INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Bewegungsplanung für nicht-holonome Vehikel

Bernhard Lorenz

Aufgabensteller: Prof. Dr. Hans Jürgen Ohlbach

Abgabetermin: 28. März 2002

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Bewegungsplanung für nicht-holonome Vehikel

Bernhard Lorenz

Aufgabensteller: Prof. Dr. Hans Jürgen Ohlbach

Abgabetermin: 28. März 2002

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 28. März 2002

.....
(Unterschrift des Kandidaten)

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der im weiteren Sinne als "Einparkproblem" bekannten Situation für autoähnliche Fahrzeuge. Das Einparkproblem ist in mehrerlei Hinsicht ein Spezialfall der gewöhnlichen Bewegungsplanung für Roboter, wie in dieser Ausarbeitung sichtbar wird.

Eine Abgrenzung zur allgemeinen Bewegunsplanung (siehe Kapitel 2.2) findet in erster Linie in Bezug auf die kinematischen Beschränkungen statt, die einem autoähnlichen Fahrzeug auferlegt sind. Unter kinematischen Beschränkungen versteht man diejenigen Beschränkungen, die es dem Fahrzeug verbieten, sich frei im Raum zu bewegen (Translation) oder zu drehen (Rotation). Eines der klassischen Beispiel hierfür ist ein autoähnlicher Roboter. Des weiteren gehen in die Bewegungsplanung Merkmale ein, welche die Umgebung und deren Wahrnehmung durch das Fahrzeug charakterisieren.

Die generelle Eignung klassischer Modelle zur Bewegungsplanung und deren mögliche Defizite werden in Kapitel 2 vorgestellt und anhand der Implementierung eines geeigneten Vertreters in Kapitel 3 weiter vertieft. Es handelt sich hierbei um ein Verfahren aus dem Bereich der Zell-Dekomposition von Jérôme Barraquand und Jean-Claude Latombe [BaLa 89].

Die Ergebnisse aus dieser Betrachtung sowie einige ausgewählte Beispiele von Ergebnissen aktueller Forschungsarbeiten werden anschließend in Kapitel 4 aufbereitet, und es wird ein Modell vorgestellt, welches die in der Aufgabenstellung genannten Anforderungen erfüllt. Ferner werden verfügbare Technologien im Hinblick auf Realisierungsmöglichkeiten untersucht. Abschließend erfolgt ein Ausblick auf zukünftige Probleme und Herausforderungen, die sich aus der Aufgabenstellung und artverwandten Themen ergeben.

Inhaltsverzeichnis

Inhaltsverzeichnis i

Abbildungsverzeichnis v

1. Einführung 1
- 1.1. Aufgabenstellung und Anforderungen . . . 1
- 1.2. Anforderungskatalog . . . 2
  - 1.2.1. Fahrzeug . . . 3
  - 1.2.2. Kinematik . . . 4
  - 1.2.3. Umwelt . . . 5
  - 1.2.4. Interaktion . . . 5

2. Grundlagen 7
- 2.1. Begriffsklärungen . . . 7
  - 2.1.1. Konfigurationsraum . . . 7
  - 2.1.2. Pfade . . . 8
  - 2.1.3. Hindernisse . . . 8
- 2.2. Kinematische Beschränkungen . . . 10
  - 2.2.1. Holonome Beschränkungen . . . 11
  - 2.2.2. Nicht-holonome Beschränkungen . . . 12
- 2.3. Kategorisierung . . . 15
- 2.4. Roadmap-Methoden . . . 15
- 2.5. Zell-Dekomposition . . . 17
  - 2.5.1. Exakte Zelldekomposition . . . 17
  - 2.5.2. Approximative Zelldekomposition . . . 18
  - 2.5.3. Vollständigkeit . . . 19
- 2.6. Potenzialfeldmethoden . . . 19
- 2.7. Reaktive sensorbasierte Bewegungsplaner . . . 21
- 2.8. Sonderfälle / Hybride Modelle . . . 21
  - 2.8.1. Bewegungsplaner für nicht-holonome Vehikel . . . 21
  - 2.8.2. Robot Work Crew . . . 22

3. Verfahren von Barraquand und Latombe 25
- 3.1. Beschreibung . . . 25
  - 3.1.1. Dekomposition des Konfigurationsraumes . . . 25
  - 3.1.2. Aufbau des Suchgraphen . . . 25
  - 3.1.3. Generierung der Nachfolger . . . 27
  - 3.1.4. Gewichtung . . . 28

3.1.5. Kennzeichnen des optimalen Pfades . . . 28
3.2. Komplexität . . . 28
3.3. Messergebnisse / Laufzeitverhalten . . . 29
3.4. Implementierung . . . 30
3.4.1. Hindernisbehandlung . . . 30
3.4.2. Kostenfunktion . . . 30
3.5. Analyse / Potenzial . . . 32
3.5.1. Prämissen . . . 33
3.5.2. Systeminhärente Vorarbeiten . . . 33
3.5.3. Präzisionsdilemma . . . 34

4. Hybridmodell: Route/Template **35**
4.1. Grundidee . . . 35
4.1.1. Vorgegebene Teilstrecken . . . 35
4.1.2. Manövrieren im unmittelbaren Umfeld / Einparken . . . 36
4.2. Charakterisierung von Konfigurationen . . . 37
4.2.1. Relative und absolute Referenzen . . . 37
4.2.2. Approximative Referenzen . . . 38
4.3. Distanzfunktionen . . . 40
4.3.1. Untere Grenze aller Distanzfunktionen . . . 40
4.3.2. Euklidische Distanz . . . 40
4.3.3. Laterale Distanzfunktionen . . . 41
4.3.4. Single-Ident Eigenschaft . . . 43
4.3.5. Double-Overlap Eigenschaft . . . 44
4.4. Modellierung des Routensystems . . . 45
4.5. Modellierung des Templatesystems ohne Kollisionsbehandlung . . . 45
4.5.1. Templates . . . 46
4.5.2. Bewegungsgeometrie . . . 47
4.5.3. Atomare Bewegungen . . . 48
4.5.4. Generierung / Algorithmus . . . 48
4.6. Modellierung der Umfelderkennung . . . 52
4.6.1. Technische Grundlagen . . . 52
4.6.2. Leistungsbeschreibung der Sensorik . . . 53
4.7. Modellierung der Hindernisbehandlung . . . 54
4.7.1. Hindernisse . . . 54
4.7.2. Kollisionsgeometrie . . . 55
4.7.3. Freiheitsgrade . . . 55
4.8. Modellierung des Templatesystems mit Kollisionsbehandlung . . . 56
4.8.1. Befreien von Konfigurationen . . . 56
4.8.2. Generierung von Interimskonfigurationen . . . 58
4.8.3. Generierung des Pfades . . . 59
4.8.4. Behandlung statischer Hindernisse . . . 60
4.8.5. Behandlung dynamischer Hindernisse . . . 61
4.9. Erfüllung der Aufgabenstellung . . . 62

5. Technische Basis **65**
5.1. Grundlagen der Sensorik 65
5.1.1. Konstruktive Aspekte 65
5.1.2. Rechtliche Aspekte 66
5.1.3. Weitere Aspekte 67
5.2. Stand der Technik 68
5.2.1. Parkassistenten 68
5.2.2. Praxiserfahrungen PDC 69
5.2.3. Zukünftige Entwicklungen 71
5.3. PATH Magnetic Guidance System 72

6. Ausblick **75**

A. Auszüge aus dem Sourcecode **77**
A.1. Configuration.h 77
A.2. ConfigSpace.h 78
A.3. Vehicle.h 80
A.4. ConfigSpace.cpp 81
A.5. Vehicle.cpp 84

Literaturverzeichnis **87**

Abbildungsverzeichnis

1.1. Einparkproblem 2
1.2. Autoähnlicher Roboter 3

2.1. Expansion eines Hindernisses 9
2.2. Roboterarm mit je einem Dreh- und Schiebegelenk 11
2.3. Geschwindigkeits-Subraum eines Roboters 13
2.4. Geschwindigkeitsvektorraum einer Konfiguration 14
2.5. Sichtbarkeitsgraph 16
2.6. Voronoi Diagramm 16
2.7. Exakte Zelldekomposition 18
2.8. Approximative Zelldekomposition 19
2.9. Schemata der Potentialfeldmethode 20

3.1. Einfache Problemstellungen 29
3.2. Hinderniskonstellationen / Paralleles Einparken 29

4.1. Fall 1: paralleles Einparken 36
4.2. Fall 2 und 3: rechtwinkliges Einparken (links vorw., rechts rückw.) 37
4.3. Abstandsinformation der Sensorik 38
4.4. Schema paralleles Einparken 39
4.5. Effizienz verschiedener Manöver im Fall lateraler Translation 41
4.6. Manöver A (links) und Manöver C 42
4.7. Manöver vom Typ 1 (links) und Typ 2 46
4.8. Bewegungsgeometrie des Roboters 47
4.9. Bezugspunkte zweier Konfigurationen 49
4.10. Illustration des Algorithmus (Beispiel für $d(\Omega, \Omega') > 2\rho_{min}$) 51
4.11. Illustration des Algorithmus (Beispiel für $d(\Omega, \Omega') < 2\rho_{min}$) 52
4.12. Echo der Sensoren (radial / angular) 54
4.13. Kollisionsgeometrie des Roboters 55
4.14. Konfiguration ohne signifikante Freiheitsgrade 57
4.15. Sukzessive Verbesserung der Freiheitsgrade 58
4.16. Illustration der Interimskonfigurationen 59
4.17. Statisches Hindernis 60
4.18. Simuliertes Sensorecho 61
4.19. Mögliche Interimskonfiguration zur Umgehung 61

5.1. Nomad 200c, Nomadic Inc. 66
5.2. Sensor für Parkassistent, Valeo 68
5.3. Erfassungswinkel und Reichweite des Sensors 69

5.4. Grafische Ausgabe der PDC 70
5.5. Lage der Sensoren, BMW 7er (E65) 70
5.6. Zukünftige Anwendungen externer Sensorik 71
5.7. PATH Magnetic Guidance System 73
5.8. Magnetometer und Stabmagnet 73

1. Einführung

Im Bereich der Robotertechnik wird seit geraumer Zeit vorrangig ein besonderes Ziel verfolgt: die Schaffung von Robotern, die eine gegebene Problemstellung *autonom* lösen. Dies soll durch die Übermittlung der eigentlichen Problemstellung erfolgen, also insbesondere indem die autonomen Akteure geeignete Aktionen selbst bestimmen. Eine genaue Spezifikation dessen, *wie* dies zu bewerkstelligen ist, soll nicht erforderlich sein.

Anwedungen für autonome Roboter finden sich in vielen Bereichen verschiedener Branchen, beispielsweise in der Produktion, Konstruktion und Wartung bei verschiedenen industriellen Prozessen, aber auch in Medizin und Umwelttechnik und generell in allen Bereichen, die durch eine "feindliche" Umwelt den Einsatz von Maschinen als Alternative zum Menschen favorisieren: Raumfahrt, Bergbau und unter Wasser.

In Kontrast zur komplexen Vielschichtigkeit der Probleme, die im Zusammenhang mit Bewegungsplanung betrachtet werden müssen, lässt sich die grundlegende Problemstellung sehr allgemein formulieren: Wie kann ein Roboter entscheiden, welche Bewegungen ausgeführt werden müssen, um eine bestimmte Zielanordnung physikalischer Objekte zu erreichen? Es ist nicht sofort offensichtlich, dass im Zuge dieser Entscheidungsfindung verschiedenste Probleme aus den Bereichen der Logik, Wahrnehmung, Steuerung und Überwachung zu lösen sind. Dies liegt nicht zuletzt daran, dass die Physionomie und Anatomie des Menschen solche Probleme anscheinend ohne jegliche Anstrengung intuitiv und augenblicklich bewältigt.

Forschungen auf dem Gebiet der Bewegungsplanung werden in etwa seit den sechziger Jahren des vorigen Jahrhunderts betrieben, wobei der größere Teil der heute verwendeten Erkenntnisse aus Arbeiten stammt, die zurück bis in die achtziger Jahre gehen. In einer groben Gliederung können die verschiedenen Ansätze zur Problemlösung in mehrere Bereiche eingeteilt werden (vgl. Kapitel 2):

- Geometrisch-modellbasierte Ansätze:
  - Roadmap-Methoden (vgl. 2.4)
  - Potenzialfeld-Methoden (vgl. 2.6)
  - Verfahren auf Basis von Zell-Dekomposition (vgl. 2.5)
- Reaktive, sensorbasierte Ansätze (vgl. 2.7)

Bevor auf die aktuell vorhandenen Methoden und deren Eignung zur Problemlösung eingegangen wird, erfolgt eine Definition der Aufgabenstellung.

1.1. Aufgabenstellung und Anforderungen

Die zu lösende Aufgabe besteht darin, ein autoähnliches Fahrzeug (gemäß Definition in 1.2.1) von einer Anfangs- in eine Zielposition zu manövrieren. Ein anschauliches Beispiel ist in

Abbildung 1.1 (vgl. [Lato 91], S. 434) dargestellt. Vereinfacht ausgedrückt geht es um das "Einparken" des Fahrzeugs in eine dafür geeignete Parklücke aus einer bestimmten oder unbestimmten Anfangsposition.

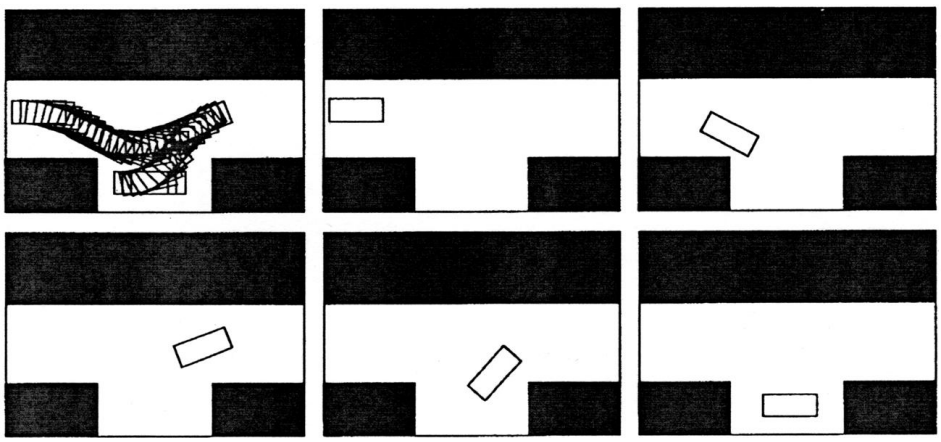

Abbildung 1.1.: Einparkproblem

Vor dem Hintergrund des dargestellten Problems werden Methoden, Modelle und Algorithmen betrachtet, welche geeignet sind, die verschiedenen Vorgänge zu überwachen und gegebenenfalls unterstützend einzugreifen. Unabhängig von den konkreten Anforderungen jeweiliger Konstellationen kommen hier zusätzlich verschiedene Grade an Interaktion zum Tragen. Angefangen von einer lediglich subsidiären Funktion durch die Überwachung verschiedener Parameter und somit Möglichkeit zur Darstellung von Referenzgrößen (im akuten Fall zur Auslösung von Warnsignalen), bis hin zur vollautomatisierten Steuerung der Bewegunsmanöver. Diese finden wiederum statt in einem durch verschiedene Grade an Dynamik geprägten Umfeld.

In der vorliegenden Arbeit werden klassische Ansätze zur Lösung des Problems untersucht und spezifische Vor- und Nachteile erfasst. Ferner wird ein effizientes Modell konstruiert, dessen Auslegung eine Kombination verschiedener Methoden darstellt. Die konkrete Zielsetzung liegt in der Maximierung spezifischer Vorteile sowie der Minimierung der Nachteile.

Zunächst wird ein Anforderungskatalog aufgestellt, welcher Zusammenhänge und Parameter charakterisiert und Schlüsselfunktionen zur jeweiligen Zielerreichung aufzeigt.

1.2. Anforderungskatalog

Der Anforderungskatalog definiert die beiden zentralen Komponenten, deren spezifische Beschaffenheit und Eigenschaften grundlegende Einflüsse auf die Bewegungsplanung haben: Der Roboter bzw. das Fahrzeug und die Umwelt, in der die Bewegung stattfindet.

Hinsichtlich einer anschaulichen Darstellung wird die Beschreibung des Roboters in zwei Abschnitte gegliedert, die die statischen von den dynamischen Aspekten trennt. Abschnitt 1.2.1 beschreibt die physikalischen Eigenschaften des Roboters und Abschnitt 1.2.2 die dynamischen, fahrphysikalischen Eigenschaften, welche daraus resultieren.

Die Klassifizierung umweltspezifischer Parameter und deren Übertragung in ein geeignetes Modell erfolgt in Abschnitt 1.2.3. Ziel ist hierbei, ein ausgewogenes Modell zu schaffen, welches eine hinreichende Genauigkeit besitzt und dennoch nicht zu komplex ist. Maßgeblich ist

dabei eine gute Handhabbarkeit und, daraus resultierend, eine für verschiedene Anwendungen ausreichende Effizienz der darauf aufsetzenden Planungsmodelle.

1.2.1. Fahrzeug

Das betrachtete Objekt ist ein autoähnliches Fahrzeug, dessen spezifische Eigenschaften und Kinematik anhand der folgenden Paramter definiert sind (vgl Abb. 1.2):

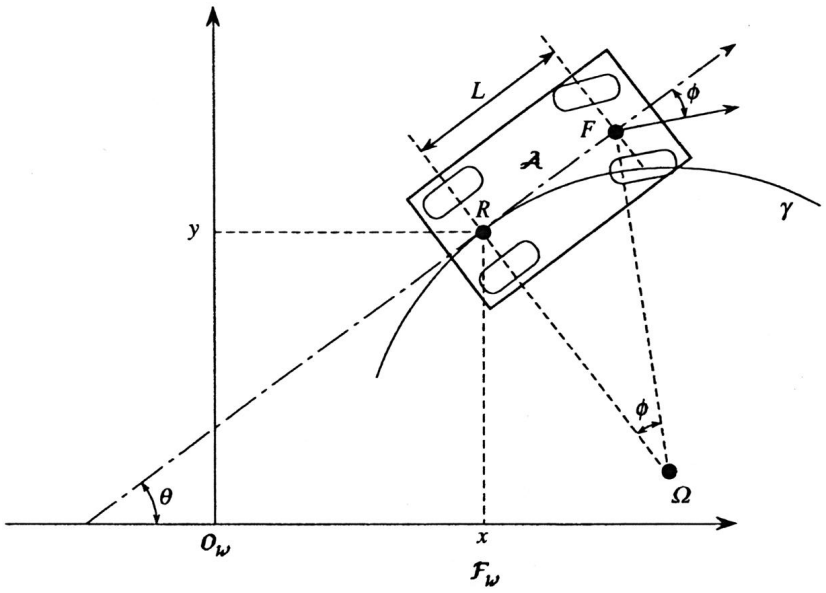

Abbildung 1.2.: Autoähnlicher Roboter

- Der **Bezugspunkt** R ist der Punkt zwischen den beiden Hinterrädern, bzw. der Schnittpunkt von Hinterachse und Fahrzeuglängsachse.
- Der **Radstand** L ist der Abstand zwischen R und dem Schnittpunkt F zwischen der Vorderachse und der Fahrzeuglängsachse.
- Der **maximale Lenkwinkel** ϕ ist die Hälfte der Summe der Vorspurwinkel der Vorderräder, bezogen auf die Längsachse des Fahrzeugs[1].
- Die **Länge des Überhangs vorne** L_f ist der Abstand zwischen dem Schnittpunkt zwischen der Vorderachse und der Fahrzeuglängsachse und der senkrecht zur Fahrzeuglängsachse verlaufenden Ebene, welche den äußersten vorderen Punkt des Fahrzeugs schneidet.
- Die **Länge des Überhangs hinten** L_r ist der Abstand zwischen R und der senkrecht zur Fahrzeuglängsachse verlaufenden Ebene, welche den äußersten hinteren Punkt des Fahrzeugs schneidet.

[1] Im Maschinenbau ist der maximale Lenkwinkel definiert als der Vorspurwinkel des *kurveninneren* Vorderrades; dieser ist geringfügig größer, als der des *kurvenäußeren* Vorderrades. Da im Zusammenhang mit der Bewegungsplanung nur der für das gesamte Fahrzeug resultierende, mittlere Lenkwinkel beider Vorderräder entscheidend ist, ist diese Differenzierung vernachlässigbar.

Anhand dieser Angaben können die folgenden Parameter berechnet werden, welche im weiteren Verlauf benötigt werden:

- Der **Radius** ρ_{min} des Kreises γ um Ω beträgt:

$$\rho_{min} = \frac{L}{\tan(\phi)}$$

- Die **Bezugspunkte** Ω_p und Ω_s (vgl. Abb. 4.8, S. 47) lauten für $R = (x, y)$:

$$\Omega_s = (x - \sin(\phi)\rho_{min}, y - \cos(\phi)\rho_{min})$$

und

$$\Omega_p = (x + \sin(\phi)\rho_{min}, y + \cos(\phi)\rho_{min})$$

Hinzu kommen diejenigen Parameter, die die Konfiguration des Roboters bestimmen, und nicht von der Form des Roboters selbst, oder von eventuell vorhandenen nicht-holonomen Beschränkungen abhängen:

- x, die Position von R auf der X-Achse
- y, die Position von R auf der Y-Achse
- θ, den Winkel der Fahrzeuglängsachse zur X-Achse

Diese Parameter genügen nicht nur den Anforderungen bezüglich der Bewegunsplanung, sondern insbesondere der Kollisionserkennung und der Behandlung der kinematischen Beschränkungen.

Bezüglich des **Wendekreises** eines Fahrzeugs haben sich in der Praxis verschiedene Definitionen gleichermaßen durchgesetzt. Bisweilen ist mit diesem Begriff der Kreis gemeint, welcher durch das innere Hinterrad beschrieben wird. An anderer Stelle ist derjenige Kreis gemeint, welchen der Punkt R beschreibt, oder der des kurvenäußeren Vorderrades.

Da für die vorliegende Problemstellung dieser Begriff vornehmlich im Zuge der Kollisionserkennung wichtig ist, wird der Wendekreis definiert als der Kreis den das *gesamten Fahrzeug bei vollem Lenkeinschlag* maximal beschreibt[2].

1.2.2. Kinematik

Wie später gezeigt wird, ist ein autoähnliches Fahrzeug nicht zur holonomen Bewegung in der Lage. Deshalb sind so genannte nicht-holonome Beschränkungen zu beachten, welche weitreichenden Einfluss auf Art und Umfang der Bewegungsplanung haben. Auf Grund der Komplexität der gesamten Thematik ist der genaueren Betrachtung ein kompletter Abschnitt im Kapitel der Grundlagen gewidmet. Eine Diskussion der kinematischen Beschränkungen allgemein findet sich in Abschnitt 2.2, die der nicht-holonomen Beschränkungen in Abschnitt 2.2.2.

Wichtigster Aspekt im Hinblick auf das Bewegungsplanungsmodell ist die Möglichkeit solche Beschränkungen behandeln zu können. Einerseits kann dies bedeuten, dass im Voraus ein geeigneter Ansatz gewählt werden muss. Andererseits sind Varianten denkbar, welche

[2] Vgl. hierzu Abb. 4.13 auf Seite 55: Wendekreis links ist der Kreis mit Radius ρ_o um Ω_p.

zunächst eines der klassischen Modelle anwenden und anschließend eine geeignete Anpassung vornehmen, so dass das Ergebnis, also der gefundene Pfad, ebenfalls den (nicht-holonomen) Beschränkungen genügt.

Das angestrebte Modell ist ein Vertreter der ersten Kategorie. Die Konsequenzen der kinematischen Beschränkungen sind von Anfang an substanzieller Bestandteil der Planung.

1.2.3. Umwelt

Offensichtlich sind Informationen unverzichtbar, welche die Umwelt beschreiben, in der die Bewegung erfolgen soll. Auch wenn in einem experimentellen Umfeld keinerlei Hindernisse unterstellt werden, so gelten doch gewisse Grundannahmen über die Bewegungsmöglichkeiten des Roboters in dieser Umwelt.

Bei einer Bewegungsplanung sind generell Hindernisse mit einzubeziehen, welche den freien Raum für die Bewegungen des Roboters einschränken können, und den Raum insgesamt begrenzen. Im vorliegenden Fall kann vereinfachend davon ausgegangen werden, dass Hindernisse einen binären Charakter[3] haben, und zweidimensional behandelt werden können - zumindest aber in eine zweidimensionale Struktur (d.h. Rechteck, Polygon, etc.) überführt werden können. Dieses Vorgehen mag auf den ersten Blick wenig präzise wirken, genügt den gestellten Anforderungen jedoch voll und ganz. Bestrebungen zur differenzierteren Behandlung von Hindernissen sind beipielsweise in [Laub 99] zu finden.

Eine lediglich statische Sicht der Hindernisse ist allerdings nicht ausreichend. Ein für den öffentlichen Straßenverkehr gedachtes System muss in der Lage sein, dynamisch auftretende und mobile Hindernisse zu erfassen bzw. zu erkennen und entsprechend auf diese zu reagieren. Auch hier reicht eine binäre, zweidimensionale Handhabung aus, jedoch verleiht die dynamische Komponente dem Problem eine zusätzliche Dimension.

Zusammenfassend muss das Planungssystem also folgende Bedingungen hinsichtlich der Umwelt erfüllen:

- Es kann **zu keinem Zeitpunkt** innerhalb des Planungs- bzw. des Ausführungshorizontes **von vollkommener Information** ausgegangen werden.
- **Dynamisch auftretende** und **sich bewegende Hindernisse** müssen erfasst und verarbeitet werden.
- **Zu jedem Zeitpunkt** müssen **Änderungen am Bewegungsplan** bzw. der Bewegungsstrategie möglich sein.

1.2.4. Interaktion

Des Weiteren sind folgende Eigenschaften wünschenswert, welche die Interaktion des Fahrers mit dem System betreffen:

- Das System muss sowohl einen **vollautomatisierten** (autonomen), als auch einen **subsidiären**, d.h. unterstüzenden **Modus** kennen.

[3] Der dem Straßenverkehr zu Grunde liegende Hindernisbegriff hat - einer Betrachtung der Draufsicht gemäß - einen stark binären Charakter und ist nicht geprägt durch mehr oder weniger überfahrbare Hindernisse, wie dies beispielsweise beim Fahren in schwerem Gelände oder bei sonst fehlender Infrastruktur der Fall wäre (vgl. hierzu [LaBu 99]).

- Eine **fortlaufende Fehlererkennung** muss Abweichungen vom Sollwert erkennen und gegebenenfalls geeignete Maßnahmen auslösen.
- Das System muss insgesamt **echtzeittauglich** sein.

Diese Anforderungen zusammengenommen bilden ein Pflichtenheft, dessen Erfüllung klassische Bewegungsplanungsverfahren in den meisten Fällen nicht gewährleisten. Oft ist allein die Unterstellung unvollkommener Information oder die Forderung dynamischer Komponenten ein ausschließendes Kriterium. Das folgende Kapitel behandelt diese klassischen Methoden und gibt Einblick in die Anwendungsmöglichkeiten und den jeweiligen Grad der Eignung.

2. Grundlagen

Dieser Abschnitt liefert sowohl einen Überblick über bisherige Arbeiten auf dem Gebiet der Bewegungsplanung, als auch eine Betrachtung einiger Beispiele der aktuellen Forschung, welche für die vorliende Arbeit relevant sind.

Zunächst wird eine gemeinsame begriffliche Grundlage geschaffen. Im Anschluss daran erfolgt eine grobe Kategorisierung etablierter Verfahren in mehrere Bereiche und eine Betrachtung einiger signifikanter Vertreter und deren unterschiedlicher Merkmale und Besonderheiten. Insbesondere liegt der Schwerpunkt auf den klassischen Verfahren: Roadmap-Methoden, Potenzialfeldmethoden und Methoden der Zell-Dekomposition. Ferner wird die Relevanz von sensorbasierten Verfahren aufgezeigt und Sonderfälle bzw. hybride Modelle vorgestellt.

Die folgenden Abschnitte enthalten eine Zusammenfassung der jeweiligen Thematik. Eine ausführlichere Darstellung ist in [Lato 91], Kapitel 5-7 und Kapitel 9 zu finden, an deren Aufbau sich auch die folgenden Ausführungen orientieren. Für tiefergehende Recherche, besonders im Hinblick auf detaillierte Teilaspekte und -probleme der Bewegungsplanung, empfiehlt sich [GoO' 97].

2.1. Begriffsklärungen

Um die verschiedenen dargestellten Methoden verstehen und miteinander vergleichen zu können, müssen zunächst einige Begriffe geklärt werden. In den folgenden Abschnitten werden die notwendigen Begriffsklärungen vorgenommen und ein gemeinsamer Nenner geschaffen.

Besonders hervor zu heben ist der im Folgenden erläuterte Begriff des Konfigurationsraumes. Ursprünglich von Lozano-Pérez [LP 83] hervorgebracht, hat sich dieses Konzept als ein wichtiges Werkzeug zur Beschreibung in der Robotik etabliert, wie im folgenden Abschnitt ersichtlich wird.

Die zu Grunde liegende Idee des Konfigurationsraumes ist, einen Roboter als Punkt in einem adäquaten Raum - dem Konfigurationsraum des Roboters - darzustellen, und eventuell vorhandene Hindernisse in diesen zu übertragen. Diese Übertragung dient der Umformulierung des Problems: Die Bewegungsplanung für ein "dimensionales" Objekt wird umgeformt in die Bewegungsplanung für einen Punkt. Dies erleichtert den Umgang mit den Beschränkungen, welche für eine Bewegung des Roboters gelten.

Ein weiterer Begriff verdient im Hinblick auf die Besonderheiten nicht-holonomer Roboter spezielle Beachtung. Es handelt sich hierbei um den Begriff der Distanzfunktion, welcher später in diesem Abschnitt erörtert wird.

2.1.1. Konfigurationsraum

Sei ein Roboter A (an einer bestimmten Position und mit einer bestimmten Orientierung) beschrieben als eine kompakte (d.h. geschlossene und begrenzte) Teilmenge vom Workspace

$W = R^N$, $N = 2$ oder 3, und die Hindernisse $B_1, \ldots, B_q$ seien ebensolche kompakten Teilmengen von W. Des Weiteren seien F_A und F_W kartesische Koordinatensysteme, eingebettet in A bzw. W, wobei F_A beweglich und F_W feststehend ist. Entsprechend dieser Definition hat ein Punkt a aus A bezüglich F_A eine festgelegte Position. Die Postition von a in W hängt jedoch ab von der Position und der Orientierung von F_A bezüglich F_W. Solange alle B_i unveränderlich und unbeweglich in W sind, hat auch jeder Punkt b aus B_i, für alle $i \in [1, q]$, eine feststehende Postition bezüglich F_W.

Die Konfiguration eines beliebigen Objektes ist nun eine Spezifikation der Position jedes Punktes dieses Objektes in Relation zu einem feststehenden Rahmen [Arno 78]. Deshalb ist eine **Konfiguration** q von A eine Spezifikation der Position T und Orientierung Θ von F_A bezüglich F_W. Der **Konfigurationsraum** von A ist dann der Raum C aller Konfigurationen von A. Die Teilmenge von W, welche durch A an einer Konfiguration q eingenommen wird, bezeichnet man als $A(q)$. Gleichermaßen bezeichnet man den Punkt a in A an einer Konfiguration q als $a(q)$ in W.

Die Beschreibung einer Konfiguration in Form einer Liste m unabhängiger Parameter, mit $m = 3$ (falls $N = 2$) und $m = 6$ (falls $N = 3$), erfolgt analog der Bezeichnung $C = R^m$. Mit m bezeichnet man die **Dimension** von C.

2.1.2. Pfade

Der Begriff *Kontinuität* ist von grundlegender Wichtigkeit bei der Definition eines Pfades. Um eine Formalisierung vorzunehmen, muss eine Topologie definiert werden. Eine klassische Möglichkeit dies zu tun, ist die Spezifikation einer Distanzfunktion $d : C \times C \to R$. Dann sei die von d induzierte, metrische Topologie die Topologie in C. Übertragen auf die Wahrnehmung der Umwelt, sollte die Entfernung zwischen zwei Konfigurationen q und q' abnehmen und gegen 0 tendieren solange die Regionen $A(q)$ und $A(q')$ sich aneinander annähern und zur Deckung gelangen. Eine einfache Distanzfunktion für diese Bedingungen wäre:

$$d(q, q') = \max_{a \in A} ||a(q) - a(q')||,$$

wobei $||x - x'||$ die euklidische Distanz zwischen zwei beliebigen Punkten x und x' in R^N ist.

Ein **Pfad** von A von der Konfiguration q_{init} zur Konfiguration q_{goal} ist eine kontinuierliche Abbildung:

$$\tau : [0, 1] \to C,$$

mit

$$\tau(0) = q_{init} \text{ und } \tau(1) = q_{goal},$$

wobei q_{init} bzw. q_{goal} die Start- bzw. Zielkonfiguration des Pfades ist. Die Kontinuität von τ hat zur Folge, dass für alle $s_0 \in [0, 1]$ gilt, $\lim_{s \to s_0} \max_{a \in A} ||a(\tau(s)) - a(\tau(s_0))|| = 0$, mit $s \in [0, 1]$. Falls keine Hindernisse vorhanden sind und sofern A im Raum frei beweglich ist, ist jeder so definierte Pfad ein realisierbarer freier Pfad[1].

