Diplomarbeit*†

Trajektorienplanung und Trajektorienfolgeregelung im Konfigurationsraum nicht-holonomer Fahrzeuge

Christian Mandel

Mai 2002

*eingereicht im Studiengang Informatik, Fachbereich 3 der Universität Bremen
†erster Gutachter: Dr. Thomas Röfer, zweiter Gutachter: Prof.Dr. Bernd Krieg-Brückner

Zusammenfassung

In dieser Arbeit wird am Beispiel des *Bremer Autonomen Rollstuhls Rolland*, der im weiteren Verlauf *Rolland* genannt wird, die Entwicklung eines Navigationsmoduls dokumentiert, das den Roboter aus einer Startkonfiguration in eine beliebige Zielkonfiguration überführt. Eine Konfiguration setzt sich dabei aus der Position und Orientierung im Raum, dem aktuellen Lenkwinkel und der aktuellen Geschwindigkeit des Fahrzeuges zusammen.

In diesem Zusammenhang wird eine Klasse von Bahnen konstruiert, von denen jede aus Kreis-, Klothoiden- und Geradensegmenten zusammengesetzt ist. Die Bahnen, die Start- und Zielkonfiguration miteinander verbinden, weisen alle einen kontinuierlichen Krümmungsverlauf auf. Diese Eigenschaft entspricht der stetigen Lenkwinkelveränderung des Roboters während seiner Fahrt.

Eine nach ihrer Zusammensetzung aus vorwärts und rückwärts zu fahrenden Abschnitten ausgewählte Bahn wird mit einem Zeitgesetz kombiniert, das dem Roboter an jedem Punkt der Bahn eine einzuhaltende Geschwindigkeit zuordnet. Die anschließende Berechnung der für die Fahrt benötigten Referenz-Steuerungssignale wird aus dem kinematischen Modell von *Rolland* abgeleitet. Dieses wird schließlich mittels Eingangs-Ausgangs-Linearisierung in eine Form gebracht, die es erlaubt bei der Fahrt auftretende Abweichungen von der geplanten Bahn nachzuregeln.

Die Dokumentation von ausgewählten Testfahrten und eine abschließende Diskussion der erzielten Ergebnisse beschließen diese Arbeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Einordnung von Rolland in den Bereich der Robotik

Der Versuch, *Rolland* in seiner Gesamtheit als computergesteuertes, mechanisch angetriebenes Fahrzeug in den Bereich der Robotik einzuordnen, hält sich im Folgenden an zwei Sichtweisen. Die erste rückt die Realisation der Haupteigenschaft von *Rolland*, sich autonom fortzubewegen in den Vordergrund und wird dem Bereich der mobilen Robotik zugeordnet. Der zweite Ansatzpunkt beschränkt sich dagegen nicht auf die technischen Aspekte. Vielmehr wird hier (im Bereich der Servicerobotik) versucht die Anforderungen des Benutzers Mensch an die Maschine mit einzubeziehen.

1.1.1 Rolland in der mobilen Robotik

Nach einer umfassenden Studie des historischen Kontextes von mobilen Robotern, die beispielsweise ein frühes Modell eines Roboters mit autoähnlichen Fahreigenschaften, dem *Stanford Cart* enthält, werden in [3] die vier folgenden Aufgabenbereiche der mobilen Robotik definiert.

- Maschinenbau - Wie ist das Fahrzeug, insbesondere der Antriebsmechanismus aufgebaut?
- Informatik - Wie werden Umgebungs- und Fahrzeuginformationen mittels Sensoren erfasst und repräsentiert? Wie werden die Aufgaben des Roboters in Algorithmen umgesetzt?
- Elektrotechnik - Wie werden Teilsysteme in den Roboter integriert?
- Kognitive Psychologie und Wahrnehmung - Wie lösen biologische Organismen verwandte Probleme?

Diese Arbeit befasst sich hauptsächlich mit dem zweiten Punkt, die von *Rolland* zu lösende Aufgabe in einen Algorithmus umzusetzen. Die Fortbewegung, als Hauptaufgabe eines mobilen Roboters, kann dabei in die beiden Teilprobleme der vorwärts gerichteten und der inversen Kinematik unterteilt werden. Ersteres stellt die Frage, wie sich das Fahrzeug bei vorgegebenen Steuerungsbefehlen verhält, wohingegen im zweiten Fall eine im Voraus geplante Bewegung in Steuerungsbefehle konvertiert wird. Gegenstand der folgenden Abschnitte ist die inverse Kinematik. So werden einer unter Berücksichtigung von *Rollands* Fahreigenschaften konstruierten Bahn mit Hilfe des in Abschnitt 1.4 beschriebenen kinematischen Modells die entsprechenden Steuerungssignale zugeordnet.

Es sei an dieser Stelle bemerkt, dass in der mobilen Robotik eine Reihe weiterer, mit Rädern angetriebenen Plattformen untersucht wird. In diesem Zusammenhang werden in [3] die mit einem Differential- bzw. Synchronantrieb ausgestatteten Roboter untersucht. Beide Typen besitzen aufgrund ihres Antriebsmechanismus die für *Rolland* unmögliche Fähigkeit, ihre Fahrzeugausrichtung im Stand zu verstellen.

1.1.2 Rolland in der Servicerobotik

Um *Rolland* in den Bereich der Servicerobotik einzuordnen, soll zunächst folgende Definition deren Aufgabengebiet skizzieren.

Das Gebiet der Servicerobotik beinhaltet alle Aspekte eines Roboter-Systems, das mit Menschen im selben Arbeitsraum agiert. Dies impliziert viele, in der industriellen Robotik zu vernachlässigende, Aspekte wie Sicherheitsmerkmale, Mensch-Maschine-Schnittstellen, sowie einen höheren Grad an Autonomie in teilweise unstrukturierten Umgebungen.[9]

Nach dieser Definition zählen nicht nur mobile, sondern auch stationäre Roboter-Systeme zum Gebiet der Servicerobotik. Ihr Hauptmerkmal ist vielmehr, dass sie nicht zu Produktionszwecken eingesetzt werden. Aus dieser Beschränkung lassen sich all die Aufgabengebiete ableiten, die eine dem Menschen nützliche und unterstützende Funktion haben. Dazu gehören klassische Transport- und Reinigungsaufgaben in Büro- oder Klinikumgebungen, Überwachungsfunktionen in Kaufhäusern, Flughafengebäuden und anderen öffentlichen Gebäuden ebenso, wie die Unterstützung von durch Krankheit oder Behinderung betroffenen Menschen. Diese Aufzählung lässt sich beliebig erweitern. Festzuhalten bleibt, dass Serviceroboter mit ihrer Umwelt interagieren, und dabei auch ein Kontakt mit Menschen möglich sein kann.

Rolland gliedert sich in dieses Entwicklungsfeld als Rollstuhl der Firma *Meyra* ein, der an der Universität Bremen[12] mit einem Bordrechner und Sensorik ausgestattet wurde. Der daraus resultierenden mobilen Plattform können in Verbindung mit den auf ihr implementierten Algorithmen die oben definierten Merkmale eines Serviceroboters zugeordnet werden. So wird mit der in [5] dokumentierten, und in Abschnitt 4.1.1 näher beschriebenen, Softwarekomponente SAM den Sicherheitsanforderungen an einen den Menschen befördernden Roboter Rechnung getragen. Weiterhin wird die Mensch-Maschine-Schnittstelle momentan durch die Entwicklung eines Spracheingabemoduls aufgebaut. Mit dieser Komponente werden assistierende Funktionen intuitiv aufrufbar sein. Letztlich kann das im Rahmen dieser Arbeit entwickelte Navigationsmodul als Beitrag zur Verbesserung der Autonomie gesehen werden, da es dem Roboter ermöglicht, sich selbständig aus seiner aktuellen Konfiguration in eine vorgegebene Zielkonfiguration zu überführen.

1.2 Einführende Fragestellungen

Eine einfache Übersetzung des Titels dieser Arbeit könnte lauten: „Das Planen und das Abfahren einer Bahn aus einer Konfiguration A in eine Konfiguration B, für ein Fahrzeug mit autoähnlichen Fahreigenschaften“. Diese zugängliche Beschreibung hat jedoch den Nachteil einiger Unschärfen. So stellen sich in Zusammenhang mit der Aufgabenstellung erste Fragen wie:

- Welche formale Beschreibung genügt den abstrakten Begriffen Konfiguration, Bahn und Trajektorie?

- Wie lassen sich die Fahreigenschaften eines Automobils hilfreich beschreiben?
- Was macht eine geplante Bahn aus? Schließlich gibt es unendlich viele Möglichkeiten ein Fahrzeug aus seiner Startkonfiguration in eine Zielkonfiguration zu überführen.
- Wie werden die notwendigen Steuerungsbefehle wie „fahre mit 2 km/h“ oder „lenke 5° nach links“ generiert?
- Falls die von dem Fahrzeug gefahrene Bahn von der zuvor geplanten durch äußere Störeinflüsse abweicht: Wie werden dann die Steuerungsbefehle nachgeregelt?

Um ein Fahrzeug wie *Rolland* computergesteuert in der Realität zu navigieren, bedarf es jedoch weit mehr als der Beantwortung von Fragen zur Bahnplanung und Bahnregelung. Die hier gewählte Beschränkung auf das Planen einer Bahn in einem hindernisfreien Raum wird in [4] *steering method* genannt. Eine solche Herangehensweise ermöglicht die Realisierung einer Softwareschicht, die der Lösung eines deutlich abgegrenzten Aufgabengebietes dient.

Weitere zur Realisierung eines alltagstauglichen Fahrzeuges notwendige und in dieser Arbeit nicht berücksichtigte Softwarekomponenten haben folgende Aufgaben zu lösen:

- Zur präzisen Positionsregelung bedarf es einer guten Selbstlokalisation.
- Hindernisse müssen daraufhin analysiert werden, ob sie eine Gefährdung bei der Verfolgung der aktuellen Bahn darstellen.
- Bei gegebener Kollisionsgefahr muss die Bahn verworfen und eine neue, hindernisfreie Bahn geplant werden.

Diese Schichten der Softwarehierarchie greifen dabei auf Umgebungsinformationen zurück, die kontinuierlich durch fahrzeuginterne Sensorik wie Ultraschallsensoren und einen Laserscanner gewonnen werden.

Im folgenden Abschnitt werden nun grundlegende Definitionen gegeben. Diese erlauben es, die zuvor gestellten Probleme der Trajektorienplanung und Trajektorienfolgeregelung im mathematischen Sinne greifbar zu machen.

1.3 Grundlegende Definitionen

Rolland ist ein Beispiel für einen Roboter, der in seinen Fahreigenschaften einem Automobil gleicht. Ein solches zeichnet sich durch seine starre Karosserie und den zwei parallel zueinander liegenden Achsen aus. Die daran montierten Räder stellen den Kontakt zum Boden her. Das sich in idealisierter Weise auf einer planen Ebene bewegende Fahrzeug besitzt zwei angetriebene und zwei gelenkte Räder. Bei *Rolland* liegt die vorantreibende Kraft an den Rädern der Vorderachse an, während die beiden Hinterräder lenkbar sind. Der Abstand zwischen den beiden Achsen sei mit l bezeichnet. Wird die Ebene, auf der sich das Fahrzeug bewegt, als globales Koordinatensystem aufgefasst, dann ist es sinnvoll die Position des Fahrzeuges als Vektor zu beschreiben. Dieser Referenzpunkt ist bei *Rolland* der Mittelpunkt der Vorderachse und dient gleichzeitig als

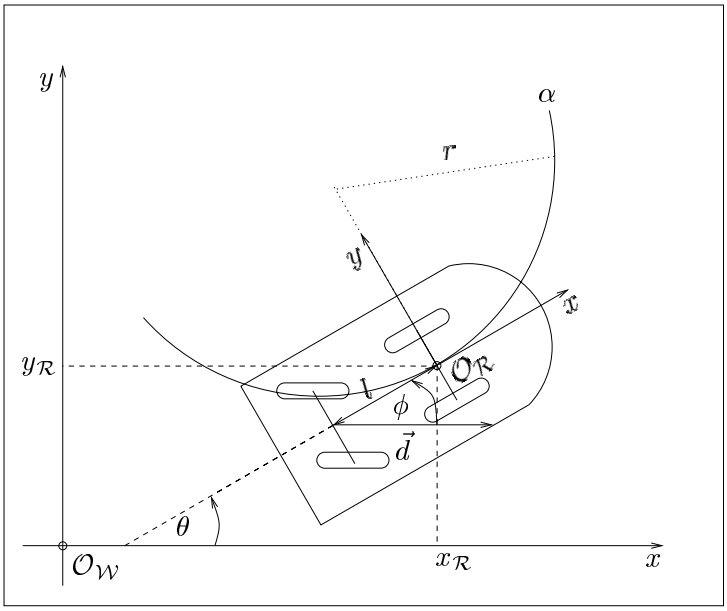


Abbildung 1: Schematische Darstellung von *Rolland*

Ursprung eines lokalen, dem Roboter eigenen Koordinatensystems. Die Ausrichtung des Fahrzeuges kann so als Winkel zwischen den x-Achsen der beiden Koordinatensysteme beschrieben werden. Um eine eindeutige Beschreibung zu gewährleisten, ist es üblich die Fahrzeugausrichtung auf das Intervall $]-\pi...\pi[$ zu beschränken. Der Lenkwinkel des Fahrzeuges ist durch den Winkel zwischen der lokalen x-Achse und der Ausrichtung der gelenkten Räder gegeben. Er ist durch mechanische Zwangsbedingungen beschränkt. Das bedeutet, dass die gelenkten Räder niemals senkrecht zur Ausrichtung des Fahrzeuges stehen können. Eine eindeutige Beschreibung des Roboters ist durch seinen Referenzpunkt, seine Ausrichtung und seinen Lenkwinkel gegeben. In Anlehnung an [6] sollen die soeben beschriebenen Punkte nun formal erfasst werden:

- Der Arbeitsraum des Roboters ist die euklidische Ebene $\mathcal{W} = \Re^2$. Diese sei mit einem stationären Koordinatensystem $\mathcal{F}_\mathcal{W}$ ausgestattet, dessen Ursprung mit $\mathcal{O}_\mathcal{W}$ bezeichnet wird.

- Der sich in $\mathcal{W}$ bewegende Roboter $\mathcal{R}$ wird als kompakte Untermenge von $\mathcal{W}$ beschrieben. Sein Referenzpunkt sei mit $\mathcal{O}_\mathcal{R}$ bezeichnet. Dieser Punkt ist Ursprung eines sich in $\mathcal{F}_\mathcal{W}$ bewegenden Koordinatensystems $\mathcal{F}_\mathcal{R}$. Dieses erlaubt es, jeden Punkt des Roboters mit festen Koordinaten in $\mathcal{F}_\mathcal{R}$ zu beschreiben.

- Eine Konfiguration von $\mathcal{R}$ ist durch die Position von $\mathcal{O}_\mathcal{R}$ innerhalb von $\mathcal{F}_\mathcal{W}$, der Rotation von $\mathcal{F}_\mathcal{R}$ bezüglich $\mathcal{F}_\mathcal{W}$ und dem Winkel zwischen $\vec{d}$ und der x-Achse von $\mathcal{F}_\mathcal{R}$ gegeben. Sie wird mit $q = [x, y, \theta, \phi]$ bezeichnet. Es gilt: $x, y \in \Re$, $\theta \in]-\pi...\pi]$ und $\phi \in [-\phi_{max}...\phi_{max}]$ mit $\mid \phi_{max} \mid < \frac{\pi}{2}$.

- Der Konfigurationsraum $\mathcal{C}$ beinhaltet alle möglichen Konfigurationen q von $\mathcal{R}$.

Wie in Abbildung 1 zu sehen ist, befindet sich $\mathcal{R}$ in einer Konfiguration auf einem Kreissegment α. Diese sei mit q_{start} bezeichnet. Um den Begriff der Bahn anschaulich zu machen, stelle man sich vor, dass $\mathcal{R}$ aus q_{start} in eine Konfiguration q_{ziel} fährt. Der Vektor $[x_{ziel},y_{ziel}]$ liege dabei auf α. Es ist offensichtlich, dass $\mathcal{O}_{\mathcal{R}}$ bei dieser Fahrt eine stetige Kurve beschreibt. Diese überlagert bei konstantem Lenkwinkel ϕ das Kreissegment α. Aus dieser Überlegung ergibt sich die Standarddefinition einer Bahn.

Eine Bahn eines Roboters $\mathcal{R}$ von q_{start} nach q_{ziel} ist die stetige Abbildung

$$\tau : [0...1] \rightarrow \mathcal{C} \ , \ mit \ \tau(0) = q_{start}, \tau(1) = q_{ziel} \tag{1}$$

Üblicherweise gibt der Parameter λ die von q_{start} aus auf der Bahn gefahrene Strecke des Fahrzeuges an. Wird dem Roboter nun ein Zeitgesetz zugeordnet, das beschreibt, zu welchem Zeitpunkt er sich in einer bestimmten Konfiguration der Bahn befinden soll, ergibt sich die Definition einer Trajektorie.

Eine Trajektorie eines Roboters $\mathcal{R}$ von q_{start} nach q_{ziel} ist die stetige Abbildung

$$\upsilon : [0...T] \rightarrow \mathcal{C} \ , mit \ \upsilon(0) = q_{start}, \upsilon(T) = q_{ziel} \tag{2}$$

Zum Zeitpunkt $t = 0$ befindet sich das Fahrzeug in der Konfiguration q_{start}, nimmt dann seine Fahrt auf und beendet sie zum Zeitpunkt $t = T$ in der Konfiguration q_{ziel}.

1.4 Kinematisches Modell

Um die Konfiguration einer mobilen Plattform wie *Rolland* zu verändern, stehen zwei Stellgrößen zur Verfügung, die jedem Fahrer eines Automobils bekannt sind. Mit der einen kann dieser die Geschwindigkeit seines Fahrzeuges beeinflussen. Sie hat die Einheit $\frac{Strecke}{Zeit}$ und wird im folgenden mit $v1$ bezeichnet. Die zweite Stellgröße beschreibt die Lenkwinkelveränderung über die Zeit. Sie wird mit $v2$ bezeichnet und hat die Einheit $\frac{Winkel}{Zeit}$. Es sei gefordert, dass bei unendlich kleinem Voranschreiten der Zeit $v1$ und $v2$ unendlich kleinen Veränderungen unterliegen. Das bedeutet: $\lim_{\Delta t \rightarrow 0} \Delta v1 = 0$. Bei der Aufstellung des kinematischen Modells von *Rolland* wird die Annahme gemacht, dass sich die Räder des Roboters entweder rollend auf der Ebene $\mathcal{W}$ bewegen oder still stehen. Diese Vereinfachung erlaubt es, in der Realität beobachtbare Effekte zu vernachlässigen. Dort kann es z.B. vorkommen, dass ein Rad mangels Bodenhaftung in eine Richtung rutscht, die nicht der Ausrichtung des Rades entspricht.

Es soll nun der Geschwindigkeitsvektor $\dot{q} = [\dot{x}, \dot{y}, \dot{\theta}, \dot{\phi}]$ von $\mathcal{R}$ beschrieben werden. Wie in Abbildung 1 zu erkennen ist, bewegt sich der Referenzpunkt des Roboters zu jedem Zeitpunkt in die Richtung seiner ihm eigenen x-Achse. Deren Ausrichtung im globalen Koordinatensystem wird durch die Fahrzeugausrichtung θ beschrieben. Durch einfache Betrachtung der Winkelfunktionen am mathematischen Einheitskreis ergibt sich $\dot{x} = cos(\theta)v1$ und $\dot{y} = sin(\theta)v1$. Die Beschreibung der Geschwindigkeitskomponente $\dot{\theta}$ folgt der Beobachtung, dass sich der Referenzpunkt $\mathcal{O}_{\mathcal{R}}$ des Roboters auf einer Kurve mit dem Krümmungsradius $r = \frac{l}{tan\phi}$ in dem Punkt $\mathcal{O}_{\mathcal{R}}$ bewegt. Die Krümmung $k = \frac{tan\phi}{l}$ in diesem Punkt ist als Kehrwert des Krümmungsradius definiert und somit direkt abhängig

von dem Lenkwinkel ϕ. Wird k mit $v1$ multipliziert, ergibt sich die Geschwindigkeit der Fahrzeugausrichtung als $\dot{\theta} = \frac{tan\phi}{l} v1$. Die gesamte kinematische Gleichung von *Rolland* ist einem Beispiel aus [8], dem eines hinterradangetriebenen und vorderradgelenkten Fahrzeuges, entnommen.

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} cos\theta \\ sin\theta \\ \frac{tan\phi}{l} \\ 0 \end{bmatrix} v1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v2 \tag{3}$$

Die zuvor beschriebene Beschränkung des Lenkwinkels auf das Intervall $[-\phi_{max}...\phi_{max}]$ mit $\mid \phi_{max} \mid < \frac{\pi}{2}$ schließt die Singularität in dem Ausdruck $\frac{tan\phi}{l}$ aus.

1.5 Nicht-holonome Beschränkungen

Bei der Frage nach den für ein Automobil typischen Fahreigenschaften spielt der Begriff der *nicht-holonomen Beschränkungen* eine wichtige Rolle. Auf der hindernisfreien Ebene ist es offensichtlich, dass ein Fahrzeug durch wiederholtes Rangieren jede beliebige Konfiguration einnehmen kann. Die einzelnen Komponenten einer Konfiguration werden in diesem Zusammenhang auch Freiheitsgrade genannt. Ebenfalls klar ist, dass eine zur Fahrzeugausrichtung senkrechte Bewegung unmöglich ist. Diese Eigenschaft lässt sich als Gleichung zwischen den Freiheitsgraden q und deren erster Ableitung nach der Zeit $\dot{q}$ beschreiben. Sie hat nach [7] folgende allgemeine Form:

$$G(q, \dot{q}, t) = G(q_1, ..., q_m, \dot{q_1}, ..., \dot{q_m}, t) = 0 \tag{4}$$

G ist hier eine Funktion, die stetige partielle Ableitungen beliebiger Ordnung besitzt. Sie beschränkt das Fahrzeug, das sich zum Zeitpunkt t in der Konfiguration q befindet, in seinen möglichen Geschwindigkeitskomponenten $\dot{q}_i$. Wenn sich durch Integration der Gleichung 4 der Geschwindigkeitsvektor $\dot{q}$ eliminieren lässt, ergibt sich nach [7] die allgemeine Form einer holonomen Zwangsbedingung:

$$H(q, t) = H(q_1, ..., q_m, t) = 0 \tag{5}$$

Genauso wie G in der Gleichung 4, ist H eine glatte Funktion. Hier beschreibt sie jedoch, in welcher Konfiguration q sich der Roboter zum Zeitpunkt t nicht befinden kann. Üblicherweise wird in dieser Form die Einschränkung des Konfigurationsraumes $\mathcal{C}$ durch Hindernisse oder mechanische Zwangsbedingungen, wie die Beschränkung des Lenkwinkels, beschrieben. Die nicht-holonome Beschränkung von *Rolland* ergibt sich in Form der Gleichung 4 wie folgt:

$$G(q, \dot{q}, t) = -\dot{x}\ sin\theta + \dot{y}\ cos\theta = 0\ \forall t \in \Re^+ \tag{6}$$

Die Beschränkung des Lenkwinkels ϕ von $\mathcal{R}$ auf das Intervall $[-\phi_{max}...\phi_{max}]$ soll nun noch in Form der Gleichung 5 beschrieben werden:

$$H(q, t) = \phi_{max} - \mid \phi \mid \geq 0\ \forall t \in \Re^+ \tag{7}$$

1.6 Ausblick auf die Arbeit

Das bisher auf *Rolland* implementierte Verfahren zur Bahnplanung wird in Abschnitt 2.3 vorgestellt. Dabei stellt sich heraus, dass zwischen den verwendeten Geraden und Kreissegmenten eine unstetige Krümmungsentwicklung der Bahn existiert. Da der Roboter an diesen Stellen eine Lenkwinkelveränderung in Nullzeit vollziehen oder einen Zwangsstopp einlegen müsste, wird in Abschnitt 2.2 eine neue Klasse von Bahnsegmenten, die Klothoide, eingeführt.

