Algorithms* Bd. 6, Nr. 1 (1985), 104–111.
- [Pac06] PACEJKA, H. B. *Tire and Vehicle Dynamics*. 2nd. Society of Automotive Engineers, Inc., 2006, 5.
- [Pho75] PHONG, B. T. Illumination for computer generated pictures. *Commun. ACM* Bd. 18, Nr. 6 (1975), 311–317.
- [PK05] PITVORAİKO, M. und KELLY, A. Efficient constrained path planning via search in state lattices. In: *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*. 2005.
- [PKK09] PIVTORAİKO, M., KNEPPER, R. A. und KELLY, A. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics* Bd. 26, Nr. 3 (2009), 308–333.
- [Pon+64] PONTRYAGIN, L. S., BOLTIANSKI, V. G., GAMKRELIDZE, R. V., MISHCHENKO, E. F. und BROWN, D. E. *The mathematical theory of optimal processes*. A Pergamon Press book. London, Paris, 1964.
- [RS91] REEDS, J. und SHEPP, R. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics* Bd. 145, Nr. 2 (1991), 144–154.
- [SKF13] SCHREIBER, M., KNOPPEL, C. und FRANKE, U. Laneloc: lane marking based localization using highly accurate maps. In: *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE. 2013, 449–454.
- [SMK12] ŞUCAN, I. A., MOLL, M. und KAVRAKI, L. E. The open motion planning library. *IEEE Robotics & Automation Magazine* Bd. 19 (Dez. 2012). <http://ompl.kavrakilab.org>, 72–82.
- [SR61] SESHU, S. und REED, M. B. *Linear Graphs and Electrical Networks*. Addison-Wesley Series in Engineering Sciences. Addison-Wesley, Reading, MA, 1961.
- [Suc+08] SUCAN, I., KRUSE, J., YIM, M. und KAVRAKI, L. Kinodynamic motion planning with hardware demonstrations. In: *International Conference on Intelligent Robots and Systems*. 2008.

- [Tak+89] TAKAHASHI, A., HONGO, T., NINOMIYA, Y. und SUGIMOTO, G. Local path planning and motion control for AGV in positioning. In: *International Workshop on Intelligent Robots and Systems*. 1989.
- [Tsi95] TSITSIKLIS, J. N. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control* Bd. 40 (1995), 1528–1538.
- [Vol96] VOLKMANN, L. *Fundamente der Graphentheorie*. Springer, Wien, New York, 1996.
- [WDS12] WINNER, H., DANNER, B. und STEINLE, J. *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. Vieweg+Teubner Verlag, Wiesbaden, 2012. Kap. Adaptive Cruise Control, 478–521.
- [Wer+10] WERLING, M., ZIEGLER, J., KAMMEL, S. und THRUN, S. Optimal trajectory generation for dynamic street scenarios in a Frenet frame. In: *IEEE International Conference on Robotics and Automation*. 2010, 987–993.
- [Wer12] WERLING, M. Ein neues Konzept für die Trajektoriengenerierung und -stabilisierung in zeitkritischen Verkehrsszenarien. *Automatisierungstechnik* Bd. 60, Nr. 1 (2012), 53–54.
- [Zie+14a] ZIEGLER, J., BENDER, P., LATEGAHN, H., SCHREIBER, M., STRAUSS, T., DANG, T. und STILLER, C. Kartengestütztes automatisiertes Fahren auf der Bertha-Benz-Route von Mannheim nach Pforzheim. In: *Workshop Fahrerassistenzsysteme*. UNiDAS e.V., Walting, Altmühltal, 2014.
- [Zie+14b] ZIEGLER, J., BENDER, P., DANG, T. und STILLER, C. Trajectory planning for Bertha – a local, continuous method. In: *Intelligent Vehicles Symposium*. IEEE. Dearborn, Michigan, USA, 2014.
- [Zie+14c] ZIEGLER, J., BENDER, P., SCHREIBER, M., LATEGAHN, H., STRAUSS, T., STILLER, C., DANG, T., FRANKE, U. u. a. Making Bertha drive - an autonomous journey on a historic route. *IEEE Intell. Transport. Syst. Mag.* Bd. 6, Nr. 2 (2014), 8–20.
- [Zie+14d] ZIEGLER, J., LATEGAHN, H., SCHREIBER, M., KELLER, C. G., KNÖPPEL, C., HIPPE, J., HAUEIS, M. und STILLER, C. Video based localization for Bertha. In: *Intelligent Vehicles Symposium*. IEEE. Dearborn, Michigan, USA, 2014.

- [ZP08] ZIEGLER, J. und PFITZER, B. Bahnplanung für das autonome Fahrzeug AnnieWAY. In: *Workshop Fahrerassistenzsysteme*. UNiDAS e.V., Walting, Altmühltal, 2008.
- [ZS09] ZIEGLER, J. und STILLER, C. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In: *International Conference on Intelligent Robots and Systems*. IEEE. 2009.
- [ZS10] ZIEGLER, J. und STILLER, C. Fast collision checking for intelligent vehicle motion planning. In: *Intelligent Vehicles Symposium*. IEEE. San Diego, CA, USA, 2010.
- [ZWS08] ZIEGLER, J., WERLING, M. und SCHRÖDER, J. Navigating car-like vehicles in unstructured environment. In: *Intelligent Vehicles Symposium*. IEEE. Eindhoven, 2008, 787–791.