2.1.3. Hindernisse

Sobald Hindernisse vorhanden sind, reicht die vorangegangene Definition eines Pfades nicht mehr aus. Im Folgenden werden Mengen von Teilpfaden betrachtet, die bei Vorhandensein

[1] Engl.: feasible free path; d.h. der Pfad ist frei von Hindernissen und er ist konform der Anorderungen konstruiert.

von Hindernissen die gewünschte Lösung ermöglichen. Zu diesem Zweck werden Hindernisse in den Konfigurationsraum abgebildet und der verbleibende, freie Teil von C betrachtet, welcher nicht innerhalb der Hindernisse liegt.
Jedes Hindernis B_i, $i = 1, \ldots, n$, im Workspace W wird auf eine Region in C abgebildet:

$$CB_i = \{n \in C/A(n) \cap B_i \neq 0\}.$$

Dies wird als C-**Hindernis**[2] bzw. C-**Obstacle** bezeichnet. Die Menge aller C-Obstacles:

$$\bigcup_{i=1}^{n} CB_i$$

bezeichnet man als C-**Obstacle region**, und die Menge:

$$C_{free} = C \setminus \bigcup_{i=1}^{n} CB_i = \{n \in C/A(n) \cap (\bigcup_{i=1}^{n} B_i) = 0\}$$

bezeichnet man als **freien Raum**. Jede beliebige Konfiguration aus C_{free} ist eine **freie Konfiguration**.

Ein **freier Pfad** zwischen zwei freien Konfigurationen q_{init} und q_{goal} ist eine kontinuierliche Abbildung $\tau : [0,1] \rightarrow C_{free}$, mit $\tau(0) = q_{init}$ und $\tau(1) = q_{goal}$. Zwei Konfigurationen gehören dann und nur dann demselben verbundenen Segment von C_{free} an, wenn sie mit einem freien Pfad verbunden werden können.

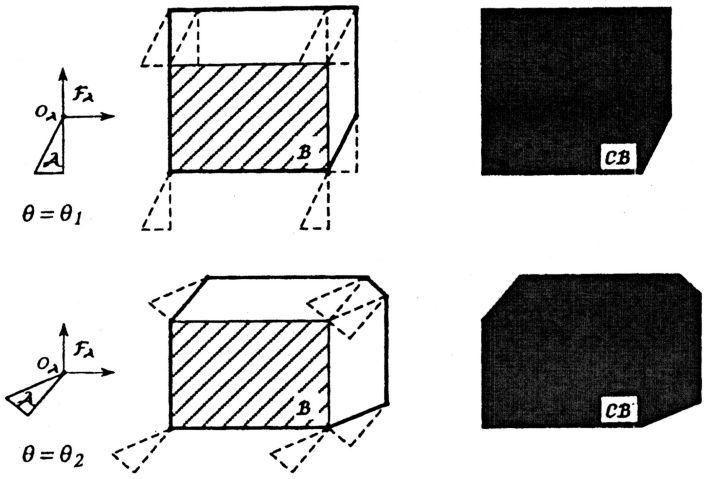

Abbildung 2.1.: Expansion eines Hindernisses

Besteht A lediglich aus einem Punkt, so hat es für gewöhnlich keine Bedeutung für die Hindernisbehandlung, welche Orientierung von A vorliegt. Dann ist, wie auch W, der Konfigurationsraum von A eine Kopie von R^N und daher ein euklidischer Raum. Die C-Obstacles sind zu den Hindernissen in W identisch.

Ist die Form von A eine andere, d.h. A ist ein "dimensionales Objekt", und die Konfigurationen von A unterliegen keinen besonderen Beschränkungen, so ist C immer noch in R^N. Die

[2]Engl.: Obstacle.

C-Obstacles haben jedoch dann eine andere Form. Wie in Abbildung 2.1 illustriert, entstehen sie durch die Erweiterung der Hindernisse in W mit der (negierten) Form von A. Dargestellt ist das Hindernis B und zwei C-Obstacles, entsprechend zweier Winkel θ_1 und θ_2 von A.

2.2. Kinematische Beschränkungen

Gelten für einen Roboter kinematische Beschränkungen, hat dies erheblichen Einfluss auf das Problem der Bewegungsplanung[3].

Kinematische Beschränkungen treten in verschiedenen Formen auf. Ein Roboterarm, dessen einzelne Glieder durch Gelenke miteinander verbunden sind, ist ein Beispiel hierfür, da sich die einzelnen Glieder nur auf eine bestimmte Art und Weise relativ zueinander bewegen können. Gleiches gilt für einen autoähnlichen Roboter, welcher zwar vorwärts bzw. rückwärts und Kurven fahren kann, jedoch nicht seitwärts (keine freie Translation) und dessen minimaler Kurvenradius auf Grund mechanischer Elemente begrenzt ist. Nichtsdestotrotz kann ein solcher Roboter eine beliebige Position und Orientierung in der Ebene einnehmen.

Diese beiden Beispiele illustrieren die zwei verschiedenen Arten kinematischer[4] Beschränkungen, die die Bewegung eines Roboters beeinflussen. Im Falle des Roboterarmes können diese Beschränkungen durch Gleichungen ausgedrückt werden, die die einzelnen Parameter einer Konfiguration in Relation setzen. Diese Gleichungen können dann dazu verwendet werden, verschiedene Parameter zu eliminieren und auf diese Weise die Dimension des Konfigurationsraumes zu reduzieren. Man nennt sie **holonome Beschränkungen**. Im Falle des autoähnlichen Roboters führen die Beschränkungen zu einer zusätzlichen Ungleichung. Diese Ungleichung beinhaltet Parameter die von den Geschwindigkeitsvektoren des Roboters abhängen und nicht zu einer Reduktion der Dimension des Konfigurationsraumes, sondern des Geschwindigkeitsraumes führen. Diese nennt man **nicht-holonome Beschränkungen**.

Während holonome Beschränkungen das Problem der Bewegungsplanung nicht grundlegend ändern, sind nicht-holonome Beschränkungen wesentlich schwieriger zu handhaben und werfen folgende Fragestellungen auf:

- Sei eine möglicherweise nicht-holonome Beschränkung gegeben. Wie kann festgestellt werden, dass es sich tatsächlich um eine *nicht-holonome* Beschränkung handelt? Falls sie integrierbar ist, dann reduziert sich eine Gleichung, die unter anderem Geschwindigkeitsparameter beinhaltet, auf eine holonome Beschränkung.
- Ist eine potenziell nicht-holonome Beschränkung als solche identifiziert, muss als nächstes entschieden werden können, ob diese nicht-holonome Beschränkung einen Einfluss auf die Menge der, von einer gegebenen Konfiguration aus gesehen, erreichbaren Konfigurationen hat.
- Wie könnte eine effektive Bewegungsplanung, unter Berücksichtigung der Gesamtheit der vorliegenden Einflüsse, aussehen?

Einer eingehenden Betrachtung - vornehmlich der interessanteren, nicht-holonomen Beschränkungen - folgend, wird im nächsten Kapitel am Beispiel der Implementierung eines Bewegungsplaners von J. Barraquand und J.-C. Latombe [BaLa 89] eine Charakterisierung der Grundprobleme nicht-holonomer Bewegungsplanung vorgenommen.

[3] Vgl. [Lato 91], Kapitel 9, Seite 403 ff.

[4] Verschiedentlich auch kinodynamische Beschränkungen genannt.

2.2.1. Holonome Beschränkungen

Sei ein Roboter[5] A gegeben, mit einem m-dimensionalen Konfigurationsraum C. Jede Konfiguration q in C kann durch eine Liste von Koordinaten $(q_1, \ldots, q_m)$ repräsentiert werden.

Den Konfigurationen von A sei zum Zeitpunkt t eine zusätzliche skalare Beschränkung auferlegt:

$$F(q,t) = F(q_1, \ldots, q_m, t) = 0 \tag{2.1}$$

wobei F stetig differenzierbar sei. Diese Beschränkung selektiert eine Teilmenge der Konfigurationen von C (diejenigen, die dieser Beschränkung genügen), welche wiederum definiert, wo A sich befinden darf.

Gleichung 2.1 kann nun dazu benutzt werden, um eine der Koordinaten, beispielsweise q_m, zu beschreiben, indem man sie als eine Funktion g der verbliebenen $m-1$ Koordinaten und der Zeit t ausdrückt: $g(q_1, \ldots, q_{m-1}, t)$. Die Funktion g ist ebenfalls stetig, so dass die Gleichung 2.1 eine $(m-1)$-dimensionale, stetige Kopie von C bestimmt. Diese Kopie ist der eigentliche Konfigurationsraum von A[6]

Definition 1: *Eine skalare Beschränkung der Form $F(q,t) = 0$, mit F ist stetig und differenzierbar, bezeichnet man als* **holonome** *Gleichheitsbeschränkung.*

Seien allgemein betrachtet nun k holonome Gleichheitsbeschränkungen gegeben, mit $k \leq m$. Wenn diese unabhängig sind, bestimmen sie eine $(m-k)$-dimensionale Kopie[7] von C, welche den wirklichen Konfigurationsraum von A darstellt.

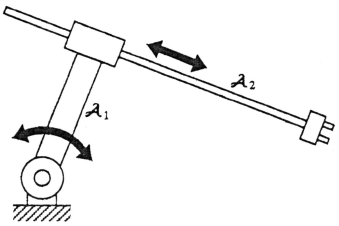

Abbildung 2.2.: Roboterarm mit je einem Dreh- und Schiebegelenk

Beispiel: Typische holonome Beschränkungen sind solche, die von Dreh- und Schiebegelenken eines Roboterarmes auferlegt werden. Ein planarer Arm mit zwei Drehgelenken (vgl. Abb. 2.2) besteht aus zwei beweglichen Objekten A_1 und A_2. Der zusammengesetzte Konfigurationsraum dieser Objekte ist sechsdimensional: $R^2 \times S^1 \times R^2 \times S^1$. Jedes Glied bestimmt zwei skalare Beschränkungen der Form $F(q) = 0$ hinsichtlich der Konfigurationen des Armes. Das erste Gelenk bestimmt, dass ein Punkt von A_1 (Drehpunkt des ersten Gliedes) relativ zum Raum fixiert ist. Das zweite Glied bestimmt wiederum, dass ein Punkt von A_2 (Drehpunkt

[5] Vgl. [Lato 91], Seite 411 f.

[6] Falls 2.1 von t abhängt, ist dieser Raum von der Zeit abhängig, anderenfalls ist er unabhängig. Viele holonome Beschränkungen, z.B. diejenigen generiert von den Dreh- nud Schiebegelenken eines Roboterarmes, sind unabhängig von der Zeit.

[7] Engl.: submanifold; Damit ist ein geeigneter Subraum von C gemeint, der ausreicht, um eine Konfiguration von A eindeutig zu bestimmen.

des zweiten Gliedes) relativ zu A_1 fixiert ist. Mit diesen vier Beschränkungen ergibt sich für den Roboter ein nur noch 2-dimensionaler Konfigurationsraum: $S^1 \times S^1$.
Eine Beschränkung der Form:

$$F(q,t) < 0, \text{ oder } F(q,t) \leq 0$$

entspricht typischerweise einem mechanischen Anschlag oder einem Hindernis. Sie bestimmt eine Teilmenge von C mit üblicherweise derselben Dimension wie C. Man bezeichnet solche als *holonome Ungleichheitsbeschränkungen.*

2.2.2. Nicht-holonome Beschränkungen

Nun betrachte man einen sich bewegenden Roboter[8] A. Dessen Konfiguration q ist eine differenzierbare Funktion der Zeit t. Es wird unterstellt, dass dessen Bewegungen einer skalaren Beschränkung der folgenden Form genügen:

$$G(q,\dot{q},t) = G(q_1,\ldots,q_m,\dot{q}_1,\ldots,\dot{q}_m,t) = 0 \tag{2.2}$$

Hierbei sei G stetig und $\dot{q}_i = \frac{dq_i}{dt}$ für alle $i = 1,\ldots,m$. Der Geschwindigkeitsvektor $\dot{q} = (\dot{q}_1,\ldots,\dot{q}_m)$ ist ein Vektor aus $T_q(C)$, dem Tangentenraum von C and der Stelle der Konfiguration q. Läge keine Beschränkung der Form (2.2) vor, dann wäre der Tangentenraum der Raum der Geschwindigkeiten von A.

Eine Beschränkung der Form (2.2) ist holonom, falls sie integrierbar ist, d.h. falls alle Parameter $\dot{q}_1$ bis $\dot{q}_m$ eliminiert und die Gleichung (2.2) in die Form (2.1) überführt werden kann. Anderenfalls ist die Beschränkung *nicht-holonom.*

Definition 2: *Eine nicht integrierbare, skalare Beschränkung der Form*

$$G(q_1,\ldots,q_m,\dot{q}_1,\ldots,\dot{q}_m,t) = 0,$$

wobei G stetig ist, bezeichnet man als **nicht-holonome** *Gleichheitsbeschränkung.*

Beispiel: Man betrachte einen autoähnlichen Roboter, der sich auf einer ebenen Fläche bewegt. Analog zu Abb. 1.2 kann dieser Roboter als ein sich in $W = R^2$ bewegendes Rechteck modelliert werden. Der Konfigurationsraum C ist $R^2 \times S^1$. Eine Konfiguration wird als Tripel (x,y,θ) geschrieben, analog Abschnitt 1.2.1. Des Weiteren wird angenommen, dass der Kontakt zwischen den Rädern und der Fahrbahn perfekt ist, d.h. es handelt sich um einen *rein rollenden Kontakt* ohne Relativbewegung. Sobald der Roboter sich bewegt, beschreibt der Punkt R eine Kurve γ, die tangential zur Längsachse des Fahrzeugs sein muss. Die Bewegungen des Roboters sind daher beschränkt durch die folgende Gleichung:

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0,$$

was der Form (2.2) entspricht. Vorwegnehmend sei gesagt, dass diese Beschränkung nicht integrierbar ist und daher nicht-holonom. Jede Konfiguration (x,y,θ) unterstellt, dass die

[8] Vgl. [Lato 91], Seite 412 ff.

Geschwindigkeit $(\dot{x}, \dot{y}, \dot{\theta})$ in einem zweidimensionalen Vektor-Subraum des Tangentenraumes liegt. Wenn man nun die x, y, θ-Repräsentation des Konfigurationsraumes des Roboters mit seiner $\dot{x}, \dot{y}, \dot{\theta}$-Repräsentation des Tangentenraumes überlagert, kann man den Geschwindigkeits-Subraum als eine Ebene rechtwinklig zur xy-Ebene entlang der Längsachse des Fahrzeugs darstellen (vgl. Abb. 2.3). Lokal betrachtet, ist der vom Fahrzeug im x, y, θ-Raum beschriebene Pfad τ eine Spindel.

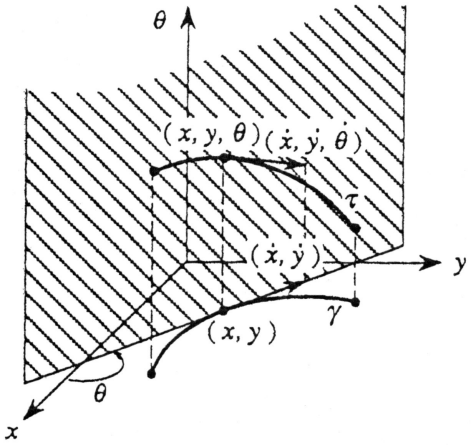

Abbildung 2.3.: Geschwindigkeits-Subraum eines Roboters

Obgleich ein autoähnlicher Roboter im Mittelpunkt der Betrachtungen in dieser Ausarbeitung steht, ist dies keineswegs das einzige Beispiel für einen Roboter, dessen Bewegungen nicht-holonomen Beschränkungen unterliegt. Andere Arbeiten auf diesem Gebiet befassen sich beispielsweise mit schwebenden und fliegenden Roboter, welche durch so genannte Thrusters gesteuert werden [Alex 87], oder auch "geschickte Hände"[9], deren Fingerkuppen rollende Bewegungen in Kontakt mit einem Objekt vollführen [Cutk 85], [Cai 88], [LCS 89].

Eine nicht-holonome Gleichheitsbeschränkung beschränkt den Geschwindigkeitsraum für A bei einer beliebigen Konfiguration q auf einen $(m-1)$-dimensionalen Vektor Subraum von $T_q(C)$. Wenn k unabhängige Gleichheitsbeschränkungen der Form (2.2) für die Bewegungen eines Roboters gelten, dann ist der resultierende Geschwindigkeitsraum ein Subraum von $T_q(C)$ mit der Dimension $m-k$.

Nicht-holonome Gleichheitsbeschränkungen werden oft durch einen rollenden Kontakt verursacht (im obigen Beispiel der Kontakt zwischen den Rädern des Fahrzeugs und der Fahrbahn). Rollender Kontakt bedeutet, dass keine Relativbewegung - in diesem Fall zwischen der Lauffläche des Rades und der Fahrbahnoberfläche - stattfindet. Sobald eine solche Relativbewegung vorliegt, d.h. eine Kombination aus Rollen und Rutschen, dann wird die Gleichung nicht-linear, in Abhängigkeit des Reibbeiwertes der beiden Oberflächen. Im Rahmen dieser Ausarbeitung wird nicht auf nicht-lineare Beschränkungen eingegangen, zur Vertiefung dieser Thematik sei auf [BaLa 90] verwiesen.
Eine Ungleichheitsbeschränkung der Form:

$$G(q, \dot{q}, t) < 0 \text{ oder } G(q, \dot{q}, t) \leq 0$$

[9]Engl.: dexterous hands.

schränkt die für A möglichen Geschwindigkeiten üblicherweise ein, ohne dessen Dimension zu ändern. Dies wird als *nicht-holonome Ungleichheitsbeschränkung* bezeichnet.

Beispiel: Man betrachte abermals einen Roboter entsprechend Abb. 1.2. Sei ϕ der Winkel zwischen der Längsachse des Fahrzeugs und dem Geschwindigkeitsvektor des Punktes F zwischen den beiden Vorderrädern. Man bezeichnet ϕ als *Lenkwinkel.* Üblicherweise ist ϕ durch mechanische Anschläge im Lenkgetriebe beschränkt, so dass gilt:

$$|\phi| \leq \phi_{max} < \frac{\pi}{2}.$$

Die Folgen dieser Beschränkung für die Bewegungsmöglichkeiten des Roboters sind offensichtlich. Der Punkt R zwischen den beiden Hinterrädern muss eine Kurve beschreiben, deren Krümmung der folgenden oberen Schranke unterliegt:

$$\frac{1}{\rho_{min}} = \frac{1}{L} \tan \phi_{max},$$

wobei L der Abstand zwischen R und F ist. $\rho_{min} = L/\tan\phi_{max}$ ist der *minimale Kurvenradius* des Fahrzeugs. Ein Roboter mit der Eigenschaft $|\phi| \leq \phi_{max} < \pi/2$ besitzt einen *begrenzenten Lenkeinschlag* oder auch *minimalen Wendekreis.*

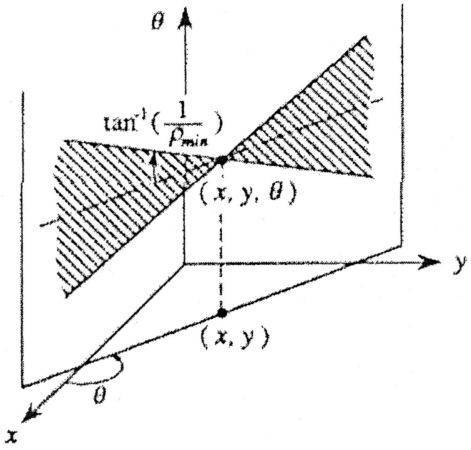

Abbildung 2.4.: Geschwindigkeitsvektorraum einer Konfiguration

Sei v die Geschwindigkeit von R, gemessen entlang der Längsache des Fahrzeugs, mit $-v_{max} \leq v \leq v_{max}$. Falls $v > 0$ (bzw. $v < 0$) und $\phi = 0$ ist, dann fährt das Fahrzeug vorwärts (bzw. rückwärts) geradeaus. Falls $v > 0$ und $\phi > 0$ (bzw. $\phi < 0$) ist, dann fährt das Fahrzeug vorwärts durch eine Linkskurve (bzw. Rechtskurve).
Die Beschränkung $|\phi| \leq \phi_{max} < \pi/2$ kann umgeformt werden zu:

$$|\dot{\theta}| \leq \frac{v}{\rho_{min}}$$

oder zu:

$$\dot{x}^2 + \dot{y}^2 - \rho_{min}^2 \dot{\theta}^2 \leq 0.$$

Daher handelt es sich hierbei um eine Beschränkung der Form $G(q, \dot{q}) \leq 0$. Somit wird jeder beliebigen Konfiguration q auferlegt, dass der Geschwindigkeitsvektor $(\dot{x}, \dot{y}, \dot{\theta})$ in einen zweiseitigen Konus mit Winkel $2\tan^{-1}(1/\rho_{min})$ zeigt, wie in Abbildung 2.4 ersichtlich.

In diesem und dem vorhergehenden Kapitel wurden sowohl die klassischen Verfahren, als auch die für das vorliegende Problem entscheidenden Einflussfaktoren betrachtet. Im folgenden Kapitel wird die Problemstellung anhand der exemplarischen Implementierung eines passenden Verfahrens konkretisiert.

2.3. Kategorisierung

Abhängig vom jeweiligen Ansatz können Modelle zur Bewegungsplanung in verschiedene Kategorien eingordnet werden. Zunächst erfolgt eine grundlegende Unterscheidung in **geometrisch-modellbasierte Ansätze**, man nennt diese auch **deliberative Ansätze**, und in **reaktive, sensorbasierte Ansätze**. Vertreter der ersteren Kategorie priorisieren den Planungsaspekt und konzentrieren sich auf eine geometrische Repräsentation der Problemstellung. Letztere hingegen basieren auf einer engen Kopplung von Sensorik und Aktuatoren und weisen einen kontinuierlichen bzw. inkrementellen Planungscharakter auf. Es sind außerdem verschiedene hybride Modelle denkbar, welche verschiedene Merkmale beider Konzepte besitzen.

Eine Differenzierung findet dabei in erster Linie hinsichtlich der Erfassung und Handhabung der Umwelt bzw. der Informationen statt, die diese repräsentieren. Reaktive Modelle schließen die Erfassung ausdrücklich in den Planungsprozess mit ein. Dabei ist es oft notwendig, noch vor Beginn der eigentlichen Bewegungsplanung zusätliche Probleme zu lösen, welche wiederum eine eigene Planungsphase erfordern können: Wie ist der Roboter zu manövrieren, um das relevante Umfeld möglichst effizient zu erfassen? Allein diese Frage ist Inhalt verschiedener Arbeiten, wie beispielsweise [HGB 99].

Deliberative Modelle setzen mehr oder weniger vollkommene Kenntnis über die Umwelt voraus, worin sich der statische Charakter manifestiert. Bevor jegliche Bewegungen ausgeführt werden, muss der Pfad vollständig berechnet sein. Inkrementelle Änderungen sind systembedingt in den meisten Fällen nicht vorgesehen. Änderungen der Strategie haben dann eine völlig neue Planung zur Folge.

Wie in der Einleitung bereits angekündigt, werden die drei klassischen Vertreter geometrisch-modellbasierter Verfahren in den folgenden Abschnitten betrachtet. Anschließend wird die Struktur reaktiver sensorbasierter Verfahren und andere relevante Modelle diskutiert.

2.4. Roadmap-Methoden

Das Grundprinzip der Roadmap-Methoden liegt darin, bezüglich eines Roboters ein Netzwerk (*Roadmap*) möglicher Verbindungen in C_{free} zu bilden. Eine solche Roadmap R wird dann als eine Menge standardisierter Pfade genützt. Die Bewegungsplanung reduziert sich in diesem Fall auf die Verbindung von q_{init} und q_{goal} mit R und die Suche nach einem Pfad innerhalb R zwischen den gefundenen Verbindungspunkten. Sofern ein Pfad gefunden werden kann, besteht er grundsätzlich aus der Verknüpfung dreier Teilpfade: eine Teilverbindung zwischen q_{init} und R, einer in R enthaltenen Verbindung und einer Teilverbindung zwischen R und q_{goal}.

In dieser Kategorie gibt es eine Anzahl verschiedener Ansätze. Je nach gewähltem Ansatz werden die berechneten Roadmaps verschieden benannt: *Sichtbarkeitsgraphen*, *Voronoi Diagramme*, *Freeway* oder *Silhouette*. Im Folgenden werden zwei ausgewählte Beispiele beschrieben.

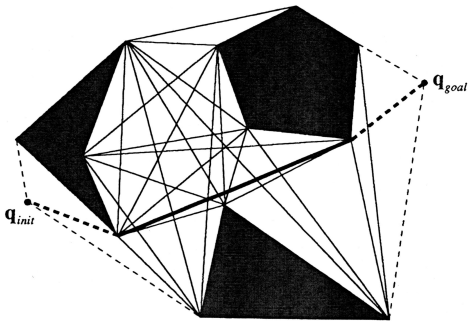

Abbildung 2.5.: Sichtbarkeitsgraph

Methoden auf Basis von **Sichtbarkeitsgraphen** stammen aus der Anfangszeit der Bewegungsplanung. Betrachtungsgegenstand ist meist ein zweidimensionaler Konfigurationsraum mit polygonalen C-Obstacle Regionen. Der Sichtbarkeitsgraph ist hierbei ein ungerichteter Graph G, dessen Knoten die Konfigurationen q_{init} und q_{goal}, sowie die Eckpunkte aller C-Obstacles sind. Kanten des Graphen sind alle gradlinigen Verbindungen zwischen zwei Knoten, die kein C-Obstacle schneiden, wie am Beispiel in Abb. 2.5 ersichtlich. In der Abbildung ist R mittels durchgehender Linien dargestellt, die verbleibenden Kanten von G verbinden q_{init} und q_{goal} mit R. G kann nun beispielsweise anhand der euklidischen Metrik nach den kürzesten halb-freien Pfad durchsucht werden, welcher, sofern vorhanden, q_{init} und q_{goal} mit einer polygonalen Linie verbindet.

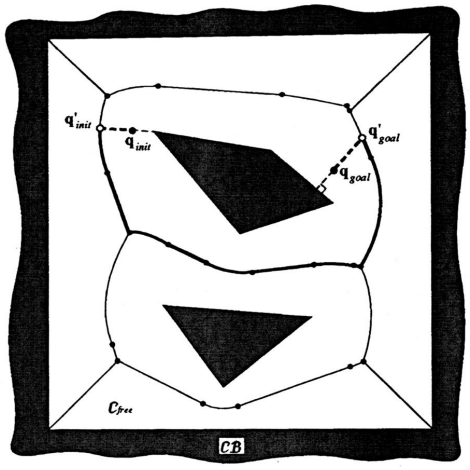

Abbildung 2.6.: Voronoi Diagramm

Eine weitere Methode definiert eine Funktion von C_{free} auf eine eindimesionales Teilmenge von sich selbst, man nennt dies **Refraction**. In einem zweidimensionalen Konfigurationsraum

C_{free} wird typischerweise auf dessen Voronoi Diagramm "retraktiert"[10]. Dieses Diagramm enthält alle freien Konfigurationen, deren minimaler Abstand zu CB mit mindestens zwei zugehörigen Punkten aus CB erreicht wird, vgl. hierzu Abb. 2.6. Die Tendenz, einen Pfad zu schaffen, welcher die Abstände zu den Hindernissen maximiert, ist zugleich ein entscheidender Vorteil der Methode. Dadurch erhält man einen Pfad, der frei von Hindernisberührungen ist. Andere Methoden liefern unter Umständen zunächst einen Pfad, der Berührungen mit einem oder mehreren Hindernissen aufweist und möglicherweise in einem zusätzlichen Schritt so verändert werden muss, dass keine Berührungen stattfinden. Besteht die Menge der C-Obstacles aus Polygonen, setzt sich das Voronoi Diagramm aus geraden Strecken und parabolischen Kurven zusammen. Start- und Zielkonfiguration werden, wie in Abbildung 2.6 ersichtlich, auf q'_{init} und q'_{goal} zurückgezogen. Analog zur vorherigen Methode wird anschließend auch hier der Pfad aus den Teilverbindungen zu q_{init} und q_{goal} und dem in R enthaltenen Teilpfad konstruiert.

2.5. Zell-Dekomposition

Methoden der Zell-Dekomposition sind vielleicht die bisher am ausführlichsten untersuchten Methoden zur Bewegungsplanung. Das zu Grunde liegende Prinzip ist die Zerlegung, d.h. die Dekomposition, des freien Raumes in begrenzte und der Größe nach leicht zu handhabende Regionen, Zellen genannt, so dass ein Pfad zwischen zwei beliebigen Konfigurationen in einer solchen Zelle leicht generiert werden kann. Ein ungerichteter Graph zur Modellierung aneinander angrenzender Zellen wird dann konstruiert und durchsucht. Die Knoten dieses Graphen (auch Verbundenheitsgraph[11]) repräsentieren die aus C_{free} extrahierten Zellen. Es sind genau die unmittelbar aneinander grenzenden Zellen, die in diesem Graphen durch Kanten verbunden sind. Eine Suche in diesem Graphen liefert dann einen so genannten Kanal, d.h. eine Sequenz von Zellen, welcher wiederum die Berechnung eines kontinuierlichen freien Pfades erlaubt.

Methoden dieser Art können in zwei Kategorien getrennt werden: *exakte Zelldekomposition* und *approximative Zelldekomposition*

2.5.1. Exakte Zelldekomposition

Bei der **exakten Zelldekomposition** wird der freie Raum in Zellen aufgeteilt, deren Vereinigung wieder *exakt*[12] dem freien Raum entspricht. Die Grenze einer Zelle entspricht dann einer sich auf irgend eine Weise manifestierenden "kritischen" Eigenschaft des Roboters, wie beispielsweise einer plötzlichen und grundsätzlichen Änderung der kinematischen Beschränkungen.

Die Abbildung 2.7 illustriert die Methode der exakten Zelldekomposition anhand eines zweidimensionalen Konfigurationsraumes. Der freie Raum ist nach außen durch ein, nach innen durch drei Polygone begrenzt, und er ist exakt in trapezförmige und dreickige Zellen zerlegt. In diesem Fall wurden die Zellen durch Hinzufügen senkrechter Linien (Abb. 2.7(b))

[10] Der engl. Begriff "retracted" lässt sich in diesem Kontext nur schwerlich passend übersetzen. Gemeint ist "zurückgezogen", "reduziert" bzw. "vereinfacht".

[11] Engl.: connectivity graph.

[12] Dies ist exakt bezüglich der formalen Problemstellung der Bewegungsplanung. Das wirkliche, physische Problem kann bezüglich der Formalisierung lediglich näherungsweise erfasst werden.

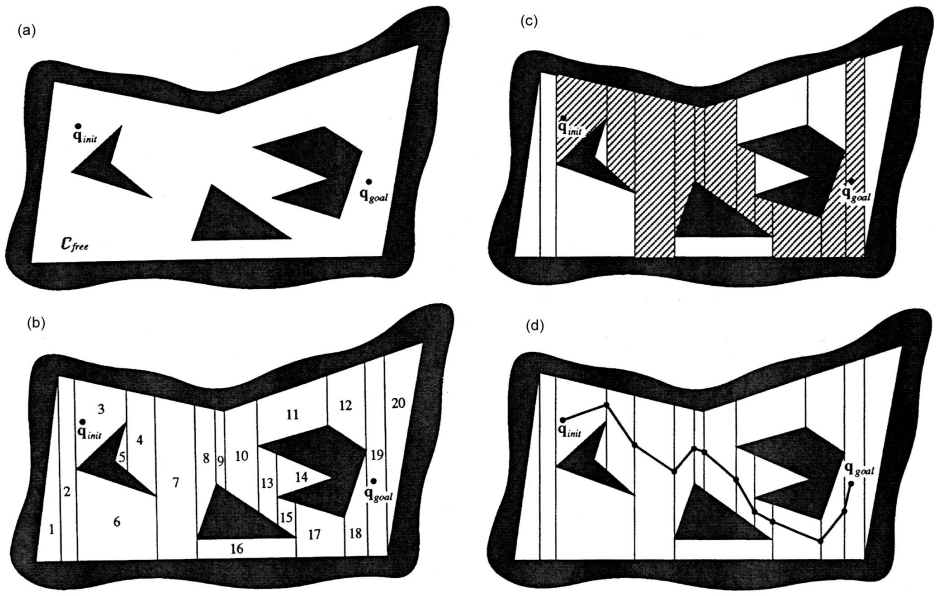

Abbildung 2.7.: Exakte Zelldekomposition

durch die Eckpunkte der C-Obstacles voneinander abgegrenzt. Wobei zwei Zellen dann benachbart sind, wenn sie einen gemeinsamen Teil einer solchen senkrechten Linie besitzen, dessen Länge größer 0 ist. Sobald der Verbundenheitsgraph konstruiert (ohne Abb.) und ein Kanal gefunden wurde (Abb. 2.7(c)), kann eine freier Pfad errechnet werden, indem die Startkonfiguration mit der Zielkonfiguration, anhand der Mittelpunkte der gemeinsamen Kanten benachbarter Zellen, miteinander verbunden wird (Abb. 2.7(d)).