Die schließlich aus Kreissegmenten, Geraden und Klothoiden zusammengesetzten Bahnen weisen einen kontinuierlichen Krümmungsverlauf auf. Diese werden mit einem Zeitgesetz kombiniert (Abschnitt 2.5), das dem Roboter in jeder Konfiguration der Bahn eine passende Geschwindigkeit zuordnet.

Um der resultierenden Trajektorie zu folgen, wird in Abschnitt 3 das kinematische Modell von *Rolland* mittels *Eingangs-Ausgangs-Linearisierung* entkoppelt und ein passendes Regelungsgesetz aufgestellt.

Der Abschnitt 4 beschreibt das für diese Arbeit implementierte Navigationsmodul, sowie ein Visualisierungsprogramm, welches die in Abschnitt 2.4 eingeführten Bahntypen darstellt.

Abschließend werden in Abschnitt 5 die hier vorgestellte Trajektorienplanung sowie repräsentative Testläufe diskutiert.

2 Bahnplanung und Trajektoriengenerierung

2.1 Anforderungen an die Bahnplanung

Die Aufgabe der Bahnplanung ist es, eine geometrische Beschreibung der Bewegung des Punktes $\mathcal{O}_\mathcal{R}$ zu finden, die dieser bei der Fahrt des Roboters $\mathcal{R}$ von q_{start} nach q_{ziel} beschreiben soll. Die resultierende Kurve ist so zu wählen, dass sie die in Abschnitt 1.4 angegebene Eigenschaft eines kontinuierlichen Krümmungsverlaufes besitzt. Ist sie zudem zweimal stetig differenzierbar, können die fehlenden Komponenten θ und ϕ einer beliebigen Bahnkonfiguration berechnet werden. Die resultierende Bahn soll in Form der Gleichung 1 beschrieben werden.

2.2 Bahnsegmente: Kreis, Gerade und Klothoide

Bei der Suche nach einer von q_{start} nach q_{ziel} zu fahrenden Bahn ist es offensichtlich, dass es eine unendlich große Menge an Lösungsmöglichkeiten gibt. Dem Fahrer eines Automobils wird dies deutlich, wenn er sich überlegt, wie er sein Fahrzeug aus seiner Garage rangiert. Auch wenn aus idealisierter Sicht die Start- und Zielkonfigurationen jedes Mal dieselben sind, wird die Bahn, die er dabei abfährt immer eine andere sein. Um das Planungsproblem algorithmisch zu lösen und die resultierenden Kurven analytisch beschreibbar zu machen, wird eine Bahn aus verschiedenen Typen von Kurven zusammengesetzt, die alle den Anforderungen der Bahnplanung genügen. Dabei ist insbesondere an den Übergangspunkten zwischen den einzelnen Kurvensegmenten auf die Einhaltung der Forderung nach einer kontinuierlichen Krümmungsänderung zu achten. Im Folgenden werden die verwendeten Kurventypen Gerade, Kreis und Klothoide beschrieben.

2.2.1 Bahnsegment: Gerade

Die in Abbildung 2 dargestellte Gerade ist ein Bahnsegment, das die Bewegung des Roboters bei einem konstanten Lenkwinkel $\phi = 0$ beschreibt. Wird λ als die Strecke interpretiert, die sich $\mathcal{O}_\mathcal{R}$ von $[x_{start}, y_{start}]$ aus auf der Geraden entlang bewegt hat, so lässt sich die folgende Bahngleichung aufstellen:

$$\begin{aligned} \tau_{Gerade} : [0...1] &\rightarrow \Re^2 \\ \lambda &\mapsto \begin{bmatrix} x_{start} + \lambda(x_{ziel} - x_{start}) \\ y_{start} + \lambda(y_{ziel} - y_{start}) \end{bmatrix} \end{aligned} \tag{8}$$

Die Gerade ist ein Beispiel für eine entartete Kurve. Sie hat den Krümmungsradius $r = \infty$ und die Krümmung $k = 0$.

2.2.2 Bahnsegment: Kreisausschnitt

Der zweite von der Bahnplanung verwendete Kurventyp ist der in Abbildung 2 dargestellte Kreisausschnitt. Dieses Bahnsegment beschreibt die Bewegung von $\mathcal{O}_\mathcal{R}$ bei einem

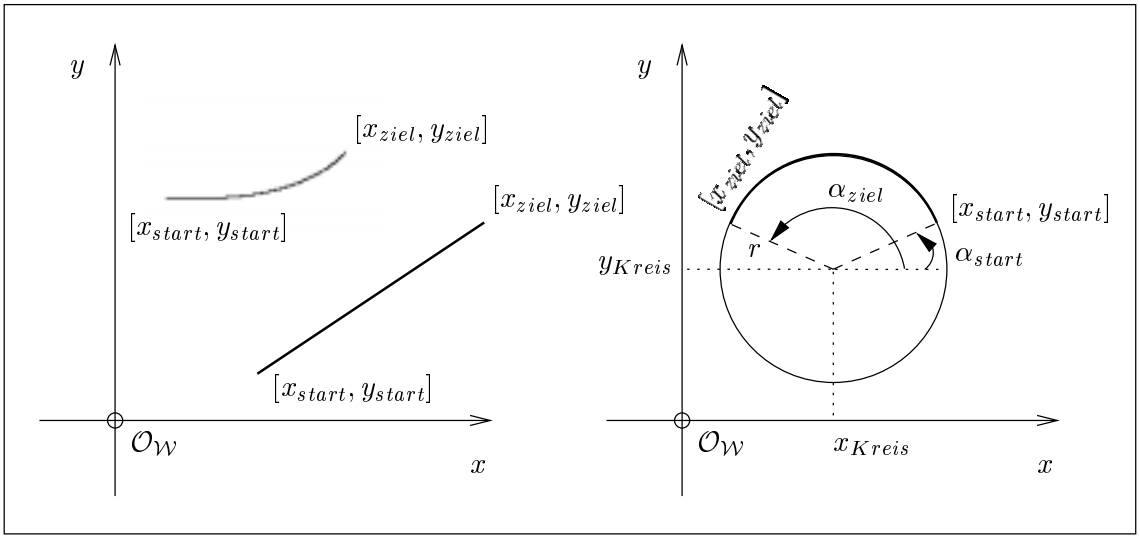


Abbildung 2: Bahnsegmente: Gerade, Klothoidenausschnitt und Kreisausschnitt

konstanten Lenkwinkel $\phi \neq 0$. In Zusammenhang mit dem Radstand l und dem Lenkwinkel ϕ von $\mathcal{R}$ hat der Kreisausschnitt an jeder Stelle den Krümmungsradius $r = \frac{l}{tan\phi}$ und die Krümmung $k = \frac{tan\phi}{l}$. Wird λ wieder als die Strecke interpretiert, die der Roboter auf dem Kreisausschnitt zurückgelegt hat, kann folgende Bahngleichung aufgestellt werden:

$$\begin{aligned} \tau_{Kreis} : [0...1] &\rightarrow \Re^2 \\ \lambda &\mapsto \begin{bmatrix} rcos(\alpha_{start} + \lambda(\alpha_{ziel} - \alpha_{start})) + x_{Kreis} \\ rsin(\alpha_{start} + \lambda(\alpha_{ziel} - \alpha_{start})) + y_{Kreis} \end{bmatrix} \end{aligned} \tag{9}$$

2.2.3 Bahnsegment: Klothoidenausschnitt

Im Gegensatz zur Geraden und zum Kreisausschnitt wird mit der Klothoide die Bewegung des Punktes $\mathcal{O}_{\mathcal{R}}$ bei variablem Lenkwinkel ϕ modelliert. So wächst bei der Klothoide, die auch Spinnkurve genannt wird, die Krümmung r im Kurvenverlauf linear an. Dieses Verhalten entspricht einer Situation, in der *Rolland* bei konstanter Geschwindigkeit $v1$ seinen Lenkwinkel ϕ mit konstanter Geschwindigkeit $v2$ verändert. Die im Folgenden angegebenen Eigenschaften einer Klothoide beziehen sich auf [1]. Im Punkt $[x_{start}, y_{start}]$ hat die Klothoide den Krümmungsradius $r_{start} = \infty$. Dieser entspricht einem Lenkwinkel $\phi_{start} = 0$ des Roboters. Hat das Fahrzeug im Punkt $[x_{ziel}, y_{ziel}]$ den Lenkwinkel $\phi_{ziel} \neq 0$ eingestellt, ist der Krümmungsradius dort durch $r_{ziel} = \frac{l}{tan\phi_{ziel}}$ gegeben. Die Länge l_K des zwischen diesen beiden Punkten liegenden Klothoidenausschnittes ist definiert als:

$$l_K = \frac{a_K^2}{r_{ziel}} \tag{10}$$

Die Konstante a_K gibt in dieser Gleichung an, wie stark die Krümmung der Klothoide mit größer werdender Kurvenlänge ansteigt. Je größer a_K ist, um so kleiner ist der Anstieg. Die für eine Aufstellung der Bahngleichung erforderliche explizite Darstellung der Klothoidenkoordinaten ist durch folgende Gleichungen gegeben[1]. Der Parameter l bezeichnet die Bogenlänge der Kurve bis zu dem Punkt $[x_l, y_l]$:

$$x_l = \int_0^l cos\frac{l^2}{2a_K^2}dl, \quad y_l = \int_0^l sin\frac{l^2}{2a_K^2}dl \tag{11}$$

Da sich diese sogenannten *Fresnel'schen Integrale* nicht geschlossen integrieren lassen, müssen sie mit Hilfe einer Reihenentwicklung angenähert werden. Die ersten fünf Glieder erlauben nach [1] eine Genauigkeit im mm-Bereich.

$$\begin{aligned} x_l &= l - \frac{l^5}{40a_K^4} + \frac{l^9}{3456a_K^8} - \frac{l^{13}}{599040a_K^{12}} + \frac{l^{17}}{175472640a_K^{16}} \mp ... \\ y_l &= \frac{l^3}{6a_K^2} - \frac{l^7}{336a_K^6} + \frac{l^{11}}{42240a_K^{10}} - \frac{l^{15}}{9676800a_K^{14}} + \frac{l^{19}}{3530096640a_K^{18}} \mp ... \end{aligned} \tag{12}$$

Abschließend wird nun die Bahngleichung eines Klothoidenausschnittes in Form der Gleichung 1 angegeben:

$$\begin{aligned} \tau_{Klothoide} : [0...1] &\rightarrow \Re^2 \\ \lambda &\mapsto \begin{bmatrix} x_{start} + x_{\lambda l_K} \\ y_{start} + y_{\lambda l_K} \end{bmatrix} \end{aligned} \tag{13}$$

2.3 Bahnplanung mit Kreisausschnitten und einer Geraden

In diesem Abschnitt wird ein Bahnplanungsverfahren vorgestellt, das den Roboter auf einer aus Kreissegmenten und einer Geraden zusammengesetzten Bahn von seiner Startkonfiguration in eine Zielkonfiguration überführt. Es wurde von der Wegplanungsgruppe des studentischen Projektes *Roses* vom Wintersemester 1998 bis zum Sommersemester 2000 auf *Rolland* implementiert und in [2] dokumentiert.

Die Grundidee hinter dem verwendeten Verfahren[1] ist folgende: An den Referenzpunkt des Roboters werden in q_{start} und q_{ziel} jeweils zwei Kreise mit dem Radius $r = \frac{l}{tan\phi_{max}}$ gelegt, sodass die x-Achse des dem Roboter eigenen Koordinatensystems tangential zu diesen Kreisen liegt. Die Kreise repräsentieren die Bahnen des Roboters, wenn er aus der Start- oder Zielkonfiguration mit maximalem Lenkwinkel $\phi = \pm\phi_{max}$ vorwärts bzw. rückwärts fährt. Zwischen einen ausgewählten Start- und Zielkreis werden dann jeweils zwei innere und zwei äußere Verbindungstangenten gelegt. Jede von ihnen repräsentiert eine mögliche Bahn des Roboters, die Start- und Zielkreis miteinander verbindet. Diese Situation ist in Abbildung 3 für den Fall dargestellt, in dem der Lenkwinkel auf dem Startkreis positiv und auf dem Zielkreis negativ ist. Insgesamt ergibt sich eine Menge von 64 Bahnen, zusammengesetzt aus einem

[1] Dieses Verfahren wird im Folgenden KGK-Bahnplanung genannt.

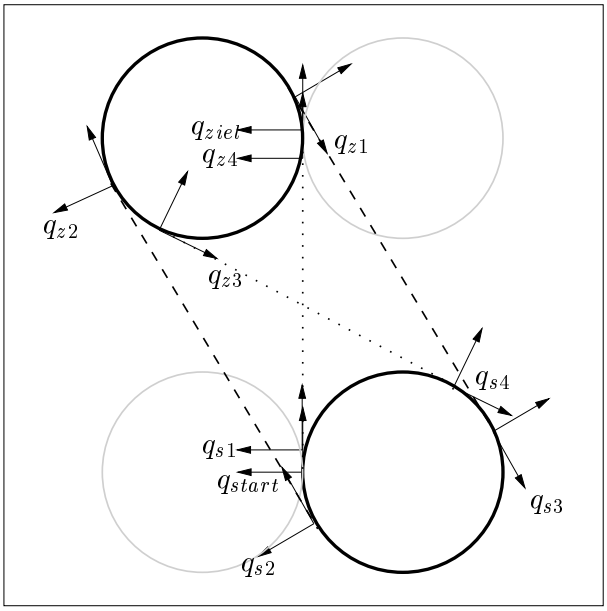


Abbildung 3: Bahn aus einer Geraden und zwei Kreisausschnitten

Startkreis $Kreis_{start} \in \{vorw._{links}, vorw._{rechts}, rückw._{links}, rückw._{rechts}\}$, einer Verbindungsgeraden $Gerade \in \{q_{s1}\vec{q}_{z4}, q_{s2}\vec{q}_{z2}, q_{s3}\vec{q}_{z1}, q_{s4}\vec{q}_{z3}, \}$ und einem Zielkreis $Kreis_{ziel} \in \{vorw._{links}, vorw._{rechts}, rückw._{links}, rückw._{rechts}\}$. Um die gewünschte Zielausrichtung zu erreichen, können immer nur zwei Tangenten in der Praxis befahren werden, was zu einer Menge von 32 möglichen Bahnen führt. In Abbildung 3 sind dies die inneren Tangenten. In dem Fall, bei dem sich die zu fahrenden Start- und Zielkreise überschneiden, fallen die Verbindungsgeraden weg und die Menge der möglichen Bahnen reduziert sich auf 16.

Nach diesem Verfahren geplante Bahnen haben den Nachteil, dass der Lenkwinkel ϕ in den Verbindungskonfigurationen $q_{s1}, ..., q_{s4}, q_{z1}, ..., q_{z4}$ einen Sprung macht. Während auf den Kreissegmenten $\phi \neq 0$ gilt, hat ϕ auf den Geraden offensichtlich den Wert 0. Dieses Problem kann zudem in q_{start} und q_{ziel} auftreten, wenn der dort vorgegebene Lenkwinkel ungleich $\pm\phi_{max}$ ist. Um ein exaktes Abfahren der geplanten Bahn sicherzustellen, müsste der Roboter in diesen Konfigurationen seinen neuen Lenkwinkel im Stand einstellen und dann seine Fahrt aufnehmen, beenden oder fortsetzen. Diese unerwünschte Anforderung an eine zu fahrende Bahn dient als Motivation bei der Suche nach einem Algorithmus, der Bahnen mit einer stetigen Krümmungsradiusentwicklung und somit einem sich stetig entwickelnden Lenkwinkel generiert.

2.4 Bahnplanung mit Kreisausschnitten, Klothoidensegmenten und einer Geraden

Im Folgenden werden Erweiterungen der KGK-Bahnplanung vorgestellt, die zu einem neuen Bahnplanungsverfahren führen. Insbesondere wird das in Abschnitt 2.2 eingeführte

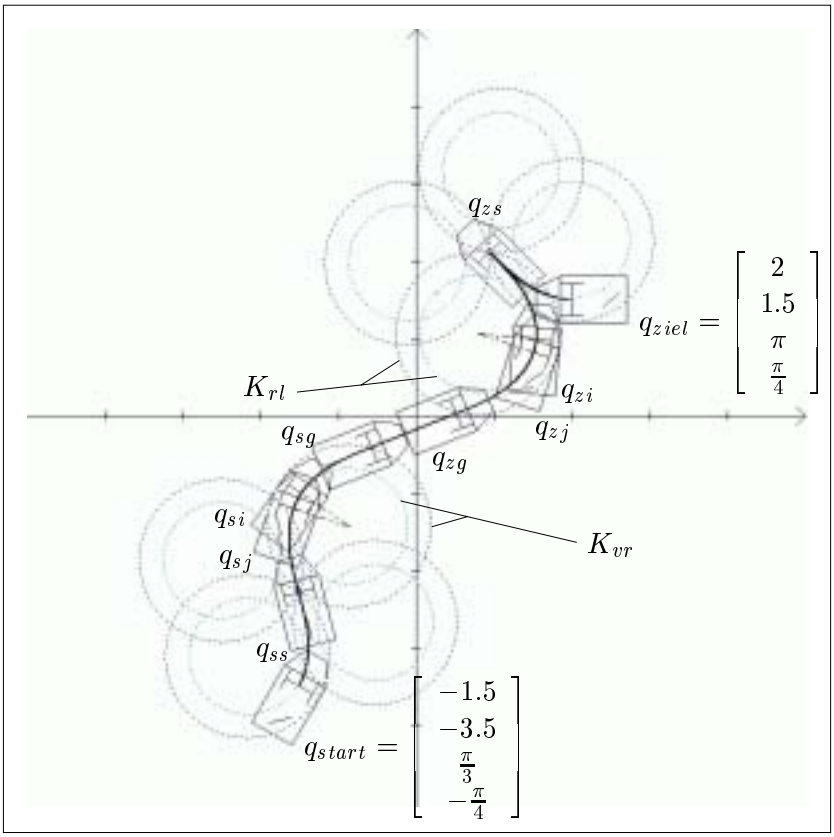


Abbildung 4: Bahn aus Kreisausschnitten, Klothoidensegmenten und einer Geraden

Klothoidensegment dazu benutzt, einen kontinuierlichen Krümmungsverlauf der gesamten Kurve zu garantieren. Das im Rahmen dieser Arbeit auf *Rolland* implementierte Verfahren basiert auf [14].

2.4.1 Grundidee

Das hier beschriebene Verfahren[2], eine Bahn aus Kreisausschnitten, Klothoidensegmenten und einer Geraden zu planen, erweitert die KGK-Bahnplanung in dem Sinne, dass die bei der KGK-Bahnplanung verwendeten Kreisausschnitte nun durch eine Klothoidensegment-Kreisausschnitt-Klothoidensegment Kombination[3] ersetzt werden. Ein CCT repräsentiert dabei die Bahn des Roboters auf der sein Lenkwinkel kontinuierlich von 0 über $\pm\phi_{max}$ wieder nach 0 variiert. Abbildung 4 zeigt einen Screenshot des im Rahmen dieser Arbeit entstandenen Visualisierungstools *Rolland-Bahnplaner*. Die dort dargestellte Bahn soll nun als einleitendes Beispiel dienen.

[2]Dieses Verfahren wird nach [14] im weiteren Verlauf CCT-Bahnplanung genannt. CCT steht dabei für *Continuous Curvature Turn*.
[3]Eine solche Kombination wird im Weiteren nur noch CCT genannt.

In der Konfiguration q_{start} hat $\mathcal{R}$ einen negativen und in q_{ziel} einen positiven Lenkwinkel. In einem solchen Fall wird der Roboter zuerst über ein Klothoidensegment in eine Konfiguration überführt, in der $\phi = 0$ gilt. Diese wird in der Startsituation mit q_{ss} und in der Zielsituation mit q_{zs} bezeichnet. Beide Konfigurationen dienen als Ausgangspunkt für die eigentliche Bahnplanung. In einem ersten Schritt werden an sie jeweils vier Klothoiden mit zunehmender Krümmung gelegt. Sie repräsentieren die Fahrt des Roboters, bei der dieser in Vorwärts- bzw. Rückwärtsfahrt seinen maximalen Lenkwinkel $\pm\phi_{max}$ einstellt. Die so erreichten Konfigurationen werden mit q_{si} und q_{zi} bezeichnet. Von ihnen aus beschreibt der Referenzpunkt des Roboters bei konstantem Lenkwinkel in Vorwärts- oder Rückwärtsfahrt einen Kreisausschnitt, der in den Konfigurationen q_{sj} bzw. q_{zj} endet. Um nun einen Lenkwinkel von 0 zu erreichen, werden die Konfigurationen q_{sj} und q_{zj} mit Klothoiden verbunden, deren Krümmung abnimmt. Diese enden in den Konfigurationen q_{sg} und q_{zg}, die schließlich mit einer Geraden verbunden werden.

Wie der Abbildung 4 entnommen werden kann, liegen die Konfigurationen q_{ss}, q_{si}, q_{sj} und q_{sg} (bzw. deren Entsprechungen in der Zielsituation) eines CCT auf gestrichelten Kreispaaren. Diese spielen eine Schlüsselrolle bei der CCT-Bahnplanung. Von ihnen existieren an q_{ss} und q_{zs} jeweils vier. Ihre Bezeichnungen K_{vl}, K_{vr}, K_{rl} und K_{rr} resultieren aus der Fahrtrichtung, die der Roboter bei einem entsprechenden CCT einschlägt. Die an erster Stelle stehenden Indexe v und r bezeichnen dabei eine Vorwärts- bzw. Rückwärtsfahrt, wohingegen r und l an zweiter Position eine Links- bzw. Rechtskurve charakterisieren. Eine geplante Bahn besteht schließlich aus der Kombination eines CCT des entsprechenden Kreispaares $K_{start} \in \{K_{vl}, K_{vr}, K_{rl}, K_{rr}\}$ mit einem Geradensegment $Gerade = q_{sg}\vec{q}_{zg}$ und einem CCT des Kreispaares $K_{ziel} \in \{K_{vl}, K_{vr}, K_{rl}, K_{rr}\}$. Dies ergibt eine Menge von 16 möglichen Bahnen. In dem in Abbildung 4 dargestellten Beispiel gilt $K_{start} = K_{vr}$ und $K_{ziel} = K_{rl}$. Es bleibt noch anzumerken, dass das Geradensegment aus der Bahnplanung wegfällt, wenn sich die äußeren Kreise von K_{start} und K_{ziel} überschneiden.

2.4.2 Continuous Curvature Turns

Wie in dem vorigen Abschnitt erwähnt wurde, spielen die einem CCT zugehörigen Kreispaare eine wichtige Rolle. Bei der Suche nach einer Bahn muss darauf geachtet werden, dass die Verbindungsgerade zwischen den Konfigurationen q_{sg} und q_{zg} glatt zwischen diese beiden Konfigurationen passt. Das bedeutet, dass die Fahrzeugausrichtung θ in q_{sg} und q_{zg} gleich sein muss. Diese Anforderung impliziert eine genaue Planung der auf den Kreisauschnitten der $CCTs$ zu fahrenden Strecken. Zur Veranschaulichung werden daher im Folgenden die geometrischen Eigenschaften eines CCT herausgearbeitet. Als Beispiel dient in der Abbildung 5 das Kreispaar K_{vl} innerhalb der Startsituation. Die beschreibenden Größen von K_{vr}, K_{rl} und K_{rr} können für die Start- und Zielsituation entsprechend der folgenden Untersuchung hergeleitet werden.

Die Konfigurationen q_{ss} und q_{sg} liegen auf dem äußeren, während q_{si} und q_{sj} auf dem inneren Kreis von K_{vl} liegen. Um den Mittelpunkt Ω dieser beiden Kreise zu bestimmen, muss zuerst die Konfiguration q_{si} berechnet werden. Sie wird erreicht, indem der Roboter bei konstanter positiver Geschwindigkeit $v1$ und konstantem $v2$ seinen maximalen

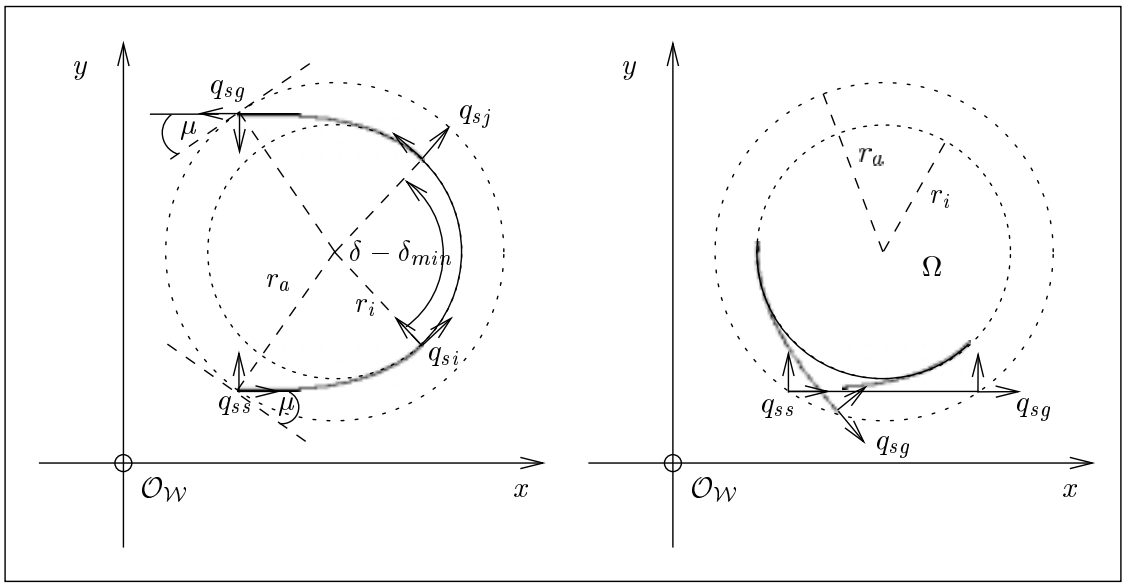


Abbildung 5: $CCTs$ mit unterschiedlichen Auslenkungen im Kreispaar K_{vl}

negativen Lenkwinkel $-\phi_{max}$ einstellt. Mit Hilfe der Gleichungen 12 und der Krümmung $k_{max} = 2\phi_{max}/\sqrt{2\phi_{max}a_K^2}$ des inneren Kreises ergibt sich:

$$\Omega = \begin{bmatrix} x_{si} - \frac{sin(\theta_{ss}+\phi_{max})}{k_{max}} \\ y_{si} + \frac{cos(\theta_{ss}+\phi_{max})}{k_{max}} \end{bmatrix} \tag{14}$$

Der Radius des äußeren Kreises ist dann durch $r_a = \sqrt{(x_\Omega - x_{ss})^2 + (y_\Omega - y_{ss})^2}$, der des inneren Kreises durch $r_i = \sqrt{(x_\Omega - x_{si})^2 + (y_\Omega - y_{si})^2}$ gegeben. Die charakterisierende Größe eines CCT ist die Auslenkung. Dieser mit δ bezeichnete Winkel beschreibt den Ausrichtungsunterschied des Roboters zwischen den Konfigurationen q_{ss} und q_{sg}. Demnach gilt $\delta = \theta_{ss} - \theta_{sg}$. Der Winkel δ wird auf das Intervall $[0...2\pi[$ beschränkt, um eine eindeutige Beschreibung der Auslenkung zu gewinnen. Die Überlegung dass δ von der Fahrtrichtung und der Bogenlänge des Kreisausschnittes zwischen q_{si} und q_{sj} abhängig ist, führt zu der Definition einer minimalen Auslenkung $\delta_{min} = k_{max}^2/\dot{\phi}/cos^2\phi$. Diese kennzeichnet die Fahrt des Roboters, bei der das Kreissegment entfällt. Dabei fallen die Konfigurationen q_{si} und q_{sj} zusammen, und der Roboter beschreibt bei einer Vorwärtsfahrt mit konstantem $v1$ von q_{si} bzw. q_{sj} nach q_{sg} einen Klothoidenausschnitt mit abnehmender Krümmung, was einem Zurückstellen des Lenkwinkels auf den Wert 0 mit konstantem $v2$ entspricht.

Die Eingangsbemerkung dieses Abschnittes, dass auf der Suche nach einer Bahn zwei Konfigurationen q_{sg} und q_{zg} auf den Start- und Zielkreisen mit gleicher Fahrzeugausrichtung zu suchen sind, soll nun wieder aufgegriffen werden. Ein CCT kann folgende vier Formen annehmen, um eine zur Verbindungsgeraden passende Auslenkung und somit passende Fahrzeugausrichtung in q_{sg} bzw. q_{zg} zu erzielen:

- $\delta = 0$: Dieser Fall ist im rechten Beispiel der Abbildung 5 dargestellt. Der CCT besteht aus einem Geradensegment von q_{ss} nach q_{sg} mit der Länge $l = 2r_a sin\mu$.
- $0 < \delta < \delta_{min}$: In diesem Fall besteht der CCT aus:
  - einer Klothoiden mit der Länge $l = \frac{a_K^2}{r_i}$, deren Krümmung von 0 auf k_{max} ansteigt,
  - einem rückwärts gefahrenen Kreisausschnitt mit der Krümmung k_{max} und der Bogenlänge $l = \delta_{min} - \delta$ sowie
  - einer Klothoiden mit der Länge $l = \frac{a_K^2}{r_i}$, deren Krümmung von k_{max} auf 0 fällt.
- $\delta_{min} \leq \delta < \delta_{min} + \pi$: Dieser Fall ist im linken Beispiel der Abbildung 5 dargestellt. Der CCT besteht aus:
  - einer Klothoiden mit der Länge $l = \frac{a_K^2}{r_i}$, deren Krümmung von 0 auf k_{max} ansteigt,
  - einem vorwärts gefahrenen Kreisausschnitt mit der Krümmung k_{max} und der Bogenlänge $l = (\delta - \delta_{min})r_i$ sowie
  - einer Klothoiden mit der Länge $l = \frac{a_K^2}{r_i}$, deren Krümmung von k_{max} auf 0 fällt.
- $\delta_{min} + \pi \leq \delta < 2\pi$: Dieser Fall ist wie der Fall $\delta = 0$ im Beispiel der Abbildung 5 dargestellt. Der CCT besteht aus:
  - einer Klothoiden mit der Länge $l = \frac{a_K^2}{r_i}$, deren Krümmung von 0 auf k_{max} ansteigt,
  - einem rückwärts gefahrenen Kreisausschnitt mit der Krümmung k_{max} und der Bogenlänge $l = (2\pi - \delta + \delta_{min})r_i$ sowie
  - einer Klothoiden mit der Länge $l = \frac{a_K^2}{r_i}$, deren Krümmung von k_{max} auf 0 fällt.

Die Winkel zwischen der x-Achse von $\mathcal{F}_\mathcal{R}$ in den Konfigurationen q_{ss} bzw. q_{sg} und den Tangenten, die in diesen Konfigurationen den äußeren Kreis berühren, sind vom Betrag her gleich. Sie werden mit μ bezeichnet.

2.4.3 Bahnklassen und deren Existenzbedingungen

In dem Artikel [10] wurde erstmals ein Bahnplanungsverfahren vorgestellt, das die nichtholonomen Beschränkungen eines Automobils und dessen Fähigkeit auf einer Bahn aus der Vorwärtsfahrt in die Rückwärtsfahrt zu wechseln berücksichtigte. Wie bei dem in Abschnitt 2.3 vorgestellten Verfahren, setzt sich dort eine Bahn aus Kreissegmenten und Geraden zusammen. Obwohl eine so konstruierte Bahn den Nachteil eines Zwangsstops

an den Segmentübergängen aufweist, bewiesen die Autoren eine weitere, vielmehr positive Eigenschaft solcher Bahnen. Sie zeigten, dass die kürzeste für einen Roboter mit autoähnlichen Fahreigenschaften realisierbare Bahn zwischen zwei Konfigurationen eine der neun folgenden Formen annimmt.

$$\begin{aligned} &C|C|C,\ CC|C,\ CSC,\ CC_u|C_uC,\ C|C_uC_u|C, \\ &C|C_{\pi/2}SC,\ C|C_{\pi/2}SC_{\pi/2}|C,\ C|CC,\ CSC_{\pi/2}|C \end{aligned} \tag{15}$$

Dabei steht ein C für einen Kreisausschnitt mit dem Krümmungsradius k_{max}, ein S für ein Geradensegment und ein | für einen Fahrtrichtungswechsel zwischen vor- und nachstehendem Segment. Der in einigen Formen untenstehende Index gibt die Bahnlänge des mit ihm indizierten Segmentes an. Bahnsegmente mit einer Länge $l \neq \frac{\pi}{2}$ können in den obenstehenden Formen wegfallen.

Werden die Kreisausschnitte durch die in [14] eingeführten $CCTs$ ersetzt, ergeben sich folgende drei Bausteine aus denen sich die in der Form 15 gegebenen Bahnen zusammensetzen lassen: CSC, CC und $C|C$. Ein C bezeichnet hier nicht mehr einen Kreisausschnitt, sondern einen entsprechenden CCT. Die im Rahmen dieser Arbeit auf *Rolland* implementierte Bahnplanung verwendet ausschließlich Bahnen der Form CSC und CC.

Die Existenzbedingungen dieser Bahnen sollen im Folgenden beschrieben werden. Diese formale Betrachtung ist sinnvoll, da beispielsweise eine CSC-Bahn nicht existiert, wenn sich die entsprechenden Start- und Zielkreispaare überschneiden. Eine wichtige Rolle spielen außerdem die Bewegungsrichtungen auf den Start- und Zielkreisen. Eine *im Uhrzeigersinn* gerichtete Bewegung wird dabei durch die Kreispaare K_{vr} und K_{rr} beschrieben, während der Roboter auf den Kreispaaren K_{vl} und K_{rl} *gegen den Uhrzeigersinn* fährt.

CSC-Bahnen Während bei der KGK-Bahnplanung die Konfigurationen auf den Start- und Zielkreisen durch Tangenten zwischen den Kreisen verbunden werden, liegen bei der CCT-Bahnplanung zwischen den äußeren Kreisen der Start- und Zielkreispaare sogenannte μ-Tangenten. Diese Bezeichnung resultiert aus deren Eigenschaft, dass zwischen ihnen und den Tangenten am äußeren Kreis in q_{sg} bzw. q_{zg} der Winkel μ liegt. Eine wie in Abblidung 6 dargestellte CSC-Bahn besteht so aus zwei $CCTs$ und einer die Konfigurationen q_{sg} und q_{zg} verbindenden μ-Tangente, die parallel zur Verbindungsgeraden zwischen Ω_{start} und Ω_{ziel} liegt. Der Roboter besitzt in der Start- und Zielsituation dieselbe Bewegungsrichtung. Die Existenzbedingung einer solchen Bahn ist durch folgende Gleichung gegeben:

$$\mid \Omega_1\vec{\Omega}_2 \mid \geq 2r_a \sin\mu \tag{16}$$

Ohne zuvor die Konfigurationen q_{sg} und q_{zg} berechnet zu haben, ergibt sich die Länge der Verbindungsgeraden als:

$$l(q_{sg}, q_{zg}) = \mid \Omega_1\vec{\Omega}_2 \mid -2r_a sin\mu \tag{17}$$

Sind die Bewegungsrichtungen des Roboters in der Start- und Zielsituation unterschiedlich, so nimmt eine CSC-Bahn die in Abbildung 7 dargestellte Form an. Die μ-Tangenten

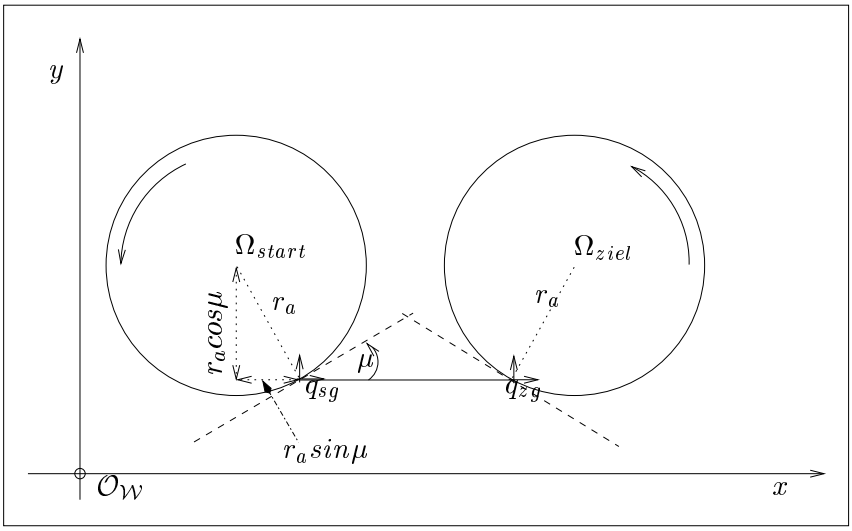


Abbildung 6: CSC-Bahn mit gleichgerichteter Bewegung in Start und Ziel

liegen in diesem Fall diagonal zwischen den ausgewählten Kreispaaren und kreuzen die Verbindungsgerade zwischen Ω_1 und Ω_2. Wie auch bei den CSC-Bahnen mit gleichgerichteten Bewegungsrichtungen, passt die μ-Tangente glatt zwischen die Konfigurationen q_{sg} und q_{zg}. Es sei daran erinnert, dass diese Aussage die gleiche Ausrichtung der x-Achsen von $\mathcal{F}_\mathcal{R}$ in q_{sg} bzw. q_{zg} und des Richtungsvektors der μ-Tangente beschreibt. Die Existenzbedingung einer solchen Bahn ist durch folgende Gleichung gegeben:

$$| \Omega_1\vec{\Omega}_2 | \geq 2r_a \tag{18}$$

Die Länge der zwischen q_{sg} und q_{zg} liegenden Verbindungsgeraden ist in [14] durch Gleichung 19 angegeben. Die Konfigurationen q_{sg} und q_{zg} müssen zur Längenberechnung nicht bekannt sein:

$$l(q_{sg}, q_{zg}) = \frac{| \Omega_1\vec{\Omega}_2 |^2 - 4r_a^2}{\sqrt{| \Omega_1\vec{\Omega}_2 |^2 - 4r_a^2 cos^2\mu} + 2r_a sin\mu} \tag{19}$$

CC-Bahnen Die zweite in der CCT-Bahnplanung verwendete Klasse von Bahnen setzt sich lediglich aus den $CCTs$ in der Start- und Zielsituation zusammen. Die sonst die Konfigurationen q_{sg} und q_{zg} verbindende Gerade entfällt, da sich die äußeren Kreise der Kreispaare K_{start} und K_{ziel} berühren. Die Konfiguration an dem Berührungspunkt wird mit q_g bezeichnet. Eine CC-Bahn existiert nur, wenn die Bewegungsrichtungen auf den Start- und Zielkreispaaren unterschiedlich sind, da sonst nicht die gewünschte Zielausrichtung θ_{zs} erreicht werden kann. Die Existenzbedingung eines gemeinsamen Berührungspunktes ist

$$| \Omega_1\vec{\Omega}_2 | = 2r_a. \tag{20}$$

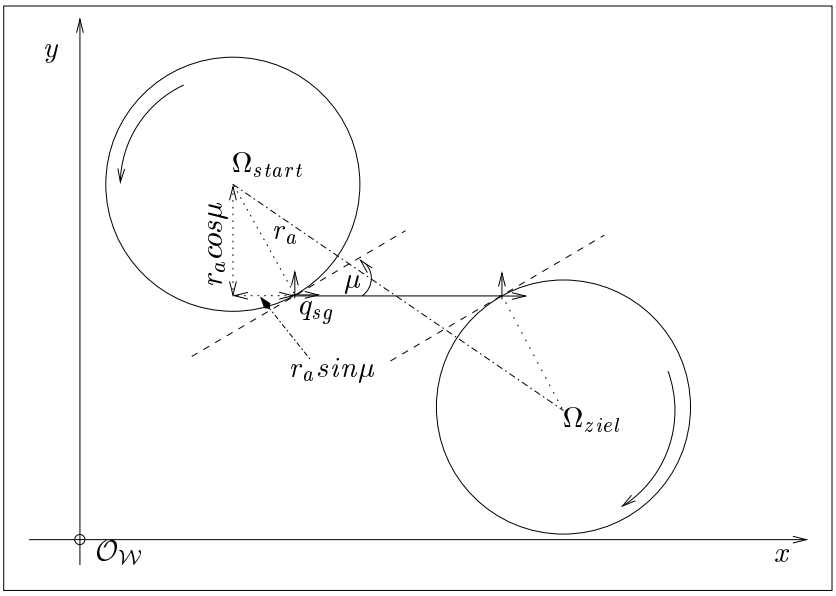


Abbildung 7: CSC-Bahn mit entgegengesetzter Bewegung in Start und Ziel

2.5 Trajektoriengenerierung

Bei der Trajektoriengenerierung ist eine der Gleichung 2 genügende Beschreibung der zu fahrenden Bahn zwischen q_{start} und q_{ziel} zu finden. Vor diesem Schritt liegt die Bahn in Form einer Liste von äquidistanten Punkten in der Ebene vor, mit deren Hilfe die Bewegung des Roboters respektive seines Referenzpunktes beschrieben wird. Die metrische Diskretisierung wird derart interpretiert, dass der Roboter die Bahn mit konstanter Geschwindigkeit $v1$ abfährt und in einem vorgegebenen Zeittakt Momentaufnahmen seiner Position gemacht werden. In diesem Licht kann die Aufgabe der Trajektoriengenerierung neu formuliert werden. Zuerst wird der metrisch diskretisierten Bahn ein Geschwindigkeitsprofil zugeordnet, um eine angemessene Geschwindigkeit des Roboters in jedem Punkt der Bahn zu beschreiben. In einem zweiten Schritt wird aus diesen Informationen eine neue Beschreibung der Bahn gewonnen. Diese nimmt wiederum die Form einer Vektorenliste an. Im Gegensatz zur bisherigen Bahnbeschreibung sind deren Elemente jedoch zeitlich diskretisiert. Das bedeutet, dass zwei in der Liste aufeinanderfolgende Punkte den metrischen Abstand zueinander haben, den der Roboter im vorgegebenen Zeittakt mit der zuvor berechneten Sollgeschwindigkeit des ersten Punktes von diesem aus fahren kann.

Es bleibt noch zu klären was unter einer angemessenen Geschwindigkeit des Roboters in einer bestimmten Position zu verstehen ist. Bei seinem Weg aus der Startkonfiguration in die Zielkonfiguration beschreibt der Roboter eine aus verschiedenen Segmenten zusammengesetzte Kurve mit einer Krümmung, die zwischen 0 und k_{max} variiert. Unter Berücksichtigung der Regel, dass der Betrag der Geschwindigkeit auf einem Geradensegment ($k = 0$) seinen maximalen und auf einem Kreissegment ($k = k_{max}$) seinen minima-

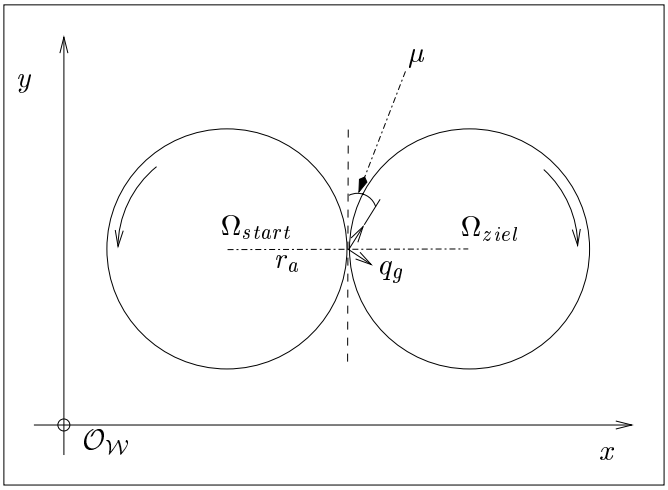


Abbildung 8: CC-Bahn mit entgegengesetzter Bewegung in Start und Ziel

len Wert annehmen soll, lässt sich die Sollgeschwindigkeit als umgekehrt proportional zur Kurvenkrümmung beschreiben. Letztlich ist noch auf die zur Planungszeit vorgegebenen Geschwindigkeiten $v1_{start}$ und $v1_{ziel}$ sowie auf ein Unterschreiten der minimalen Geschwindigkeit bei Richtungswechseln zu achten.