2.5.2. Approximative Zelldekomposition

Die **Approximative Zelldekomposition** produziert Zellen vorgegebener Größe (z.B. Rechtecke), deren Vereinigung in jedem Fall im freien Raum enthalten ist. Hierbei stellen Zellengrenzen keine irgendwie gearteten Diskontinuitäten dar, sie haben keine höhere Bedeutung im Sinne der Bewegungsplanung.

Man betrachte ein ähnliches Beispiel für die Anwendung der approximativen Zelldekomposition in Abbildung 2.8. Analog zum vorhergehenden Beispiel wird der freie Raum ebenfalls nach außen, diesmal durch ein Rechteck R, und nach innen durch drei Polygone begrenzt. Das Rechteck R wird rekursiv in kleinere Rechtecke zerlegt, wobei in jedem Dekompositionsschritt ein Rechteck in vier identische Rechtecke aufgeteilt wird. Diese Dekomposition wird auch "Quadtree" genannt, da sie durch einen Baum des Grades 4 repräsentiert werden kann. Ab einer bestimmten Größe werden zur Konstruktion des Verbundenheitsgraphen nur noch die Zellen verwendet, deren Inhalt komplett im freien Raum liegt. Ist die Suche nach einem Kanal im Graphen erfolgreich, kann ebenso auch ein freier Pfad gefunden werden. Falls die Suche kein Ergebnis liefert, ist entweder die Auflösung der Dekomposition zu gering (d.h. die kleinste Zellengröße ist noch zu groß), oder es existiert kein freier Pfad zwischen q_{init} und q_{goal}. Hierbei bietet es sich an, zunächst mit einer relativ groben Auflösung zu beginnen

und diese solange zu verfeinern, bis entweder ein Pfad gefunden, oder eine untere Grenze der Auflösung erreicht wurde.

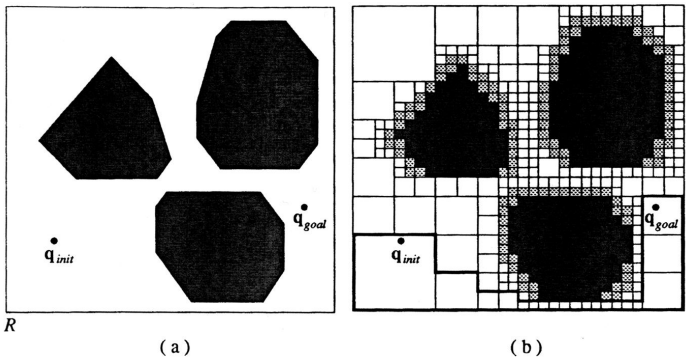

Abbildung 2.8.: Approximative Zelldekomposition

2.5.3. Vollständigkeit

Vorausgesetzt es werden passende Suchalgorithmen verwendet und exakte numerische Berechnungen durchgeführt, dann ist die Methode der exakten Zelldekomposition vollständig, d.h. ein freier Pfad - sofern existent - wird in jedem Fall gefunden, anderenfalls wird ein negatives Ergebnis ausgegeben. Methoden der approximativen Zelldekomposition hingegen sind möglicherweise nicht vollständig, obgleich in den meisten Fällen die verwendete Auflösung nahezu beliebig vergrößert werden kann, um dann auf Basis der verwendeten Auflösung Vollständigkeit zu erzielen. Dies ist dann mit entsprchend längerer Laufzeit verbunden. Der Vorteil der approximativen Methoden ist, dass sie für gewöhnlich weniger komplex und damit leichter zu implementieren sind.

2.6. Potenzialfeldmethoden

Ein klar strukturierter Ansatz zur Bewegungsplanung ist die Potenzialfeldmethode. Grundlage ist die Diskretisierung des Konfigurationsraumes in ein feines, gleichmäßiges Raster und die Suche nach dem Pfad in diesem Konfigurationsraster. Da das Raster für gewöhnlich sehr große Ausmaße besitzt, werden mächtige Heuristiken benötigt, um die Suche zu dirigieren. Hierbei sind verschiedene Heuristiken denkbar. Die wohl erfolgreichsten präsentieren sich in Form von Funktionen, die als Potenzialfelder interpretiert werden.

Um diesen Ansatz zu veranschaulichen, stelle man sich den als Punkt im Konfigurationsraum repräsentierten Roboter als ein Partikel vor, welches von den verschiedenen, künstlichen Potenzialen der Zielkonfiguration und der C-Obstacles beeinflusst wird. Die Zielkonfiguration selbst übt hierbei typischerweise eine Kraft aus, welche den Roboter in ihre Richtung zieht. Die Hindernisse hingegen erzeugen eine abstoßende Kraft auf den Roboter. Die daraus resultierende, künstliche Kraft, die auf den Roboter einwirkt, setzt sich so aus den Gradienten aller vorhandenen Potenziale zusammen. Für jede beliebige Konfiguration lässt sich dadurch die bevorzugte Richtung zum Ziel bestimmen, sie entspricht der gemeinsamen Kraftrichtung aller auf den Roboter einwirkenden Potenziale.

Abbildung 2.9 illustriert den Begriff der anziehenden und abstoßenden Potenziale und deren Kombination. Die anziehende Kraft (Abb. 2.9(b)) ist eine quadratische Senke, deren Minimum bei q_{goal} liegt. Je nach Entfernung zum jeweiligen Hindernis ist die abstoßende Kraft (Abb. 2.9(c)) zunächst 0, steigt mit der Annäherung an das Hindernis an und geht schließlich gegen unendlich (die Abb. ist bei einem Grenzwert abgeschnitten), sobald der Abstand zum Hindernis gegen Null geht. Ein Pfad zwischen q_{init} und q_{goal} ergibt sich dann aus dem negierten Gradienten des jeweils lokal wirkenden, totalen Potenzials (Abb. 2.9(e)). Eine Matrix aus den negierten Gradienten des Vektorfeldes findet sich in Abbildung 2.9(f).

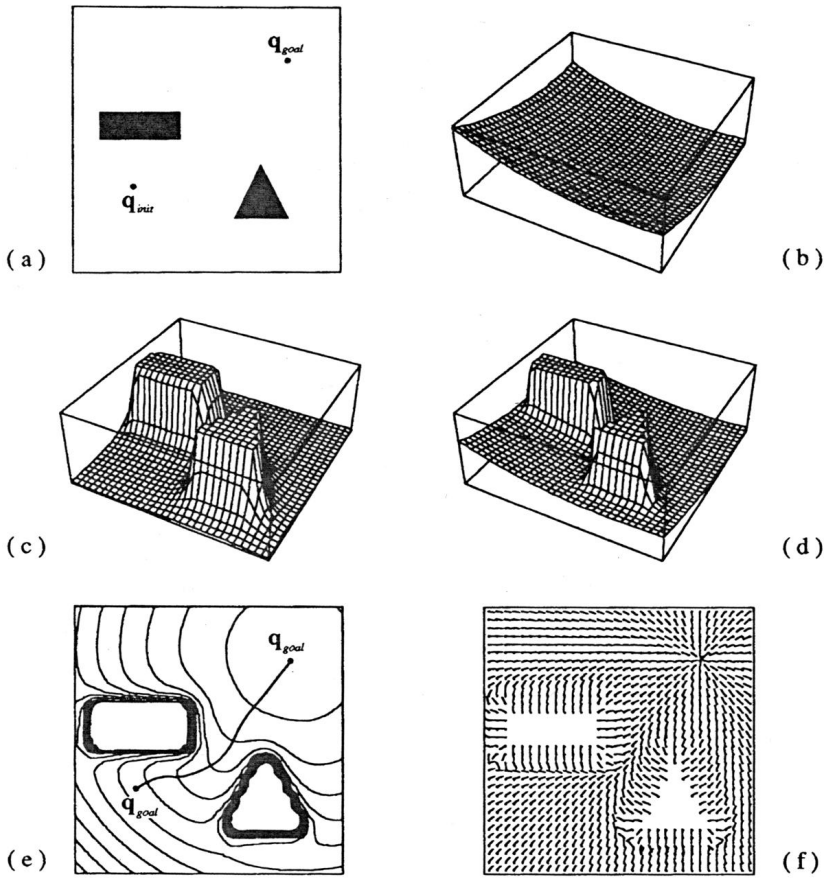

Abbildung 2.9.: Schemata der Potentialfeldmethode

Im Vergleich zu anderen Ansätzen können Potentialfeldmethoden sehr effizient sein, haben jedoch einen essenziellen Nachteil. Nachdem es sich prizipiell um ein Verfahren handelt, welches den schnellsten "Abstieg" zur Zielkonfiguration optimiert, kann es passieren, dass der Roboter in einem lokalen Minimum gefangen wird, welches nicht zur Zielkonfiguration führt. Diesem Problem muss separat begegnet werden, zum Beispiel indem von vorneherein Funktionen gewählt werden, welche keine lokalen Minima außer der Zielkonfiguration besitzen. Eine andere Maßnahme sind Mechanismen, die eine Befreiung aus lokalen Minima ermöglichen.

2.7. Reaktive sensorbasierte Bewegungsplaner

Für eine Einordnung in diese Kategorie der Bewegungsplaner müssen mindestens die folgenden zwei Eigenschaften erfüllt sein: Zum einen muss die Erfassung der Umwelt auf dem Wege einer Sensorik geschehen, was zumeist mit einem unvollkommenen Informationsmodell verbunden ist. Zum anderen muss eine enge Verbindung zwischen Sensoren und Aktuatoren bestehen, d.h. es findet eine kontinuierliche Bewegungsplanung statt, was in Kontrast zu klassischen Planungsansätzen steht.

Unter anderem auf Grund der folgenden Umstände ist sensorbasierte Bewegungsplanung ein wichtiger Teilbereich der Bewegungsplanung:

- Der Roboter hat a priori meist **keine vollständigen Informationen** über die Umwelt.
- Die **Repräsentation der Umwelt** liegt auf Grund der begrenzten Speicherkapazität des Roboters zumeist lediglich in einer **vereinfachten** bzw. **groben Struktur** vor.
- Eventuell **vorhandene Ungenauigkeiten** in der Repräsentation können durch sensorbasierte Ansätze **verkleinert bzw. eliminiert** werden.
- Die Umwelt unterliegt für gewöhnlich **oft auftretenden, dynamischen Veränderungen** und ist nicht statisch erfassbar.

Der hierbei unterstellte Charakter steht jedoch nicht automatisch in Widerspruch zu den weiter oben angeführten Methoden. In den meisten Fällen stellt er eine Kombination dar, welche klassische Ansätze mit den Eigenschaften sensorbasierter Verfahren vereint bzw. erweitert. [ChBu 94] beinhaltet beispielsweise eine Kombination aus der sensorbasierten Konstruktion eines generalisierten Voronoi Graphen (vgl. Abschnitt 2.4) und dem "Opportunistic Path Planner" von Canny und Lin [CaLi 90], [CaLi 93] und [Cann 88].

Auf Grund der Parallelen in Bezug auf dynamische Komponenten, unvollständige Information und Interaktivität ist dieser Teilbereich für die vorliegende Arbeit besonders ausschlaggebend. Weitere Ausführungen hinsichtlich der Aspekte sensorischer Umfelderkennung finden sich in Abschnitt 4.6.1. Ferner werden in Abschnitt 4.6 unter anderem technische und rechtliche Aspekte erörtert, und in Abschnitt 5.2.3 erfolgt ein Ausblick auf zukünftige Anwendungen und Entwicklungen.

2.8. Sonderfälle / Hybride Modelle

Es liegt nahe, einige Sonderfälle zu betrachten, die die vielfältigen Anwendungen der Bewegungsplanung und deren Teilbereiche verdeutlichen. Besonders interessant sind in diesem Zusammenhang zum Thema artverwandte Arbeiten und artverwandte hybride Modelle, welche Gegenstand der folgenden Abschnitte sind.

2.8.1. Bewegungsplaner für nicht-holonome Vehikel

Das Feld der Bewegungsplanung für Roboter unter nicht-holonomen Beschränkungen bietet Raum für unterschiedlichste Lösungen. Verschiedene Ansätze sind Gegenstand einiger Veröffentlichungen, aus denen einige Beispiele im Folgenden skizziert werden.

Ein Ansatz besteht aus einem konstruktiven Beweis der (lokalen) Kontrollierbarkeit [Laum 86, LaSi 89, LJTM 94]. In einer ersten Phase wird, unter Vernachlässigung der nicht-holonomen Beschränkungen, ein freier Pfad konstruiert. In einer zweiten Phase wird der gefundene Pfad dann in einen topologisch äquivalenten Pfad transformiert, welcher den Beschränkungen genügt. Nachteil dieses Vorgehens ist das eventuelle Vorhandensein unnötig vieler "Cusps"[13]. In manchen Umgebungen kann diese Zahl möglicherweise durch Verwendung verschiedener Heuristiken verringert werden.

Ein weiterer Ansatz, speziell entwickelt für autoähnliche Roboter, basiert auf einem Netzwerk von "Korridoren" [ToJe 88] bzw. "Fahrbahnen" [Wilf 88]. Diese werden in einem ersten Schritt der Planung aus dem Workspace extrahiert[14]. Anschließend werden lokale Planungstechniken verwendet, um die Bewegungen des Roboters an den Kreuzungspunkten der Korridore (bzw. Fahrbahnen) zu generieren, welche die Transitionen ermöglichen. Die größte Schwierigkeit liegt dabei in der Definition eines intrinsischen Geflechtes an Korridoren (bzw. Fahrbahnen), die für die meisten Workspaces nicht möglich ist. Hinzu kommt, dass die im Laufe eines Pfades generierten vielfältigen Übergänge üblicherweise Interdependenzen zeigen und die an den Kreuzungspunkten verwendeten lokalen Planungstechniken möglicherweise suboptimale Teilpfade liefern.

Weitaus weniger ausführlich wurden bisher lokal nicht-kontrollierbare Roboter untersucht. Hierbei konzentrieren sich die Forschungen fast ausschließlich auf Roboter, welche nur vorwärts fahren können. Anzuführen sind unter anderem Modelle zur Behandlung polygonaler [FoWi 88] und konvexer Hindernisse [ART 95] bzw. Modelle ohne Hindernisbehandlung [SoLa 93].

Um die vorliegende Grundproblematik eingehend zu verdeutlichen, wird in Kapitel 3 ein weiterer Vertreter nicht-holonomer Planungsverfahren genau beschrieben und die spezifischen Anforderungen der Problemstellung anhand der Implementierung aufgezeigt. Es handelt sich dabei um das bereits angesprochene Verfahren von J. Barraquand und J.-C. Latombe [BaLa 89].

Die Modellierung eines neuen Verfahrens, speziell adaptiert für die vorliegende Problemstellung, ist schließlich Inhalt des Kapitels 4. Sie stellt einen Ansatz dar, die spezifischen Vorteile verschiedener Verfahren zu vereinen, indem ein Vorgehen entwickelt wird, welches die Schlichtheit und Präzision klassischer Methoden mit der Echtzeittauglichkeit reaktiver Verfahren kombiniert und außerdem hinsichtlich einer konkreten technischen Umsetzung konzipiert ist.

2.8.2. Robot Work Crew

Angesichts der in der Raumfahrt gestellten Herausforderungen ist es nicht verwunderlich, dass in entsprechenden Einrichtungen[15] zahlreiche Anwendungen und Problemstellungen erforscht werden, die in Zusammenhang mit dem Thema dieser Arbeit stehen. Die unbemannte Raumfahrt ist eines derjenigen Einsatzfelder, welche vorrangig nach autonomen Robotern verlangen.

Projekte im Bereich der Raumfahrt zeichnen sich oft durch außerordentlich hohe Kosten und entsprechenden Umfang aus. Für Projekte in einem solchen Umfang reicht eine auf einen einzelnen Roboter ausgerichtete Bewegungsplanung oft nicht mehr aus. Beim Einsatz

[13] Engl. "changes of sign of the linear velocity", dt.: Änd. der (linearen) Fahrtrichtung; vgl. [GoO' 97], S. 768.
[14] Beispielsweise mit der "freeway extraction technique" aus [Lato 91], Kapitel 4, Abschn. 3.1.
[15] z.B. das Jet Propulsion Laboratory (JPL) der Nasa

mehrerer autonomer Roboter entsteht ein hoher Aufwand an Koordination und Abstimmung zwischen den einzelnen Robotern. Dieser Umstand wird gleichermaßen in der Bewegungsplanung eines jeden Roboters reflektiert.

Die RWC (Robot Work Crew) am JPL ist ein solches Projekt, welches sich nicht nur mit der Bewegungsplanung eines einzelnen, sondern der Koordination und Zusammenarbeit mehrerer Roboter befasst. In der RWC arbeiten mehrere Roboter zusammen, welche jeweils mit einem Basissatz an "Intelligenz" ausgerüstet sind und ihre Aktionen in der Gruppe (zentral oder quasi-demokratisch) koordinieren (lassen), um komplexere Vorhaben durchzuführen.

Basissatz an Intelligenz bedeutet in diesem Fall, dass ein einzelner Roboter in der Lage ist, die klassischen Szenarien der singulären Betrachtung zu lösen: Positionsbestimmung, Navigation bzw. Bewegungsplanung und dergleichen mehr. Die Gruppenkomponente besteht darin, die Arbeitskraft mehrerer einzelner Roboter dahingehend zu bündeln, dass sie nicht nur in der Lage sind, autonome Aufgaben zu erfüllen, sondern durch gezielte Zusammenarbeit auch Aufgaben zu bewältigen, die die Leistung eines einzelnen überstiegen. Der Transport eines größeren Werkstücks fällt ebenso in diese Kategorie, wie auch Aufgaben, die eine zeitgleiche Durchführung örtlich getrennter Aktionen erfordern.

Die Anforderungen an die Bewegungsplanung sind in diesem Bereich nicht nur besonders vielschichtig (u.a. sensorbasierte Bewegungsplanung bei unvollkommener Information und schlechter Infrastruktur), sondern sind ebenfalls erweitert um Probleme der Kommunikation, Koordination und diverser Herausforderungen der Motorik und verwandter Gebiete. Weiterführende Informationen sind bei [RWC] zu finden.

3. Verfahren von Barraquand und Latombe

Dieses Verfahren ist ein Vertreter aus dem Bereich der Zell-Dekompositionen. Wie weiter unten ersichtlich wird, sind die Merkmale des Vorgehens ein statischer Planungshorizont, die Durchführung umfangreicher Vorarbeiten zum Aufbau diverser Datenstrukturen und die Anwendung des Dijkstra-Algorithmus zur Kennzeichnung des optimalen Pfades.

Besonders hervor zu heben sind die Möglichkeit zur Hinterlegung eines individuellen Präferenzprofils für den Fahrer und die Tatsache, dass der Algorithmus um so schneller zu einem Ergebnis kommt, je mehr Hindernisse sich im Konfigurationsraum befinden.

3.1. Beschreibung

Im Folgenden wird das Verfahren beschrieben, gefolgt von einer Betrachtung der Aspekte der Implementierung und der Detaillösungen, sowie eine Analyse des Verfahrens und dessen Potenzial.

3.1.1. Dekomposition des Konfigurationsraumes

Entsprechend der Grundidee aller Verfahren der Zell-Dekomposition ist die Grundlage dieses Verfahrens die Dekomposition des Konfigurationsraumes $R^2 \times [0, 2\pi[$ in ein Array kleiner Rechtecke[1]. Dabei wird der Workspace je auf der x- bzw. y-Achse diskretisiert sowie die θ-Werte normalisiert. Das Ergebnis ist ein dreidimensionales Array der Größe 128^3. Die Festlegung auf eine Ordnung von 128 ist ein Kompromis zwischen einer hinreichenden Genauigkeit einerseits und möglichst kleinem Speicherbedarf andererseits. Je nach dem konkretem Anwendungsfall spricht nichts gegen eine Variation dieses Wertes in beide Richtungen. Der exponentiellen Wirkung dieses Parameters auf den Zeit- bzw. Speicherbedarf sollte man sich hierbei dennoch bewusst sein.

3.1.2. Aufbau des Suchgraphen

Der nächste Schritt ist der Aufbau eines gerichteten Suchgraphen, dessen Knoten genau die Zellen der im vorigen Abschnitt angesprochenen Dekomposition sind. Zwei Knoten dieses Graphen sind genau dann benachbart, wenn es einen Pfad gibt, der von einer Konfiguration der einen Zelle zu einer Konfiguration der zweiten Zelle führt.

Ausgehend von einer Konfiguration q, anfangs q_{init}, erhält man einen solchen Pfad, indem maximal sechs Nachfolger generiert werden, deren Konfigurationen aufgrund der Variation der folgenden Parameter entstehen:

[1] Streng genommen sollten an dieser Stelle lediglich quadratische Zellen verwendet werden, da sich sonst verschiedene Anomalien im gefundenen Pfad ausmachen lassen. Es darf für eine Wegeplanungsentscheidung z.B. nicht von Bedeutung sein, in welche Richtung eine Teilstrecke weist. Solche Anomalien können i.A. durch die Verwendung quadratischer Zellen vermieden bzw. verringert werden.

3. Verfahren von Barraquand und Latombe

1. **Geschwindigkeit und Lenkwinkel** - Man erreicht sie durch eine Variation der Parameter

$$\{-v_0, v_0\} \times \{-\phi_{max}, 0, +\phi_{max}\}$$

   D.h. es wird eine Bewegung rückwärts oder vorwärts durchgeführt mit vollem Lenkeinschlag nach links oder rechts, bzw. ohne Lenkeinschlag.

   Diese Festlegung stützt sich auf die Diskussion in [Lato 91], Abschnitt 5.2. Daraus geht hervor folgendes hervor. Sobald ein möglicher freier Pfad zwischen q_{init} und q_{goal} existiert, dann existiert ebenfalls ein Pfad zwischen q_{init} und q_{goal}, der nur aus Kurven mit minimalem Kurvenradius besteht und der die Anzahl an Rückwärtsfahrten minimiert[2]. Das Zulassen von $\phi = 0$ erlaubt eine lineare Bewegung des Roboters. Diese Bewegungsart ist Voraussetzung, um die Länge des gefundenen Pfades zu reduzieren.

2. **Distanz** - Es wird lediglich eine *kurze* Distanz zurückgelegt. Mit kurz ist gemeint, dass das Zeitintervall über das integriert wird, 1 beträgt, und dass v_0 auf in etwa den zweifachen[3] Wert eines Intervalls der x-Achse gesetzt ist. Ziel ist es, zu vermeiden, dass die Bewegung in derselben Zelle endet, in der begonnen wurde und gleichzeitig nicht weiter als in eine direkt benachbarte Zelle führt.

Zu diesem Zweck werden im weiteren Sinne zwei Listen geführt von denen die eine so genannte *geschlossene* Knoten (CLOSED) enthält, die andere *offene* Konfigurationen (OPEN). Geschlossen sind diejenigen Knoten, deren sämtliche Nachfolger bereits generiert bzw. untersucht worden sind, wohingegen die offenen Konfigurationen noch zur Bearbeitung anstehen.

Auf der einen Seite *Knoten*, auf der anderen Seite *Konfigurationen* zu betrachten, ist sehr wohl beabsichtigt. Um sicherzustellen, dass mit konkreten Pfaden gearbeitet wird, ist es unerlässlich, sämtliche *Konfigurationen* auf dem Weg von q_{init} nach q_{goal} zu betrachten - nicht etwa die dazugehörigen Zellen. Auf diese Weise kann ein exakter Pfad berechnet werden, welcher lediglich in diskreter Form (Zellen) *aufgezeichnet* wird.

Anstatt einfacher Listen werden entsprechend geeignete Datenstrukturen verwendet. Geschlossene Knoten werden als solche jeweils durch ein Bit in einem Statusbyte der Gesamtstruktur, d.h. im oben angesprochenen Array, gekennzeichnet. Offene Knoten, deren Anzahl im Verhältnis üblicherweise wesentlich geringer ist, werden durch eine Double-ended Queue repräsentiert[4]. Der spätere Vorgang, am Anfang der "Dequeue" Elemente zu entnehmen, während gleichzeitig sukzessive am Ende welche hinzugefügt werden, ist der klassische Anwendungsfall für diese Datenstruktur.

Der Algorithmus startet mit der initialen Konfiguration q_{init} in OPEN. Bei jeder der nun folgenden Iterationen wird eine der Konfigurationen aus OPEN ausgewählt und geprüft, ob die vorliegende Konfiguration in der gleichen Zelle liegt, wie q_{goal}. Ist dies der Fall, so existiert ein entsprechender Pfad von q_{init} nach q_{goal} und ein entsprechendes Flag wird gesetzt. Anderenfalls wird festgestellt, ob die Konfiguration zu einer bereits als *geschlossen* markierten

[2] Bezogen auf die Menge aller möglicher freier Pfade zwischen q_{init} und q_{goal}.

[3] Vgl. [Lato 91], S.433. Auffallend diesbezüglich ist die Verwendung des *zweifachen* eines Intervalls, während dieser Multiplikatorwert m logischerweise bei $\sqrt{2} + \varepsilon \leq m$ (also $m \ll 2$) liegen müsste. Die Empirie stützt diese Vermutung.

[4] Latombe verwendet in [Lato 91] eine Baumstruktur, deren Vorteile jedoch in der knapp gehaltenen Beschreibung nicht nachvollzogen werden können.

Zelle gehört. Trifft dies zu, wird die Konfiguration verworfen[5]. Anderenfalls wird deren Zelle nun als CLOSED markiert, sämtliche Nachfolger generiert und diese wiederum an OPEN angehängt.

Sind keine Konfigurationen mehr in OPEN enthalten, bzw. alle Zellen als CLOSED markiert, und q_{goal} noch nicht erreicht, konnte kein passender Pfad gefunden werden.

3.1.3. Generierung der Nachfolger

Der Generierungsschritt beinhaltet jeweils mehrere Operationen. Zum einen müssen die bewegungsrelevanten Aktionen durchgeführt, bzw. die entsprechenden Konfigurationen am Ende der Bewegung berechnet werden. Zum anderen muss die für die spätere Bewertung notwendige Gewichtung vorgenommen werden. Zu diesem Zweck wird der mittels der in Abschnitt 3.4.2 betrachteten Kostenfunktion errechnete Wert als Kosten/Zelle-Paar im vorhergehenden Knoten hinterlegt.

Die Generierung der Nachfolgekonfigurationen geschieht folgendermaßen. Sei der Pfad des Roboters abhängig von t, der Zeit seit Beginn der Bewegung. Bei gegebenem Lenkeinschlag ϕ und Geschwindigkeit v von R, lauten die Geschwindigkeitsparamter des Roboters $v_x = v \cdot cos(\theta)$, $v_y = v \cdot sin(\theta)$, und $v_\theta = \frac{v}{L} tan(\phi)$. Somit erhält man:

- Falls $\phi = 0$:

$$x(t) = x_0 + cos(\theta(0))vt$$

$$y(t) = y_0 + sin(\theta(0))vt$$

$$\theta(t) = \theta(0)$$

- sonst:

$$x(t) = x(0) + \frac{L}{tan(\phi)}(sin(\theta(0) + \frac{v}{L}tan(\phi)) - sin(\theta(0)))$$

$$y(t) = y(0) - \frac{L}{tan(\phi)}(cos(\theta(0) + \frac{v}{L}tan(\phi)) - cos(\theta(0)))$$

$$\theta(t) = \theta(0) + \frac{v}{L}tan(\phi t)$$

Während der Generierung wird ein Zeiger auf die Ursprungszelle behalten, um die oben angesprochenen Wert/Zelle-Paare im Hinblick auf die Gewichtung zu hinterlegen. Hierzu wird eine Vektor-Datenstruktur verwendet, da je nach Situation eine unterschiedliche Anzahl Einträge abgelegt werden muss. Diese Datenstruktur wird anfangs als *leer* initialisiert und im Generierungsschritt gegebenenfalls mit einem oder mehreren Werten versehen. Auf diese Daten wird wiederum zurückgegriffen, um im Erfolgsfall den optimalen Pfad zu kennzeichnen (vgl. hierzu Abschnitt 3.1.5).

[5] Eine bereits zu einem früheren Zeitpunkt bearbeitete Konfiguration war der momentan betrachteten sehr ähnlich. Man kann demnach davon ausgehen, dass auch die Nachfolger beider Konfigurationen sehr ähnlich sind und somit auf eine explizite nochmalige Generierung verzichten.

3.1.4. Gewichtung

Im Sinne der Findung eines verwertbaren Pfades und dem intuitiv "richtigen" und logischen Führen des Fahrzeuges ist eine Kostenfunktion von Nöten, welche einen Gewichtungsparameter für den Suchbaum liefert.

Das Thema bei weitem nicht erschöpfend, sondern lediglich marginal erwähnend, führt Latombe in [Lato 91] die von ihm in den jeweiligen Beispielen verwendete Anzahl der Rückwärtsbewegungen des Fahrzeugs an. Gleichzeitig schränkt er ein, dass durch eine zu restriktive (bzw. zu freie) Handhabung dieser Kostenfunktion im Endeffekt zu lange (bzw. zu komplizierte[6]) Pfade generiert werden.

Die hier erlaubte Flexibilität ermöglicht die Verwendung unterschiedlichster Kostenfunktionen, in Abhängigkeit der jeweiligen Zielsetzung. Die für das vorliegende Verfahren entwickelte Kostenfunktion basiert auf einer Matrix von Zustandsübergängen und ist in Abschnitt 3.4.2 genauer beschrieben.

3.1.5. Kennzeichnen des optimalen Pfades

Die Suche nach dem kürzestmöglichen Pfad in dem so konstruierten Suchgraphen wird schließlich nach Dijkstra's Algorithmus[7] durchgeführt. Eine eingehende Beschreibung inklusive einer anschaulichen Beispielimplementierung ist unter anderem bei [Morr 98] zu finden.

Auf diese Weise erhält man, sofern vorhanden, die kürzeste Verbindung, die die vorgenommene Diskretisierung erlaubt. Sollte die im schlechtesten Fall maximal mögliche Abweichung vom Ziel (entsprechend knapp einem Intervall auf der x- bzw. y-Achse, respektive $\frac{\pi}{128} - \varepsilon$) nicht ausreichen, so bietet sich jederzeit die Möglichkeit einer erneuten Durchführung des gesamten Ablaufes. In diesem "Verfeinerungsschritt" wird wiederum die q_{goal} enthaltende Zelle, wie in Abschnitt 3.1.1 beschrieben, diskretisiert und die letzte, in diese Zelle führende Konfiguration als q_{init} übernommen. Global gesehen, wird auf diese Weise zwar nicht der optimale Pfad gefunden, jedoch erlaubt dieses inkrementelle Vorgehen eine Variation der Präzision in Abhängigkeit der Anforderungen.

3.2. Komplexität

Zunächst ist zu bemerken, dass auf Grund der Dekomposition des Konfigurationsraumes dieser Ansatz lediglich für Roboter mit kleinen Konfigurationsräumen geeignet ist. Konkret liegen die folgenden Komplexitäten[8] vor:

$$\text{Zeitkomplexität: } O(mr^m \log r),$$

$$\text{Speicherkomplexität: } O(r^m),$$

wobei $O(r)$ die Größe der Dekomposition entlang der Achsen und m die Dimension des Konfigurationsraumes ist.

[6] Beispielsweise Drehen des Fahrzeugs "auf der Stelle", durch andauerndes Vor- und Zurückstoßen.
[7] Lt. [Lato 91] ein so genannter "uninformed A*-Algorithm".
[8] Vgl. [Lato 91], S. 432.

3.3. Messergebnisse / Laufzeitverhalten

Messungen der Laufzeit bei verschiedenen Konstellationen bestätigen den schon angesprochenen hauptsächlichen Vorteil des Verfahrens: die Effizienz ist um so besser, je größer die von Hindernissen eingenommene Fläche des Workspace ist. Nachteilig ist die relativ unbefriedigende Effizienz bei Problemstellungen ohne Hindernisse.

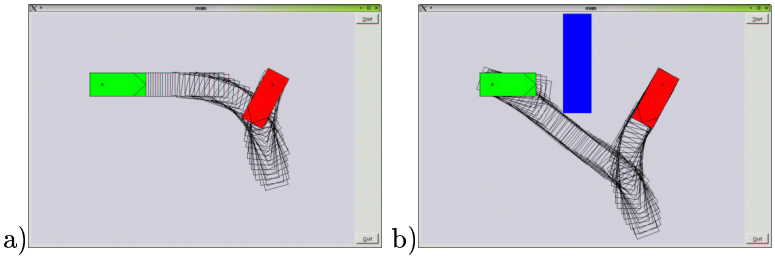

Abbildung 3.1.: Einfache Problemstellungen

Abbildung 3.1 zeigt die Fähigkeit des Verfahrens, einfache Problemstellungen lösen zu können. Die Laufzeiten unterscheiden sich im dargestellten Beispiel, unabhängig von der Länge des berechneten Pfades, nur unwesentlich: die Laufzeit liegt bei Planung ohne Hindernissen (Abb. 3.1 a) bei 65,81 Sekunden[9], bei Planung mit vorhandenem Hindernis (Abb. 3.1 b) bei 61,70 Sekunden. Was die Länge der gefundenen Pfade betrifft, liefert das Verfahren auch bei weniger komplexen Konstellationen gute Ergebnisse. Es ist jedoch besonders für Fälle geeignet, die sich durch eine höhere Anzahl großflächiger Hindernisse ausgezeichnen, wie im folgenden Absatz gezeigt wird.