2.5.1 Ableiten des Geschwindigkeitsprofils

Um ein Geschwindigkeitsprofil für eine metrisch diskretisierte Bahn zu gewinnen sind im wesentlichen zwei Schritte notwendig. Zuerst werden die Sollgeschwindigkeiten der Anfangs- und Endvektoren eines jeden Segmentes bestimmt. Diese Vektoren werden im Folgenden Schlüsselvektoren genannt. Im zweiten Schritt wird den zwischen den Schlüsselvektoren liegenden Punkten eine Geschwindigkeit zugeordnet. Dies geschieht durch lineare Interpolation zwischen den Geschwindigkeiten der angrenzenden Schlüsselvektoren.

Geschwindigkeiten an den Segmentgrenzen Für die folgende Betrachtung existiere für die Bahn eine geordnete Liste ihrer Schlüsselvektoren $\mathcal{SV} = \{sv_1, ..., sv_n\}$. Außerdem sei für jeden Schlüsselvektor die Kurvenkrümmung ($sv_i.k$) und die Fahrtrichtung des Roboters an dieser Position ($sv_i.dir \in \{v, r\}$) bekannt. Die Geschwindigkeit $v1$ des Roboters an der Position des Schlüsselvektors mit dem Index i kann dann der Tabelle 1 entnommen werden. In der Startkonfiguration ist sie mit $v1_{start}$ und in der Zielkonfiguration mit $v1_{ziel}$ vorgegeben.

Interpolation zwischen den Schlüsselvektoren Nachdem die Sollgeschwindigkeiten an den Segmentgrenzen der Bahn gesetzt wurden, gilt es den verbleibenden Vektoren einen Wert für $v1$ zuzuordnen. Dies geschieht durch lineare Interpolation zwischen $sv_i.v1$ und $sv_{i+1}.v1$. Dabei sind zwei Fälle zu unterscheiden:

Der erste Fall ist dadurch gekennzeichnet, dass am Segmentanfang und Segmentende ein Fahrtrichtungswechsel geplant ist. Somit ist die dort vorgeschriebene Geschwindigkeit $v1$ gleich null. Unter dieser Bedingung wird dem in der Mitte zwischen sv_i und sv_{i+1} liegenden Vektor vec_j die Geschwindigkeit $vec_j.v1 = (1 - \frac{vec_j.k}{k_{max}})v_{max}$ zugeordnet. Danach werden die Geschwindigkeiten in den restlichen Vektoren durch lineare Interpolation zwischen $sv_i.v1$ und $vec_j.v1$ bzw. $vec_j.v1$ und $sv_{i+1}.v1$ ermittelt. Das so für dieses Segment berechnete Geschwindigkeitsprofil beschreibt den Rollstuhl, wie er aus der Geschwindigkeit $v1 = 0$ auf eine von der in vec_j existierenden Kurvenkrümmung abhängigen Geschwindigkeit beschleunigt und bis zum Ende des Segmentes wieder auf $v1 = 0$ abbremst.

Im zweiten Fall gilt für zumindest eine der Geschwindigkeiten $sv_i.v1$ oder $sv_{i+1}.v1$, dass sie ungleich null ist. Jedem der in diesem Segment liegenden Vektoren kann dann mittels linearer Interpolation zwischen den zuvor genannten Randgeschwindigkeiten ein Wert für $v1$ zugewiesen werden.

Es sei noch bemerkt, dass die resultierende lineare Geschwindigkeitsentwicklung auf der Bahn einem vereinfachten Beschleunigungsmodell des Roboters entspricht. Es wird also hier davon ausgegangen, dass die Beschleunigung bei jedem Geschwindigkeitswert $v1$ ein und denselben Wert annimmt.

2.5.2 Von metrischer zu zeitlicher Diskretisierung

Die eine Bahn beschreibende Funktion bildet die von der Startposition aus gefahrene Strecke auf die dann erreichte Position ab. In der Implementierung wird dazu der reelle Parameter *Fahrtstrecke* diskretisiert. So ergibt sich eine Positionenliste, von der jeweils zwei aufeinanderfolgende Punkte denselben metrischen Abstand zueinander haben. Im Gegensatz dazu bildet eine Trajektorienfunktion die (seitdem sich der Roboter in der Startposition befunden hat) verstrichene Zeit auf die bis dahin erreichte Position ab. Hier wird für die Implementierung der reelle Parameter *Zeit* diskretisiert. Um nun aus einer Positionenliste die der Gleichung 1 genügt und einem entsprechenden Geschwindigkeitsprofil (Abschnitt 2.5.1) eine zeitlich diskretisierte Positionenliste genüge der Gleichung 2 zu gewinnen, wird folgendermaßen verfahren:

Beginnend mit dem ersten Vektor der Bahn als aktueller Position wird diese in die Vektorenliste der Trajektorie eingefügt. Die dem Geschwindigkeitsprofil entnommene korrespondierende Geschwindigkeit $v1$ wird nun mit dem vorgegebenen Zeittakt Δ_t multipliziert. Von der aktuellen Position aus wird dann die resultierende Strecke auf der Bahn vorangeschritten. Die so erreichte Position wird schließlich als aktuelle Position der Trajektorie hinzugefügt. Die dort gültige Geschwindigkeit $v1$[4] ergibt sich aus der Mittelung zwischen den Geschwindigkeiten der die aktuelle Position umgebenden Bahnvektoren. Dieser Ablauf wird solange wiederholt, bis die Zielposition der Bahn erreicht ist. Um einen kontinuierlichen Fortschritt zu gewährleisten, darf dabei der Betrag der Geschwindigkeit $v1$ einen minimalen Wert nicht unterschreiten.

[4] Das Geschwindigkeitsprofil einer Trajektorie entspricht der zeitlich geordneten Liste der Geschwindigkeiten einer jeden Trajektorienposition.

sv_i ist erster Vektor eines Segmentes	sv_i ist letzter Vektor eines Segmentes	sv_i ist erster Vektor der Bahn	sv_i ist letzter Vektor der Bahn	$sv_{i-1}.dir$	$sv_i.dir$	$sv_{i+1}.dir$	$sv_i.k$	$v1$
⊗		⊗						$v1_{start}$
	⊗		⊗					$v1_{ziel}$
⊗				r	v			0
⊗				v	r			0
⊗				v	v		0	$v1_{max}$
⊗				r	r		0	$-v1_{max}$
⊗				v	v		$\neq 0$	$(1-\frac{sv_i.k}{k_{max}})v1_{max}$
⊗				r	r		$\neq 0$	$-(1-\frac{sv_i.k}{k_{max}})v1_{max}$
	⊗				r	v		0
	⊗				v	r		0
	⊗				v	v	0	$v1_{max}$
	⊗				r	r	0	$-v1_{max}$
	⊗				v	v	$\neq 0$	$(1-\frac{sv_i.k}{k_{max}})v1_{max}$
	⊗				r	r	$\neq 0$	$-(1-\frac{sv_i.k}{k_{max}})v1_{max}$

Tabelle 1: Geschwindigkeiten des Roboters an den Segmentgrenzen

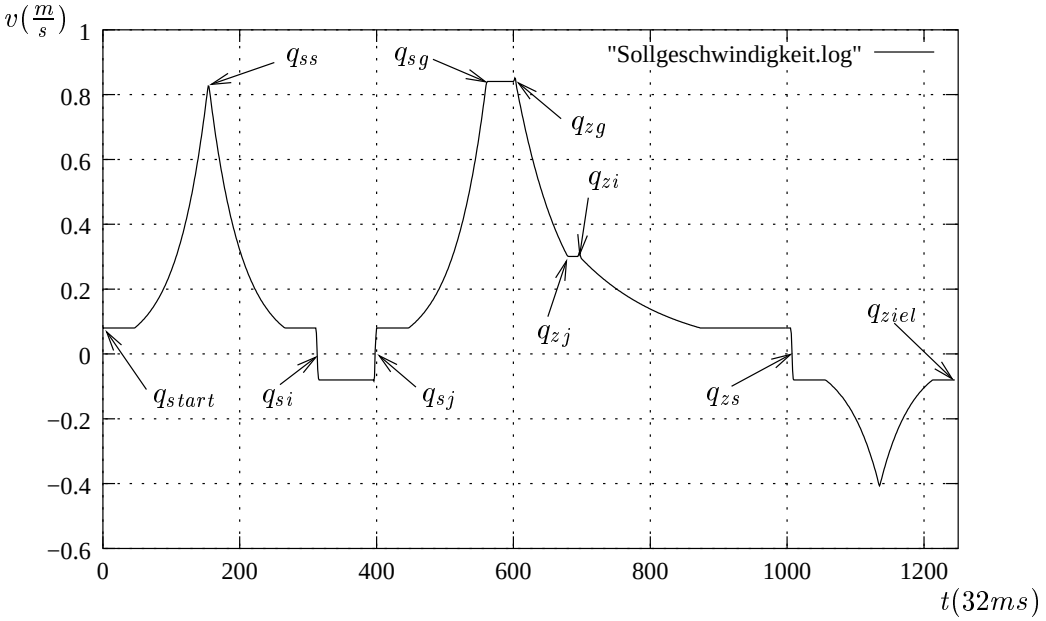


Abbildung 9: Geschwindigkeitsprofil der Trajektorie, die aus der in Abbildung 4 dargestellten Bahn gewonnen wurde

Beispielhaft ist in der Abbildung 9 das Geschwindigkeitsprofil der Trajektorie zu sehen, die aus der in Abbildung 4 dargestellten Bahn gewonnen wurde. Dabei ist die Geschwindigkeit $v1 = 0$ in q_{start} und q_{ziel} vorgegeben. Die der Abbildung zu entnehmenden Werte von $v1 \neq 0$ in q_{start} und q_{ziel}, sowie die Sprünge in q_{si}, q_{sj} und q_{zs} haben ihre Ursache in der für den oben beschriebenen Algorithmus gestellten Forderung nach einer minimalen Geschwindigkeit.

2.5.3 Vervollständigung der Trajektorienbeschreibung

Die bis zu diesem Punkt gewonnene Beschreibung einer Trajektorie zwischen den Konfigurationen q_{start} und q_{ziel} genügt insofern noch nicht der Gleichung 2, als dass die zeitliche Entwicklung der Konfigurationsparameter θ und ϕ sowie der Stellgröße $v2$ bisher fehlen. Um diese zu berechnen, wird die Forderung gestellt, dass sich die aus den Koordinatenpaaren $[x(t), y(t)]$, $t \in [0...T]$ zusammengesetzte Kurve in jedem Punkt dreimal stetig differenzieren lässt. Diese Kurve ist für den Roboter genau dann fahrbar, wenn sie sich aus dem kinematischen Modell von Rolland (Gleichung 3), gültigen Startwerten $x(0)$, $y(0)$, $\theta(0)$ und $\phi(0)$ und einer stetigen Entwicklung von $v1(t)$ und $v2(t)$ mit $t \in [0...T]$ herleiten lässt. Nach [8] lassen sich die Steuersignale $v1(t)$ und $v2(t)$ sowie die Konfigurationsparameter $\theta(t)$ und $\phi(t)$ allein aus der durch die Koordinatenpaare $[x(t), y(t)]$ gegebenen Kurve und deren ersten drei Ableitungen angeben:

$$\theta = ATAN2\left\{\frac{\dot{y}}{v1}, \frac{\dot{x}}{v1}\right\} \tag{21}$$

$$\phi = ARCTAN\frac{l[\ddot{y}\dot{x} - \ddot{x}\dot{y}]}{v1^3} \tag{22}$$

$$v1 = \pm\sqrt{\dot{x}^2 + \dot{y}^2} \tag{23}$$

$$v2 = l\ v1\frac{[\dddot{y}\ \dot{x} - \dddot{x}\ \dot{y}]v1^2 - 3[\ddot{y}\dot{x} - \ddot{x}\dot{y}][\dot{x}\ddot{x} + \dot{y}\ddot{y}]}{v1^6 + l^2[\ddot{y}\dot{x} - \ddot{x}\dot{y}]} \tag{24}$$

Zur Vereinfachung wird in den obenstehenden Formeln der Hinweis auf die Abhängigkeit eines jeden Parameters, sowie der Steuersignale von der Zeit $t \in [0...T]$ weggelassen. Die ersten drei Ableitungen der Kurve $[x, y]$ nach der Zeit sind durch $[\dot{x}, \dot{y}]$, $[\ddot{x}, \ddot{y}]$ und $[\dddot{x}, \dddot{y}]$ gegeben.

3 Trajektorienfolgeregelung

Um eine mobile Plattform wie *Rolland* aus einer Startkonfiguration in eine Zielkonfiguration zu überführen, wurde bis zu diesem Punkt eine Referenztrajektorie und eine Liste von korrespondierenden Steuerungssignalen generiert. Es bietet sich an, diese Informationen als einen virtuellen Roboter zu interpretieren, den *Rolland* während seiner realen Fahrt folgen muss, um die gewünschte Zielkonfiguration zu erreichen. Prinzipiell genügen dazu die zur Planungszeit berechneten Steuerungssignale. Ein solches Herangehen wird in der Literatur *feedforward Control* genannt. Diese Bezeichnung leitet sich aus der Tatsache ab, dass ausschließlich die zur Planungszeit berechneten Steuerungssignale dem zu regelnden System zugeführt werden.

In Abbildung 10 ist die *feedforward*-Regelung schematisch dargestellt. Zum Zeitpunkt t werden dem sich in der Konfiguration $q(t)$ befindenden Roboter die Steuersignale $v1(t)$ und $v2(t)$ zugeführt. Die Ausgangsgröße des Systems bildet die zum Zeitpunkt $t+1$ erreichte Konfiguration. In der Praxis ist diese Herangehensweise jedoch nicht ausreichend, da die Soll- und Istkonfigurationen während der Fahrt immer weiter voneinander abweichen. Gründe hierfür sind zum einen in dem idealisierten kinematischen Modell (Gleichung 3) des Roboters zu suchen, welches beispielsweise das Beschleunigungsverhalten von *Rolland* nicht berücksichtigt. Andererseits können unvorhersehbare äußere Störeinflüsse, wie eine von q_{start} abweichende initiale Konfiguration oder ein Rutschen der Räder auf glattem Untergrund den Roboter von seiner Bahn abbringen. Diese Aufzählung ließe sich beliebig erweitern. Festzuhalten bleibt nur, dass weder der Roboter inklusive seiner Fahreigenschaften, noch seine Arbeitsumgebung mathematisch exakt modelliert werden können. Aus diesen Gründen wird ein Regelungsverfahren verwendet, das *feedback Control* genannt wird.

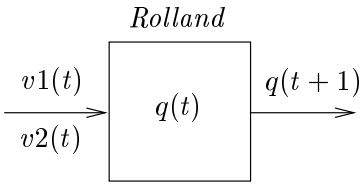


Abbildung 10: *Feedforward*-Regelung

Die Grundidee bei dem in der Abbildung 11 dargestellten Regelungsmodell ist es, während der Fahrt die Soll- und Istkonfigurationen miteinander zu vergleichen. Aus den resultierenden Fehlerwerten der einzelnen Konfigurationsparameter sowie der zur Planungszeit berechneten Referenz-Steuerungssignale werden dann die nachgeregelten Steuerungssignale $u1$ und $u2$ berechnet. Ausschlaggebende Faktoren für eine Übereinstimmung zwischen geplanter und gefahrener Trajektorie sind bei dieser Herangehensweise eine möglichst exakte Selbstlokalisation, sowie die Güte des in der Abbildung 11 als Blackbox dargestellten Regelungsalgorithmus.

3.1 Regelungstechnische Beschreibung des kinematischen Modells

Das durch die Gleichung 3 gegebene kinematische Modell von Rolland soll in diesem Abschnitt durch regelungstechnische Begriffe beschrieben werden. Das damit verbundene Ziel ist es, das Problem der Trajektorienfolgeregelung genau zu erfassen und die Grundlagen für die Beschreibung des Regelungsalgorithmus bereitzustellen. Die folgen-

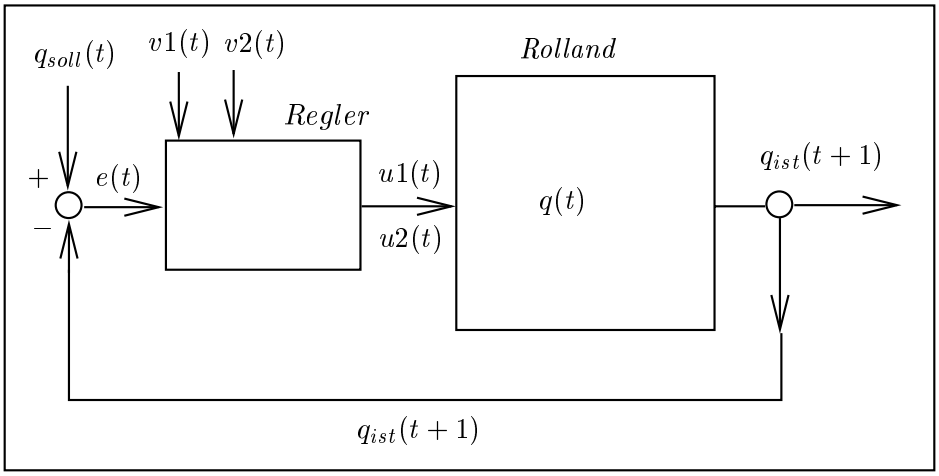


Abbildung 11: *Feedback*-Regelung

den Ausführungen basieren auf [8] und [15].

Um den Bewegungsprozess von *Rolland* zu beschreiben, ist das kinematische Modell des Roboters und seine Ausgangskonfiguration vorgegeben. Das kinematische Modell beschreibt den Geschwindigkeitsvektor $\dot{q}(t)$, also die erste Ableitung der aktuellen Konfiguration $q(t)$ nach der Zeit in Abhängigkeit von den Steuersignalen $v1(t)$, $v2(t)$ und $q(t)$. Dieser Zusammenhang wird in der Mathematik als System von gewöhnlichen Differentialgleichungen erster Ordnung erfasst, und kann für das Beispiel *Rolland* wie folgt beschrieben werden:

$$\dot{q}(t) = \mathcal{F}(q(t), v(t)),\ q(t+1) = \mathcal{G}(q(t)),\ q(0) = q_{start}\ \ \forall t \in [0...T] \tag{25}$$

Die Gleichung $q(0) = q_{start}$ wird Anfangsbedingung genannt. Die Aufgabe der Trajektorienfolgeregelung oder Nachführung ist es nun, für eine gegebene Trajektorie $q_{soll}(t)$ ein Regelgesetz $v(t) = \psi(q(t))$ zu finden, sodass der Fehlervektor $e(t) = q_{soll}(t) - q(t)$ für $t \to T$ gegen $\vec{0}$ strebt. Da es sich bei dem kinematischen Modell um ein System von Differentialgleichungen handelt ist die Funktion $\mathcal{G}$ nicht bekannt. Die Schwierigkeit liegt also darin, dass kein direkter Zusammenhang zwischen der den Ausgangsvektor $q(t+1)$ beschreibenden Funktion $\mathcal{G}$ und dem die Steuerungssignale beschreibenden Eingangsvektor v besteht. Dieser kann nur indirekt über den der Funktion $\mathcal{F}$ eigenen Parameter $q(t)$ hergestellt werden. Bei genauerer Betrachtung des Funktionenvektors

$$\mathcal{F}(q(t), v(t)) = \begin{bmatrix} f_1(q(t), v(t)) \\ f_2(q(t), v(t)) \\ f_3(q(t), v(t)) \\ f_4(q(t), v(t)) \end{bmatrix} = \begin{bmatrix} cos\theta(t)v1(t) \\ sin\theta(t)v1(t) \\ \frac{tan\phi(t)}{l} \\ v2(t) \end{bmatrix} \tag{26}$$

zeigt sich außerdem, dass Koppelungen zwischen dem Eingangsvektor $v(t)$ und dem zu regelnden Vektor $q(t)$ bestehen. Das bedeutet, dass mehrere Komponenten des Ausgangsvektors von derselben Komponente des Eingangsvektors abhängig sind.

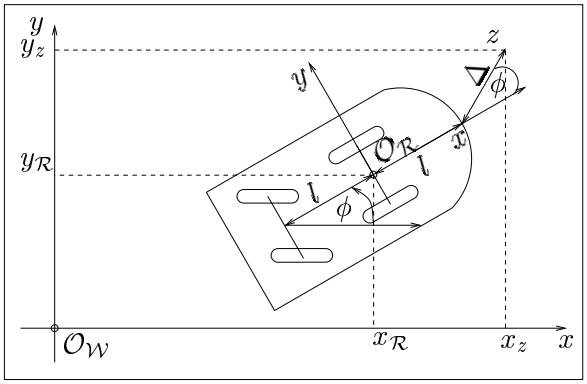


Abbildung 12: Der Vektor z als transformierter Referenzpunkt von *Rolland*

Die in dem folgenden Abschnitt beschriebene Lösung zur Entkoppelung und Regelung des kinematischen Systems von *Rolland* wird in [8] und [15] *Eingangs-Ausgangs-Linearisierung* genannt, und beruht auf der Idee, einen linearen Zusammenhang zwischen dem Eingangsvektor $v(t)$ und dem Ausgangsvektor $q(t+1)$ zu finden. Dazu wird der Ausgangsvektor $q(t+1)$ geeignet transformiert und solange mittels $\mathcal{F}$ nach der Zeit differenziert, bis jede Komponente des neuen Ausgangsvektors nur noch von einer Komponente des Eingangsvektors abhängig ist.

3.2 Eingangs-Ausgangs-Linearisierung mit statischer Rückführung

Das in [8] beschriebene Linearisierungsverfahren beginnt mit der Wahl eines neuen und $q(t+1) = [x, y, \theta, \phi]$ ersetzenden Ausgangsvektors

$$z = \begin{bmatrix} x + l\ cos\theta + \Delta cos(\theta + \phi) \\ y + l\ sin\theta + \Delta sin(\theta + \phi) \end{bmatrix}, \tag{27}$$

wobei $\Delta \neq 0$ gilt. Der Vektor z kann wie in Abbildung 12 dargestellt, als neuer Referenzpunkt des Roboters interpretiert werden. Wird z nun nach der Zeit differenziert, ergibt sich der Vektor $\dot{z}$ als

$$\dot{z} = \begin{bmatrix} cos\theta - tan\phi(sin\theta + \Delta sin(\theta + \phi)/l) & -\Delta sin(\theta + \phi) \\ sin\theta + tan\phi(cos\theta + \Delta cos(\theta + \phi)/l) & \Delta cos(\theta + \phi) \end{bmatrix} v = \mathcal{A}(\theta, \phi)v. \tag{28}$$

Da $det\mathcal{A}(\theta, \phi) = \Delta / cos\phi \neq 0$ gilt, kann $r = \dot{z}$ als zusätzlicher Eingangsvektor betrachtet werden und die Gleichung 28 nach v wie folgt aufgelöst werden:

$$v = \mathcal{A}^{-1}(\theta, \phi)r \tag{29}$$

Für die transformierten Konfigurationen $\hat{q} = [z1, z2, \theta, \phi]$ des Roboters kann mit Hilfe der durch Gleichung 29 erfüllten Forderung nach einem Regelgesetz der Form $v = \psi(\hat{q})$

nun folgendes rückgekoppeltes Gleichungssystem aufgestellt werden:

$$\dot{\hat{q}} = \hat{\mathcal{F}}(\hat{q}, \psi(\hat{q})) = \begin{bmatrix} \hat{f}_1(\hat{q}, \psi(\hat{q})) \\ \hat{f}_2(\hat{q}, \psi(\hat{q})) \\ \hat{f}_3(\hat{q}, \psi(\hat{q})) \\ \hat{f}_4(\hat{q}, \psi(\hat{q})) \end{bmatrix} = \begin{bmatrix} r1 \\ r2 \\ sin\phi[cos(\theta+\phi)r1 + sin(\theta+\phi)r2]/l \\ \{-[cos(\theta+\phi)sin\phi/l + sin(\theta+\phi)/\Delta]r1 \\ -[sin(\theta+\phi)sin\phi/l - cos(\theta+\phi)/\Delta]r2\} \end{bmatrix} \quad (30)$$

Im Gegensatz zu dem in Gleichung 26 gegebenen Differentialgleichungssystem ist das durch die Gleichung 30 gegebene System zumindest für $\dot{z}1$ und $\dot{z}2$ Eingangs-Ausgangs-linearisiert wenn für die den Regelungsfehler $e(t)$ beschreibenden Rückführungsterme $r1$ und $r2$ folgende Ausdrücke verwendet werden:

$$ri = \dot{z}_{soll}i + ki(z_{soll}i - z_{ist}i), \quad ki > 0, \quad i = 1, 2 \quad (31)$$

Es wird bei diesem Verfahren von *statischer Rückführung* gesprochen, da der Eingangsvektor $v(t)$ zu jedem Zeitpunkt t nur von dem transformierten Zustandsvektor $\hat{q}(t)$ zum selben Zeitpunkt abhängig ist. Im Gegensatz dazu wird von *dynamischer Rückführung* gesprochen, wenn $v(t)$ durch $q(t)$ aufgrund von Differentialgleichungen festgelegt wird.

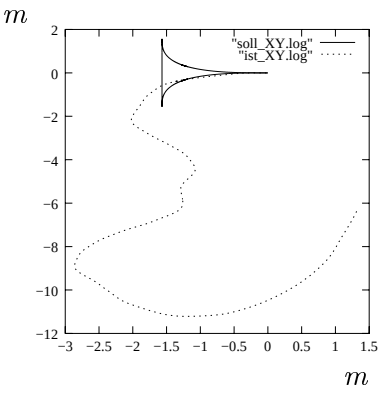


Abbildung 13: Durch Rückwärtsfahrt verursachter Regelungsfehler

Abschließend sei bemerkt, dass sich bei der Implementierung des hier vorgestellten Regelungsverfahrens die Rückwärtsfahrt des Roboters als unmöglich erwiesen hat. In diesem Zusammenhang sind in der Abbildung 13 die geplante und die real gefahrene Trajektorie der Simulation *SimRobot* zu sehen. Es ist zu erkennen, dass die Fahrt des Roboters bei anfänglicher Rückwärtsfahrt falsch nachgeregelt wird. Aus diesem Grund wird bei einem rückwärts zu fahrenden Bahnsegment vor der Berechnung der Referenzpunkte z_{soll} und z_{ist} die Ausrichtung θ des Roboters um den Winkel π gedreht. Mit Hilfe dieser Transformation, die auch einen Vorzeichenwechsel des Lenkwinkels ϕ einschließt, wird die Rückwärtsfahrt als Vorwärtsfahrt interpretiert. Es darf in diesem Fall jedoch nicht vergessen werden, das Vorzeichen der resultierenden Steuerungssignale $v1$ und $v2$ umzudrehen. Die Abbildung 25(b) zeigt das durch die beschriebenen Transformationen erzielte Ergebnis im Vergleich zu der in Abblidung 13 falsch nachgeregelten Trajektorie.

4 Implementierung

Nachdem in den Abschnitten 2 und 3 die *CCT*-Bahnplanung sowie die Trajektorienfolgeregelung mittels Eingangs-Ausgangs-Linearisierung beschrieben wurden, soll im Folgenden auf die im Rahmen dieser Arbeit implementierte Software eingegangen werden. Diese realisiert die bisher beschriebenen Verfahren innerhalb des 3D-Robotiksimulators *SimRobot* [13], sowie auf der mobilen Plattform *Rolland*, und wird in ihrer Gesamtheit von nun an A2B-Modul[5] genannt. Außerdem wird auf das Visualisierungstool *Rolland-Bahnplaner* eingegangen. Dieses Programm ermöglicht es, die nach Abschnitt 2.4 ermittelten Bahnen zwischen zwei beliebigen Konfigurationen darzustellen. Das Ziel der folgenden Ausführungen ist es zum einen, die gewählte Granulation der Datenstrukturen und Methoden in Verbindung mit den sich gestellten Aufgaben zu setzen. Zum anderen werden Schnittstellenbeschreibungen gegeben, um eine eventuelle Erweiterung oder Einbettung in weitere Programmpakete zu erleichtern.

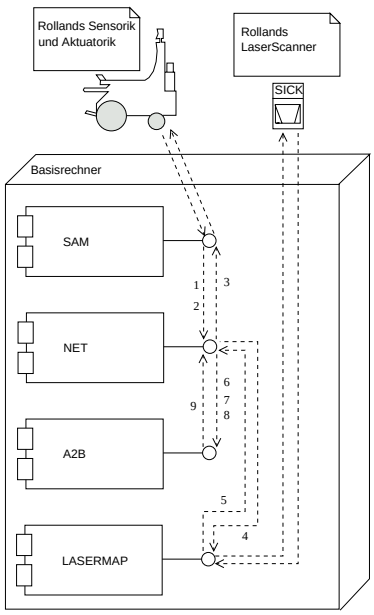


Abbildung 14: Systemarchitektur der auf Rolland eingesetzten Software

4.1 Systemumgebung des A2B-Moduls

Innerhalb der Simulation *SimRobot*, wie auch auf der realen Plattform *Rolland* arbeitet das für die Navigation zuständige A2B-Modul mit anderen Modulen zusammen, um eine klar getrennte Aufgabenteilung zu erreichen. Die in Abbildung 14 schematisch dargestellte Systemarchitektur der eingesetzten Software zeigt die vier Module SAM, NET, A2B und LASERMAP sowie zwei Piktogramme, welche die reale bzw. simulierte Sensorik und Aktuatorik von *Rolland* darstellen. Die zwischen diesen einzelnen Elementen liegenden Abhängigkeitspfeile stehen für einen Informationsfluss zwischen den Komponenten.

4.1.1 Sensorik-Aktuatorik-Modul

Das Modul SAM (Sensorik / Aktuatorik Modul) stellt in dieser Architektur eine sichere Schnittstelle zu der mobilen Plattform dar. Sicher bedeutet nach [5], dass alleine SAM für das Versenden von Steuerungskommandos an die Aktuatorik und die Bereitstellung der Rollstuhl internen Sensordaten für andere Module zuständig ist. Weiterhin ist SAM dafür verantwortlich, keine aktive Kollision des Fahrzeuges mit einem Hindernis zu verursachen. Um dies zu realisieren, erhält das Modul alle 32 ms umfangreiche Daten von der

[5] Diese Benennung beschreibt die Aufgabe des Moduls, *Rolland* aus einer Konfiguration A in eine Konfiguration B zu überführen.

Sensorik und kann mit dieser Hilfe entscheiden, ob eine von einer Anwendung wie dem Steuerungssmodul A2B geforderte Kombination aus Geschwindigkeit und Lenkwinkel innerhalb des nächsten Zeittaktes von 32 ms zu einer Kollision führen würde. In einem solchen Fall würde SAM die erhaltenen Steuerungsbefehle ignorieren und eine Vollbremsung durchführen. Die für das von SAM zu lösende Aufgabengebiet notwendigen Sensordaten setzen sich aus der aktuellen Geschwindigkeit und dem aktuellen Lenkwinkel der Plattform sowie den Entfernungsmesswerten der 32 Ultraschallsensoren zusammen. Außerdem erhält SAM hierfür die Daten des extern montierten Laserscanners über das Modul LASERMAP.

4.1.2 NET-Modul

Für die Übertragung der Informationen zwischen den einzelnen Modulen ist die in [5] dokumentierte Komponente NET zuständig. Diese stellt für die miteinander kommunizierenden Module einen gemeinsam nutzbaren Speicherbereich zur Verfügung. Um die verschiedenen Datentypen unterscheiden zu können, muss ein Daten produzierendes Modul diese erst unter einer eindeutigen ID bei dem Modul NET anmelden. Danach kann dieses, und nur dieses Modul, ein Datenpaket des Typs ID dem Modul NET übergeben. Auf demselben Weg muss auch ein Daten empfangendes Modul zunächst seine Absicht erklären, Datenpakete des Typs ID aus dem gemeinsam genutzten Datenbereich auszulesen, bevor es diese lesen kann. Daten eines Typs ID können im Gegensatz zu einem produzierenden Modul von mehreren Modulen empfangen werden. Es sei noch angemerkt, dass dieses Verfahren des Datenaustauschs ausschließlich die Kommunikation zwischen den einzelnen Softwarekomponenten regelt. So stellt SAM den anderen Modulen die aktuelle Geschwindigkeit und den aktuellen Lenkwinkel in geeigneter Form über das Modul NET zur Verfügung, erhält diese Werte aber durch ein Auslesen der Sensorikhardware. Ähnlich verhält es sich mit dem Modul LASERMAP, dessen Funktionalität im Folgenden beschrieben wird. Siehe dazu auch [11].

4.1.3 LASERMAP-Modul

Die zur Nachregelung einer vom Navigationsmodul generierten Trajektorie benötigte Selbstlokalisation könnte prinzipiell auf den von SAM bereitgestellten Odometriedaten beruhen. Diese werden von SAM durch ein Koppelnavigationsverfahren bestimmt [5]. Dazu wird die von einem internen Inkrementalsensor bestimmte Umdrehungsgeschwindigkeit der Antriebsräder sowie der von einem Potentiometer gelieferte, aktuell eingestellte Lenkwinkel innerhalb eines Zeittaktes zu einem Versatzvektor umgerechnet. Dieser wird regelmäßig von SAM zu der aktuellen Position und Orientierung des Rollstuhls hinzuaddiert. Da dieses Verfahren aufgrund von Ungenauigkeiten des Inkrementalsensors[6] und des Potentiometers[7] sowie einem möglichen Rutschen der Räder auf unterschiedlichen Bodenbelägen sehr ungenau ist, übernimmt LASERMAP die Korrek-

[6] Es wird nur einmal pro Sekunde ein gemittelter Geschwindigkeitswert von der Rollstuhlhardware zur Verfügung gestellt.

[7] Die Auflösung des Lenkwinkels beträgt $\frac{1}{2}°$.

Modul	Sender/Receiver	Beschreibung
SAM(1)	Sender(ID_ODOMETRY_POSITION, Position) sndPosition;	stellt die von SAM ermittelte Odometrieposition zur Verfügung
SAM(2)	Sender(ID_ACTUAL_MOTORICS, Motorics) sndActualMotorics;	liefert Ist-Geschwindigkeit und Ist-Lenkwinkel
SAM(3)	Receiver(ID_TARGET_MOTORICS, Motorics) recTargetMotorics;	empfängt Soll-Geschwindigkeit und Soll-Lenkwinkel
LASERMAP (4)	Receiver(ID_ODOMETRY_POSITION, Position) recPosition;	empfängt die aktuelle Odometrieposition
LASERMAP (5)	Sender(ID_POSITION_CORRECTION, Correction) sndCorrection;	liefert den Versatz der akt. zur letzten Konfiguration
A2B(6)	Receiver(ID_ACTUAL_MOTORICS, Motorics) recMotorics;	empfängt Ist-Geschwindigkeit und Ist-Lenkwinkel
A2B(7)	Receiver(ID_ODOMETRY_POSITION, Position) recPosition;	empfängt die aktuelle Odometrieposition
A2B(8)	Receiver(ID_POSITION_CORRECTION, Correction) recCorrection;	empfängt den Versatz der akt. zur letzten Konfiguration
A2B(9)	Sender(ID_TARGET_MOTORICS, Motorics) sndMotorics;	liefert Soll-Geschwindigkeit und Soll-Lenkwinkel

Tabelle 2: Informationssender und Empfänger der Module SAM, LASERMAP und A2B

tur des sich aufsummierenden Odometriefehlers. Dazu liest das Modul die Entfernungsmesswerte eines fest auf dem Rollstuhl montierten, und mit einem Öffnungswinkel von 180° ausgestatteten Laserscanners aus. Während der Fahrt werden wiederholt Scans aufgenommen, mit deren Hilfe LASERMAP den Versatz von Position und Ausrichtung des Rollstuhls zwischen zwei Aufnahmen bestimmt. Dieser Versatz wird den anderen Modulen zur Verfügung gestellt, damit diese in Verbindung mit der von SAM gelieferten aktuellen Konfiguration eine korrigierte Kombination aus Position und Orientierung bestimmen können.

4.1.4 Informationsflüsse zwischen den Software-Modulen

Zusammenfassend werden in der Tabelle 2 die wichtigsten, für das Navigationsmodul A2B notwendigen Informationssender und Empfänger aufgelistet. Die Syntax der in der Spalte Sender/Receiver angegebenen Objekte hat die Form:

```
Sender(ID,TYPE) Object;
Receiver(ID,TYPE) Object;
```

Ein Modul, das ein Objekt dieser Art besitzt, kann über die Aufrufe

```
Object.send(Var);
Object.receive(Var);
```

den Inhalt der Variablen Var (vom Typ TYPE der bei NET unter ID registriert wurde) in den vom Modul NET bereitgestellten Speicherbereich schreiben bzw. den in Var zu speichernden Wert aus ihm herauslesen. Die in der Spalte Modul angegebenen und in Klammern stehenden Zahlen verweisen auf den entsprechenden in der Abbildung 14 dargestellten Informationsfluss.

4.2 Architektur des A2B-Moduls

Die in diesem Abschnitt beschriebene Architektur des A2B-Moduls hält sich an das in der Abbildung 15 hellgrau unterlegte Klassendiagramm. Der dunkelgrau unterlegte Bereich beschreibt die Dokumentenschnittstelle zum Programm *Rolland-Bahnplaner*, und wird im folgenden Abschnitt behandelt. Die auf der mobilen Plattform *Rolland* als Prozesse realisierten Module sind in der Sprache C++[8] implementiert. Den Kern des A2B-Moduls bildet die Klasse *A2B*, und stellt in der Methode *main()* die Hauptschleife des Prozesses zur Verfügung. Diese wird in der Rahmenzeit von 32 ms einmal durchlaufen und ruft dabei die Methode *handleEvents()* der Basisklasse *A2BSocket* auf. Die Aufgabe von *handleEvents()* ist es, die im Abschnitt 4.1 beschriebene Kommunikation mit den anderen Modulen zu realisieren.

4.2.1 Methoden und Datenstrukturen der Planungsphase

In dem Fall, bei dem *handleEvents()* eine neue Zielkonfiguration empfängt, wird diese der Methode *newGoal(goalConf:Configuration)* übergeben. Von dort aus wird die Methode *initTrajectory(s:Configuration*,g:Configuration*)* der *A2B* eigenen Membervariablen *m_pTrajectory* aufgerufen. Diese Referenz zeigt auf das einzige Objekt der die Bahnplanung und Trajektoriengenerierung realisierenden Klasse *Trajectory*. Die Methode *initTrajectory(...)* initialisiert zuerst die von ihr verwalteten Konfigurationen q_{start}, q_{ziel} und q_{ss}, q_{zs}[9], sowie die zwischen diesen Konfigurationen liegenden Klothoidensegmente. Anschließend werden wie in Abschnitt 2.4.2 beschrieben, die Größen der acht Kreispaare an den Konfigurationen q_{ss} und q_{zs} berechnet. Dazu gehören der Kreismittelpunkt, die Radien des inneren und äußeren Kreises, sowie die Konfigurationen q_{si} und q_{zi}. Diese Daten werden von der Klasse *AnalyticCircle* verwaltet, von der das einzige Objekt der Klasse *Trajectory* die den acht Kreispaaren entsprechenden Objekte besitzt.

Nun wird durch den Aufruf der zur Klasse *Trajectory* gehörenden Methode *determinePossiblePaths()* ermittelt, welche der 24 möglichen Bahnen existieren. Dazu ruft diese Methode von jeder entsprechenden Referenz der Klasse *Trajectory* auf ein Objekt der Klasse *Path* die Methode *setValid(...)* auf. Die dort stattfindenden Berechnungen realisieren die im Abschnitt 2.4.3 angegebenen Formeln. Im Anschluss daran ruft die Methode *determinePathsQGs()* der Klasse *Trajectory* von jeder gültigen Referenz auf ein Objekt der Klasse *Path* die Methode *calcQGs()* auf. In dieser werden neben den

[8] Momentan wird die Watcom C++ Entwicklungsumgebung(Version 10.6) auf dem QNX Betriebssystem(Version 4.23A) eingesetzt.

[9] Es ist ausreichend q_{ss} und q_{zs} genauso wie q_{start} und q_{ziel} in einem einzigen Objekt zu verwalten, da diese Konfigurationen bei allen möglichen Bahnen gleich sind.

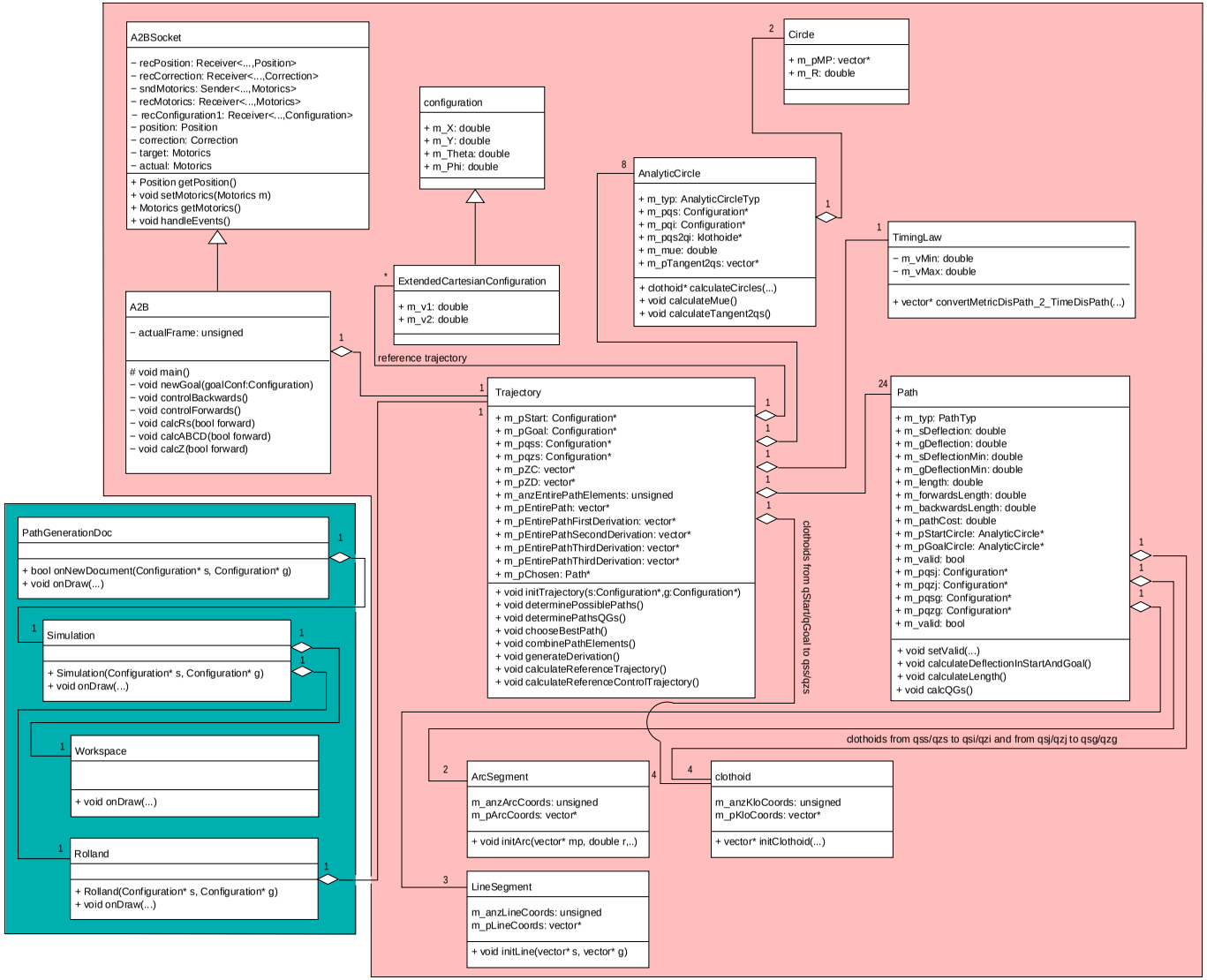


Abbildung 15: Klassendiagramm des A2B-Moduls

Auslenkungen innerhalb der Start und Zielsituation, welche in *Path::m_sDeflection* und *Path::m_gDeflection* gespeichert werden, auch die in der Klasse *Path* verwalteten Konfigurationen q_{sj}, q_{sg}, q_{zj} und q_{zg} berechnet. Danach initialisiert *calcQGs()* die zwischen den zuvor genannten Konfigurationen liegenden Klothoidensegmente, Kreisausschnitte und Geraden, bevor von hier aus die Methode *Path::calculateLength()* aufgerufen wird.

In dieser Methode werden die metrischen Längen der vorwärts und rückwärts zu fahrenden Bahnsegmente, deren Summe und der Wert für *Path::m_pathCost* berechnet. Die „Kosten"einer Bahn können an dieser Stelle gezielt beeinflusst werden, indem etwa ein rückwärts gefahrener Meter doppelt so „teuer" bewertet wird, wie ein vorwärts gefahrener. Nachdem die Methode *initTrajectory()* nun durch den Aufruf von *chooseBestPath()* die „günstigste" Bahn ausgewählt hat, kombiniert sie deren Bahnsegmente durch den Aufruf von *combinePathElements()* zu einer Liste von aufeinanderfolgenden Vektoren, welche die zu fahrende Bahn als Kurve beschreiben. Diese Methode integriert auch die von der Klasse *Trajectory* verwalteten Klothoidensegmente und speichert die Kurve in einer Liste von Vektoren, auf deren erstes Element *Trajectory::m_pEntirePath* zeigt. Anzumerken ist an dieser Stelle, dass jeweils zwei aufeinanderfolgende Vektoren dieser Liste einen konstanten metrischen Abstand zueinander haben.