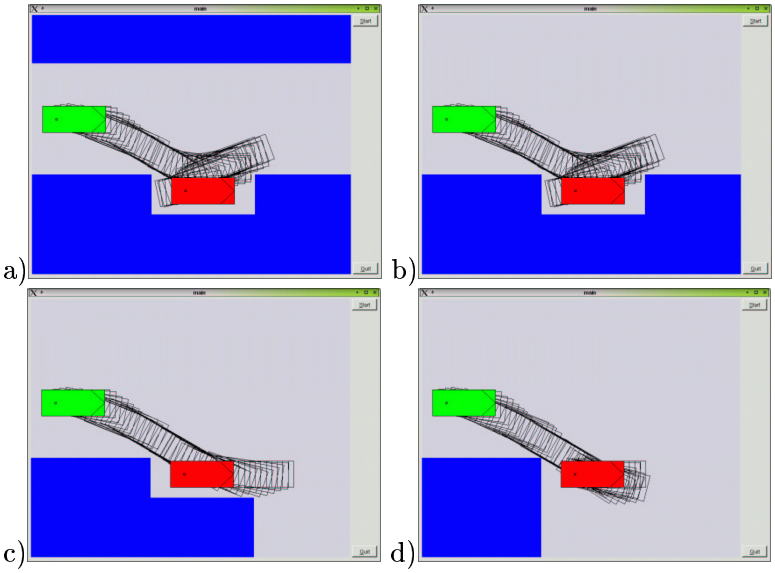

Abbildung 3.2.: Hinderniskonstellationen / Paralleles Einparken

[9]Alle Messungen erfolgten auf einem AMD Athlon basierten Linux System: 1.2 GHz, 512 MByte SDRAM; S.u.S.E. Linux 7.3 (Kernel 2.4.10), Implementierung in C++ / gcc V 2.95.3.

Am Beispiel des parallelen Einparkens (vgl. Abb. 3.2) wird deutlich, welchen Einfluss Hindernisse auf das Laufzeitverhalten haben. Mit einem flächenmäßig hohen Anteil der Hindernisse am Workspace reduziert sich die Laufzeit t_{search} der Berechnung des Pfades, da die von Hindernissen eingenommenen Teile von W beim Aufbau des Suchgraphen wegfallen und so die Größe des Graphen reduziert wird. Jedes einzelne Hindernis bedeutet jedoch eine verlängerte Vorbereitungszeit t_{prep}, da jeweils die entsprechenden C-Obstacles generiert werden müssen. Die nachstehende Tabelle enthält neben t_{search} und t_{prep} die jeweiligen Gesamtlaufzeit t_{total}, sowie Anzahl und Flächenanteil der Hindernisse. Die jeweiligen Konstellationen, d.h. Hindernisse und Pfade, sind der Abbildung 3.2 zu entnehmen. Start- und Zielkonfiguration sind in allen Fällen identisch.

Problemstellung	*Laufzeit (Sek.)*			*Hindernisse*	
	t_{total}	t_{search}	t_{prep}	#	%
Parallel Parken (3.2 a)	23,41	10,89	12,52	4	64,82
Parallel Parken (3.2 b)	29,52	19,96	9,56	3	51,18
Parallel Parken (3.2 c)	35,90	28,24	7,66	2	31,35
Parallel Parken (3.2 d)	43,14	37,67	5,47	1	18,59
Ident. Konf. (o. Abb.)	53,99	51,92	2,07	0	0,00

3.4. Implementierung

Das Verfahren erlaubt hinsichtlich der konkreten Implementierung im Detailfall an verschiedenen Stellen unterschiedliche Lösungsansätze. Dies trifft einerseits für die Hindernisbehandlung und andererseits für die zur Bewertung der Pfade benötigte Kostenfunktion zu, wie in den beiden folgenden Abschnitten ersichtlich wird. Im Übrigen bieten die Anhänge A.1 bis A.5 einen Einblick in verschiedene zentrale Teile des Quellcodes.

3.4.1. Hindernisbehandlung

Das zu Grunde liegende Verfahren erlaubt eine Kollisionserkennung auf verschiedene Art und Weise. Barraquand und Latombe verwenden exemplarisch eine kontinuierliche Methode. Jede generierte bzw. behandelte Konfiguration wird auf Kollision mit den Hindernissen im Workspace geprüft (vgl. [BaLa 89], Kapitel 7, Abschnitt 5.2).

In der vorliegenden Implementierung wurde eine alternative Variante gewählt. Es werden anfangs die entsprechenden C-Obstacles generiert, wie in Abschnitt 2.1 beschrieben. Anschließend werden die von den Hindernissen tangierten Zellen als *geschlossen* markiert. Im späteren Verlauf müssen so lediglich die *Konfigurationen* überprüft werden, was dann wesentlich weniger Aufwand bedeutet.

Das Laufzeitverhalten beider Varianten ist zwar unterschiedlich, da der im ersten Fall kontinuierlich anfallende Aufwand im zweiten Fall vorgezogen wird. Dies hat bei wiederholter Planung unter denselben Voraussetzungen den Vorteil, Mehrfachberechnungen zu vermeiden, bietet jedoch bei einmaliger Planung keine signifikanten Einsparungen bezüglich der Rechenzeit.

3.4.2. Kostenfunktion

Auf der Suche nach einer geeigneten Kostenfunktion stößt man sehr schnell auf weit mehr Einflussfaktoren, als lediglich die Länge des zurückgelegten Weges oder die Anzahl der Ände-

rungen der Bewegungsrichtung[10].

Die dem zugrundliegenden Verfahren inhärenten, diskreten Zustände lassen zumindest 36, unter Einbeziehung des anfänglich vorhandenen Stillstandes sogar 42 Zustandsübergänge zu. Diese sind durch entsprechende Variation des Lenkeinschlages und der gefahrenen Richtung gegeben.

Eine zahlenmäßige Bewertung dieser Transitionen erlaubt die Modellierung eines Präferenzmodells. Wobei man die für einen menschlichen Autofahrer typischen Präferenzen mittels einiger weniger Regeln ausdrücken kann:

1. Anstrengungen sind zu vermeiden, daher ist übermäßiges Lenken schlechter zu bewerten, als ein konstanter Lenkwinkel.

2. Eine Fahrt geradeaus ist besser zu bewerten, als eine Kurvenfahrt.

3. Nicht zuletzt bedingt durch die vorhandene Sitzposition und die daraus resultierende Blickrichtung ist eine Vorwärtsfahrt einer Rückwärtsfahrt vorzuziehen.

Die Kombination dieser Regeln ergibt die folgende 7×6 Kostenmatrix:

	v_l	v_g	v_r	r_l	r_g	r_r
ss	2	1	2	3	2	3
v_l	2	1	3	8	7	6
v_g	2	1	2	7	8	7
v_r	3	1	2	6	7	8
r_l	7	6	5	3	2	4
r_g	6	7	6	3	2	3
r_r	5	6	7	4	2	3

Den *absoluten* Werten unterliegt dabei keine höhere Bedeutung. Die Relation der Werte untereinander ist entscheidend für die Bewertung des gesamten Pfades, wobei bei gleichen Summenwerten keine bzw. lediglich eine zufällige Präferenz vorliegt ($1,1,1,2,4,2 \doteq 1,2,4,2,1,1$). Der Vorteil der Verwendung von ganzzahligen Werten liegt auf der Hand.

Das unterstellte Regelwerk kann und soll individuell an besondere Anforderungen angepasst werden. Ein Beispiel für ein speziell angepasstes Präferenzprofil wäre das Folgende. Sei das Fahrzeug maschinell gesteuert und rundherum mit gleicher Sensorik ausgestattet. Dem "Fahrer" ist in diesem Fall prinzipiell egal, in welche Richtung das Fahrzeug sich linear bewegt, er hat diesbezüglich also *keine Präferenz*. Sei der Roboter industriell eingesetzt. Man kann davon ausgehen, dass eine besondere Präferenz in Bezug auf Energieverbrauch und Verschleiß besteht. Daher ist nach wie vor eine Geradeausfahrt einer Kurvenfahrt vorzuziehen, ebenso sind übermäßig häufige Lenkmanöver und Fahrtrichtungswechsel zu vermeiden. Eine geeignete Kostenmatrix könnte in diesem Fall folgendermaßen aussehen:

[10] Engl.: cusps.

	v_l	v_g	v_r	r_l	r_g	r_r
ss	2	1	2	2	1	2
v_l	2	1	3	3	2	4
v_g	2	1	2	3	2	3
v_r	3	1	2	4	2	3
r_l	3	2	4	2	1	3
r_g	3	2	3	2	1	2
r_r	4	2	3	3	1	2

Andere Kostenfunktionen

Wie bereits verschiedentlich angedeutet, besteht in dieser Hinsicht eine relativ große Flexibilität. In Abhängigkeit des angestrebten Verwendungszwecks können die unterschiedlichsten Kriterien zur Gewichtung herangezogen werden. Dabei liegen nahezu keine weiteren Einschränkungen vor, solange ein verwertbares Gewicht berechnet werden kann. Denkbare Komponenten, deren Kombination für eine Gewichtung infrage käme, wären unter anderem die folgenden Werte, um nur einige Beispiele zu nennen:

- Die zurückgelegte, absolute **Wegstrecke**.
- Die **Anzahl diskreter Vorgänge** (Richtungsänderungen, Lenkmanöver, diskrete Aktionen zur Bedienung des Fahrzeugs, etc.).
- **Zustandsübergänge** (siehe oben).
- Der **Zeitbedarf** für verschiedene Aktionen, bzw. für bestimmte Abfolgen oder zusammengesetzte Manöver.
- Andere, klassische **Distanzmaße** (z.B. euklidische Distanz) für die Entfernung verschiedener Konfigurationen zu q_{goal}.

Die jeweilige Anwendung entscheidet über die Verwendung einzelner Kostenfunktionen, im vorliegenden Fall bietet sich das Fahrer-Präferenzmodell an.

3.5. Analyse / Potenzial

Der primäre Vorteil dieses Verfahrens wurde bereits angedeutet. Bedingt durch die Methode der Hindernisbehandlung erzielt man die besten Ergebnisse, sobald ein relativ großer Teil des Konfigurationsraumes von Hindernissen eingenommen wird. Abgesehen von der in der vorliegenden Implementierung verwendeten Generierung von C-Obstacles spielt es dabei nur eine sekundäre Rolle, ob es sich um viele kleine, oder wenige große Hindernisse handelt. Gesamtheitlich können vergleichsweise schwierige Planungsprobleme in angemessener Zeit gelöst werden. Ein daraus direkt resultierender Nachteil allerdings ist die damit verbundene, vergleichsweise schlechtere Leistung bei Planungsproblemen mit einer geringen Anzahl an Hindernissen, oder wenn gar keine vorhanden sind.

Der grundsätzliche Charakter des Verfahrens, nämlich die Zelldekomposition, bedingt einen der bedeutenderen Nachteile. Die notwendige Repräsentation des Konfigurationsraumes in Form von Zellen. Diese Repräsentation muss ständig im Speicher gehalten werden und

beansprucht entsprechende Ressourcen. Es handelt sich hierbei konzeptionell um ein Verfahren exakter Zelldekomposition. Die damit verbundene Vollständigkeit wird jedoch zu Gunsten einer gesteigerten Effizienz aufgegeben. Dies liegt daran, dass beim Aufbau des Suchgraphen ähnliche Konfigurationen verworfen werden, was wiederum möglicherweise dazu führen kann, dass die Zelle in der die Zielkonfiguration liegt, nicht gefunden werden kann. Abhängig vom jeweilgen Anwendungsfall birgt die mögliche Verwendung eines Verfeinerungsschrittes einen Vor- bzw. Nachteil. Einerseits ist es im Hinblick auf eine schnelle Berechnung möglich, ein grobes Raster zu verwenden, andererseits ist es dann eventuell notwendig, den angesprochenen Verfeinerungsschritt durchzuführen, verbunden mit entsprechend höherem Aufwand. Falls jedoch in weniger komplexen Fällen lediglich die Existenz eines Pfades bewiesen werden soll - ohne den Anspruch den optimalen Pfad zu finden - dann ist die Flexibilität bei der Festlegung der Rastergröße ein entscheidender Vorteil.

Auf den Erfüllungsgrad des Verfahrens hinsichtlich der Prämissen aus 1.2 wird im folgenden Abschnitt eingegangen.

3.5.1. Prämissen

- Wie eindeutig erkennbar ist, liegt der Fall einer **statischen Betrachtung ex ante** vor. Die fehlende Möglichkeit zur Behandlung von dynamischen Komponenten, wie z.B. sich bewegenden Hindernissen ist in der ursprünglichen Form nicht vorgesehen. Dies verstößt gegen die Prämisse eines dynamischen Umfeldes.

  Eine mögliche Adaptierung besteht in einer sensorbasierten Ausführung. Ein Pfad wird in gewohnter Weise konstruiert und anschließend wird dieser Pfad mit sensorischer Unterstützung abgefahren. Werden dabei anhand der Sensorinformationen (neue) Hindernisse wahrgenommen, die ursprünglich nicht in die a-priori-Planung aufgenommen wurden (bzw. werden konnten), so handelt es sich höchstwahrscheinlich um mobile Hindernisse. Eine simple Möglichkeit wäre, in solch einer Situation die Bewegung einzustellen und abzuwarten, bis das Hindernis den anvisierten Bereich verlassen hat. Dies wäre jedoch ein unsicheres Vorgehen, welches andere Probleme aufwerfen würde.

- Eng mit dem vorigen Punkt verbunden ist die Annahme vollkommener Information. Laut Prämisse liegt unvollkommene Information vor. Mit den daraus resultierenden Folgen für die Bewegungsplanung kann das System per se nicht umgehen. Ein trivialer Lösungsansatz besteht nicht.

3.5.2. Systeminhärente Vorarbeiten

Die bereits erwähnte Verlagerung bestimmten Aufwandes in die Vorbereitungsphase birgt sowohl Vor- als auch Nachteile.

- Die Diskretisierung des Konfigurationsraumes zeigt speziell dann einen positiven Einfluss auf die gesamte Laufzeit, wenn mehrmals in einer statischen Umgebung Teilpfade generiert werden müssen. Inkrementelle oder kontinuierliche Systeme laufen Gefahr, ein identisches Teilproblem (z.B. Kollisionsprüfung) mehrmals berechnen zu müssen. Je öfter auf bereits vorhandene Daten zurückgegriffen werden kann, desto größer ist der Effizienzvorteil.

Ein so erzeugter Gewinn bezüglich der Laufzeit wird wie so oft auch in diesem Fall mit einem erhöhten Speicherbedarf erkauft. Selbstverständlich müssen die vorbereiteten Datenstrukturen im Speicher gehalten werden, um jederzeit Zugriff zu ermöglichen. In Abhängigkeit der verwendeten Hardware kann hier ein Trade-Off, d.h. eine Adaption vorgenommen werden. Dies wird vom System unterstützt, indem an verschiedenen Stellen flexibel zwischen inkrementellen und statischen Vorgehensweisen gewählt werden kann.

- Ähnliches gilt für den Aufbau des Suchgraphen. In diesem Fall allerdings kann ein Suchbaum nicht, wie im vorhergehenden Fall die Repräsentation der Zellen, für sämtliche Probleme in derselben Umgebung verwendet werden. Ist ein Suchbaum vollständig generiert, dann kann dieser nur für *ein und dieselbe Startkonfiguration* verwendet werden. Die Suche nach *mehreren Zielkonfigurationen* ist jedoch möglich und führt in geeigneten Fällen zu verbessertem Laufzeitverhalten. Auch in diesem Fall wird Berechnungsaufwand an den Anfang verlagert.

3.5.3. Präzisionsdilemma

Wie bereits angedeutet, liegt hier ein Umstand vor, welchen man am besten mit dem Begriff *Präzisionsdilemma* beschreiben könnte. Wie bei allen Verfahren der Zelldekomposition lautet die zentrale Frage: Welches Raster muss gewählt werden, um eine Planung mit ausreichender Genauigkeit durchführen zu können? Ein zu groß gewähltes Raster führt womöglich zu gar keinem oder zu einem zu ungenauen Ergebnis. Ein zu kleines Raster hat exponentiell gesteigerte Laufzeiten des Algorithmus zur Folge. Die konkret für ein bestimmtes Problem benötigte Auflösung ist nicht auf triviale Weise ermittelbar. Hinzu kommt die enge Verknüpfung mit den Bewegungsparametern und deren Einfluss auf das Ergebnis.

4. Hybridmodell: Route/Template

Zunächst werden einige Vorüberlegungen angestellt, welche eine genauere Betrachtung der Problemstellung in der Praxis zum Inhalt haben. Die darin gewonnenen Erkenntnisse fördern, in Verbindung mit den in 1.2 festgelegten Anforderungen, ein strukturelles Gerüst zu Tage, welches richtungsweisend für die Grundidee des Verfahrens ist.

Grundsätzlich präsentiert sich das Problem der Bewegungsplanung im gegebenen Umfeld auf zwei Arten:

1. Bewegung auf vorgegebenen Strecken mit einer sehr eingeschränkten Eigenbestimmung der möglichen Manöver.

2. Bewegung auf begrenzten Flächen unter relativ freier und flexibler Auswahl der Manöver.

Betrachtet man beide Fälle genauer, ist festzustellen, dass jeweils sehr spezifische Anforderungen ausschlaggebend sind für die passende Art der Bewegungsplanung.

4.1. Grundidee

Die Grundidee des Verfahrens ist, die in der Praxis anthropogen durchgeführte, inituitive Unterscheidung der angesprochenen Anwendungsfälle in ein Modell für Bewegungsplanung zu übernehmen. Während viele andere Planungsmethoden sich auf *ein* System festlegen und dessen spezifische Vor- und Nachteile für den *gesamten* Planungszeitraum erhalten bleiben, bietet sich hier eine gestaffelte Vorgehensweise an. Selbst bei der Anwendung eines eventuell vorhandenen Verfeinerungschrittes[1] ergeben sich Nachteile gegenüber dem vorliegenden Modell. Die beiden nachfolgenden Abschnitte skizzieren die jeweils unterschiedlich angewendeten Vorgehensweisen.

4.1.1. Vorgegebene Teilstrecken

Der erste Fall repräsentiert das generelle Fahren auf einer vorgegebenen Straße bzw. Fahrbahn und unterliegt damit besonderen Regeln bezüglich der Wege- und Spurwahl, der Geschwindigkeit und weiterer Einflußfaktoren. Hier kann demnach auf eine aufwendige und komplexe *lokale* Bewegungsplanung verzichtet werden. Vielmehr ist eine Wegewahl vergleichbar mit Routenplanern einzusetzen, welche weniger eine mikroskopische als vielmehr eine makroskopische[2] Planung einsetzt. Vorteile bezüglich der Effizienz liegen auf der Hand, da der diskrete Charakter des Regelwerks (Fahrbahnen, Kreuzungen, Abzweigungen, etc.) eine erhebliche

[1] Vgl. hierzu 3.1.5.

[2] Unabhängig vom zu Grunde liegenden Maßstab ist dies unmittelbar mit der Graphentheorie verbunden, da sich die diversen Randbedingungen in einen Graphen (oder eine sehr ähnliche Struktur) überführen lassen.

Vereinfachung darstellt. Im Folgenden soll der Begriff "**Routensystem**" für einen graphenbasierten Planer verwendet werden.

Um dieses System weiter zu veranschaulichen, kann das selbe Modell auch in wesentlich kleinerem Maßstab angewandt werden, beispielsweise bei der Wegewahl auf Parkplätzen bzw. in Parkhäusern. Auch hier liegt ein Regelwerk vor, welches eine Wegewahl stark vereinfacht. Die zentrale Aufgabe des Routensystems ist die Erreichung einer Endposition, welche in *direkter Nähe* zur eigentlichen Zielposition liegt. Unter "direkter Nähe" ist eine Position zu verstehen, welche als Startposition für das lokale Planungsmodell dienen kann. Eine genaue Bestimmung ist durch die Prämissen des jeweiligen lokalen Modells gegeben, welches die Qualität einer Konfiguration diesbezüglich genau definiert. Eine durch diesen globalen Planungsvorgang erreichte Endposition führt unmittelbar zum zweiten der beiden angesprochenen Fälle.

4.1.2. Manövrieren im unmittelbaren Umfeld / Einparken

Nun wird das unmittelbare Umfeld betrachtet. Die einzubeziehende Fläche geht hierbei üblicherweise nicht über einen Umkreis um das Fahrzeug von zwei bis drei Autolängen hinaus, dies entspräche einem Radius von 10-15m (vgl. Abb. 4.1 und 4.2). Dieser Umkreis kann als ein Kriterium der Lokalität dienen, wie es im vorigen Abschnitt angesprochen wurde. Die Tatsache, dass eine Bewegungsplanung nun innerhalb enger Grenzen stattfindet und weit weniger Restriktionen als im vorhergehenden Fall unterliegt, bildet die Motivation für ein grundsätzlich unterschiedliches Vorgehen.

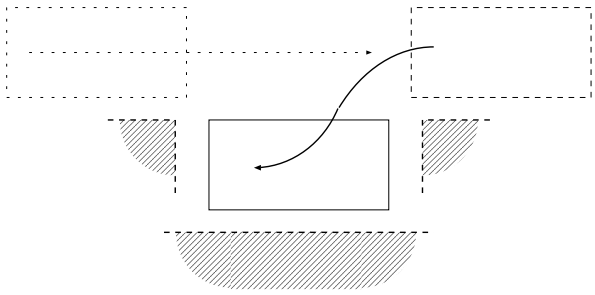

Abbildung 4.1.: Fall 1: paralleles Einparken

Betrachtet man den Fall des Einparkens in der Praxis genauer, stellt man fest, dass es nicht einfach ist, mehr als einige wenige grundlegend unterschiedliche Fälle zu konstruieren. Unter Eliminierung sehr spezieller und unter Zusammenfassung hinreichend ähnlicher Situationen erhält man lediglich 3 Fälle[3] (vgl. Abb. 4.1 u. 4.2):

1. **Parallel:** Dies ist der klassische Anwendungsfall mit keinem oder einem sehr geringen Winkel θ zur Fahrbahnrichtung ($0 \leq \theta \ll \frac{\pi}{4}$).
2. **Vorwärts:** (relativ rechtwinkliges) Einparken mit $\frac{\pi}{4} \leq \theta \leq \frac{3\pi}{4}$.
3. **Rückwärts:**[4] (relativ rechtwinkliges) Einparken mit $\frac{\pi}{4} \leq \theta \leq \frac{3\pi}{4}$.

[3] Eine gegebenenfalls spiegelbildliche Anordnung stellt keine hinreichend unterschiedliche Situation dar, und kann - rein mathematisch, abgesehen von entsprechend geänderten Vorzeichen - identisch behandelt werden.
[4] Dieser Fall sei auf Grund der normalerweise notwendigen Fahrtrichtungsänderung separat betrachtet.

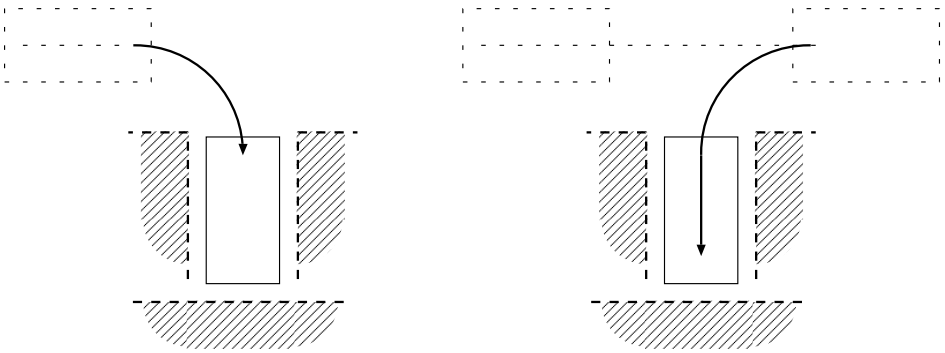

Abbildung 4.2.: Fall 2 und 3: rechtwinkliges Einparken (links vorw., rechts rückw.)

Der Begriff **Template**[5] ergibt sich aus dem gedanklichen Aneinanderreihen von atomaren Basisbewegungen bzw. dem zur Deckung Bringen der Spur des Fahrzeugs, dessen Bewegungsgeometrischer Eigenschaften[6] und den im erfassten Umfeld georteten Hindernissen.

Die detaillierte Modellierung der für die jeweiligen Bereiche relevanten Teilaspekte und der daraus entwickelten Verfahren erfolgt in den folgenden Abschnitten. Zuvor jedoch wird zum einen eine Charakterisierung der Eigenschaften verschiedener Konfigurationen vorgenommen und zum anderen das Problem der Definition geeigneter Distanzfunktionen für Konfigurationen nicht-holonomer Roboter näher betrachtet.

4.2. Charakterisierung von Konfigurationen

Die Notwendigkeit einer Charakterisierung von Konfigurationen entsteht aus dem ambivalenten Zusammenhang, in dem Konfigurationen zu behandeln sind. Prinzipiell wird eine Konfiguration q, gemäß Abschnitt 1.2.1 und 2.1.1, exakt durch die dafür vorgesehenen Parameter (x, y, θ) bestimmt. Diese Angaben können sowohl **relativen** als auch **absoluten** Charakter haben, wie im folgenden Abschnitt gezeigt wird. In der vorliegenden Problemstellung ist jedoch ebenso wahrscheinlich mit einer **näherungsweisen** bzw. **approximativen** Referenz zu rechnen. In Fällen nämlich, in denen keine absolute oder relative Positionsangabe einer Konfiguration gemacht werden kann. Konsequenzen für die Bewegungsplanung und mögliche Lösungsansätze zeigt der darauffolgende Abschnitt.

4.2.1. Relative und absolute Referenzen

Das Vorhandensein relativer Referenzen, zusätzlich zu absoluten Referenzen, stellt eine triviale Erweiterung zu den Definitionen in den Abschnitten 1.2.1 und 2.1.1 dar. Betrachtet man eine von q verschiedene Konfiguration q', kann diese relativ zu q dargestellt werden, indem der Ursprung des Konfigurationsraumes auf $q = (x, y, \theta)$ bzw. den Bezugspunkt R überführt wird. Alle Angaben erfolgen dann *relativ* zu q.

[5] Dt.: Schablone.
[6] Vgl. Abb. 4.8

Beispiel: Um das bereits angesprochene Beispiel des Parkhauses aufzugreifen, kann eine Referenz in folgendem Zusammenhang auftreten. Eine Referenz auf q' wird in Relation zu einem eindeutig definierten Bezugspunkt übermittelt. Die Fahrt in einem Parkhaus wird durch magnetische Marker geleitet, wie sie in Abschnitt 5.3 beschrieben sind. Einer dieser Marker, der, ebenso wie alle anderen, an einer bekannten, festen Position angebracht ist, dient als Bezugspunkt. Eine zuvor vorgenommene, manuelle Vermessung der lokalen Örtlichkeiten liefert *exakte, relative Informationen* über die Zielkonfiguration q' in Bezug auf den Marker. Dieser stellt für die Routenplanung die Zielkonfiguration und für die folgende lokale Bewegungsplanung die Startkonfiguration q dar.

4.2.2. Approximative Referenzen

Approximative Referenzen bringen, bedingt durch ihre nur näherungsweise erfassbaren Parameter, zusätzliche Komplexität in das Problem ein.

Definition: Eine **approximative Referenz** q' liegt dann vor, wenn die exakten Parameter (x', y', θ') nur **mittelbar bestimmbar** sind. Das bedeutet in diesem Zusammenhang, dass eine Bestimmung bzw. Berechnung lediglich **in Relation zu bekannten Parametern** erfolgen können und außerdem **mit einer Restfehlerrate behaftet** sein können.

Beispiel: Das Vorkommen approximativer Referenzen wird im Folgenden am Beispiel des parallelen Einparkens dargestellt. In der Praxis gestaltet sich dieser Vorgang, schematisch dargestellt in Abbildung 4.4, folgendermaßen. Ausgehend von der Startkonfiguration q sei, abgesehen von der ungefähren Lage "in der Lücke rechts vom Fahrzeug", keine weitere, vor allem aber keine exakte Information gegeben. Im Hinblick auf die Bewegungsplanung muss zunächst - zumindest näherungsweise - die Zielkonfiguration q' erfasst werden. Zu diesem Zweck werden Sensorinformationen herangezogen, die ein Abbild der Umgebung zeigen und anhand derer der Zielbereich näher beschrieben werden kann. Ohne Beschränkung der Allgemeinheit und ohne auf verschiedene technische Details einzugehen sei mittels einer Vorbeifahrt entsprechende Sensorinformation erfasst worden (vgl. Abb. 4.3).

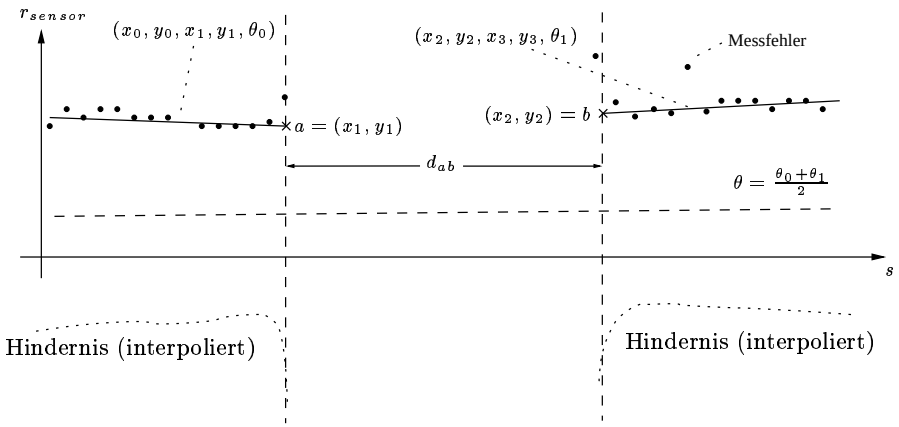

Abbildung 4.3.: Abstandsinformation der Sensorik

Hierbei sind die zurückgelegte Wegstrecke auf der horizontalen und die von den Sensoren in diskreten Intervallen festgestellten Abstände auf der vertikalen Achse angetragen. Schematisch ist zusätzlich unterhalb der s-Achse die interpolierte Form der Hindernisse skizziert. Es handelt sich hierbei um eine willkürlich entworfene, schematische Darstellung zur Veranschaulichung des Vorgehens.

Analytisch können die Parameter der Zielkonfiguration q' in Abhängigkeit der in Abbildung 4.4 dargestellten Punkte a und b festgelegt werden. Hierbei wird unterstellt, dass je ein gleichgroßer Abstand des Fahrzeugs nach vorne und hinten eingehalten werden soll. Soweit eine Begrenzung zur rechten Seite erfassbar ist, soll der seitliche Abstand ebenfalls auf diese Weise festgelegt werden. Anderenfalls dient für die Ausrichtung der linken Fahrzeugseite die Ausdehnung der vorhandenen Hindernisse vor bzw. hinter dem Fahrzeug, so dass die linke Fahrzeugseite "bündig" abschließt.

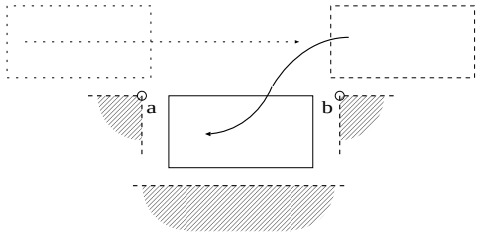

Abbildung 4.4.: Schema paralleles Einparken

Wie jedoch können diese neuralgischen Punkte a und b erfasst werden? Betrachtet man die von der Sensorik gelieferte Information genauer und kombiniert diese mit Daten über die Kinematik des Fahrzeuges, erlaubt dies eine Berechnung bzw. hinreichend genaue Schätzung der benötigten Daten. Hierbei sind besonders folgende Werte von Interesse:

- Lage der **Bezugspunkte** a **und** b, d.h. deren Position bezüglich x und y relativ zum Fahrzeug und daraus resultierend der
- **Abstand** der beiden Punkte, d.h. die Länge der Parklücke.
- **Orientierung** θ des Fahrzeugs in Bezug auf die Fahrbahnrichtung.