Die Klasse *Trajectory* verwaltet eine Referenz auf ein Objekt der Klasse *TimingLaw*. Diese Klasse ist durch den aus *initTrajectory(...)* erfolgenden Aufruf von *TimingLaw::convertMetricDisPath_2_TimeDisPath()* dafür zuständig die Bahnvektoren von konstantem metrischen Abstand zu konstantem zeitlichen Abstand zu konvertieren. Siehe dazu auch Abschnitt 2.5.

Nach Bildung der ersten drei Ableitungen der Vektorenliste *m_pEntirePath* durch die Methode *Trajectory::generateDerivation()*[10] werden die fehlenden Konfigurationsparameter sowie Steuerungssignale in der Methode *Trajectory::calculateReferenceTrajectory()* berechnet. Auf die sich ergebende Liste von Objekten der Klasse *ExtendedCartesianConfiguration* verweist der Zeiger *Trajectory::m_pRefTrajectory*. Siehe dazu auch Abschnitt 2.5.3.

Zum Abschluss der von *initTrajectory(...)* ausgeführten Trajektorienplanungsphase wird die Methode *calculateReferenceControlTrajectory()* aufgerufen. Diese berechnet den Verlauf des für das verwendete Regelungsverfahren (siehe Abschnitt 3.2) zu transformierenden Roboterreferenzpunktes und dessen erste Ableitung. Auf die resultierenden Vektorenlisten verweisen die Zeiger *Trajectory::m_pZC* und *Trajectory::m_pZD*.

4.2.2 Methoden und Datenstrukturen der Regelungsphase

Mit der abgeschlossenen Initialisierung der Datenstruktur *A2B::m_pTrajectory* beendet auch die Methode *A2B::newGoal(...)* ihre Arbeit, und der Fokus kehrt zur Hauptschleife der Methode *A2B::main()* zurück. In jedem der folgenden Zeitrahmen wird die zu diesem Zeitpunkt aktuelle Rollstuhlkonfiguration der entsprechenden geplanten Konfiguration nachgeregelt. Dies geschieht über den Aufruf der Methode *A2B::controlForwards()* bzw. *A2B::controlBackwards()*. Diese Methoden ermitteln für den nächsten Rahmen eine neu

[10] Zur Glättung der abgeleiteten Kurven wird ein in [16] vorgestellter Ableitungsfilter verwendet.

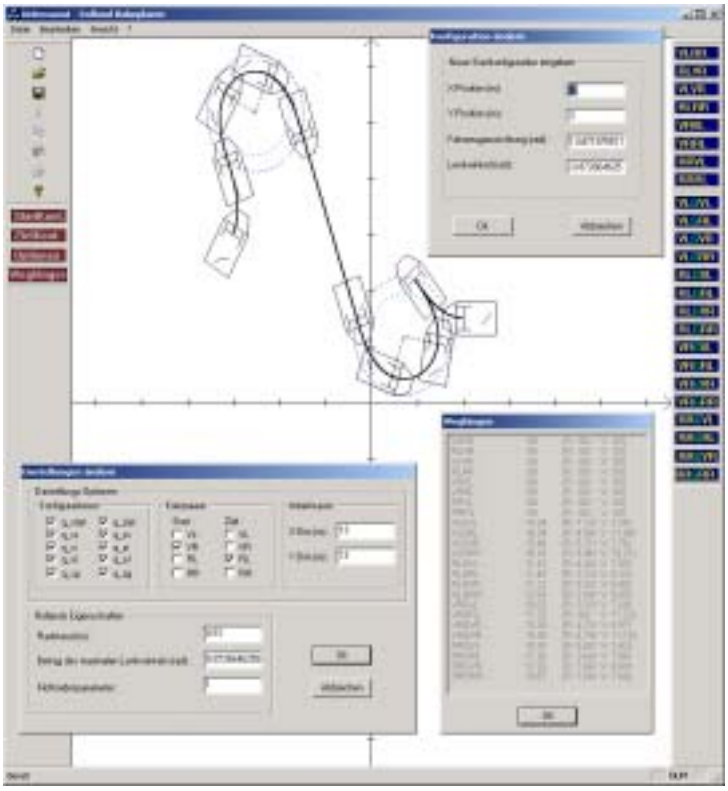


Abbildung 16: Benutzerschnittstelle des Programms Rolland-Bahnplaner mit den zur Verfügung stehenden Dialogfeldern

anzulegende Kombination aus Geschwindigkeit und Lenkwinkel mit Hilfe der Gleichungen 29 und 31.

4.3 Rolland-Bahnplaner

Das Programm *Rolland-Bahnplaner* ist wie das A2B-Modul in der Sprache C++ implementiert. Es wurde mit der Microsoft Visual C++ Entwicklungsumgebung erstellt, und stützt sich auf deren Klassenbibliothek MFC Version 4.2. Aufgabe des Programms ist es, Instanzen der in Abschnitt 2.4.3 beschriebenen Bahntypen zwischen zwei beliebigen Konfigurationen darzustellen.

4.3.1 Architektur von Rolland-Bahnplaner

Grundsätzlich entspricht die Architektur von *Rolland-Bahnplaner* dem in der Abbildung 15 hellgrau unterlegten Klassendiagramm. Die Unterschiede zum A2B-Modul, die sich daraus ergeben, dass *Rolland-Bahnplaner* als Applikation einer grafischen Oberfläche realisiert und mit reinen Darstellungsfunktionen ausgestattet ist, werden im Folgenden beschrieben.

Das in einem mit grafischer Oberfläche ausgestattete, Multitaskingsystem arbeitende Programm, ist neben der Nachrichtenbehandlung auch für die strikte Kapselung der zu verarbeitenden Daten zuständig. Diese werden auf der verwendeten WinXX-Plattform mit dem Synonym Dokument bezeichnet. Das Rahmenprogramm von *Rolland-Bahnplaner*, welches die in Abbildung 15 dargestellten Klassen *A2B* und *A2BSocket* ersetzt und selbst dort nicht dargestellt ist, besitzt eine Referenz auf das einzige ein Dokument realisierende Objekt der Klasse *PathGenerationDoc*. Diese sowie die Klassen *Simulation*, *Workspace* und *Rolland* sind auf dem dunkelgrau unterlegten Teil des in Abbildung 15 dargestellten Klassendiagramms abgebildet. Sie formen im Wesentlichen eine Aggregationskette von der Dokumentenklasse bis zur Klasse *Trajectory*, welche, wie im Abschnitt 4.2.1 beschrieben wurde, den Kern der Bahnplanung realisiert. Im Unterschied zum A2B-Modul kann hier auf die Klasse *TimingLaw* und die aus *Trajectory::initTrajectory(...)* stattfindenden Berechnungen der Referenztrajektorien *Trajectory::m_pRefTrajectory* und *m_pZC* verzichtet werden. Der Grund hierfür ist, dass für Darstellungszwecke eine Beschreibung der Bahnen als Vektorenliste mit konstantem metrischem Abstand zwischen zwei aufeinanderfolgenden Vektoren ausreicht.

Wird vom Rahmenprogramm aus die Methode *PathGenerationDoc::onDraw(...)* aufgerufen, wird dieser Aufruf bis zu den Klassen durchgereicht, deren Objekte sich selbst darstellen. Dazu gehören die Klasse *Workspace*, die das Koordinatensystem zeichnet, und die Klasse *Configuration*, die den stilisierten Rollstuhl in den ausgewählten Konfigurationen darstellt. Die gelenkten Räder des Rollstuhls sind dabei zu einem einzelnen Rad in der Mitte der Hinterachse zusammengefasst. Außerdem besitzen die Klassen *ArcSegment*, *Clothoid*, *LineSegment* und *Circle* eine *onDraw(...)*-Methode um die verschiedenen Bahnsegmente sowie die ausgewählten Kreispaare an den Konfigurationen q_{ss} und q_{zs} zu zeichnen.

4.3.2 Benutzerschnittstelle von Rolland-Bahnplaner

Der in der Abbildung 16 dargestellte Screenshot zeigt das Programm *Rolland-Bahnplaner* sowie die zur Verfügung stehenden Dialogfelder. In diesem Abschnitt wird die Funktionalität der einzelnen Bedienelemente beschrieben.

Nach dem Start des Programms wird *Rolland* in den mit Standardwerten initialisierten Konfigurationen q_{start} und q_{ziel} angezeigt. Diese befinden sich innerhalb eines den Arbeitsraum $\mathcal{W}$ andeutenden Koordinatensystems. Zusätzlich werden auch die Klothoidensegmente zwischen den Konfigurationen q_{start} und q_{ss} bzw. q_{ziel} und q_{zs} dargestellt. Diese gehören immer dann zu einer beliebigen Bahn, wenn der Lenkwinkel innerhalb der Start- bzw. Zielkonfiguration ungleich null ist. Durch Druck auf einen Knopf in der rechts angeordneten Menüleiste wird die entsprechende Bahn zwischen q_{ss} und q_{zs} dargestellt, falls diese existiert. Die Beschriftungen der einzelnen Knöpfe halten sich an die Benennungen der zu einer Bahn gehörenden Kreispaare K_{vl}, K_{rl}, K_{vr} und K_{rr}. Dabei stehen die ersten beiden Buchstaben für das Kreispaar des zu fahrenden CCT in der Startsituation und die letzten beiden für den CCT der Zielsituation. Ein dazwischen stehendes G deutet eine Bahn mit einem zwischen den $CCTs$ liegenden Geradensegment an.

Die Konfigurationen q_{start} und q_{ziel} können durch Drücken der Knöpfe *StartKonf.* und *ZielKonf.* in der links angeordneten Menüleiste verändert werden. Das sich öffnende, und in der Abblidung 16 oben rechts dargestellte Dialogfenster *Konfiguration ändern* nimmt Eingaben für eine neue metrische Position, die Fahrzeugausrichtung und den Lenkwinkel entgegen.

Die Einstellungen des Programms können in dem Dialogfenster *Einstellungen ändern* angepasst werden. Das in der Abbildung 16 unten links dargestellte Fenster öffnet sich durch Druck auf den in der linken Menüleiste platzierten Knopf *Optionen*. Hier können neben reinen Darstellungsoptionen auch die Eigenschaften der simulierten mobilen Plattform verändert werden.

Zu den Darstellungsoptionen zählen die Auswahl der anzuzeigenden und zwischen den einzelnen Bahnsegmenten liegenden Konfigurationen, und die Auswahl der acht Kreispaare an den Konfigurationen q_{ss} und q_{zs}. Weiterhin kann der darzustellende Bereich des Koordinatensystems verändert werden. Dies entspricht einem Hereinzoomen in bzw. Herauszoomen aus den angezeigten Arbeitsraum.

Die Eigenschaften von *Rolland* setzen sich aus dem Radstand l der Plattform, dem Betrag des maximalen Lenkwinkels und dem Klothoidenparameter a_K(Siehe dazu Gleichung 10 sowie folgende Erläuterungen.) zusammen. Das Variieren des Radstandes hat in diesem Programm Auswirkungen auf die Darstellung einer Konfiguration, indem sich der Abstand zwischen Vorder- und Hinterachse vergrößert bzw. verkleinert[11]. Durch das Verändern des Wertes für den Betrag des maximalen Lenkwinkels kann die Grenze für den entsprechenden Wert der Konfigurationen q_{start} und q_{ziel} gesetzt werden. Ein kleinerer bzw. größerer Wert vergrößert bzw. verkleinert die Radien der an q_{ss} und q_{zs} anliegenden Kreispaare. Letztlich bewirkt eine Veränderung des Klothoidenparameters a_K die Länge der einzelnen Klothoidensegmente. Ein größerer Wert für a_K verlängert ein solches Segment, was so zu interpretieren ist, dass *Rolland* bei konstanter Fahrtgeschwindigkeit länger braucht, um seinen Lenkwinkel zu verstellen.

Durch das Betätigen des in der linken Menüleiste platzierten Knopfes *Weglängen* öffnet sich schließlich das in der Abbildung 16 unten rechts dargestellte Dialogfenster *Weglängen*. Hier werden die metrische Gesamtlänge einer jeden Bahn, und deren vorwärts und rückwärts zu fahrender Anteil aufgelistet. Für den Fall, dass eine Bahn nicht existiert, sind alle drei Werte auf null gesetzt.

[11] In dem A2B-Modul wirkt sich der Radstand l auf die Berechnung der Referenztrajektorienwerte ϕ und $v2$ aus. Siehe Gleichungen 22 und 24.

5 Ergebnisse und Diskussion

In diesem Abschnitt werden die im Rahmen dieser Diplomarbeit erzielten Ergebnisse beschrieben. Neben der Analyse einzelner Testläufe wird der Versuch unternommen, grundlegende Eigenschaften der mit Hilfe der CCT-Bahnplanung gewonnenen Bahnen zu benennen. Die sich daraus ableitende Kritik liefert schließlich die Grundlage einiger Ansätze zur Weiterentwicklung.

5.1 Testläufe

Die in den vorigen Abschnitten vorgestellte Trajektorienplanung und Trajektorienfolgeregelung ist sowohl auf der realen Plattform *Rolland*, als auch innerhalb der Simulationssoftware *SimRobot* getestet worden. Beispielhaft zeigen die Diagramme im Abschnitt 5.1.1 die Ergebnisse von fünf Testläufen. Die zugehörigen initialen Werte für die Start- und Zielkonfiguration, die Geschwindigkeitsvorgaben, sowie die ausgewählten Bahntypen sind der Tabelle 3 zu entnehmen.

Die Parameter für die Anstiegsgeschwindigkeit der Klothoidenkrümmung (a_K siehe Gleichung 10) und die Regelungsphase (Δ siehe Gleichung 27, $r1$ und $r2$ siehe Gleichung 31) wurden wie folgt gewählt:

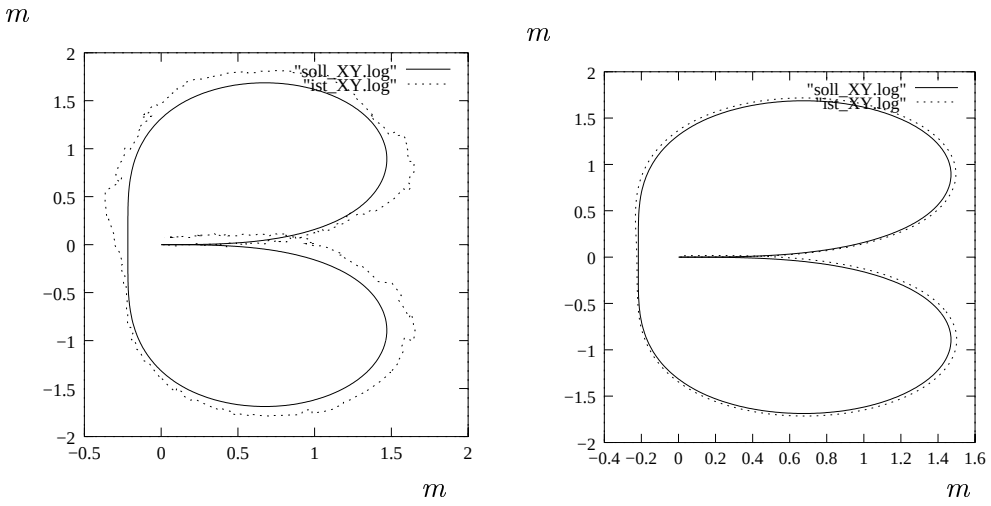
- *Rolland*: $a_K = 1.05$, $\Delta = 0.01625$, $r1 = 0.9$, $r2 = 0.9$
- *SimRobot*: $a_K = 1.05$, $\Delta = 0.01625$, $r1 = 3.45$, $r2 = 3.45$

Testlauf	Typ	$q_{start} = [m, m, rad, rad]^T$	$q_{ziel} = [m, m, rad, rad]^T$	$v1_{start} = \frac{m}{s}$	$v1_{ziel} = \frac{m}{s}$
1	VLGRL	$[0, 0, 0, 0]^T$	$[0, 0, \pi, 0]^T$	0	0
2	VRGRL	$[0, 0, 0, 0]^T$	$[-2, -2, -\frac{\pi}{4}, 0]^T$	0	0
3	RLGVL	$[0, 0, 0, 0]^T$	$[0, 0, \pi, 0]^T$	0	0
4	VLGRL	$[0, 0, 0, 0]^T$	$[0, -1, \frac{\pi}{2}, 0]^T$	0	0
5	VLGRR	$[0, 0, 0, 0]^T$	$[2, 5, 0, 0]^T$	0	0.4

Tabelle 3: Start- und Zielkonfigurationen, sowie Geschwindigkeitsvorgaben und Bahntypen der in Abschnitt 5.1.1 dargestellten Testläufe

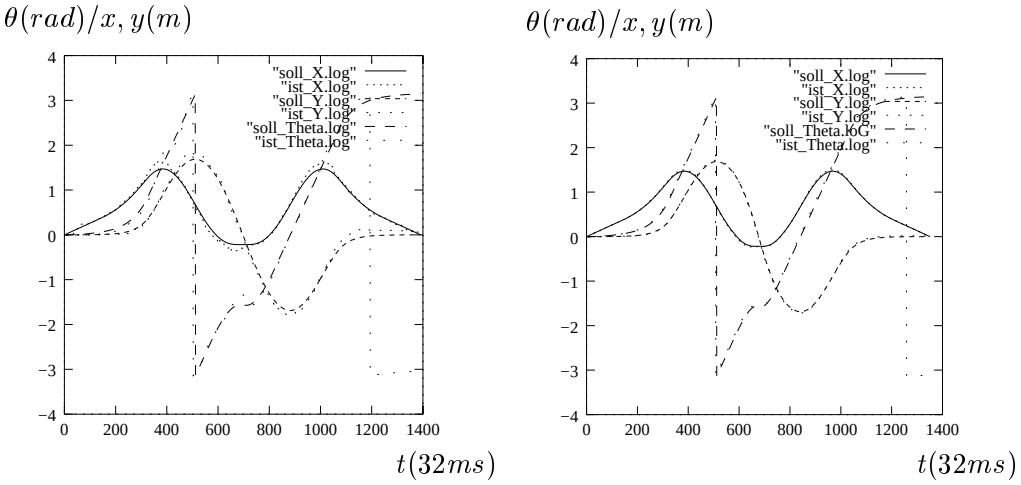
Für die fünf ausgewählten Testläufe sind auf den folgenden Seiten für *Rolland* und *SimRobot* jeweils vier Diagramme dargestellt. Das erste Diagramm repräsentiert die geplante und gefahrene Trajektorie innerhalb des Arbeitsraumes $\mathcal{W}$. Das zweite Diagramm zeigt die geplante und reale Entwicklung der einzelnen Konfigurationsparameter x, y und θ, während im dritten Diagramm die Abweichungen zwischen geplanten und erzielten Konfigurationsparametern zu sehen sind. Im vierten Diagramm werden schließlich die vorberechneten und geregelten Stellgrößen $v1$ und ϕ miteinander verglichen. Anzumerken ist noch, dass die Konfigurationsparameter, Fehlerwerte und Stellgrößen in ihrem zeitlichen Verlauf dargestellt sind, deren Einheit die vorgegebene Taktzeit von 32 ms liefert.

5.1.1 Auswertung der Testläufe



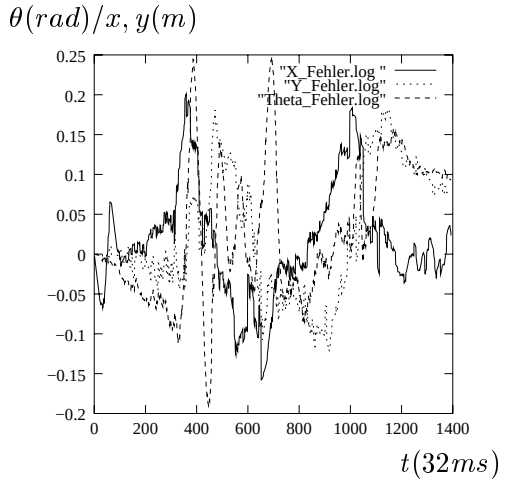
(a) Rolland (b) SimRobot

Abbildung 17: Geplante und gefahrene Trajektorie 1

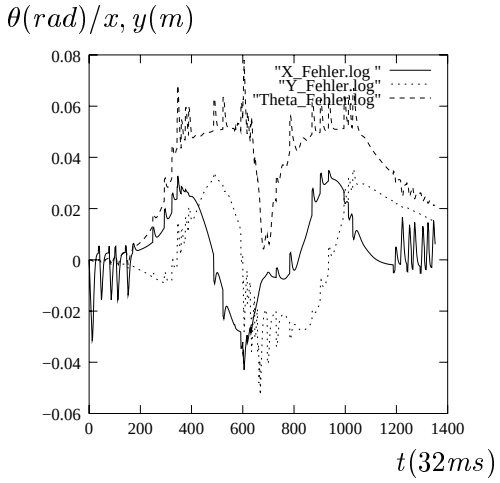


(a) Rolland (b) SimRobot

Abbildung 18: Entwicklung der Konfigurationsparameter x, y, θ der Trajektorie 1

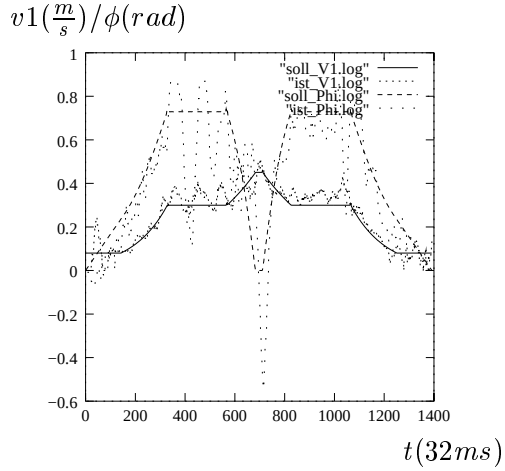


(a) Rolland

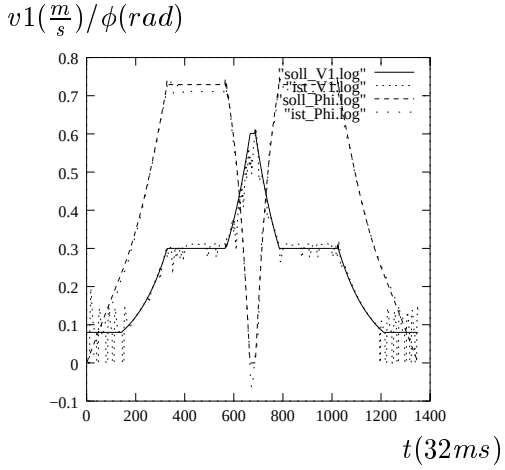


(b) SimRobot

Abbildung 19: Abweichung der Konfigurationsparameter x, y, θ von den geplanten Werten der Trajektorie 1

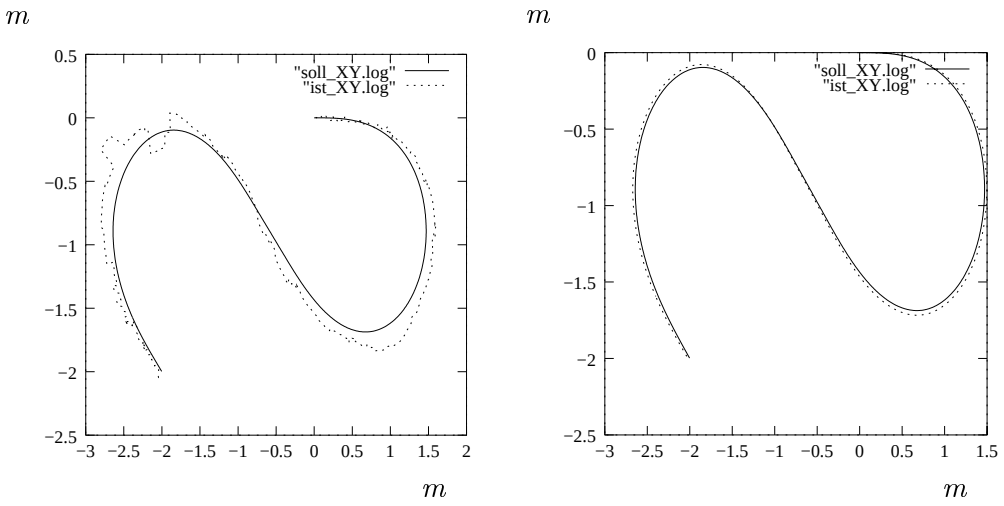


(a) Rolland



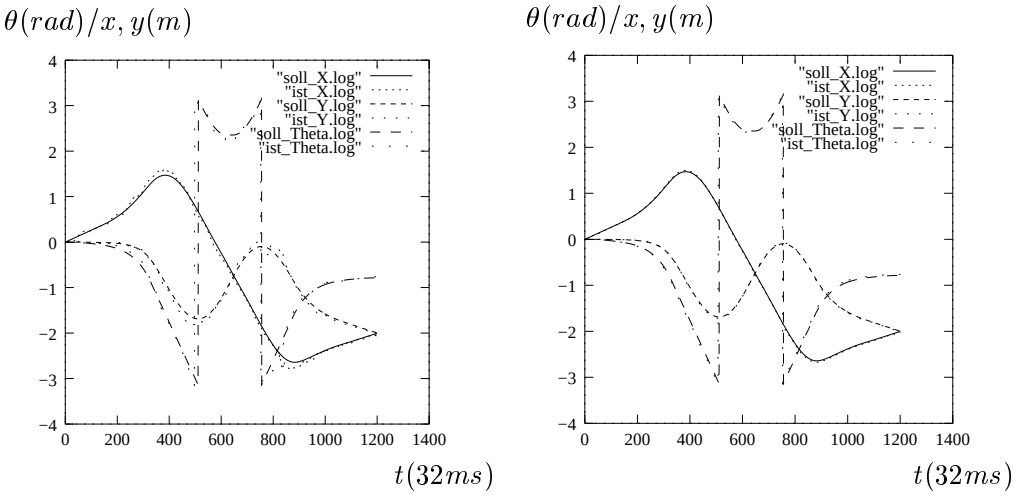
(b) SimRobot

Abbildung 20: Entwicklung der Stellgrößen $v1, \phi$ der Trajektorie 1



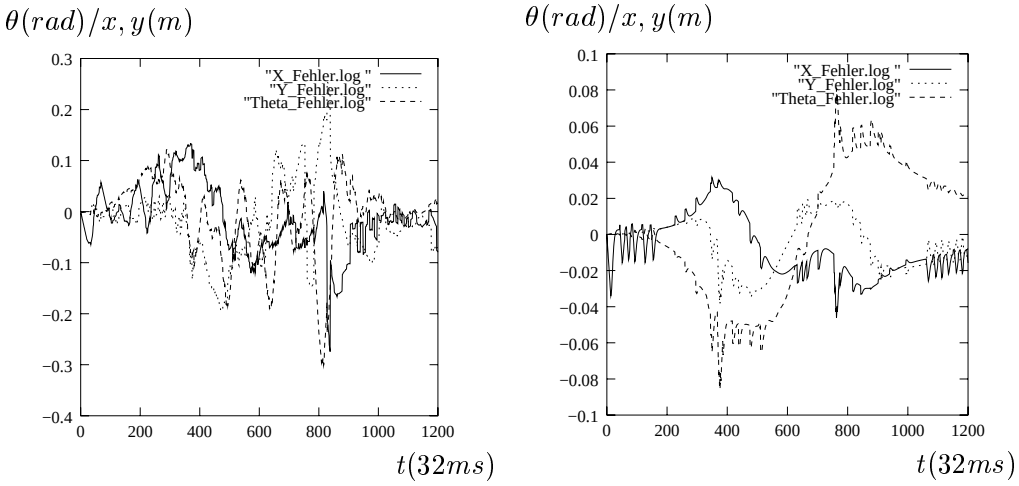
(a) Rolland (b) SimRobot

Abbildung 21: Geplante und gefahrene Trajektorie 2



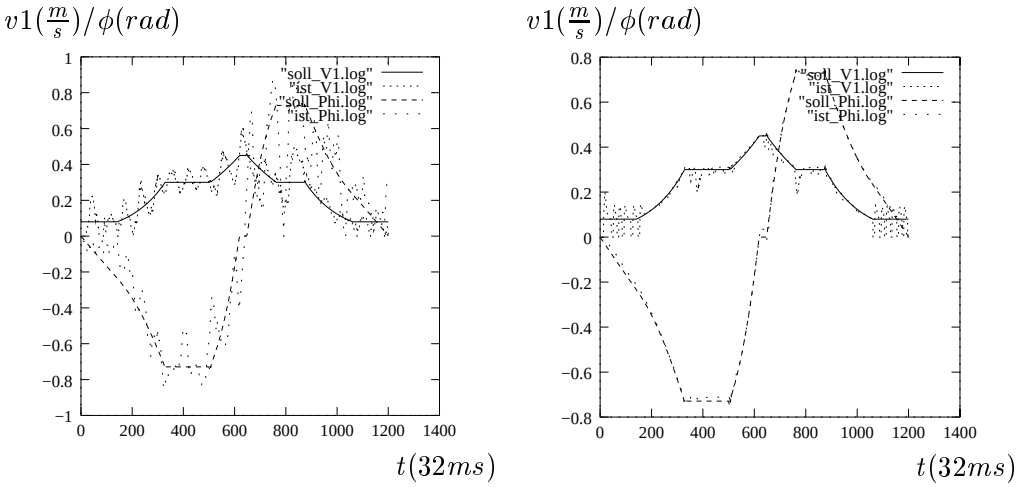
(a) Rolland (b) SimRobot

Abbildung 22: Entwicklung der Konfigurationsparameter x, y, θ der Trajektorie 2



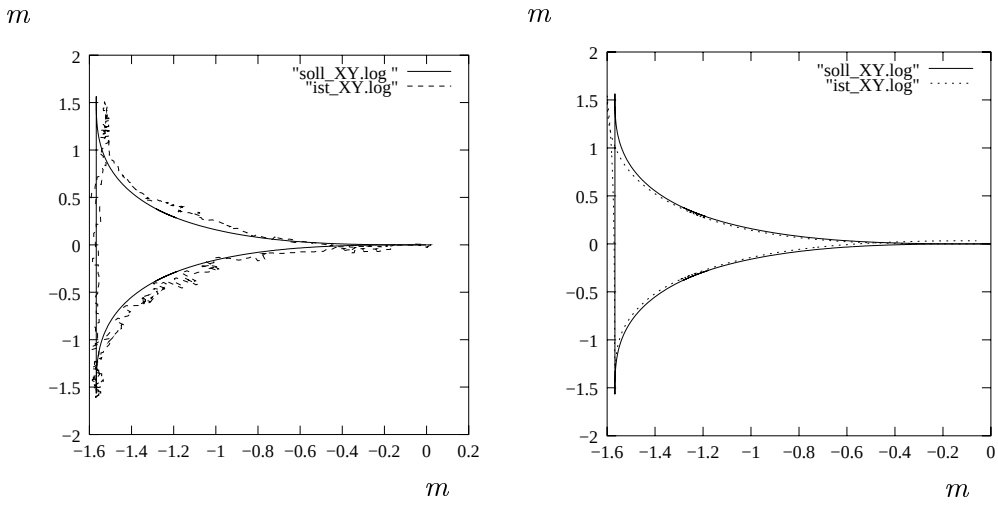
(a) Rolland (b) SimRobot

Abbildung 23: Abweichung der Konfigurationsparameter x, y, θ von den geplanten Werten der Trajektorie 2



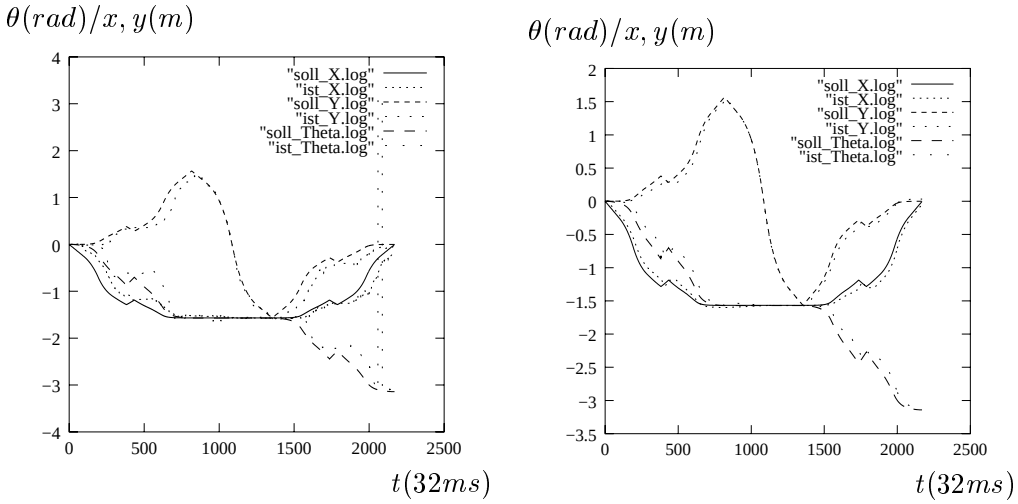
(a) Rolland (b) SimRobot

Abbildung 24: Entwicklung der Stellgrößen $v1, \phi$ der Trajektorie 2



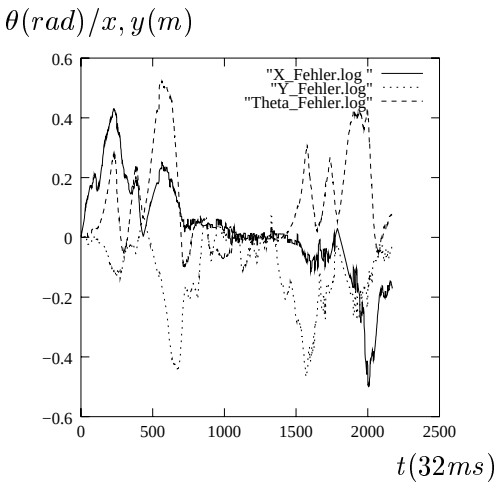
(a) Rolland (b) SimRobot

Abbildung 25: Geplante und gefahrene Trajektorie 3

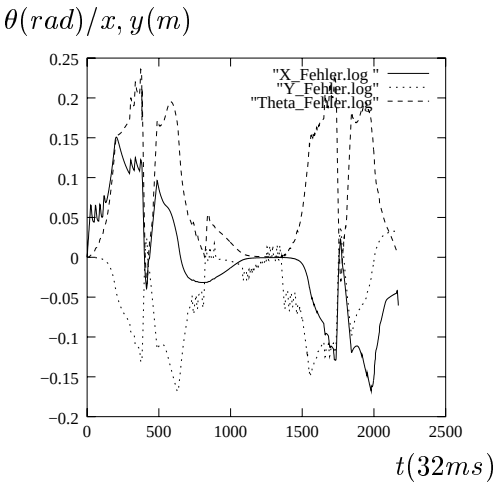


(a) Rolland (b) SimRobot

Abbildung 26: Entwicklung der Konfigurationsparameter x, y, θ der Trajektorie 3

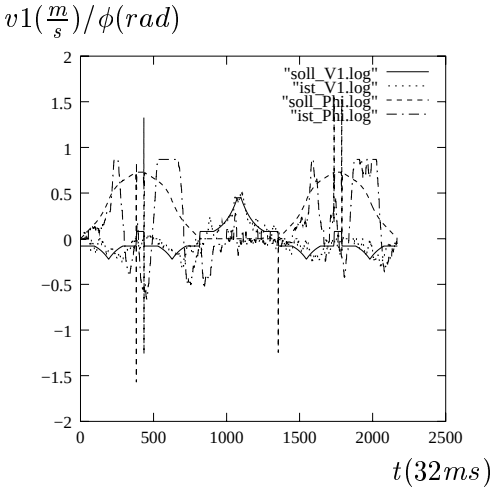


(a) Rolland

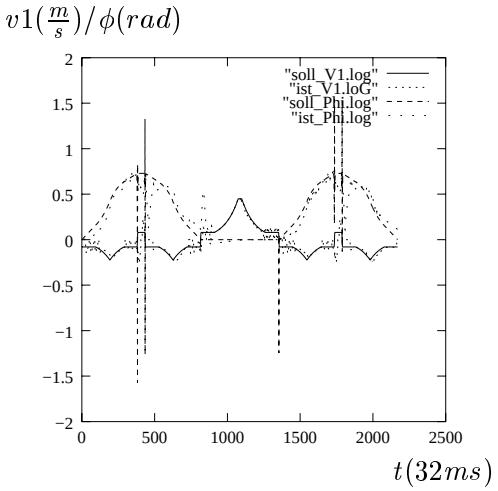


(b) SimRobot

Abbildung 27: Abweichung der Konfigurationsparameter x, y, θ von den geplanten Werten der Trajektorie 3

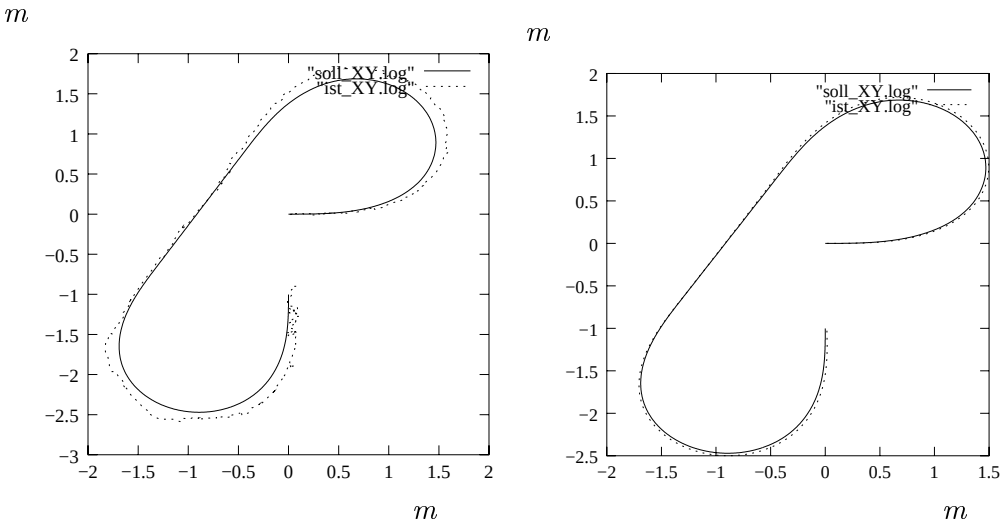


(a) Rolland



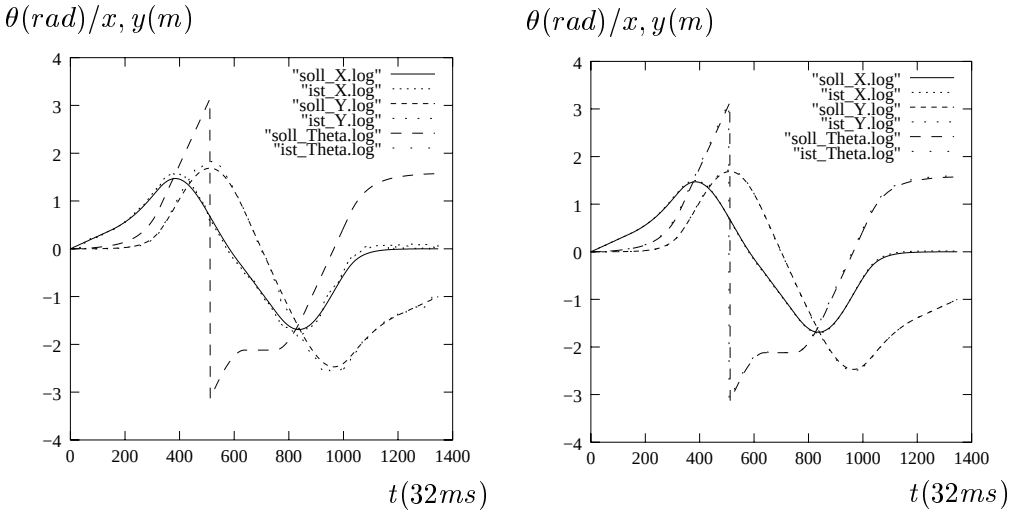
(b) SimRobot

Abbildung 28: Entwicklung der Stellgrößen $v1, \phi$ der Trajektorie 3



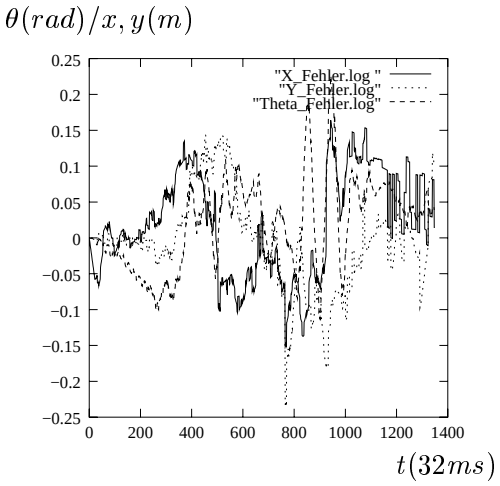
(a) Rolland (b) SimRobot

Abbildung 29: Geplante und gefahrene Trajektorie 4

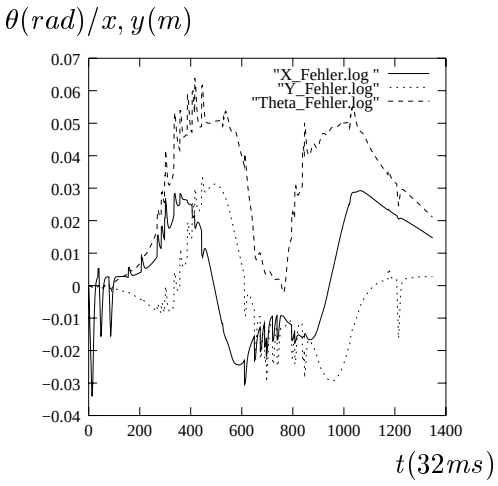


(a) Rolland (b) SimRobot

Abbildung 30: Entwicklung der Konfigurationsparameter x, y, θ der Trajektorie 4

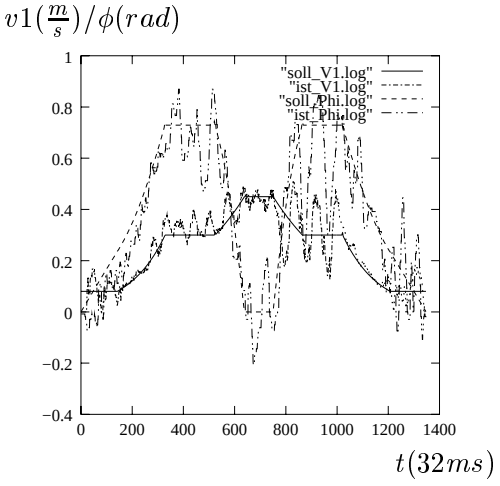


(a) Rolland

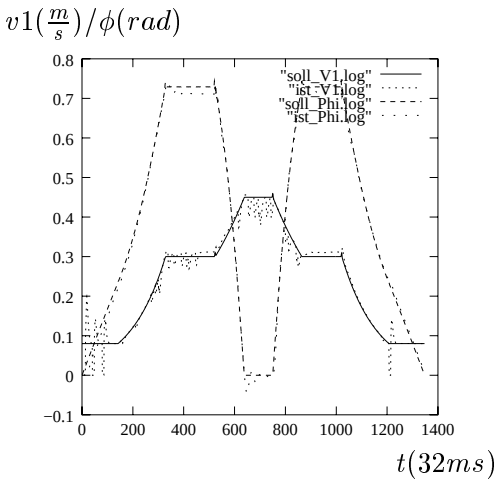


(b) SimRobot

Abbildung 31: Abweichung der Konfigurationsparameter x, y, θ von den geplanten Werten der Trajektorie 4