In Abhängigkeit der zurückgelegten Wegstrecke werden die Echos bzw. die festgestellten Abstände zu den am rechten Fahrbahnrand abgestellten Objekten aufgezeichnet. Sobald der Gradient des geglätteten Signals einen bestimmten Schwellenwert dauerhaft übersteigt (in Abb. 4.3 im Bereich um a), wird die Ausdehnung des Objektes auf den Punkt a hochgerechnet. Ebenso verfährt man bei der Erfassung des Puntkes b, wobei hier die dauerhafte *Unterschreitung* des Schwellenwertes ausschlaggebend ist. In Relation zu den durch Dead-Reckoning fortgeschriebenen Koordinaten des Fahrzeugs erhält man so die benötigten Parameter der Zielkonfiguration q'. Unterstellt man eine unvollkommen genaue Ausrichtung des Fahrzeugs zur Fahrbahn, kann auch in Bezug auf θ eine Korrektur erfolgen. Hierzu werden wiederum die Gradienten der Sensorinformationen hinzugezogen. Eventuell vorhandene Messfehler gehen durch die Glättung der Daten nur unwesentlich in die Berechnung ein - natürlich in Abhängigkeit der Auftrittshäufigkeit[7].

[7] Eine genauere Betrachtung der bei der Messung möglichen Anomalien ist in Abschnitt 5.1 zu finden.

Zusätzlich zur Berechnung der Parameter von q' erlaubt dieses Vorgehen, in Kombination mit den bekannten fahrzeugspezifischen Parametern, im Voraus eine Entscheidung, ob die anvisierte Zielposition überhaupt erreichbar ist. Liegen die Punkte a und b beispielsweise näher zusammen, als ein bestimmter, fahrzeugspezifischer Schwellenwert, erübrigt sich jegliche Planung, da der Platz für das gewünschte Manöver nicht ausreichen würde.

4.3. Distanzfunktionen

Nicht nur im Hinblick auf eine Bewertung verschiedener Interimskonfigurationen (diese werden im Verlauf dieses Kapitels benötigt), sondern auch um eine Basis für den Vergleich beliebiger Konfigurationen zu schaffen, wird eine Distanzfunktion benötigt. Eine Distanzfunktion für Konfigurationen nicht-holonomer Roboter ist sehr schwierig zu definieren, da sich verschiedene Parameter gegenseitig beeinflussen. In der einschlägigen Literatur findet sich kein vollständiger Lösungsansatz für dieses Problem.

Die beste Eignung im Sinne einer Distanzfunktion hätte die **Länge des optimalen Pfades** zur Verbindung zweier Konfigurationen unter Beachtung der kinematischen Beschränkungen. Nachdem jedoch die gesuchte Distanzfunktion zumindest im vorliegenden Fall *zur Berechnung des optimalen Pfades* benötigt wird, stellt dies keine Option dar. Die gesuchte Funktion sollte also eine möglichst genaue *näherungsweise Berechnung* bzw. *Abschätzung* der Länge des optimalen Pfades gewährleisten, bei gleichzeitig minimalem Aufwand.

Im Folgenden werden verschiedene Ansätze betrachtet und deren Vor- und Nachteile erläutert. Gegeben seien jeweils zwei Konfigurationen q und q' mit den Eigenschaften (x, y, θ) respektive (x', y', θ').

4.3.1. Untere Grenze aller Distanzfunktionen

Im vorliegenden Fall des autoähnlichen Roboters läßt sich zunächst analytisch eine untere Grenze für beliebige Distanzfunktionen definieren. Hierbei kann man zwei Fälle unterscheiden:

1. $\theta = \theta'$: Im günstigsten Fall liegen q und q' auf einer Geraden, deren Winkel θ ist. Dann hat der optimale Pfad die Länge

$$l = d(q, q') = \sqrt{(x' - x)^2 + (y' - y)^2}.$$

2. $\theta \neq \theta'$: Im günstigsten Fall liegen q und q' auf demselben Kreis mit Radius ρ_{min} und die Winkel θ und θ' entsprechen den Winkeln der Tangenten an den Punkten R und R'. Dann hat der optimale Pfad die Länge

$$l = |\theta' - \theta| \rho_{min}.$$

4.3.2. Euklidische Distanz

Die einfache euklidische Distanz

$$d(q, q') = \sqrt{(x' - x)^2 + (y' - y)^2}$$

der Koordinaten x und y unter Vernachlässigung der Orientierung θ ist im Falle nicht-holonomer Vehikel nicht ausreichend. Sie drückt lediglich die reine *Entfernung* der Punkte

R und R' voneinander aus, welche für sich allein genommen im Hinblick auf die Bewegungsplanung nur selten genug Aussagekraft besitzt. Gemäß der Definitionen im vorigen Absatz kann diese Abschätzung nicht über der wirklichen Länge des optimalen Pfades liegen. Der wirkliche Pfad hat demnach im besten Fall genau diese Länge, meistens jedoch ist er erheblich länger. Die euklidische Distanz stellt also eine **optimistische Abschätzung** dar.

Eine mögliche Verbesserung stellt die zusätzliche Berücksichtigung des Winkels $\delta\theta = |\theta' - \theta|$ dar. Hierbei muss eine Umrechnung des Bogenmaßes in die entsprechende Wegstrecke vorgenommen werden:

$$d(q, q') = \sqrt{(x' - x)^2 + (y' - y)^2} + \delta\theta\rho_{min}$$

Der Wert dieser Abschätzung wiederum kann sowohl unter, als auch erheblich über der wirklichen Länge des optimalen Pfades liegen.

4.3.3. Laterale Distanzfunktionen

Eine besondere Schwierigkeit für autoähnliche Roboter stellt die laterale Translation dar. Die fehlende Möglichkeit, Bewegungen seitwärts durchführen zu können, wird durch Kombination verschiedener atomarer Bewegungen kompensiert. Eines der möglichen Manöver wird detailliert in Abschnitt 4.5.1 vorgestellt.

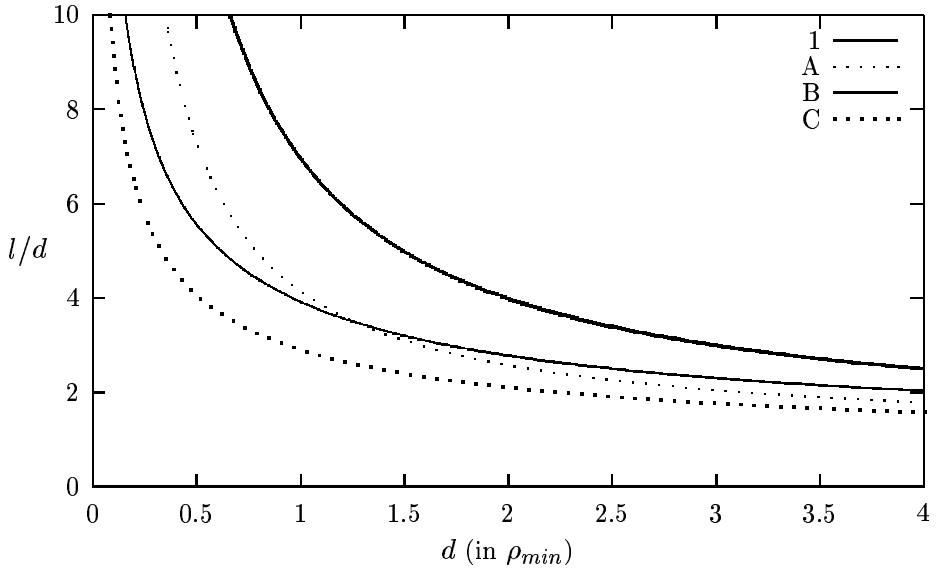

Abbildung 4.5.: Effizienz verschiedener Manöver im Fall lateraler Translation

Zum Zweck eines direkten Vergleichs wurden drei weitere Templates bzw. Manöver entwickelt, deren Effizienz in Abbildung 4.5 graphisch dargestellt ist. Die jeweiligen Graphen ergeben sich aus der Länge des lateralen Versatzes d, angegeben als Vielfaches von ρ_{min}, und aus dem Quotienten $\frac{l}{d}$, dem Verhältnis der Länge des gefunden Pfades l zu d. Die Manöver 1[8] bzw. A bis C sind wie folgt aufgebaut:

[8] Vgl. [Lato 91], Seite 425.

- **Manöver 1:** Je eine Kurvenfahrt am Anfang und am Ende des Pfades, dazwischen eine Geradeausfahrt (vgl. Abb. 4.7 auf Seite 46), mit

$$l = 2\left(\delta\theta + \left(\sin\left(\delta\theta\right)\left(\frac{\rho_{min}}{\cos\left(\delta\theta\right)}\right)\right)\right)$$

  und

$$\delta\theta = \arccos\left(\frac{\rho_{min}}{(d/2 + \rho_{min})}\right)$$

- **Manöver A:** Je eine Kurvenfahrt am Anfang und am Ende des Pfades mit $\delta\theta = \pi/2$, dazwischen eine Geradeausfahrt der Länge d (vgl. Abb. 4.6, links), mit

$$l = (\pi\rho_{min} + d)$$

- **Manöver B:** Je ein Manöver 2 (vgl. Abb. 4.7, rechts) am Anfang und am Ende des Pfades, dazwischen eine Geradeausfahrt der Länge d, mit

$$l = ((\pi\rho_{min} + 4(sin(\frac{\pi}{4})\rho_{min}) + d)$$

- **Manöver C:** Vier gleichlange Kurvenfahrten, jeweils zwei mit entgegengesetztem Lenkeinschlag bzw. Fahrtrichtung (vgl. Abb. 4.6, rechts), mit

$$l = (4\rho_{min}\arccos(\frac{-d}{4\rho_{min}} + 1))$$

  Die Definition des Manövers C gilt in dieser Form nur für alle Entfernungen $0 \leq d \leq 4\rho_{min}$.

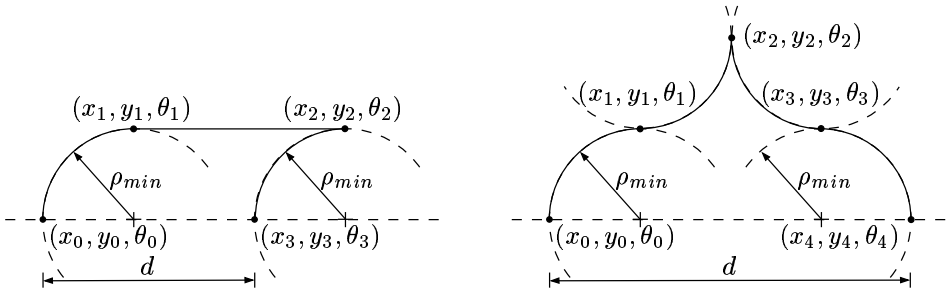

Abbildung 4.6.: Manöver A (links) und Manöver C

Es fällt auf, dass speziell bei sehr geringen Entfernungen $d \leq \frac{\rho_{min}}{2}$ eine relativ schlechte Effizienz vorliegt, bedingt durch die nicht-holonomen Beschränkungen. Außerdem haben alle Manöver die asymptotische Annäherung $l/d \to 1$ gemein:

$$\lim_{d\to\infty} l = d \tag{4.1}$$

Trivialerweise gilt (4.1) für alle denkbaren Templates zur lateralen Translation, in Verbindung mit den Konfigurationen q und q', mit $\theta = \frac{\pi}{2}$, $0 \leq \theta' < 2\pi$ und $y = y'$.

4.3.4. Single-Ident Eigenschaft

Dieser Begriff bezeichnet eine bestimmte Konstellation zweier Konfigurationen, die eine Länge des optimalen Pfades von $\delta\theta\rho_{min}$ garantiert. Das bedeutet, der optimale Pfad besteht nur aus nicht-linearen Wegstücken minimaler Kurvatur und eine Drehung des Roboters um die Hochachse erfolgt stets in *eine* Richtung. Im Fall $\delta\theta < 0$ erfolgt eine Drehung nach links, anderenfalls nach rechts. Für die Beurteilung einer Konstellation müssen in einem ersten Schritt die relevanten Bezugspunkte gewählt werden[9] und in einem zweiten Schritt die (euklidischen) Distanzen zwischen ihnen berechnet werden.

1. Die relevanten Bezugspunkte werden folgendermaßen gewählt:

   Wähle als Bezugspunkt (von q) $\Omega = \Omega_p$ (und $\dot{\Omega} = \Omega_s$), falls

   $$d(\Omega_p, R') < d(\Omega_s, R'),$$

   sonst wähle $\Omega = \Omega_s$ (und $\dot{\Omega} = \Omega_p$).
   Wähle entsprechend als Bezugspunkt (von q') $\Omega' = \Omega'_s$ (und $\dot{\Omega}' = \Omega'_p$), falls

   $$\Omega = \Omega_p$$

   sonst wähle $\Omega' = \Omega'_p$ (und $\dot{\Omega}' = \Omega'_s$).

2. Folgende Distanzen sind für eine Beurteilung ausschlaggebend:

   $$d_\Omega = d(\Omega, \Omega') \text{ und } d_{\dot{\Omega}} = d(\dot{\Omega}, \dot{\Omega}').$$

Wenn gilt:

$$d(\Omega, \dot{\Omega}') = 0 \ , \tag{4.2}$$

dann beträgt die Länge des optimalen Pfades:

$$l = \delta\theta\rho_{min}$$

und die maximale Obergrenze für diese Länge beträgt:

$$l_{max} = \pi\rho_{min}$$

Die Bezeichnung dieser Eigenschaft als **Single-Ident** leitet sich aus der Identität der Bezugspunkte Ω und $\dot{\Omega}'$ ab.

Beweis: Trivialerweise liegt auf Grund von (4.2) R' auf dem ρ_{min}-Kreis um Ω, und demnach liegt ebenso $\dot{\Omega}$ auf dem $2\rho_{min}$-Kreis um Ω. Eine Fahrt in Richtung R' führt nach spätestens $\delta\theta \leq \pi$ dazu, dass $R = R'$ bzw. $\dot{\Omega} = \Omega'$, q.e.d.

[9] Ein Beispiel für die ordnungsgemäße Benennung der Bezugspunkte ist in den Abbildungen 4.9 (S. 49) und 4.10 (S. 51) illustriert.

4.3.5. Double-Overlap Eigenschaft

Dieser Begriff bezeichnet eine bestimmte Konstellation zweier Konfigurationen, die ebenfalls eine Länge des optimalen Pfades von $\delta\theta\rho_{min}$ garantiert. Die Merkmale des Pfades gelten ebenso wie das Setzen der Bezugspunkte analog dem vorhergehenden Fall, Abschnitt 4.3.4. Wenn gilt:

$$d_{\Omega} \leq 2\rho_{min} \text{ und } d_{\dot{\Omega}} \leq 2\rho_{min} \ , \tag{4.3}$$

dann beträgt die Länge des optimalen Pfades:

$$l = \delta\theta\rho_{min}$$

und die maximale Obergrenze für diese Länge beträgt:

$$l_{max} = \pi\rho_{min}$$

Die Bezeichnung dieser Eigenschaft als **Double-Overlap** leitet sich aus dem Überlappen der sich gegenüberliegenden[10] ρ_{min}-Kreise beider Konfigurationen ab. Sie ist nur dann erfüllt, wenn beide Kreise sich gemäß (4.3) tangieren oder überlappen. Ferner kann diese Eigenschaft nur dann bestehen, wenn auch gilt $d(R, R') \leq 2\rho_{min}$. Dies ist unter Umständen ein geeignetes Kriterium für den frühzeitigen Ausschluss von Konstellationen bzw. Konfigurationen.

Beweis: Der Schlüssel für einen analytischen Beweisansatz liegt offensichtlich in der Voraussetzung (4.3). Ein Pfad minimaler Länge setzt sich in diesem Fall aus einzelnen Stücken minimaler Kurvatur zusammen, da keine Distanzen $d(R, R') > 2\rho_{min}$ überbrückt werden müssen. Die Voraussetzung (4.3) hat zur Folge, dass bei einer Drehung um Ω (bzw. $\dot{\Omega}$) in Richtung $\delta\theta < \pi$ entweder

$$d_{\Omega} = 2\rho_{min}$$

oder

$$d_{\dot{\Omega}} = 2\rho_{min}$$

erreicht werden kann. Dies wiederum bedeutet, dass nach einer weiteren Drehung um $\dot{\Omega}$ (bzw. Ω) in gleicher Richtung mit Betrag $\delta\theta' < \delta\theta$ die Punkte Ω und $\dot{\Omega}'$ (bzw. $\dot{\Omega}$ und Ω') zur Deckung gebracht werden können. Eine weitere Drehung um die zur Deckung gebrachten Punkte führt dann zwangsläufig zur Deckung von R und R', da anfänglich entweder

$$d(\Omega, \dot{\Omega}') = 2\rho_{min} \text{ und } d(\dot{\Omega}, \Omega') = 0$$

oder

$$d(\dot{\Omega}, \Omega') = 2\rho_{min} \text{ und } d(\Omega, \dot{\Omega}') = 0$$

gilt, und auf Grund von (4.2) gilt der Beweisansatz zu 4.3.4.

[10] Da sich die Bezeichnung "Double-Overlap" auf die seiten-ungleiche *Benennung* der Bezugspunkte bezieht, bedeutet eine Überlappung der Kreise um identisch benannte Bezugspunkte die Überlappung "gegenüberliegender" Kreise. D.h. in diesem Fall, dass jeweils der ρ_{min}-Kreis auf der *Backbordseite* der einen Konfiguration mit dem ρ_{min}-Kreis auf der *Steuerbordseite* der anderen verglichen wird.

4.4. Modellierung des Routensystems

Man stelle sich vor, der Roboter fahre auf einem virtuellen Gleis[11], dann reduzierte sich der planungstechnische Aufwand auf einfache, diskrete Entscheidungen an fest definierten Punkten (Abzweigungen, Kreuzungen, etc.) wie beispielsweise "links" oder "geradeaus". Einen Weg durch ein entsprechend definiertes Umfeld, (d.h. Straßen, Abzweigungen, Kreuzungen, etc.) könnte man dann durch einen äquivalentenGraphen beschreiben. Die Bewegungs- oder vielmehr die Routenplanung[12] reduziert sich somit auf die Suche nach dem kürzestem Weg in einem gerichteten, gewichteten Graphen.

Probleme dieser Art sind bezüglich unterschiedlichster Aspekte bereits seit geraumer Zeit ausführlich in der Literatur behandelt worden, zur Vertiefung sei unter anderem auf [Dask 95], [Doms 90], [LMW 88] oder [FMW 92] verwiesen, um nur wenige Beispiele zu nennen.

Nachdem das eigentliche Planungsproblem für diesen Anwendungsfall bereits hinreichend gelöst scheint, stellt sich die Frage nach dem Brückenschlag zwischen dem theoretischen Vorgehen und der Praxisintegration. Für diesen Zweck muss eine veränderte und/oder zusätzliche Infrastruktur geschaffen werden, welche das Bindeglied zwischen dem autonom agierenden Fahrzeug und der Umwelt darstellt, d.h. das bereits angesprochene virtuelle Gleis muss gelegt werden. Hierbei bieten sich mehrere Möglichkeiten und Kombinationen an, beispielsweise das satellitengestützte GPS-System und/oder Dead-Reckoning. Ein bereits experimentell vorhandener Ansatz auf Basis induktiver Sensoren zeigt, wieviele weitere Möglichkeiten in der Nutzung einer parallel vorhandenen Infrastruktur liegen. Eine eingehende Betrachtung dieses experimentellen Systems findet sich in Abschnitt 5.3. Die Verknüpfung dieser Technologie mit einem Routen- bzw. Tourenplaner komplettiert den ersten Teilbereich des Planungsmodells. Für einen tiefergehenden Einblick sei auf die bereits angegebene Literatur bzw. auf Abschnitt 5.3 verwiesen. Der gesamte verbleibende Teil dieses Kapitels hat aufgrund der konkreteren Relation zur Aufgabenstellung deshalb speziell das hier neu entworfene Templatesystem zum Inhalt.

4.5. Modellierung des Templatesystems ohne Kollisionsbehandlung

Der Kerngedanke des Vorgehens ist die Erfassung von Mustern oder Templates, welche die verschiedenen Anwendungsfälle in ihrer essenziellen Eigenschaft kennzeichen. Im eigentlichen Planungsschritt werden die einzelnen Templates dann mittels geeigneter atomarer Bewegungen substituiert und aus diesen der Pfad konstruiert.

Zunächst wird ein Ansatz ohne Berücksichtigung von Hindernissen vorgestellt. Der in diesem Abschnitt dargestellte Algorithmus erlaubt die Konstruktion des optimalen Pfades anhand eines geometrisch-modellbasierten Vorgehens. Beispiele für die Anwendung des Algorithmus auf zwei verschiedene Konstellationen finden sich in den Abbildungen 4.10 und 4.11. In späteren Abschnitten wird das Modell dann um eine Umfelderfassung und Hindernisbehandlung erweitert.

[11] Ein System zur Realisierung solcher Automation existiert zumindest experimentell, wie in Abschnitt 5.3 beschrieben.

[12] Verschiedentlich auch: Tourenplanung

4.5.1. Templates

Um dem Begriff *Template* mehr Plastizität zu verleihen, betrachte man die sogenannten Manöver vom Typ 1 und vom Typ 2 (Abb. 4.7), wie sie in [Lato 91], S. 425 ff., zu finden sind. Diese beschreiben, wie verschiedene, auf Grund nicht-holonomer Beschränkungen unmögliche, Bewegungen des Roboters mittels Aneinanderreihen erlaubter (atomarer) Bewegungen zu substituieren sind. Auf diese Art und Weise sind beispielsweise sowohl laterale Translation (1) als auch Rotation (2) möglich.

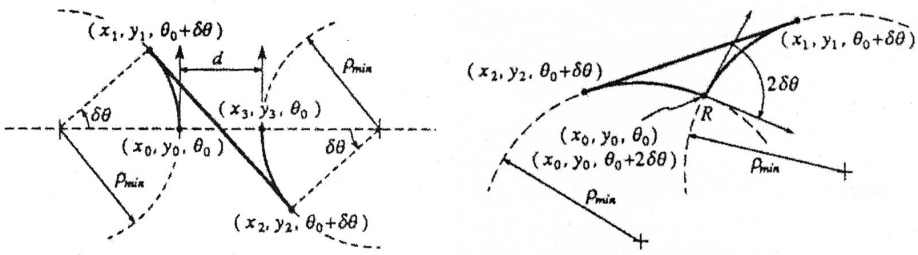

Abbildung 4.7.: Manöver vom Typ 1 (links) und Typ 2

Im Rahmen des entwickelten Planungsmodells repräsentieren diese beiden Beispiele je ein Template. Die Lösung dieser Problemstellungen, das heißt die Konstruktion eines geeigneten Pfades, liegt in der Generierung der atomaren Bewegungen.

Prämisse für die Planung ist unvollkommene Information, was bedeutet, dass anfangs lediglich die Startposition und in Relation dazu die Zielposition bekannt ist. Kenntnis über eventuelle Hindernisse besteht ex ante nicht. In verschiedenen Fällen ist es möglich, in einem frühen Stadium der Ausführung (nicht der Planung) Hindernisse als solche über die Sensorik zu erkennen. Diese Information kann dann unmittelbar in die weitere Planung/Ausführung mit einbezogen werden, was im Falle dynamisch auftretender Hindernisse ohnehin über den gesamten Planungs- und Ausführungshorizont geschehen muss. Nichtsdestotrotz ist die Erkennung von Hindernissen um so nützlicher, je früher sie stattfindet. Zur anschaulicheren Darstellung des unterliegenden Planungsalgorithmus soll zunächst jedoch von Hindernissen, seien sie dynamischen Charakters oder nicht, abgesehen werden.

Eingangs sei nochmals eine grundlegende Besonderheit des Modells angesprochen, nämlich die Tatsache, dass es sich in Kontrast zu vielen anderen Verfahren nicht um ein diskret getrenntes Planungs- und Ausführungsmodell handelt. Entsprechend der in 1.2 angeführten Anforderungen handelt es sich hierbei um ein stark echtzeitorientiertes Modell, welches keine ausgeprägte Trennung von Planung und Ausführung anstrebt. Vielmehr ist einer der Hauptaspekte des Verfahrens die permanente Steuerung und Kontrolle der Parameter und eine flexible Struktur zur Behandlung von dynamischen Effekten. Eine diskrete Abstufung ist insofern vorhanden, dass planungsgemäß eine Zäsur nur zwischen zwei atomaren Bewegungen vorgesehen ist. Im Fall von dynamischen Hindernissen, wie sie in einem späteren Abschnitt behandelt werden, kann selbst eine atomare Bewegung für eine entsprechende Behandlung unterbrochen werden.

4.5.2. Bewegungsgeometrie

Wie schon oben beschrieben, geht primär das unmittelbare Umfeld in die Planung ein. Um dies zu konkretisieren, betrachte man Abb. 4.8. Demnach kann eine Unterteilung in folgende Flächenkategorien vorgenommen werden (mit $A = A_b \cup A_s \setminus T$ und $T = T_p \cup T_s$):

- **innerhalb der Wendekreise**[13]: T_p und T_s
- außerhalb T und **innerhalb des** $2\rho_{min}$**- Radius**[14]: A_b und A_s
- **sonstiger Bereich**: außerhalb A und T: O

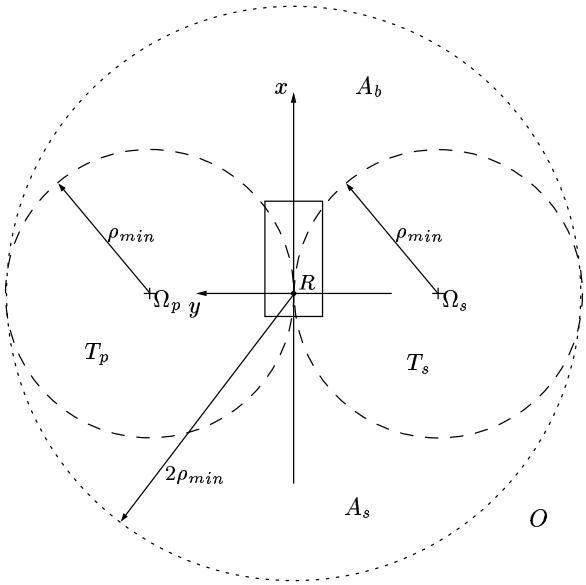

Abbildung 4.8.: Bewegungsgeometrie des Roboters

Ausschlaggebend für die Art der Planung ist ein Regelwerk, welches sich aus der Lage von Start- und Zielposition ergibt, die mit q und q' (Gemäß Kapitel 3 setze man $q = q_{init}$ und $q' = q_{goal}$.) benannt werden. Hierbei bezeichnen O, A und T die Bereiche um R und O', A' und T' die Bereiche um R'. Es sind verschiedene Konstellationen möglich:

1. $R \in O'$ (und folglich $R' \in O$)
2. $R \in A'$ und $R' \in A$; $R \notin T'$ und $R' \notin T$
3. $R \in A'$ und $R' \in T$; $R \notin T'$
4. $R \in T'$ und $R' \in A$; $R' \notin T$
5. $R \in T'$ und $R' \in T$

[13] T für "turning" ("p" für port, engl. Backbord, links und "s" für starboard, engl. Steuerbord, rechts)
[14] A für "approach" ("b" für bow, engl. Bug/vorne und "s" für stern, engl. Heck/hinten)

Der erste Fall bedingt auf Grund der großen Entfernung zwischen q und q' ein globaler orientiertes Planungsmodell und soll deshalb vernachlässigt werden. Eine eventuelle Überführung der Konfiguration q mit $R \notin O'$ in eine Interimskonfiguration q_i mit $R_i \in O'$ sollte nach einem System entsprechend Abschnitt 4.4 in jedem Fall erreichbar sein, da hier die Schnittstelle zwischen den beiden Planungsmodellen anzusiedeln ist.

4.5.3. Atomare Bewegungen

Alle weiteren Kombinationen können unmittelbar anhand des Templatesystems behandelt werden. Hierbei bedient man sich atomarer Bewegungen, konform der nicht-holonomen Beschränkungen:

- **Fahren geradeaus**
- **Fahren einer Kurve** mit vollem Lenkeinschlag

Der kürzestmögliche Pfad zwischen zwei Konfigurationen muss grundsätzlich aus diesen atomaren Bewegungen konstruiert sein, da die kürzeste Verbindung zweier Punkte eine Gerade ist und die größtmögliche Gierrate (Drehung des Fahrzeugs um die Hochachse zur Angleichung zweier unterschiedlicher Winkel θ und θ' der Konfigurationen q respektive q') bei maximalem Lenkeinschlag vorliegt[15].

4.5.4. Generierung / Algorithmus

Der Algorithmus zur Konstruktion des optimalen Pfades aus einzelnen atomaren Bewegungen unter Vernachlässigung von Hindernissen ist aufgebaut, wie im folgenden Absatz beschrieben (vgl. hierzu die Abbildungen 4.9, 4.10 und 4.11, der optimale Pfad ist rot gekennzeichnet). Die Besonderheit dieses Modells liegt in der Konzentration auf die Punkte Ω_p und Ω_s (sowie deren Umkreise mit Radius ρ_{min}) und nicht etwa auf Relationen, welche R und R' betreffen. Des Weiteren ist der Betrag der Geschwindigkeit v_0 für die Planung nicht weiter ausschlaggebend, für die spätere Umsetzung der verschiedenen Manöver hingegen schon. Näheres hierzu in den Abschnitten 4.6 und 4.7.

Algorithmus:

Sei $d(X, Y)$ die euklidische Distanz der Punkte X und Y. Die Funktion $sgn(x)$ sei die Signum-Funktion und liefere -1 falls $x < 0$, 0, falls $x = 0$ und 1 sonst.

1. **Setzen der Bezugspunkte:**
   Wähle als Bezugspunkt (von q) $\Omega = \Omega_p$ (und $\dot{\Omega} = \Omega_s$), falls

   $$d(\Omega_p, R') < d(\Omega_s, R'),$$

   sonst wähle $\Omega = \Omega_s$ (und $\dot{\Omega} = \Omega_p$).
   Wähle entsprechend als Bezugspunkt (von q') $\Omega' = \Omega'_s$ (und $\dot{\Omega}' = \Omega'_p$), falls

   $$\Omega = \Omega_p$$

   sonst wähle $\Omega' = \Omega'_p$ (und $\dot{\Omega}' = \Omega'_s$).

[15] Vgl. dazu [Lato 91], Kapitel 9, Abschnitt 5.2.

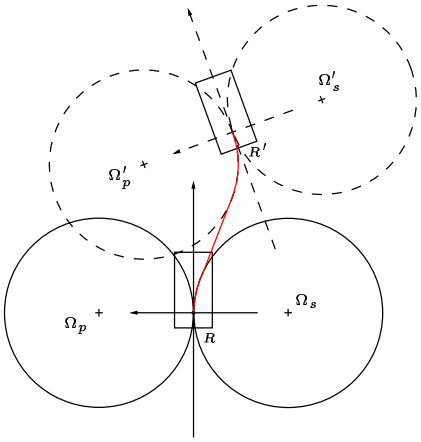

Abbildung 4.9.: Bezugspunkte zweier Konfigurationen

2. **Setzen der bewegungsrelevanten Parameter:**
   Setze
   $$v = \begin{cases} v_0 * sgn(R'_x) & \text{, falls } d(\Omega, \Omega') \geq (2\rho_{min}) \\ -v_0 * sgn(R'_x) & \text{, sonst.} \end{cases}$$
   (bezogen auf R als Ursprung des kartesischen Systems). Anschließend setze $v = -v$, falls
   $$sgn(R'_y) < 0 \text{ und } \frac{\pi}{2} < \theta' < \pi, \text{ oder falls } sgn(R'_y) > 0 \text{ und } \pi < \theta' < \frac{3\pi}{4}.$$
   Setze außerdem
   $$\phi = \begin{cases} sgn(R'_y) * \phi_{max} & \text{, falls } d(\Omega, \Omega') \geq (2\rho_{min}) \\ (-1) * sgn(R'_y) * \phi_{max} & \text{, sonst,} \end{cases}$$
   d.h. Lenkeinschlag in Richtung R', es sei denn, es besteht eine Überlappung der ρ_{min}-Kreise um Ω und Ω', dann Lenkeinschlag entgegengesetzt.

3. **Ausführen der Bewegung:** Nun erfolgt eine Bewegung des Roboters mit den festgelegten Parametern v und ϕ, in Abhängigkeit von $d(\Omega, \Omega')$:

   a) $d(\Omega, \Omega') > 2\rho_{min}$: Bewegung fortführen, bis
   $$sin(\beta) = 1,$$
   mit $\beta = \angle(\Omega, \dot{\Omega}, \Omega')$. Dann setze $\phi = 0$ und führe Bewegung fort, bis
   $$d(\dot{\Omega}, \Omega') = 0.$$
   Dann erneut beginnen bei 1.

   b) $d(\Omega, \Omega') = 2\rho_{min}$: Bewegung fortführen, bis
   $$d(\dot{\Omega}, \Omega') = 0.$$
   Dann erneut beginnen bei 1.

c) $d(\Omega, \Omega') < 2\rho_{min}$: Bewegung fortführen, bis entweder

$$d(\Omega, \Omega') = 2\rho_{min}, \tag{4.4}$$

oder

$$sin(\beta) = 1, \tag{4.5}$$

mit $\beta = \angle(\dot{\Omega}', \Omega, \dot{\Omega})$. Anschließend setze $v = -v$, und

$$\phi = \begin{cases} -\phi & \text{, im Fall (4.4). Dann Bewegung fortführen, bis } d(\dot{\Omega}, \Omega') = 0. \\ 0 & \text{, im Fall (4.5). Dann Bewegung fortführen, bis } d(\Omega, \dot{\Omega}') = 0. \end{cases}$$

Dann erneut beginnen bei 1.