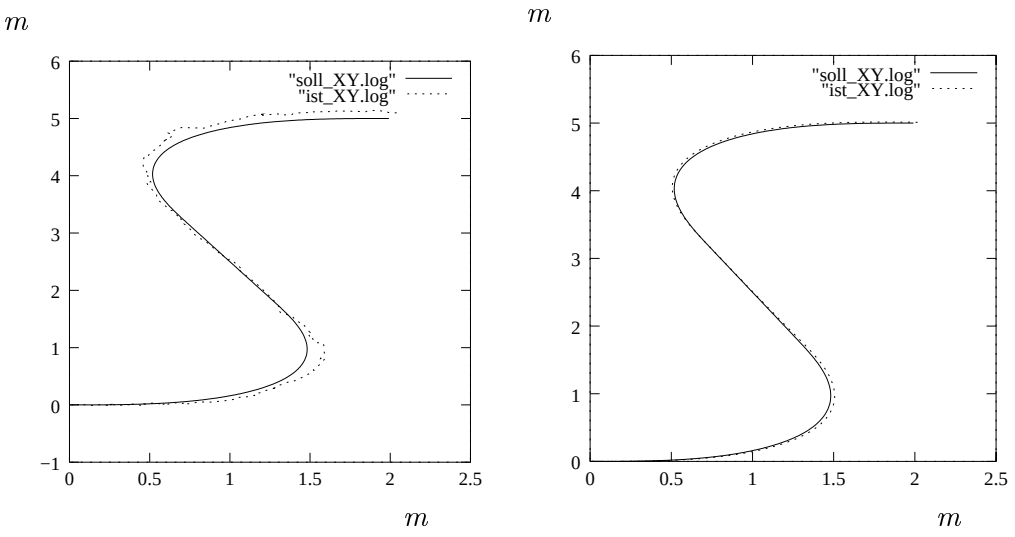


(a) Rolland



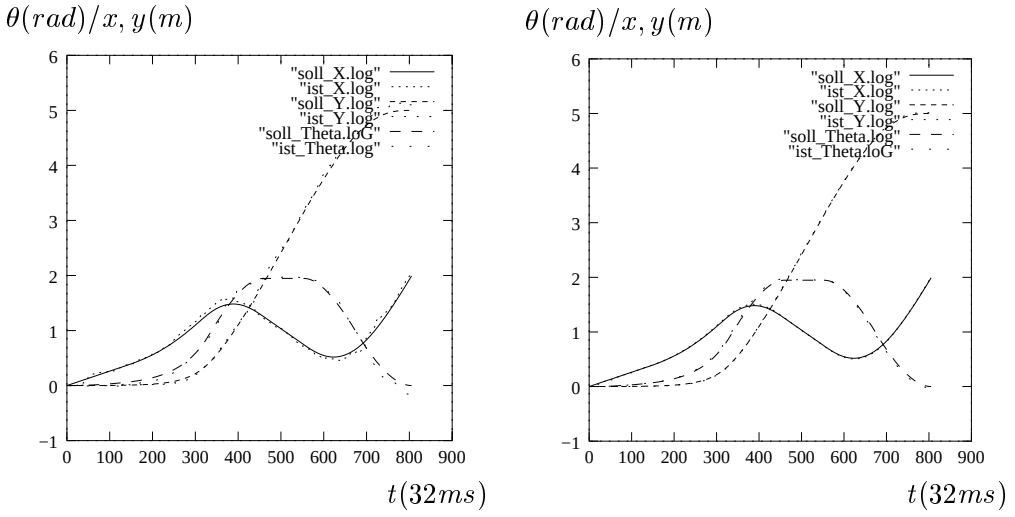
(b) SimRobot

Abbildung 32: Entwicklung der Stellgrößen $v1, \phi$ der Trajektorie 4



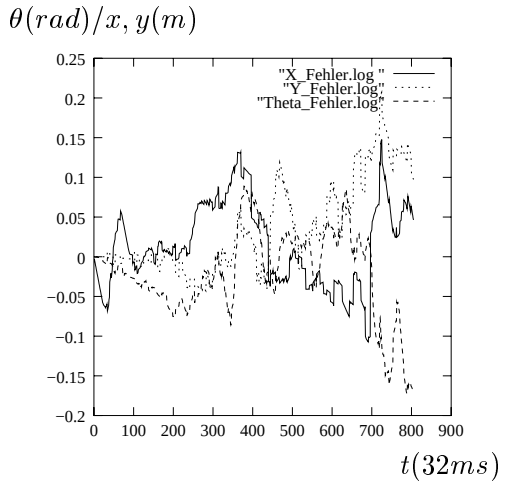
(a) Rolland (b) SimRobot

Abbildung 33: Geplante und gefahrene Trajektorie 5

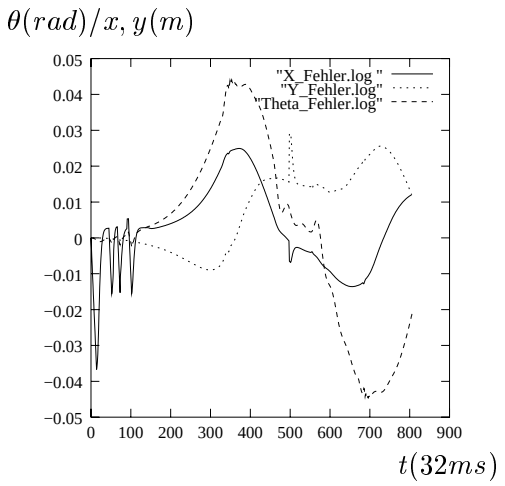


(a) Rolland (b) SimRobot

Abbildung 34: Entwicklung der Konfigurationsparameter x, y, θ der Trajektorie 5

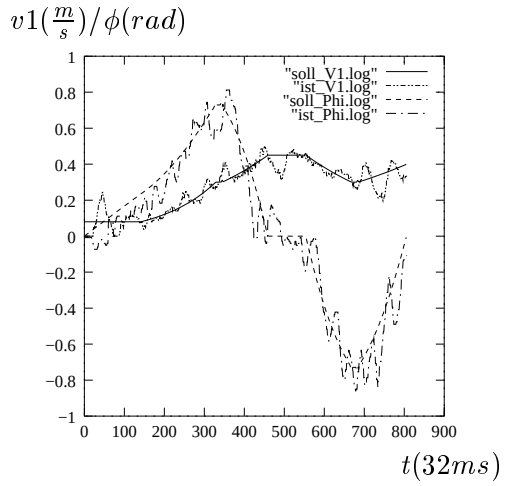


(a) Rolland

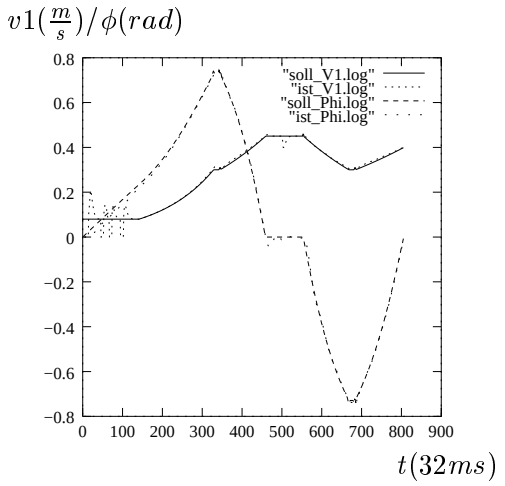


(b) SimRobot

Abbildung 35: Abweichung der Konfigurationsparameter x, y, θ von den geplanten Werten der Trajektorie 5



(a) Rolland



(b) SimRobot

Abbildung 36: Entwicklung der Stellgrößen $v1, \phi$ der Trajektorie 5

5.1.2 Diskussion der Testläufe

Wie den Abbildungen 17, 21, 25, 29 und 33 zu entnehmen ist, wird das Fahrzeug aus seiner Startkonfiguration über eine Bahn, deren Kurvenkrümmung kontinuierlich variiert, in eine Zielkonfiguration überführt. Dabei wird die geplante Konfiguration q_{ziel} von *Rolland* maximal um den Fehlervektor $q_{MaxZielfehler} \approx [\pm 0.2m, \pm 0.15m, \pm 0.17rad]$ verfehlt. Erwartungsgemäß bleibt die Abweichung der Simulation *SimRobot* mit $q_{MaxZiehlfehler} \approx [\pm 0.06m, \pm 0.04m, \pm 0.02rad]$ deutlich geringer. Außer diesen Werten ist den Abbildungen 19, 23, 27, 31 und 35 noch der Fehlervektor $q_{MaxFehler}$ zu entnehmen, der die maximalen Abweichungen über die gesamte Trajektorie beschreibt. Für *Rolland* kann dieser mit $q_{MaxFehler} \approx [\pm 0.5m, \pm 0.4m, \pm 0.3rad]$ und für *SimRobot* mit $q_{MaxFehler} \approx [\pm 0.15m, \pm 0.18m, \pm 0.24rad]$ angegeben werden.

Der direkte Vergleich zwischen den von *Rolland* und *SimRobot* erzielten Werten macht deutlich, dass ein qualitativer Unterschied zwischen dem mathematisch idealisierten Modell und der realen Plattform besteht. Hierfür sind mehrere Gründe zu nennen.

So zeigen die Abbildungen 20, 24, 28, 32 und 36, dass die Stellgrößen $v1$ und ϕ innerhalb der Simulation verhältnismäßig präzise den vorgegebenen Werten folgen, während sie sich bei *Rolland* oszillierend den Vorgaben annähern.

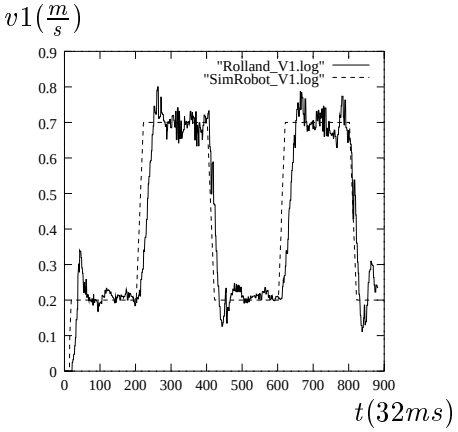


Abbildung 37: Konstante Geschwindigkeitsvorgaben und deren Umsetzung

Dieser Effekt ist selbst bei konstanten Geschwindigkeitsvorgaben auf den beiden Testplattformen zu beobachten. Die Abbildung 37 zeigt das Verhalten von *SimRobot* und *Rolland* bei dem Versuch, jeweils 200 Zeittakte lang eine konstante Geschwindigkeit von $v1 = \frac{0.2m}{s}$ bzw. $v1 = \frac{0.7m}{s}$ einzuhalten. Das von *Rolland* erzielte Geschwindigkeitsprofil ist so zu erklären, dass ein interner Hardwareregler die aktuell gemessene Geschwindigkeit an die Vorgabe anzugleichen versucht. Diese Verfahrensweise ist notwendig, um die Ungenauigkeiten der Geschwindigkeitsmessung und -umsetzung auszugleichen, beeinträchtigt aber die Güte des für die Trajektorienfolgeregelung verwendeten Softwareregelers.

Ein weiterer Grund für das relativ unpräzise Folgen einer vorgegebenen Trajektorie ist bei *Rolland* in dem Selbstlokalisationsverfahren zu suchen. Während die Simulation exakte Daten bezüglich der aktuellen Konfiguration zur Verfügung stellt, ist die mobile Plattform auf Sensorikinformationen angewiesen. Die Abblidung 21(a) zeigt beispielsweise das Resultat der Verfolgung einer S-förmigen Trajektorie. Die in der Abbildung oben links zu erkennenden Sprünge entsprechen nicht der realen Fahrt des Fahrzeuges, da in diesem Fall eine zur Fahrzeugausrichtung senkrechte und damit unmögliche Bewegung notwendig gewesen wäre. Vielmehr sind diese Sprünge auf die vom Modul LASERMAP (siehe dazu Abschnitt 4.1.3) korrigierten Odometriedaten zurückzuführen.

Neben den beschriebenen Unterschieden zwischen *SimRobot* und *Rolland* bleibt zu klären, ob sich die hier diskutierten Planungs- und Regelungsalgorithmen für einen realen Einsatz eignen. Es zeigt sich, dass die Trajektorien 1, 2, 4 und 5 von *Rolland* mit einem maximalen Fehler von $q_{MaxFehler*} \approx [\pm 0.25m, \pm 0.25m, \pm 0.3rad]$ nachgeregelt werden. Diese unterscheiden sich von der für $q_{MaxFehler}$ verantwortlichen Trajektorie 3 darin, dass sie keinen Fahrtrichtungswechsel benötigen. Gerade diese Möglichkeit der *CCT*-Bahnplanung, einen Vorzeichenwechsel der Stellgröße $v1$ einzuplanen, erschwert deren Regelung. So weisen die Fehlerwerte für die Konfigurationsparameter x, y und θ sprunghafte Ausschläge zu den Zeitpunkten auf, an denen die Stellgröße $v1$ das Vorzeichen wechselt. Siehe dazu die Abbildungen 27 und 28. Folgt auf eine solche Situation ein Trajektorienabschnitt bei dem die Geschwindigkeit $v1$ ein konstantes Vorzeichen aufweist, ist das Regelungsverfahren in der Lage die Fehler wieder auszugleichen. So kann erklärt werden, dass $q_{MaxZielfehler}$ bei den hier vorgestellten Testläufen deutlich geringer als $q_{MaxFehler}$ bleibt, da alle 5 Trajektorien zwischen den Konfigurationen q_{zi} und q_{ziel} eine ausschließlich positive oder negative Geschwindigkeit $v1$ aufweisen.

Letztlich werden die Anforderungen der mit dem A2B-Modul zusammenarbeitenden Softwarekomponenten bestimmen, ob die erreichte Regelungsgüte ausreichend ist. So wird ein für die globale Navigation zuständiges Modul die (in dieser Arbeit nicht berücksichtigte) Hindernisvermeidung realisieren müssen. Der dann einzuhaltende Sicherheitsabstand zwischen den geplanten Konfigurationen einer Trajektorie und den umgebenden Hindernissen wird die Anforderungen an den Fehlervektor $q_{MaxFehler}$ definieren. Stellt sich schließlich heraus, dass die hier vorgestellte Trajektorienfolgeregelung mittels *Eingangs-Ausgangs -Linearisierung mit statischer Rückführung* unzureichend ist, kann auf eine Vielzahl von weiteren Regelungsverfahren zurückgegriffen werden.

So werden in [8] zwei Regelungsverfahren vorgestellt, die das kinematische Modell des nicht-holonomen Roboters zuerst in eine kanonische Form überführen, bevor mittels *annähernder Linearisierung* bzw. *kompletter Zustandslinearisierung via dynamischer Rückführung*[12] ein linearer Zusammenhang zwischen den Stellgrößen $v(t)$ und der Konfiguration des Roboters $q(t+1)$ hergestellt wird. Das letztere Verfahren ist in der Literatur auch unter dem Begriff der *differentiellen Flachheit* zu finden.

5.2 Diskussion der CCT-Bahnplanung

Bei der Beschreibung der *CCT*-Bahnplanung, wurde in Abschnitt 2.4 im Wesentlichen der kontinuierliche Krümmungsverlauf der resultierenden Kurve herausgearbeitet. Das damit verbundene Ziel war es, an den Übergängen zwischen den einzelnen Bahnsegmenten eine den Fahreigenschaften von *Rolland* entsprechende Bahn zu planen, und die Abweichungen zur Fahrtzeit möglichst gering zu halten.

Wie sich bei den im Abschnitt 5.1.1 dokumentierten Testläufen herausgestellt hat, divergieren die geplanten und gefahrenen Bahnen insbesondere, wenn die Stellgröße $v1$ ihr Vorzeichen wechselt. Dieses Verhalten motiviert bei der Suche nach redundanten Fahrtrichtungswechseln einer Bahn. Eine Optimierung in dieser Hinsicht würde auch

[12] In [8] wird darauf hingewiesen, dass auch diese beiden Verfahren eine Singularität aufweisen, wenn die Stellgröße $v1$ im Verlauf der Trajektorie ihr Vorzeichen wechselt.

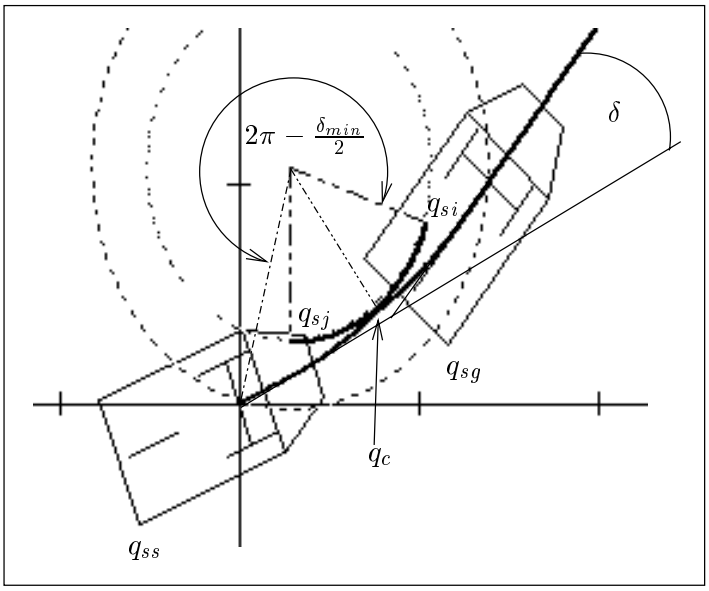


Abbildung 38: CCT mit der Auslenkung $0 < \delta < \delta_{min}$

einem Benutzer des Serviceroboters *Rolland* entgegenkommen, da unnötiges Rangieren im Allgemeinen als unangenehm empfunden wird.

In diesem Zusammenhang zeigt die Abbildung 38 einen CCT, bei dem die Auslenkung δ zwischen den Konfiguration q_{ss} und q_{sg} kleiner als die minimale Auslenkung δ_{min} ist[13]. Der zwischen den Konfigurationen q_{si} und q_{sj} rückwärts zu fahrende Kreisausschnitt lässt sich nach [14] vermeiden, indem der CCT aus der Abbildung 38 durch zwei zueinander symmetrische Klothoiden ersetzt wird. Deren Krümmung variiert von 0 in q_{ss} nach $k_c < k_{max}$ in der zwischen den Klothoiden liegenden Konfiguration q_c, bzw. von k_c in q_c zurück nach 0 in q_{sg}. Beide Klothoiden repräsentieren in diesem Fall die Fahrt von *Rolland*, bei der dieser seinen Lenkwinkel von $\phi_{ss} = 0$ über $\phi_c < \phi_{max}$ wieder nach $\phi_{sg} = 0$ verstellt. Die Implementierung dieser Planungserweiterung bietet sich neben dem Vorteil der Vermeidung von Fahrtrichtungswechseln an, da die aktuelle Version der in dieser Arbeit realisierten Bahnplanung schon Klothoidensegmente in dem Krümmungsbereich von 0 bis $k < k_{max}$ unterstützt. Diese werden im Moment für die Bahnsegmente von q_{start} nach q_{ss} und von q_{ziel} nach q_{zs} eingesetzt.

Eine abschließende Beurteilung der CCT-Bahnplanung kann nur im Zusammenhang mit einem möglichen Einsatzszenario des A2B-Moduls gesehen werden. So wird eine Minimalanforderung den Einsatz in einer a priori bekannten und strukturierten Umgebung vorsehen. Beispiele hierfür sind Büro- und Klinikumgebungen, die durch zumeist stationäre Hindernisse wie Raum- und Flurwände, und unverrückbares Mobiliar gekennzeichnet sind.

Zunächst wird es, im Gegensatz zu dem in dieser Arbeit angenommenen hindernisfrei-

[13] Es sei daran erinnert, dass für $\delta = \delta_{min}$ ein CCT ausschließlich aus zwei Klothoiden besteht, deren Krümmung von 0 bis k_{max} variiert.

en Arbeitsraum, darauf ankommen eine breitere Palette von Bahntypen zur Verfügung zu stellen. Dazu bietet sich die Implementierung der in der Aufzählung 15 gegebenen Bahntypen an. Es sei daran erinnert, dass der in Abschnitt 2.4.2 beschriebene CCT ein Grundbaustein für diese Bahntypen ist. Bei der Berechnung einer zu fahrenden Bahn wird so die Wahrscheinlichkeit erhöht, dass zumindest eine der zur Verfügung stehenden Bahntypen eine kollisionsfreie Überführung des Roboters gewährleistet. Ist dies nicht der Fall, müssen ebenso wie für den Umgang mit mobilen Hindernissen geeignete Heuristiken entwickelt werden, um *Rolland* in eine temporäre Konfiguration zu überführen aus der eine kollisionsfreie Bahnplanung möglich ist.

Um eine Bahn schließlich in Echtzeit auf ihre Hindernisfreiheit zu analysieren, könnte sich der Einsatz von hardwarebeschleunigten Grafikkarten als nützlich erweisen. Würden die Sensorikinformationen, im voraus vorhandene Umgebungskarten sowie eine Liste von den Rollstuhl auf seinen Bahnpositionen repräsentierenden Polygonen in einer für die Grafikhardware kompatiblen Datenstruktur vorliegen, könnte der Grafikprozessor die Schnittpunktberechnungen im Gegensatz zur Bearbeitung durch den Hauptprozessor drastisch beschleunigen.

6 Danksagung

Ich bedanke mich an dieser Stelle für für die freundliche Unterstützung bei all denen die für die kleinen und großen Probleme, die sich mir im Laufe dieser Arbeit stellten, ein offenes Ohr gefunden haben. Insbesondere sind hier Thomas Röfer und Axel Lankenau zu nennen, die mir aus dem Bereich der *kognitiven Robotik* hilfreiche Tipps geben konnten.

Nicht vergessen möchte ich auch meine Familie und meine Freunde, die mir im Laufe meines Studiums in gemeinsam verbrachter Zeit und durch ihre Unterstützung gezeigt haben, dass nicht nur das Informatikerdasein zu einem erfüllten Leben gehört.

7 Literatur

[1] M. Bäumker. Rechenverfahren der Ingenieurvermessung, die Klothoide. Vorlesungsskript der FH Bochum, FB Vermessungswesen und Geoinformatik.

[2] Christian Drücker, Robert Hormozi, and Paul Ziemann. Wegplanung. Projektbericht des studentischen Projektes Roses, 1998-2000.

[3] Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.

[4] J.-P.Laumond, S.Sekhavat, and F.Lamiraux. *Robot Motion Planning and Control*, volume 229 of *Lectures Notes in Control and Information Sciences*, chapter Guidelines in Nonholonomic Motion Planning for Mobile Robots, pages 1–53. Springer, 1998.

[5] Axel Lankenau and Oliver Meyer. Der autonome Rollstuhl als sicheres eingebettetes System. Master's thesis, Universität Bremen, Fachbereich 3, Studiengang Informatik, September 1997.

[6] Jean-Claude Latombe. *Robot Motion Planning*, chapter Configuration Space of a Rigid Object, pages 58–104. The Cluwer International Series in Engineering and Computer Science, Robotics: Vision, Manipulation and Sensors. Cluwer Academic Publishers, 1991.

[7] Jean-Claude Latombe. *Robot Motion Planning*, chapter Kinematic Constraints, pages 403–451. The Cluwer International Series in Engineering and Computer Science, Robotics: Vision, Manipulation and Sensors. Cluwer Academic Publishers, 1991.

[8] A.De Luca, G. Oriolo, and C.Samson. *Robot Motion Planning and Control*, volume 229 of *Lectures Notes in Control and Information Sciences*, chapter Feedback Control of a Nonholonomic Car-like Robot, pages 171–253. Springer, 1998.

[9] Robotics Group-Homepage of Service Robotics Lund University. Internet: http://www.robotics.lu.se/, 04 2002.

[10] J.A. Reeds and L.A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145:367–393, 1990.

[11] Thomas Röfer. Konsistente Karten aus Laserscans. Technical report, Bremer Institut für Sichere Systeme, TZI, FB3, Universität Bremen, 2001.

[12] Thomas Röfer. Rolland - der Bremer Autonome Rollstuhl. Internet: www://tzi.de/rolland/index_d.htm, 03 2002.

[13] Uwe Siems, Christoph Herwig, Thomas Röfer, and Jan Kuhlmann. SimRobot - 3D Robotiksimulator. Internet: www://informatik.uni-bremen.de/~ simrobot, 03 2002.

[14] Th.Fraichard, A.Scheuer, and R.Desvigne. From reeds and shepp's to continuous-curvature paths. In *Int. Conf. on Advanced Robotics*, pages 585–590, October 25-27 1999.

[15] Rolf Unbehauen. *Systemtheorie 2.* R. Oldenbourg Verlag, 1998.

[16] Th. Wittmann. *Insektennavigation: Modelle und Simulationen.* PhD thesis, Universität Bremen, 1996.