4. Falls zu irgendeinem Zeitpunkt gilt

$$d(R, R') = 0,$$

kann der Algorithmus beendet werden, das Ziel ist dann erreicht.

Eine schematische Darstellung der einzelnen Schritte analog zu dieser Aufzählung ist in Abbildung 4.10 (keine Überlappung) bzw. 4.11 (vorhandene Überlappung) dargestellt. Mit diesem Modell sind, abgesehen von der Behandlung eventueller Hindernisse, sämtliche Konfigurationen ineinander überführbar. Es ist keine Konfiguration q' mit $R' \notin O$ denkbar, welche nicht durch einen so geschaffenen Pfad mit q verbunden werden kann. Im folgenden Abschnitt hierzu ein analytischer Beweisansatz.

Beweis

I **Schritt 1:** Die Festlegung der Bezugspunkte nach 1 sorgt dafür, dass eine gespiegelte Benennung der Ω vorliegt. Eine seitengleiche Benennung ist per definitionem ausgeschlossen.

II **Schritt 2:** Eine genauere Betrachtung dieses Regelwerkes liefert die Einsicht, dass grundsätzlich eine Bewegung in Richtung R' durchgeführt wird, mit Lenkeinschlag in Richtung R'. Ausnahmen stellen 2 Fälle dar:

a) Es besteht eine Überlappung zwischen den ρ_{min} Kreisen um Ω bzw. Ω'. Dann erfolgt zunächst eine "Kurskorrektur", d.h. eine Ausrichtung von θ in Richtung R' durch eine Bewegung von R' weg, mit Lenkeinschlag wegweisend von R'. Diese Bewegung ist allerdings nur von kurzer Dauer und endet spätestens, wenn $d(\Omega, \Omega') = 2\rho_{min}$. Dieser Fall tritt sicher ein, da anfangs $d(\Omega, \Omega') + \varepsilon = 2\rho_{min}$ (sei $\varepsilon < 2\rho_{min}$ der Betrag der Überlappung) und sich $d(\Omega, \Omega')$ durch die festgelegte Bewegung im Bereich von $2\rho_{min} \pm \varepsilon$ variieren lässt.

b) Der Winkel θ' ist um mindestens $\frac{\pi}{2}$ von θ verschieden und weist in Richtung der q-orientierten x-Achse (vgl. hierzu Abb. 4.8 bzw. 4.9). Diese Konstellation birgt erneut zwingend eine Überlappung der ρ_{min} Kreise um Ω bzw. Ω' und die Initialbewegung terminiert aus denselben Gründen, wie im vorigen Absatz (IIa) beschrieben.

III **Schritt 3:** Angenommen der Algorithmus würde in Schritt 3 nicht terminieren, d.h. keine der angeführten Bedingungen könnte erfüllt werden. Das würde bedeuten,

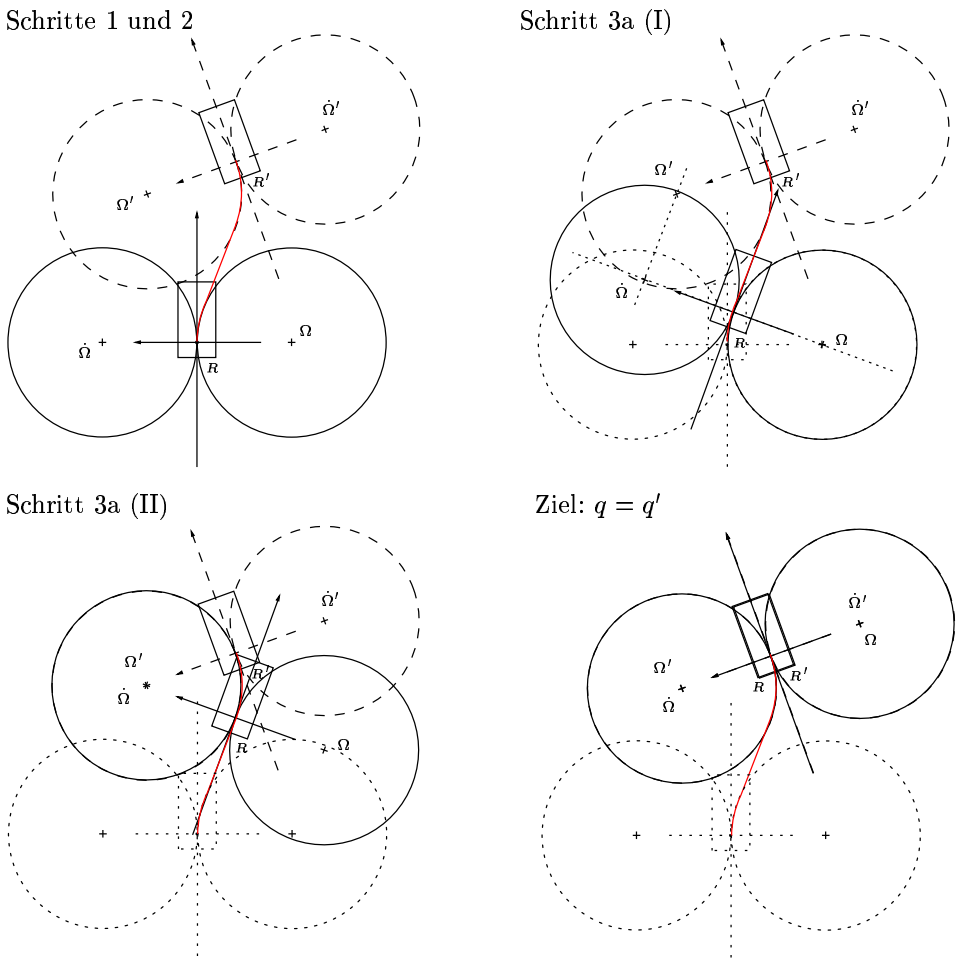

Abbildung 4.10.: Illustration des Algorithmus (Beispiel für $d(\Omega, \Omega') > 2\rho_{min}$)

a) **Punkt 3a:** dass es zwei Konfigurationen q und q' gäbe, für welche die Bedingung nicht erfüllbar wäre. Trivialerweise jedoch ist $sin(\angle(\Omega', \dot{\Omega}, \Omega)) = 1$, für zwei beliebige Konfigurationen q und q' (in Abhängigkeit von θ bzw. θ') in der Ebene erfüllbar, da jeweils gilt $\exists\theta$ für welches gilt $sin(\angle(\Omega', \dot{\Omega}, \Omega)) = 1$ für beliebige $0 \leq \theta' < 2\pi$. Die anschließende Bewegung in Richtung R' mit $\phi = 0$ führt wegen $sin(\angle(\Omega', \dot{\Omega}, \Omega)) = 1$ zwangsläufig zu einer Konfiguration q für die gilt $d(\dot{\Omega}, \Omega') = 0$.

b) **Punkt 3b:** dass bei einer Bewegung um Ω gelten müsste $\nexists\theta$, mit $d(\dot{\Omega}, \Omega') = 0$. Auf Grund der Voraussetzung von 3b ist jedoch garantiert, dass $\exists\theta$, mit $d(\dot{\Omega}, \Omega') = 0$, q.e.d.

c) **Punkt 3c:** dass bei einer Bewegung R um $\dot{\Omega}$ keine Konfiguration q mit Winkel θ erreicht werden könnte, für die gilt $d(\Omega, \dot{\Omega}') = (2\rho_{min})$. Da jedoch wegen 2 gilt $d(\dot{\Omega}, \Omega') \neq 0$ und $d(\dot{\Omega}, \Omega') < 4\rho_{min}$ muss gelten $\exists\theta$ mit $d(\Omega, \dot{\Omega}') = (2\rho_{min})$. Es gilt IIIa analog. Im zweiten Teil von 3c wird anschließend die Umkehrung der Parameter v und ϕ aus 2 rückgängig gemacht, wobei darauf ein Beweis analog IIIb folgt.

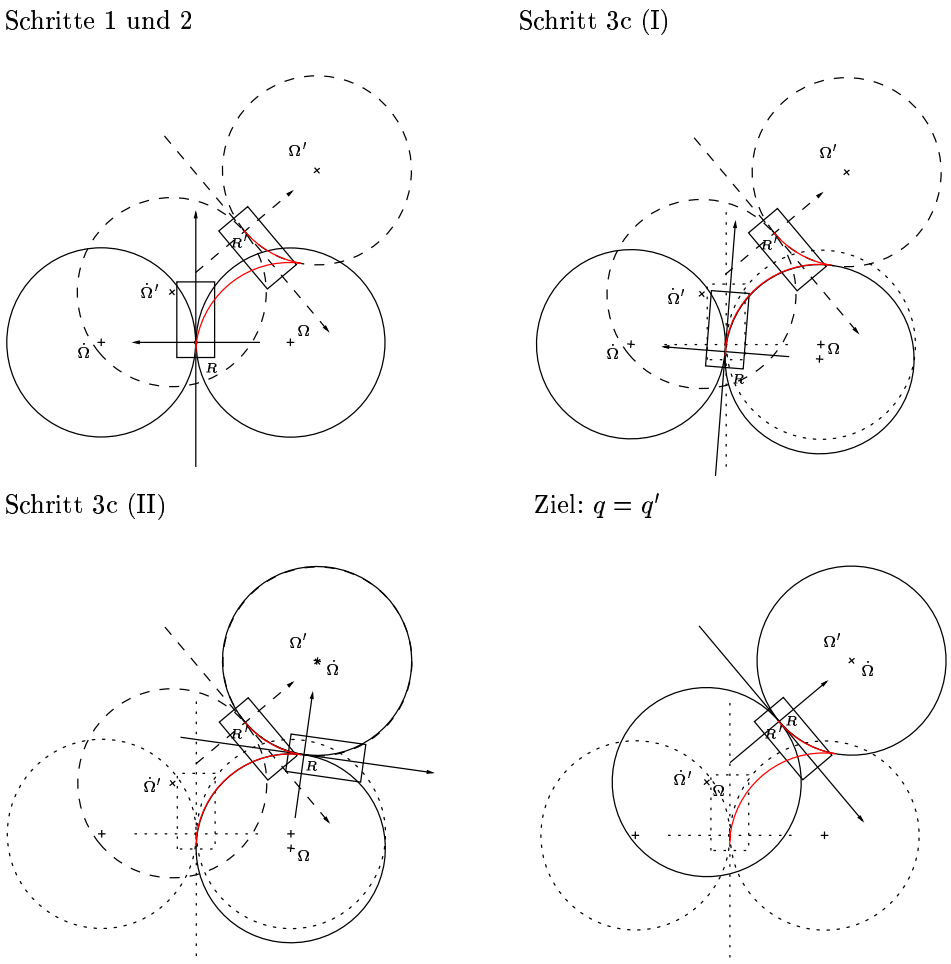

Abbildung 4.11.: Illustration des Algorithmus (Beispiel für $d(\Omega, \Omega') < 2\rho_{min}$)

4.6. Modellierung der Umfelderkennung

Im Laufe der Anwendung jeglicher Verfahren zur Bewegungsplanung stellt sich grundsätzlich früher oder später die Frage nach der Umfelderkennung - systembedingt entweder a priori oder inkrementell. Die für das Templatesystem anvisierte dynamische Steuerung ist abhängig von einer ebenso dynamischen - also inkrementellen - Erfassung der Umwelt. Obwohl ein Straßenfahrzeug mit einer ausreichenden Ausstattung an Sensoren noch nicht in der Serie realisiert wurde, erscheint ein solches Vorhaben zumindest prototypisch möglich. Aktuell verfügbare Systeme aus dem Serien-Automobilbau erfüllen die nötigen Anforderungen jedenfalls nur sehr bedingt (vgl. Abschnitt 5.2.2).

4.6.1. Technische Grundlagen

Die umfassende Diskussion einer technischen Realisierung eröffnet ein völlig neues Themengebiet und ist nicht im Rahmen dieser Ausarbeitung durchführbar. Solange die Anforderungen an die Sensorik in ausreichender Weise formal festgesetzt werden, haben solche rein techni-

schen Fragen allerdings keinen entscheidenden Einfluß auf die Entwicklung des vorliegenden Planungsmodells. Der folgende Abschnitt behandelt daher eine **abstrakte Leistungsbeschreibung** mit dem Ziel der Definition einer theoretischen Mindestausstattung an Sensorik, welche das Modell voraussetzt.

Um relevante Aspekte - soweit umfänglich vertretbar - zumindest ansatzweise zu erfassen, findet sich in Abschnitt 5.1 eine konzise Einführung zum Thema Sensorik. Hier werden verschiedene **konstruktive Aspekte** (5.1.1) beleuchtet und anhand von momentan in der Forschung verwendeten Geräten illustriert. Anschließend wird ein weiterer signifikanter Aspekt exemplarisch behandelt, stellvertretend für den **rechtlichen Bereich** (5.1.2).

4.6.2. Leistungsbeschreibung der Sensorik

Primäre Anforderung ist die Überwachung des unmittelbaren Umfeldes um das Fahrzeug. Dabei muss eine lückenlose Erfassung stattfinden, welche sicher stellt, dass eine Berührung mit dynamischen und/oder statischen Objekten im Bewegungsraum verhindert werden kann. Es genügt also nicht, das Fahrzeug hinsichtlich der primären Bewegungsrichtungen vorne und am Heck mit Sensoren zu bestücken, sondern ebenfalls an der Seite.

Eine der Hauptschwierigkeiten in diesem Zusammenhang ist die Kombination verschiedener Sonsortechnologien. Prinzipiell gewünscht wäre eine kontinuierlich arbeitende Einheit, welche in Echtzeit einen kombinierten Datenstrom von hoher Genauigkeit liefert. Verfügbar sind allerdings meist nur solche Komponenten, deren Funktionsweise konkurrierende Merkmale zu Gunsten *eines* Merkmals optimiert. D.h. diese Komponenten arbeiten entweder sehr genau aber dafür nur in diskreten Abständen oder aber kontinuierlich, jedoch dafür mit geringerer Genauigkeit oder anderen Defiziten.

Für die vorliegende Ausarbeitung wird davon ausgegangen, dass sowohl ausreichende **angulare**, als auch **radiale Information** über das gesamte *sichtbare* Umfeld vorliegt. Das heißt in diesem Zusammenhang, dass Hindernisse, welche im "Schatten" anderer Hindernisse liegen, nicht erfasst werden können. Dieser Umstand birgt auf Grund des inkrementellen Vorgehens jedoch kein Problem. Des Weiteren soll keine Abhängigkeit von der vertikalen Ausdehnung[16] bestehen, ebenfalls soll die spezielle Formgebung eines Hindernisses keine Anomalien erzeugen (vgl. Abschnitt 5.2.1).

Abbildung 4.12 enthält simulierte Daten, die zeigen, wie eine Erfassung der radialen und angularen Information aussehen könnte. Die angulare Komponente ist auf der x-Achse und die jeweilige (radiale) Entfernung zum nächstliegenden Hindernis auf der y-Achse angetragen. An einem Echo dieser Form kann die bevorzugte Bewegungsrichtung abgelesen werden. Im simulierten Fall erlaubt eventuell eine Drehung[17] des Fahrzeuges um $\delta\theta \approx -\pi/2$ (nach links) eine weitere Bewegung aus der ansonsten eingegrenzten Umgebung. Deutlich sind hier Hindernisse im Abstand von ca. 0.5 bis 2.5 Metern auszumachen, die vor und hinter dem Fahrzeug bzw. rechts davon liegen. Ziel einer Drehung wäre die Verbesserung des in Abschnitt 4.7.3 eingeführten Freiheitsgrades der vorliegenden Konfiguration.

[16] Hiermit ist gemeint, dass die vertikale Ausdehnung eines Objektes oder dessen Größe nicht entscheidend sein sollen. Ein 15 *cm* hoher Randstein soll ebenso erfasst werden, wie eine Wand derselben horizontalen Ausdehnung an derselben Stelle.

[17] Nachdem ein Drehen auf der Stelle nicht zugelassen ist, werden hier die substituierenden atomaren Bewegungen unterstellt. Voraussetzung für die atomaren Bewegungen ist eine ausreichende lokale Bewegungsfreiheit.

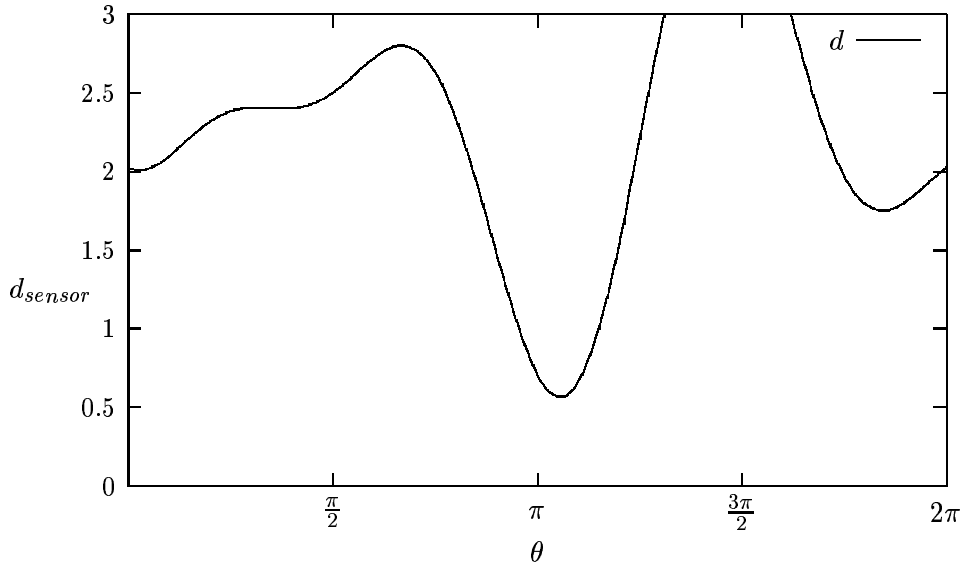

Abbildung 4.12.: Echo der Sensoren (radial / angular)

4.7. Modellierung der Hindernisbehandlung

Der Hindernisbegriff der vorliegenden Problematik unterscheidet sich insofern vom klassischen Hindernisbegriff, als dass die Bandbreite der zu erwartenden Konstellationen relativ gering ist. Während das Umfeld für den Einsatz eines Roboters beispielsweise auf dem Mars eine äußerst komplexe Vielfalt an verschiedensten Hindernissen unterschiedlicher Formen, Größen und Verteilungen aufweist [JPLM], sind im vorliegenden Fall grundlegend unterschiedliche Prämissen zu beachten. Diese bringen in verschiedener Hinsicht eine mögliche Vereinfachung mit sich.

4.7.1. Hindernisse

Wie schon in 1.2.3 angedeutet, haben die hier vorkommenden Hindernisse einen binären Charakter. Dies ist im System des Straßenverkehrs verankert, der von Hindernissen geprägt ist, welche entweder vorhanden sind, oder nicht. Hindernisse, die nur unter bestimmten Umständen als solche behandelt werden müssen, oder einen anderweitig "durchlässigen" Charakter besitzen, sind in Grenzfällen zwar vorstellbar, jedoch für gewöhnlich kein essenzieller Bestandteil der *lokalen* Bewegungsplanung. Bezüglich der vorgelagerten und nicht weiter beschriebenen Routenplanung haben "semi-permeable" Hindernisse einen höheren Stellenwert. Beispielsweise sind Verbindungen denkbar, die nur in einer bestimmten Bewegungsrichtung benützt werden dürfen, oder solche, die einer zeitlichen Restriktion unterliegen (z.B. Nachtfahrverbot).

Ähnliche Möglichkeiten der vereinfachten Modellierung bestehen bezüglich der Formgebung der Hindernisse. Üblicherweise wird eine Darstellung in Form konvexer Polygone gewählt. In Anbetracht der Grenzen sensorischer Erfassung, welche zusätzlich verschiedenen Techniken der Glättung bzw. Interpolation unterliegt, und der nur bedingten Relevanz für das eigentliche Planungsproblem, ist diese vereinfachte Modellierung ausreichend.

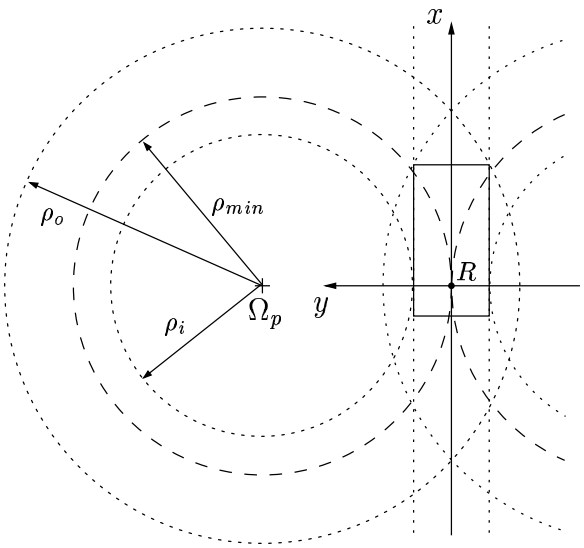

Abbildung 4.13.: Kollisionsgeometrie des Roboters

4.7.2. Kollisionsgeometrie

Zur anschaulichen Behandlung von Hindernissen in Bezug auf die Bewegungen des Fahrzeugs wird der Begriff der **Kollisionsgeometrie** eingeführt. Abbildung 4.13 zeigt die primären für die Kollisionsbehandlung relevanten Bereiche. Diese sind begrenzt durch die inneren und äußeren Wendekreise ρ_i und ρ_o, sowie durch die parallel zur Längsachse verlaufenden Begrenzungslinien des Fahrzeugs. Im weiteren Verlauf sind vorrangig diese Bereiche ausschlaggebend, da sie diejenigen Flächen repräsentieren, welche bei den verwendeten atomaren Teilbewegungen von Fahrzeug beansprucht werden. Demnach muss, beispielsweise für das Fahren einer 360° Linkskurve die Fläche zwischen ρ_i und ρ_o um Ω_p völlig frei von Hindernissen sein. Ist dies nicht der Fall, ist die *Freiheit* der Konfiguration q eingeschränkt, d.h. eine Fahrt in diesem Bereich ist nur eingeschränkt oder gar nicht möglich. Dieser Begriff ist zugleich das Stichwort für die Überleitung zum nächsten Abschnitt.

4.7.3. Freiheitsgrade

Konfigurationen können hinsichtlich eines oder mehrerer Hindernisse eine Qualität besitzen, welche am besten mit **Freiheitsgrad** zu bezeichnen ist. Der jeweilige Grad der Freiheit drückt dabei aus, ob und wenn ja, inwiefern, die Menge der Bewegungsalternativen einer Konfiguration q eingeschränkt sind. Entsprechend der Philosophie atomarer Bewegungen, besitzt eine Konfiguration maximal 6 verschiedene Freiheitsgrade, resultierend aus den Bewegungsparametern:

$$\{-v_0, v_0\} \times \{-\phi_{max}, 0, +\phi_{max}\}.$$

Freiheitsgrade der geraden Vorwärts- bzw. Rückwärtsfahrt könnten durch Angabe eines Längenmaßes bewertet werden, Freiheitsgrade bei Kurvenfahrt durch eine Winkelangabe. Um ein einheitliches Maß zu setzen und damit Vergleiche zu erleichtern, können entweder Längen- in Winkelmaßen oder umgekehrt ausgedrückt werden. Der Charakter des Templatesystems,

nämlich der eines lokalen Modells, und eine gewisse Bevorzugung von Bewegungen mit vollem Lenkeinschlag, sprechen für eine Verwendung des Winkelmaßes für alle Freiheitsgrade.

Beispiel: Eine Konfiguration habe einen Freiheitsgrad von $f_{lf} = \pi/2$ und einen weiteren Freiheitsgrad $f_{sb} = \pi$. Die erste Angabe bedeutet, dass eine Vorwärtsfahrt nach links eine Bewegung um $\triangle\theta = \pi/2$ ermöglicht. Anders, d.h. in Längeneinheiten ausgedrückt, beträgt die mögliche Fahrtstrecke $s = f_{lf}\rho_{min}$. Der zweite Freiheitsgrad beschreibt eine mögliche Rückwärtsfahrt geradeaus, auf einer Strecke von $s = f_{sb}\rho_{min}$.

Nach der Festlegung einer Maßeinheit gilt es, die Eigenschaft der Freiheit einer Konfiguration näher zu bestimmen. Der Schwellenwert, ab dem eine Konfiguration als *frei* zu bewerten ist, ist in Abhängigkeit vom spezifischen Umfeld anzupassen. Dazu mehr im nächsten Abschnitt.

4.8. Modellierung des Templatesystems mit Kollisionsbehandlung

Das Templatesystem mit Kollisionsbehandlung gliedert sich in mehrere Teilschritte.

- **Befreien der Konfiguration(en):** Die möglichen Start- bzw. Zielkonfigurationen können derart durch Hindernisse begrenzt sein, dass lediglich ein sehr schmaler und spezifischer "Kanal" von der Konfiguration weg führt. In manchen Fällen ist ein solcher Kanal auch nur dann erreichbar, wenn zunächst einige Rangiermanöver durchgeführt werden.

  Das Kriterium der Freiheit bezeichnet in diesem Fall den Grad der Erreichbarkeit einer Konfiguration. Für eine effiziente Bewegungsplanung muss ein Mindestfreiheitsgrad vorhanden sein, welcher eventuell in einem separaten Schritt hergestellt werden muss.

- **Generierung der Interimskonfiguration(en):** Besitzen beide Konfigurationen q und q' den minimal benötigten Freiheitsgrad (oder wurden sie in die Konfigurationen q_f bzw. q'_f überführt, die diese Eigenschaft erfüllen), ist eventuell die Generierung von Interimskonfigurationen notwendig. Dies kann durch das Vorhandensein von Hindernissen bedingt sein.

- **Generierung des Pfades:** Stehen q (bzw. q_f) und q' (bzw. q'_f) sowie sämtliche Interimskonfigurationen $q_{1,\ldots,n}$ fest, kann die Generierung der Teilpfade durchgeführt werden. Die Aneinanderreihung eventuell vorhandener Rangiermanöver und der Teilpfade liefert dann den gesamten Pfad.

Zwischen den Teilschritten bestehen prinzipiell keine zwangsläufigen Interdependenzen, die eine lineare Ausführung erzwingen würden. Es ist beispielsweise möglich, während mittels lokaler Rangiermanöver q in q_f überführt wird, bereits die weitere Planung durchzuführen, oder ein eventuell notwendiges Manöver auf Seiten q' vorzubereiten - falls die dafür benötigten Lageinformationen bereits vorliegen.

4.8.1. Befreien von Konfigurationen

Das Befreien einer Konfiguration ist immer dann erforderlich, wenn kein ununterbrochener Pfad, bestehend aus einer atomaren Bewegung, existiert, der eine wesentliche Änderung der

Position einer Konfiguration ermöglichen würde. Ein klassisches Beispiel für eine Situation, in der dieser Fall auftritt ist ein in einer parallelen Parklücke stehendes Fahrzeug (vgl. Abb[18]. 4.14).

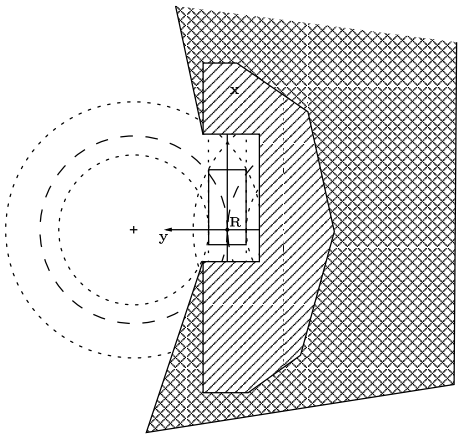

Abbildung 4.14.: Konfiguration ohne signifikante Freiheitsgrade

Trotz der relativ großen Ausmaße der Parklücke liegen in jedem primären Bewegungsbereich Hindernisse, oder zumindest Teile davon. Primäre Bewegungsbereiche sind die vom Fahrzeug benötigten Bereiche zwischen ρ_i und ρ_o, bzw. der lineare Bereich von einer Fahrzeugbreite vor und hinter dem Fahrzeug (gemäß Abb. 4.13). Die durch die Sensorik wahrgenommen Werte sind der Form nach den in Abbildung 4.12 simulierten Werten ähnlich, abgesehen von den dort angegebenen absoluten Entfernungsangaben. Auch hier ist lediglich die linke Fahrzeugseite nicht unmittelbar durch Hindernisse eingegrenzt.

Um den Freiheitsgrad der Konfiguration zu verbessern, muss im Rahmen der lokalen Möglichkeiten das Fahrzeug rangiert werden. Ziel hierbei ist eine Drehung[19] des Fahrzeugs, so dass die Abstände zu etwaigen Hindernissen nach vorne bzw. hinten maximiert wird.
In Abbildung 4.15 (links) ist ersichtlich, wie mit einer kurzen Bewegung rückwärts (Lenkeinschlag nach rechts) bereits eine deutliche Verbesserung für den Freiheitsgrad f_{lf} (vorwärts, links) erzielt werden kann. Oberhalb der auf R bezogenen y-Achse tangiert den Bereich zwischen ρ_i und ρ_o nach diesem Manöver kein Hindernis mehr. Die in Abbildung 4.15 (rechts) dargestellte, anschließende Bewegung nach links vorne hat denselben Effekt hinsichtlich des Freiheitsgrades f_{sf} (vorwärts, geradeaus). Man beachte an dieser Stelle auch die unterschiedliche Wahrnehmung der Sensorik in Abhängigkeit der Position des Fahrzeugs in Relation zum Hindernis (doppelt schraffierter Bereich).

Die atomaren Bewegungen der verschiedenen Rangiermanöver werden zum Zweck der späteren Wiederverwendung aufgezeichnet. Sie müssen im Falle des Befreiens der Zielkonfiguration $q' \rightarrow q'_f$ in umgekehrter Reihenfolge reproduziert werden, um die eigentliche Zielkonfiguration q' zu erreichen. Streng genommen ist eine befreite Konfiguration q_f bzw. q'_f ebenfalls

[18]Die einfache Schraffur stellt den Bereich dar, der durch ein oder mehrere reale Hindernisse eingenommen wird. Der von der Sensorik nicht wahrgenommene Bereich (da dieser durch das/die Hindernisse verdeckt ist) ist mit doppelter Schraffur gekennzeichnet.

[19]Nachdem ein Drehen auf der Stelle nicht zugelassen ist, werden hier abermals die substituierenden atomaren Bewegungen unterstellt.

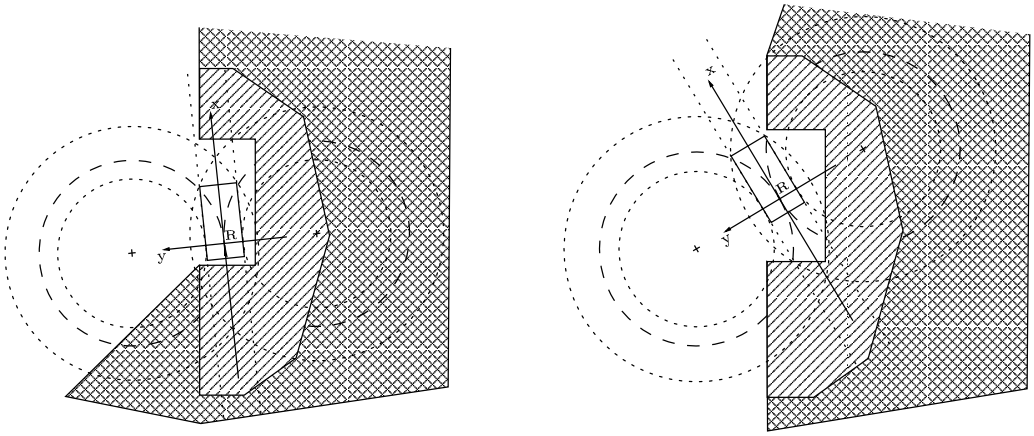

Abbildung 4.15.: Sukzessive Verbesserung der Freiheitsgrade

eine Interimskonfiguration, wie sie im nächsten Abschnitt betrachtet werden. Die unterschiedlichen Anforderungen im Zusammenhang mit terminalen (Start- und Ziel-) Konfigurationen legt jedoch eine separate Behandlung nahe.

Die detaillierte Generierung von befreiten Konfigurationen hängt in hohem Maße von Bauart, Erfassungswinkel, Auflösung und weiteren technischen Merkmalen der verwendeten Sensorik ab.

4.8.2. Generierung von Interimskonfigurationen

Interimskonfigurationen werden immer dann benötigt, wenn Hindernisse den direkten[20] Weg zwischen q und q' versperren. Ferner werden sie benötigt, wenn mindestens eine der Konfigurationen q bzw. q' eingeschränkte Freiheitsgrade aufweist. Im obigen Fall (vgl. Abb. 4.15, rechts) führt q'_f zu einer weiteren Interimskonfiguration, auf Grund der eingeschränkten Freiheitsgrade.
Abbildung[21] 4.16 illustriert verschiedene Interimskonfigurationen für das Beispiel des parallelen Einparkens. In Abbildung 4.16 a) sind die Konfigurationen q und q'_f dargestellt. Man beachte außerdem den für die erste Interimskonfiguration entscheidenden Schnittpunkt c. Dieser Punkt kommt zustande, indem die zu θ parallelen Geraden durch Ω und $\dot{\Omega}$ auf ihren Schnittpunkt mit den durch die entsprechenden Freiheitsgrade qualifizierten Geraden durch Ω'_f (bzw. $\dot{\Omega}'_f$) bzw. $2\rho_{min}$-Kreisen Ω'_f (bzw. $\dot{\Omega}'_f$) geprüft werden. Die jeweils ausschlaggebenden Linien sind gestrichelt dargestellt. Abbildung 4.16 b) zeigt die aus einer ersten Translation von q bis $\Omega = c$ resultierende Interimskonfiguration. Entsprechend der Bewegungsgeometrie liegt nun bei c_2 der Punkt Ω' auf dem $2\rho_{min}$-Kreis um Ω. Anschließend (4.16 c) erfolgt eine Rotation bis $\dot{\Omega} = \Omega'_f$, die in der zweiten Interimskonfiguration resultiert. Die Konfiguration q'_f

[20] Die zu Grunde gelegten Leistungen der Sensorik liefern mit ihren angularen und radialen Komponenten nicht genügend Informationen, um Hindernisse zu erfassen, die einen *indirekten* Einfluss auf die Bewegungsplanung haben. Auf das zu Grunde liegende inkrementelle bzw. iterative Vorgehen hat dies keinen signifikanten Einfluss.

[21] Die Hinderniskonstellation entspricht prinzipiell der aus den Abbildungen 4.14 und 4.15. Im Sinne einer übersichtlichen Darstellung wurden die Hindernisse jedoch so weit wie möglich reduziert, ohne deren Einfluss auf die Planung zu verändern.

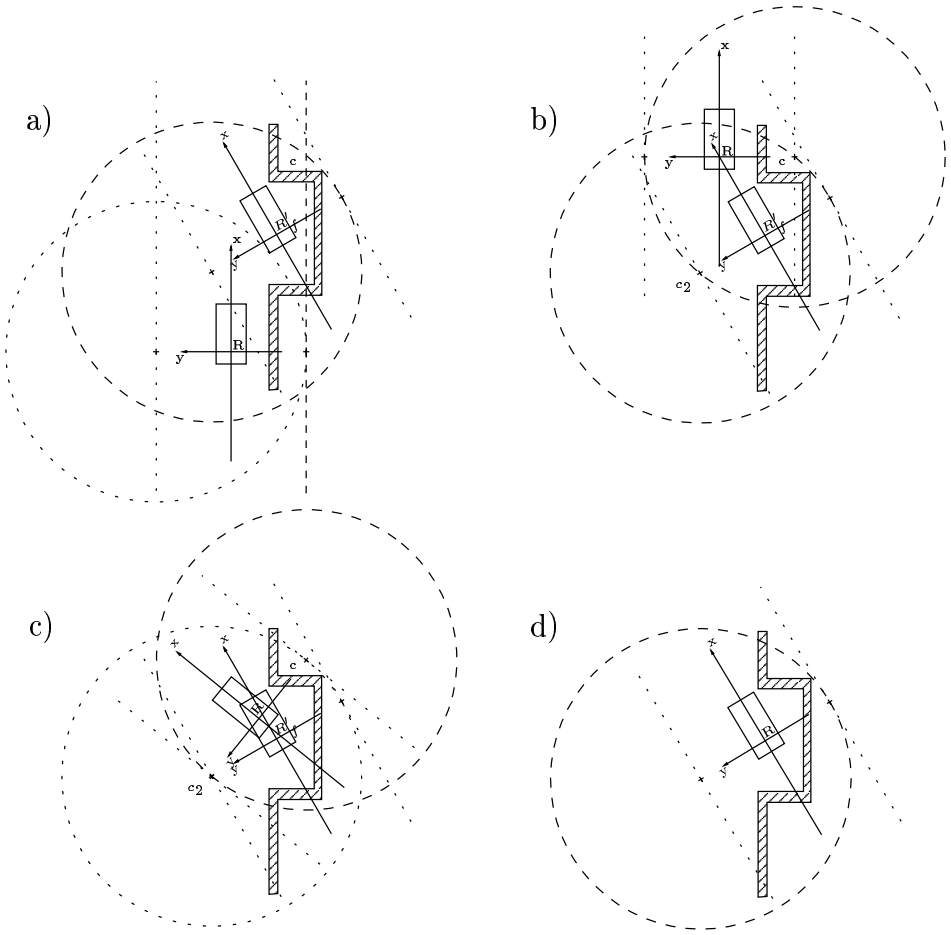

Abbildung 4.16.: Illustration der Interimskonfigurationen

ist schließlich in Abbildung 4.16 d) erreicht, anschließend erfolgen die bereits aufgezeichneten Rangiermanöver von q'_f nach q' (ohne Abb.).

4.8.3. Generierung des Pfades

Der Begriff "Generierung" bedeutet im vorliegenden Fall das Zusammensetzen der Elemente, die aus den bisherigen Teilschritten hervorgehen. Diese Elemente sind sämtliche (Start-, Ziel- und Interims-) Konfigurationen und die Menge der zwischen diesen entwickelten Rangiermanöver bzw. atomaren Bewegungen. Im vorliegenden Fall sind das die Konfigurationen q, q'_f und q', sowie die Interimskonfigurationen zwischen q und q'_f. Hinzu kommen die atomaren Bewegungen zwischen den Konfigurationen und die Rangiermanöver zwischen q' und q'_f.

Prinzipiell sollten die Pfade zwischen einer Konfiguration und einer Interimskonfiguration lediglich aus atomaren Bewegungen bestehen, was die Generierung vereinfacht. Im Falle komplexerer Verbindungen kann jedoch jederzeit der in Abschnitt 4.5 beschriebene Algorithmus verwendet werden.

In dieser Phase fällt vergleichsweise geringer Berechnungsaufwand an, im Gegensatz zu den

Phasen, in denen die Generierung der verschiedenen Konfigurationen erfolgt.

4.8.4. Behandlung statischer Hindernisse

Werden während der Ausführung von Bewegungen Hindernisse erfasst, die bisher nicht in die Planung einbezogen wurden, kann und muss die momentane Bewegung eingestellt werden, sofern Kollisionsgefahr besteht. Solche Hindernisse treten auf, weil sie entweder bisher verdeckt waren und erst durch eine Positionsänderung des Roboters in das sensorisch erfasste Feld gelangten, oder weil sie sich durch eine unmittelbar zuvor stattfindende Eigenbewegung in das Umfeld bewegten und dort in einen statischen Zustand übergingen.

Besteht eine Überdeckung zwischen dem geplanten Pfad und einem Hindernis, wird der berechnete Pfad an dieser Stelle unterbrochen und es werden weitere Interimskonfigurationen gebildet. Auf diese Weise wird der Pfad um einige Elemente erweitert und anschließend wird die Bewegung auf dem neuen Pfad fortgesetzt.

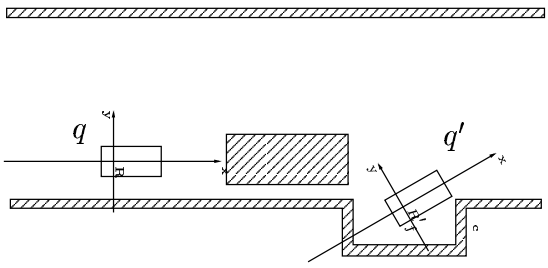

Abbildung 4.17.: Statisches Hindernis

Man betrachte abermals das bereits aus den vorherigen Abschnitten bekannte Beispiel des parallelen Einparkens. In dieser Konstellation (vgl. Abb. 4.17) liegt die Zielkonfiguration im Schatten eines Hindernisses. Die "Peilung" von q in Richtung R' liegt bei ca. $\frac{\pi}{16}$.

Man betrachte nun ein simuliertes Sensorecho für diese Konstellation in Abbildung 4.18. Dem Verlauf des Graphen kann entnommen werden, dass die Bereiche hinter bzw. links vor dem Fahrzeug relativ frei von Hindernissen sind. Eine eventuell besser gestellte Interimskonfiguration liegt vorzugsweise im Bereich von $\frac{\pi}{16} \pm \frac{\pi}{4}$, um eine Bewegung in Richtung von q' zu favorisieren. Hierbei ist im Falle komplexerer Konstellationen möglicherweise ein zufallsgesteuertes und/oder heuristisches Vorgehen anzuwenden. Gemäß der Sensorinformation (4.18) wird die Bewegungsrichtung in Abhängigkeit einer Optimierung des Freiheitsgrades festgelegt, welcher in diesem Fall nahe bei $7\pi/4$ liegen dürfte. Auf diese Weise kann die Sensorik die "On-the-Fly"-Generierung einer weiteren Interimskonfiguration unterstützen, wie in Abbildung 4.19 dargestellt ist. Sobald keine weitere Beeinträchtigung durch Hindernisse gegeben ist, wird die ursprüngliche Planung wieder aufgenommen, indem die neu geschaffene Interimskonfiguration mittels (atomarer) Bewegung(en) mit der nächstgelegenen Konfiguration der bisherigen Planung verbunden wird. Stehen mehrere Alternativen zur Verfügung, kann mit Hilfe einer geeigneten Distanzfunktion eine Auswahl getroffen werden.

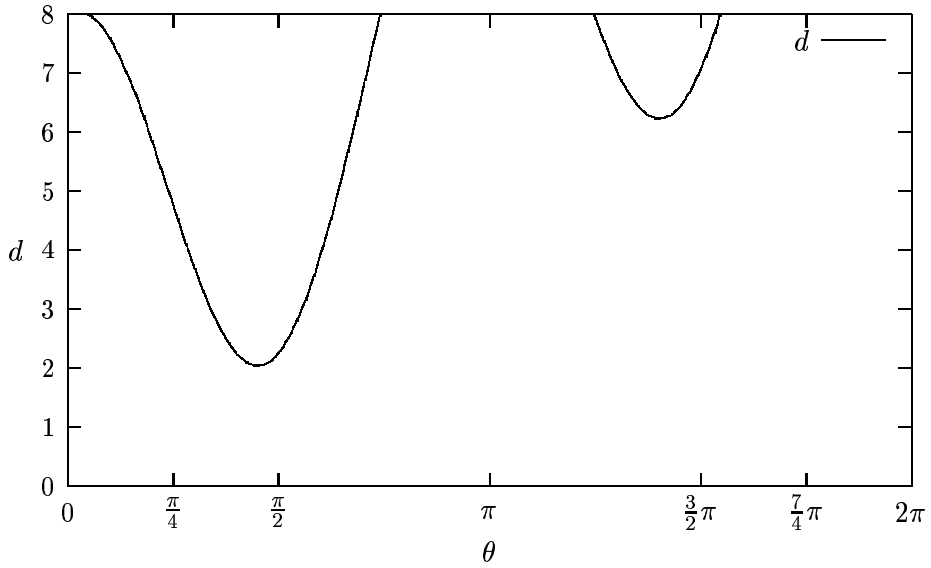

Abbildung 4.18.: Simuliertes Sensorecho

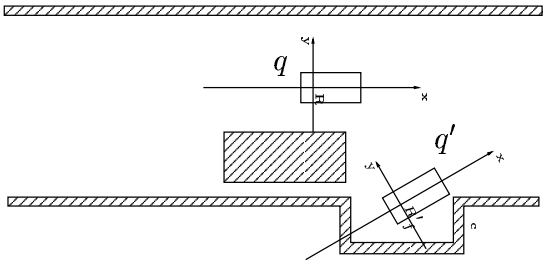

Abbildung 4.19.: Mögliche Interimskonfiguration zur Umgehung

4.8.5. Behandlung dynamischer Hindernisse

Unter dem Begriff der dynamischen Hindernisse seien alle Hindernisse zusammengefasst, welche nicht statisch sind. In diese Kategorie fallen zunächst alle sich bewegenden Hindernisse, aber auch solche, die ein hybrides Verhalten zeigen. Bei einem Übergang eines Hindernisses vom statischen in den dynamischen Zustand ist die Behandlung eindeutig. Der Bewegungsplaner ändert das Attribut von statisch auf dynamisch und fährt mit der Behandlung fort, als wäre das Hindernis von Anfang an als dynamisch eingestuft worden. Der umgekehrte Fall, in dem ein Hindernis sich zu einer Postition bewegt und dann dort verweilt, verlangt nach einem zeitlichen Schwellenwert. Zunächst ist anzunehmen, dass das Hindernis wieder eine Bewegung durchführt. Ist dies nicht der Fall, dann wird es - nach Ablauf einer bestimmten Zeitspanne - wie ein statisches Hindernis behandelt.

Die Erkennung dynamischer Hindernisse liegt in dem Vergleich von Relativgeschwindigkeiten. Alle statischen Hindernisse dürfen in Bezug auf die Bewegungen des Roboters keine Relativgeschwindigkeiten besitzen. Sobald eine solche vorliegt und durch die Sensorik wahrgenommen wird (z.B. Änderungen in der Entfernung eines lateral positionierten Objektes oder

eine von v_0 unterschiedliche Änderung des Abstandes entlang der Längsachse) ist mit einem dynamischen Hindernis zu rechnen. In diesem Fall wird die Bewegung bei Unterschreitung kritischer Entfernungen eingestellt, ansonsten fortgeführt. Besteht Kollisionsgefahr, wird eine vorbestimmte Zeitspanne abgewartet, um festzustellen, ob sich die Konstellation positiv verändert. Dies ist dann der Fall, wenn das dynamische Objekt den kritischen Bereich auf einem geeigneten Transienten verlässt. Ist nach Ablauf der Zeitspanne keine signifikante Änderung eingetreten, wird der Bewegungsprozess unter Einbeziehung der veränderten Prämissen wieder begonnen.

Ist ein hochdynamisches Umfeld zu erwarten, in dem mit hoher Wahrscheinlichkeit mehrere dynamische Komponenten (Hindernisse, Roboter, etc.) um kollisionsfreie Pfade konkurrieren, müssen zusätzliche Mechanismen in die Bewegungsplanung aufgenommen werden. Dies stellt eine erhebliche, komplexe Erweiterung des Problems dar und verlangt nach entscheidenden Anpassungen des zu Grunde liegenden Systems bzw. Modells. Beispielsweise verlangt diese Adaption nach einer Erweiterung des Konfigurationsraumes auf einen **Konfigurations-Zeit-Raum**[22].

Man unterstelle einen dreidimensionalen Workspace mit beliebig vielen rotierenden Hindernissen. In [ReSh 85] wurde gezeigt, dass das daraus resultierende dynamische Bewegungsplanungsproblem P-hart ist, falls der Modulo der Geschwidigkeiten beschränkt ist, sonst NP-hart[23]. Ähnliche Untersuchungen wurden in Bezug auf die dynamische Bewegung eines Punktes in der Ebene durchgeführt [Cann 88]. Weitere Informationen zu diesem Themenbereich finden sich in [Lato 91], Kapitel 8 ff.

4.9. Erfüllung der Aufgabenstellung

Im Folgenden wird der Zielerreichungsgrad bezüglich der primären und sekundären Ziele überprüft.

Primäre Ziele

- Im Falle des Templatesystems handelt es sich um ein **Planungsmodell bei unvollkommener Information**, nicht zuletzt durch die mögliche Handhabung approximativer Referenzen und der fortlaufenden Einbindung sensorischer Information in den inkrementellen, reaktiven Planungsablauf.

  Die Abhängigkeit des im Vorfeld verwendeten Routensystems bleibt von den Forderungen unberührt, da diese lediglich das lokale Modell betreffen.

- Die Möglichkeiten der **Handhabung dynamischer und statischer Hindernisse** hängen zwar in hohem Maße von der verwendeten Sensorik ab, werden jedoch vom zu Grunde liegenden Modell ausdrücklich vorgesehen.

- Der reaktive Charakter des Modells ermöglicht individuelle Reaktionen auf dynamische Ereignisse und damit Änderungen an Strategie bzw. Planung. Phasenweise werden Sensorinformationen in Echtzeit überwacht und verarbeitet.

[22] In der Literatur werden in diesem Zusammenhang die Begriffe "Konfigurations-Raum-Zeit" und "Konfigurations-Zeit-Raum" gleichbedeutend verwendet.
[23] Vgl. [Lato 91], S. 360.

Sekundäre Ziele

- Bedingt durch die fortlaufende Sensorüberwachung und durch die Zerlegung verschiedener Pfade in atomare Teilbewegungen werden gute Voraussetzungen für einen **subsidiären Modus**, zusätzlich zu einer vollautomatischen Betriebsart, geschaffen.
- Durch die **fortlaufende Fehlererkennung** können Abweichungen von diversen Sollwerten jederzeit erkannt und entsprechende Maßnahmen eingeleitet werden. Zudem ist eine aktive Kollisionsvermeidung durch automatisierten Eingriff möglich. (Aufgrund Echtzeit kein Problem)

5. Technische Basis

Im folgenden wird die Eignung des Verfahrens anhand der Erfüllung technischer Voraussetzungen diskutiert. Von einer kurzfristigen Einführung kann ad hoc nicht ausgegangen werden, da auf breiter Basis entsprechende drive-by-Wire Komponenten weder vorhanden noch verfügbar sind, welche eine computergesteuerte bzw. -unterstützte Fahrzeugführung erlaubten. Zudem sind solche Systeme vom Gesetzgeber meist - zumindest vorerst noch - verboten[1]. Nichtsdestoweniger sollte in Anbetracht der installierten Basis so weit wie möglich auf vorhandene und eingeführte oder zumindest verfügbare Hardware zurückgegriffen werden.

5.1. Grundlagen der Sensorik

Sensoren, welche die oben angeführten technischen Eigenschaften besitzen, sind in heutigen Fahrzeugen am ehesten in Form von Ultraschallsensoren für Parkassistenten zu finden. Seit einigen Jahren erfolgreich in der Serie eingesetzt, bieten sie eine akzeptable Basis für eine Untersuchung der Praxistauglichkeit des Verfahrens. Im nächsten Abschnitt werden am Beispiel der Sensoren der Firma Valeo, unter anderem Ausrüster für die Park Distance Control (PDC) der BMW 5er und 7er Baureihe, die technischen Eigenschaften genauer betrachtet.

Zunächst jedoch wird auf zwei wichtige Aspekte näher eingegangen, um die generelle Komplexität des Themas zu illustrieren. Es handelt sich zum einen um konstruktive, zum anderen um rechtliche Einflussfaktoren.

5.1.1. Konstruktive Aspekte

Betrachtet man die für verschiedene Forschungsarbeiten verwendeten Roboter[2] und deren Ausstattung an Sensoren, zeigen sich deutliche Unterschiede zur vergleichsweise einfachen Sensorik eines Parkassistenten in Serien-PKW. Eine Bestückung autonomer Roboter mit Sensoren im Umfang der folgenden Auflistung ist nicht außergewöhlich:

- **16 Berührungssensoren** zur Erkennung von Kollisionen mit sehr kleinen Objekten
- **16 oder mehr Sonarsensoren und/oder Infrarotsensoren** für Abstandsmessungen mit unterschiedlichen Abtastwinkeln im Bereich mehrerer Zentimeter bis einige Meter
- **ein 2D Laser Abstandsmesssystem** zur Abtastung eines (Teil-)Bereiches (bis 360°) mit sehr hoher Auflösung ($< 1°$) und Genauigkeit
- **eine oder mehrere steuerbare S/W Cameras** zur Erfassung visueller Information

[1] Vgl. u.a. [Beck 01], StVZO §41 Abs. 1 ff.
[2] Ein oft verwendetes Modell ist der Nomad 200, vgl. Abb. 5.1.

Abbildung 5.1.: Nomad 200c, Nomadic Inc.

Jedes der angeführten Systeme hat spezifische Vor- und Nachteile mit weitreichenden Konsequenzen für die jeweilige Verwendung. Die Aufstellung zeigt außerdem, wie vielfältig die technischen Möglichkeiten für sensorbasierte Erfassung sind. Zu den verschiedenen technischen Anforderungen (d.h. mechanische, physikalische und ähnliche Aspekte) kommen weitere Einflussfaktoren hinzu, so z.B. auch rechtliche Bestimmungen. Der nächste Abschnitt zeigt einen kleinen Ausschnitt aus den rechtlichen Einflussfaktoren anhand eines konkreten Beispiels.

5.1.2. Rechtliche Aspekte

Unter anderem stellt sich die Frage nach der mindestens benötigten, maximalen Reichweite der Sensoren. Für eine Festlegung dieses Wertes ist, neben anderen Faktoren, der bei der Durchführung von verschiedenen Manövern angestrebte Geschwindigkeitsbereich. Während sich für die benötigte Reichweite intuitiv keine obere Schranke ergibt (Informationen über zu weit entfernte Objekte können verworfen bzw. vernachlässigt werden), so liefert der Gesetzgeber mit §3.1(4) bzw. §3.1(5) der StVO (vgl. u.a. [Beck 01]) die Grundlage für eine untere Schranke:

> [...]
>
> (4) Er [Der Fahrzeugführer] darf nur so schnell fahren, daß er innerhalb der übersehbaren Strecke halten kann.
>
> (5) Auf Fahrbahnen, die so schmal sind, daß dort entgegenkommende Fahrzeuge gefährdet werden könnten, muß er jedoch so langsam fahren, daß er mindestens innerhalb der Hälfte der übersehbaren Strecke halten kann.

Für das zu steuernde Fahrzeug ist Satz (5) anzuwenden, da anzunehmen ist, dass auf den angestrebten Transienten für zwei automatisierte Fahrzeuge Kollisionsgefahr besteht. Hinzu kommt der Einfluss von Reaktionsgeschwindigkeit und Bremsleistung des Roboters. Das ist zum einen die Zeit, die das System benötigt, um zu entscheiden, dass ein Zusammenstoß droht, zuzüglich der Zeit für die Auslösung von Gegenmaßnahmen (Notbremsung). Zum anderen ist das die Zeit, die für den eigentlichen Bremsvorgang benötigt wird. Demnach lässt sich die benötigte Reichweite der Sensorik in Abhängigkeit von Geschwindigkeit v_0, Reaktionszeit t_r und Bremsleistung a folgendermaßen definieren:

$$s = 2\left(\frac{v_0^2}{a} + v_0 t_r\right)$$

Für ein Fahrzeug mit einer maximalen Bremsleistung[3] von $10m/s^2$ und einer Reaktionszeit[4] der Steuerung von $30ms$ ergibt sich bei einer gefahrenen Geschwindigkeit von $10km/h$ eine minimale Sensorenreichweite von ca. $0,94m$.

Alle diese Annahmen unterstellen eine *kontinuierliche* Überwachung der Sensorik. Eine diskrete Abtastung, beispielsweise regelmäßge Messungen in Intervallen von $\triangle t$, beeinflusst obige Berechnungen insofern, als dass im schlechtesten Fall gelten müsste $t_{r_{ges}} = t_r + \triangle t$ und demzufolge:

$$s = 2\left(\frac{v_0^2}{a} + v_0 * (t_r + \triangle t)\right)$$

Im oben angegebenen Beispiel würde sich somit der Wert von $0,94m$ bei einer diskreten Abtastung in Intervallen von $50ms$ (20 Werte pro Sekunde) auf fast $1,22m$ erhöhen.

Hinsichtlich der Umsetzung in die Praxis kommen weitere Einflussfaktoren hinzu. Beispielsweise ist es nicht mit einer sicheren und komfortablen Personenbeförderung vereinbar, dass grundsätzlich alle Sicherheitsreserven ausgenützt werden. Ebenso ist, auch im Hinblick z.B. auf Verschleiß, nicht immer eine Planung auf Basis maximaler Verzögerungswerte sinnvoll. Radargesteuerte Tempomaten zur Geschwindigkeits- und Abstandsregelung arbeiten für gewöhnlich mit weit niedrigeren Verzögerungswerten[5]. Wenn man obiges Beispiel mit einem gewünschten Rest-Sicherheitsabstand s_s erweitern möchte, erhält man:

$$s = 2\left(\frac{v_0^2}{a} + v_0 * (t_r + \triangle t)\right) + s_s,$$

und unter der Annahme $s_s = 0,5m$ und $a = 3m/s^2$ erhält man nunmehr eine minimale Sensorenreichweite von über $3,5m$.

5.1.3. Weitere Aspekte

Die beiden vorhergehenden Abschnitte zeigen anhand von nur zwei Beispielen die Komplexität der Thematik in aller Deutlichkeit. Die Liste weiterer Faktoren ist allerdings damit bei weitem noch nicht ausgeschöpft. Weitaus schwieriger zu behandeln und dennoch von großem Einfluss

[3] §41.4 StVZO fordert allgemein für Kraftfahrzeuge eine Mindestbremsleistung von $5m/s^2$; die Bremsleistung aktuell produzierter PKW liegt im Bereich von $8 - 10m/s^2$, in Einzelfällen auch knapp darüber.

[4] Der angenommene Wert liegt bei ca. 1/10 bis 1/20 der Reaktionszeit eines trainierten menschlichen Fahrers. Der Gesetzgeber lässt hier eine Reaktionszeit von bis zu 1 Sekunde zu.

[5] Das Active Cruise Control (ACC) im BMW 7er (E38) arbeitet beispielsweise mit einer maximalen Verzögerung von $2m/s^2$.

sind beispielsweise äußere Bedingungen, wie z.B. Witterungsverhältnisse, Temperatur oder die Beschaffenheit der Fahrbahnoberfläche. Diese Werte beeinflussen mehrere ausführungskritische Parameter und sind zudem nur schwer mittels Sensoren in hinreichender Genauigkeit erfassbar.

5.2. Stand der Technik

Wie schon erwähnt, sind bereits seit geraumer Zeit Sensoren in Serienfahrzeugen zu finden, welche einen zur Umfelderkennung artverwandten Einsatzzweck erfüllen. Parkassistenten wie die PDC arbeiten auf Basis von Ultraschallsensoren. Der Einsatz in der Praxis gestaltet sich wie im Folgenden beschrieben.

5.2.1. Parkassistenten

Ein Fahrzeug ist üblicherweise ausgerüstet mit bis zu acht Ultraschallsensoren (vgl. Abb. 5.2), je vier an der vorderen und hinteren Stoßstange, die den Raum vor bzw. hinter dem Fahrzeug und an den Ecken überwachen[6].

Abbildung 5.2.: Sensor für Parkassistent, Valeo

Ein solcher Sensor[7] arbeitet auf einer Frequenz von 38,4 kHz, besitzt eine Reichweite von maximal 220 cm und einen Erfassungswinkel von horizontal 100-140 Grad, vertikal 40-70 Grad (vgl. Abb. 5.3). Das erfasste Signal wird in 12 zeitabhängige Schwellenwerte (Eingangssignal mit Funktion Schmitt-Trigger) eingeteilt, welche im Abstand von 0,5 msec., später 1 msec. liegen.
Obwohl theoretisch eine Auflösung von ca. 2 cm erreicht werden kann, ist die spezielle Verwendung für eine sensorbasierte Bewegungsplanung nur eingeschränkt möglich. Hierfür sind mehrere Gründe und Effekte ausschlaggebend:

- Die **Reichweite** beträgt unter optimalen Umständen maximal 220 cm, was selbst für eine Bewegungsplanung in der unmittelbaren Umgebung nur bedingt ausreicht. Ein zuverlässiger Einsatz scheint eher im Bereich der lokalen Hindernis- bzw. Kollisionserkennung möglich.

[6] Vgl. hierzu die grün gekennzeichneten Felder in Abb. 5.6; neuerlich sind zudem Systeme mit sechs Sensoren vorne erhältlich (vgl. Daimler-Chrysler W203 Sportcoupé).
[7] Angaben lt. [Valeo], weitere Informationen von Hr. Gotzig, Valeo.

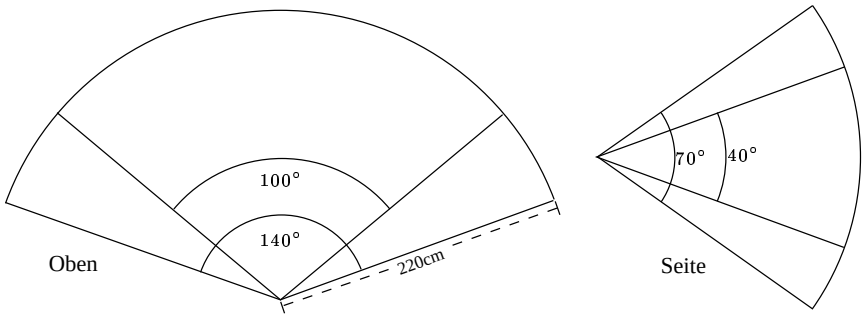

Abbildung 5.3.: Erfassungswinkel und Reichweite des Sensors

- Zum Umstand der radialen Messung[8] kommt hinzu, dass die vom Sensor gelieferten Daten ohne weitere Aufbereitung **keine absoluten Entfernungsmessungen** zulassen. Dies kommt zustande, da das gemessene Signal in hohem Maße von der Beschaffenheit des Objektes abhängig ist, genauer gesagt von dessen Rückstreuquerschnitt. Dies sorgt dafür, dass Objekte mit einem großen Rückstreuquerschnitt (z.B. eine Wand) näher erscheinen, als solche mit einem kleinen (z.B. Rohr, Zylinder). Eine Eichung des Gerätes ist zwar möglich, jedoch im Sinne des angedachten Einsatzumfeldes nur schwer durchführbar.

- Des Weiteren erfolgt **keine Temperaturkompensation**. Da die Schallgeschwindigkeit nicht unerheblich von der Lufttemperatur abhängt, müsste eine solche Kompensation jedoch erfolgen. Anderenfalls ergeben sich rein physikalische Messunterschiede von annähernd 10% in einem durchaus realistischen Temperaturbereich von -20 bis $+30°C$.

Trotz der gefundenen Defizite zeigt diese Technologie die Richtung künftiger Entwicklung auf. Der im nächsten Abschnitt beschriebene Test zeigt anhand der Praxis, wie groß die Unterschiede zwischen Soll- und Istzustand sind.

5.2.2. Praxiserfahrungen PDC

In einem Praxisversuch wurde die PDC[9] des BMW 7er/E65 einem Test unter realen Bedingungen unterzogen. Hauptaugenmerk lag hierbei auf den von der Sensorik gelieferten Informationen und deren Darstellung. Insbesondere wurde untersucht, ob die Sensorik eine ausreichende Auflösung, Reichweite und Zuverlässigkeit bietet, um das Fahrzeug lediglich auf Basis der gelieferten Informationen sicher zu bewegen bzw. einzuparken.

Betreffend die Sensorik unterscheidet sich die PDC im E65 nicht grundlegend von der anderer Modelle oder Hersteller. Was jedoch die Darstellung anbetrifft, liegen einige Unterschiede vor. Im Gegensatz zu anderen am Markt verfügbaren Systemen ist hier zusätzlich zum akustischen Signal eine grafische Darstellung vorhanden, die eine Fahrzeug-Draufsicht und etwaige Hindernisse wiedergibt. Diese ist sichtbar in der so genannten "Comfort Area", d.h. im Monitor auf dem Armaturenbrett in der Fahrzeugmitte. Die Anzeige ist in Abb.[10] 5.4 dargestellt.

[8] Durch Triangulation ist hier eine Verbesserung möglich.
[9] PDC: Park Distance Control, der Parkassistent von BMW.
[10] Abbildungen und technische Informationen in diesem Abschnitt sind [BMWa] und [BMWb] entnommen.

Abbildung 5.4.: Grafische Ausgabe der PDC

Nähert man sich mit dem Fahrzeug einem durch die Sensoren erfassten Hindernis, so wird dies durch einen akustischen Warnton und Einfärben der Flächen um das Fahrzeug auf dem Display mitgeteilt. In Abhängigkeit der Entfernung werden die Intervalle des akustischen Signals (Pieptöne in verschiedenen Abständen) kürzer, bis zu einem durchgehenden Warnton ab Unterschreitung eines bestimmten Schwellenwertes. Die grafische Anzeige variiert grob von Grün über Gelb nach Rot, mit jeweils einigen Zwischenschritten. Bedingt durch die Beschaffenheit der Sensoren (vgl. Abschnitt 5.1) erscheint das Hindernis allerdings relativ unscharf bzw. geglättet. Genaue Konturen sind nicht erkennbar. Die Erfassung beginnt bei einer Entfernung von ca. 1,5 bis 1,8m und endet bei ca 15cm, wobei ein durchgehender Warnton ausgelöst, bzw. die niedrigstmögliche Entfernung grafisch angezeigt wird.

Der Erfassungswinkel der Sensorik ist vorne und hinten unterschiedlich, was konstruktiv beabsichtigt ist und durch die Einbaulage der einzelnen Sensoren beeinflusst werden kann. Die genaue Positionierung an der vorderen und hinteren Stoßstange ist Abb. 5.5 zu entnehmen (Vgl. rote Markierungen).

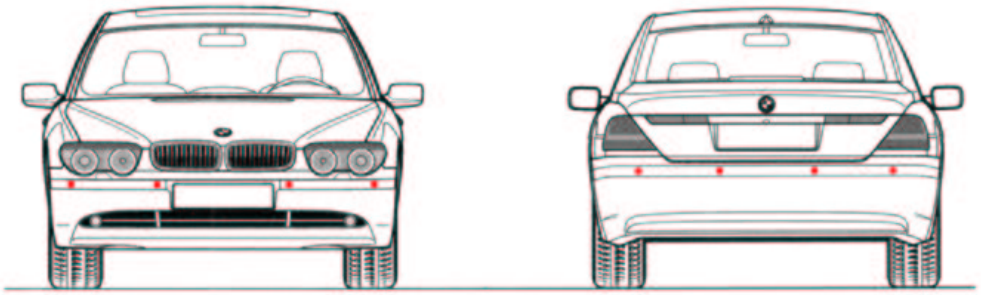

Abbildung 5.5.: Lage der Sensoren, BMW 7er (E65)

Ergebnisse

Die Durchführung verschiedener Parkmanöver in realistischer Umgebung ergab folgende Erkenntnisse. Grundsätzlich wäre es möglich, das Fahrzeug nur auf Grund der Sensorinformationen, d.h. ohne weitere Hilfsmittel und ohne Sicht nach Draußen, einzuparken. Die gelieferten

Signale haben eine ausreichende Auflösung und Aussagekraft, was statische Hindernisse betrifft. Ein solches Manöver ist allerdings auch nur dann möglich, wenn ex ante Kenntnis über das Nichtvorhandensein eventueller Hindernisse im Bereich der toten Winkel der Sensoren vorliegt.

Eine entsprechende maschinelle Nutzung ist folglich in dieser Form nicht möglich. Zum einen werden, wie bereits weiter oben angeführt, nicht alle Hindernisse gleichermassen erfasst. Probleme bereiten beispielsweise sehr kleine Hindernisse oder solche, welche einen sehr geringen Rückstreuquerschnitt aufweisen. Zum anderen können verschiedene Bereiche nicht durch die Sensorik erfasst werden, wie zum Beispiel seitlich des Fahrzeugs, welche aber sehr wohl von den Fahrzeugbewegungen betroffen sind. Diese Bereiche müssen zusätzlich durch optische Kontrolle des Fahrers überwacht werden. Des Weiteren bleibt Raum für Verbesserung in der Reichweite der Sensoren und der Wunsch nach einer zusätzlich angularen Erfassung, zur Verfeinerung der Winkelauflösung.

Abschließend sei gesagt, dass trotz der unterschiedlichen Zielsetzungen bei der Entwicklung der PDC eine erstaunliche Leistung im Hinblick auf die sensorbasierte Bewegungsplanung erzielt wird. Und obwohl eine unmittelbare Nutzung momentan zwar nicht möglich erscheint, liegt eine Aufrüstung im Bereich des technisch Machbaren und ist in naher Zukunft durchaus als realistisches Ziel anzusehen. Der nächste Abschnitt gibt einen kurzen Einblick in die aktuelle Forschung und Entwicklung auf dem Gebiet der PKW-Sensorik.

5.2.3. Zukünftige Entwicklungen

Die extensive Entwicklung fahrzeugexterner Sensorik begann erst in den letzten Jahren. Abgesehen von den bereits vor längerer Zeit eingeführten Parkassistenzsystemen ist in der aktuellen Modellpalette der meisten Hersteller nicht viel anderes zu finden. Eine Ausnahme sind aktive Tempomaten, welche durch eine, vorne unter der Stoßstange des Fahrzeugs montierte, engstrahlige Radaranlage Abstandsmessungen durchführen, um so den Abstand zum vorausfahrenden Fahrzeug und die Geschwindigkeit zu kontrollieren. Trotz der Verfügbarkeit am Markt gehören auch diese Systeme (noch) nicht zum alltäglichen Straßenbild. Gemäß Abb. 5.6 sind verschiedene Anwendungen für neuartige externe Fahrzeugsensoren denkbar:

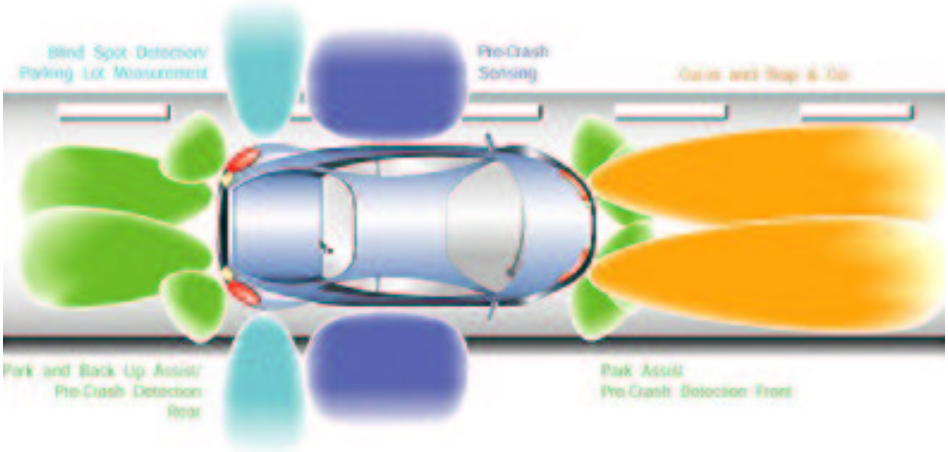

Abbildung 5.6.: Zukünftige Anwendungen externer Sensorik

- **Pre-Crash Detection:** Für eine noch bessere passive Sicherheit werden, zusätzlich zu den Beschleunigungssensoren im Fahrzeug, neue Sensoren benötigt, die einen drohenden Aufprall melden, bevor dieser stattfindet. Eine solche Funktion ist in alle Richtungen, d.h. vorne, hinten und seitlich wünschenswert, um gezielte Gegenmaßnahmen (z.B. Ausweichmanöver oder Auslösung des Airbags) frühzeitig zu veranlassen. In diesem Zusammenhang ist sogar schon mit am Fahrzeug außen angebrachten Airbags experimentiert worden.

- **Blind Spot Detection:** Dies dient der Erfassung von Objekten außerhalb des Sichtfeldes des Fahrers (z.B. auch im Fahrzeugheck, **Back up Assistant**), im Hinblick sowohl auf eine klassische, als auch auf eine automatisierte Fahrzeugführung.

- **Cut In and Stop & Go:** Zur bequemeren (oder automatisierten) Fahrzeugführung für Fahrten bei hoher Verkehrsdichte und/oder geringen Geschwindigkeiten. Hierbei sind Abstandsregelungssysteme genauso denkbar, wie automatisierte Park- bzw. Rangiermanöver.

Für solche und weitere Funktionen sind beispielsweise Sensoren auf Basis von Ultraschall oder Mikrowelle bzw. Radar denkbar. In jedem Fall werden die subsidiären Funktionen der Fahrzeuge von Morgen an Bedeutung gewinnen. Auf dem Weg zur Sicherung der Individualmobilität bei stetig steigender Verkehrsdichte sind solche Systeme aller Wahrscheinlichkeit nach unverzichtbar.

5.3. PATH Magnetic Guidance System

Grundvoraussetzung für das in Abschnitt 4.4 vorgestellte Vorgehen ist eine zusätzliche Informationsinfrastruktur. Wie diese Infrastruktur aussehen könnte, wird anhand eines bereits experimentell vorhandenen Systems aufgezeigt. Es handelt sich hierbei um das so genannte "PATH Magnetic Guidance System" der "Partners for Advanced Transit and Highways"[11]. Detailliertere Informationen darüber finden sich unter anderem in [ZSTP 98] und bei [PATH]. PATH Magnetic Guidance wurde entwickelt, um eine computergesteuerte Führung für, und somit die Kontrolle über ein Fahrzeug zu ermöglichen. Insbesondere liefert das System sehr genaue laterale und longitudinale Angaben[12] über die Position des Fahrzeugs, sowie Information über Straßenverläufe, Fahrbahnbeschaffenheit und dergleichen mehr.

Das System (vgl. Abb. 5.7) besteht zum einen aus im Fahrzeug installierten Sensoren und Daten verarbeitenden Komponenten, zum anderen aus externen Markern. Die Sensoren erfassen Lageinformationen durch Abtasten der in die Fahrbahn eingelassenen magnetischen Markern, welche als absolute örtliche Referenz dienen. Solche Marker sind einfache, stabförmige Dauermagneten[13] (vgl. Abb. 5.8 rechts), die im Abstand von 1,2m die Fahrbahnmitte markieren. Über die rein lokative Information hinaus lässt sich durch die gezielte Variation der Polarität ein einfacher binärer Code übertragen: ein oben befindlicher Nordpol entspricht

[11] Abgek. PATH, eine Kollaboration des California Dept. of Transportation (Caltrans), der University of California und verschiedenen anderen akademischen und privaten Institutionen sowie der privaten Industrie, zur Verbesserung der Kapazität und Sicherheit auf den Highways und zur Reduzierung von Staus, Umweltverschmutzung und des Energieverbrauchs.

[12] Genauigkeit: lateral 5mm, longitudinal 5cm; geringe Empfindlichkeit gegenüber den üblichen Umwelteinflüssen wie Temperaturschwankungen, Niederschlag, Vibration, etc.

[13] Techn. Daten: Durchmesser 2.4cm, Länge 10cm, Kosten pro Stück ca. 10 Eurocent.

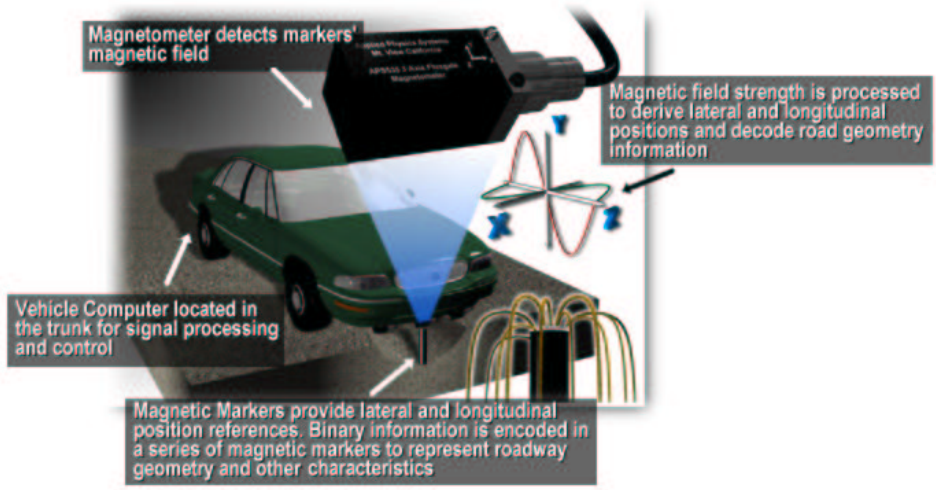

Abbildung 5.7.: PATH Magnetic Guidance System

einer binären "1", ein Südpol dagegen einer "0". Auf diese Weise lassen sich Informationen über den Straßenverlauf, Geschwindigkeitsbegrenzungen, Beschilderung, Ein- und Ausfahrten und vieles mehr, fest in die Fahrbahn einkodieren. Durchschnittlich könnte man so in einem Straßenabschnitt von etwa 40m Länge eine 32-Bit-Nachricht einbetten, welche von einem vorüberfahrenden Fahrzeug bei einer Geschwindigkeit von 100 km/h in etwas mehr als einer Sekunde gelesen würde.

Abbildung 5.8.: Magnetometer und Stabmagnet

Dieses System stellt eine technische Basis, deren unterschiedliche Anwendungen unter anderem die Folgenden beinhalten:

- Automatische Steuerung und Kontrolle (Kriterium für Modell in Abschnitt 4.4)
- Warnsystem für Fahrbahnabweichungen
- Präzisionsleitsystem für Haltebuchten

- Leitsystem für Schneepflüge

Mögliche Erweiterungen dieses oder ähnlicher Systeme[14] sind denkbar in verschiedener Hinsicht.

Um die Menge der übertragbaren Informationen zu erhöhen, könnten die Feldstärke der Magneten sowie deren Abstände variiert werden. So könnten pro Magnet mehrere Bits dagestellt und die Informationsdichte durch Einfügen (oder Weglassen) einzelner Marker entsprechend angepasst werden.

[14] Parallel zu diesem Projekt, oder aufbauend darauf, betreibt sowohl die europäische als auch die japanische Industrie weiterführende Forschungen, vgl. [ZSTP 98] und [Laph 97].

6. Ausblick

Obwohl in Kapitel 4 ein ausführliches Modell zur Bewältigung von Planungsproblemen für nicht-holonome Vehikel vorgestellt wird, bleiben verschiedene Detailfragen ungeklärt, die in engem Bezug mit der sensorischen Erfassung stehen. Diese besonderen Aspekte treten überwiegend im Zusammenhang mit der Behandlung statischer und dynamischer Hindernisse auf. Sie hängen essenziell von der spezifischen Leistungsfähigkeit sowohl der einzelnen Sensoren, als auch vom resultierenden System ab, das vom gezielten Zusammenwirken aller beteiligten Komponenten geprägt ist. Dieses gezielte Zusammenwirken konnte bisher nicht erschöpfend untersucht werden, aufgrund weit reichender Interdependenzen zwischen verschiedenen Modellen, Ansätzen und Vorgehensweisen und der technischen Aspekte der Realisierung.

In Zukunft werden effiziente Möglichkeiten gesucht, die spezifischen Vorteile verschiedener Vorgehensweisen zu vereinen und eine Brücke zwischen der reaktiven, sensorbasierten Bewegungsplanung einerseits und den geometrisch-modellbasierten Ansätzen andererseits zu schaffen. Wie bereits an verschiedenen Stellen dieser Ausarbeitung angedeutet wurde, eröffnen sich im Falle einiger Teilaspekte völlig neue Thematiken, deren ausführliche Diskussion beträchtlichen Umfang einnehmen wird.

Abschließend seien einige neue Herausforderungen, neuartige Anwendungen und kommende Problemstellungen aufgeführt. Diese von einem Gremium unter der Leitung Kamal Guptas [GudP 98] zusammengetragenen Fragestellungen zeigen auf, welchen Stellenwert auch künftig die Bewegungsplanung mit all ihren Teilbereichen einnehmen wird und welch umfangreiche Forschungsarbeiten hierfür notwendig werden.

- Wie kann eine Interaktion zwischen der Bewegunsplanung und den Erfassungs- bzw. Kontrollkomponenten stattfinden?
- Bewegung in einer teilweise unbekannten Umwelt erfordert Möglichkeiten zur Erfassung derselben. Sensorbasierte Echtzeitbewegungsplaner sollten visuelle Systeme integrieren. Wie kann eine solche Integration *inkrementell* erfolgen?
- Einen neuen Bereich stellen Kleinstroboter dar, in deren mikroskopischem Anwendungsbereich zum Beispiel auch die thermodynamischen Gesetze der Physik beachtet werden müssen.
- Die anvisierte, so genannte Dienstrobotik birgt neue Herausforderungen. Da Bewegungsplanung in diesem Zusammenhang bedeutet, die Zukunft zu bestimmen, werden Themenbereiche wie die *Behandlung von Unvollständigkeit*, *Fehlertoleranz* und *Robustheit bei Unsicherheit* von entscheidender Wichtigkeit sein.
- Da Bewegungsplanung nur ein Teilbereich dessen ist, wie der Mensch mit seiner Umwelt interagiert, wird die Verbindung mit anderen Teilbereichen zukünftig immer wichtiger: Planungen für Greifoperationen, Bearbeitung von Objekten und verfeinerte Bewegungsplanung.

A. Auszüge aus dem Sourcecode

A.1. Configuration.h

```
#ifndef GUARD_Configuration_h
#define GUARD_Configuration_h

#include <string>
#include <iostream>
#include <iomanip>

#include "datatypes.h"

class Configuration {
public:
  // Constructors...
  Configuration::Configuration( void );
  Configuration::Configuration( point, double );
  Configuration::Configuration( int, int, double );

  // Operations...
  void Configuration::setX( double );
  void Configuration::setY( double );
  void Configuration::setTheta( double );
  void Configuration::setType( int );
  double Configuration::getX( void );
  double Configuration::getY( void );
  double Configuration::getTheta( void );
  int Configuration::getType( void );
  void Configuration::print( void );

private:
  point R;
  double theta;
  int type;
};

#endif
```

A.2. ConfigSpace.h

```
#ifndef GUARD_ConfigSpace_h
#define GUARD_ConfigSpace_h

#include <vector>
#include <deque>
#include <iomanip>
#include <string>
#include <stdexcept>
#include <list>

#include "datatypes.h"
#include "ConfigSpace.h"
#include "Cell.h"
#include "Vehicle.h"
#include "Configuration.h"
#include "Geometric.h"

// qt ------------------
#include <qwidget.h>
#include <qpainter.h>

#include "lcdrange.h"

class ConfigSpace : public QWidget
{
  Q_OBJECT
public:
  static const int CLSD = 1;
  static const int INIT = 2;
  static const int GOAL = 4;
  static const int PERM = 8;

  // Constructors...
  ConfigSpace( QWidget *parent=0, const char *name=0 );

  // Operations...
  int angle() const { return ang; }   // qt
  QSizePolicy sizePolicy() const;     // qt
  int ConfigSpace::initialize( int, range, range );
  void ConfigSpace::setVehicle( class Vehicle* );
  void ConfigSpace::setQInit( Configuration );
  void ConfigSpace::setQGoal( Configuration );
  int ConfigSpace::getOrder( void );
  void ConfigSpace::closeFrame( class Vehicle* );
  void ConfigSpace::addObstacle( class Geometric* );
```

```
  void ConfigSpace::printObstacles( void );
  void ConfigSpace::generateCObstacles( class Vehicle* );
  void ConfigSpace::printCell( Configuration* );
  int ConfigSpace::discreteX( double );
  int ConfigSpace::discreteY( double );
  int ConfigSpace::discreteTheta( double );
  int ConfigSpace::discrete( double, double );
  bool ConfigSpace::buildGraph( void );
  bool ConfigSpace::markPath( void );
  void ConfigSpace::printPath( void );
  void ConfigSpace::paintPath( void );
  bool ConfigSpace::isGoal( Configuration* );
  bool ConfigSpace::isClosed( Configuration* );
  bool ConfigSpace::isOpen( Configuration* );
  bool ConfigSpace::isValid( Configuration* );
  bool ConfigSpace::isValid( double, double, double );
  double ConfigSpace::normalizeTheta( double );
  void ConfigSpace::closeCell( Configuration* );
  class Cell* ConfigSpace::getCell( Configuration* );

  range ConfigSpace::getXRange( void );
  range ConfigSpace::getYRange( void );
  range ConfigSpace::getThetaRange( void );

  double ConfigSpace::rastFloorX( double );
  double ConfigSpace::rastCeilX( double );
  double ConfigSpace::rastFloorY( double );
  double ConfigSpace::rastCeilY( double );

  double ConfigSpace::calcDistance( int, int, Configuration* );

  void ConfigSpace::paintEnvironment( void );

  void ConfigSpace::delay( int );

public slots:
  void setAngle( int degrees );
  void generatePath( void );

signals:
  void angleChanged( int );

protected:
  void paintEvent( QPaintEvent * );

private:
  bool pathFound;
```

```
  int order;
  double time;
  range xRange;
  range yRange;
  range thetaRange;
  Vehicle* vehicle;
  Configuration qInit;
  Configuration qGoal;
  std::vector<std::vector<std::vector<class Cell> > > configSpace;
  std::vector<Geometric*> obstacles;
  std::vector<Geometric*> cObstacles;
  std::vector<std::vector<double> > transition;
};

#endif // GUARD_ConfigSpace_h
```

A.3. Vehicle.h

```
#ifndef GUARD_vehicle_h
#define GUARD_vehicle_h

#include <cmath>
#include <vector>

#include "datatypes.h"
#include "Geometric.h"
#include "Configuration.h"
#include "Cell.h"

class Vehicle {
 public:
  // Constructors...
  Vehicle( void );
  Vehicle( double, double, double, double, double, double );

  // Operations...
  class Configuration drive( class Configuration*, int, int, double );
  double getLength( void );
  double getLengthFront( void );
  double getLengthRear( void );
  double getWidth( void );
  double getMaxPhi( void );
  double getVelocity( void );
  double getRadius( void );
  double getDeltaB( void );
  double getDeltaS( void );
```

```
  double getAlpha( void );
  double getBeta( void );
  class Point getA( double );
  class Point getB( double );
  class Point getC( double );
  class Point getD( double );
  class Point getCenter( class Configuration* );
  class Point getCenter( class Cell* );

 private:
  double L;       // length
  double Lr;      // overhang rear
  double Lf;      // overhang front
  double W;       // width
  double phi_max; // max steering angle (rad)
  double v;       // velocity (cm/sec)
  double radius;  // minimal radius around midpoint
  double deltaB;  // Diagonal distance R to front corner
  double deltaS;  // Diagonal distance R to rear corner
  double alpha;   // Angle between deltaB and main axis
  double beta;    // Angle between deltaS and main axis
};

#endif
```

A.4. ConfigSpace.cpp

```
[...]

bool ConfigSpace::buildGraph(){
   Cell* cellSucc;
   Cell* cellCurr;
   Configuration current;
   Configuration successor;
   deque<Configuration> openConfigs;
   double distance;

   // Clear OPEN
   openConfigs.clear();

   // Place q_init in OPEN
   openConfigs.push_back( qInit );

   pathFound = false;
```

```
  while( !openConfigs.empty() ) {
    current = openConfigs.front();
    openConfigs.pop_front();

    if( this->isValid(&current) ) {
      if( this->isGoal(&current) )
        pathFound=true;

      try{
        cellCurr=this->getCell(&current);
      } catch (range_error) {
        cellCurr=0;
        continue; }

      if( cellCurr->isClosed() ){
        continue;
      } else {
        cellCurr->close();

        for( int direction=1 ; direction>=-1 ; direction-=2 ) {
          for( int steering=-1 ; steering<=1 ; ++steering ) {
            successor = vehicle->drive(&current,
                                       direction,
                                       steering,
                                       this->time);
            try{
              cellSucc=this->getCell(&successor);
            } catch (range_error) {
              cellSucc=0;
              continue;
            }
            if( this->isValid(&successor) ) {
              distance=this->calcDistance( direction, steering, &current );
              cellCurr->addSuccessor( cellSucc, distance );
              openConfigs.push_back(successor);
            }
          }
        }
      }
    }
  }

  this->markPath();

  return pathFound;
}
```

```
bool ConfigSpace::markPath( void ){
  if( pathFound ){
    Cell* current;
    Cell* another;
    vector<cellDist>* successors;
    Cell* successor;
    double distance, newDistance;
    list<Cell*> pending;
    double currentDistance;

    unsigned int count=0;

    double lastDistance=0;

    pending.clear();

    try{
      current=this->getCell( &qInit );
    } catch (range_error) {
      cout << "markPath: range error - q_init invalid!";
      return false;
    }

    pending.push_back( current );

    while( !pending.empty() ){
      count++;
      current=pending.front();
      if( current!=0 ) {
        currentDistance=current->getDistance();
        if( lastDistance!=currentDistance ) {
          pending.sort( Compare_by_distance() );
          current=pending.front();
        }
      }

      pending.pop_front();

      if( current->isPermanent() ) {
        continue;
      } else {
        current->markPermanent();
        if( count%100==0 )
          current->paint( this, vehicle, 800 );
        currentDistance=current->getDistance();
        lastDistance=currentDistance;
        successors=current->getSuccessors();
```

```
        if( successors!=0 ) {

          for( vector<cellDist>::iterator pos=successors->begin();
               pos!=successors->end()-1;
               ++pos ){
            successor=pos->cell;
            distance=pos->distance;
            if( successor->isPermanent() ) {
              continue;
            } else {
              newDistance = currentDistance+distance;
              if( successor->getDistance()>newDistance ){
                successor->setDistance( newDistance );
                successor->setAncestor( current );
                pending.push_back( successor );
              }
            }
          }
        }
      }
    }
    return true;
}

[...]
```

A.5. Vehicle.cpp

```
[...]

Configuration Vehicle::drive( Configuration* config,
                              int direction,
                              int steering,
                              double time )
{
  Configuration newConfig = Configuration();
  double theta0=config->getTheta();
  double x0=config->getX();
  double y0=config->getY();
  double v=(this->v/time)*direction;
  double phi=this->phi_max*steering;

  if( phi==0 ){
    newConfig.setX(x0+cos(theta0)*(v*time));
    newConfig.setY(y0+sin(theta0)*(v*time));
```

```
    newConfig.setTheta(theta0);
  } else {
    newConfig.setX(x0+(L/tan(phi))*(sin(theta0+v/L*tan(phi))-sin(theta0)));
    newConfig.setY(y0-(L/tan(phi))*(cos(theta0+v/L*tan(phi))-cos(theta0)));
    newConfig.setTheta(fmod(((theta0+(v/L)*tan(phi)*time)+2*PI),(2*PI)));
  }

  return newConfig;
}

[...]
```

Literaturverzeichnis

[Alex 87] ALEXANDER, H.L.: *Experiments in Control of Satellite Manipulators.* Doktorarbeit, Department of Electrical Engineering, Stanford University, 1987.

[Arno 78] ARNOLD, M.: *Mathematical Methods of Classical Mechanics.* Springer Verlag, New York, 1978.

[ART 95] AGARWAL, P.K., P. RAGHAVAN und H. TAMAKI: *Motion Planning for a Steering-Constrained Robot through Moderate Obstacles.* In: *Proceedings of the 28th Annual ACM Symposium on Theory Computation*, Seiten 343–352, 1995.

[BaLa 89] BARRAQUAND, J. und J.-C. LATOMBE: *On Non-Holonomic Mobile Robots and Optimal Maneuvering*, Seiten 77–103. Revue d'Intelligence Artificielle, 3(2). Hermes, Paris, 1989.

[BaLa 90] BARRAQUAND, J. und J.-C. LATOMBE: *Controllability of Mobile Robots with Kinematic Constraints.* Technical Report No. STAN-CS-90-1317, Department of Computer Science, Stanford University, 1990.

[Beck 01] BECK, C.H. (Herausgeber): *Straßenverkehrsrecht (StVR).* Beck-Texte im DTV. DTV Verlag, 39. Auflage, 2001.

[BMWa] *Produktargumenter (E65).* Bayerische Motorenwerke Aktiengesellschaft, München, 2001.

[BMWb] *Owner's Handbook for Vehicle (E65).* Bayerische Motorenwerke Aktiengesellschaft, München, November 2001. (Engl. Ausgabe).

[Cai 88] CAI, C.: *Instantaneous Robot Motion with Contact Between Surfaces.* Report No. STAN-CS-88-1191, Department of Computer Science, Stanford University, 1988.

[CaLi 90] CANNY, J.F. und M.C. LIN: *An opportunistic global path planner.* In: *Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, Ohio*, Seiten 1554–1559, 1990.

[CaLi 93] CANNY, J.F. und M.C. LIN: *An opportunistic global path planner.* In: *Algorithmica, 10*, Seiten 102–120, 1993.

[Cann 88] CANNY, J.F.: *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[ChBu 94] CHOSET, HOWIE und JOEL BURDICK: *Sensor Based Planning and Nonsmooth Analysis.* Technical Report, Department of Mechanical Engineering, California Institute of Technology, 1994.

[Cutk 85] CUTKOSKY, M.: *Grasping and Fine Manipulation.* Kluwer Academic Publishers, Boston, 1985.

[Dask 95] DASKIN, M. S.: *Network and Discrete Location: Models, Algorithms and Applications.* John Wiley & Sons, New York, 1995.

[Doms 90] DOMSCHKE, W.: *Logistik: Rundreisen und Touren.* Oldenbourg Verlag, München, 3 Auflage, 1990.

[FMW 92] FRANCIS, R. L., L. F. MCGINNIS und J. A. WHITE: *Facility Layout and Location: An Analytical Approach.* Prentice Hall, Englewood Cliffs, 2 Auflage, 1992.

[FoWi 88] FORTUNE, S.J. und G.T. WILFONG: *Planning Constrained Motions.* In: *Proceedings of the 21st Annual ACM Symposium on Theory Computation*, Seiten 445–459, 1988.

[GoO' 97] GOODMAN, JACOB E. und JOSEPH O'ROURKE (Herausgeber): *Handbook of Diskrete and Computational Geometry.* CRC Press, Boca Raton, Florida, 1997.

[GudP 98] GUPTA, KAMAL und ANGEL P. DEL POBIL (Herausgeber): *Practical Motion Planning in Robotics: Current Approaches and Future Directions.* John Wiley & Sons, Chichester, England, 1998.

[HGB 99] HECTOR GONZÁLES-BAÑOS, ET AL.: *Planning Robot Motion Strategies for Efficient Model Construction.* In: *Proceedings of the Intl. Symp. on Robotics Research, Snowbird, Utah*, Seiten 345–352, Oktober 1999.

[JPLM] *JPL Marsyard.* Jet Propulsion Laboratory, Nasa, `http://telerobotics.jpl.nasa.gov/infrastructure/marsyard/homepage.html`, 2002.

[LaBu 99] LAUBACH, S. und J. BURDICK: *An Autonomous Sensor-Based Path-Planner for Planetary Microrovers.* In: *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'99), Detroit MI*, Mai 1999.

[Laph 97] LAPHAM, EDWARD: *Buick Experiments with Hands Free Driving.* Automotive News, 21. April 1997.

[LaSi 89] LAUMOND, J.P und T. SIMÉON: *Motion Planning for a Two Degrees of Freedom Mobile Robot with Towing.* Technical Report LAAS/CNRS 89-148, LAAS, Toulouse, 1989.

[Lato 91] LATOMBE, JEAN-CLAUDE: *Robot motion planning.* The Kluwer International Series in Engineering and Compuer Science. Kluwer Academic Publishers, Boston, 1991.

[Laub 99] LAUBACH, SHARON L.: *Sensor-based path planning using gaze control, for planetary rovers.* California Institute of Technology, `http://secretary.erc.caltech.edu/~burdick/research/sharon/rover_detaile%d.html`, 1999.

[Laum 86] LAUMOND, J.P.: *Feasible Trajectories for Mobile Robots with Kinematic and Environment Constraints.* In: *Preprints of the International Conference on Intelligent Autonomous Systems*, Seiten 346–354. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1986.

[LCS 89] Li, Z., J.F. Canny und S.S. Saastry: *On Motion Planning for Dexterous Manipulation, Part I: The Problem Formulation.* In: *Proceedings of the IEEE International Conference of Robotics and Automation, Scottsdale, AZ*, Seiten 775–780, 1989.

[LJTM 94] Laumond, J.P., P.E. Jacobs, M. Taix und R.M. Murray: *A motion planner for nonholonomic mobile robots.* In: *IEEE Transactions on Robotics and Automation*, Nummer 10, Seiten 577–593, 1994.

[LMW 88] Love, R. F., J. G. Morris und G. O. Wesolowsky: *Facilities Location: Models & Methods.* North-Holland, Amsterdam, 1988.

[LP 83] Lozano-Pérez, T.: *Spatial Planning: A Configuration Space Approach.* In: *Proceedings of the IEEE Transactions on Computers, C32(2)*, Seiten 108–120, 1983.

[Morr 98] Morris, John: *Dijkstra's Algorithm.* Electrical and Electronic Engineering, University of Western Australia, `http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/dijkstra.html`, 1998.

[PATH] *PATH - Partners for Advanced Transit and Highways.* `http://www.path.berkeley.edu/`, 2002.

[ReSh 85] Reif, J.H. und M. Sharir: *Motion Planning in the Presence of Moving Obstacles.* In: *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, Seiten 144–154, 1985.

[RWC] *Robot Work Crew (RWC).* Jet Propulsion Laboratory, Nasa, `http://prl.jpl.nasa.gov/projects/rwc/rwc_index.html`, 2002.

[SoLa 93] Souères, P. und J.P. Laumond: *Shortest path synthesis for a car-like robot.* In: *Proceedings of the 2nd European Control Conference*, 1993.

[ToJe 88] Tournassoud, P. und O. Jehl: *Motion Planning for a Mobile Robot with a Kinematic Constraint.* In: *Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia*, Seiten 1785–1790, 1988.

[Valeo] *Valeo.* Valeo Switches and Detection Systems, Créteil, France, `http://www.valeo.fr`, 2002.

[Wilf 88] Wilfong, G.: *Motion Planning for an Autonomous Vehicle.* In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Seiten 529–533, 1988.

[ZSTP 98] Zhang, Wei-Bin, Steve Shladover, Han-Shue Tan und PATH: *The Path Magnetic Guidance System.* Intellimotion, Research Updates in Intelligent Transportation Systems, 7(4), 1998.